UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

GUSTAVO KIRA

DA RESOLUÇÃO DE EQUAÇÕES PARA A PRODUÇÃO DE TEXTOS: A ABSTRAÇÃO COMO MEDIAÇÃO ENTRE CONCRETUDES NO ENSINO E APRENDIZAGEM DE PROGRAMAÇÃO DE *SOFTWARE*

TESE

CURITIBA

2022

GUSTAVO KIRA

DA RESOLUÇÃO DE EQUAÇÕES PARA A PRODUÇÃO DE TEXTOS: A ABSTRAÇÃO COMO MEDIAÇÃO ENTRE CONCRETUDES NO ENSINO E APRENDIZAGEM DE PROGRAMAÇÃO DE *SOFTWARE*

ENGLISHFROM ABSTRACT CALCULATIONS TO CONCRETE UTTERANCES: ABSTRATION AS MEDIATION AMONG LEARNING AND TEACHING SOFTWARE PROGRAMMING

Tese apresentada como requisito parcial à obtenção do título de Doutor em Tecnologia e Sociedade, do Programa de Pós-Graduação em Tecnologia e Sociedade, da Universidade Tecnológica Federal do Paraná.

Orientação: Luiz Ernesto Merkle

CURITIBA

2022

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Campus Curitiba

GUSTAVO KIRA

DA RESOLUÇÃO DE EQUAÇÕES PARA A PRODUÇÃO DE TEXTOS: A ABSTRAÇÃO COMO MEDIAÇÃO ENTRE CONCRETUDES NOENSINO E APRENDIZAGEM DE PROGRAMAÇÃO DE SOFTWARE

Trabalho de pesquisa de doutorado apresentado como requisito para obtenção do título de Doutor Em Tecnologia E Sociedade da Universidade Tecnológica Federal do Paraná (UTFPR). Área de concentração: Tecnologia E Sociedade.

Data de aprovação: 28 de Março de 2022

Prof Luiz Ernesto Merkle, Doutorado - Universidade Tecnológica Federal do Paraná

Prof.a Flavia Mendes De Andrade E Peres, Doutorado - Universidade Federal Rural de Pernambuco (Ufrpe)

Prof Hugo Cristo Santanna, Doutorado - Universidade Federal do Espírito Santo (Ufes)

Prof Leonelo Dell Anhol Almeida, Doutorado - Universidade Tecnológica Federal do Paraná

Prof.a Nivea Rohling, Doutorado - Universidade Tecnológica Federal do Paraná

#EleNão

AGRADECIMENTOS

Agradeço a todos que depositaram em algum momento sua confiância em mim e também aqueles que tiveram paciência comigo nestes anos. Não cito nominalmente todas as pessoas que deveria, pois tenho pavor de deixar registrado para sempre que esqueci alguém. Obrigado.

O verdadeiro perigo não é que computadores começarão a pensar como pessoas, mas que pessoas começarão a pensar como computadores.

adaptado de Sydney J. Harris

RESUMO

KIRA, Gustavo. **Da resolução de equações para a produção de textos: As concretudes da abstração no ensino e aprendizagem de programação de *software***. 2022. 319 f. Tese (Doutorado em Tecnologia e Sociedade) – Universidade Tecnológica Federal do Paraná. Curitiba, 2022.

Esta tese tem como objetivo explorar os enunciados concretos, dentro de uma perspectiva bakhtiniana, como base de projetos e exercícios para o ensino e aprendizagem de programação de computadores. Para tanto, primeiro problematizamos o conceito de "pensamento computacional" e sua recepção dentro de um recorte específico da Informática na Educação através de um levantamento documental/bibliográfico de artigos publicados nos anos de 2015, 2016 e 2017 no WAlgProg, um *workshop* parte do Congresso Brasileiro de Informática na Educação (CBIE). Em um segundo movimento, caracterizamos o "pensamento computacional" como algo muito similar ao que é chamado pela Ciência da Computação de abstração. Ao identificar uma falta de especialidade na forma como este conceito é tratado, propomos visitá-lo a partir de autores mais próximos de uma perspectiva materialista, com o apoio, em especial, de autores como Valentin Voloshinov, Mikhail Bakhtin, Lev Vigotski, Henry Lefebvre e Alvaro Vieira Pinto, entre outros. Esta mudança de perspectiva permite entender a abstração como um processo de mediação entre duas concretudes e a liguagem como uma atividade concreta e não somente um sistema de signos abstraído. Para tentar dar conta de algumas questões que vem à tona com este entendimento da abstração e linguagem, propomos entender a atividade de escrita de um programa de computador como uma enunciação concreta, em especial, na maneira como Bakhtin o faz. Por fim, apresentamos uma análise documental com base em materias (e-mails e códigos fontes) gerados por três atividades didáticas as quais tentam evidenciar as diferenças de se trabalhar alinhado com perspectiva bakhtinana de escrita de texto.

Palavras-chave: linguagem e educação; abstração; dialogismo; Vigotski; Bakhtin.

ABSTRACT

KIRA, Gustavo. **From abstract calculations to concrete utterances: Abstration as mediation among learning and teaching software programming**. 2022. 319 p. Thesis (Doctor of Philosophy in Technology and Society Studies) – Universidade Tecnológica Federal do Paraná. Curitiba, 2022.

This thesis aims to explore a bakhtinian's perspective of concrete utterance as bases for exercises and projects to teaching and learning computer programing. First, we problematize "computational thinking" as a concept and its reception at an especific frame of Informatics at Education field using a documental/bibliographic survey with papers published between 2015 and 207 at WAlgProg, a workshop part of Congresso Brasileiro de Informática na Educação (CBIE). Second, we define "computational thinking" as something very simmilar to what Computer Science calls abstraction. As we found a lack of specification in the way this concept is deal with, we propose to understand it looking from a more materialist approach supported by authors like Valentin Voloshinov, Mikhail Bakhtin, Lev Vigotski, Henry Lefebvre, Alvaro Vieira Pinto and others. This perspective change enables to understand as a mediation process between two concrete stances and language as a concrete activity and not just an abstracted system of signs. Trying to acomodate some questions that emerges when we understand abstraction and language, we propose to understand computer program writing as a concrete utterance, specially, in Bakhtin's way. Finally, we present a documental survay based on material (e-mails and source codes) genereted by three learning activities which tries to show differences from working with a bakhtinian's approach to write a text.

Keywords: language and education; abstraction; dialogism; Vigotski; Bakhtin.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS, SIGLAS E ACRÔNIMOS

SIGLAS

ACM	*Association for Computing Machinery*
CBIE	Congresso Brasileiro de Informática na Educação
CTS	Ciência, Tecnologia e Sociedade
IDE	*Integrated Development Environment*
IME	Instituto de Matemática e Estatística
NCSES	*National Center for Science and Engineering Statistics*
WAlgProg	Workshop de Ensino em Pensamento Computacional, Algoritmos e Programação

SUMÁRIO

1 INTRODUÇÃO

Esta é uma tese sobre ensino e aprendizagem de fundamentos de programação. De início, a temática parece ser bem específica do nicho acadêmico da Ciência da Computação[1], mas nos últimos anos, com a difusão da informatização nas sociedades, em especial, com um aumento de práticas sociais que envolvem o uso de telefones celulares com sistemas operacionais (*smartphones*), no cotidiano de diversos locais do mundo[2] e os impactos econômicos[3] deste crescimento fizeram com que a discussão sobre o ensino e aprendizagem de programação não esteja mais restrita aos domínios, mesmo que nem sempre centrais, acadêmicos da Ciência da Computação.

Apenas a título de exemplo do quão longe as discussões sobre ensino e aprendizagem de programação chegaram, em dezembro de 2014, Barack Obama, o então presidente dos Estados Unidos, encontrou-se com estudantes em uma iniciativa chamada "A Hora do Código"[4] e, com a ajuda de uma estudante, escreve uma linha de código usando uma linguagem de programação chamada *javascript* (OBAMA WHITE HOUSE, THE, 2014).

Obama comenta sobre a importância de se aprender conceitos ligados à ciência da computação em outro vídeo feito para a mesma campanha do ano anterior: "Aprender estas habilidades (Ciência da Computação) não é só importante para o seu futuro, é importante para o futuro do nosso país[5]" (CODE.ORG, 2013).

"A Hora do Código" é uma iniciativa de uma organização sem fins lucrativos chamada Code.org [6]. Se olharmos para os parceiros da organização, encontramos diversas *Big Techs*, tais como Facebook, Amazon, Google e Microsoft, o que explica, em parte, o interesse do governo estadunidense no ensino de Ciência da Computação. É verdade que a máquina estatal dos Estados

[1] Ciência da Computação e Computação serão usados como sinônimos no decorrer do texto, salvo quando usados de forma mais específica. Ambos os termos devem ser entendidos como o campo de conhecimento que envolve pessoas, teorias e artefatos ligados à produção, consumo e estudo de programas, processos e artefatos computacionais.

[2] É importante reconhecer que esta presença é desigual, tanto na questão qualitativa quanto quantitativa. Por exemplo, em países como Noruega e Suécia, o acesso era superior a 90% em 2012 (UNION, 2012). No continente africano, apenas um a cada quatro usam internet (COMMISSION, 2019). Já no Brasil, ignorando suas diferenças regionais, por volta de 75% de brasileiras e brasileiros têm acesso à internet (TOKARNIA, 2020). Entretanto, este número deve ser visto com cuidado, pois considera que uma pessoas que acessou a internet uma vez nos últimos três meses como alguém que tem acesso. Para mais detalhes, ver capítulo 5 de Silva (2020)

[3] Apenas para ilustrar, Alphabet (controladora da Google), Microsoft, Amazon e Apple, cada uma, tem mais de um trilhão de dólares em valor de mercado. (G1, 2020).

[4] *Hour of Code.*

[5] *Learning this skills isn't just important for your future, it's important for our country's future.*

[6] https://code.org

Unidos sempre teve um interesse na Ciência da Computação, mas isto nunca significou um interesse direto maior pela dimensão que envolve seu ensino e aprendizagem.

Code.org faz parte de um conjunto de iniciativas[7] que envolvem, de alguma maneira, ensino e aprendizagem e tem como foco os problemas de diversidade e representatividade dentro da Ciência da Computação. No caso da Code.org, o foco é dirigido especialmente ao K-12[8], em que, para eles, é onde o problema da diversidade começa:

> Junto com as mulheres, aqueles que têm contato com o *Advanced Placement in Computer Science* no Ensino Médio têm a probabilidade dez vezes maior de cursar Ciência da Computação. Estudantes negros e hispânicos que têm contato com o *Advanced Placement in Computer Science* no Ensino Médio têm a probabilidade de sete a oito vezes de cursar Ciência da Computação[9] (CODE.ORG, 2018).

A falta de diversidade na computação não é um problema recente. Por exemplo, em 1998, Angwin (1998) comenta que das 33 grandes empresas do Vale do Silício, na média, 4% da força de trabalho eram negros e 7% latinos, enquanto estes grupos representam 8% e 14%, respectivamente, da força de trabalho total na localidade. Ainda assim, têm uma chance maior de ocupar cargos de chão de fábrica, serviços e suporte.

Uma das causas citadas por Angwin (1998) é o foco do Code.org: "Mais de dois terços dos estudantes negros e latinos frequentam o Ensino Fundamental e Médio em escolas predominantemente frequentadas por minorias – que tendem a ter menos dinheiro que as escolas majoritariamente frequentadas por brancos[10]".

Este problema não se limita à dimensão econômica, como mostra Margolis *et al.* (2010, loc.2496) ao estudar três diferentes escolas em Los Angeles, incluindo uma que não tinha necessariamente problemas com recursos financeiros e descobrindo que nelas existia "uma interação entre normas institucionais e sistemas impactando o porquê de tão poucos estudantes de Ensino Médio afro-americanos e latinos/as estarem aprendendo Ciência da Computação[11]".

O problema da representatividade não está delimitado somente à questão de raça e etnia. De forma quantitativa, uma pesquisa da *National Center for Science and Engineering*

---

7 Sobre estas iniciativas, algumas questões relacionadas ao contexto estadounidense são descritas em (ASPRAY, 2016). Mais a frente nesta introdução são descritas algumas outras.

8 K-12 é uma expressão que designa a parte do sistema de ensino estadunidense responsável pelo jardim de infância (*Kindergarten*) e os 12 anos seguintes. (OXFORD LEARNERS DICTIONARIES, 2021).

9 *Among women, those who try AP Computer Science in high school are 10 times more likely to major in computer science. Black and Hispanic students who try AP Computer Science in high school are 7-8 times more likely to major in computer science.*

10 *More than two-thirds of black and Latino students go to elementary and high schools with a predominantly minority population – which tend to have less money than majority white schools.*

11 *was an interaction between institutional norms and belief systems impacting why so few African American and Latino/a high school students are learning computer science.*

Statistics (NCSES) sobre o perfil de quem recebeu um diploma de bacharelado em Ciência da Informação e/ou Computação mostra que, nos Estados Unidos, desde 2008, existe um leve aumento da quantidade de diplomas entregues para mulheres no campo, mas ainda assim não passa de um quarto do total de diplomas entregues (NCSES, 2021). De forma qualitativa, o estudo mostrado em Vassallo *et al.* (2018) sobre a percepção de mulheres a respeito do seu ambiente de trabalho deixa claro que apenas "frequentar" o espaço não significa reconhecimento e igualdade de oportunidades. Das mais de 200 mulheres que respoderam a pesquisa, 60% reportaram ter recebido investidas sexuais não desejadas e, destas, 65% vindas de pessoas com cargos superiores.

Não só isso, mas o acesso aos dados sobre diversidade não é dos mais fáceis. Evans e Rangarajan (2017) apresentam alguns dados retirados dos relatórios EEO-1[12] disponibilizados publicamente por algumas empresas de tecnologia e colocam que nem todas as empresas disponibilizam estas informações ou as fazem com dados agregados que não têm muita serventia para uma análise.

Evans e Rangarajan (2017) mostram dados importantes, como por exemplo, a baixa proporção de mulheres em empresas como Google e Apple, ambas com 25% ou menos. Ebay tem menos de 1% de negros como parte de sua força de trabalho e tem a menor proporção com relação a trabalhadores latinos. Já empresas como Uber e Lyft contam com certa de 8,7% e 14,2% de profissionais colocados como negros, latinos, índios norte-americanos, provenientes de ilhas do pacífico e mestiços.

É interessante notar que o campo da computação nem sempre foi assim. Henn (2014) coloca que foi por volta de 1984 que a quantidade de mulheres na Ciência da Computação estagnou e caiu. Ele também coloca que os computadores pessoais surgiram neste momento e eram vendidos como brinquedos voltados somente para homens e garotos. Mesmo não sendo possível afirmar categoricamente que exista uma relação direta entre o tipo de *marketing* e a baixa presença de mulheres na computação, é importante localizar esta questão como parte de um padrão maior.

Por exemplo, a pesquisa feita por Margolis e Fisher (2002), durante os anos de 1995 até 1999, com estudantes da Universidade Carnegie Mellon, mostra alguns indícios destes processos que acabam por deixar de lado as mulheres no campo da computação. A pesquisa afirma que

[12] O EEO-1 é um relatório anual que obriga todas as empresas com 100 ou mais empregados a submeter dados sobre a composição de sua força de trabalho, incluíndo dados sobre raça/etnia, sexo e cargos. (U.S. Equal Employment Opportunity Commission, 2021).

este processo começa cedo com construções sociais atravessadas diretamente por questões de gênero ligadas ao desenvolvimento infantil que levam a computação ser considerada como algo "para meninos". Continua na adolecência, sendo reforçadas pela pressão de grupos sociais, jogos de computador e o áte papeis sociais atribuidos aos gêneros durante o Ensino Médio.

Um desdobramendo disto é o esquecimento das contribuições históricas feitas por mulheres dentro da computação. Por exemplo, Katherine Johnson e Annie Easley. Duas mulheres que trabalham na NASA e foram fundamentais para as viagens à lua. Um outro caso que também merece destaque é o das primeiras programadoras do ENIAC: Kathleen McNulty, Frances Bilas, Betty Jean Jennings, Elizabeth Snyder, Ruth Lichterman e Marlyn Wescoff, junto com Ruth Rauschenberger, Lila Todd, Homé McAllister, Marie Bierstein e Willa Wyatt Sigmund (FRITZ, 1996).

Para tentar reverter este estado, no começo dos anos 2000, a Universidade Carnegie Mellon promoveu mudanças em currículos e adaptações aos seus processos. Foram feitas mudanças desde a admissão de novos estudantes até mudanças de professores em pontos mais críticos do currículo em que mulheres relataram maior distresse (MARGOLIS; FISHER, 2002, p.129-139).

Já a Universidade da Califórnia mudou a maneira como é dada uma disciplina introdutória de Ciência da Computação. De "Introdução a programação simbólica" para "A beleza e alegria de computar", e em 2014, o curso teve mais mulheres do que homens inscritos (COMPUTERSCIENCE.ORG, 2018).

A Faculdade Harvey Mudd, que em 2017 tinha mulheres como pelo menos metade de seus formandos em Ciência da Computação, Engenharias e Física, também implementou uma estratégia parecida. Para o curso de Ciência da Computação, a disciplina introdutória foi redesenhada para lidar com perfis diferentes de estudantes: aqueles que nunca programaram na vida, aqueles que estavam começando e aqueles que não precisavam da disciplina introdutória (WEISUL, 2017).

Dada a forma como a computação se desenvolveu ao longo do tempo, este contexto estadunidense tem reflexos ou se repete no Brasil. Por exemplo, a primeira turma do Bacharelado em Ciências da Computação do Instituto de Matemática e Estatística (IME) tinha 20 estudantes, sendo 14 mulheres e 6 homens. A título de comparação, a turma de 2016 teve 41 estudantes, com apenas 6 mulheres (SANTOS, 2016).

Maia (2016) apresenta este contexto com mais profundidade ao mostrar que entre 2000 a 2013 o número de concluintes homens de cursos em Ciência da Computação no Brasil cresceu 98% enquanto o de mulheres decresceu 8%. Isto mostra um relativo descolamento quanto aos dados da NCSES (2021), em que o número de diplomas relacionados às áreas de computação entregues para mulheres nos Estados Unidos teve um leve aumento em um período parecido.

Quanto à questão do ambiente de trabalho, Maia (2016) coletou depoimentos de mulheres que trabalham na área da informática no Brasil, os quais corroboram com os dados de Vassallo *et al.* (2018) sobre o tratamento desigual dado às mulheres com relação àquele dado aos homens. Nos depoimentos, existem indícios de uma "desequilibrada carga de trabalho doméstico e de cuidado familiar que atravessa a divisão sexual do trabalho", "percepção estereotipada da mulher nos espaços de trabalho"e papéis das mulheres "demarcados por dispositivos sociais que distinguem a qualificação feminina da masculina"(MAIA, 2016, p.236-240).

Maia (2016) ainda relaciona o baixo número de mulheres em cargos de liderança na área de TI com a baixa presença deste grupo nos cursos de Ensino Superior ligados à área. Assim como no contexto estadunidense, os processos de exclusão das mulheres do campo da computação citados por Margolis e Fisher (2002) têm suas contrapartes no Brasil.

Por exemplo, Lima (2013) identifica que existe um tratamento diferente para as professoras ao entrevistar docentes de computação e informática. O tratamento não parte somente de colegas homens, mas também do corpo discente e outros trabalhadores de suas instituições. As professoras relatam que "precisam demonstrar que têm a mesma competência que os professores para serem tratadas sem discriminação"(LIMA, 2013, p.809) e "são testadas e colocadas à prova pelos professores quanto à sua competência para aprovação de trabalhos científicos"(LIMA, 2013, p.811).

Quanto às estudantes, Lima (2013) mostra que os professores até dizem acreditar que homens e mulheres têm a mesma capacidade, mas ao mesmo tempo comentam que "para as alunas, a aprendizagem em certas disciplinas é mais demorada que para os homens, exigindo maior auxílio deles"(LIMA, 2013, p.808).

As entrevistas feitas por Amaral *et al.* (2017) com cinco estudantes de um curso de Bacharelado em Sistemas de Informação de uma universidade pública reforçam as considerações feitas por Lima (2013). Em especial, nas respostas para a pergunta: "quando você enfrentou algum problema de discriminação relacionado a gênero no seu curso?"em que todas as entrevistadas respoderam de forma afirmativa à questão.

Os problemas relacionados à raça e etnia também parecem persistir em contexto brasileiro. No censo de 2010, a área de TI tinha uma composição majoritariamente formada por pessoas brancas (78,5%) seguida de pessoas negras (20,6%). Além de uma presença pequena se comparada com a composição brasileira, os profissionais negros têm uma média salarial inferior se comparada com a dos profissionais brancos (SOUZA; TOSTA, 2020).

Nunes (2016) complementa a questão ao mostrar que os negros tendem a ocupar postos que exigem qualificação mais baixa e têm uma participação pequena em funções gerenciais. Além disso, Nunes (2016) também destaca a existência de uma diferença salarial entre negros e brancos em posto de trabalho equivalentes que, segundo o autor, não podem ser explicadas por elementos observáveis, podendo ser creditada à discriminação.

Dado que o Brasil apresenta problemas similares aos identificados nos Estados Unidos quanto à baixa presença de minorias no campo, não é supresa que também tenha propostas de solução similares. Por exemplo, o *Rails Girls São Paulo* ou *Rails Girls Porto Alegre*, versões brasileiras do *Rails Girls*, evento criado em 2010 por Linda Liukas e Karri Saarinen com o objetivo de "abrir a área de tecnologia e fazê-la mais convidativa para garotas e mulheres[13]". A primeira versão do evento foi feita em 2010 na cidade de Helsinki e teve mais de 100 garotas inscritas (GIRLS, 2021).

A análise de dados do censo 2010 feita por Souza e Tosta (2020) apresenta as mulheres negras como o grupo com as médias mais baixas e menos representadas dentro da TI. Iniciativas como o *Preta lab*, que "trabalha para trazer diversidade para a tecnologia e inovação"(PRETALAB, 2021) ou empresas como a *Info Preta*, com o objetivo de "o inserir pessoas negras, LGBTQIAP+ e mulheres no mercado de tecnologia"(PRETA, 2021), atuam diretamente para mudar esta questão.

Além destas inciativas mais ligadas ao mercado de trabalho, também existem aquelas ligadas mais ao campo acadêmico, tais como as apresentadas em Amaral *et al.* (2015), em que a Interação Humano Computador é usada para introduzir conceitos ligados à àrea de computação para meninas no Ensino Médio. Na mesma linha, Berardi *et al.* (2019) têm como alvo o mesmo perfil, mas trabalham com conceitos ligados a banco de dados.

São nestas problemáticas de representatividade e diversidade que as iniciativas de levar o ensino de programação para todas e todos encontram suas justificativas. Também é por esta via que iniciativas “avançam” em direção ao Ensino Médio e Básico, como é o caso de Amaral

[13] *Open up technology and make it more approachable for girls and women.*

et al. (2015) e Berardi *et al.* (2019). Entretanto, mesmo com este esforço, as dificuldades no ensino e aprendizagem de linguagens de programação parecem ainda continuar, como identifica Aureliano, Tedesco e Giraffa (2016).

Trabalhos como Pimentel *et al.* (2003), Gomes, Henriques e Mendes (2008) e Gomes e Mendes (2014) mostram que existe uma aceitação tácita sobre a dificuldade de estudantes nas disciplinas introdutórias de programação em dominar conceitos abstratos de programação. Inclusive ela é usada como base para justificar iniciativas ligadas ao ensino e aprendizagem na área. Por exemplo, Raiol *et al.* (2015) propõe e relata uma experiência com o uso da linguagem. Silva *et al.* (2016) apresenta um jogo sério para o ensino de Programação Orientada a Objetos e Raeder *et al.* (2016) integra disciplinas consideradas de início de curso (Programação, Lógica e Matemática para Computação).

A abstração é um conceito central para a computação (BUCCI; LONG; WEIDE, 2001; SPRAGUE; SCHAHCZENSKI, 2002; KRAMER, 2007; ZEHETMEIER *et al.*, 2019) e, por consequência, para o seu ensino e aprendizagem. Mesmo assim, é difícil encontrar uma reflexão um pouco mais profunda sobre o termo. Nos textos que discutem o contexto educacional da computação, é comum a referência a Wing (2006), sobre pensamento computacional como uma espécie de sinônimo, mas ainda assim, pouca reflexão sobre o que é a abstração e como ela é feita.

Para Milne e Rowe (2002, p.63), a dificuldade sobre alguns conceitos de programação pode vir da falta de conhecimento de como a memória principal do computador funciona. Já Craig e Petersen (2016) e Lippert (2018) têm conclusões parecidas, mas identifica a falta de conhecimento sobre a arquitetura dos computadores como fonte. Além disso, ele restringem suas observações a um conceito específico de programação, o de ponteiro.

Relacionar conceitos de programação, como ponteiros, com a arquitetura de computadores mostra que certos conceitos que são considerados abstratos têm uma certa dependência de outros conceitos materializados nos artefatos da própria computação. Algo reconhecido por Tedre e Denning (2016) ao defender a importância do entendimento dos modelos computacionais para os quais algoritmos são planejados. Entretanto, a Ciência da Computação, como comenta Margolis, Goode e Binning (2018), "é um campo que em sua maioria tem sido ensinado através de aulas expositivas e instruções diretas[14]". Ainda podemos reforçar que uma grande parte

[14] *A field that has been largely taught through lecture and direct instruction.*

da disciplina entende que, dependendo do nível de abstração trabalhado, estes conceitos com conexões materiais muito fortes não são requisitos.

Paul (2015), em sua coluna no *The New York Times*, levanta a possibilidade de que o formato de aula expositiva universitária possa ter um papel discriminatório, pois é "uma forma cultural específica que favorece algumas pessoas brancas enquanto discrimina outros, incluindo mulheres, minorias, pessoas de baixa-renda e primeiras gerações a entrarem no Ensino Superior[15]". Entre possíveis razões para isto, Paul (2015) coloca que o formato expositivo, "quando usado sozinho, sem outros suportes instrucionais oferece vantagens desleais para uma já privilegiada população[16].".

Uma das possíveis razões colocadas por Paul (2015) seria a de que "estudantes de baixa renda e minorias têm uma chance desproporcional de ter frequentado uma escola de baixo rendimento e perdido um enriquecimento acadêmico e extracurricular familiar para seus mais abastados colegas brancos, chegando à universidade com menos bagagem de conhecimento[17]" o que seria um problema, já que "estudos têm demonstrado que nós aprendemos novos conteúdos através de relações com o conhecimento que já temos[18]".

A análise de Bourdieu e Catani (1999, p.49-50), sobre o sistema de ensino regular francês, tem conclusões que reforçam a questão colocada por Paul (2015) ao afirmar que:

> [. . .] Elas (As crianças oriundas dos meios mais favorecidos) herdam também saberes (e um "savoir-faire"), gostos e um "bom-gosto", cuja rentabilidade escolar é tanto maior quanto mais frequentemente esses imponderáveis da atitude são atribuidos ao dom. A cultura "livre", condição implícita do êxito em certas carreiras escolares, é muito desigualmente repartida entre os estudantes universitários originários das diferentes classes sociais [. . .] (BOURDIEU; CATANI, 1999, p.49-50)

Ou seja, Bourdieu e Catani (1999, p.49-50) chama a atenção para como as experiências prévias dos e das estudantes mais favorecidos são reconfiguradas como dom ou aptidão natural e escondem um processo de estratificação social. Freire (2014, p.79-80) oferece ao mesmo tempo uma explicação e crítica daquilo colocado por Paul (2015) e Bourdieu e Catani (1999, p.49-50) e dá destaque para o papel do educador neste processo:

> Falar da realidade como algo parado, estático, compartimentado e bem comportado, quando não falar ou dissertar sobre algo completamente alheio à experiência existencial dos educan-

[15] *A specific cultural form that favors some people while discriminating against others, including women, minorities and low-income and first-generation college students.*

[16] *When used on its own without other instructional supports - that offers unfair advantages to an already privileged population* (PAUL, 2015)

[17] *Poor and minority students are disproportionately likely to have attended low-performing schools and to have missed out on the rich academic and extracurricular offerings familiar to their wealthier white classmates, thus arriving on campus with less background knowledge.*

[18] *Research has demonstrated that we learn new material by anchoring it to knowledge we already possess.*

> dos, vem sendo, realmente, a suprema inquietação desta educação. A sua irrefreada ânsia. Nela, o educador aparece como o seu indiscutível agente, como seu real sujeito, cuja tarefa indeclinável é "encher" os educandos dos conteúdos de sua narração. Conteúdos que são retalhos da realidade desconectados da totalidade que se engendram e em cuja visão ganhariam significação (FREIRE, 2014, p.79-80).

A educação ao qual Freire (2014) se refere no trecho é aquela que chama de bancária, em que "a única margem de ação que se oferece aos educandos é a de receber depósitos, guardá-los e arquivá-los" (FREIRE, 2014, p.80-81) e nela, a educação "se torna um ato de depositar, em que os educandos são os depositários e o educador, o depositante" (FREIRE, 2014, p.80). Neste modelo, aqueles mais favorecidos tem maior pontencial de converter suas experiências prévias em "depósitos" iniciais, pois tiveram muito mais chances de ter contato com conhecimentos, valores e experiências alinhadas com o tipo de conhecimento acadêmico/escolar.

Com isso, a passagem de Freire (2014) consegue mostrar um problema que as iniciativas do tipo Code.org não necessariamente reconhecem ou deixam de lado propositalmente: o quanto a própria forma do conhecimento da Ciência da Computação pode ter um papel nesta exclusão das minorias. O tipo de abstração privilegiada pela computação é resultado de um processo histórico com raízes em um desenvolvimento matemático específico e cuja inclusão não necessariamente é feita em diálogo com as formas de conhecimento diferentes.

Freire (2014) nos permite perguntar se o caminho de "levar a computação para o outro", através destas inserções anteriores ao Ensino Superior e disponibilizar materiais de ensino seriam o suficiente para que a computação seja um campo mais inclusivo. A continuidade dos problemas de representação na indústria e a permanência das dificuldades relatadas no começo dos cursos de graduação mostram que existe, pelo menos, um espaço para este tipo de reflexão. E é nesta lacuna que este trabalho se posiciona.

Antes de dar continuidade, é importante ressaltar que não temos a pretensão de esgotar o tema, principalmente quanto às questões sociais, históricas, culturais e econômicas ligadas ao ensino e à aprendizagem de programação.

1.1 MOTIVAÇÃO DA PESQUISA

Na minha curta[19] carreira como docente, tive a oportunidade de trabalhar em três instituições públicas na condição de professor temporário/substituto. A primeira como docente substituto em um departamento de Desenho Industrial, a segunda como professor temporário

[19] Formalmente, ao longo de 10 anos, trabalhei 4 anos e meio como professor substituto e 1 ano como bolsista ligado a iniciativas relacionadas com aprendizagem.

EBTT[20] no Ensino Médio da Rede Federal e por último como docente temporário em um departamento de Ciência da Computação de uma universidade estadual.

A segunda experiência foi especificamente em um curso Técnico Integrado de Informática no ano de 2018. Este contexto me fez pensar sobre algumas questões ligadas ao ensino e aprendizagem de Computação e/ou Informática, sobretudo quando essa é direcionada para estudantes jovens e com um perfil muito plural.

Primeiro, tenho a percepção de que as discussões sobre que conteúdos e/ou quais habilidade/competências deveriam ser ensinados no Ensino Médio e no Ensino Superior sobre Informática e Ciência da Computação ainda não estão bem consolidadas. Existem diversas iniciativas, tais como um grupo de estudo na ACM (http://ccecc.acm.org/), diversas publicações em congressos como o Simpósio Brasileiro de Informática na Educação, mas, mesmo assim, parece existir uma distância entre estas iniciativas e as salas de aula.

Se esta questão não fosse um problema grande o suficiente, ainda temos a visão de que um estudante do curso técnico deveria ir direto para o mercado de trabalho. Acredito que estas duas visões têm um papel relevante na forma como um curso de informática de nível médio técnico é construído. No caso desta instituição, em especial, as disciplinas ligadas à Informática/Computação eram praticamente uma transposição das disciplinas de um curso superior equivalente.

Se existe uma grande quantidade de literatura relatando as dificuldades em lidar com os conceitos e práticas que estudantes têm ao entrar em cursos de Ensino Superior ligados ao campo da Ciência da Computação, é de se pensar em como lidar com esta mesma questão em um curso ofertado para um público mais jovem e com menos bagagem escolar como é o caso daqueles que frequentam o Ensino Médio.

Vários dos e das estudantes que faziam o curso Técnico Integrado em Informática não necessariamente tinham a intenção de seguir na área. Estavam ali "apesar" da informática, pois existia a percepção por parte das e dos estudantes e também de seus responsáveis de que a instituição era uma boa escola com professores qualificados. É importante colocar que em um curso técnico integrado às disciplinas do eixo básico e do eixo técnico são ofertadas de forma conjunta. Ou seja, neste curso, ter aulas de Português, Sociologia e Geografia, significava também ter aulas de Banco de Dados e Arquitetura de Computadores.

[20] Ensino Básico Técnico e Tecnológico.

Além deste interesse deslocado, a assimetria quanto aos conhecimentos ligados à informática também era uma questão representativa. Desde estudantes sem acesso a computadores em casa até estudantes com níveis de conhecimento muito superiores aos que se esperaria de alguém no Ensino Médio.

Minha terceira e última experiência como professor temporário foi no ano seguinte, desta vez no Ensino Superior como professor no curso de Tecnologia em Análise e Desenvolvimento de Sistemas em uma instituição de ensino pública estadual. Em um outro estado, em uma outra instituição e em um outro nível educacional, o problema da assimetria de conhecimento ainda existia. Ao conversar com as e os estudantes, era interessante notar que as turmas eram formadas tanto por pessoas que frequentaram o Ensino Médio Técnico[21], ou seja, tiveram um treinamento formal em informática, quanto por pessoas que nunca escreveram antes uma linha de código. Além disso, inclusive, diversos estudantes já tinham empregos ou estágios na área de Computação/Informática. Uma questão complexa que surge quanto a esta assimetria é a própria grade curricular[22] dos cursos de Informática/Computação. Se, no caso do Ensino Médio, a transposição do curso de Ensino Superior para o Médio sem nenhum tipo de reflexão eram problemático, no caso do Ensino Superior a rigidez se mostra um problema. Com um grupo de pessoas com trajetórias distintas, como esperar que uma grade fixa dê conta de uma prática complexa?

Na minha vida acadêmica, passei muito mais tempo na condição de discente do que de docente. Independente do papel, não são poucas as estórias que escutei sobre pessoas fazendo a mesma disciplina cinco ou seis vezes, em que a disciplina e o/a docente é vista como uma barreira pelos discentes e como um filtro para alguns docentes.

Este pode ser um exemplo limítrofe, mas mostra como pode funcionar o papel do docente quando se ignora por completo o corpo de estudantes no processo de ensino e aprendizagem, dando protagonismo para o conteúdo ou para uma coleção de conhecimentos que um estudante imaginário perfeito deveria saber para ser aprovado na matéria. Entretanto, mesmo ao levar em conta o grupo de estudantes, é preciso lembrar que não é incomum no campo da Ciência da Computação ter pessoas com níveis muito diferentes de familiaridade com o tema de uma disciplina. Se nivelar a disciplina por aqueles que já têm um conhecimento na área, não

[21] Um pequeno indício de que a ida direto ao mercado de trabalho não necessariamente ocorre ao final de um curso de Ensino Médio técnico.

[22] Mesmo ciente de que a grade de disciplinas é um instrumento diferente de um currículo, o uso da palavra grade parece apropriada aqui. Pois, além de representar a organização do curso, ela também impõe, prende e limita diversos movimentos e possibilidades pedagógicas.

parece justo, pois ter conhecimento em informática não é pré-requisito para o curso, ao mesmo tempo, adotar que nenhum estudante tem nenhum conhecimento é ignorar o contexto real.

Este é um pequeno resumo da problemática que levaram a, pelo menos, três questionamentos iniciais quanto aos processos de ensino e aprendizagem de conceitos ligados à programação de computadores: 1) Como lidar com os diferentes perfis de estudantes no ensino de programação dentro de uma estrutura como uma disciplina? 2) Como lidar com a assimetria de conhecimentos sobre Computação? 3) É possível pensar em uma forma de ensino de programação que dê conta destas duas questões?

A primeira questão permite discutir algumas diferenças e semelhanças desta investigação com as iniciativas ligadas à representatividade, tais como as citadas na seção anterior. Talvez a principal diferença seja a compulsoriedade que a Computação tem nos contextos deste trabalho, algo que não necessariamente é pré-requisito para as atividades com foco em representatividade.

Diversas iniciativas que trabalham com representatividade apresentam a Computação de uma forma menos restritiva e usam desta via para tentar reverter as assimetrias quanto à raça/etnia e/ou gênero. Elas funcionam como "convites" para o ingresso no campo, enquanto esta tese tenta refletir o que fazer depois que o convite foi aceito e se o problema era só a falta de convites. É importante resaltar que a permanência nos círculos ligados à Ciência da computação também faz parte das preocupações dos trabalhos ligados à representatividade, mas não necessariamente são o foco das incursões ao Ensino Médio, já que a quantidade de cursos de Ensino Médio regulares é muito maior que a quantidade de cursos técnicos de nível médio.

Com isso, podemos citar que as preocupações que levaram as mudanças citadas por Margolis e Fisher (2002) no curso da *Carnegie Mellon* é um dos pontos de semelhança entre este trabalho e outros ligados à representatividade, entretanto, o foco aqui também lança um olhar para a dimensão conceitual da própria Computação, em uma forma de autocrítica.

É importante deixar claro que as considerações deste trabalho devem ser vista de forma complementar às dicussões sobre raça/etnia e gênero na Computação. Acredito que estou olhando para o mesmo problema, apenas de um ângulo e posição diferente[23].

[23] Em especial quanto à raça e etnia, apesar de não aparentar, sou um brasileiro que têm pelo menos duas gerações de antepassados nascidos no Brasil. Foram meus avós por parte de mãe e meus bisavós por parte de pai que imigraram do Japão para o Brasil. Este fato é relevante, principalmente ao lembrar que seja por descendência ou por fenótipo, também faço parte de uma minoria. Entretanto, "asiáticos" ou "amarelos", categoria que englobam as etnias mais ao oeste (em uma perspectiva eurocêntrica a qual não posso ignorar), tem sua identidade articulada um pouco diferente de outras, pois é usada como um modelo de integração. Este esteriótipo de "minoria-modelo" é muito usado como contra-argumento em relação à necessidade das iniciativas afirmativas ligadas à raça e etnia. Entendo que ter uma identidade hifenizada (nipo-brasileira) junto com outras clivagens (homem, classe média, com acesso a educação e saúde etc) aumentam minha distância quanto a algumas possibilidades de atuação

De forma geral, esta tese é uma tentativa de organizar *a posteori* a busca por respostas e encaminhamentos destas questões. As descrições e considerações sobre as atividades relatadas foram tentativas de oferecer algumas respostas, mesmo que parciais, as duas últimas questões. A base teórica usada neste processo tem uma perspectiva materialista (em oposição a um idealismo, psiquismo e teoricismo) e é resultado de um caminho que começou por volta de 2010 na minha primeira experiência como professor no departamento de Desenho Industrial da UTFPR, quando tive a oportunidade de participar do grupo de estudos chamado Design & Cultura.

Através deste grupo, tive contato com autores ligados aos Estudos Culturais, tais como Stuart Hall (1932-2014) e Néstor García Canclini (1939-). Em um segundo momento, quando os interesses do grupo se voltaram mais especificamente para a cultura material, leituras de autores como Daniel Miller (1954-) sobre o papel de artefatos na constituição da cultura e considerações sobre o que seria uma antropologia digital foram um primeiro passo em direção a uma perspectiva materialista.

Alguns anos mais tarde, já no mestrado no PPGTE (mesmo programa dentro do qual esta tese foi escrita) e ainda fazendo parte do grupo de estudos, prossegui a leitura de autores ligados aos Estudos Culturais, como Jesús Martín-Barbero (1937-2021) e Raymond Williams (1921-1988). Além disso, tive contato com obras de autores ligados à Filosofia da Tecnologia, tais como Alvaro Vieira Pinto (1909-1987), Andrew Feenberg (1943-) e Herbert Marcuse (1898-1979)[24].

Podemos, com certo cuidado, dizer que boa parte[25] dos autores citados até aqui tem algum tipo de relação com o Marxismo com diferentes intensidades, vertentes e posições. Esta posição no plano filosófico colocam estes autores muito mais próximos de uma perspectiva materialista do que idealista. Olhar com esta "bagagem" nas costas as questões de ensino e aprendizagem em programação de computadores foi o que me levou a alguns autores usados nesta tese, tais como Louis Althussser (1918-1990), Lev Vigotski (1896-1934), Henri Lefebvre

dentro de iniciativas ligadas à representatividade, em especial, em relação ao protagonismo (algo exigido, em certo grau por uma tese e o trabalho como professor). Entretanto, esta posição no mundo ao mesmo tempo abre certos espaços para inciativas que talvez outras minorias não tenham. Se as iniciativas que buscam uma participação efetiva de grupos normalmente excluídos tensionam as barreiras do campo vindos da borda para o centro, gosto de pensar que alguém na minha posição pode atuar também na borda, mas tensionando pelo outro lado da barreira. Talvez esta seja a melhor explicação que eu possa dar do porquê esta tese não tenta avançar no mesmo caminho das iniciativas citadas na introdução, por mais que acredite que seja um movimento necessário e que o plano de fundo seja o mesmo.

[24] Gaston Bachelard (1884-1962) também entra como alguém que escreveu sobre filosofia da ciência e é relativamente importante para esta tese. Mas minhas leituras sobre este autor não são diretamente relacionadas com o programa de mestrado.

[25] Com exceção de Daniel Miller, que em uma de suas obras inclusive trabalha com a visão de Hegel sobre dialética ao invés da de Marx.

(1901-1991) e aos autores ligados ao Círculo de Bakhtin: Mikhail Bakhtin (1895-1975) e Valentin Voloshinov (1895-1936).

1.2 OBJETIVOS

1.2.1 Objetivo Geral

Investigar a construção de abstrações sob uma perspectiva da filosófia materialista/concreta e do dialogismo em atividades e processos de ensino-aprendizagem de programação de computadores.

Depois de ler a tese completa, acho que o objetivo geral seria algo como [sic] "investigar a construção de abstrações, consideradas de modo concreto e dialógico na perspectiva filosófica materialista, em atividades e processos de ensino-aprendizagem de programação de computadores"

1.2.2 Objetivos Específicos

- Evidenciar a existência de uma lacuna e falta de profundidade na discussão sobre os conceitos de pensamento computacional e abstração, ambos importantes para o ensino e aprendizagem de programação de computadores.
- Analisar as implicações teóricas para o ensino e aprendizagem de programação de computadores em possíveis mudanças que o conceito de abstração teria se caracterizado sobre uma base materialista.
- Explorar algumas alternativas desenvolvidas em sala de como as questões teóricas da base materialista podem ser articuladas em atividades didáticas ligadas ao ensino e aprendizagem de programação de computadores através de uma análise documental de e-mails e códigos trocados entre professor e grupos de estudantes junto com seus contextos e atividades.

1.3 MATERIAIS E MÉTODOS

A primeira etapa desta pesquisa, a ruptura, consiste em uma levantamento documental/-bibliográfico com o uso de fontes secundárias. Os documentos são os artigos publicados nos anos

de 2015, 2016 e 2017 no WAlgProg, *workshop* parte do Congresso Brasileiro de Informática na Educação (CBIE). Estes textos estão disponíveis dentro do Anais dos Workshops do Congresso Brasileiro de Informática na Educação em seus respectivos anos. A Tabela 1 sumariza o passos para a construção do *corpus* de análise:

Tabela 1 – Passos para a construção do corpus analisado na primeira etapa de pesquisa

	descrição do passo
1	Download dos textos do WAlgProg (2015, 2016, 2017) em formato pdf
2	Transformação dos textos do formato de arquivo pdf para txt
3	Limpeza manual dos textos em txt
4	Identificação dos parágrafos dos textos que tem ocorrências do termo "pensamento computacional"
5	Particionamento dos textos em grupos de acordo com a quantidade de ocorrências do termo "pensamento computacional"

Fonte: Autoria própria

1) Foi feito o *download* da versão em formato pdf de todos os artigos publicados nos anos de 2015, 2016 e 2017 do WAlgProg. 2) Todos os arquivos foram processados e transformados em arquivos de texto plano com o auxílio de um programa do tipo motor de buscas[26]. 3) Todos os arquivos foram "limpos" manualmente para remover as sobras do processo de extração (espaços duplos, *tags* típicas do formato pdf, texto de rodapés etc). Nesta etapa também foi feita a leitura inicial dos textos. 4) Com uma IDE, foram identificados todos os parágrafos que continham o termo "pensamento" (sem o computacional) e depois, dentro daqueles, foram selecionados os que continham o termo completo ("pensamento computacional"). Este processo foi feito em duas fases para revisar qualquer problema com relação às variações do termo resultantes do processo de extração que não tenham sido normalizadas na limpeza como, por exemplo, pensamento computacional com dois espaços entre as palavras ou pensamento em uma linha e computacional em outra. 5) O uso da IDE também possibilitou separar os textos de acordo com a quantidade de vezes que usavam o termo "pensamento computacional", ação importante para verificar a profundidade com que os textos trabalhavam com o termo.

Na segunda etapa, na construção do objeto de pesquisa, é feita uma revisão bibliográfica com o objetivo de entender por que vias a Ciência da Computação entende o processo de abstração. E, a partir disto, esta etapa também tem como objetivo mostrar o potencial que o uso de uma outra base teórica, no caso materialista, tem para tentar dar conta de problemas importantes quanto ao ensino e aprendizagem de fundamentos de computação. A Tabela 2 sumariza os principais aportes teóricos usados, assim como os conceitos articulados:

[26] Foi usado uma instância local de Elasticsearch que por sua vez usa o Apache Lucene.

Tabela 2 – Referências teóricas e suas contribuições para a segunda etapa

autor	área	conceitos
Meksenas (1992), Machado (2011)	Educação	Entendimento da abstração como medição entre concretudes.
Vygotsky (1995)	Educação	Zona de desenvolvimento proximal e instrução como um processo mediado composto por signos, instrumentos e pessoas.
Althusser (2019)	Filosofia	Visão de abstração dentro de uma base materialista. Entendimento da abstração não como um processo subtrativo, mas aditivo.
Lefebvre (1991)	Filosofia	Limitações da lógica formal e sua relação com a lógica dialética.
Vieira Pinto (2005a)	CTS	Crítica ao processo descontextualizado de produção acadêmica/tecnológica.
Franchi (2008)	Letras	Relação entre estudos sobre gramática e suas relações com a lógica formal e lógica dialética. Ponte para uma discussão entre escrita de códigos-fonte e escrita de textos.
Kuhn (2007)	Filosofia da Ciência	Noção de paradigma "inicial" para um contraste com a forma que o conceito é usado para classificar "formas" de programar ou tipos de programação.
Bakhtin (2014), Voló-chinov (2017 [1929]), Volochinov (2019 [1926])	Filosofia da Linguagem	Visão do ensino e aprendizagem como um processo de interação dialógico e marcado necessariamente pela presença do outro. Base para análise das trocas de e-mails da última etapa.

Fonte: Autoria própria

A terceira etapa é composta de duas partes: o experimento didático e a análise documental. Vamos chamar de experimentos didáticos o enquadramento das atividades desenvolvidas em duas instituições de ensino durante os anos de 2017, 2018 e primeiro semestre de 2019 dentro desta pesquisa. Nenhuma destas atividades foi planejada levando em conta a escrita desta tese. Elas foram planejadas de acordo com percepções e motivações tácitas oriundas, principalmente, da vivência do docente dentro de instituições de ensino.

No ano de 2017, elas foram desenvolvidas com turmas de primeiro, segundo e terceiro anos de um Curso Técnico em Informática Integrado ao Ensino Médio da Rede Federal de ensino, localizada em uma cidade da região metropolitana de Curitiba (PR). Em 2018 e no primeiro semestre de 2019, as atividades foram feitas em disciplinas do curso de Tecnologia em Análise e Desenvolvimento de Sistemas de uma universidade estadual catarinense. Mais detalhes destes contextos são apresentados no Capítulo 4.

As atividades no Ensino Médio foram feitas dentro das disciplinas de Banco de Dados, Programação Orientada a Objetos 1, Programação Orientada a Objetos 2, Arquitetura de Computadores e Sistemas Operacionais. Já as no Ensino Superior, foram feitas dentro das disciplinas de Programação Orientada a Objetos e Estrutura de dados.

A segunda parte desta etapa é apresentada diretamente neste texto através de uma análise documental de e-mails e códigos-fonte produzidos em algumas destas atividades didáticas. Estes dois tipos de documentos são uma parte do processo de ensino e aprendizagem e não os representam em sua totalidade. Entretanto, ambos permitem ilustrar as questões teóricas levantadas nas etapas anteriores. Assim, esta parte do processo é feita através de uma pesquisa documental, com base em dois tipos de documentos escritos sobre um suporte digital e interligados, como será melhor apresentado no Capítulo 4. Estes documentos fazem parte do arquivo particular do docente. O *corpus* de e-mails foi montado conforme os passos da Tabela 3:

Tabela 3 – Passos para a construção do corpus analisado na terceira etapa de pesquisa

	Descrição do passo
1	Seleção de todos os e-mails trocados com discentes em disciplinas.
2	Exclusão de e-mails sobre outros temas, tais como notas e avisos.
3	Agrupamento por "tripas" de e-mails.
4	Seleção de casos representativos e de atividades para um aprofundamento descritivo.
5	Levantamento dos códigos-fonte e trechos de código pertinentes para o aprofundamento descritivo.

Fonte: Autoria própria

1) Foram selecionados todos os e-mails trocados entre docente e os dicentes de todas as disciplinas ministradas (inclusive as não descritas anteriormente, tais como Informática na Educação e Introdução a Ciência da Computação). 2) Daqueles foram selecionados somente os e-mails que tinham como conteúdo alguma questão sobre as atividades didáticas, excluindo e-mails trocados sobre notas ou conceitos e outras questões burocráticas, parte de um componente curricular. 3) Os e-mails são agrupados pelo serviço de e-mail em "tripas"[27], que contém réplicas e tréplicas ao primeiro e-mail enviado. Estes agrupamentos têm tamanho variado, indo desde e-mails sem resposta (muitas vezes a conversa era feita em sala de aula) até tripas com mais de dez e-mails. Ao todo foram selecionadas 1.680 tripas de e-mail. 4) Análise das conversas por e-mails permite a seleção de casos representativos e atividades para uma descrição mais aprofundada. 5) Identificação dos códigos-fonte e trechos de código pertinentes para o entendimento nas trocas de e-mails.

Em síntese, a ruptura usa de métodos da pesquisa documental a fim de qualificar como o conceito de "pensamento computacional" é articulado em um recorte específico. O arquivo consultado é um arquivo do tipo público (anais do congresso disponíveis no formato digital) e os documentos em formato de texto são os artigos publicados no *workshop*. A construção do objeto de pesquisa traz um embasamento teórico para o conceito de abstração com o objetivo de dar

[27] *Threads.*

fundamento para uma abordagem que permite um olhar para o referido processo como algo que necessariamente envolve tanto quem está a abstrair quanto a quem se destina tal abstração. Por fim, a terceira etapa usa novamente a pesquisa documental, em que foi consultado um arquivo particular (serviço de e-mails) que possui dois tipos de documentos a serem analisados: os próprios e-mails e códigos-fonte de programas ou trechos de código.

1.4 DA FORMA DE PESQUISA ESCOLHIDA

1.4.1 Do carácter interdisciplinar da tese

Este texto foi produzido dentro de um programa de pós-graduação interdisciplinar (PPGTE/UTFPR) em uma área de concentração denominada Tecnologia e Sociedade. Estar sob o guarda-chuva teórico da Ciência, Tecnologia e Sociedade (CTS) significa entender uma relação de constituição mútua entre sociedade, ciência e tecnologia. Ou, como coloca o próprio programa: "a sociedade modela a ciência e a tecnologia e essas, por sua vez, modelam a sociedade e o ambiente" (PPGTE, 2018b).

De forma mais específica, esta tese foi pensada dentro da linha de pesquisa Mediações e Culturas, que tem como um de seus pressupostos "fomentar a compreensão crítica e cidadã de tecnologia" (PPGTE, 2018a). Por tecnologia, a linha entende-a como um conceito não aplicável somente a artefatos e instrumentos, mas parte de uma formação coletiva do ser humano dentro de relações sociais, fazendo das tecnologias "mediações sociais (materiais ou simbólicas), situadas e circunstanciadas axiológica, cultural e historicamente" (PPGTE, 2018a).

Este plano de fundo é importante para o presente trabalho como um todo e permeia, muitas vezes, de forma implícita, cada capítulo, com destaque para o entendimento do tema e, por consequência, do objeto de estudo como situados e circunstanciados axiológico e historicamente.

Sem, por hora, adentrar muito a discussão sobre as diferenças nos usos dos termos interdisciplinaridade, multidisciplinaridade, transdisciplinaridade, ciências auxiliares ou ciências vizinhas[28], de forma inicial, propomos apenas trabalhar com a classificação das áreas de conhecimento da fundação CAPES a fim de mostrar a existência de um carácter "interdisciplinar" na tese. Mesmo que a atribuição deste caráter seja dependente da validade da classificação da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), algo que não está sendo discutido, acreditamos que ela sirva, ao menos, para estabelecer um limiar mínimo quanto

[28] Termos retirados da introdução de Brun (2017).

à filiação disciplinar variada deste trabalho. Assim, nossa temática aproxima duas grandes áreas presentes na Tabela de Áreas de Conhecimento/ Avaliação da CAPES: a grande área das Ciências Humanas (70000000) e a grande área da Ciências Exatas e da Terra (10000003).

Além da temática, existe uma outra aproximação "interdisciplinar" feita pelo eixo teórico-metodológico, ou mais precisamente, de sua construção, essa sim basilar para o trabalho descrito aqui.

De um ponto de vista epistemológico amplo, esta tese está filiada, por um lado, à filosofia da linguagem do Círculo de Bakhtin e, por um outro lado, a uma parte específica da filosofia da ciência de Gaston Bachelard, fonte inicial de algumas considerações sobre o método sociológico encontrado nos trabalhos de Pierre Bourdieu. Reforçamos a localidade da contribuições de Bachelard, pois os seus conceitos de ruptura e construção do objeto científico são base para a duas primeiras partes da tese, entretanto, outras de suas colocações sobre ciência não necessariamente são compatíveis com o entendimento sobre ciência, tecnologia e sociedade apresentados anteriormente.

Silva, Silva e JUNCKES (2009, p.27) identificam três fases pelas quais o processo de pesquisa científica proposto por Bachelard/Bourdieu deve passar: a ruptura, a construção e a verificação.

A ruptura, de acordo com Bourdieu, Chamboredon e Passeron (2015), é uma descontinuidade com a opinião comum feita pela forma de conhecer da sociologia, uma vez que esta "só pode se constituir como ciência realmente separada do senso comum, com a condição de opor às pretensões sistemáticas da sociologia espontânea [opinião comum]" (BOURDIEU; CHAMBOREDON; PASSERON, 2015, p.25).

Esta perspectiva que separa o comum do científico presente na frase de Bourdieu, Chamboredon e Passeron (2015) pode ser traçada até Bachelard (2016, p.14): "a experiência que não retifica nenhum erro, que é monotonamente verdadeira, sem discussão, para que serve? A experiência científica é portanto uma experiência que contradiz a experiência comum" (BACHELARD, 2016, p.14).

Tanto Bachelard quanto Bourdieu entendem que a ciência é construída contra um outro tipo de conhecimento. Bachelard (2016, p.17) comenta que o "ato de conhecer dá-se contra um conhecimento anterior, destruindo conhecimentos mal estabelecidos, superando o que, no próprio espírito, é obstáculo à espiritualização".

De forma mais específica, a ciência de Bachelard se opõe absolutamente à opinião:

> de modo que a opinião está, de direito, sempre errada. A opinião pensa mal; não pensa: traduz necessidades em conhecimentos. Ao designar os objetos pela utilidade, ela se impede de conhecê-los. Não se pode basear nada na opinião: antes de tudo, é preciso destruí-la. Ela é o primeiro obstáculo a ser superado. (BACHELARD, 2016, p.18)

Bachelard (2016) também tece críticas ao que chama de experiência primeira, aquela que é, para ele, a "experiência colocada antes e acima da crítica – crítica essa que é, necessariamente, elemento integrante do espírito científico"(BACHELARD, 2016, p.29). Para o autor, a primeira experiência ou a observação primeira "é sempre um obstáculo inicial para a cultura científica" (BACHELARD, 2016, p.29), "não oferece nem o desenho exato dos fenômenos, nem ao menos a descrição bem ordenada e hierarquizada dos fenômenos" (BACHELARD, 2016, p.37).

Bourdieu, Chamboredon e Passeron (2015, p.24) têm a mesma visão quanto às questões ligadas à sociologia ao colocar que as "opiniões primeira sobre os fatos sociais apresentam-se como uma coletânea falsamente sistematizada de julgamentos com uso alternativo" (BOURDIEU; CHAMBOREDON; PASSERON, 2015, p.24).

Mesmo atribuindo uma grande carga negativa à opinião e ao senso comum, Bachelard e Bourdieu não veem a ciência como um empreendimento perfeito. Ambos reconhecem alguns limites e problemas com a forma de conhecimento científico, mas ainda assim têm uma visão que o coloca acima dos outros tipos de conhecimento.

Para lidar com estas questões, Bachelard usa a ideia de retificação como base do movimento e peculiaridade do pensamento científico. Para Lemos (2018, p.140), Bachelard vê a ciência contemporânea como um "discurso objetivo, isto é, em estruturas algébrico matemáticas que devem objetivar-se em fenômenos técnicos. O que há, portanto, é apenas um processo infinito de retificações de erros que não quer chegar, mas apenas instruir e construir, até onde permite a aventura da imaginação humana".

O conhecimento retificado muda o estatuto do que seria a "verdade" na formação do conhecimento científico, uma vez que para o filósofo francês "não há possibilidade de verdade, mas reconhecimento e consequente retificação de erros, que podemos traduzir como sendo isto a própria essência do ato de conhecer, ato este que, em si mesmo, não parte nem quer chegar, já que é puro devir" (LEMOS, 2018, p.144). Nas palavras do próprio Bachelard: "Cientificamente, considera-se o verdadeiro como rectificação histórica de um longo erro, considera-se a expêriencia como rectificação de uma ilusão comum e inicial" (BACHELARD, 2001, p.143).

A possibilidade de retificação é um dos elementos que segrega o senso comum, conhecimento primeiro, do conhecimento científico: "mas há pensamentos que não recomeçam: não os

pensamentos que foram rectificados, alargados, completados. Não retornam à sua área restrita ou vacilante. Ora, o espírito científico é essencialmente uma rectificação do saber, um alargamento dos quadros do conhecimento" (BACHELARD, 2001, p.143). Em uma outra passagem, Bachelard (2016, p.29) resume a mesma questão ao afirmar que "O espírito científico deve formar-se enquanto se reforma". Ou seja, a retificação é um processo pelo qual o conhecimento científico revisa a si próprio e pode ser entendido como um vetor que une a ruptura, a construção do objeto e a verificação, citados anteriormente.

A possibilidade de retificação em si do conhecimento científico representa uma ruptura inicial com o conhecimento primeiro ou senso comum. Uma segunda parte é a construção discursiva do objeto de pesquisa, como afirma Bachelard: "a ciência constrói seus objetos, que nunca ela os encontra prontos" (BACHELARD, 2016, p.77).

No caso de Bachelard (2016, p.7-8), esta construção é feita sob o domínio de uma matemática que deixa de ser descritiva para tornar-se formadora do objeto. Em outras palavras, o fenômeno não existe independente do processo de conhecer, ele é construído de forma discursiva. Entretanto, a matemática de Bachelard é uma matemática específica que, inclusive, deixa de lado a geometria, considerando esta última como baseada em uma espécie de "realismo ingênuo das propriedades espaciais, implica ligações mais ocultas, leis topológicas menos nitidamente solidárias com as relações métricas imediatamente aparentes" (BACHELARD, 2016, p.7).

Já o processo de construção de Bourdieu também passa por uma objetificação, mas significativamente diferente da de Bachelard, uma vez que aquele renega um enquadramento puramente analítico, tecendo críticas a uma abordagem objetivista da sociologia que parece mais próxima da forma de trabalho pouco empírica de Bachelard. A crítica de Bourdieu se estende também a abordagem fenomenológica, culminando em um processo que consiga transcender a dualidade das duas escolas (BOURDIEU; CHAMBOREDON; PASSERON, 2015, p.8-10).

Levando em conta o que consideramos ruptura, nesta tese, este movimento é representado por uma análise crítica de como o conceito de pensamento computacional é articulado em artigos publicados nos anos de 2015, 2016 e 2017 em um Workshop de Ensino em Pensamento Computacional, Algoritmos e Programação (WAlgProg) do Congresso Brasileiro de Informática na Educação (CBIE).

O movimento de construção do objeto de pesquisa é feito de duas maneiras: primeiro, um resgate histórico do que é chamado de abstração dentro da computação e em um segundo momento, oferecemos uma nova interpretação, com base em uma filosofia materialista, ironicamente

parte do corpo teórico dos objetivistas renegados por Bourdieu, do que poderia ser o processo de abstração. Para que a mudança não seja somente superficial, também articulamos a troca do conceito de paradigma de programação, esse diretamente ligado ao ensino e aprendizagem de programação de *software* pelo de gêneros discursivos oriundos da abordagem bakhtiniana.

Neste ponto, antes de entrar na verificação, podemos discutir um elemento importante para teoria bourdieusiana[29] e como nesta tese fazemos, ao mesmo tempo, um movimento de aproximação e distanciamento deste conceito: a objetivação participante, ou a própria reflexividade de seu método. É pela objetivação participante que Bourdieu convoca o pesquisador em ciências sociais a aplicarem os mesmos métodos de análise aplicados aos seus objetos de pesquisa a si mesmos (GRENFELL, 2018b, p.291). Para Bourdieu, isto é necessário para evitar o que chama de "falácia escolástica" em que "aquilo que é oferecido em nome do conhecimento científico é na realidade simplesmente a reprodução de uma certa relação escolástica com o mundo, relação essa imbuída de seus próprios interesses" (GRENFELL, 2018b, p.290-291).

A escolha dos textos que fazem parte de um congresso recorrente da área de pesquisa em ensino e aprendizagem em Ciência da Computação é o primeiro, único e curto passo nosso com relação a esta reflexividade.

Por fim, na verificação é em que propomos uma mudança quanto à forma de pesquisa apresentada até aqui. Esta última etapa, que para Bachelard é o experimento e para Bourdieu é a ida ao "campo"[30], foi substituída por duas iniciativas: o experimento didático trabalhado por Hiele-Geldof e Hiele (1984), melhor explicitado na próxima seção e uma análise documental de materiais produzidos nestas atividades, em especial, os e-mails trocados entre docente e discentes e os códigos-fonte produzidos.

1.4.2 Do carácter não positivista da tese

Dina van Hiele-Geldof abre sua tese de doutorado com uma discussão sobre o que seria um experimento didático. Ela começa diferenciando esta modalidade dos experimentos feitos dentro das ciências naturais com o argumento de que é possível isolar fatores e descartar aqueles indesejados, algo muito mais difícil de se fazer nos experimentos didáticos. Além disso,

[29] Escolhemos usar "bourdieusiana" no lugar de outras grafias, como bourdieurianos ou bourdieunianos apenas para seguir a usado nos materiais citados. Para entender um pouco melhor as implicações disto, ver notas de rodapé em Corrêa e Peters (2011).

[30] Campo no sentido de ir até o encontro do objeto de estudo, no mesmo sentido de operacionalização do estudo etnográfico.

Hiele-Geldof e Hiele (1984) aponta que a principal diferença entre ambas reside na subjetividade do observador: nas ciências naturais, os fenômenos são observados indiretamente, diferente do que ocorre nos experimentos didáticos. Isso faz com que a subjetividade do observador, no tipos de experimento de interesse de Hiele-Geldof e Hiele (1984), seja um fator impossível de ser eliminado (HIELE-GELDOF; HIELE, 1984, p.13).

A tese de Dina van Hiele-Geldof tinha como objetivo investigar melhoras na aprendizagem com a mudança do método de aprendizado. Junto com a definição das suas questões de investigação, ela oferece duas razões, das quais compartilhamos, do porquê não usar um grupo de controle ensinado com meios tradicionais:

> Primeiro, eu não consigo mais ensinar de acordo com os métodos tradicionais na primeira turma. O grupo que seria ensinado por mim pelo meio tradicional não estaria mais sob circunstâncias normais. Segundo, os resultados dos dois métodos não são comparáveis. Os grupos devem ser avaliados de acordo com uma referência. Esta referência é criada com base em um ou outro método. Por isso, os resultados parecem ser predeterminados. (HIELE-GELDOF; HIELE, 1984, p.16) [31]

Esta questão é um desdobramento direto do entendimento de Hiele-Geldof e Hiele (1984) quanto à impossibilidade de se eliminar o olhar subjetivo do(a) pesquisador(a)/docente. Ela rejeita o uso de grupos de controle, uma das técnicas tradicionais de pesquisa científica, pois sua atuação como professora não é um dos fatores possíveis de isolar e descartar.

A título de exemplo e discussão, podemos citar o trabalho de Vera *et al.* (2016), em que é descrita uma experiência de ensino de programação com elementos do raciocínio musical para trabalhar com o conceito de ordenação. Os estudantes, 14 ao todo, foram divididos aleatoriamente, metade teve aulas no formato tradicional e a outra com o uso do raciocínio musical. Além disso, o primeiro grupo teve aula com um estudante concluinte do curso de Bacharelado em Ciência da Computação, enquanto o segundo, um estudante de mestrado em Ciência da Computação com Licenciatura em Matemática e cursando Licenciatura em Computação.

De um ponto de vista de um experimento controlado, existem diversas variáveis independentes, que tornam difícil estabelecer uma relação de causa e efeito e uma posterior generalização. Talvez, a mais óbvia é o perfil de quem ministrou cada um dos materiais diferentes. Entretanto, isto não invalida o estudo de Vera *et al.* (2016), apenas mostra algumas questões que o modo de trabalho dos experimentos tradicionais exige e, em um contexto de ensino e aprendizagem, tornam-se muito difíceis de serem levados em conta.

[31] *First, I no longer feel able to teach according to the traditional methods in the first class. The group which would be taught in the traditional way by me would then not work under normal conditions. Second, the results of the two methods are not really comparable. The groups should be judged according to a standard. This standard is set in relation to one method or the other. Hence, the results seem to be predetermined.*

A contribuição de olhar para o trabalho de Vera *et al.* (2016), sob o entendimento de Hiele-Geldof e Hiele (1984), é entender que o estudo comparativo é uma escolha das e dos envolvidos com a pesquisa e não uma exigência para que se tenha um teor científico, pois pesquisas práticas envolvendo ensino e aprendizagem não ocorrem em ambientes controlados e nem com objetos inanimados, mas com pessoas.

Bakthin oferece uma visão similar a de Hiele-Geldof e Hiele (1984), mas com bases em uma filosofia da linguagem. Para ele:

> As ciências exatas são uma forma monológica do saber: o intelecto contempla uma coisa e emite enunciado sobre ela. Aí só há um sujeito: o cognicente (contemplador) e falante (enunciador). A ele só se contrapõe a coisa muda. Qualquer objeto do saber (incluindo o homem) pode ser percebido e conhecido como coisa. Mas o sujeito como tal não pode ser percebido e estudado como coisa porque, como sujeito e permanecendo sujeito, não pode se tornar mudo; consequentemente, o conhecimento que se tem dele só pode ser dialógico. (BAKHTIN, 2017, p.66)

É importante notar que nesta curta passagem, Bakhtin (2017) estabelece uma diferença fundamental do que seriam os objetos de estudo das ciências exatas e das ciências humanas. Na primeira, existe um sujeito (pesquisador ou pesquisadora) que é responsável pelos atos de pesquisa com um objeto que apenas "reage" a estímulos, sejam eles quais forem. Já nas ciências humanas, aquilo que chamamos de objeto é também um sujeito que não deve ser reduzido a algo que apenas "reage" a estímulos. Souza e Albuquerque (2012, p.110) resumem bem a questão ao colocar que o desafio inicial de uma epistemologia de base bakhtiniana é a "caracterização do que é conhecer um objeto, e o que é conhecer um indivíduo, outro sujeito cognoscente" (SOUZA; ALBUQUERQUE, 2012, p.110).

Souza e Albuquerque (2012, p.111) afirmam que uma epistemologia das ciências humanas de Bakhtin "tem como premissa problematizar a forte presença do positivismo no pensamento ocidental moderno, criando outra possibilidade de se produzir conhecimento dentro das ciências humanas" (SOUZA; ALBUQUERQUE, 2012, p.111) e também considera fundamental entender os conceitos de dialogismo e alteridade bakhtinianos para refletir sobre a relação entre o pesquisador e seu "outro" no processo de pesquisa.

A antropologia talvez seja um dos campos de conhecimento que mais preocupou-se com esta subjetividade do observador, dada a importância do trabalho de campo e sua inerente relação com a subjetividade de quem faz a pesquisa (GROSSI, 1992, p.7-8). Além do olhar para suas próprias práticas, a antropologia também nos ajuda a mostrar que esta percepção de Dina sobre subjetividade adentra, inclusive, nas ciências naturais. Por exemplo, Latour e Woolgar (1986,

p.176-177) mostram que mesmo dentro de um laboratório de bioquímica, a construção do que é um fato científico é um processo complexo que envolve interpretação, edição e readequação de declarações sobre um objeto de análise. Em outras palavras, o trabalho científico aparenta ser uma observação neutra do que seria a realidade, porque ele é construído discursivamente como tal e não necessariamente de forma consciente pelo pesquisador.

Além da antropologia, um outro campo de pesquisa que parece ter as mesmas indagações que van Hiele-Geldof sobre a relação entre sujeito-objeto ou entre sujeito-sujeito é a pesquisa-ação. Tripp (2005, p.445) define a Pesquisa-ação dentro da educação como "uma estratégia para o desenvolvimento de professores e pesquisadores de modo que eles possam utilizar suas pesquisas para aprimorar seu ensino e, em decorrência, o aprendizado de seus alunos". Tanto o experimento didático em Hiele-Geldof e Hiele (1984) quanto a pesquisa-ação em Tripp (2005) trazem à tona questões ligadas à mescla entre trabalho docente e de pesquisa acadêmica.

Hiele-Geldof e Hiele (1984) comentam sobre a falta de publicações por parte de professores de escolas experimentais - professores acreditam ser quase impossível conseguir tempo para escrever artigos além de lidar com suas tarefas cotidianas na escola[32] (HIELE-GELDOF; HIELE, 1984, p.21).

Tripp (2005, p.451), ao oferecer um exemplo pessoal sobre o que chama de processo indutivo dentro da pesquisa-ação, chega a um lugar parecido com o de Hiele-Geldof e Hiele (1984):

> Embora esteja claro que estou envolvido em alguns processos de teorização indutiva, estes não passam de meios para o fim de melhorar a prática, mas não um fim em si mesmo, o que explica por que os práticos não desenvolvem sua teorização sob a forma de uma teoria disciplinar: estão muito ocupados com suas práticas para perseguirem questões "puras" de pesquisa. (TRIPP, 2005, p.451)

Tanto a questão da não neutralidade do observador/investigador quanto a problemática citada sobre trabalho docente e pesquisa acadêmica são condensadas na questão bakhtiniana da responsabilidade sem álibi que é viver, Faraco (2017) expressa-a muito bem:

> [...] eu não posso ser não ser participante da vida real. E essa obrigação decorre de eu ser único e ocupar um lugar único, irrepetível, insubstituível e impenetrável da parte de um outro. Sou insubstituível e esse fato me obriga a realizar minha singularidade peculiar: tudo o que pode ser feito por mim não poderá nunca ser feito por ninguém mais, nunca. (FARACO, 2017, p.155)

A passagem de Faraco (2017) deve ser entendida como o reconhecimento de que o concreto da vida é algo sempre singular e irrepetível, maior do que as formas abstratas de

[32] *Teachers find it almost impossible to find time to write publications in addition to their particularly difficult daily tasks at school.*

conhecimento, incluso o científico. Além disso, também atesta o caráter axiológico de nossas ações, essas sempre responsivas.

É importante colocar que não estamos a negar a validade do método científico clássico, mas mostrar que ele exige contornos bem definidos que, do ponto de vista bakhtiniano, acabam por "desfigurar" o próprio objeto da análise na tentativa de gerar conhecimento sobre o mundo social. Existem duas questões importantes aqui: a primeira é análoga ao experimento mental conhecido como Demônio de Laplace[33], pois para criar um experimento realmente controlado, além do problema já colocado por Hiele-Geldof e Hiele (1984), seria preciso modelar todas as variáveis e constantes envolvidas no experimento, um empreendimento virtualmente possível, mas pouco provável, pois cada turma de estudantes é composta por pessoas diferentes com caminhos de vida diferentes.

Segundo, mesmo que não seja possível modelar uma equação que ajuste o processo de aprendizagem, a construção do experimento que vise à generalização é a segunda questão. Baudrillard comenta sobre um conto de Borges em que os cartógrafos de um reino, que para produzir mapas extremamente precisos, o fizeram com as mesmas dimensões dos objetos representados. Da mesma forma que o mapa "substituiu" o local original, ao configurar uma situação de ensino e aprendizagem a fim de ser mensurável e objetivada, coloca-se o processo de escrita científica acima da situação concreta que envolve professores e estudantes. Esse não deixa de ser um desdobramento da questão da subjetividade do observador, mas desta vez colocando de forma clara que a tentativa de objetivação também é parte da subjetividade do observador.

Além disso, existe a questão ética que também foi expressada por Hiele-Geldof e Hiele (1984): Se, mesmo que subjetivamente e, ao levar em conta a fragilidade de uma racionalização neste campo eu ainda devo tentar usar algo como um grupo de controle ou criar exercícios que são mensuráveis com facilidade? Em outros termos: Devo submeter os processos concretos de ensino e aprendizagem com pessoas reais em situações irrepetíveis e únicas a uma lógica da escrita científica abstrata e distante?

Com estas questões em mente, podemos definir esta pesquisa como qualitativa e exploratória. Para Deslauriers e Kérisit (2008, p.130), a pesquisa de natureza exploratória pode servir "para determinar os impasses e os bloqueios, capazes de entravar um projeto de pesquisa em grande escala" ao mesmo tempo que "possibilita familiarizar-se com as pessoas e suas preocupações". No caso desta tese, queremos investigar a possibilidade de articular bases teóricas

[33] Conhecidas todas as variáveis de uma única equação que rege o universo, dado um instante t seria possível prever qualquer instante $t+1$ de qualquer coisa dentro do universo.

materialistas, essas fora do escopo tradicional do ensino e aprendizagem de fundamentos de computação com o intuito de gerar atividades didáticas com potencial de contribuir com as iniciativas que tentam ampliar o acesso da Ciência da Computação para o público em geral.

Quanto ao carácter qualitativo, além da recusa ao uso de métodos quantitativos (com relação a pessoas) e suas perspectivas teóricas, também podemos adicionar as considerações de Denzin, Lincoln e Netz (2006, p.17) quanto à contingência do trabalho do pesquisador qualitativo: "Havendo a necessidade de que novas ferramentas ou técnicas sejam inventadas ou reunidas, assim o pesquisador o fará". A escolha das práticas de pesquisa depende do contexto e do que é possível ao pesquisador naquele cenário (DENZIN; LINCOLN; NETZ, 2006, p.18).

1.5 DA ORGANIZAÇÃO DA TESE

O corpo da tese está estruturado em três partes representadas pelos três próximos capítulos. Primeiro, o Capítulo 2, com o título: "Crítica ao Pensamento Computacional", representa a ruptura descrita anteriormente. Este capítulo visa mostrar que o entendimento do campo da Computação ligado ao seu ensino e aprendizagem adota a abstração como um conceito-chave, mas o vê de forma bem específica e sem muito aprofundamento filosófico explícito.

Dado este diagnóstico, no Capítulo 3, intitulado "As concretudes da abstração", propomos usar uma filosofia de base materialista para trabalhar com o conceito de abstração. Com esta base ententendos o abstrato como um processo e que obrigatoriamente deve levar em conta outros elementos além daquilo que é alvo da abstração, tais como as pessoas, contextos e instrumentos envolvidos no processo.

O Capítulo 4 tem como título "Da Resolução de Equações para a escrita de textos". Neste capítulo, apresentamos uma análise das trocas de e-mails em dois contextos educacionais que mobiliza alguns conceitos da análise dialógica de disrcurso para evidenciar como as teorias dos dois capítulos anteriores podem ser articuladas. Estas trocas de e-mail são apresentadas dentro de três atividades feitas em disciplinas de programação orientada a objetos. O primeiro é uma atividade derivada do jogo de cartas Super Trunfo. A segunda atividade tem como base um tabuleiro de xadrez e, ao final, a terceira mostra o uso de filmes como conteúdo de uma atividade.

No último capítulo, "Epílogo", fazemos uma curta retropectiva sobre os temas abordados nos capítulos anteriores. Este "capítulo" não estava nos planos iniciais, mas dado o rumo que discussões sobre educação tomaram durante a pandemia, pareceu importante deixar claras

algumas posições teóricas e instrumentais, principalmente quanto às questões que se aproximam do ensino a distância.

Por fim, ao longo do texto escolhemos trabalhar com a primeira pessoa do plural por dois motivos: primeiro, por considerar que o processo de escrita desta tese é coletivo, mesmo que seu formato seja de monografia. E em segundo lugar, explicitar que a pesquisa foi feita por pessoas, logo sua descrição tem julgamentos de valor implícitos que dada à base bakthiniana, a negação do uso da primeira pessoa apenas tenta mascarar.

2 O PENSAMENTO COMPUTACIONAL

Neste capítulo, pretendemos discutir as implicações do uso do termo "pensamento computacional" para o ensino e aprendizagem de computação. Queremos apontar que o termo é usado de forma indiscriminada como sinônimo de algum tipo de habilidade que cientistas da computação devem ter e fomenta diversas iniciativas sobre ensino e aprendizagem no campo.

De início, oferecemos uma análise do texto que é responsável pela popularização do termo (WING, 2006) e enfatizamos que sua construção e retórica é diferente do padrão de um artigo científico. A autora provavelmente nem tinha como objetivo o uso de uma retórica científica e, inclusive, sua tecitura tem semelhanças com manifestos artísticos, muito usados na virada do séc XIX para o XX.

Além de apresentar a problemática quanto à definição inicial do que seria o "pensamento computacional", também oferecemos uma análise de seu uso em artigos publicados no Workshop de Ensino em Pensamento Computacional, Algoritmos e Programação (WAlgProg) do Congresso Brasileiro de Informática na Educação (CBIE) nos anos de 2015, 2016 e 2017.

A escolha deste *workshop* se deu por três motivos: primeiro, pelo fato de seu tema ser diretamente o ensino de pensamento computacional. Segundo, este é um evento que faz parte do congresso mais importante da Informática na Educação (CBIE) em território brasileiro. Por fim, para justificar a escolha de um congresso e não uma revista científica, o campo da Ciência da Computação, conforme levantado por Zhang e Glänzel (2012), Vrettas e Sanderson (2015), Fiala e Tutoky (2017) e Bar-Ilan (2010), parece dar muito mais importância a congressos do que outros campos de conhecimento. Isto não significa que os congressos substituam as revistas, uma vez que estes estudos também mostram que os artigos publicados naqueles tendem a citar mais revistas do que outros textos de congressos.

Como ponto de partida para analisar estes textos, levantamos a possibilidade de que dentro da escrita científica no campo ensino e aprendizagem de computação existe a persistência de um discurso muito próximo do senso-comum. Como o trecho a seguir, parte de um dos artigos publicados no *workshop* em 2017: "O dinâmico contexto educacional elege os dispositivos móveis como uma ferramenta para auxiliar o professor a promover no aluno sua aprendizagem, autonomia, criticidade e criatividade" (LEITE *et al.*, 2017).

Os textos apresentados neste capítulo mostram que a atividade científica, dentro do ensino e aprendizagem de computação, não necessariamente faz uma ruptura total com senso-

comum, como é o caso de diversas formas e situações em que o termo "pensamento computacional" aparece.

Ao final do capítulo, também chamamos a atenção para problemas nas visões sobre tecnologia que aparecem em diversos textos do *workshop*. Com o aporte dos estudos em Ciência, Tecnologia e Sociedade, em especial, em algumas colocações específicas de Vieira Pinto (2005a) e seu conceito de Catabase da Técnica, tentamos mostrar que existem reflexões importantes a serem feitas quanto ao papel de conceitos-chave na dinâmica do processo de pesquisa científica em Ensino e Aprendizagem ligadas à Computação.

2.1 O PENSAMENTO COMPUTACIONAL PARA JEANNETTE WING

Em 2006, Jeannette Marie Wing[1], na época professora do departamento de Ciência da Computação da Universidade Carnegie Mellon, escreve um texto para a coluna "*Point of View*" da revista "*Communications of the ACM*[2]" com três páginas entitulado Pensamento computacional[3]. Em uma espécie de manifesto ou panfleto, Wing (2006) defende que o pensamento computacional é uma habilidade fundamental para todas as pessoas, não somente para cientistas da computação.

Wing aborda o tópico em diversos momentos nos anos seguintes (HENDERSON; CORTINA; WING, 2007; WING, 2008; WING, 2011; WING, 2021; WING; STANZIONE, 2016), mas sem se afastar muito do conteúdo do texto de 2006. Além disso, o primeiro texto é de longe o mais citado entre os textos que abordam o tema. Assim, escolhemos abordar o primeiro texto nesta análise, pois a autora não o renega posteriormente e é para ele que outros textos fazem referência. De forma geral, Wing (2006) equipara a importância do pensamento computacional a habilidades como leitura, escrita e aritimética, inclusive advogando para que faça parte das habilidades básicas de uma criança.

Em diversos momentos do texto, Wing (2006) defende que o pensamento computacional é algo abstrato, sem uma concretude específica. O texto é relativamente curto e ela começa vários parágrafos com a frase "Pensamento computacional é...[4]" seguido diversas vezes de "Ele é ...[5]" para criar várias definições curtas do que seria o pensamento computacional. É um texto

[1] Jeannette M. Wing (1956-), além de ser conhecida por popularizar o termo pensamento computacional, é professora da *Carnegie Mellon University* e da *Columbia University*, foi vice-diretora de Pesquisa na Microsoft.

[2] *Communications of the ACM* é a revista mensal da Association for Computing Machinery (ACM). CACM é enviada a todos os membros ACM.

[3] O título original em inglês é *Computational Thinking*.

[4] *Computational thinking is*.

[5] *It is...*

com uma grande circulação no meio acadêmico, em especial dentro das comunidades ligadas à educação e computação.

Mesmo definindo várias vezes o que é pensamento computacional, Wing (2006) não oferece uma definição precisa. Neste primeiro momento, ela constrói sua retórica através da anáfora e oferece diversos predicados para a oração "Pensamento computacional é...". A seguir, estão todos aqueles que são predicados da primeira frase de um parágrafo: "...uma habilidade fundamental para todos, não somente para cientistas da computação[6]","...pensar recursivamente[7]", "...usar de abstração e decomposição ao encarar uma tarefa grande e complexa ou projetar um sistema grande e complexo[8]", "...pensar em termos de prevenção, proteção e recuperação no pior caso possível através de redundância, contenção de danos e correção de erros[9]" e "usar de pensamento baseado em heurísticas para descobrir uma solução[10]".

O primeiro predicado, "...uma habilidade fundamental para todos, não somente para cientistas da computação", é o argumento principal do texto de Wing (2006).

O segundo predicado, "...pensar recursivamente", vem acompanhado também de "processamento paralelo[11]", "interpretar código como dado e dado como código[12]" "verificação de tipos como generalização da análise dimensional[13]", "reconhecer tanto as virtudes quanto os perigos de apelidos ou dar a alguém ou algo mais de um nome[14]", "reconhecer tanto o custo e o poder de endereçamento indireto e chamada de procedimento[15]", "julgar um programa não somente pela sua corretude e eficiência, mas por estética, e o projeto de um sistema por simplicidade e elegância[16]".

Todos os itens acima parecem ter alguma relação com linguagens de programação. Apenas recorrendo à interpretação e apenas a título de exemplo, pode-se associar a questão de "reconhecer tanto o custo e o poder de endereçamento indireto e chamada de procedimento" com o uso de ponteiros e funções, características importantes da linguagem de programação C e

6 *...is a fundamental skill for everyone, not just for computer scientists* (WING, 2006, p.33).
7 *...is thinking recursively* (WING, 2006, p.33).
8 *...is using abstraction and decomposition when attacking a large complex task or designing a large complex system* (WING, 2006, p.33).
9 *...is thinking in terms of prevention, protection, and recovery from worst-case scenarios through redundancy, damage containment, and error correction* (WING, 2006, p.34).
10 *...is using heuristic reasoning to discover a solution* (WING, 2006, p.33).
11 *...parallel processing* (WING, 2006, p.34).
12 *...interpreting code as data and data as code*(WING, 2006, p.33).
13 *...type checking as the generalization of dimensional analysis* (WING, 2006, p.33).
14 *...recognizing both the virtues and the dangers of aliasing, or giving someone or something more than one name* (WING, 2006, p.33).
15 *...recognizing both the cost and power of indirect addressing and procedure call* (WING, 2006, p.33).
16 *...judging a program not just for correctness and efficiency but for aesthetics, and a system's design for simplicity and elegance*(WING, 2006, p.33).

"julgar um programa não somente pela sua corretude e eficiência, mas por estética" às questões ligadas à escrita de código "legível" para que mais de uma pessoa possa manipular uma base de código com facilidade.

O terceiro, "...usar de abstração e decomposição ao encarar uma tarefa grande e complexa ou projetar um sistema grande e complexo" vem antes de "separação de preocupações[17]", "escolher uma representação apropriada para um problema ou modelar seus aspectos relevantes para torná-lo tratável[18]", "usar de invariantes para descrever o comportamento de um sistema de forma sucinta e declarativa[19]", "ter a confiança que podemos seguramente usar, modificar e influenciar um sistema grande e complexo sem entender cada detalhe dele[20]", "modularizar algo para antecipar usuários multiplos ou pré buscar e guardar em cache para antecipar uso futuro[21]".

Este terceiro item parece fazer referência direta a uma forma recorrente da computação entender o que seria a abstração. Por exemplo, Kramer (2007) trabalha com uma noção de que abstrair inclui "o ato de retirar ou remover algo e o ato ou processo de deixar de fora uma ou mais propriedades de um objeto complexo para atender a outros"[22]. É esta noção que Wing (2006) evoca ao escolher palavras como "representação apropriada", "aspectos relevantes" e "sucinta".

Junto com o quarto, "...pensar em termos de prevenção, proteção e recuperação no pior caso possível através de redundância, contenção de danos e correção de erros", temos "chamar engarrafamento, impasse e interfaces de contrato[23]" e "aprender a evitar condições de corrida quando for sincronizar encontros com outros[24]". Este tópico faz referência direta a conhecimentos associados a Sistemas Gerenciadores de Banco de Dados (SGBD) e Sistemas Operacionais (SO).

No último, "usar de pensamento baseado em heurísticas para descobrir uma solução", temos "planejar, aprender e agendar na presença da incerteza.[25]" "busca, busca e mais busca, resultando em uma lista de páginas da *web*, uma estratégia para ganhar um jogo ou

17 *...separation of concerns* (WING, 2006, p.33).
18 *...choosing an appropriate representation for a problem or modeling the relevant aspects of a problem to make it tractable* (WING, 2006, p.33).
19 *...using invariants to describe a system's behavior succinctly and declaratively* (WING, 2006, p.33).
20 *...having the confidence we can safely use, modify, and influence a large complex system without understanding its every detail* (WING, 2006, p.33).
21 *...modularizing something in anticipation of multiple users or prefetching and caching in anticipation of future use* (WING, 2006, p.34).
22 *The act of withdrawing or removing something, and the act or process of leaving out of consideration one or more properties of a compolex object so as to attend to others.*
23 *...calling gridlock, deadlock and contracts interfaces* (WING, 2006, p.34).
24 *...learning to avoid race conditions when synchronizing meetings with one another* (WING, 2006, p.33).
25 *...planning, learning, and scheduling in the presence of uncertainty* (WING, 2006, p.33).

um contra-exemplo[26]", "é usar grandes quantidades de dado para aumentar a velocidade de computação[27]", "fazer trocas entre tempo e espaço e entre poder de processamento e capacidade de armazenamento[28]".

É interessante notar o esforço de Wing (2006) em tentar descrever várias tarefas associado aos conhecimentos ligados à Ciência da Computação sem usar seus termos e conceitos, ao mesmo tempo que os usa em outras situações. Por exemplo, "busca, busca e mais busca, resultando em uma lista de páginas da *web* ..." faz referência direta aos serviços de busca e "fazer trocas entre tempo e espaço e entre poder de processamento e capacidade de armazenamento" são os problemas abordados pelo campo da complexidade em algoritmos. Ambos usam conceitos corriqueiros da computação, enquanto "ter a confiança que podemos seguramente usar, modificar e influenciar um sistema grande e complexo sem entender cada detalhe dele" tem relação clara com conceitos do campo tais como modularização, compartimentação e encapsulamento, mas tenta descrever a problemática sem citá-los.

A tentativa de generalizar é uma forma de construir o pensamento computacional descolado de seu nicho e de seus instrumentos e artefatos.

Alguns destes predicados, tais como "separação de preocupações" e "processamento paralelo" são genéricos e amplos, sem muito significado. Outros, mesmo que sejam parte da computação, também não parecem ser exclusividade do pensamento computacional: "planejar, aprender e agendar na presença da incerteza". Ela ainda tenta relacionar estas definições a exemplos do cotidiano, associando cacheamento com a arrumação da mochila de uma criança antes dessa ir para a escola, escolher uma fila no mercado com modelagem de sistemas com múltiplos servidores e outros casos mais.

Na segunda parte do texto, ela tenta definir o pensamento computacional por meio daquilo que ele é e não é. Wing (2006) usa seis tópicos: "Conceitualizar, não programar", "Fundamental, não habilidade rotineira", "Uma forma como humanos, não computadores, pensam", "Complementa e combina pensamento matemático e engenheirístico", "Ideias, não artefatos" e "Para todos, em todo lugar".

No primeiro tópico, Wing (2006, p.35) afirma:

[26] *...search, search, and more search, resulting in a list of Web pages, a strategy for winning a game, or a counterexample* (WING, 2006, p.33).

[27] *...is using massive amounts of data to speed up computation* (WING, 2006, p.33).

[28] *...making trade-offs between time and space and between processing power and storage capacity* (WING, 2006, p.33).

> Conceitualizar, não programar. Ciência da computação não é programação de computadores. Pensar como um cientista da computação significa mais que conseguir programar um computador. Requer pensar em múltiplos níveis de abstração.[29] (WING, 2006, p.35)

De forma conceitual, não existem problemas ao separar programação de computadores de Ciência da Computação. Mas de um ponto de vista histórico e concreto, esta separação é mais complicada. Primeiro, o desenvolvimento da teoria associada à computação que Wing (2006) quer separar da programação, historicamente, surge e é feita de forma integrada.

Por exemplo, nos anos 40 do século XX, programas eram escritos em sua maioria por matemáticos e engenheiros usando diretamente código binário ou em mnemônicos (assembly). Foi em 1952 que um programa escrito por Grace Hopper, o primeiro compilador, permite a escrita de programas em uma linguagem de "alto-nível", que depois seriam convertidos para linguagem binária pelo compilador. É a partir desta linha de esforços que Grace Hopper projeta as linguagens MATHMATIC e FLOW-MATIC que, posteriormente, levam ao desenvolvimento da linguagem COBOL (GRUDIN; WILLIAMS, 2013).

A criação do compilador mostra o quanto o desenvolvimento da teoria sobre computação e a programação de computadores é interligada. Inclusive, ajuda a pensar que o "pensar em múltiplos níveis de abstração" atribuindo somente ao cientista da computação por (WING, 2006) também pode ser encontrado na programação de computadores. Mesmo que as formas de trabalho com computador tenham mudado desde então, no mínimo, é possível entender que existe uma área na qual não é muito clara a separação entre o que é necessário para programar um computador e os conhecimentos de um cientista da computação. Indo além, programação de computadores não é uma atividade secundária dentro da Ciência da Computação, como parece entender Wing (2006).

Ao notar que é possível criar narrativas que mostrem o quanto programação de computadores e Ciência da Computação são interligadas, fica clara a intenção da autora em separar a Ciência da Computação da programação de computadores. Com isso, cabe perguntar: Por que caracterizar Ciência da Computação e programação de computadores como diferentes? E quais são as implicações de aceitar este tipo de visão?

Separar as duas áreas de conhecimento é importante para que seja possível levar o que Wing (2006) chama de pensamento computacional para além dos limites do campo da computação. Ao particionar os dois campos de conhecimento sem nenhuma forma de sombreamento,

[29] *Conceptualizing, not programming. Computer science is not computer programming. Thinking like a computer scientist means more than being able to program a computer. It requires thinking at multiple levels of abstraction.*

o computador como objeto indispensável tende a ficar no lado da programação. Porém, este movimento de abstrair para universalizar tem em si um conceito de universalidade e racionalidade que precisam ser explicitados.

Além disso, a passagem "Pensar como um cientista da computação significa mais que conseguir programar um computador" dá indícios de que Wing (2006) acredita que os conhecimentos da Ciência da Computação sejam, de alguma maneira, superiores aos da programação de computadores. Este é um desdobramento que deve ser entendido junto com a racionalidade e universalidade acima citados.

Sobre o segundo tópico, Wing (2006) coloca:

> Fundamental, não uma habilidade rotineira. Uma habilidade fundamental é algo que todo ser humano deve saber para funcionar na sociedade moderna. Rotineira significa uma rotina mecânica. Ironicamente, enquanto a Ciência da Computação não resolver o grande desafio da Inteligência Artificial de fazer computadores pensar como humanos o pensamento será rotineiro.[30] (WING, 2006, p.35)

Esta é uma passagem que não contribui muito para a definição de pensamento computacional, mas serve como parte da justificativa sobre o quanto é necessário levar a computação a todos. Com isso, passamos para a terceira afirmação de Wing (2006):

> Uma forma que humanos, não computadores, pensam. Pensamento computacional é uma forma que humanos resolvem problemas; Não é tentar fazer com que pessoas pensem como computadores. Computadores são maçantes e chatos; humanos são espertos e imaginativos. Nós humanos fazemos computadores interessantes. Equipados com aparelhos computacionais, usamos nossa esperteza para abordar problemas que não teríamos coragem de encarar antes da era da computação e construir sistemas com funcionalidades limitadas somente por nossa imaginação. [31] (WING, 2006, p.35)

Neste trecho, existem duas ideias: em primeiro lugar, o pensamento computacional é uma faculdade humana e independente do computador. Esta questão está exposta de uma forma muito clara na segunda frase da passagem: "Pensamento computacional é uma forma que humanos resolvem problemas" (WING, 2006, p.35). Caracterizar o pensamento computacional como uma propriedade humana independente do computador é uma outra forma de dizer que programação de computadores e Ciência da Computação são coisas diferentes. Inclusive, este

[30] *Fundamental, not rote skill. A fundamental skill is something every human being must know to function in modern society. Rote means a mechanical routine. Ironically, not until computer science solves the AI Grand Challenge of making computers think like humans will thinking be rote.*

[31] *A way that humans, not computers, think. Computational thinking is a way humans solve problems; it is not trying to get humans to think like computers. Computers are dull and boring; humans are clever and imaginative. We humans make computers exciting. Equipped with computing devices, we use our cleverness to tackle problems we would not dare take on before the age of computing and build systems with functionality limited only by our imaginations.*

ponto reforça uma desmaterialização da dimensão concreta do trabalho com computadores. Ao tornar o computador um objeto opcional para a computação, são postas de lado as questões que envolvem sua produção, tais como quem, onde e sob quais condições são feitos seus componentes. Também tornam-se minoritárias questões sobre seus usos, como problemas associados aos seus gestos (LER, problemas de postura etc) e econômicos (uso do computador para acelerar uma desqualificação de trabalho, hipertrofia de sistemas para controle social etc).

A segunda ideia que existe neste trecho é o entendimento do computador como um instrumento ou até mesmo uma prótese sob o uso de pessoas, bem representado pelo trecho final do parágrafo: "Equipados com aparelhos computacionais [...] construir sistemas com funcionalidades limitadas somente por nossa imaginação" (WING, 2006, p.35). Esta sentença representa bem uma problemática abordada por pesquisadores ligados à corrente construtivista dos estudos em CTS. Winner (2010), por exemplo, coloca que máquinas e sistemas tecnológicos não deveriam ser julgados somente por quesitos como eficiência ou benfícios, mas também ser julgados pela maneira que incorporam visões específicas de poder ou autoridade (DAGNINO, 2008, p.87-88).

Um dos exemplos citados por Winner (2010) é o caso da adoção de um sistema de energia nuclear. Este tipo de matriz energética exige uma organização social autoritária para que seja garantida suas exigências mínimas de funcionamento (controle de acesso às usinas, controle sobre a produção do combustível, constrole sobre as formas de descarte dos resíduos do processo etc).

A visão ufanista da computação apresentada (WING, 2006, p.35), em que temos controle total sobre o computador e este processo é sempre benéfico, esconde um jogo de influência mútua que pode existir entre computação e sociedade. Por exemplo, controle de acesso a banco de dados ou sistemas operacionais, com as devidas proporções, tem embutidas em si o mesmo carácter autoritário levantado por Winner (2010) sobre a produção de energia nuclear.

É justamente o tom ufanista que esconde uma complexidade maior quanto às relações entre computação e sociedade. Em outro texto, Wing (2011) coloca que "os benefícios educacionais de conseguir pensar computacionalmente [...] melhoram e reforçam habilidades intelectuais e podem ser transferidas para qualquer domínio[32]". Wing (2011) ignora que podemos reforçar habilidades que não necessariamente são boas, tais como sistemas de controle autoritários. Também ficam de lado as práticas computacionais que são, na verdade, reflexos

[32] *The educational benefits of being able to think computationally [...] enhance and reinforce intellectual skills, and thus can be transferred to any domain.*

ou desdobramentos de práticas de outros campos do saber. Muitas vezes, artefatos e práticas computacionais, principalmente com relação a controle de acesso e disponibilidade, são regidos pelas mesmas regras que regem outras instituições sociais. Ou seja, nestes casos, pensamento computacional pode servir de cavalo de Troia para difundir práticas sociais específicas com uma roupagem algorítimica e formal.

A seguir, existe o reconhecimento de que o pensamento computacional, definitivamente, não tem uma concretude específica:

> Ideia, não artefatos. Não serão somente os softwares e hardwares que nós produzimos que estarão presentes fisicamente em todo lugar presentes em nossas vidas a todo o momento, serão os conceitos computacionais que usamos para resolver problemas, gerenciar nosso cotidiano e comunicar e interagir com outras pessoas.[33] (WING, 2006, p.35)

Além da separação entre uma dimensão concreta e abstrata que a primeira frase denota, de forma clara e explícita, encontramos novamente o movimento de abstrair para universalizar, principalmente no uso das palavras "em todo lugar"[34] e "em todo momento"[35], que são potencializados ao final, quando Wing (2006, p.35) afirma que tem em mente, inclusive, a comunicação e interação entre pessoas diretamente.

Por fim, temos:

> Para todos, em todo lugar. Pensamento computacional será realidade quando for tão integrado com as façanhas humanas que deixará de ser uma filosofia explicita[36] (WING, 2006, p.35)

Nesta passagem, mais uma vez temos a universalização do pensamento computacional, que mostra-se presente em dois momentos: no começo da passagem, "Para todos, em todo lugar"(WING, 2006, p.35) e na vontade da autora de que o pensamento computacional seja praticamente uma forma de cultura universal da humanidade.

Em resumo, tanto neste texto quanto em um segundo (WING, 2008), intitulado "Pensamento computacional e pensando sobre computação[37]", as definições e exemplos de Wing (2006), Wing (2008) giram em torno de dois elementos: 1) uma visão específica de abstração não muito bem definida, mas com desdobramentos importantes; 2) na defesa de que o pensamento computacional é uma forma de entendimento de mundo e que todos devem saber.

---

[33] *Ideas, not artifacts. It's not just the software and hardware artifacts we produce that will be physically present everywhere and touch our lives all the time, it will be the computational concepts we use to approach and solve problems, manage our daily lives, and communicate and interact with other people.*

[34] No original, *everywhere.*

[35] No original, *all the time.*

[36] *For everyone, everywhere. Computational thinking will be a reality when it is so integral to human endeavors it disappears as an explicit philosophy.*

[37] *Computational thinking and thinking about computing.*

Estes dois processos são integrados, como colocado anteriormente. O movimento de abstrair, no caso de Wing (2006), Wing (2008), serve como uma forma de ignorar a dimensão material do trabalho com computadores para que seja possível pensá-lo como universal.

Durante o período de escrita deste capítulo (2017-2018), existem, pelo menos, 417 textos na biblioteca digital da *Association for Computing Machinery* (ACM) que citaram diretamente Wing (2006). Já no Google Scholar, constam 2971 trabalhos que, de alguma maneira, citam o mesmo texto. No contexto brasileiro, vale a pena citar que o Workshop de Ensino em Pensamento Computacional, Algoritmos e Programação (WAlgProg) do Congresso Brasileiro de Informática na Educação (Congresso Brasileiro de Informática na Educação (CBIE))[38] e, como o nome sugere, tem como objetivo concentar "pesquisas e discussões sobre o ensino de Pensamento Computacional, Algoritmos e Programação" (WALGPROG, 2015).

Dado o que foi apresentado até então, é interessante notar que um texto não segue uma estrutura clássica quanto ao método científico[39], publicado na seção *Viewpoint* da revista *Comunications of the ACM*. É a origem de uma série de experiências e práticas ligadas à computação e à educação (TEDRE; DENNING, 2016).

Esta questão, mesmo que não reconhecida abertamente, foi abordada ao longo dos anos, como é possível ver na discussão feita em Selby e Woollard (2013), Selby e Woollard (2014) sobre os consensos e termos encontrados em publicações que de alguma maneira encontram-se ligadas ao pensamento computacional. Entre os encaminhamentos propostos, existem sugestões de definições mais precisas, de abandono das tentativas de nomeação, de foco na prática e de que ao ensinar computação, o pensamento computacional seria desenvolvido também automaticamente (SELBY; WOOLLARD, 2013, p.2).

Tedre e Denning (2016, p.125-126) fazem duras críticas ao movimento iniciado pelos textos de Wing (2006) e levantam sete riscos que este entusiasmo sobre "pensamento computacional" pode criar. Primeiro, colocam a falta de ambição[40] para o "pensamento computacional", uma vez que aqueles que estavam a pensar o que seria o "pensamento computacional" nos anos 80 do século XX tinham como objeto refletir sobre epistemologia e métodos e não em ferramentas e técnicas. Segundo, acusam o "pensamento computacional" de Dogmatismo[41], criticando, em especial, o discurso de que "pensamento computacional" seria o melhor método de pensar sobre

[38] Durante o levantamento de dados, o congresso estava em sua terceira edição (2017). Existiram mais duas edições posteriores em 2018 e 2019.

[39] O texto não tem referências a outros estudos, não tem dados empíricos e não usa nenhum tipo de método que vise à coesão e coerência quanto suas proposições. Em resumo, é um texto de opinião.

[40] *Lack of ambition.*

[41] *Dogmatism.*

e resolver problemas. Terceiro, Saber contra Fazer[42]. Tedre e Denning (2016) criticam uma visão de que "pensamento computacional" seria um corpo de conhecimento e não uma habilidade (que leva tempo) para ser desenvolvida.

Em quarto lugar, Tedre e Denning (2016) comentam sobre o que seriam as Reivindicações exageradas[43] do "pensamento computacional", dando ênfase nas afimações de que "pensamento computacional" confere habilidades em resolução de problemas em ambientes não computacionais, inclusive trazendo evidência de que isto não necessariamente ocorre.

Em quinto, visões limitadas sobre Computação[44]. Tedre e Denning (2016) se mostram contrários a uma ideia de que programação seria a essência do "pensamento computacional" ou da própria Ciência da Computação, deixando claro que codificar é cada vez menos importante. Dado o teor desta tese, precisamos fazer um contraponto a este tópico. Concordamos com Tedre e Denning (2016) que "pensamento computacional" ou Ciência da Computação não são equivalentes à programação. Mas a força com que esta divisão é repetida, mesmo que exista a componente histórica desta reação, Tedre e Denning (2016) comentam que o "mito" de que Ciência da "Computação = programação", o qual surge nos anos 70 e emerge novamente nos anos 90, é recorrente em círculos acadêmicos. Entendemos a programação como uma das atividades históricas da Ciência da Computação e importante para articular diversos conceitos computacionais. Qualificá-la como uma "outra coisa" muito diferente do que seria esta "alma" da computação é no mínimo contraditório em um texto que pede um olhar para a história do campo.

Tedre e Denning (2016) colocam como o sexto risco um Foco exagerado em "formulação"[45], sem uma definição espefícica. É abrangente demais, propondo que, talvez fosse melhor usar o termo *design* para tal. E, por fim, Perder de vista os modelos computacionais, em que Tedre e Denning (2016) defendem a importância de se saber o que os algoritmos estão controlando, dando destaque para o comportamento de máquinas, sejam elas teóricas ou concretas.

Dada a inconsistência que o conceito original de Wing (2006), Wing (2008) possui, faz-se necessário explorar os desdobramentos que o pensamento computacional teve em diferentes lugares e ao longo do tempo. Nas próximas seções é oferecida um estudo bibliográfico sobre os usos do "pensamento computacional" feitos em artigos publicados no Workshop de Ensino em

[42] *Knowing Versus Doing.*
[43] *Exaggerated Claims.*
[44] *Narrow Views of Computing.*
[45] *Overemphasis on Formulation.*

Pensamento Computacional, Algoritmos e Programação (WAlgProg) do Congresso Brasileiro de Informática na Educação (CBIE).

2.2 O PENSAMENTO COMPUTACIONAL NO WALGPROG (2015-2017)

Para separar os artigos em grupos, partimos de um pressuposto de que quanto maior for a quantidade de ocorrências de um termo (neste caso pensamento computacional), maior deve ser sua importância para o texto. Com isso, os artigos foram separados em quatro (a quantidade de grupos foi definida de modo a operacionalizar a análise, apenas mantendo uma quantidade aceitável em cada um dos grupos) grupos distintos dependendo de quantas vezes o termo "pensamento computacional" aparece nos texto: 1) Aqueles que não citam nenhuma vez; 2) Aqueles que citam no máximo dez vezes; 3) Aqueles que citam de onze até trinta vezes e 4) Aqueles que citam mais que trinta vezes.

Dentro do primeiro grupo, temos quinze artigos do ano de 2015, doze de 2016 e dezesseis de 2017. No segundo grupo, temos sete em 2015, oito em 2016 e sete em 2017. No terceiro, temos cinco no ano de 2015, sete em 2016 e seis em 2017. Por fim, no último grupo, temos somente três em 2015, dois em 2016 e três em 2017. A Tabela 4 apresenta os textos que fazem parte de cada grupo ainda dentro de seus anos de publicação.

Tabela 4 – Agrupamento de textos por grupo e ano de textos do WAlg Prog que citam "pensamento computacional"

Grupo	Ano	Quant.	Textos
2	2015	7	Arantes e Ferreira (2015), Lima e Sousa (2015b), Aguiar *et al.* (2015), Gomes *et al.* (2015), Lima e Sousa (2015a), Aguiar (2015), Raabe, Zanchett e Vahldick (2015)
2	2016	8	Souza e Castro (2016), Andrade *et al.* (2016), Oliveira e Barbosa (2016), Queiroz, Sampaio e Santos (2016), Vera *et al.* (2016), Silva, Souza e Silva (2016), Timmermann e González (2016), Ferreira *et al.* (2016)
2	2017	7	Borges *et al.* (2017), Anjos Jr. *et al.* (2017), Silva *et al.* (2017), Silva (2017), Souto e Tedesco (2017), Avila e Cavalheiro (2017), Barcelos *et al.* (2017a)
3	2015	5	Paiva *et al.* (2015), Zanetti e Oliveira (2015), Mestre *et al.* (2015), Araujo, Andrade e Serey (2015), Schoeffel *et al.* (2015)
3	2016	7	Fernandes e Silveira (2016), Bathke e Raabe (2016), Silva *et al.* (2016), Barcelos, Bortoletto e Andrioli (2016), Zimmermann *et al.* (2016), Souza, Rodrigues e Andrade (2016), Kampff *et al.* (2016)
3	2017	6	Geraldes *et al.* (2017), Barcelos *et al.* (2017b), Brackmann *et al.* (2017), Ortiz e Pereira (2017), Pinheiro, Franco e Leite (2017), Bauer *et al.* (2017)
4	2015	3	Farias, Andrade e Alencar (2015), França e Tedesco (2015), Barcelos *et al.* (2015)
4	2016	2	Cavalcante, Costa e Araujo (2016), Araujo, Andrade e Guerrero (2016)
4	2017	3	Costa *et al.* (2017), Leite *et al.* (2017), Raabe *et al.* (2017)

Fonte: Autoria própria

Para esta análise, o primeiro grupo se mostra irrelevante, uma vez que estamos interessados em entender de que maneira é articulado o discurso sobre "pensamento computacional" nos artigos publicados dentro do WalgProg nos anos de 2015, 2016 e 2017.

2.2.1 Artigos que citam no máximo dez vezes "pensamento computacional"

No segundo grupo, é claro o uso do termo "pensamento computacional" como uma espécie de termo genérico e não muito bem definido para um tipo de conhecimento que, de alguma forma, tem relação com a computação. O fato de aparecer poucas vezes nestes textos mostra que não existe a preocupação de um entendimento rico e aprofundado. Em todos os textos deste grupo, o "pensamento computacional" não é o objeto de estudo direto, algo que justifica, dentro deste contexto, a forma como o termo é usado. Por exemplo, em Aguiar (2015), "pensamento computacional" aparece somente uma vez na conclusão e é usado como uma forma de associar o objeto do texto à prática docente no campo da computação: "É interessante, portanto, que os docentes reflitam sobre a possibilidade de adotarem mecanismos de gamificação para o ensino de pensamento computacional e programação [...]" (AGUIAR, 2015, p.1444).

Nesta passagem, também temos um pequeno indício de que "pensamento computacional" não necessariamente engloba a atividade de programar computadores, uma vez que tanto "pensamento computacional" e "programação" aparecem separados. Algo que também está presente em Timmermann e González (2016), ao comentar que o conteúdo da disciplina de Algoritmos pode vir se tornar programas:

> Seu conteúdo [disciplina de Algoritmos] está diretamente relacionado com o desenvolvimento do pensamento lógico ou computacional (GIRAFFA, MULLER e MORAES, 2015) necessário para a elaboração de soluções algorítmicas que, mais tarde ou em concomitante, poderão se tornar programas escritos em diversas linguagens de programação. (TIMMERMANN; GONZáLEZ, 2016, p.1296)

Em Arantes e Ferreira (2015), a única passagem que faz referência ao termo é uma paráfrase de Resnick *et al.* (2009), em que é possível inferir que "pensamento computacional" não inclui usos corriqueiros do computador, mas sim as habilidades específicas do campo da computação: "Não basta saber criar textos, usar redes sociais e usar a Internet, é preciso saber também como os computadores funcionam e estimular o desenvolvimento do pensamento computacional [Resnick et al. 2009]" (ARANTES; FERREIRA, 2015, p.1218).

Em Barcelos *et al.* (2017a), o termo é escrito com o uso de caixa alta nas letras iniciais ("Pesamento Computacional"), indicando que o termo é importante o suficiente para ser

tratado como um nome próprio. Neste texto também encontramos uma passagem que enfatiza o "pensamento computacional" e a programação de computadores como distintas:

> O desenvolvimento de competências e habilidades relacionadas ao Pensamento Computacional e à programação de computadores envolve cada vez mais atividades colaborativas e complexas. (BARCELOS *et al.*, 2017a, p.1207)

Em Silva, Souza e Silva (2016), também existe esta separação na única passagem do texto que usa o termo:

> [. . .] sendo daí selecionados os alunos que participaram do projeto para aplicar com o Scratch o pensamento computacional e o ensino de programação na escola. (SILVA; SOUZA; SILVA, 2016, p.1293-1294)

Ainda na linha de separar de forma clara "pensamento computacional" do uso efetivo de uma linguagem de programação, temos em Aguiar *et al.* (2015):

> O principal objetivo das aulas é apresentar uma fundamentação da computação, principalmente trabalhar o pensamento computacional e o raciocínio lógico. Como ementa das aulas foi atribuído o conceito e tipos de representação de algoritmos (descrição narrativa, fluxograma e pseudocódigo), e conceitos como arranjo, vetor, matriz, ponteiro, sub-rotina e operadores (lógicos, aritméticos e relacionais). Nenhuma linguagem de programação é ensinada durante o Pré-Comp, visto que o foco principal é construir o pensamento computacional e oferecer ferramentas para que posteriormente o algoritmo para solução de um problema possa ser facilmente traduzido para uma linguagem. (AGUIAR *et al.*, 2015, p.1343)

É importante notar que na passagem "Nenhuma linguagem de programação é ensinada durante o Pré-Comp visto que o foco principal é construir o pensamento computacional"(AGUIAR *et al.*, 2015), fica claro o corte entre programação e "pensamento computacional", entretanto a presença de conceitos diretamente associados a linguagens de programação procedural, específicas, derivadas da linguagem C, tais como "arranjo, vetor, matriz, ponteiro, sub-rotina e operadores"(AGUIAR *et al.*, 2015, p.1343) deixa claro que na prática esta distinção é mais complexa do que estes textos fazem parecer.

Junto dos usos genéricos e indefinidos, também existem textos que associam o termo a algo benéfico, sem nenhuma especificação de como isso funcionaria. Por exemplo, em Anjos Jr. *et al.* (2017), existe uma defesa da presença do "pensamento computacional" no Ensino Médio:

> Com base nisto, a implementação do pensamento computacional em estudantes do EM poderia representar uma nova forma de se desenvolver, neste público, competências e habilidades que facilitassem a construção de soluções de problemas nas mais variadas áreas do conhecimento. (ANJOS JR. *et al.*, 2017, p.962)

Curiosamente, o uso do "Com base nisto" refere-se ao parágrafo anterior em que é comentado a grande taxa de mudança de curso e o baixo rendimento nas disciplinas iniciais de

cursos ligados à computação. Ou seja, “pensamento computacional” pode ajudar em diversas áreas, mas de forma concreta, aborda-se somente uma, a computação.

Em Silva (2017), existe uma citação que exagera o potencial universal do “pensamento computacional”:

> Nunes (2011) exemplifica a situação dos advogados que, na medida em que leem um texto, usam o pensamento computacional para extrair dele fatos e regras que justifiquem um parecer conclusivo. Outros procedimentos, como organizar processos, também podem ser apresentados de forma algorítmica. (SILVA, 2017, p.1142)

A citação atribuída a Nunes (2011) reduz todo um saber de um outro campo de conhecimento, no caso o direito, aos conceitos da computação, sem dar qualquer tipo de valor específico a práticas associadas a uma leitura que um advogado possa fazer. Não se está contestando a existência de tarefas análogas em ambos os campos, mas o exemplo evidencia a existência de uma crença na validade universal do que é chamado de “pensamento computacional”, que não necessariamente apresenta bases para tal. Este tipo de construção retórica é a mesma empregada por Wing (2006) ao tentar mostrar que nas práticas do cotidiano existe computação.

Além desses, existem os textos em que o termo aparece de forma circunstancial, como, por exemplo, em Oliveira e Barbosa (2016) e em Souto e Tedesco (2017). Em ambos, a ocorrência do termo deve-se ao uso do WalgProg como uma base de pesquisa. Já em Borges *et al.* (2017), o termo faz parte apenas do título de uma das referências usadas. Enquanto em Lima e Sousa (2015a), Lima e Sousa (2015b) e Andrade *et al.* (2016), “pensamento computacional” é usado nas análises de trabalhos similares ou relacionados.

Raabe, Zanchett e Vahldick (2015) e Souza e Castro (2016) usam o termo como um conceito com um entendimento partilhado pela comunidade, sem precisar de uma definição:

> A compreensão sobre a forma como estes jogos são percebidos pelos estudantes auxilia no planejamento de atividades de introdução ao pensamento computacional, possibilitando também delinear diretrizes para a concepção e o projeto de novos jogos. (RAABE; ZANCHETT; VAHLDICK, 2015, p.1486)
>
> Várias iniciativas vêm sendo tomadas para que o pensamento computacional e o raciocínio lógico sejam desenvolvidos desde cedo, uma delas é o uso de ferramentas para o auxílio do ensino desses conceitos. (SOUZA; CASTRO, 2016, p.1078)

Por fim, também existem aqueles que parecem citar o pensamento computacional como uma forma de conectar os temas de seus trabalhos com a temática do WalgProg, como é feito em Ferreira *et al.* (2016), Vera *et al.* (2016) e Avila e Cavalheiro (2017). A quantidade de vezes em que o termo aparece nestes textos (dois, um e cinco respectivamente) é algo que corrobora com esta constatação. Em Ferreira *et al.* (2016), uma das ocorrências é no título de uma das

referências bibliográficas e em Avila e Cavalheiro (2017), quatro fora do corpo principal do texto: uma no título, uma no resumo e duas nas referências bibliográficas.

A Tabela 5 mostra a quantidade de ocorrências do termo "pensamento computacional", a referência e o título dos artigos publicados no WalgProg em 2015, 2016 e 2017 com uma até dez ocorrências.

Tabela 5 – Textos do WAlgProg que usam o termo "pensamento computacional" no máximo dez vezes

Quant.	Citação	Título
1	Arantes e Ferreira (2015)	Uma dinâmica para ensino de conceitos fundamentais de programação
4	Lima e Sousa (2015b)	Experiência no Programa Institucional de Bolsas de Iniciação à Docência (PIBID): Desenvolvimento do Raciocínio Lógico e Algoritmo na Educação Básica
3	Aguiar *et al.* (2015)	Pré-Comp: introduzindo os fundamentos da Computação e contribuindo com a motivação e aproveitamento acadêmico
8	Gomes *et al.* (2015)	Avaliação de um Jogo Educativo para o Desenvolvimento do Pensamento Computacional na Educação Infantil
10	Lima e Sousa (2015a)	Desenvolvimento do Raciocínio Lógico e Algoritmo Através do Programa Institucional de Bolsas de Iniciação à Docência no Ensino Fundamental
1	Aguiar (2015)	Experiência baseada em Gamificação no Ensino sobre Herança em Programação Orientada a Objetos
2	Raabe, Zanchett e Vahldick (2015)	Jogos de Programar como uma Abordagem para os Primeiros Contatos dos Estudantes com à Programação
2	Souza e Castro (2016)	Investigação em programação com Scratch para crianças: uma revisão sistemática da literatura
6	Andrade *et al.* (2016)	Uma Proposta de Oficina de Desenvolvimento de Jogos Digitais para Ensino de Programação
1	Oliveira e Barbosa (2016)	Perfis de jogadores em contextos de ensino/aprendizagem em disciplinas de programação
3	Queiroz, Sampaio e Santos (2016)	DuinoBlocks4Kids : Ensinando conceitos básicos de programação a crianças do Ensino Fundamental I por meio da Robótica Educacional
1	Vera *et al.* (2016)	Dó, Ré, Mergesort: um relato de experiência interdisciplinar de ensino de computação com matemática e música
2	Silva, Souza e Silva (2016)	Aplicação da Ferramenta Scratch para o Aprendizado de Programação no Ensino Fundamental I
1/2	Timmermann e González (2016)	Mediações que os professores e alunos estabelecem com o conteúdo da disciplina de Algoritmos de cursos superiores: estudo de caso
3	Ferreira *et al.* (2016)	Hello World: relato de experiência de um curso de iniciação à programação
1	Borges *et al.* (2017)	Tutor Inteligente para Recomendação de Atividades de Programação em um Ambiente Virtual de Aprendizagem
3	Anjos Jr. *et al.* (2017)	Avaliação de linguagens visuais de programação no ensino médio a partir da utilização do conceito de Robótica Pedagógica
6	Silva *et al.* (2017)	Raciocínio Lógico nas Escolas: Uma Introdução ao Ensino de Algoritmos de Programação
4	Silva (2017)	Desenvolvendo a Programação de Jogos Digitais no Ensino Médio: um Relato de Experiência Utilizando a Ferramenta Construct2
2	Souto e Tedesco (2017)	Uma Revisão sistemática da Literatura sobre conhecimentos, habilidades, atitudes e competências desejáveis para auxiliar a aprendizagem de programação

(continua)

Tabela 5 – Textos do WAlgProg que usam o termo "pensamento computacional" no máximo dez vezes (continuação)

Quant.	Citação	Título
5	Avila e Cavalheiro (2017)	Robótica Educacional como Estratégia de Promoção do Pensamento Computacional - Uma Proposta de Metodologia Baseada em Taxonomias de Aprendizagem
5	Barcelos *et al.* (2017a)	Análise automatizada do discurso de aprendizes de programação: relações entre emoções e nível de experiência

Fonte: Autoria própria

2.2.2 Artigos que citam de onze até trinta vezes "pensamento computacional"

O terceiro grupo é composto pelos textos que possuem de onze até vinte e sete ocorrências do termo "pensamento computacional". Esté é o grupo com mais artigos, possuindo dezoito no total.

A maioria das ocorrências do termo ainda se dá sem discussão, deixando de lado tensionamentos sobre o seu significado. Por exemplo, em Araujo, Andrade e Serey (2015) e Silva *et al.* (2016) temos:

> A comunidade de Educação em Computação acolheu a ideia inicial e nos últimos anos inúmeros trabalhos foram publicados. Esses versam sobre propostas de atividades, relatos de experiências e pesquisas empíricas sobre adoção e avaliação do **pensamento computacional** nos diversos níveis de ensino (Mannila et al, 2014) (Walden et al, 2013) (Barr e Stephenson, 2009). Outros investigam a respeito da definição formal para o PC (Selby e Woollard, 2014)(Hu, 2011). (ARAUJO; ANDRADE; SEREY, 2015, p.1455)

> O ensino de conceitos de lógica de programação e algoritmos através da robótica tem sido explorado por pesquisadores e educadores de diferentes formas. No entanto, pode ser notado no trabalho de Neto et al. (2015) que ainda é necessário maiores esforços, por parte de pesquisadores, para que se tenha um maior número de trabalhos focados no tema **Pensamento Computacional** e Robótica Pedagógica. Ainda existem relativamente poucas publicações científicas envolvendo estes conceitos. Neste sentido, o uso da robótica no âmbito educacional é mostrado a seguir através de algumas iniciativas recentes com contextos aproximados ao presente trabalho. (SILVA *et al.*, 2016, p.1189)

Araujo, Andrade e Serey (2015) apenas citam o termo a fim de definir a existência de, pelo menos, dois tipos de texto ligados à educação que têm alguma relação com pensamento computacional. Já Silva *et al.* (2016) usam o termo apenas para comentar a existência de poucas pesquisas que abordam diretamente pensamento computacional (temática do *workshop*) e robótica pedagógica (sua própria temática). Schoeffel *et al.* (2015), assim como os dois supracitados, usam pensamento computacional para se referir à busca de trabalhos correlatos:

> Com relação aos relatos de ensino de **pensamento computacional** e raciocínio lógico por meio da programação, diversos trabalhos foram encontrados, porém a maioria deles descreve ações isoladas e de curta duração. Com relação ao incentivo e participação na OBI, nenhum

> dos artigos encontrados descreve resultados de forma sistemática, além de não terem sido encontrados artigos relacionando um curso para o ensino de pensamento computacional com resultados na OBI, como é o caso deste trabalho. (SCHOEFFEL *et al.*, 2015, p.1477)

Esta forma de uso é importante para a escrita do texto, uma vez que não é preciso definir o que é pensamento computacional a todo momento. Mas sem um entendimento profundo do conceito, é difícil ter clareza de como estes textos trabalham com o pensamento computacional sem apelar para sua relação direta com a computação. É importante também reconhecer a hipótese de uma prática retórica na qual o pensamento computacional serve apenas de gancho com a temática do congresso. Além disso, pensamento computacional é um termo que, de forma rasa e junto com a temática da educação, serve como uma justificativa que em si, aparenta ser o suficiente para estas iniciativas.

A influência de Wing (2006) neste conjunto de textos também é grande, sendo citada em treze dos dezoito artigos presentes no grupo. Cinco destas citações reconhecem explicitamente Wing como a precursora ou a pessoa que introduziu o uso do termo pensamento computacional através de termos como “introduzido”, “proposto” ou “apresentado”:

> O termo Pensamento Computacional (em inglês Computational Thinking) foi introduzido por Wing (2008) [...] (ZANETTI; OLIVEIRA, 2015, p.1237)

> Pensamento Computacional (PC) foi apresentado em 2006 por Jeannette Wing associado às ideias de resolução de problemas [...] (ARAUJO; ANDRADE; SEREY, 2015, p.1454)

> Desde a proposta inicial de Jeanette Wing de classificar como Pensamento Computacional (PC) um conjunto de competências e habilidades do cientista da computação [...] (BARCELOS; BORTOLETTO; ANDRIOLI, 2016, p.1228)

> Em 2006, Wing sistematizou uma abordagem para resolução de problemas que combinou o pensamento crítico com os fundamentos da computação [...] (ZIMMERMANN *et al.*, 2016, p.1250)

Também notamos que o aumento de ocorrências do termo pensamento computacional em relação ao segundo grupo acompanha o aumento da sua importância para o texto, algo evidenciado pela quantidade de vezes (onze de dezoito textos) em que o termo aparece em algum subtítulo.

Nesta mesma linha, existe uma grande quantidade de textos (onze dos dezoito textos) que usam, na maioria das vezes, o termo como nome próprio ao escolher usar as primeiras letras em caixa alta (Pensamento Computacional). Este uso, intencional ou não, denota um entendimento que “Pensamento Computacional” seria algo preciso e específico (no caso desta

análise, algo que gira em torno das definições de Wing (2006)) e não um conjunto genérico e amplo de habilidades e/ou conceitos[46].

Como já colocado, o aumento da relevância do termo "pensamento computacional" não necessariamente é acompanhada de um aprofundamento em seu entendimento, como no caso das citações a Wing (2006), em que o significado do termo é tido como livre de disputa e suficiente para o argumento.

Por exemplo, em Zanetti e Oliveira (2015) ainda temos a visão universalista do que seria o "pensamento computacional", em especial quanto a sua validez para, inclusive, o "comportamento humano":

> O termo Pensamento Computacional (em inglês Computational Thinking) foi introduzido por Wing (2008), com o objetivo de englobar desde a estruturação do raciocínio lógico, até o comportamento humano para a ação de resolução de problemas. (ZANETTI; OLIVEIRA, 2015, p.1237)

Nesta mesma linha, mas com outros argumentos, ainda temos uma universalidade com relação a outros campos de conhecimento:

> O Pensamento Computacional traz um conjunto de processos cognitivos, técnicas e conceitos para resoluções de problemas que podem ser aplicados em várias áreas do campo de conhecimento. (FERNANDES; SILVEIRA, 2016, p.1070)

> O conceito de Pensamento Computacional se refere ao domínio de competências e habilidades da Computação que podem ser aplicadas à compreensão de conteúdos de outras áreas da ciência. A tecnologia computacional, que permeia nosso mundo e rotinas, é vista como importante recurso provedor de subsídios para que os indivíduos possam não apenas utilizar a tecnologia, mas também compreendê-la e serem capazes de implementar soluções para problemas utilizando recursos computacionais. (ZIMMERMANN *et al.*, 2016, p.1250)

> No Brasil, atualmente, esses conceitos são aprendidos somente por aqueles que optam por cursos técnicos ou de graduação na área da computação. Esse fato se contrapõe às exigências atuais do mercado de trabalho, onde grande parte das profissões exige uma compreensão da Ciência da Computação. Esse é o caso das áreas ligadas à arte e entretenimento, comunicação, saúde, entre outros. (ZIMMERMANN *et al.*, 2016, p.1250)

Além disso, ainda temos afirmações sem nenhum tipo de embasamento teórico ou empírico que contribuem pouco para a discussão:

> Pensamento Computacional (PC) é o processo de reconhecimento de aspectos da Ciência da Computação a partir da aplicação e utilização de ferramentas e técnicas de computação. Segundo Nunes (2011), a introdução desse conceito se justifica pelo seu caráter transversal, pois em um mundo cada vez mais globalizado é necessário dominar suas aplicações tornando o país mais rico e competitivo nas diversas áreas de aplicação da computação e da tecnologia da informação. (FERNANDES; SILVEIRA, 2016, p.1071)

[46] Para uma discussão similar sobre o uso da maiúscula, ver Burke (2008) e a questão do "Renascimento" e "renascimento".

> Vivemos em tempos marcados pela fluidez da informação e pela valorização do conhecimento. Mais do que nunca, para lidar com a informação, processá-la e transformá-la em aptidões para a vida, exige-se, em primeiro lugar, o domínio de uma série de ferramentas e recursos tecnológicos que devem ser acessíveis a todos, sem distinção de qualquer natureza. Nos tempos atuais, o desafio que se impõe aos usuários é criar seus próprios sistemas (por exemplo, programas e jogos) ou modificar os existentes de acordo com sua necessidade pessoal. É neste contexto que surge a habilidade que é vista como crucial no século 21: o Pensamento Computacional (PC) (Kologeski et al., 2016). Devido a essa tendência mundial, o PC vem sendo adotado em escolas da educação básica em diversos países (Brackmann et al., 2016). (BRACKMANN *et al.*, 2017, p.982)

Vale a pena citar o caso de Paiva *et al.* (2015), que faz uma reflexão mais aprofundada, questionando o uso do termo "pensamento" e escolhendo trabalhar com outro, "raciocínio", além não se prender às frases de efeito encontradas em Wing (2006). Entretanto, ainda assim, temos uma passagem que considera "pensamento computacional" um conceito e reclama de sua redução a um senso comum:

> Como podemos ver, a autora apresenta o que chama de computational thinking, como um conjunto de ferramentas mentais usado para raciocínio heurístico (no cotidiano, para além dos cientistas). A tradução livre de interpretação, reduz o conceito explicado por Wing ao senso comum. (PAIVA *et al.*, 2015, p.1302)

Também encontramos usos do "pensamento computacional" ora como um conjunto ou uma habilidade a ser desenvolvida de maneira relativamente alinhada com Wing (2006):

> Resumo. A robótica educativa visa aproximar o que se ensina à realidade do aluno utilizando atividades baseadas em problemas. Contudo, o ensino é demasiadamente dificultado diante da deficiência na formação docente e discente em competências do Pensamento Computacional (PC). (SOUZA; RODRIGUES; ANDRADE, 2016, p.1265)

> Associado a este movimento em prol da aprendizagem de linguagens de programação propriamente ditas, observa-se também o esforço protagonizado por pesquisadores, educadores e organizações de diversos tipos para fomentar um conjunto de competências, habilidades e disposições relacionadas ao universo da informática, as quais dão forma ao conceito de Pensamento Computacional [Blikstein, 2013]. (KAMPFF *et al.*, 2016, p.1316)

> Pensamento Computacional (PC) diz respeito à resolução de problemas baseada em conceitos da computação e que podem ser utilizado em diversas atividades cotidianas. Sobre este aspecto, é preciso compreender como as habilidades relacionadas ao PC podem ser disseminadas no ambiente educacional pelas diversas disciplinas curriculares dos mais diversos cursos de formação.(GERALDES *et al.*, 2017, p.902)

> Da mesma forma que em outros contextos do processo de ensino-aprendizagem, avaliar adequadamente o desenvolvimento de competências e habilidades relacionadas ao Pensamento Computacional se constitui como um desafio. A avaliação da aprendizagem em contextos experimentais tem sido frequentemente realizada com a utilização de técnicas de pesquisa relacionadas ao paradigma qualitativo, tais como a observação etnográfica e as entrevistas, bem como testes e questionários específicos (ARAÚJO; ANDRADE; GUERRERO, 2016). Por outro lado, outras iniciativas para assegurar o desenvolvimento de um conjunto mínimo de competências e habilidades incluem a definição de currículos de referência (CSTA, 2011) e rubricas educacionais para contextos mais específicos

> (MORENO-LEÓN; ROBLES, 2015). (BARCELOS *et al.*, 2017b, p.932)
>
> Na literatura é raro encontrar pesquisas a respeito da aplicação e avaliação de estudantes de maneira desplugada. Para preencher esse vazio, o presente trabalho apresenta uma pesquisa realizada em duas escolas primárias na Espanha para avaliar se as crianças têm uma alteração no desempenho das habilidades relacionadas ao Pensamento Computacional através de atividades sem computadores. Sendo assim, esta pesquisa tem como objetivo verificar se aulas de Pensamento Computacional Desplugado na educação primária são eficazes através da aplicação de um pré e pós teste, realizados antes e após as aulas de PC Desplugado. (BRACKMANN *et al.*, 2017, p.983)

Já em Zanetti e Oliveira (2015), Schoeffel *et al.* (2015), Ortiz e Pereira (2017) e Pinheiro, Franco e Leite (2017), além do alinhamento com Wing (2006), existe um presuposto de que o pensamento computacional pode ser desenvolvido ao utilizar as atividades normais da computação. Isto reforça que ao trazer o conceito para a o dia a dia, é um pouco mais difícil ignorar a dimensão material das práticas históricas do campo.

> Disciplinas como a programação de computadores ou Lógica de Programação permite realizar um encadeamento coerente de uma sequência lógica de instruções para resolução de problemas, constituindo um dos saberes elementares para desenvolver o Pensamento Computacional. (ZANETTI; OLIVEIRA, 2015, p.1238)
>
> Um dos objetivos de ensinar programação para crianças é o desenvolvimento do Pensamento Computacional, que caracteriza-se pelo uso de princípios da computação para ajudar a separar elementos de um problema em outras áreas, determinar seus relacionamentos e desenvolver passos lógicos para chegar a soluções automatizadas. (KAFAI; BURKE, 2013). (SCHOEFFEL *et al.*, 2015, p.1475)
>
> Recentemente, diversas iniciativas têm surgido no Brasil com o intuito de ensinar algoritmos, programação e outros fundamentos computacionais para desenvolver habilidades e aptidões relacionadas ao pensamento computacional. Públicos bastante visados para estas pesquisas são os alunos do próprio ensino superior em Ciência da Computação, alunos dos ensinos fundamental e médio, e professores. O público da educação de jovens e adultos (EJA), entretanto, não tem sido contemplado por estas iniciativas, sendo um contexto passível de exploração e com características diferenciadas, pois possuem diversos contextos sociais, culturais e econômicos desafiadores, muitas vezes gerados pelo abandono dos estudos. (ORTIZ; PEREIRA, 2017, p.1069-1070)
>
> Ainda nesta perspectiva, Cavalcante e Costa [2016] trazem contribuições voltadas às competências utilizadas em um curso de programação, que propiciam o desenvolvimento do Pensamento Computacional. Como elemento de investigação, os autores utilizam um framework responsável por avaliar a eficácia de ambientes de programação em blocos no desenvolvimento do Pensamento Computacional. Nos resultados obtidos, os autores apresentam a eficiência do framework e as práticas elencadas no ambiente relacionadas ao Pensamento Computacional não reconhecidas pelo framework. Por fim, os autores propõem a análise de outros cursos e outros frameworks relacionados ao Pensamento Computacional. França e Tedesco [2015] corroboram os ideais de Cavalcante e Costa quando discutem os desafios que circundam o ensino do Pensamento Computacional na educação básica brasileira; evidenciam diferentes pesquisas na área da Computação sobre o Pensamento Computacional, além de apresentar abordagens que possibilitam a utilização da programação enquanto meio de desenvolver o Pensamento Computacional no ensino médio, trazendo como referencial pesquisas desenvolvidas em diferentes estados do Brasil. (PINHEIRO; FRANCO; LEITE,

> 2017, p.1114)

Além disso, também encontramos um entendimento de que o pensamento computacional é um conceito a ser compreendido ou usado, não necessariamente uma habilidade a ser desenvolvida:

> Pensamento Computacional é uma abordagem focada na resolução de problemas explorando processos cognitivos, técnicas e ferramentas comuns na Ciência da Computação. Embora o termo faça uma alusão direta à Computação, nem todos os processos e técnicas associados são exclusivos da Computação. Mesmo assim, eles foram incorporados e são amplamente explorados no contexto de buscar soluções mais eficientes para resolver problemas ou atividades complexas. (ARAUJO; ANDRADE; SEREY, 2015, p.1456)

> O objetivo da abordagem é de estimular o Pensamento Computacional nos estudantes envolvidos nas atividades. Como diferencial, a metodologia proposta considera a progressão dos alunos entre as tecnologias Lego Mindstorms e Arduino. Além disso, a abordagem leva em consideração a utilização de poucas unidades de kits de robótica para se adequar a realidade das escolas públicas brasileiras. (SILVA *et al.*, 2016, p.1188)

> No contexto das discussões sobre a incorporação do Pensamento Computacional na educação básica um dos principais desafios atuais é a adequada formação de professores para desenvolver atividades relacionadas. Neste artigo apresentamos um curso para formação inicial e continuada de professores de Matemática, baseado na construção de jogos digitais e oferecido em uma plataforma online, visando capacitá-los a criar atividades que envolvam tópicos matemáticos juntamente com o desenvolvimento de competências do Pensamento Computacional. Os resultados preliminares indicam que os participantes atingiram um nível avançado de competência em alguns tópicos do Pensamento Computacional e tiveram maior interesse por atividades diretamente relacionadas com a construção de jogos. (BARCELOS; BORTOLETTO; ANDRIOLI, 2016, p.1228)

> O conceito de Pensamento Computacional se refere ao domínio de competências e habilidades da Computação que podem ser aplicadas à compreensão de conteúdos de outras áreas da ciência. (ZIMMERMANN *et al.*, 2016, p.1250)

> O conceito de Pensamento Computacional é apresentado segundo a definição de Wing (2006 e 2008). Em seguida, são descritos os objetivos da oficina, os tópicos abordados e a atividade prática realizada com os participantes, baseada em um framework visual de organização de procedimentos para a solução de problemas, criado especialmente para a ocasião do curso. Os resultados da oficina, observados com base em uma pesquisa com os participantes, são apresentados na última seção. (KAMPFF *et al.*, 2016, p.1316)

> A metodologia empregada reúne conceitos do Pensamento Computacional, assim como a realização de atividades com Aprendizagem Baseada em Problemas (ABP) e Práticas Colaborativas. Segundo Andrade et al. (2013), pensamento computacional é um processo de raciocínio que se origina da combinação do pensamento crítico com os fundamentos da computação envolvido na formulação de problemas e das suas soluções, estando diretamente vinculada à Aprendizagem Baseada em Problemas. (BAUER *et al.*, 2017, p.1212)

Tanto como uma habilidade a ser desenvolvida, quanto como um conceito, pouco é dito sobre como seriam as práticas associadas ao pensamento computacional. Esta falta de concretude

torna o pensamento computacional tanto uma habilidade que é necessária para programar ao mesmo tempo que é desenvolvida ao programar.

A Tabela 6 mostra a quantidade de ocorrências do termo "pensamento computacional", a referência e o título dos artigos publicados no WalgProg em 2015, 2016 e 2017 com onze até trinta ocorrências.

Tabela 6 – Textos do WAlgProg (2015-2017) que usam o termo "pensamento computacional" entre onze e trinta vezes

Quant.	Autor	Título
18	Zanetti e Oliveira (2015)	Prática de ensino de Programação de Computadores com Robótica Pedagógica e aplicação de Pensamento Computacional
14	Mestre *et al.* (2015)	Pensamento Computacional: Um estudo empírico sobre as questões de matemática do PISA
13	Paiva *et al.* (2015)	Uma Experiência Piloto de Integração Curricular do Raciocínio Computacional na Educação Básica
17	Araujo, Andrade e Serey (2015)	Pensamento Computacional sob a visão dos profissionais da computação: uma discussão sobre conceitos e habilidades
17	Schoeffel *et al.* (2015)	Uma Experiência no Ensino de Pensamento Computacional e Fomento à Participação na Olimpíada Brasileira de Informática com Alunos do Ensino Fundamental
19	Fernandes e Silveira (2016)	Pensamento computacional: iniciativas para o seu desenvolvimento por meio da modalidade de ensino a distância
13	Silva *et al.* (2016)	Aplicação de Robótica na Educação de Forma Gradual para o Estímulo do Pensamento Computacional
22	Barcelos, Bortoletto e Andrioli (2016)	Formação online para o desenvolvimento do Pensamento Computacional em professores de Matemática
22	Zimmermann *et al.* (2016)	Proposta de Aplicação e Avaliação de Conceitos do Pensamento Computacional em Crianças Hospitalizadas
18	Souza, Rodrigues e Andrade (2016)	Introdução do Pensamento Computacional na Formação Docente para Ensino de Robótica Educacional
30	Kampff *et al.* (2016)	Pensamento Computacional no Ensino Superior: Relato de uma oficina com professores da Universidade do Vale do Rio dos Sinos
18	Geraldes *et al.* (2017)	O Pensamento Computacional no Ensino Profissional e Tecnológico
17	Barcelos *et al.* (2017b)	Mensurando o desenvolvimento do PensamentoComputacional por meio de Mapas Auto-Organizáveis: um estudo preliminar em uma Oficina de Jogos Digitais
22	Brackmann *et al.* (2017)	Pensamento Computacional Desplugado: Ensino e Avaliação na Educação Primária da Espanha
25	Ortiz e Pereira (2017)	Pensamento Computacional na Educação de Jovens e Adultos: desafios e oportunidades
22	Pinheiro, Franco e Leite (2017)	Desenvolvimento do Pensamento Computacional e discussões sobre representação feminina na Computação: um estudo de caso
13	Bauer *et al.* (2017)	Projeto codIFic@r: Oficinas de Programação em Dispositivos Móveis no Ensino Fundamental

Fonte: Autoria própria

2.2.3 Artigos que citam mais que trinta vezes “pensamento computacional”

Como esperado, no último grupo em que estão os textos em que “pensamento computacional” aparece mais de trinta vezes, o termo é central para o argumento dos artigos, uma vez que aparece em diversos momentos nos textos.

Neste grupo, mesmo que a definição de Wing (2006), Wing (2008) tenham menos relevância dada a densidade das discussões propostas (ainda assim é citada com uma grande frequência), sua essência ainda se faz presente. Por exemplo, quanto à necessidade de universalização do “pensamento computacional”, em Barcelos *et al.* (2015) temos logo no começo do texto:

> A comunidade de pesquisa em Informática na Educação tem considerado que a Ciência da Computação deveria fazer parte do currículo escolar desde as séries iniciais, sendo assim posicionada no mesmo nível das chamadas “ciências basicas”, como a Física, a Biologia e a Química. (BARCELOS *et al.*, 2015, p.1370)

Algo parecido também existe em Farias, Andrade e Alencar (2015):

> A comunidade acadêmica está convergindo cada vez mais para a primordial importância da ampliação do ensino de conceitos da Computação, contemplando alunos desde a tenra idade, tornando assim, uma ciência basilar para a formação de habilidades primárias para bom desempenho de qualquer profissão. (FARIAS; ANDRADE; ALENCAR, 2015, p.1226)

E também em França e Tedesco (2015):

> Com o rápido crescimento computacional e tecnológico ocorrido no mundo moderno surge a necessidade de ensinar, desde a educação básica, conceitos fundamentais da Ciência da Computação como forma de melhorar o aprendizado escolar dos indivíduos e possibilitar o uso mais eficaz dessas tecnologias em benefício da sociedade. (FRANçA; TEDESCO, 2015, p.1464)

Nos dois primeiros textos, existem a referência a um terceiro indefinido presente nas figuras da “comunidade acadêmica”, “comunidade de pesquisa”. Ambas servem como um elemento retórico para justificar um certo consenso científico. Já a necessidade de ensinar conceitos ligados à computação no começo da vida escolar junto com a demanda por equiparação com as “ciências básicas” ou entender a Computação (com ‘c’ maiúsculo) como “ciência basilar” ou “como forma de melhorar o aprendizado escolar” são instâncias diretas da demanda por universalizar o “pensamento computacinal” encontrado em Wing (2006), Wing (2008).

Neste grupo, também encontramos mais uma outra forma de trabalhar esta questão. Em Cavalcante, Costa e Araujo (2016), temos uma visão de que “pensamento computacional” é tão ou até mais importante que os conhecimentos da própria área:

> É uma forma de aplicar conceitos trabalhados na computação, mas que não são exclusivos somente desta área, pois podem ser aplicados na resolução de problemas dos mais variados campos do saber. A aplicabilidade transversal e multidisciplinar ressignifica o "pensar computacionalmente" como uma competência fundamental para todas as pessoas, não apenas para profissionais da computação, despontando como um requisito elementar para a formação básica dos profissionais de todas as áreas nos próximos anos. (CAVALCANTE; COSTA; ARAUJO, 2016, p.1117)

Em Leite *et al.* (2017), existe uma passagem com a mesma função:

> O Pensamento Computacional está centrado na perspectiva da resolução de problemas mediados ou não pelo uso computador. Essa abordagem se expande por todos os contextos na busca de soluções mais eficientes para resolver problemas e atividades complexas. (LEITE *et al.*, 2017, p.1004)

Também nesta mesma linha, Costa *et al.* (2017) cita diretamente Wing (2006):

> Wing (2006) argumenta ainda que a sua utilização (pensamento computacional) direciona a solucionar problemas nos mais diversos campos do conhecimento, não sendo exclusiva para profissionais da computação. (COSTA *et al.*, 2017, p.993).

Dentro do WAlgProg, o texto de Wing (2006) tem a função de validar uma espécie de senso comum sobre o que é o "pensamento computacional". Como colocado anteriormente, Wing (2006), Wing (2008) não têm a preocupação em definir de forma específica o que é o "pensamento computacional", dando apenas pistas, no mínimo genéricas, do que ele pode ou poderia ser. Já os textos apresentados mostram que os trabalhos que se apoiam na definição de Wing (2006), Wing (2008) também não têm esta preocupação.

É interessante notar que apesar dos textos do quarto grupo terem como foco o "pensamento computacional", somente dois (França e Tedesco (2015) e Leite *et al.* (2017)) se apoiam em Wing (2006), Wing (2008) de forma concreta para desenvolver suas pesquisas.

Além da universalidade já colocada anteriormente, França e Tedesco (2015) também apresentam um trecho em que é possível entender que o "pensamento computacional" é uma dimensão abstrata. Por exemplo, ao comentar quando o que devem ser ensinados na educação básica, França e Tedesco (2015) afirmam que: "...deve levar em consideração que o ensino do pensamento computacional não deve cobrir apenas a manipulação de recursos digitais, mas sim os fundamentos da Computação, enquanto ciência." (FRANçA; TEDESCO, 2015). É importante chamar a atenção para o uso do "mas sim" de forma a dar entender que o manipular não dá conta do que deve ser ensinado na educação básica.

Leite *et al.* (2017), por sua vez, também carregam uma percepção de que "pensamento computacional" é mais do que o uso de *software*, como na seguinte passagem, uma paráfrase de Gomes e Melo (2013):

> É importante lembrar que o Pensamento Computacional não se fundamenta em saber navegar na Internet, acessar e-mails, editar um texto, utilizar planilhas eletrônicas, elaborar uma apresentação ou manipular um equipamento eletrônico. Sua importância está para o processo de resolução de problemas lógicos nos diversos contextos da sociedade, permitindo que se possa aplicar a Computação nas suas ações cotidianas. (LEITE *et al.*, 2017, p.1004)

Além dessa, existe uma outra passagem sobre o uso de recursos computacionais que contém um desdém por práticas de outros campos do saber:

> Os resultados encontrados apontam evidências de atividades abordadas com pouca complexidade, tais como ler notícias, copiar conteúdos, visualizar mapas, desenhar, escrever com editores de texto, calcular com calculadora. A partir desses dados é possível afirmar que a forma como a tecnologia foi explorada por estas escolas não satisfaz a perspectiva do Pensamento Computacional. (LEITE *et al.*, 2017, p.1007)

Caracterizar todos estes processos como pouco complexos e logo após colocar que este tipo de uso do computador não satisfaz "a perspectiva do Pensamento Computacional" ignora que desenhar, para apenas ficar em um caso que reflete o simplismo da afirmação, é um processo de abstração quase que por definição. Ao desenhar um objeto, é preciso escolher que características devem ser usadas na representação. Inclusive, em Kramer (2007) temos o reconhecimento disto. Neste artigo publicado na *Communications of the ACM*, ele cita obras do artista Henri Matisse (1869 - 1954) e do gravurista/pintor Katsushika Hokusai (1760 - 1849) como exemplos de abstrações.

Barcelos *et al.* (2015) faz uma revisão sistemática de literatura sobre pensamento computacional com o objetivo de verificar publicações que trabalham com o conceito junto com conteúdos da matemática. Mesmo com diversas ocorrências do termo, Barcelos *et al.* (2015) limitam a definir o pensamento computacional de acordo com a noção mais genérica presente em Wing (2006): "O Pensamento Computacional representa um conjunto de habilidades relacionadas à Ciência da Computação que deveriam ser desenvolvidas pelos estudantes da Educação Básica".

Araujo, Andrade e Guerrero (2016) fazem um mapeamento sistemático sobre publicações que lidam com avaliação e pensamento computacional e reconhecem o trabalho de Wing (2006) como um ponto importante para a reflexão sobre pensamento computacional nas escolas. Este trabalho identifica o uso de programação como principal articulador de iniciativas de pensamento computacional.

A Tabela 7 mostra a quantidade de ocorrências do termo "pensamento computacional", a referência e o título dos artigos publicados no WalgProg em 2015, 2016 e 2017 com trinta ou mais ocorrências.

Tabela 7 – Textos do WAlgProg que usam o termo "pensamento computacional" mais que trinta vezes

Quant.	Citação	Título
42	Farias, Andrade e Alencar (2015)	Pensamento Computacional em Sala de Aula: Desafios, Possibilidades e a Formação Docente
41	França e Tedesco (2015)	Desafios e oportunidades ao ensino do pensamento computacional na educação básica no Brasil
31	Barcelos *et al.* (2015)	Relações entre o Pensamento Computacional e a Matemática: uma Revisão Sistemática da Literatura
32	Cavalcante, Costa e Araujo (2016)	Um Estudo de Caso Sobre Competências do Pensamento Computacional Estimuladas na Programação em Blocos no Code.Org
83	Araujo, Andrade e Guerrero (2016)	Um Mapeamento Sistemático sobre a Avaliação do Pensamento Computacional no Brasil
56	Costa *et al.* (2017)	Um Estudo Exploratório da Aplicação de Pensamento Computacional Baseado nas Perspectivas de Professores do Ensino Médio
35	Leite *et al.* (2017)	Pensamento Computacional nas Escolas: Limitado pela Tecnologia, Infraestrutura ou Prática Docente?
33	Raabe *et al.* (2017)	Um Instrumento para Diagnóstico do Pensamento Computacional

Fonte: Autoria própria

2.3 A CATÁBASE DO PENSAMENTO COMPUTACIONAL

A dissonância que o conceito de pensamento computacional possui ao olhar estes textos como um todo pode ser explicada por aquilo que Vieira Pinto (2005a) chama de Catábase da Técnica.

Catábase é usada por Vieira Pinto com o sentido da "descida ao inferno" ou "das lutas que a humanidade trava no plano da história real" (VIEIRA PINTO, 2005a, p.294). Ele afirma que esta "descida" só será conseguida por um "espírito de concreticidade", oposto a um "espírito de abstração". Para o autor, o conceito de Catábase da Técnica revela, no caso da técnica e da tecnologia, uma "exigência de compreender o conceito abstrato não em si mesmo, mas nas manifestações singulares e concretas"(VIEIRA PINTO, 2005a, p.294).

Ou seja, Vieira Pinto está chamando a atenção para a dependência de contexto que técnicas e tecnologias possuem, mas que é ignorada em um plano abstrato/conceitual. Ele coloca ainda, que "não podem [as técnicas] pois ser entendidas fora do plano de existência material da espécie humana, porque não teriam então qualquer significado [...]" (VIEIRA PINTO, 2005a, p.294). De uma forma mais precisa, ele coloca que técnicas aparentemente iguais podem ser essencialmente diferentes, dependendo, principalmente, das finalidades do indivíduos ou grupos sociais a que servem (VIEIRA PINTO, 2005a, p.295).

Este é ponto de entrada de sua crítica à prática científica na periferia e no centro[47]. Ele começa argumentando que cientistas especialistas de certas áreas do conhecimento, mesmo que bem intencionados, emitem opiniões sobre "tecnologia" sem um entendimento correto dessa. Ele coloca que estes especialistas "anseiam por lançar-se em voos de pensamento generalizado, filosófico, e passam então, com a maior cantadura, a emitir opiniões idealmente abstratas, sempre recebidas com o respeito merecido, pela dignidade e reputação dos autores, mas nem por isso menos representativas de formas imaturas, ingênuas de compreensão" (VIEIRA PINTO, 2005a, p.295).

Este parece ser o caso de Wing (2006). Mesmo que tenhamos dedicado uma seção inteira para crítica do seu conceito, é preciso reconhecer que o contexto no qual ela escreve o texto, a ênfase no técnico/operacional, muitas vezes com o protagonismo das linguagens de programação, precisava ser problematizada. Mesmo que bem intencionada, (WING, 2006) é uma especialista (acreditamos que bem intencionada) que emitiu uma opinião sobre tecnologia e, pelo que mostra os textos do WAlgProg, foi recebida com respeito, mas ainda assim imaturas e igênuas, de acordo com Vieira Pinto (2005a).

Neste ponto, vale a pena lembrar que, para Vieira Pinto, "o pensamento, divorciado da prática, acaba ignorando a razão de ser desta, chegando ao ponto de divinizá-la sob o nome de "tecnologia" '(VIEIRA PINTO, 2005a, p.294). Um dos efeitos desta separação é um apagamento de um exame das técnicas "de acordo com o contexto social onde surge ou onde se aplica" (VIEIRA PINTO, 2005a, p.295).

Nos textos do WAlgProg apresentados, encontramos estas duas questões. Primeiro, quanto à "divinização" da tecnologia, ela aparece diversas vezes sob a forma de um determinismo tecnológico, definido por Dagnino (2008, p.51) como "uma visão evolucionária linear, alimentada pela força da eficiência, que se apresenta como objetiva, neutra e livre de qualquer intervenção social". Em outras palavras, uma visão em que aquilo que chamamos coloquialmente de tecnologia seria alheia às forças sociais, logo neutra, pois não tem em si nenhuma valoração social. A tecnologia também teria uma lógica de desenvolvimento própria em que um estágio supera o anterior por completo, medida por sua eficiência em algum processo específico e com o poder de definir e regular os processos da sociedade como um todo. Por exemplo, tanto em

[47] Centro e periferia são usados aqui como conceitos cepalianos, que, para Freitas (2005, p.4), eram usadas tanto pelo CEPAL quanto Vieira Pinto para "descrever uma situação assimétrica na apropriação de ganhos originados na divisão internacional do trabalho".

Pinheiro, Franco e Leite (2017) quanto em Farias, Andrade e Alencar (2015) podemos perceber claramente esta visão:

> No atual contexto da sociedade, as novas tecnologias de comunicação e informação (NTIC) são fundamentais ao desenvolvimento de novas ideias, práticas e metodologias. (PINHEIRO; FRANCO; LEITE, 2017).

> Esta ciência está imersa na participação social, corroborando na compreensão de conhecimentos complexos, levando a humanidade para patamares de saber muito mais refinados e superiores do que se dominavam em décadas e séculos passados. (FARIAS; ANDRADE; ALENCAR, 2015).

Um estudo de Bijker *et al.* (2012) sobre as mudanças tecnológicas no projeto de bicicletas na segunda metade do século XIX e na primeira do século XX evidencia que a questão é mais complexa. Bijker *et al.* (2012) mostra que diversas mudanças no funcionamento das bicicletas (um tipo de tecnologia) foram profundamente influenciadas por grupos sociais junto com as percepções desses do que seria uma bicicleta. Este estudo coloca em xeque o entendimento da tecnologia como uma força independente da sociedade, inclusive mostrando que a eficiência como métrica de desenvolvimento tecnológico é muito mais flexível do que o determinismo parece considerar.

Para Smith e Marx (1994), artistas, publicitários e historiadores profissionais seriam os responsáveis por popularizar a crença da tecnologia como motor da sociedade, mais especificamente a sociedade estadunidense pós século XVIII. Neste contexto em especial, também ocorre a associação de tecnologia com uma ideia abstrata de progresso. Smith e Marx (1994, p.8) colocam que não era incomum encontrar em certos escritores uma tendência de ver "novas tecnologias tanto como instrumentos de força ao mesmo tempo como símbolos triunfantes do progresso humano". Em Farias, Andrade e Alencar (2015) é possível encontrar uma visão muito similar:

> Esta ciência está imersa na participação social, corroborando na compreensão de conhecimentos complexos, **levando a humanidade para patamares de saber muito mais refinados e superiores do que se dominavam em décadas e séculos passados**. (FARIAS; ANDRADE; ALENCAR, 2015).

O trecho destacado na citação de Farias, Andrade e Alencar (2015) implica no entendimento de que qualquer tipo de processo científico dentro da Ciência da Computação é benéfico para a "humanidade", não existindo nenhum tipo de retrocesso em seu desenvolvimento.

Uma outra forma que a questão do progresso se apresenta nos textos é por um certo ufanismo com relação à Computação. Nos casos apresentados abaixo, ela é entendida como um tipo de conhecimento que não é limitado pelo seu próprio campo de atuação:

> Em escolas públicas, por exemplo, com a inserção da sala de informática os alunos têm a possibilidade de englobar, através do auxílio do computador, diversas matérias em diferentes áreas do conhecimento de uma forma mais ampla, dinâmica e interdisciplinar, o que torna o computador uma ferramenta importante para a construção do conhecimento dos alunos. (LIMA; SOUSA, 2015a).

> A Ciência da Computação é uma área de conhecimento que pelo seu inerente caráter universal e mediatizador, permite um leque de possibilidades para a realização de conexões interdisciplinares, tanto para a sua evolução enquanto ciência, quanto para a sua aplicação na resolução de problemas mais complexos. (PAIVA *et al.*, 2015).

Tanto em Lima e Sousa (2015a) quanto Paiva *et al.* (2015), a computação seria universal e interdisciplinar, servindo como uma espécie de cola entre disciplinas diferentes. A constatação da ubiquidade da informática/computação, nos dois casos, é associada a movimentos necessariamente positivos. Lima e Sousa (2015a) usam termos como "mais ampla" e "dinâmica" para descrever a associação entre computação e outras disciplinas, enquanto para Paiva *et al.* (2015), ela serviria para resolver problemas "mais complexos" e evoluir "como ciência". Além disso, em Lima e Sousa (2015a) temos o uso do "diversas" como uma forma de não identificar quais seriam estas disciplinas que se beneficiam tanto com o uso do computador.

Sobre a segunda questão, Vieira Pinto (2005a) também coloca em foco os mecanismos de recepção destas ideias, as quais ganham uma certa validade dada pela autoridade científica destes pesquisadores. "A autoridade científica destes mestres, entre os quais se contam às vezes os maiores gênios de nosso tempo, obriga a levar em conta a sua palavra, recebida quase sempre com acatamento mais de modo acrítico" (VIEIRA PINTO, 2005a, p.295).

Novamente, argumentamos que o WAlgProg mostra indícios desta recepção sem uma crítica, em especial no uso do "pesamento computacional" como se fosse um conceito bem definido, acabado e aceito dentro da comunidade.

Ainda sobre o papel de cientista e técnicos, Vieira Pinto coloca que se esses não entenderem o seu papel na transformação geral das estruturas de sua sociedade, correm o risco de "divinizar a tecnologia do país rico, acreditando que a simples transferência dela para o meio atrasado modificará as condições existenciais da realidade deste último"(VIEIRA PINTO, 2005a, p.299). Se entendermos que os conceitos ligados à produção tecnológica faz parte do que se chama de tecnologia, a forma como o "pensamento computacional" é usado no WalgProg mostra exatamente o que Vieira Pinto (2005a) está a colocar.

No caso da Educação e Computação, podemos citar dois desdobramentos. Primeiro, quanto a uma igualdade do tratamento da Ciência da Computação com relação a outras disciplinas do Ensino Básico, tais como Matemática, História e Português. Independente de julgar o mérito

da questão, nos textos do WAlgProg não parece existir muita preocupação com as questões relativas ao contexto social local. Admite-se o argumento de Wing (2006) como válido de antemão a suas considerações contextuais.

Em segundo lugar, temos as iniciativas que acreditam que a simples presença do computador como artefato é o suficiente. Dada a forma como o pensamento computacional é trabalhado nos textos analisados, esta visão não é muito proeminente. Entretanto, a grande quantidade de artefatos propostos como encaminhamento merece ser mencionado.

Dando continuidade ao raciocínio, Vieira Pinto acredita que somente a importação de técnicos e tecnologia não é o suficiente. É preciso remodelar a qualidade de trabalho das massas do país em sua totalidade. Ele admite que o contato com bens e técnicas estrangeiras traz uma "qualificação" para o corpo igualmente competente da colônia, mas especula que com isso, o país "estará criando perigosamente "metrópoles" internas e aumentando desnível das condições de vida do povo" (VIEIRA PINTO, 2005a, p.299).

Vieira Pinto concluiu a obra que inclui este trecho durante a década de 70 do século XX, antes de diversas aberturas econômicas feitas nas décadas seguintes e a piora dos índices de concentração de renda dos últimos anos. No caso brasileiro, vale a pena citar a questão dos "unicórnios" e *startups* que surgiram nos últimos anos, destino de diversos formados nos cursos relacionados à Ciência da Computação.

Uma vez que este recorte de "usos" do pensamento computacional não tem uma unidade nem uma profundidade quanto ao conceito, na próxima seção faremos um pequeno resgate histórico para tentar qualificar um pouco melhor o que seria este pensamento computacional.

3 AS CONCRETUDES DAS ABSTRAÇÕES

Este capítulo tem como objetivo apresentar questões ligadas ao ensino e aprendizagem de programação de computadores articuladas com teorias ligadas a uma perpectiva materialista como uma forma de contrapor a perpectiva que parece existir no que foi apresentado no capítulo anterior. A análise dos textos publicados no WAlgProg sugere a existência de uma ligação forte entre "pensamento computacional" e abstração, mas a relação de ambas com o formalismo matemático acaba por ficar subentendida. Para reforçar que o uso do termo "pensamento computacional" é uma forma de reduzir o processo complexo de abstração a somente a aplicação do formalismo matemático, fazemos um breve resgate histórico para evidenciar as relações que esta corrente matemática tem com o campo da computação.

Uma vez qualificada a forma como é encarado o processo de abstração, começamos a construção do objeto de pesquisa ao redefinir o que é abstração. Trabalhamos de um modo para que este seja um conceito frutífero para iniciativas dentro do ensino e aprendizagem de programação de computadores. Para tanto, trazemos conceitos de autores mais ligados a uma corrente materialista da filosofia, tais como Althusser (2019) e Lefebvre (1991).

Articular o conceito de abstração a partir de Althusser (2019) e Lefebvre (1991) para trabalhar com ensino e aprendizagem exige também discutir o processo social envolvido. Para esta dimensão em especial, usamos o conceito de dialogismo dos autores ligados ao Círculo de Bakhtin Bakhtin (2017 [1919/1921]) e considerações da teoria de Vigotski (2004b), em especial seus conceitos que dão um papel mais específico neste processo.

Propomos usar um conceito mais amplo do que "pensamento computacional", algo que chamamos por hora de "fazeres computacionais". "Fazeres" em contraste ao conceito de pensamento fortemente ligado a uma perspectiva mentalista e "computacionais", no plural rebatendo o universalismo que o "computacional" do "pensamento computacional" representa, algo forte na retórica que gira ao seu redor.

Por fim, apresentamos um resumo do que seriam os paradigmas de programação, conceito muito usado para discutir ensino e aprendizagem de programação. Entendemos que o termo paradigma é bem difundido como uma espécie de sinônimo de uma forma de programar e acaba por funcionar, de modo meio informal, como uma forma de classificação de "estilos de programação". O relatório feito para o projeto *Computing Curricula 2001* (CURRICULA, 2001), inclusive, usa paradigma (de forma relaxada, inclusive alternando com *approach*) para se referir à programação orientada a objetos e à imperativa.

Já a versão de 2013 do *Computing Curricula* faz referência a versão de 2001 ao colocar que metade dos programas introdutórios tinham como base um paradigma de programação (Imperative-first, Objects-first, Functional-first) e ainda comenta a presença de outros paradigmas (scripting vs. procedural programming or functional vs. object-oriented programming) como uma forma de ampliar perpectivas sobre programação dos estudantes (CURRICULA; SOCIETY, 2013).

Entretanto, também argumentamos que a forma que o conceito é articulado é um desdobramento da perspectiva formalista que não necessariamente leva em conta as pessoas no processo de ensino e aprendizagem.

3.1 O ABSTRATO COMO HERANÇA DA MATEMÁTICA

A ciência da computação como campo de conhecimento tem uma forte relação com a matemática. A título de ilustração, temos um curso aberto oferecido pelo MIT, chamado "Matemática para Ciência da Computação[1]", o qual possui como tópicos, conhecimentos que são bem divididos pelos dois campos, tais como provas matemáticas, probabilidade e contagem. Estes são tópicos comuns associados à disciplina de Matemática discreta, bem presente no campo da Ciência da Computação.

Além disso, Fonseca Filho (2007, p.22) também lembra que "pelo menos nos círculos acadêmicos, a Computação apareceu como algo dentro dos Departamentos de Matemática, e ainda hoje, em muitas Universidades, a Ciência da Computação aparece como um Departamento de um Instituto de Matemática". Complementamos que mesmo como um departamento autônomo, o departamento de informática ou computação dificilmente está fora da área das exatas.

[1] *Mathematics for Computer Science.*

Esta relação também emerge com um olhar histórico sobre a computação. Das diversas perspectivas historiográficas possíveis[2], duas, em especial, ajudam a qualificar o que seria uma herança matemática na computação e, mais importante para este texto, suas implicações para o ensino e aprendizagem de programação de computadores. A primeira, entende que a história da computação é o "desenvolvimento dos conceitos teóricos que formaram a base para o surgimento da Computação" (FONSECA FILHO, 2007, p.14). Fonseca Filho (2007) aborda "as ideias e conceitos relevantes que fundamentaram a incansável busca pela mecanização do raciocínio" (FONSECA FILHO, 2007, p.21) em uma perspectiva teleológica.

A segunda perspectiva é o desenvolvimento material dos artefatos computacionais. Nesta visão instrumental do desenvolvimento tecnológico, a computação é vista como uma evolução constante das ferramentas usadas para o auxílio de cálculos. O computador é o estado atual de uma linhagem de máquinas de calcular criadas ao longo da história da humanidade. Monteiro (2000, p.13), por exemplo, inclui nesta linhagem o Ábaco (3000 a.C.), a Régua de Cálculo (1621), a Pascalina (1642), o Tear de Jacquard (1801), as duas máquinas de Babbage (1823), a máquina tabuladora de cartões perfurados de Hollerith (1889), ENIAC (1943) e os demais computadores eletrônicos.

A perspectiva teleológica é direta quanto às relações entre matemática e computação, principalmente em dois pontos: primeiro, ao entender que os conceitos e ideias "relevantes" citadas por Fonseca Filho (2007) são todas matemáticas e segundo, a explicitação de que diversas figuras importantes da história da matemática também se fazem presentes em uma história da computação.

Fonseca Filho (2007) inclui conceitos matemáticos na história da computação, tais como conceito de número, cálculo, lógica, geometria euclidiana e álgebra. Além disso, dá um grande destaque para o séc. XIX e considera que vários dos conceitos desenvolvidos neste período dentro da matemática são precurssores diretos da computação. Ele chama este período de "idade áurea" da matemática, pois nele surgem os trabalhos de figuras como George Cantor (1845-1918), Gottlob Frege (1848-1925), Giuseppe Peano (1858-1932), Kurt Gödel (1906–1978), George Boole (1815-1864), Bertrand Russell (1872-1970) e David Hilbert (1862-1943).

Do segundo ponto, oferecemos dois exemplos para ilustrar. Primeiro, Muhammad Ibn Musa Al-Kharazmi (780-850), um matemático e astrônomo persa, foi um dos responsáveis por

[2] Para uma discussão mais aprofundada sobre historiografia, ver Dosse (2012), Burke (1992), Malerba (2006), Barros (2012) e Bloch (2002). Para uma breve visão sobre o tema dentro da computação, ver o primeiro capítulo de Ceruzzi *et al.* (2003).

introduzir a escrita de cálculos no lugar do uso do ábaco, além da expansão do uso dos numerais hindus no mundo árabe. É de seu nome que vem os termo algarismo e algoritmo, ambos parte da matemática e da computação (FONSECA FILHO, 2007).

John von Neumann (1903-1957) é outra destas figuras com um pé na matemática e na computação. Entre suas diversas contribuições para a computação, talvez a mais difundida seja a "arquitetura de von Neumann", um modelo de computador com programa armazenado. Este conceito foi amplamente difundido por um relatório para a construção do EDVAC (*Eletronic Discret Variable Automatic Computer*), mas as origens do conceito podem ser traçadas anos antes nos trabalhos de John Mauchly[3] e John Adam Presper Eckert[4] (CERUZZI *et al.*, 2003, p.21-23). Na matemática, podemos destacar seus trabalhos com álgebra e formalismo feitos com proximidade a Hilbert.

Gostaríamos de destacar a importância que os conceitos de lógica e de formalismo têm na conexão entre matemática e computação dentro da perspectiva teleológica. Por formalismo, entendemos a escola matemática que tem como um de seus principais nomes David Hilbert e que trata das teorias axiomáticas. A base desta escola reside no conceito de axiomas que, neste caso, "podem ser entendidos como definições implícitas dos termos específicos da teoria em questão" (SILVA, 2007, p.185). Os termos, por sua vez, "só têm as propriedades que lhe são dadas pelos axiomas e suas consequências lógicas" (SILVA, 2007, p.185).

A base deste tipo de teoria está no método axiomático-dedutivo, entendido por nós como "a fundação de toda uma ciência em uma base de verdades não demonstráveis – os axiomas da teoria – a partir das quais se podem derivar todas as verdades desta ciência por meios exclusivamente lógicos" (SILVA, 2007, p.183). Entretanto, os formalistas tinham o interesse em um tipo especial de teoria axiomática dedutiva, aquelas diferentes da geometria euclidiana, em que "as asserções são destituídas de qualquer significado determinado e podem ser vistas simplesmente como uma sucessão de símbolos da linguagem em que a teoria é expressa" (SILVA, 2007, p.184-185).

É sob a influência deste tipo de pensamento matemático que Alan Turing (1912-1954) desenvolveu um dos primeiros modelos de máquina abstrata, que implementa um sistema formal automático: a máquina de Turing (FONSECA FILHO, 2007, p.75). Publicada em um trabalho

[3] John William Mauchly (1907 – 1980) foi um físico estadunidense. Mauchly foi projetistas do ENIAC e co-fundador da Eckert–Mauchly Computer Corporation, um dos pioneiros da produção de computadores eletrônicos.

[4] John Adam Presper Eckert Jr (1919 – 1995) foi um engenheiro eletricista estadunidense. Eckert foi projetistas do ENIAC e co-fundador da Eckert–Mauchly Computer Corporation, um dos pioneiros da produção de computadores eletrônicos.

chamado *On Computable Numbers with an aplication to the Entscheidungsproblem* em 1936, a máquina de Turing é um dos mais concretos elos entre os formalistas e a computação. Silva (2007, p.76) define um sistema formal automático como "um dispositivo físico que manipula automaticamente os símbolos de um sistema formal de acordo com as suas regras".

De uma forma simples, podemos dizer que uma máquina de Turing "é um tipo específico de máquina idealizada, cuja função é executar computações, especialmente computações nos inteiros positivos representados em notação monádica" (JEFFREY *et al.*, 2013, p.43). De forma geral, uma máquina de Turing tem duas partes: uma fita dividida em quadrados infinita em suas duas extremidades e um computador que sempre está sobre um destes quadrados e pode mover-se um quadrado para qualquer um dos dois lados e escrever ou apagar um traço sobre o quadrado em que se encontra situado.

Com este dispositivo de pensamento, Turing provou que existe uma equivalência entre um sistema axiomático formal semelhante à lógica e um sistema formal automático. Uma teoria axiomática formal, um dos objetos de estudo dos formalistas, é constituída como "um sistema de símbolos de uma linguagem explicitamente dada, manipulados segundo regras (de formação e transformação) também explicitamente dadas" (SILVA, 2007, p.183-184). Estes conceitos são a base para os modelos computacionais feitos na década de 40 do séc. XX e nas máquinas com a mesma base, nos 60 anos seguintes (SILVA, 2007, p.76), desde que abstraídos os detalhes.

Papert (1994, p.139)[5], ao comentar sobre estes primeiros computadores, é assertivo ao colocar que estas máquinas carregam em si as práticas e conceitos de seus criadores: "os pioneiros eram matemáticos e construíram máquinas à sua própria imagem". Kay (1984, p.4)[6] também tem uma opinião parecida: "os primeiros programas de computador foram projetados por matemáticos e cientistas que pensavam que a tarefa deveria ser direta e lógica[7]". Gostaríamos de chamar a atenção o fato de que esta percepção inicial de Papert (1994) e Kay (1984), com

[5] Seymour Papert (1928 - 2016) foi um educador estadunidense e sul-africano. Foi responsável por articular as teorias propostas por Jean Piaget (1896 - 1980) na Educação & Informática através do que foi chamado de Construtivismo. Papert também é conhecido por ser um dos criadores da linguagem LOGO (projetada dentro de suas propostas educacionais) e por ser um dos pioneiros do que é conhecido hoje como MIT lab.

[6] Alan Kay (1940) é um cientista da computação e músico. Trabalhou no Xerox PARC, Atari, Apple e HP. Kay também é um dos responsáveis pela linguagem Smalltalk, precursora da programação orientada a objeto e é um dos criadores das GUI (*Graphic User Interfaces*). Dentro de suas iniciativas sobre educação estão o *Dynabook* e o *One laptop per child*. Em 2003, Alan Kay recebeu o *ACM Turing Award* por suas contribuições para a programação e computação pessoal.

[7] *The earliest computer programs were designed by mathematicians and scientists who thought the task should be straightforward and logical.*

o aporte teórico de Vieira Pinto (2005b)[8], ganha mais dimensões e essas são importantes para pensar o ensino e aprendizagem de programação de computadores.

Primeiro, Vieira Pinto (2005b, p.315) nos chama atenção para uma constante presença de um componente indireto, estrutural, na produção de um *software*. "A construção de uma programação para a máquina é obra do operador que a utiliza, mas depende da programação original, a protoprogramação, que consiste na primitiva construção da máquina para a execução de determinados tipos de programas". Nota-se que aquilo que Vieira Pinto (2005b, p.315) chama de "operador", com o tempo, ganhou o nome de programador.

Podemos entender que a "programação original ou protoprogramação" de Vieira Pinto (2005b, p.315) é a estrutura matemática subjacente ao computador, chamada de arquitetura (organização do *hardware*, eletrônica, núcleo sistema operacional, linguagem de programação, etc) em um sentido geral. Como coloca Vieira Pinto (2005b), é justamente esta dimensão que manifesta concretamente os conceitos e ideias que deram gênese ao computador. Para Papert (1994, p.140), estes conceitos e ideias eram parte de um "tipo particular de cultura matemática no qual o cálculo acurado desempenha o papel dominante, e o técnico e o analítico têm mais peso do que o intuitivo e o experimental."

Papert (1994, p.140) ainda complementa quanto aos efeitos desta característica: "o que é significativo aqui é como elementos da cultura original do computador persistiram até mesmo quando a tecnologia não mais os requereu ou favoreceu". Ou seja, qualquer argumento funcionalista calcado em necessidades como justificativa deve ser visto minuciosamente. Com isso, um importante desdobramento desta questão para a programação de computadores é constatar que o trabalho com um artefato computacional implica em, quase que obrigatoriamente, negociar com esta tradição matemática de forma concreta nas tarefas do cotidiano.

Grenfell (2018a, p.285) lembra que para Bourdieu as "palavras se apresentam como se fossem neutras em relação a valores, quando na verdade são construções sócio-históricas, consideradas óbvias como expressões do "senso comum", mas com pressuposições especializadas sobre seus significados e imbuídas de implicações logicamente práticas desses significados".

[8] Álvaro Vieira Pinto (1909 - 1987) foi um autor brasileiro conhecido pela sua participação no Instituto Superior de Estudos Brasileiros (ISEB) e pela sua relação com Paulo Freire. Atuou como filósofo, professor, cientista e tradutor. A frente do ISEB no golpe de 64, Vieira Pinto sai do país, exila-se primeiro na Iuguslávia e depois no Chile (a convite de Paulo Freire). Retorna ao Brasil somente em 68 pouco antes do decreto do Ato Institucional n.º 5 (AI-5). Sua volta foi aceita pelo regime militar sob a condição de não ministrar aulas em universidades e de não realizar conferências. (GONZATTO; MERKLE, 2017).

Esta é a mesma visão de Bakhtin (2014, p.42) quando afirma que as "palavras são tecidas a partir de uma multidão de fios ideológicos e servem de trama a todas relações sociais, em todos os domínios".

Ou seja, reconhecer a lógica e a matemática como parte de um desenvolvimento histórico e localizado da computação permite qualificar melhor o que é esta "abstração" para o campo. Argumentamos que a abstração defendida pelo "pensamento computacional" enfatiza, na verdade, um processo mais específico – o de formalização com base no método axiomático-dedutivo.

Em algumas discussões sobre abstração, a filiação matemática é explícita como no caso de Bucci, Long e Weide (2001), em que é defendida a modelagem matemática como forma de abstração, pois ela seria testada e provada pelo tempo. Em outras, como o caso de Kramer (2007), ela é implicita e reside no entendimento de que estudar matemática seria uma forma de desenvolver o pensamento abstrato.

A problemática ganha espessura quando lembramos que Wing (2006) é enfática sobre o quanto o "pensamento computacional" deve ser uma habilidade fundamental para todos. Não reconhecer que tal habilidade é desdobramento de uma herança específica da trajetória histórica do computador, faz com que, novamente, a dimensão técnica e analítica tenham mais peso do que o intuitivo e o experimental. É importante lembrar que Wing é alguém envolvida com a pesquisa e desenvolvimento desta dimensão, algo que pode ser visto em sua produção bibliográfica[9].

Em um quadro maior, de um ponto de vista mais próximo da filosofia, podemos dizer que esta forma de entender o "pensamento computacional" é parte do que Santos (2002, p.241-242) chama de razão metonímica, que seria uma forma de entender o mundo:

> obsecada pela ideia de totalidade sob a forma da ordem. Não há compreensão nem acção que não seja referida a um todo e o todo tem absoluta primazia sobre cada uma das partes que o compõem. Por isso, há apenas uma lógica que governa tanto o comportamento do todo como o de cada uma dass suas partes. Há, pois, uma homogeneidade entro o todo e as partes e estas não têm existência fora da relação com a totalidade. (SANTOS, 2002, p.241-242)

Para Santos (2002, p.246), esta racionalidade produz de forma ativa a não-existência daquilo que não cabe em sua totalidade. Ele separa em cinco grupos as formas de produção desta não-existência e afirma que todas essas seriam manifestações de uma "monocultura racional" (SANTOS, 2002, p.247).

[9] Para exemplificar, o primeiro texto em seu currículo disponível em sua página pessoal da Universidade de Columbia tem como título "*Some Remarks on Putting Formal Specifications to Productive Use*" (GUTTAG; HORNING; WING, 1982). Ela também é coautora de "*A Behavioral Notion of Subtyping*" (LISKOV; WING, 1994) com Barbara Liskov. Este é o texto base do Princípio da Substituição de Liskov, conceito que dá uma base formal para algumas questões envolvendo tipos e subtipos, pricipalmente, em programação orientada a objetos. O currículo consultado está disponível em: https://www.cs.columbia.edu/~wing/resume.pdf.

O primeiro grupo é chamado de monocultura do saber e do rigor do saber. Santos (2002, p.247) considera esta forma de produção a mais poderosa e ela consiste na "transformação da ciência moderna e da alta costura em critérios únicos de verdade de qualidade estética, respectivamente. [...] cada uma no seu campo, cânones exclusivos de produção de conhecimento ou de criação artística".

Mesmo entendendo que a visão de Santos (2002) sobre ciência e alta costura precisem que ambas sejam homogêneas para não invalidar seu argumento, o pensamento computacional é derivado de uma empreitada científica que quando simplificada de forma errônea dá a entender que sua forma de trabalho é um cânone exclusivo, formado por uma mistura da matemática com uma idea de que existe apenas uma forma correta de se fazer algo.

No segundo grupo, Santos (2002, p.251) trata sobre a questão da monocultura do tempo linear. Aceitar o tempo como algo que flui somente em uma direção e que o contemporâneo é a ponta deste vetor permite hierarquizar práticas e classificar como residuais aquelas que não são compatíveis com a temporalidade dominante. Ou seja, considerar certas práticas sociais entendidas como "atrasadas" só faz sentido sob o entendimento do tempo como linear. Estas práticas, na verdade, se filiam a temporalidades não reconhecidas pela racionalidade metonímica.

A Informática e a Ciência da Computação contribuem para esta questão, pois representam o "progresso", carregam em si o *status* da "tecnologia de ponta", como argumentado no capítulo anterior. Também é por seus artefatos que a globalização opera de forma efetiva ao mesmo tempo que impõe sua temporalidade aos diversos contextos locais. Por exemplo, para aumentar a rotatividade de vagas no centro, cidades como São Paulo e Curitiba usam um sistema de vagas rotativo. O controle que antes era feito em um papel, nas duas cidades, foi migrado para o formato digital através de aplicativos de celular. Ao aposentar o sistema antigo, as prefeituras obrigaram aqueles que não têm familiaridade com celulares e afins a se submeterem ao ritmo destes sistemas, caso contrário, são excluídos desta prática que já não era das mais cidadãs.

O terceiro grupo é chamado de lógica da classificação social. Neste, Santos (2002, p.252) chama a atenção para o efeito que a razão metonímica tem sobre os agentes, que de forma indireta, também atinge as práticas das quais esses são protagonistas. Esta é a questão levantada na introdução desta tese, em que apresentamos o contexto em que as iniciativas ligadas à representatividade e inclusão de minorias operam no campo da Informática e Ciência da Computação.

O quarto grupo, o da lógica da escala dominante, “a escala adotada como primordial determina a irrelevância de todas as outras possíveis escalas” (SANTOS, 2002, p.248). Para Santos (2002, p.248) a modernidade ocidental escolheu a globalização e o universalismo como primordiais. Mais à frente, a crítica de Vieira Pinto (2005a).

O quinto agrupamento é chamado de lógica produtivista, em que o crescimento econômico seria um objetivo racional inquestionável e, por consequência, também torna o critério de produtividade inquestionável. Segundo esta lógica, “a não-existência é produzida sobre a forma do improdutivo que, aplicada a natureza, é esterilidade e, aplicada ao trabalho, é a preguiça ou desqualificação profissional” (SANTOS, 2002, p.248).

A própria forma como um curso ligado à informática e à computação é montado mostra como esta lógica produtivista opera. Cursos técnicos de nível médio e tecnologias de nível superior são mais curtos e não é raro ter como objetivo formar mão de obra para o mercado de trabalho. Além disso, a própria organização das disciplinas também carrega este tipo de lógica, uma vez que do ponto de vista estrito da grade curricular, um ou uma “bom/boa estudante” seria aquele que consegue terminar o curso no tempo regular.

Para o pensamento computacional, dada sua definição e articulação sem muita profundidade, Santos (2002) mostra que se não for reconhecido o horizonte do qual o conceito tira seu significado, defender sua inclusão no Ensino Básico ou argumentar que é um conceito fundamental faz com que ele seja um elo, mais uma forma de continuidade desta racionalidade metonímica.

Assim, quanto à questão de ensino e aprendizagem, entender o “pensamento computacional” como uma forma diferente de apresentar o processo de formalização matemática faz com que esse seja um conceito que apenas reforça as práticas correntes do campo, contribuindo pouco para uma reflexão sobre as bases teóricas nas quais são pensados os processos de ensino e aprendizagem em computação.

A seguir, iniciamos a construção do nosso objeto de pesquisa com a busca de uma forma de ver o processo de abstração que, ao mesmo tempo, reconheça a herança da matemática para a computação e consiga ir além de considerar o processo em si como benéfico, sem nenhum tipo de reflexão crítica.

3.2 O ABSTRATO COMO MEDIAÇÃO ENTRE CONCRETUDES

O minidicionário Aurélio define abstração como “resultante de abstração”, “que opera com qualidades e relações, e não com a realidade” e “que expressa qualidade ou característica separada do objeto a que pertence ou está ligada”. Já o concreto, o mesmo dicionário define como “que existe em forma material” e “de consistência mais ou menos sólida” (HOLANDA, 2010).

Como mostrado na seção anterior, uma parte considerável dos textos do recorte apresentado usa o termo “pensamento computacional” de uma forma parecida. Estes contextos parecem partir de um pressuposto em que uma simples definição é o suficiente para desenvolver seu argumento.

Meksenas (1992) chama a atenção para que a oposição entre concreto e abstrato é uma espécie de senso comum com raízes na filosofia. Para ele, o problema da oposição entre os dois termos é histórica e apoia-se em Lefebvre (1991, p.49-50) para localizar a origem desta dicotomia na Grécia Antiga. Ambos autores dão destaque para a influência do modelo social vigente na época para a construção dos dois conceitos:

> “Ou seja, com Aristóteles consolida-se a distinção entre contemplação (entendida com abstração) e ação (entendida como concretude), possuindo, na época, um forte componente social: a escravidão. Todo trabalho prático e produtivo cabia aos escravos, enquanto o pensamento metafísico era uma atividade própria da aristocracia, livre para pensar.” (MEKSENAS, 1992, p.93)

Esta perspectiva sobre os conceitos de concreto e de abstrato pode levar a uma visão elitista de conhecimento, a qual associa o abstrato a algo somente acessível a uma minoria preparada para tal (MEKSENAS, 1992). Este argumento é reforçado pela visão de Silva (2000) sobre os conceitos de identidade e de diferença em que ambas são “resultado de atos de criação linguística” (SILVA, 2000, p.76), ou seja, “elas não são criaturas do mundo natural ou de um mundo transcendental, mas do mundo cultural e social”(SILVA, 2000, p.76). O mesmo raciocínio pode ser aplicado aos termos sob análise. De acordo com Lefebvre (1991, p.49-50), tanto o concreto quanto o abstrato têm raízes históricas, são interdependentes e são produtos e produtores dos meios sociais nos quais estão inseridos.

Silva (2000) também tenta explicar uma outra problemática que surge do uso de oposições binárias. Neste caso, o abstrato e o concreto, como forma de marcação social e linguística: “em uma oposição binária, um dos termos é sempre privilegiado, recebendo um valor positivo, enquanto o outro recebe uma carga negativa” (SILVA, 2000, p.83). O positivo é tratado como

a “norma”, enquanto o negativo é colocado como uma espécie de desvio do que é normal. A problemática surge ao entender que esta relação não é simétrica: a norma define o que é correto, muitas vezes percebida como a única forma, delegando ao polo negativo a função de ser “readequado” à norma.

Para Meksenas (1992, p.93), é o conceito de concreto que ganha conotações positivas associadas com outros termos, tais como “realidade”, “coisas palpáveis e perceptivas” e “compreensível”. Já ao abstrato são relegadas conotações negativas “imaginário puro”, “deslocado da realidade” e “distante”. É importante lembrar que Meksenas (1992) está tratando da percepção de professores sobre os dois termos. Machado (2011) segue a mesma linha, mas sem restringir-se ao contexto educacional:

> “No dia a dia, com frequência a referência a algo com o *abstrato* ocorre impregnada de conotações negativas, como as associações à dificuldade de compreensão e ao interesse de poucos, ou de sentidos contraditórios, que situam pendularmente as abstrações entre a essência do real e o que nada tem de real”. Machado (2011, p.48)

É possível notar novamente o peso negativo associado ao abstrato. Entretanto, no caso da computação, os pólos são invertidos, sendo o abstrato o pólo positivo e o concreto o negativo. Papert (1994, p.123) trabalha dentro desta mesma perspectiva, inclusive colocando de forma explícita a questão ao afirmar que “Um tema central da minha mensagem é que a tendência dominante a supervalorizar o abstrato é um importante obstáculo ao progresso na Educação”. Além disso, em outro momento, o mesmo Papert (1994, p.130) também reconhece a associação com o positivo do polo abstrato: “Nossa cultura intelectual tradicionalmente tem sido tão dominada pela identificação de bom pensamento como pensamento abstrato...”.

Em especial, a valorização do abstrato tem uma forte aderência quanto a sua importância para as disciplinas ligadas à área de exatas. Por exemplo, na pesquisa de Andriola e Cavalcante (1999) sobre o raciocínio abstrato no Ensino Médio, temos:

> Especificamente em relação à avaliação do raciocínio abstrato, pode-se destacar a importância que assume para os que estudam, principalmente, matemática e física, pelo fato de essas disciplinas exigirem um elevado grau de capacidade para perceber e aplicar relações entre fenômenos e símbolos abstratos. (ANDRIOLA; CAVALCANTE, 1999, p.33)

Dentro da área da computação, não custa lembrar que Wing (2006) explicitamente coloca que “pensamento computacional” não se restringe a programar, mas ir além disso, é conseguir “pensar em múltiplos níveis de abstração”(WING, 2006, p.35). O desdém pelo programar, parte concreta do processo, mostra a clara associação positiva que o polo abstrato tem dentro do “pensamento computacional”.

Turkle e Papert (1992, p.18) são mais diretas/os ao colocar que o problema de exaltar o abstrato é deixar de lado pessoas que não necessariamente abordam problemas usando as ferramentas associadas associadas ao formalismo matemático:

> ...em nossa cultura, os pensadores estruturados, orientado ao planejamento e abstratos não somente compartilham um estilo, mas constituem uma elite epstemológica. Linguajar como "ciência pura" e "matemática pura" implicam que sua superioridade é alcançada por filtar o concreto, o que implica em rejeitar continuamente pessoas como Lisa e Robin[10,11]. (TURKLE; PAPERT, 1992, p.18)

Já Vieira Pinto (2005a) relaciona esta supervalorização do abstrato com a divisão de trabalho. Para ele, existe um fenômeno de Desvalorização Social das Técnicas presente nas sociedades em que "vigoram distinções de nível na apreciação do trabalho humano"(VIEIRA PINTO, 2005a, p.316), o qual deve "ser entendido no sentido do processo pelo qual se realiza na verdade o movimento de democratização das técnicas, pelo acesso a elas, mesmo quando nos degraus menos exigentes, das massas trabalhadoras" (VIEIRA PINTO, 2005a, p.316).

Para Vieira Pinto (2005a, p.316), o processo de conhecer deve necessariamente terminar em um ato de trabalho para não ficar restrito no domínio da imaginação. Este processo, que em outras épocas era feito pelo cientista, fazia com que o pesquisador fosse o próprio construtor de seus aparelhos. Agora, com a divisão de trabalho "essa exigência manifesta-se pela desvalorização e redução dos conhecimentos numa escala descendente de atribuições técnicas" (VIEIRA PINTO, 2005a, p.317).

Vieira Pinto fornece como exemplo desta escala de valorização das capacidades humanas os eletrodomésticos. Estes objetos banais, quando foram concebidos, representavam "assombrosos triunfos da inteligência humana, na pesquisa das forças naturais e no domínio delas para um fim útil" (VIEIRA PINTO, 2005a, p.317). Hoje, transformados em bens de consumo, fazem parte do cotidiano. Além disso, como exemplo concreto desta cadeia, ele usa os tipos de trabalho associados à televisão: "físico - engenheiro - chefe de oficina - operário - vendedor - concertador" (VIEIRA PINTO, 2005a, p.317).

No caso da computação, mesmo sabendo que a organização social é mais complexa que uma linha, conseguimos encontrar posições similares as dadas por Vieira Pinto (2005a). O físico seria o cientista da computação, enquanto o engenheiro serial algo equivalente aos analistas/engenheiros de software, desenvolvedor e demais equivalentes. Chefe de oficina, operário, vendedor

[10] Lisa e Robin são duas personas do texto caracterizadas como garotas, uma poeta e outra pianista, e que não necessariamente abordam problemas da forma privilegiada pela computação.

[11] *...in our culture, the structured, plan-oriented, abstract thinkers don't only share a style but constitute an epistemological elite. Language such as "pure science" and "pure mathematics" implies that their superiority is achieved by filtering out the concrete, and this means a continual put-down of people like Lisa and Robin.*

e consertador não precisam nem de equivalência, pois artefatos computacionais operam quase que da mesma maneira que os eletrodomésticos.

Quanto a uma valoração pejorativa desta escala, Vieira Pinto coloca que essa seria uma posição de uma consciência arrogante que não consegue enxergar que neste processo existe uma "elevação cultural do trabalho manual" (VIEIRA PINTO, 2005a, p.317), pois este processo mostra "a progressiva incorporação ao patrimônio social, ainda em formas consideradas modestas, dos elementos do conhecimento científico que principiaram sendo propriedade de um único, ou de poucos indivíduos, os homens da ciência" (VIEIRA PINTO, 2005a, p.317).

Nos textos analisados, não encontramos um desdém pelo concreto explicitamente, mas o entendimento de abstração como algo positivo está presente. Por exemplo, nos três casos a seguir, temos em Zanetti e Oliveira (2015) e Vera *et al.* (2016), o entendimento da abstração como algo "fundamental" enquanto em Raabe, Zanchett e Vahldick (2015), a abstração aparece como um pré-requisito importante:

> A capacidade de abstração é algo fundamental para o sucesso na aprendizagem de programação, principalmente para compreender problemas e propor soluções. A adoção do Pensamento Computacional pode orientar de forma ampla a atividade mental de abstrair problemas e formular soluções descritas em algoritmos. (ZANETTI; OLIVEIRA, 2015)

> A abstração é um elemento fundamental para a Ciência da Computação, que tradicionalmente tem-se articulado com áreas como Matemática, Estatística, Física e Engenharias. Considerando a importância dos estudos interdisciplinares na Ciência da Computação, para sua evolução enquanto ciência e aplicação prática para solucionar os mais complexos problemas sociais (Cassel, 2011), novas iniciativas de integração epistemológica têm sido estudadas (Souza, 2016; Matos, Paiva e Corlett, 2016). (VERA *et al.*, 2016)

> Kinnunen et al. (2007) ressaltam que programação requer um alto nível de abstração, prática e um esforço intenso. Gomes e Mendes (2007) destacam a motivação como um dos fatores principais para o sucesso, principalmente devido ao fato de que muitos alunos possuem uma visão negativa associada à programação e aos programadores, que são em boa parte vistos como pessoas antissociais ou introspectivas. (RAABE; ZANCHETT; VAHLDICK, 2015)

Novamente, não está sendo julgado se as proposições são certas ou erradas, mas argumentamos que dada a importância colocada sobre o conceito de abstração, é importante ir além de um termo dado. Inclusive, como mostra Blikstein *et al.* (2014), até mesmo dentro desta "abstração" podem existir diversas estratégias para abordar um problema computacional.

Para Althusser (2019, p.74), o ato de abstrair seria " 'separar' uma parte da realidade do restante da realidade. Abstração é, primeiramente, essa operação e seu resultado. O abstrato opõe-se ao concreto como a parte separada do todo se opõe ao todo". É importante atentar-se principalmente ao trecho "...como a parte separada do todo se opõe ao todo", pois nela é possível

identificar que a relação entre concreto e abstrato não necessariamente é de oposição e exclusão mútua.

Blaha e Rumbaugh (1994), em um livro sobre projetos de programas orientados a objetos, oferece uma definição de abstração mais próxima das questões que envolvem computação:

> Abstração é o exame seletivo de determinados aspectos de um problema. O objetivo da abstração é isolar os aspectos que sejam importantes para algum propósito e suprimir os que não o forem. A abstração deve sempre visar um propósito, porque este determina o que é e o que não é importante. Muitas abstrações diferentes da mesma coisa são possíveis, dependendo do propósito para o qual forem feitas. (BLAHA; RUMBAUGH, 1994)

É interessante notar que para Blaha e Rumbaugh (1994), abstração é um processo ao mesmo tempo que é um produto:

> Todas as abstrações são incompletas e inexatas. A realidade é um tecido sem costuras. Tudo o que se disse sobre ela, qualquer descrição dela é apenas um resumo. Todas as palavras e linguagens são abstrações – descrições incompletas do mundo real. Isso não elimina sua utilidade. O propósito de uma abstração é limitar o universo para que possamos fazer coisas. Na construção de modelos, portanto, não se deve procurar a verdade absoluta e sim a adequação a algum propósito. Não há um único modelo "correto"de uma situação, apenas modelos adequados e inadequados. (BLAHA; RUMBAUGH, 1994)

Zehetmeier *et al.* (2019) fazem um levantamento de 22 definições de abstração que vão desde a dada por dicionários (como a do início deste capítulo), definições tiradas da filosofia (Locke), de diversas ciências (Psicologia, Educação, etc), inclusive da Ciência da Computação. Neste estudo, Zehetmeier *et al.* (2019) agrupam as definições em três conjuntos, de acordo com suas características: *Commonalities and Differences*, *Hide and Keep* e *Expand*. É importante notar que a abstração é definida por características e não como um processo.

Para Althusser (2019, p.75, 79), a primeira abstração é a linguagem em si, e ao comentar sobre os seus usos em diferentes contextos, chega à conclusão de que “a abstração da linguagem serve para designar o concreto mais concreto”, ou seja, existe uma espécie de apropriação do concreto pelo abstrato que o enriquece, não sendo somente uma espécie de operação subtrativa da realidade, como é a dada por Blaha e Rumbaugh (1994) ou Kramer (2007): “O que caracteriza a abstração é o fato de ser outra coisa, e não uma parte do concreto, visto que acrescenta algo a ele” (ALTHUSSER, 2019, p.83).

A partir disto, Althusser (2019, p.83) considera a existência de um ciclo em que “o concreto está no início, em seguida vem o abstrato, depois novamente o concreto”. Meksenas (1992) faz o mesmo ao propor entender o concreto e o abstrato de uma outra forma a fim de fugir da perspectiva dicotômica, apoiando-se no entendimento do abstrato como mediação, definida por Machado (2011, p.55) como:

> "(...) no processo de elaboração do conhecimento, as abstrações são mediações indispensáveis. Situam-se sempre no meio do processo, constituindo em condição de possibilidade do conhecimento em qualquer área, em vez de ponto de partida ou ponto de chegada. São um degrau necessário que conduz de um patamar de concretude a outro". (MACHADO, 2011, p.55)

Meksenas (1992, p.95) redefine o processo de forma mais resumida ao colocar que "as diferentes práticas de ensino partem sempre de certos níveis de concretude, chegando, pela abstração, a outros níveis de concretude". Ou seja, o concreto não é o contrário do abstrato, nem mesmo são entidades de mesmo tipo. Pode-se dizer que o concreto é um dado estado de percepção de um objeto enquanto o abstrato é um processo de passagem para um outro tipo de estado de percepção.

Meksenas (1992, p.95) ainda coloca que o processo não acaba no segundo estágio de "concretude", mas é um processo que se repete eternamente. Como colocado em Kira e Merkle (2018, p.4), na computação, isto implica em tratar os temas habitualmente abstratos como outros tipos de concretude, apenas "mais específicos e mais complexos que os habituais. Ou seja, ensinar e aprender o conceito de árvore apresentado como uma estrutura de dados é fazer uma passagem do conceito de árvore (ser que existe no mundo) para o de árvore (estrutura análoga que possui relações e propriedades ligadas ao computador, à matemática e à planta árvore). Ambos são concretos, mas estão sob óticas diferentes de conhecimento."

O entendimento do abstrato como um processo de mediação[12] feito por Meksenas (1992) tem como base a teoria histórico cultural, corrente da psicologia que trabalha com o materialismo dialético e tem como seu nome mais conhecido Lev S. Vigotski, o qual será mais discutido na próxima seção.

3.3 O PAPEL DO PROFESSOR PARA VIGOTSKI

Lev Vigotski (1896-1934) viveu durante o final do regime czarista russo e começo da União Soviética. É neste contexto, principalmente após a Revolução Russa (1917), ou seja, nos anos em que Stalin está a consolidar seu poder dentro do Partido Comunista, que Vigotski desenvolve sua teoria, hoje conhecida como teoria histórico-cultural[13].

[12] Valorizar o concreto não é exclusividade da linha teórica escolhida neste trabalho. É preciso reconhecer que tal movimento poderia ser feito sobre uma base pragmatista Pierciana ou fenomenológica com o uso de Heidegger ou Husserl.

[13] Dependendo de quem e como são organizadas as teorias, também é chamada de Teoria da Atividade Histórico-Cultural. Neste último termo, é comum incluir os trabalhos associados a Yrjö Engeström(1948-) e Alexei Leontiev (1903-1979).

Vigotski era um estudioso da pedologia, ramo da psicologia na Rússia que tinha associações com os estudos da criança norte-americanos e a pedagogia experimental alemã (VEER; VALSINER, 2014, p.319). Para Veer e Valsiner (2014, p.319), Vigotski "participou dos esforços para fazer da pedologia um meio que pudesse levar às metas da criação do "novo homem" durante a reestruturação social". Entretanto, a pedologia era uma disciplina que tinha uma tradição relativamente estabelecida na Rússia pré-revolução, algo que junto com outras questões, colocou-a em choque com as diretrizes do partido soviético, que na década de 30 já estava sob a tutela de Stalin.

Vigotski vem a falecer de tuberculose em 1934, com 38 anos. Em 1936, o Decreto da Pedologia estabelece a disciplina como ilegal, colocando a obra de Vigotski em um hiato, sendo redescoberta somente quase 20 anos mais tarde (PRESTES, 2021, p.50-51).

Para Lefrancois (2016, p.254), exitem três temas sobrepostos que unificam a teoria de Vigotski: o papel da fala (no original, o autor comenta sobre o papel da linguagem, mas como será argumentado a seguir, optamos em usar fala), a importância da cultura e a relação entre estudante e professor. O autor também destaca o papel dos conceitos de funções psicológicas superiores e inferiores[14] na centralidade da cultura dentro da teoria de Vigotski.

O livro *Michlenie i retch* (*Pensamento e Fala*) foi o último livro de Lev Vigotski (1896-1934) e foi escrito em seus últimos dias de vida. Já com a saúde debilitada, vários trechos foram ditados e corrigidos posteriormente pelo autor. Lançado em 1934, foi proibido em 1936 e só foi publicado novamente em 1956 (PRESTES, 2021, p.138-139).

A versão de 1956 era uma versão resumida da original, com certa de 40% menos conteúdo e foi a versão popularizada fora da Rússia/União Soviética. Foi esta a versão traduzida para o português pela Martins Fontes, a partir de um texto em inglês, com o título de Pensamento e Linguagem (PRESTES, 2021, p.144).

Prestes (2021, p.207-218) é enfática sobre o quanto erronêo é traduzir *retch* como linguagem e argumenta que o mais apropriado seria traduzir a palavra *retch* como fala, mesmo que em outras língua tenha sido feito traduções parecidas, como é o caso dos títulos de publicações do mesmo livro em inglês (*Thought and Language*) ou em francês (*Pensee et langage*).

Podemos ver esta ênfase na fala em diversos momentos como, por exemplo, quando Vygotsky (2007, p.43-44) faz comentários sobre as pesquisas de seus próprio grupo (Zankov e Yussevich). As crianças de ambos os estudos não demonstraram a capacidade de associar

[14] Na tradução consultada, são usados os termos funções mentais superiores e inferiores. Optamos por usar funções psicológicas apenas por padronização.

arbitrariamente um signo qualquer com uma palavra conhecida em uma tentativa recuperá-la indiretamente. Para Vygotsky (2007), "a criança reproduziu a palavra solicitada através de um processo de representação direta, em vez de uma simbolização mediada".

Discutir se Vigotski trabalha com a fala ou linguagem é importante para este trabalho, pois estamos a tratar sobre ensino e aprendizagem de linguagens de programação de computadadores que, na visão de Eco (2001, p.337), é herdeira da busca pelas línguas perfeitas e tem duas limitações fundamentais advindas de sua relação com as línguas filosóficas: "1) Constroem suas regras com base na lógica elaborada pela civilização ocidental que, na opinião de muitos, afunda as suas bases na estrutura das línguas indo-europeias; 2) São limitadamente dizíveis, não permitindo exprimir tudo aquilo que pode exprimir uma língua natural" Eco (2001, p.337).

De certa maneira, o primeiro ponto de Eco (2001) é mais ou menos o que tratamos no resgate histórico feito no Capítulo 2. Já o segundo, a falta de oralidade das linguagens de programação, é algo que Machado (2011, p.111) também identifica na Matemática: "a Matemática não comporta a oralidade, caracterizando-se como um sistema simbólico exclusivamente escrito". Assim, se o foco de Vigotski é a fala e uma linguagem de programação, não a tem como proceder?

Prestes (2021, p.208) afirma que quando Vigotski usa o termo fala, esse está a se referir a "algo expresso oralmente ou de forma escrita" (PRESTES, 2021, p.208). Ou seja, as considerações sobre fala de Vigotski podem ser usadas para o processo de escrita. Entretanto, é importante comentar que Vigotski reconhece fala e escrita como coisas diferentes. Além de usar o termo *pismennaia retch*, que pode ser traduzido como fala escrita, ele comenta também que nos anos iniciais da escola, o desenvolvimento da fala oral e da fala escrita não coincidem (PRESTES, 2021, p.217) em um claro reconhecimento de suas diferenças.

Se a fala de Vigotski engloba tanto fala quanto escrita e suas relações com o pensamento, com as devidas proporções, podemos articular suas teorias dentro do ensino e aprendizagem de linguagens de programação, dada a importância que a fala tem em seus escritos, como por exemplo, na consideração a seguir:

> o momento de maior significado no curso do desenvolvimento intelectual, que dá origem às formas puramente humanas de inteligência prática e abstrata, acontece quando a fala e atividade prática, então duas linhas completamente independentes de desenvolvimento, convergem. (VYGOTSKY, 2007, p.11-12)

Prestes (2021, p.208) afirma que Vigotski entende a fala e o pensamento como dois processos psíquicos diferentes que, em um dado momento do desenvolvimento, se unem e

formam uma unidade: o pensamento verbal. O pensamento verbal faz uso da fala e signos como como instrumentos de uma ação e é o que diferencia a forma como humanos abordam um problema da forma como outros abordam animais (VYGOTSKY, 2007, p.12).

Vygotsky (2007, p.12) usa como exemplo experimentos com crianças que estão em processo de desenvolvimento do pensamento verbal. Ele compara a forma como estas crianças e macacos abordam um mesmo problema e afirma que aquelas têm uma maior gama de possíveis ações devido à fala (neste caso, a fala egocêntrica). Vygotsky (2007, p.23) entende a fala como um processo analítico e é o que dá as crianças "sua maior independência em relação à estrutura da situação visual concreta" (VYGOTSKY, 2007, p.14). Ele ainda atribui à fala o início de uma capacidade de planejamento e um controle sobre o próprio comportamento:

> A manipulação direta é substituida por um processo psicológico complexo através do qual a motivação interior e as inteções, postergadas no tempo, estimulam o seu próprio desenvolvimento e realização. (VYGOTSKY, 2007, p.14)

A fala egocêntrica tem uma ligação com a fala social. Por exemplo, uma criança que percebe ser incapaz resolver um problema sozinha dirige-se a um adulto em busca de auxílio descrevendo verbalmente o seu método. Uma mudança grande em seu desenvolvimento ocorre quando esta fala socializada, dirigida ao adulto, é internalizada. No lugar de apelar ao outro, a criança passa a apelar para si. A criança reorganiza o seu comportamento ao impor a si uma forma social de comportamento (VYGOTSKY, 2007, p.16).

Nos experimentos usados por Vigotski, as crianças, ao se depararem com um problema um pouco mais complicado, adotam diversas estratégias: tentam atingir o objetivo diretamente, usam instrumentos, falam com quem está a conduzir o experimento ou uma fala que apenas acompanha as ações. Para Vygotsky (2007, p.20), todas estas atitudes mostram que:

> O caminho do objeto até a criança e desta até o objeto passa através de outra pessoa, Essa estrutura humana complexa é o produto de um processo de desenvolvimento profundamente enraizado nas ligações entre história individual e história social. (VYGOTSKY, 2007, p.20)

Na programação de computadores, podemos dizer que existem algumas atividades análogas ao processo descrito por Vigotski. Não é incomum, ao aprender a resolver problemas com o uso de uma linguagem de programação, recorrer a outras pessoas para obter um auxílio. Esta escrita em várias "mãos" em algum momento deste processo torna-se internalizada, aparentando funcionar de forma similar ao que Vigotski descreve nos experimentos.

A teoria de Vygotsky (2007) trabalha com dois tipos de funções psicológicas: as elementares e as superiores. As elementares ‘’‘têm como característica fundamental o fato

de serem total e diretamente determinadas pela estimulação ambiental" (VYGOTSKY, 2007, p.33). Essas seriam as respostas condicionadas biologicamente como reações diretas ao contexto apresentado.

Já o conceito de função psicológica superior tem como característica essencial "a estimulação autogerada, isto é, a criação e o uso de estímulos artificiais que se tornam a causa imediata do comportamento" (VYGOTSKY, 2007, p.33). Vygotsky (2007, p.32) chama estes estímulos artificiais de signo e dá como exemplo marcar um pedaço de madeira ou atar um nó em um pano para o uso como um dispositivos mnemônicos.

Ainda sobre o uso de signos, Vygotsky (2007, p.33) afirma que este tipo de comportamento está ausente nas espécies superiores de animais e coloca as operações com signos como o "produto das condições específicas do desenvolvimento social" (VYGOTSKY, 2007, p.33).

Os instrumentos, junto com os signos, são a outra parte que formam as funções psicológicas superiores, mas Vigotski deixa claro que ambos não esgotam a função: "podemos usar o termo função psicológica superior ou comportamento superior com referência à combinação entre o instrumento e o signo na atividade psicológica" (VYGOTSKY, 2007, p.56). Tanto o signo quanto o instrumento são usados no processo mediado, mas orientam de formas diferentes o comportamento humano: O signo é um meio de atividade interna dirigido para o controle do próprio indivíduo, enquanto o instrumento é orientado externamente e dirigida para o controle e domínio da natureza. Ambos são um terceiro elemento, parte de atos mediados que se colocam entre humanos e a natureza (VEER; VALSINER, 2014, p.242).

Veer e Valsiner (2014, p.217) comentam que esta separação que Vigotski faz entre biologia e história baseava-se no que foi escrito por Marx e Engels. A forma como Vigotski apresenta o fucionamento do instrumento é compátivel com o a forma que Engles define trabalho (transformação da natureza) e também seria aquilo que separa a humanidade da natureza.

As funções psicológicas superiores representam a forma como a cultura e a história interagem com o indivíduo dentro do quadro teórico de Vigotski. Com este entendimento, podemos afirmar que a Ciência da Computação faz parte da "cultura" humana, logo pode ser entendida como parte e sob as regras que regem os funções psicológicas superiores. Colocamos entre aspas a cultura, pois é importante entender que a visão de Vigostki sobre ela é a visão de alguém que viveu no começo do século XX e interagia com as teorias da época.

Para Veer e Valsiner (2014, p.233), Vigotski tinha uma visão etnocêntrica da cultura e de certa maneira, linear. Por exemplo, sobre a cultura islâmica no Uzubequistão, Vigotski

afirmou que o nível de desenvolvimento social e cultural era baixo e que ela deveria dar um salto histórico para alcançar o nível da cultura socialista unificada (VEER; VALSINER, 2014, p.235). Apenas para situar um pouco o estado que se encontravam as discussões sobre cultura, Émile Durkheim (1858-1917), um dos precursores da sociologia e Fraz Boas (1958-1942), pioneiro do método etnográfico, eram autores relativamente recentes. Antropólogos de muita importância como Levi-Strauss (1908-2009) e Clifford Geertz (1926-2016) ainda mal tinham começado suas carreiras.

É importante afirmar que Vigotski não atribuia este "atraso" a fatores biológicos, mas as próprias dinâmicas das culturas. Um outro fator relevante é entender o contexto da União Soviética pós revolução e sua busca por uma igualdade, sob uma identidade unificada (VEER; VALSINER, 2014, p.233).

Na computação, existe também um certo ufanismo quanto à importância de seus próprios processos. Com base nessa infelicidade, temos um certo paralelo possível entre a cultura que Vigotski trabalha com a "cultura" da computação, em especial em sua vertente escolar. Esta visão é bem representada pelos textos que aderem a questão de Wing (2006), que a computação é necessária para todos e que representa uma forma superior de conhecimento. Dada esta compatibilidade, é preciso deixar claro que rechaçamos este lado específico da teoria de Vigotski, dado o desenvolvimento do campo dos estudos de cultura que ocorreram em quase um século que nos separa do contexto em que ele escreveu seus textos.

Assim, argumentamos que os processos ligados ao campo da Computação são parte das funções psicológicas superiores e podem ser internalizados por indivíduos de forma similar a como Vigotski apresenta o processo de aquisição do pensamento verbal por uma criança: Primeiro de forma interpessoal, mediada por signos e instrumentos, para que ao longo do desenvolvimento torne-se intrapessoal. Além disso, este processo altera próprias estruturas de pensamento do indivíduo. De forma curiosa, podemos entender que estamos aqui mostrando uma forma de ver o pensamento computacional sob a lente da teoria histórico cultural de Vigotski.

Se a fala é uma das ligações entre o interpessoal e o extraspessoal e as funções psicológicas superiores corespondem aos mecanimos de operação cultural, o papel do professor está na mediação do fluxo entre os conhecimentos históricos e culturais com os da experiência individual. Esta questão envolve diretamente o conceito de zona de desenvolvimento iminente[15], que é definido como:

[15] Também traduzido como zona de desenvolvimento proximal. Usamos iminente, tradução defendida por Prestes (2021).

> a distância entre o nível do desenvolvimento atual da criança, que é definido com ajuda de questões que a criança resolve sozinha, e o nível do desenvolvimento possível da criança, que é definido com a ajuda de problemas que a criança resolve sob a orientação dos adultos e em colaboração com companheiros mais inteligentes. (VIGOTSKI, 2004b apud PRESTES, 2021, p.204)

A referência de Vigotski a "companheiros mais inteligentes", independente do contexto e significado, será entendida como referente a colegas com mais experiência ou com domínio mais regular de conceitos e habilidades ligadas ao conjunto de técnicas e teorias associadas à Ciência da Computação. Acreditamos que esta troca não prejudica a base do conceito ao mesmo tempo que evita qualquer direcionamento para um inatismo causado somente pelos termos escolhidos[16].

O nível de desenvolvimento atual, para Vigotski (2004a, p.537), seria o nível das funções amadurecidas, ou seja, aquilo que a criança já tem conhecimento e consegue fazer sem a ajuda de outra pessoa. Dentro da zona de desenvolvimento iminente, estariam os conceitos que a criança domina com um auxílio de alguém em um processo mediado. Prestes (2021, p.205) coloca que a característica essencial da zona de desenvolvimento iminente está na possibilidade de desenvolvimento e ressalta que um amaduracimento não é compulsório:

> pois se a criança não tiver a possibilidade de contrar com a colaboração de outra pessoa em determinados períodos da vida, poderá não amadurecer certas funções intelectuais e, mesmo tendo essa pessoa, isso não garante, por si só, o seu amadurecimento. (PRESTES, 2021, p.205)

Para Vygotsky (2007), ter uma zona de desenvolvimento iminente é algo que animais não possuem e por isso seriam incapazes de aprender da forma como humanos o fazem: "o aprendizado humano pressupõe uma natureza social específica e um processo através do qual as crianças penetram na vida intelectual daqueles que as cercam" (VYGOTSKY, 2007, p.101).

Sobre o que seria aprendizagem para Vigostski, Prestes (2021, p.218) coloca que diversas traduções usam ora aprendizagem, ora ensino ou os dois termos juntos para se referir ao que o autor, na verdade, tratava como um processo simultâneo que envolve instrução, estudo e aprender por si mesmo. Prestes (2021, p.220) também argumenta que o termo aprendizagem não consegue transmitir o que o conceito original em russo (*obutchenie*) contém em seu significado e que algumas interpretações do termo não têm uma ênfase no processo, dando um peso maior ao seus resultados.

[16] Apenas a título de informação, em outras traduções são usados outros termos para se referir as outras crianças que auxiliam no processo e que não são os adultos.

Para Wertsch e Sohmer (1995, p.333), *obutchenie* é mais próximo do termo instrução[17] do que de aprendizagem[18] no contexto da lìngua inglesa. Entretanto, Wertsch e Sohmer (1995, p.333) destaca que instrução, no inglês, tem um foco naquilo que o instrutor faz, não dando muito destaque para as outras questões e que a forma em que Vigostki usa *obutchenie* envolve tanto professores e estudantes em uma colaboração ativa.

Prestes (2021, p.220) faz observações similares e também prefere o termo instrução a aprendizagem para traduzir *obutchenie*, mesmo levando em conta a carga negativa que o termo instrução tem no contexto brasileiro. Mas, mais imporante, *obutchenie* tem em si uma dimensão que "leva em conta o conteúdo e as relações concretas da pessoa com o mundo" (PRESTES, 2021, p.220). Com isso, ela oferece que uma definição que julga ser mais próxima do que Vigotski queria ao empregar o termo *obutchenie*:

> uma atividade autônoma da criança, que é orientada por adultos ou colegas e preessupõe, portanto, a participação ativa da criança no sentido de apropriação dos produtos da cultura e da experiência humana. (PRESTES, 2021, p.220)

Vigotski estudava o desenvolvimento de crianças quando falava na zona de desenvolvimento iminente. Boa parte dos experimentos, palestras e textos de Vigotski tem seu foco voltado para crianças (objeto de estudo privilegiado pela pedologia), enquanto os materiais apresentados neste trabalho tiveram como público adolecentes e adultos.

No final das contas, Vigotski oferece para este trabalho um papel para o professor no processo de ensino e aprendizagem, ou de instrução formal. Esta figura seria alguém que atua na medição entre a cultura e indivíduo de forma a trabalhar dentro da zona de desenvolvimento iminiente em uma atividade social e ativa do indivíduo que neste processo deve ser incorporada aos seus processos psicológicos.

3.4 O DIALOGISMO PARA O CÍRCULO DE BAKHTIN

Círculo de Bakhtin é o nome dado a um grupo de intelectuais que se reuniu regularmente entre 1919 e 1929 em Nevel, Vitebsk e Leningrado. Comtemporâneos de Vigotski, o Círculo enfrenta o mesmo contexto político e com as mesmas questões. O grupo era constituído por pessoas de formações diversas que incluía pessoas da filosofia, biologia, música e estudiosos

[17] *instruction*
[18] *learning*

da literatura, com destaque para três nomes: Mikhail Bakhtin[19], Valentin Voloshinov[20] e Pavel Medvedev[21] (FARACO, 2013, p.11).

Faraco (2013, p.100) coloca que as questões de linguagem no Círculo foram abordadas, principalmente, por Bakhtin e Voloshinov. Elas não são apresentadas em um texto específico, mas trabalhadas em diversas obras, sendo a segunda metade da década de 20 do século XX um dos períodos mais relevantes para esta temática. Este período foi interrompido pela dispersão do grupo. Bakhtin retoma as discussões anos mais tarde, inclusive tratando da questão em três de seus textos inacabados: "O Problema dos Gêneros do Discurso"(1952-53) "O Problema do Texto" (1959-61) e "Para uma metodologia das Ciências Humanas" (1974).

Em síntese, o Círculo tem uma compreensão da linguagem como dialógica. Para Sobral (2009, p.32), isto significa que a linguagem tem seus sentidos produzidos dentro de uma intersubjetividade em situações concretas de seu exercício. A sujetividade é entendida dentro de termos psíquicos, sociais e históricos, sendo formada e formadora em um processo de intercâmbio verbal. Além disso, a linguagem deve ser vista como atividade e não como um sistema de signos, em contraposição a outras visões de linguagem, principalmente as ligadas a uma vertente da linguística daquela época. A linguagem também não é vista como uma manifestação puramente psicológica, o que implica na necessidade da figura do outro para realizar-se em sua totalidade.

Esta forma de ver a linguagem, para Faraco (2013), já aparecia em "Filosofia do Ato" (BAKHTIN, 2017 [1919/1921]), principalmente, em três elementos: 1) A perspectiva avaliativa da relação entre sujeito e mundo; 2) A relação entre eu/outro e; 3) A unicidade dos eventos do mundo da vida. Na sequência, vamos usar estes três tópicos para desenvolver a visão dialógica de linguagem e suas relações com o trabalho proposto.

A primeira questão envolve o distanciamento da visão abstrata da linguagem, pois como afirma Bakhtin ([1930] 2015, p.69): "A língua[22], para a consciência em que ela vive, não é um sistema abstrato de formas normativas, mas uma opinião concreta e heterodiscursiva sobre o mundo." (BAKHTIN, [1930] 2015, p.69). Nesta passagem, temos de forma clara uma recusa

[19] Mikhail Bakhtin (1895-1975) tem sua formação em estudos literários, trabalhou como professor e foi preso em 1929. Exilado no Cazaquistão, pode apenas assumir um emprego permanente após a Segunda Guerra como professor de literatura do Instituto Pedagógico de Saransk. Se aposenta em 1969 e vem a falecer em 1975.

[20] Valentin Volóchinov (1895-1936) trabalhou como professor e tinha interesse em história da música. Faleceu vítima de tuberculose. (FARACO, 2013, p.13).

[21] Pavel Medvedev (1892-1940?) formou-se em direito e teve carreira como educador e gestor cultural. Trabalhou no Instituto Pedagógico Herzen. Foi vítima dos expurgos stalinistas da década de trinta e possivelmente morreu em 1940. (FARACO, 2013, p.13).

[22] O tradutor optou em usar língua quando se trata do sistema de comunicação geral de um povo e linguagem como emprego individualizado ou literário. Para mais informações, ver a explicação em Bezerra (2015, p.247-248).

em ver a língua ou a linguagem como um sistema abstrato junto com a ênfase em sua dimensão avaliativa (opinião concreta) e sua caracterização como heterodiscursiva, que seria:

> A estratificação interna de uma língua nacional única em dialetos sociais, modos de falar de grupos, jargões profissionais, as linguagens dos gêneros, as linguagens das gerações e das faixas etárias, as linguagens das tendências e dos partidos, as linguagens das autoriadades, as linguagens dos círculos e das modas passageiras, as linguagens dos dias sociopolíticos e até das horas (cada dia tem sua palavra de ordem, seu vocabulário, seus acentos). (BAKHTIN, [1930] 2015, p.30)

Faraco (2013, p.58) coloca que, para Bakhtin, mais do que uma heterodiscursividade,[23] o que importa seria a dialogização destes discursos (Faraco (2013) usa vozes sociais), isto é, o encontro entre elas e a dinâmica que se estabelece: "elas vão se apoiar mutuamente, se interiluminar, se contrapor parcial ou totalmente, se diluir em outras, se parodiar, se arremedar, polemizar velada ou explicitamente e assim por diante" (FARACO, 2013, p.58).

As línguas que formam o heterodiscurso "são pontos de vista específicos sobre o mundo, formas de compreensão verbalizada, horizontes concreto-semânticos e axiológicos específicos" (BAKHTIN, [1930] 2015, p.67). Estas estratificações recobrem os objetos do mundo com enunciados[24] fundamentalmente avaliativos. Cada novo enunciado encontra seu objeto "já difamado, contestado, avaliado, envolvido ou por uma fumaça que o obscurece ou, ao contrário, pela luz de discursos alheios já externados a seu respeito" (BAKHTIN, [1930] 2015, p.48).

Considerar a dimensão avaliativa da linguagem mais importante tem implicações para o entendimento do que é e como se aprende e se ensina a programação de computadores. Como colocamos anterioremente, entender o abstrato da forma como a Computação o faz, sem deixar clara as filiações ao formalismo matemático, de um ponto de vista baktiniano, é uma posição avaliativa.

A forma da Computação trabalhar com a abstração funciona de um jeito muito parecido com a forma descrita por Bakhtin ([1930] 2015, p.39-40) quanto ao funcionamento da língua única da linguística:

> A língua única e comum é um sistema de normas linguísticas. Contudo, essas normas não são um imperativo abstrato, mas *forças criadoras* da vida da língua, que superam o heterodiscurso da linguagem, unificam e centralizam o pensamento verboideológico, criam no interior da língua nacional heterodiscursiva um núcleo linguístico firme e estável da língua literária oficialmente reconhecida ou protegem essa língua já formada contra a pressão do crescente heterodiscurso. (BAKHTIN, [1930] 2015, p.40)

[23] Faraco usa Heteroglossia para ser referir a esta questão.
[24] O conceito de enunciado é trabalho com mais detalhe em uma seção do próximo capítulo.

Neste trecho, Bakhtin argumenta que a língua única é também uma posição avaliativa que tenta controlar as formas de discurso de uma língua de forma regulatória. Por exemplo, os paradigmas de programação, em especial aquilo que apresentamos no capítulo anterior, parecem ser uma manifestação clara de tentativas de estabilizar uma certa heterodiscursividade na escrita de programas de computador. Organizar formas de escrever um programa por paradigmas permite não levar em conta quem escreve ou para quem se escreve um código-fonte.

Entender a linguagem como dialógica exige que não sejam deixados de lado locutores e interlocutores no processo. Para Bakhtin ([1930] 2015), o outro está sempre presente na palavra proferida, mesmo que não apareça como discurso direto. Amorim (2001, p.139) destaca este aspecto: "não é a presença real e física de dois locutores e de dois enunciados que constitui o princípio dialógico, mas sim a presença de duas ou mais vozes no inteior de um mesmo enunciado de um mesmo locutor".

Assim, toda enunciação sempre está a responder enunciados anteriores, pois nos escritos do Círculo, "o acesso à realidade é sempre mediado pela linguagem e que o real apresenta-se sempre semioticamente"(FIORIN, 2016, p.22). Todo objeto é recoberto por um emaranhado de enunciados com posições valorativas que interditam o acesso ao dado "puro" (FARACO, 2013, p.49) e esses têm uma carga valorativa fruto das estratificações heterodiscursivas. Isto implica que uma enunciação sobre um objeto está sempre a interagir com os enunciados que vieram antes, mas também com aqueles que virão depois.

Bakhtin ([1930] 2015, p.52) coloca que "o discursos falado vivo está voltado de modo imediato e grosseiro para a futura palavra-resposta: provoca a resposta, antecipa-a e constrói-se voltado para ela". Ou seja, todo discurso concreto sempre leva em conta um interlocutor, mesmo que figurativo, no momento de sua produção. Isto acontece devido ao fato do interlocutor ser uma entidade ativa na interação. Bakhtin ([1930] 2015, p.54) ainda afirma que "a filosofia do discurso e a linguística conhecem apenas a interpretação passiva da palavra, e ademais predominantemente no plano da língua comum".

Quanto à escrita de programas de computador, podemos identificar estas questões de duas formas. Primeiro, a relação com enunciados anteriores é bem explícita. Ao tentar resolver *bugs* de *software* de tamanho médio ou grande, é comum conversar sobre trechos de código que não foram escritos pelas pessoas que devem resolvê-los. Algo similar acontece nas revisões de código antes desses serem incorporados ao sistema.

Não só isso, mas muitas vezes estas referências a discursos anteriores é hiperatrofiada como é o caso de trechos de código copiados da internet. Copiar códigos de programas da internet é uma atividade válida e corriqueira na programação de computadores. Entretanto, é preciso filtrar, entender o que está disponível e julgar o quanto o código é relevante para o contexto no qual se está trabalhando.

Já sobre a orientação do discursos para a resposta, pode-se dizer que também podemos observar esta questão na escrita de códigos para programas de computador. Ao escrever um código em um teste de seleção para uma vaga de emprego específica ligada ao desenvolvimento de *software*, o candidato irá levar em conta que será avaliado por alguém da área e irá escrever um código que, pelo menos, mostre que ele está filiado aos "padrões" do que é considerada escrita de código profissional.

No caso de se escrever um código para abordar problemas de outros campos, tais como economia ou logística, não existe uma necessidade de mostrar o domínio das formas específicas usadas em densenvolvimento de sistemas, uma vez que o código é escrito para alguém que não dá muita importância para esta questão. Entretanto, elementos como entrada de dados e saída ou até mesmo formas de resposta do programa ainda sim são pensados levando em conta os interlocutores.

Em sala de aula, ocorre o mesmo. Do lado de discentes, ao entregar um código para o professor, aquele é pensado dentro do contexto acadêmico e instrucional. Mesmo que o professor seja alguém com mais experiência na escrita de códigos, assim como seria um desenvolvedor supervisor em uma empresa, os códigos escritos são diferentes, pois antecipam respostas de interlocutores diferentes parte de estruturas institucionais diferentes, com pressões e liberdades diferentes.

Do lado da docência, também existe esta questão. Ao disponibilizar códigos para os ou as estudantes, seja como parte de uma atividade ou como exemplo, existe a mesma antecipação a possibilidade de respostas que moldam a própria enunciação. Disponibiliza-se códigos considerados "importantes" ou acessíveis àqueles na posição de discentes.

Por fim, a unicidade dos eventos do mundo da vida tratada em "Para uma filosofia do ato responsável", texto de Bakhtin do início da decada de 1920, há atrelada em si uma discussão sobre um posicionamento ético ou moral. Como explica Faraco (2017, p.152), Bakthin critica os sistemas éticos material e formal, vendo neles "o mesmo defeito do teoricismo, da pretensão universal se deduz necessariamente a minha ação" (FARACO, 2017, p.152).

Bakhtin (2017, p.96) entende a experiência, o existir, como algo único e irrepetível. Ele afirma que “neste preciso ponto singular no qual agora me encontro, nenhuma outra pessoa jamais esteve no tempo singular e no espaço singular de um existir único”. Além disso, “tudo o que pode ser feito por mim não poderá nunca ser feito por ninguém mais, nunca”. Bakhtin (2017, p.96)

Para Bakhtin (2017, p.96), “a singularidade do existir presente é irrevogavelmente obrigatória”, não existe um alibi para a própria existência. Ou seja, como complementa Sobral (2019, p.160), “sem álibi, responsabilizar-se por seus atos, responsabilizar-se diante do outro, o que ele faz ao reconhecer e afirmar seus atos, ou seja, ao apor sua assinatura a esses seus atos”. Sobral (2019, p.160).

A questão da responsabilidade criada pelo “não-álibi” da existência é um dos conflitos centrais que levam às atividades apresentadas no Capítulo 4. Inclusive tem influência quanto à forma apresentada, já que o ato responsável seria “sempre único e irrepetível, e por isso mesmo, só possível apreendê-lo de seu interior, só é possível descrevê-lo participativamente”. (FARACO, 2017)

Estas atividades são as respostas em forma de atos responsivos, frutos de um entendimento de que a situação era algo único, irrepetível e singular. O não-álibi acaba por exigir que o posicionamento a partir deste entendimento seja responsável no sentido baktiniano do termo, possibilitando conciliar o repetível (Computação) com o singular (Contexto e pessoas). Faz com que o juízo de certo e errado não seja algo a priori e teoricizado, mas um ato concreto e em resposta ao contexto único.

Ou seja, não ter um álibi da própria existência significa que ao se deparar com estudantes que tenham dificuldades em questões relacionadas à programação de computadores que não fazem parte da ementa da disciplina, não pode ser ignorado sob a justificativa de imperativos teoricistas. Alegar que “isto é problema de outra disciplina” ou que não é parte do escopo de trabalho docente é tentar resolver a questão do interior da perspectiva teoricizada. Para Bakhtin (2017, p.49): “Qualquer que seja a tentativa de superar o dualismo entre consiência e vida, entre o pensamento e a realidade concreta singular, é do interior do conhecimento teórico, absolutamente sem esperança”.

Nas seções a seguir, apresentamos uma forma de entender a atividade de programar um computador que seja compatível com o que foi apresentado até aqui. Primeiro, apresentamos como uma visão dialética da lógica pode contribuir e depois mostramos que o conceito de

paradigmas de programação, relativamente bem difundido no ensino de programação, carrega em si as cargas abstratas já comentadas.

3.5 OS FAZERES COMPUTACIONAIS

Nas seções anteriores, mostramos que o pensamento computacional é um conceito importante em pelo menos uma parte das iniciativas recentes sobre ensino e aprendizagem de programação de computadores. Também fizemos um pequeno resgate histórico na tentativa de mostrar que, uma vez não muito bem definido, o pensamento computacional representa uma espécie de senso comum e que, na verdade, é um processo muito mais restrito e recorrente do que se dá a entender. Pensamento computacional nada mais é do que uma roupagem nova para os processos de formalização e abstração encontrados nas disciplinas de Lógica, de Linguística e outras ciências, representados dentro da computação pela lógica formal.

É preciso lembrar que o problema não está na lógica formal em si, mas na redução de qualquer processo a ela. Em especial, damos destaque às tentativas de universalização dos processos derivados da lógica formal, sem levar em conta outros processos históricos e culturais. Em outras palavras, para nós, o problema é acreditar que somente a lógica formal leva a uma maestria aos processos de trabalho ligados à computação e que a lógica formal *per se* é uma espécie de forma de conhecimento que pode ser aplicado a qualquer coisa sem precisar levar em conta nada além de si mesma.

Para Lefebvre (1991), a lógica formal é uma parte de um processo maior, o da lógica dialética. A lógica formal seria “um dos sistemas de redução de conteúdo, através do qual o entendimento chega às "formas"sem conteúdo, às formas puras e rigorosas, nas quais o pensamento lida apenas consigo mesmo, isto é, com "nada"de substancial. No limite extremo, essas formas se desvanecem, tornam-se o vazio, o nada de pensamento e de realidade, o absurdo. ” (LEFEBVRE, 1991, p.132). Esta redução, dentro das propostas de Lefebvre (1991, p.131), seria uma

> “negação dialética, que ainda envolve o que é negado; e isso de tal modo que a operação que restabelece a totalidade positiva (agora analisada e compreendida) é não apenas possível, mas também exigida pela redução dialética do conteúdo”. (LEFEBVRE, 1991, p.131)

Com isso, Lefebvre (1991, p.131) define que o processo tem duas partes intimamente opostas e complementares: “a redução do conteúdo (abstração) e o retorno para o concreto”. É importante notar que a forma como o processo é definido, não é possível separar as duas partes. A abstração, ou aplicação da lógica formal no caso do ensino e aprendizagem de programação de

computadores, exige a volta para o concreto para que seja completa. Não custa também lembrar que este processo é o mesmo descrito por Meksenas (1992).

De certa maneira, diversas das iniciativas dentro do WAlgProg levantadas nas seções anteriores acabam fazendo esta volta ao concreto. Entretanto, o que o enquadramento de Lefebvre (1991) e Meksenas (1992) permite é o planejamento da etapa de redução já levando em conta a existência deste retorno. Permite o uso da lógica formal como uma forma de "designar o concreto mais concreto" (ALTHUSSER, 2019, p.75) e deixa clara a necessidade da lógica formal levar em conta o conteúdo em suas aplicações.

Lefebvre (1991, p.80) também identifica uma relação de parentesco estreito entre lógica e gramática, colocando que a segunda "deixa de lado o sentido, o conteúdo, a verdade ou a falsidade da afirmação" e ocupa-se somente "da correção da linguagem, ou seja, da conformidade com as regras".

As gramáticas também são parte de uma das áreas de interesse da Ciência da Computação, ficando dentro do guarda chuva da disciplinas ligadas à teoria da computação e são estudadas sob a influência dos estudos formalistas com base na matemática e/ou linguística[25]. É neste agrupamento que são estudadas as máquinas de Turing, linguagens formais e autômatos.

Como dito até então, o campo da computação preocupa-se muito mais com a primeira parte do processo da lógica dialética: a uma redução baseada em formalismo matemático. Os estudos de gramática da teoria da computação refletem este foco e exemplificam a relação estreita entre lógica e gramática colocada por Lefebvre (1991, p.80). Esta visão de gramática abstrata e formalista matemática é a visão que informa os estudos e desenvolvimento de linguagens de programação e, por consequência o ensino e aprendizagem dessas. Assim, em uma tentativa de entender as gramáticas de uma forma mais concreta, logo compatível com a perspectiva teórica apresentada até então, recorremos à discussão sobre o tema dentro do campo de ensino e aprendizagem de escrita de textos.

Deste modo, Franchi (2008), ao discutir a recepção da gramática nas escolas, permite enquadrar a questão de uma forma mais abrangente que o ponto de vista da teoria da computação. Sua abordagem não necessariamente abre mão ou ignora a gramática como um sistema de normas, o que permite um diálogo mais direto com a questão do ensino e aprendizagem de

[25] É possível identificar ao longo da história da Ciência da Computação a influência de outras disciplinas no estudo das gramáticas. mas nem sempre alinhadas com uma "direção" covergente com o resto do corpo teórico do campo. Para uma visão rica deste processo histórico, ver o capítulo 1 em Merkle (2002).

linguagens de programação. Ele inicia definindo duas atitudes quanto à gramática: uma normativa e uma descritiva.

Gramática normativa seria o "conjunto sistemático de normas para bem falar e escrever, estabelecidas pelos especialistas, com base no uso da língua consagrado pelos bons escritores."(FRANCHI, 2008, p.16). Para Franchi (2008, p.18), os gramáticos normativos reconhecem a existência de diferentes modalidades e dialetos no uso da linguagem, que são dependentes de condições regionais, de idade, de gênero sexual e, principalmente, de condições sociais. Entretanto, ele também lembra que não existe uma valorização destas modalidades explicada por diversos tipos de preconceitos subjacentes de todo tipo.

As linguagens de programação possuem um forte componente normativo, reforçado pelo ecossistema de artefatos com os quais quem escreve um programa deve interagir. Em especial, os compiladores e interpretadores fazem com que as "gramáticas" criadas pelos projetistas de uma linguagem se façam presentes e ativas na escrita de um programa de computador.

Entretanto, mesmo dentro dos códigos de programas que "compilam", existe um julgamento do que é um código bom e um código ruim. Diferentes métricas podem ser evocadas para julgar a qualidade de um código: legibilidade, eficiência quanto ao processamento, eficiência quanto ao espaço ocupado na memória etc.

Por exemplo, em Wiese *et al.* (2017, p.41), existe o uso do termo *"beautiful code"* para designar um código que "é elegante, eficiente, idiomático e revelador da intenção do algoritmo [26]" e "faz com que a linguagem pareça feita para o problema". Não é preciso ir muito a fundo para entender a existência de um certo grau de subjetividade nisto, inclusive reconhecida pelos autores ao comentar sobre a existência de métricas qualitativas e quantitativas sobre *"beautiful code"*, "mas isto não elimina o julgamento subjetivo de programador experiente, incluindo a habilidade de reconhecer onde melhorias em "beleza" são possíveis[27]".

Em Java, é comum que o nome das classes tenham a primeira letra em caixa-alta, enquanto os métodos comecem em caixa-baixa. Tanto uma classe que comece com uma letra em caixa-baixa ou um método que comece em caixa-alta, ambos estão corretos de um ponto de vista sintático exigido pelo compilador. Entretanto, as convenções de nome da linguagem definiram que tal abordagem é a "mais correta". É possível ver esta recomendação em um documento técnico da linguagem de 1997 (MICROSYSTEMS, 1997), sua permanência no arquivo online

[26] Is elegant, efficient, idiomatic, and revealing of design intent.

[27] But these do not eliminate an expert programmer's subjective judgment, including the ability to recognize where improvements in beauty are possible.

da Oracle atualizado pela última vez em 1999 (ORACLE, 1999), sua recomendação nos guias de estilo[28] (GOOGLE, 2018) da Google para Java e na grande maioria de tutoriais sobre algum aspecto da linguagem encontrados na internet.

Já na linguagem C#, um método deve seguir o que a Microsoft chama de *PascalCase*,[29] o que implica que o método deve ter seu nome iniciado em caixa alta. As classes em C# também seguem o mesmo padrão. Em Java e em C#, o uso de caixa alta e baixa em nomes de classes e métodos não têm implicações quanto ao desempenho computacional (processamento e memória) do programa. Um dos argumentos fortes para os guias de estilo é a necessidade de consistência na escrita (ROSSUM, 2001). Entretanto, a amplitude que o padrão deve ter não necessariamente atravessa o domínio de uma linguagem, mesmo com sintaxes muito próximas, como pode ser visto neste caso.

A notação húngara é uma forma de nomear funções e variáveis em que o tipo da dessas deve fazer parte de seu nome. Por exemplo, uma variável do tipo *String* que representa o título de um texto poderia ser chamada de *sTitle*[30]. Este tipo de notação foi amplamente utilizada no sistema operacional *Windows* e em suas aplicações, uma vez que o seu proponente, Charles Simonyi, trabalhou na empresa de 1987 até 2002 e foi responsável por diversas iniciativas dentro da empresa.

De acordo como próprio Charles Simonyi, a ideia da notação húngara surgiu como uma forma de introduzir "tipagem" em linguagem de máquina (BOOCH, 2008, p.17). Inclusive, guias de estilos mais recentes da *Microsoft* não recomendam formas tipicamente associadas à notação húngara (MICROSOFT, 2021), mostrando a sensibilidade ao tempo que estas convenções possuem.

Ou seja, convenções de nome e guias de estilo são normas estabelecidas socialmente assim como as regras presentes nos compiladores e interpretadores. São criadas em contextos sociais muito próximos, mas mediadas por outras vias. Enquanto os primeiros têm como suporte ferramentas, as convenções e guias de estilo funcionam mais como diretrizes ou leis. Ambas representam bem a gramática normativa de Franchi (2008), mas operam em níveis diferentes em cima do mesmo objeto.

Sobre *PascalCase* e *camelCase* usados em C#, é um caso que representa com precisão a definição de Franchi (2008), uma vez que Brad Abrams e Anders Heilsberg, dois dos projetistas

[28] *Style guides.*
[29] Para uma visão geral sobre o tema, ver: https://en.wikipedia.org/wiki/Naming_convention_(programming).
[30] Para mais detalhes de como a notação funciona, ver Simonyi (1999).

originais do Turbo Pascal, definiram, talvez não sozinhos, que seriam essas as convenções de nome usadas no *framework* .NET (ABRAMS, 2004).

O outro tipo de gramática que Franchi (2008) define é a descritiva que seria:

> Gramática é um sistema de noções mediante as quais se descrevem os fatos de uma língua, permitindo associar a cada expressão dessa língua uma descrição estrutural e estabelecer suas regras de uso, de modo a separar o que é gramatical do que não é gramatical. "Saber gramática" significa, no caso, ser capaz de distinguir, nas expressões de uma língua, as categorias, as funções e as relações que entram em sua construção, descrevendo com elas sua estrutura interna e avaliando sua gramaticalidade. (FRANCHI, 2008, p.22)

A gramática descritiva parece mais neutra que a normativa, mas Franchi (2008, p.22-23) alerta que ela pode introduzir de forma sorrateira um ponto de vista normativo de duas formas: Primeiro, na seleção de que fatos devem ser considerados pela gramática. E segundo, usar de critérios sociais para excluir como não gramatical tudo aquilo que não considerar como "uso consagrado". Inclusive, para ele, na prática escolar, "a gramática descritiva se transforma em um instrumento para as prescrições da gramática normativa"(FRANCHI, 2008, p.23).

De conhecimento destas duas formas de entender a gramática, Franchi (2008) apresenta uma terceira, mais contemporânea[31] que leva em conta a insuficiência das perspectivas normativa e descritiva. Assim, ele parte do pressuposto de que a linguagem é uma propriedade ou habilidade humana que independe de contexto. Não é "aprendida", desenvolve e amadurece desde que seja assegurado à criança acesso a manifestações linguísticas. Com isso, "todo falante, independentemente da modalidade de linguagem de que se sirva, possui uma gramática interna" (FRANCHI, 2008, p.25) ou "a interioriza já em tenra idade, a partir de suas próprias experiências linguísticas" (FRANCHI, 2008, p.25). Isto tudo implica em que "toda a criança já chega à escola dominando com perfeição uma complicadíssima gramática"(FRANCHI, 2008, p.25).

Esta terceira gramática de Franchi (2008) corresponde "ao saber linguístico que o falante de uma língua desenvolve dentro de certos limites impostos pela sua própria dotação genética humana, em condições apropriadas de natureza social e antropológica (FRANCHI, 2008, p.25). Esta terceira forma diferente das duas primeiras e não ignora a questão da variação linguística. São reconhecidas as diferenças entre as modalidades culta e coloquial, mas com diferenças significativas quanto à conceituação e atitude (FRANCHI, 2008, p.28).

Reconhecer que existem regras nas construções textuais consideradas coloquiais faz com que o modo culto de escrita não seja o padrão único e certo, mas um outro entre tantos. Se o normativo se impõe no desprezo pelas outras manifestações e o descritivo opera ignorando

[31] O texto original de Franchi é dos anos 80 do sécnolo XX.

diversas formas, reconhecer a possibilidade de várias gramáticas implica uma outra atitude quanto ao ensino de escrita na escola. Se a gramática normativa leva a coibir desvios da norma culta e a descritiva a nomear objetos e relações, o reconhecimento de uma gramática prévia faz com que o objetivo da escola seja levar a "criança a dominar também a modalidade culta da escrita de sua língua, que se realiza, principalmente, oferecendo-se à criança condições, instrumentos e atividades que a façam ter acesso às formas linguísticas diferenciadas e operar sobre elas" (FRANCHI, 2008, p.29). Ainda na mesma linha, mais especificamente, "deve-se diagnosticar a nível de detalhe a realidade linguística de nossos alunos, o estágio em que se encontram para levá-los a ampliar suas experiências linguísticas e suas hipóteses gramaticais; para fazê-los dispor não somente de uma gramática passiva, mas de uma gramática cada vez mais rica e operativa" (FRANCHI, 2008, p.31).

Isto significa que "não faz sentido contrapor uma linguagem erudita a uma linguagem vulgar, nem tentar substituir uma pela outra. Trata-se de levar a criança a dominar uma outra linguagem, por razões culturais, sociais e políticas bastante justificáveis" (FRANCHI, 2008, p.30).

Pelos exemplos dados nesta seção, tanto as perspectivas da gramática normativa quanto as da descritiva estão bem presentes na Ciência da Computação. A primeira pelos diversos *style guides* e as chamadas boas práticas e a segunda, de uma forma um pouco mais rígida, em especial, nos *frameworks* e compiladores, ainda que nestes últimos existam diversos níveis, desde aqueles que impedem a compilação até os *warnings* que são apenas "sugestões" de trechos que podem ser problemáticos.

É importante ver que as gramáticas normativas e descritivas representam uma forma de abstração que, em um primeiro momento, são estáticas e não carregam em si um dimensão de movimento e mudança. Ambas representam um modelo de conhecimento que pode ser transferido, adquirido e não necessariamente contém em si uma dimensão processual.

Já a terceira forma de ver gramática leva em consideração não só os termos de uma língua e suas regras, mas também o processo de sua aquisição junto com as pessoas que participam deste processo. É uma visão voltada ao processo de ensino e aprendizagem, além de ser compatível com o processo de abstração apresentado neste texto até então.

A discussão de Franchi (2008) sobre estes três modos de ver gramática no ensino, de certa maneira, se mostra a mesma que Bakhtin ([1930] 2015) estava a fazer sobre heterodiscurividade, a língua única da linguística.

Se, como Meksenas (1992) coloca, “as diferentes práticas de ensino partem sempre de certos níveis de concretude, chegando, pela abstração, a outros níveis de concretude”, o ensino de programação de computadores, algo que articula em si a lógica formal, deveria fazer o mesmo para trabalhar com tal perspectiva.

Floyd (1979, p.458), ao discutir sobre formas de programar, reconheceu esta questão: “Se eu pergunto para outro professor o que ele ensina na disciplina de introdução à programação, se ele responder orgulhoso “Pascal” ou timidamente “FORTRAN”, eu sei que ele está ensinando gramática, um conjunto de regras semânticas[32] e alguns algoritmos terminados, deixando aos estudantes a descoberta, por eles mesmos, alguma estratégia projetual[33].”

A gramática normativa e a descritiva oferecem um modelo compatível com a “educação bancária” descrita por Freire (2014). Para tentar se afastar desta perspectiva dentro do ensino e aprendizagem de programação de computadores, propomos definir algo chamado de fazeres computacionais. Entendemos o termo como uma forma de compreender e ensinar que privilegiem uma visão integrada do que é o processo de programação de computadores. Desta forma, a abstração é um processo de mediação entre formas diferentes de concretude. Dentro desta perspectiva, o ensino de programação de computadores e de lógica formal deve levar em conta não só o processo de abstração, mas o contexto concreto do qual se parte (concretude das e dos estudantes, professoras e professores) e a concretude a qual se almeja chegar (uma espécie de dialética resultante do encontro do contexto concreto com as teorias e conceitos vindos do campo computacional).

Também propomos entender este processo levando em conta sua dialogicidade, especialmente na perspectiva bakhtiniana que apresentamos, a qual necessariamente exige a presença da alteridade. De forma complementar, também entendemos o processo de abstração ou da instrução de abstração da mesma forma como Vigostki entende a aquisição das funções psicológicas superiores: Como um processo intra e inter pessoal mediado por signos, instrumentos e pessoas.

Estas duas perspectivas teóricas deixam clara uma necessidade de que o processo de abstração e, por consequência, do ensino e aprendizagem de programação de computadores, precisa levar em conta junto com o objeto a ser abstraído tanto aquele que abstrai quanto para quem esta abstração se dirige. Pelo que apresentamos até então, parte significativa do processo de

[32] Aquilo que Floyd (1979) chama de semântica não é necessariamente o que as teorias usadas neste trabalho chamariam de semântica. A semântica de Floyd (1979), pelo menos neste trecho, parece ser mais compatível com o que entendemos como sintaxe.

[33] *If I ask another professor what he teaches in the introdutory programming course, whether he answers proudly “Pascal” or diffidently “FORTRAN”, I know he is teaching a grammar, a set of semantic rules, and some finished algorithms, leaving the students to discover, on their own, some process of design.*

ensino e aprendizagem de linguagens de computação leva em conta somente a visão formalista e dedutiva da abstração e tem como foco o objeto da operação.

Isso tem implicações quanto à escrita de códigos-fonte para programas de computador. Primeiro, um trecho de código deve ser sempre pensado em relação a suas práticas e contexto. Por exemplo, uma estrutura condicional (if/else) aplicada ao desenvolvimento de *frontend* em programação *web* usando um *framework* declarativo envolve uma série de elementos muito diferentes do que usar a estrutura condicional em um *comand line interface* ou até mesmo um jogo. A estrutura pode ser a mesma em todos, mas o contexto e o tipo de programa, nestes casos, tornam o uso relativamente diferente.

Segundo, um trecho de código é parte de uma rede composta por outros "textos/códigos" e pessoas. Deve-se sempre entendê-los dentro desta rede. Por exemplo, uma prática comum ao se deparar com o *bug* ao usar um *framework* é digitar o erro ou parte dele em um buscador para verificar se o site *stackoverflow.com*[34] tem alguma resposta para o problema. Desta maneira, a escrita de código não é somente saber a sintaxe e uso adequado do *framework*, mas todo o processo que envolve sua produção. Isto também envolve saber "ler" códigos, pois nem sempre existe a resposta específica para o seu problema.

Quanto ao conteúdo, argumentamos que escolher conteúdos familiares fornece uma base concreta para que a pessoa que está aprendendo consiga extrair significado sobre o código que escreve. Por exemplo[35], no Capítulo 4 apresentamos uma atividade de programação orientada a objetos cujo conteúdo são filmes, algo que parece ser familiar o suficiente para os públicos das atividades descritas neste trabalho.

Ainda é importante colocar que não estamos contestando a importância de se entender o funcionamento de compiladores, gramáticas livres de contexto ou pregando contra o entendimento profundo de como uma linguagem de programação funciona. Mas, problematizando uma forma de compreensão teórica diferente para lidar com as dificuldades relatadas no ensino de programação de computadores, segundo alguns aportes de base materialista histórica (ALTHUS-

[34] É um *website* com perguntas e respostas sobre qualquer tipo de código.

[35] Um certo dia, um dos autores deste trabalho estava ajudando estudantes no contraturno dentro do laboratório de informática quando trabalhava como professor temporário. Um estudante que fazia a disciplina de OO com ele, mas estava fazendo um trabalho de banco de dados perguntou: Professor, o que é um departamento? Na hora, o professor sem saber do que se tratava o trabalho sabia que estava sendo usado o exemplo clássico das aulas de BD, o esquema do departamento acadêmico com estudantes, professores, aulas e departamentos. Porém, a estrutura do campus não tem departamentos. Um exemplo que em um primeiro momento parece contextualizado, se mostra abstrato e sem relação com o cotidiano do estudante.

SER, 2019; VYGOTSKY, 1995; LEFEBVRE, 1991) e articulados aos estudos das linguagens (FRANCHI, 2008; BAKHTIN, 2014).

Ou seja, para trabalhar dentro desta perspectiva, os exercícios, projetos, trabalhos e exemplos devem levar em conta a realidade das pessoas envolvidas e propiciar que quem está em processo de aquisição do domínio de uma linguagem de programação consiga aproximar o seu contexto concreto do contexto, também concreto, da computação. Além disso, damos um destaque importante para o papel da mediação no processo que deve levaram em conta as pessoas envolvidas.

Na próxima seção, discutimos estas questões de uma forma mais próxima da escrita de programas de computador, usando como fio condutor o conceito de paradigma de programação.

3.6 O PROBLEMA DOS PARADIGMAS DE PROGRAMAÇÃO

Thomas Kuhn escreveu "A Estrutura das Revoluções Científicas" (KUHN, 2007) em 1962. Originalmente um físico, a partir de então, Kuhn torna-se um autor fortemente associado aos campos da epistemologia e história da ciência.

Um dos conceitos principais deste texto é o de paradigma. Já de início, (KUHN, 2007) define-o como "as realizações científicas universalmente reconhecidas que, durante algum tempo, fornecem problemas e soluções modelares para uma comunidade de praticantes de uma ciência" (KUHN, 2007, p.13). Como exemplos de paradigma, dentro da física, Kuhn (2007, p.30) cita: a astronomia ptolomaica, a dinâmica aristotélica e a óptica corpuscular.

O conceito de paradigma está associado intrinsecamente com o de ciência normal (KUHN, 2007, p.30). Por ciência normal, Kuhn (2007, p.29) entende "a pesquisa firmemente baseada em uma ou mais realizações científicas passadas" que "não têm como objetivo trazer 'à tona novas espécies de fenômeno" (KUHN, 2007, p.44). Ou seja, "está dirigida para a articulação daqueles fenômenos e teorias já fornecidos pelo paradigma" (KUHN, 2007, p.45).

Kuhn (2007) cita três focos que a ciência normal tem: a "classe de fatos que o paradigma se mostrou particularmente reveladora da natureza das coisas" (KUHN, 2007, p.46), os fenômenos que "podem ser diretamente comparados com as predições da teoria do paradigma" (KUHN, 2007, p.46) e, a que considera mais importante, o "trabalho empírico empreendido para articular a teoria do paradigma, resolvendo algumas de suas ambiguidades residuais e permitindo a solução de problemas para os quais ela anteriormente só tinha chamado a atenção"

(KUHN, 2007, p.48). Kuhn (2007, p.55) chama os três de “determinação do fato significativo, harmonização dos fatos com a teoria e, por fim, a articulação da teoria”, respectivamente.

A ciência normal seria uma espécie de “empreendido não dirigido para as novidades e que a princípio tende a suprimí-las” (KUHN, 2007, p.91), ao mesmo tempo que serve como um plano de fundo referencial para o surgimento de inconsistências, as quais ele chama de anomalias, que podem desencadear um processo de superação dentro de um paradigma. Assim, a ciência normal “prepara o caminho para sua própria mudança” (KUHN, 2007, p.92).

Para Kuhn (2007), uma crise dentro de um paradigma ou ciência normal faz com que possam surgir novas teorias dentro de uma ciência. Dos fenômenos possíveis de se aplicar novas teorias, Kuhn (2007, p.131) argumenta que somente as “anomalias reconhecidas, cujo traço característico é a sua recusa obstinada a serem assimiladas pelos paradigmas existentes” realmente podem desencadear o surgimento de novas teorias.

Kuhn (2007, p.125) dá o nome de revolução científica para este processo pelo qual uma teoria é substituída por outra: “consideramos revoluções científicas aqueles episódios de desenvolvimento não-cumulativo, nos quais um paradigma mais antigo é total ou parcialmente substituído por um novo, incompatível com o anterior”.

De forma resumida, Kuhn (2007) entende que uma dada comunidade acadêmica científica específica é unida por um paradigma. O trabalho dentro de um paradigma é um trabalho de resolução de quebra-cabeças a fim de testar a efetividade do paradigma em diversas situações diferentes. Este processo pode levar ao encontro de anomalias que devem ser acomodadas ou não pelas teorias e instrumentos de um paradigma. Caso o paradigma não dê conta da anomalia, Kuhn (2007) argumenta que um período de crise se instala na comunidade. O resultado desta crise pode ser a reacomodação da anomalia que se mostrou um quebra-cabeças mais complicado ou ser realmente um fenômeno que o paradigma atual não tem aparatos para explicar. Se for o segundo caso, um período extraordinário se instala em que diversos grupos têm explicações diferentes para esta anomalia, até que uma destas explicações seja aceita como o novo paradigma da comunidade. Com isso, o período de ciência normal se instala novamente e os rastros do antigo paradigma são descartados ou readequados ao novo.

Existem pelo menos 22 definições diferentes do conceito de paradigma ao longo da “A Estrutura das Revoluções Científicas”. Em um posfácio de 1969, Kuhn (2007, p.228) diz que esta grande quantidade deu-se mais por estilística do que outra coisa. Com isso, ele afirma que todas estas definições podem ser agrupadas em dois grandes usos do termo paradigma:

primeiro, “indica toda a constelação de crenças, valores, técnicas etc. partilhadas pelos membros de uma comunidade determinada” (KUHN, 2007, p.220). Segundo, “um tipo de elemento desta constelação: as soluções concretas de quebra-cabeças que empregadas como modelos ou exemplos, podem substituir regras explícitas como base para a solução dos restantes quebra-cabeças da ciência normal” (KUHN, 2007, p.220).

Para o primeiro grupo, Kuhn (2007, p.228) comenta que ficaria satisfeito em usar o termo teoria no lugar de paradigma, mas dado o seu uso específico como conceito dentro da filosofia da ciência, propõe o uso de matriz disciplinar: “disciplinar porque se refere a uma posse comum aos praticantes de uma disciplina particular; “matriz”, porque é composta de elementos ordenados de várias espécies, cada um deles exigindo uma determinação mais pormenorizada” (KUHN, 2007).

Kuhn (2007, p.228-234) cita quatro componentes da matriz disciplinar ou paradigma. Primeiro, as generalizações simbólicas, “aquelas expressões, empregadas sem discussão ou dissensão pelos membros do grupo, que podem ser facilmente expressas em uma forma lógica” (KUHN, 2007, p.229). São os “componentes formais ou facilmente formalizáveis da matriz disciplinar” (KUHN, 2007, p.229).

O segundo seriam os modelos, “as partes metafísicas dos paradigmas”. Kuhn (2007, p.230) tem em mente os “compromissos coletivos com crenças”, que fornecem analogias ou metáforas de modo a auxiliar em qualificar aquilo que é ou não explicação ou solução de um quebra-cabeça.

O terceiro seriam os valores partilhados por diferentes comunidades. Kuhn (2007, p.231) comenta que, provavelmente, cientistas aderem a valores que dizem respeito a predições: “devem ser acuradas; predição quantitativas são preferíveis às qualitativas; qualquer que seja a margem de erro permissível, deve ser respeitada regularmente em uma área dada”.

O quarto são os exemplares e Kuhn (2007) trabalha-os como o segundo grupo.

Assim, no segundo grupo, como dito, Kuhn (2007, p.228) trabalha o conceito de paradigma como exemplo compartilhado. Ele chama de exemplares as “soluções concretas de problemas que os estudantes encontram desde o início de sua educação científica, seja nos laboratórios, exames ou no fim dos capítulos dos manuais científicos” (KUHN, 2007, p.234). Ele também destaca o papel destes elementos concretos na introdução de alguém a uma disciplina, indo contra uma visão de que “o conhecimento científico está fundado na teoria e nas regras: os problemas são fornecidos para que se alcance destreza daquelas” (KUHN, 2007, p.235).

O argumento de Kuhn (2007) consiste em reconhecer que os exemplos e modelos fornecidos durante o treinamento de um estudante de uma disciplina científica têm um papel fundamental no entendimento de leis e regras gerais da área. Para ele, "depois de resolver certo número de problemas [...] o estudante passa a conceber as situações que o confrontam como um cientista, encarando-as a partir do mesmo contexto (*gestalt*) que os outros membros do seu grupo de especialistas" (KUHN, 2007, 237). Ou seja, de certa maneira Kuhn (2007) vai ao encontro de autores citados anteriormente, tais como Papert (1994) e Meksenas (1992) em reconhecer o papel do concreto não somente como um desdobramento do abstrato, mas constitutivo deste último. Para lidar com esta concretude, Kuhn (2007) recorre ao conceito de conhecimento tácito de Michael Polanyi.

Mesmo que Thomas Kuhn nunca tenha proposto ir além do campo científico e tratar de temas como cultura e sociedade (MENDONçA, 2012), seus termos e conceitos, em especial o de paradigma, foram muito além. Ainda é importante colocar que em "A Estrutura das Revoluções Científicas", Kuhn (2007) limita-se a trabalhar com exemplos da física e da química, mostrando que o seu entendimento de campo científico é restrito e específico.

Vale ressaltar que os conceitos usados por Kuhn (2007) têm em si uma dimensão histórica. Esta concepção está embutida na forma como ele modela as mudanças de paradigmas, em especial nas mudanças descritas no processo de uma crise no modo normal de fazer ciência, a revolução, e subsequentemente, a volta à normalidade dentro de outro paradigma. Entretanto, como será discutido à frente, isto não significa que a recepção de seus termos trabalhe da mesma maneira. Inclusive, o conceito de paradigma, na forma modelada por Kuhn (2007), tem a sua parcela de culpa no entendimento da ciência de si mesma como um empreendimento cumulativo e ajuda a inibir a percepção de uma dimensão histórica nos objetos em que é aplicado.

O uso da palavra paradigma é relativamente bem difundido no campo das linguagens de programação, entretanto isto não significa um consenso. Como será trabalhado a seguir, existem algumas concordâncias e divergências entre os diversos textos que tratam do conceito e muito pouca relação com os outros conceitos que compõe o quadro analítico de Thomas Kuhn (2007).

Stolin e Hazzan (2007, p.65) colocam que "um paradigma é uma forma de ver e fazer coisas, uma quadro de referência de pensamento no qual alguém interpreta o mundo ou realidade[36]". Junto com esta definição, existe o reconhecimento de Kuhn (2007) como difusor do conceito e também uma definição de paradigma de programação com base em diversos

[36] *A paradigm is a way of doing and seeing things, a framework of thought in which we interpret one's world or reality.*

outros trabalhos: “paradigmas de programação são heurísticas usada para solução algorítmica de problemas. Um paradigma de programação formula uma solução para um dado problema quebrando a solução em bases específicas e definindo a relação entre eles[37]” (STOLIN; HAZZAN, 2007, p.65).

Stolin e Hazzan (2007, p.66) mencionam três paradigmas diretamente: o funcional, em que a base são as funções e a relação entre elas se dá por composição. O procedural, em que a base são procedimentos ou sequência de comandos e a relação dá-se por hierarquia. E por fim, o orientado a objetos, em que a base são classes e a relação é dada por herança.

Para Van Roy (2009, p.10), “um paradigma de programação é uma abordagem para programar computadores baseada em teoria matemática ou em um conjunto coerente de princípios[38]” e funciona como uma espécie de generalização de linguagens de programação: “linguagens como Java, Javascript, C#, Ruby e Python são virtualmente idênticas: elas todas implementam o paradigma orientado a objetos somente com pequenas diferenças, pelo menos dentro do ponto de vista dos paradigmas [39]” (VAN ROY, 2009, p.12).

Van Roy (2009, p.13) identifica 27 grandes paradigmas de programação e cria uma taxonomia em que cada um deles é um conjunto de conceitos de programação. Seus paradigmas se relacionam na medida em que um novo paradigma nada mais é do que um anterior com alguma característica nova. Por exemplo, *concurrent object-oriented programing* nada mais é do que a *sequential object-oriented programing* com *threads*[40].

Além disso, o autor também comenta que um paradigma “quase sempre deve ser Turing completo para ser usável[41]” (VAN ROY, 2009, p.13), o que reforça o entendimento de que, para ele, um paradigma é uma generalização das linguagens de programação.

Watt (2004, p.263) trabalha com uma quantidade menor de paradigmas, pelo menos com aqueles que ele considera principais, elegendo seis: a programação imperativa, a orientada a objetos, a concorrente, a funcional, a lógica e o *scripting*.

---

[37] *Programming paradigms are heuristics used for algorithmic problem solving. A programming paradigm formulates a solution for a given problem by breaking the solution down to specific building blocks and defining the relationship among them.*

[38] *A programming paradigm is an approach to programming a computer based on a mathematical theory or a coherent set of principles.*

[39] *Such languagens as Java, Javascript, C#, Ruby and Python are all virtually identical: they all implement the object-oriented paradigm with only minor differentes, at least from the vantage point of paradigms.*

[40] No texto original de Van Roy (2009) existe um diagrama que mostra todas estas relações entre os paradigmas.

[41] *A paradigm almost always has to be Turing complete to be pratical.*

Programação imperativa tem o seu nome derivado do verbo em Latin *imperare*, que significa comandar, pois funciona com base em comandos que atualizam valores guardados na memória (WATT, 2004, p.265).

Além disso, também comenta que a programação imperativa era o paradigma dominante até os anos 1990s do século XX, quando foi "desafiada" pela programação orientada a objetos.

Por programação orientada a objetos, Watt (2004, p.297) entende como uma forma de programar baseada em classes com relações hierárquicas entre si, um passo além do uso de tipos abstratos de dados, presente na programação imperativa. Já a programação concorrente permite a execução de comandos sobrepostos, sejam intercalados em intervalos ou paralelos em CPUs diferentes. A palavra concorrente tem raízes latinas e significa "correr junto". Na programação funcional, a forma básica de trabalho é a aplicação de funções em argumentos. Como conceitos-chave, Watt (2004, p.367) coloca: expressões, funções e polimorfismo paramétrico.

Na programação lógica, são implementadas relações, estas definições mais gerais que um mapeamento, algo que torna, potencialmente, a programação lógica de uma ordem maior que a imperativa ou funcional.

Watt (2004, p.414) caracteriza a programação com *scripts* como paradigma que "usa *scripts* para conectar sub-sistemas[42]", "desenvolvimento rápido e evolução dos *scripts*[43]", "requisitos de eficiência modestos[44]" e "funcionalidades de alto-nível em aplicações específicas para certas áreas[45]".

Wells e Kurtz (1989) propõem um pseudo-código generalista que pode ser traduzido para diversas linguagens como uma solução para o problema que identificam: "Exposição exagerada inicial ao paradigma de programação imperativa pode fazer ser bem difícil introduzir estudantes a outros paradigmas[46]". Como parte de seu embasamento, caracterizam quatro paradigmas e reconhecem-nos como não mutuamente exclusivos: imperativo, "caracterizado por declaração, atribuição e uso: declarações em um bloco precedem uma sequência de *statements*, incluindo atribuição[47]". Funcional, "caracterizada pela construção e aplicação; este tipo de linguagem

[42] *Use of scripts to glue subsystems together.*
[43] *Rapid development and evolution of scripts.*
[44] *Modest efficiency requirements.*
[45] *Very high-level functionality in application-specific area.*
[46] *Initial overexposure to the imperative programming paradigm can make it very difficult to introduce students to other paradigms.*
[47] *Characterized by declare, assign and use: declarations in a block precede a sequence of slatements, including assignment.*

envolve aplicações funcionais e a construção direta de objetos e dados de programa[48]". Orientada a objetos, "caracterizada pelo tratamento uniforme de objetos; estas linguagens enfatizam objetos, classes e herança de propriedades de classes parentes[49]". Lógico, "caracterizado por relações lógicas e busca automatizada de possíveis caminhos que levam à solução: o mecanismo primário é resolução[50]."

Miller (2015, p.34) entende que programação em planilhas é um paradigma e define como uma linguagem que "contém os elementos comuns de uma tabela de células nas páginas. Cada célula é capaz de conter uma expressão que pode fazer referência a outras células[51]".

Mycroft (1996) apenas cita a existência, sem descrevê-los, de vários paradigmas para discutir integrações entre eles: imperativo, orientado a objetos, concorrente, funcional e lógico.

As discussões sobre qual linguagem usar na introdução à programação, como mostram Hartel e Hertzberger (1995), é um dos locais em que existe uma preocupação com paradigmas. Por exemplo, Reinfelds (1995) defende e descreve o uso de três paradigmas (programação lógica, programação funcional e programação imperativa, incluindo as duas primeiras dentro de um grupo chamado programação declarativa) para uma disciplina introdutória de dois semestres para o curso de Ciência da Computação, entendendo que um paradigma é uma ferramenta de solução de problemas.

Zuhud (2013, p.87) vai na mesma linha e entende que um paradigma de programação é uma abordagem para resolver um problema de programação. Ele cita como exemplos: imperativa, procedural, orientada a objeto, lógica, funcional e declarativa.

Beheshti e Gunawardena (2001) descreve o trabalho com laboratórios para o aprendizado de paradigmas de programação, dando ênfase nos quatro que considera principais: imperativo ou procedural, orientado a objetos, funcional e lógico.

Velázquez-Iturbide (2005) discute como ensinar linguagens de programação para calouros e cita como linguagens: funcional, lógica, imperativa (VELÁZQUEZ-ITURBIDE, 2005, p.272) e concorrente (VELÁZQUEZ-ITURBIDE, 2005, p.273). É interessante notar que Velázquez-Iturbide (2005) reconhece tanto programação procedural quanto orientada a objetos

[48] *Characterized by construct and apply; such languages involve functional application and the direct construction of data and program objects.*

[49] *Characterized by the uniform treatment of objects; such languages stress object classes and inheritance of properties from parent classes.*

[50] *Chancterized by logical relations and automated search of possible paths leading to solution: the primary mechanism is resolution.*

[51] *. . . contain the common elements of a grid of cells contained in sheets. Each cell is capable of containing an expression which can reference other cells.*

como imperativa (em um comentário sobre sombreamentos do currículo do curso): “Programação imperativa, seja procedural ou orientada a objetos[52]”.

Reinfelds (1995) defende o ensino de mais de um paradigma, algo também feito por Maheshwari (1996, p.32) “existe uma forte necessidade prática de que estudantes sejam introduzidos a mais de um paradigma[53]”. Inclusive, este último relata uma proposta de exercícios que discentes deveriam usar dois paradigmas diferentes (imperativo e orientado a objetos) para resolver um problema.

Ragonis e Haberman (2010) argumentam que o ensino de dois paradigmas, orientado a objetos e lógico, pode levar a um conhecimento de conceitos-chave e avançados de Ciência da Computação.

Brilliant e Wiseman (1996) investigam a escolha de qual linguagem de programação usar para a introdução em programação de computadores nos cursos de Ciência da Computação. Com base em uma pesquisa feita por eles e literatura da área, foram descritos três paradigmas usados para começar o ensino de programação: procedural (Pascal, C, C++), orientado a objetos (C++) e funcional (Scheme).

Sung e Snyder (2014), em um estudo sobre formas de fazer com que estudantes tenham contato com princípios de Ciência da Computação, usam dois paradigmas: um com base em texto que seria o “método tradicional de desenvolver programas pela entrada de texto em um editor de texto, por exemplo: programar com Java ou C# [54]”, enquanto o outro seria um paradigma de programação com base em diagramas visuais[55], descrito como “a sintetização de soluções programadas através da manipulação pré-definida de ícones em ambientes especializados, por exemplo: programar com Alice ou *Scratch*[56]”.

Existem também os trabalhos que propõe novos paradigmas ou tentam identificar o surgimento de novos. Cacace *et al.* (1990) propõem integrar a modelagem de dados orientada a objetos ao paradigma baseado em regras de banco da dados para criar um sistema estendido de banco de dados. Dvorak *et al.* (2009) usam o conceito de *alternation* como base para um novo tipo de paradigma de computação. Este é um caso interessante, pois mostra a confusão que o uso de um termo um tanto quanto genérico permite. Rowley (2011) chama de *Guidance trees*

[52] *Imperative programming, either procedural or object-oriented.*
[53] *There is a strong practical need that students should be introduced to more than one paradigms.*
[54] *Traditional method of developing programs by entering text in a text editor, e.g., programming with Java or C#.*
[55] *Visual-based programming paradigm.*
[56] *Visual-based programming paradigm” of synthesizing programming solutions via manipulating predefined icons in specialized environments, e.g., programming with Alice, or Scratch.*

um novo paradigma de desenvolvimento que "pode ser melhor descrito como uma mistura de árvores de decisão com fluxos de trabalho[57]".

Fleissner e Baniassad (2008) identificam uma relação entre a programação orientada a objetos e a filosofia grega que, para eles, é a raiz da filosofia ocidental. A partir disto, propõe um paradigma chamado de "Orientado a Harmonia[58]" com base em conceitos (harmonia, ressonância e contexto) que eles acreditam ser uma filosofia oriental, em especial chinesa.

O Quadro 1 sumariza a que paradigmas os textos citados fazem referência. Ele não leva em conta o significado dado pelos autores quanto ao termo, mas a citação da palavra em si, algo evidente na presença de "paralela" e "concorrente" como valores da coluna paradigmas. Não foram adicionados os paradigmas citados por Van Roy (2009), pois a forma escolhida de organizar o quadros junto com a grande quantidade de paradigmas definidos pelo autor, os quais não têm interseção com os dos outros autores, não resultaria em uma adição de informação relevante ao quadro.

Quadro 1 – Matriz de paradigmas de programação e autores que os citam.

Autores	orientada a objeto	imperativa	concorrente	procedural	lógica	scripting	declarativa	paralela	functional	com base em texto	com base em diagramas visuais	sistemas programados	linguagem de programação	com base em regras	guidance trees	planilhas	orientado à harmonia	alternation
Zuhud (2013)	●	●	●	●		●		●										
Watt (2004)	●	●	●	●	●		●											
Mycroft (1996)	●	●	●	●	●													
Velázquez-Iturbide (2005)		●	●	●	●													
Wells e Kurtz (1989)	●	●	●	●														
Beheshti e Gunawardena (2001)	●	●	●	●														
Reinfelds (1995)		●	●	●			●											
Stolin e Hazzan (2007)	●	●				●												
Brilliant e Wiseman (1996)	●	●							●									
Taft *et al.* (2011)					●						●							
Sung e Snyder (2014)										●	●							
Ragonis e Haberman (2010)	●			●														
Maheshwari (1996)	●		●															
Gabriel (2012)												●	●					
Cacace *et al.* (1990)	●													●				
Rowley (2011)															●			
Miller (2015)																●		
Fleissner e Baniassad (2008)																	●	
Dvorak *et al.* (2009)																		●

Fonte: Autoria própria.

[57] *Can best be described as a mix between decision trees and workflows.*
[58] *Harmony-Oriented.*

O Quadro 1 evidencia, pelo menos, duas características sobre os tipos de paradigmas de programação de computadores existentes. Primeiro, uma certa estabilidade quanto ao *status* de paradigma, por hora sem se preocupar muito com o significado disto, dado para a programação funcional, lógica, imperativa e orientada a objetos. Em segundo lugar, mesmo com a regularidade de alguns termos, isto não significa consenso quanto ao significado. Por exemplo, o caso de Velázquez-Iturbide (2005), que considera a programação procedural e a orientada a objetos como imperativas, uma organização que não é a mesma dos demais textos.

É interessante notar que o uso feito do conceito de paradigma tem divergências e aproximações com a versão de Kuhn (2007). Talvez estas aproximações expliquem o porquê da adoção do termo por parte dos estudos sobre linguagens de programação[59].

Sobre o surgimento de novos paradigmas, o que ocorre na programação parece ser parcialmente compatível com a descrição de Kuhn (2007) deste processo. O período de crise dentro da ciência normal faz com que possam surgir novas teorias dentro de uma ciência, ou seja alguns tipos de anomalias exigem abordagens diferentes. A forma como Van Roy (2009) organiza a relação entre seus paradigmas coloca os princípios como as respostas às "saliências" que os anteriores não conseguiam lidar. Por exemplo, o caso da *concurrent object-oriented programing* e da *sequential object-oriented programing* citados anteriormente.

Taft *et al.* (2011, p.165) comentam que duas abordagens fundamentalmente diferentes foram usadas para lidar com o problema da concorrência na programação com múltiplos processadores: criar novas linguagens especialmente adaptadas para estas questões ou estender linguagens já existentes. Nota-se que tanto a criação de um novo paradigma (novas linguagens) quanto o uso de um mais antigo (extensão das existentes) foram as abordagens da comunidade para lidar com a crise (programação de um processador para vários).

Este caso evidencia o porquê de considerarmos parcial a compatibilidade do modelo de Kuhn (2007) com as linguagens de programação. Tanto o velho quanto o novo paradigma fornecem uma solução ao problema, mas não chega necessariamente ao período revolucionário. Ambas as abordagens seguem existindo. Ou seja, não ocorre uma substituição total ou parcial do paradigma anterior, algo importante dentro do esquema de Kuhn (2007).

Um segundo ponto de semelhança é quanto às generalizações simbólicas citadas por Kuhn (2007). No caso dos paradigmas de programação, isto é bem evidente. Por exemplo no paradigma imperativo, temos as variáveis como uma metáfora para acesso à memória principal

[59] Na literatura, também existe citações a algo chamado de *paradigm shift*.

do computador, algo que no orientado a objeto é papel dos atributos de um objeto. Um paradigma fornece uma série de conceitos comuns para orientar os trabalhos de programação.

Por fim, a ideia de que paradigmas funcionam como exemplares também está presente, algo reforçado pelo grande número de trabalhos que versam sobre paradigmas de programação com o objetivo de lidar com o ensino de programação. Ou seja, o carácter pedagógico do paradigma enfatizado por Kuhn (2007) é partilhado nos estudos sobre programação.

Quanto às diferenças, é fácil começar por aquilo que o Quadro 1 explicita: o reconhecimento da existência simultânea de diversos paradigmas, algo incomum dentro de uma comunidade de pesquisa, pelo menos da forma como Kuhn (2007) encara. Acreditamos que esta diferença se dá pelo fato da Ciência da Computação não ser uma ciência da mesma natureza que a física ou química, campos dos quais Kuhn (2007) retira a maior parte de seus exemplos. A Ciência da Computação, como percebe Gabriel (2012), estuda uma "realidade criada por nós" que, de certa maneira, é o entendimento de Simon (1996) e também de Margolin (2014). Ambos defendem que as formas de trabalho das ciências do artificial (engenharias, design, computação etc.) têm métodos e abordagens diferentes das ciências da natureza e da ciências humanas tradicionais.

Além da existência simultânea, existem as linguagens que implementam mais de um paradigma de programação, ou pelo menos permitem trabalhar com mais de um. No esquema de Kuhn (2007), os instrumentos gerados por um paradigma são parte dele e raramente são partilhados entre diferentes paradigmas.

As permanências entre paradigmas também funcionam de forma um pouco diferente. Kuhn (2007, p.230), ao comentar sobre as generalizações simbólicas, coloca que "a aceitação da lei de Ohm exigiu, entre outras coisas, uma redefinição dos termos corrente e resistência. Se estes dois termos continuassem a ter o mesmo sentido que antes, a lei de Ohm não poderia estar certa". Em outras palavras, novas generalizações simbólicas, às vezes, exigem redefinições de termos base de um paradigma. Nas linguagens de programação, algo parecido ocorre com certos termos que ganham novos nomes, mas permanecem similares em sua forma. Por exemplo, o que é chamado de função no paradigma imperativo torna-se método no orientado a objetos. A grande diferença está em que um método obrigatoriamente é parte de uma classe ou objeto enquanto a função não precisa. Por exemplo, no bloco 3.1 temos o caso da função main() em um programa simples escrito em linguagem C:

Algoritmo 3.1 – exemplo de função na linguagem de programação C.

```c
#include <stdio.h>
int main(int argc, char* argv[]) {
    printf("Hello, World!");
    return 0;
}
```

Fonte: Autoria própria. Gustavo Kira, 2021. CC BY-NC-SA 4.0

No bloco 3.2, temos o mesmo programa escrito na linguagem Java. É importante notar que a função *int main* no código em C e o método *main* no código em Java servem ambos como ponto de início de um programa. No caso do último, o método main exige que exista uma classe para abrigá-lo, pois assim é definido o conceito de método.

Algoritmo 3.2 – exemplo de método na linguagem Java.

```java
public class App {
    public static void main(String[] args) {
        System.out.println("Hello, World!")
    }
}
```

Fonte: Autoria própria. Gustavo Kira, 2021. CC BY-NC-SA 4.0

De um ponto de vista sintático, ambas trabalham com uma estrutura muito similar, uma vez que a sintaxe do Java é derivada do C. Entretanto, C trabalha dentro de um paradigma imperativo (ou procedural, dependendo da definição) enquanto Java é uma linguagem orientada a objetos. Este tipo de permanência vai mais do que simplesmente ressignificar os termos, mas a existência concreta de um paradigma dentro do outro. Os exemplos dos blocos 3.3 e 3.4 ajudam a exemplificar melhor esta questão:

Algoritmo 3.3 – Exemplo de programação imperativa na linguagem C.

```c
#include <stdio.h>

int main(void) {
  int i, j;

   printf("digite um valor\n");
  scanf("%d", &i);

   printf("digite outro valor\n");
  scanf("%d", &j);

  int k;
  if(i>j)
    { k = i;}
  else
    { k = j;}

   printf("maior valor digitado:%d\n",k);

  return 0;
}
```

Fonte: Autoria própria. Gustavo Kira, 2021. CC BY-NC-SA 4.0

No bloco 3.4, temos um programa que recebe dois valores inteiros (linhas 6-7 e 9-10) e imprime na saída (linha 18) o valor do maior deles. Além das entradas, existe apenas uma estrutura condicional que implementa a lógica necessária (linhas 12-16).

Algoritmo 3.4 – exemplo de programação orientada a objetos na linguagem Java.

```
1  import java.util.Scanner;
2
3  class Main {
4    public static void main(String[] args) {
5      Scanner s = new Scanner(System.in);
6
7      System.out.println("digite um valor");
8      int primeiroValor = s.nextInt();
9
10     System.out.println("digite outro valor");
11     int segundoValor = s.nextInt();
12
13     Comparador c = new Comparador();
14     int maior = c.maiorValor(primeiroValor, segundoValor);
15
16     System.out.println("maior valor:"+maior);
17
18   }
19 }
20
21 class Comparador{
22   public int maiorValor(int i, int j){
23     int k;
24
25     if(i>j)
26       { k = i;}
27     else
28       { k = j;}
29
30     return k;
31   }
32 }
```

Fonte: Autoria própria. Gustavo Kira, 2021. CC BY-NC-SA 4.0

Já o código em Java, se feito da maneira esperada pelo paradigma orientado a objetos, exige a criação de uma classe para executar a lógica que decide o maior valor[60]. Entretanto, ao olhar o código que vai dentro do método "maiorValor" da classe "Comparador" (linhas 22-30) e o código do programa em C (linhas 12-20), fica claro que parte do paradigma imperativo é parte constituinte do orientado a objetos. Apenas para ilustrar a questão melhor, também é possível escrever um programa na linguagem Java que seja válido sintaticamente do ponto de vista da orientação a objetos (não existe nenhum símbolo fora de um local adequado), mas com "cara" de imperativo, como mostra o bloco 3.5:

Algoritmo 3.5 – Exemplo de programação orientada a objetos em linguagem Java.

```
1  import java.util.Scanner;
2
3  class Main {
```

[60] Tanto o programa escrito em C quanto em java não tem tratamento de erros. Ou seja, caso sejam digitados como entrada valores que não sejam reconhecidos como números pela linguagem, o programa terminará. No caso do C, poderia ser usado um "return 1" para indicar uma execução que não terminou em sucesso. Já em Java, o mesmo poderia ser atingido usando "System.exit(1)" ou um tratamento de erro usando "try catch".

```
4     public static void main(String[] args) {
5       Scanner s = new Scanner(System.in);
6
7       System.out.println("digite um valor");
8       int i = s.nextInt();
9
10      System.out.println("digite outro valor");
11      int j = s.nextInt();
12
13      int k;
14      if(i>j)
15        { k = i;}
16      else
17        { k = j;
18      }
19
20      System.out.println("maior valor:"+k);
21    }
22  }
```

Fonte: Autoria própria. Gustavo Kira, 2021. CC BY-NC-SA 4.0

Estes desalinhamentos entre o uso feito pelos estudos em linguagens de programação e o conceito original de Kuhn (2007) permitem olhar os paradigmas de programação mais como uma metáfora que organiza os estudos das linguagens de programação do que um efetivo uso do conceito original.

Floyd (1979), muitos anos antes, oferece uma visão diferente dos paradigmas e, com isso, permite continuar a reflexão sobre o termo. Ele começa citando o paradigma estruturado, o que na época, parece ser o paradigma dominante. Ele é constituído de duas partes: Primeiro uma fase de cima para baixo, em que "o problema é decomposto em sub-problemas bem pequenos e simples[61]" (FLOYD, 1979, p.455). Depois, trabalha-se de baixo para cima, "dos objetos e funções concretas da máquina por baixo para os objetos e funções mais abstratas usadas nos módulos produzidos[62]" pela primeira parte (FLOYD, 1979, p.455).

Chama a atenção que o paradigma de Floyd (1979) é uma forma de programar bem distante dos apresentados até então. Isto fica evidente quando ele fornece mais alguns exemplos de paradigmas: branch-and-bound e dividir e conquistar[63]. Hoje em dia, ambos são considerados abordagens para o projeto de algoritmos.

Ao comentar sobre exemplos que usa em sala, Floyd (1979) cita o paradigma para entrada interativa[64], que consiste em um programa que trabalha com uma sequência pronta, leitura, validação e escrita:

Algoritmo 3.6 – paradigma para entrada interativa de Floyd (1979).

[61] *The problem is decomposed into a very small number of simpler subproblems.*
[62] *From the concrete objects and functions of the underlying machine to the more abstract objects and functions used throughout the modules.*
[63] *Divide-and-conquer.*
[64] *Paradigm for interactive input.*

```
PROMPT, a variable V to be read, and a condition BAD which characterizes bad data;
PRINT_ON_TERMINAL(PROMPT);
READ_FROM_TERMINAL(V);
WHILE BAD(V) DO
      BEGIN
      PRINT_ON_TERMINAL("invalid data");
      READ_FROM_TERMINAL(V);
      END;
PRINT_ON_FILE(V);
```

Mesmo que aquilo que Floyd (1979) chame de paradigma não seja a mesma coisa que os textos apresentados até então, ele também tem preocupações com o ensino de programação: “Se o avanço da arte geral de programação requer a contínua invenção e elaboração de paradigmas, avanço na arte de programação de um programador ou programadora individual requer que ele ou ela expanda seu repertório de paradigmas[65]” (FLOYD, 1979, p.457). Esta passagem mostra que, para Floyd (1979), um paradigma de programação não é algo que se está imerso, não é uma série de modelos, não são valores partilhados, mas algo que se conhece ou tem contato: um exemplar.

Se um dos objetivos de trabalhar com paradigmas de programação é o suporte para o seu ensino e aprendizagem, é justo considerar tanto as abordagens classificativas de Van Roy (2009) e Watt (2004) quanto a de Floyd (1979) como válidas. Com isso, podemos perguntar quais as implicações do uso do paradigma como metáfora, uma vez que existe uma escolha daquilo que é mantido e o que se perde na transposição do campo original.

O conceito de paradigma de Kuhn (2007) é o responsável, em seu quadro teórico, pela percepção do desenvolvimento científico como a-histórico, mesmo que a teoria proposta por Kuhn (2007) não o seja. A questão da invisibilidade das revoluções científicas ilustra bem esta questão. Para Kuhn (2007, p.175-176), são as fontes autorizadas que “disfarçam sistematicamente . . . a existência e o significado das revoluções científicas”. Por fontes, ele entende os “manuais científicos, juntamente com os textos de divulgação e obras filosóficas” (KUHN, 2007, p.176).

Kuhn (2007, p.177) coloca que, principalmente, os manuais e a literatura derivada destes, após uma revolução científica, precisam ser reelaborados, uma vez que são veículos pedagógicos. Estes manuais têm como propósito de existência perpetuar a ciência normal, “devem ser parcial ou totalmente reescritos toda vez que a linguagem, a estrutura dos problemas ou as normas da ciência normal se modifiquem” (KUHN, 2007, p.177).

Assim, estes manuais “começam truncando a compreensão do cientista a respeito da história de sua própria disciplina” (KUHN, 2007, p.177) e fazem referência somente “àquelas partes

[65] *If the advancement of general art of programming requires the continuing invention and elaboration of paradigms, advancement of the art of the individual programmer requires that he expand his reportory of paradigms.*

do trabalho de antigos cientistas que podem facilmente ser consideradas como contribuições ao enunciado e à solução dos problemas apresentados pelo paradigma dos manuais" (KUHN, 2007, p.177). Ou seja, um paradigma, ao se estabelecer, reorganiza não só as relações entre os conceitos do campos, mas também elege os fatos que levaram até ela.

Dado o que foi mostrado até aqui, talvez esta seja uma das qualidades de um paradigma que os paradigmas de programação almejam, mas deixam escapar parcialmente. Abordagens como *Object first* tentam truncar suas relações com outras formas consideradas de baixo nível[66], mas ainda assim, muitas vezes tem consigo elementos de outras formas de programar, como foi mostrado anteriormente.

Esta reorganização que trunca, em especial, elementos de outros tempos da computação cria uma ilusão de a-historicidade para estes paradigmas de programação. Umas vez que eles não trabalham com foco o desenvolvimento histórico destas permanências e descontinuidades, estes resíduos são reagrupados a fim de ter seu sentido dentro de um esquema sincrônico, deixando de lado tudo aquilo que é diacrônico neste processo[67].

Além disso, os paradigmas de programação parecem tentar lidar com formas relativamente estáveis de programar computadores, mas ainda dentro da perspectiva abstrata formalista: Tem seu foco em elementos que compõe uma forma de programar, sem levar em contas as pessoas envolvidas no processo.

Dado o referencial teórico trabalhado até aqui, deixar as pessoas de lado é problemático, principalmente ao levar em conta que os paradigmas de programação são muito discutidos junto com questões ligadas ao ensino e aprendizagem de computação. Para nós, o conceito de paradigma de programação é uma das formas que a visão de abstração formalista e dedutiva se apresenta de forma muito próxima com o ensino e aprendizagem de programação.

De um ponto de vista bakhtiniano, os paradigmas de programação são uma das formas de generalização teoricistas que ignoram os detalhes dos eventos únicos do mundo da vida. Entretanto, há implicações concretas, pois são a base de disciplinas universitária e são usadas como exigência em empregos.

No próximo capítulo, apresentamos uma análise de trocas de e-mail como evidências destes processos em três atividades feitas com turmas de Ensino Médio e de disciplinas de um curso superior ligados à área de Computação.

---

66 Linguagens de programação tem um nível com relação a proximidade da arquitetura da máquina.

67 Para uma visão deste lado diacrônico das linguagens de programação, ver o diagrama feito por Éric Lévénez: Lévénez (2021) ou Lévénez (2004).

4 DO CÁLCULO DE FORMULAS MATEMÁTICAS À ENUNCIAÇÃO DE TEXTOS

Aproximar a área da Computação da esfera das Letras não é algo novo para nenhuma das duas. Do lado das Letras, existe uma extensa discussão sobre práticas associadas ao uso do computador como um meio de leitura e escrita. Já do lado da Computação, talvez a faceta mais aparente é aquela ligada ao estudo de linguagens computacionais[1].

Além das relações interdisciplinares entre as duas áreas de saber, existe uma ligação estabelecida pelo próprio processo de criação de software. Para Vee (2013, p.48), o ato de programar tem uma relação complexa com o ato de escrita, uma vez que ele "é escrita, mas suas conexões com tecnologia e computadores também o diferencia da escrita de textos. Ao mesmo tempo, o sistema de escrita de código diferencia computadores de outras tecnologias infraestruturais, como automóveis".

Concordamos com Vee (2013) sobre a complexa relação com a escrita, mas entender a organização das etapas de produção e consumo de um *software* ajuda a diminuir, um pouco, esta ambivalência. Por exemplo, a diferência do código-fonte e do artefato produzido pelo processo de compilação desse, o qual será usado efetivamente por alguém.

O artefato final, que tem um de seus mais difundidos tipos um programa de computador, não necessariamente tem relação com a escrita, salvo seja esta sua finalidade. Quem tem uma relação, até um tanto direta com a escrita, é a produção do código-fonte. Por exemplo, a relação entre um programa de computador e seu código-fonte é similar a de uma planta de casa e a casa em si. A planta tem alguma relação com o ato de desenho da mesma maneira que o código-fonte tem com a escrita. A casa, por sua vez não tem uma relação com o ato de desenho.

Mesmo que as referências usadas por Vee (2013) sejam quase todas parte da Computação, o público-alvo da autora ainda parece estar mais direcionado para a esfera das Letras. Isso fica claro em alguns trechos do último parágrafo: "entender a programação de computadores como letramento computacional nos leva em direção a uma pedagogia da escrita mais inclusiva e compreensiva (VEE, 2013, p.60)[2]"; advoga para que programação "não seja deixada exclusiva-

[1] Se olharmos para o desenvolvimento histórico da área da Computação, a relação entre ambas é bem mais complexa, como por exemplo o uso de modelos de comunicação para o desenvolvimento de interfaces entre pessoas e computadores. Para uma visão muito abrangente do tema, ver Merkle (2002).

[2] *Understanding computer programming as cmptutational literacy leads the way foward towards a more comprehensive and inclusive writing pedagogy.*

mente sob o domínio da Ciência da Computação[3]"; e que "é importante abrir nossos conceitos sobre escrita para incluir programação[4]".

Assim, partimos de um pressuposto de que uma grande parcela do ensino de fundamentos de programação estabelece uma relacionamento muito forte com as disciplinas tidas como exatas, tais como cálculo e matemática. Isto se dá tanto por como o campo da computação foi estabelecido historicamente quanto pelo próprio processo de ensino e sua reprodução.

Tratamos a questão histórica brevemente no final do capítulo 2. Já sobre a questão do ensino e aprendizagem, Bazzo (2011, p.13) afirma que "a educação tecnológica brasileira pode ser classificada como tendo fortes raízes positivistas" e o processo de seu ensino é "uma indisfarçada afirmação da realidade do objeto por parte do professor e uma apassivada memorização de informações técnicas, de preferência matematizadas, por parte dos alunos". Este também parece ser o caso dos cursos ligados ao campo da Ciência da Computação e/ou Informática.

Ou seja, o ensino de fundamentos de programação, quando sem nenhuma orientação pedagógica, epstemológica e/ou filosófica é feito em uma espécie de auto-piloto que, por falta de suporte, acaba por emular uma forma específica de ensino, muitas vezes parecida com a da matemática de Ensino Superior.

Com isso, a escrita de programas acaba por ser vista como uma empreitada dedutiva, assim como é o ensino de cálculo: dada a regra geral e alguns exemplos, espera-se que as e os estudantes consigam resolver qualquer expressão númerica que seja uma manifestação da regra geral. No caso da computação, ensina-se a linguagem e suas estruturas básicas e espera-se que as e os estudantes consigam fazer qualquer tipo de programa de computador.

Vee (2013, p.43) analisa o empréstimo do termo letramento feito pela Computação e coloca que desde os anos 60, entusiastas da computação usaram o conceito de letramento para destacar a importância de escrever para e com computadores. Ela também chama a atenção para a forma não muito sofisticada de como é feita a conexão entre letramento e programação: "letramento limitado à leitura e escrita de texto; letramento divorciado do contexto social e histórico; letramento como uma forma absoluta de progresso[5]" (VEE, 2013, p.43).

Para a computação, este entendimento de como se dá o processo de aprendizagem da língua escrita tem uma grande saliência a ser resolvida: Linguagens de programação não

[3] *. . . programming cannot be relegated to the exclusive domain of computer science.*
[4] *It is also important to open up our concepts of writing to include programming.*
[5] *Literacy as limited to reading and writing text; literacy divorced from social or historical context; literacy as an unmitigated form of progress.*

necessariamente têm fala. A análise sobre o ensino da gramática que Faraco (2006b) faz fornece algumas respostas, em especial, ao comentar sobre o aprendizado de latim no período medieval.

Segundo Faraco (2006b, p.21), no período medieval, o estudo do latim, a língua dos letrados, adquire um carácter cada vez mais artificial após a desintegração de Roma. Neste período, a língua viva do cotidiano sofreu grandes transformações pelo contato com várias línguas germânicas e dialetos latinos, criando um espaço entre a língua usada na escrita nos monastérios e no dia a dia.

Desta maneira, aprender o latim erudito era aprender uma segunda língua sem nenhum falante nativo, mudando o status da gramática na aprendizagem:

> Isso exigia um elaborado trabalho intelectual que tinha de começar por uma descrição da língua. Com isso, a gramática deixou de ser suplemento para falantes interessados em aperfeiçoar o domínio de sua língua materna e passou a ser ponto de partida para se chegar a conhecer a língua que lhes era, de fato, estrangeira. (FARACO, 2006b, p.21)

O problema para Faraco (2006b, p.21) é que este processo resultou em um modelo pedagógico que partia sempre do gramatical e tem dois vícios que perduram até hoje: o normativismo, que seria "a atitude diante da norma padrão que não consegue apreendê-la como apenas uma das variedades da língua, com usos sociais determinados" (FARACO, 2006b, p.21) e a gramatiquice, que seria "estudo da gramática como um fim em si mesmo" (FARACO, 2006b, p.21).

O ensino de uma linguagem de programação, ou do cálculo diferencial e integral, parecem sofrer de ambos os vícios. Por exemplo, Faraco (2006b, p.25) comenta que estudar os termos gramaticais "normalmente listados pelo índice das gramáticas e postos numa sequência desprovida de qualquer articulação funcional [...] não tem a menor razão de ser". É possível dizer o mesmo quanto às variáveis, estruturas condicionais e laços de repetição, os quais muitas vezes são apresentados como itens em uma gramática, assim como os conceitos de limite, derivadas e integrais.

Faraco (2006b, p.26) ainda coloca que "reduzir, contudo, o estudo da concordância a uma lista de regras, cobrando sua aplicação em exercícios insossos e descontextualizados, é atividade inócua". No caso da computação, não é difícil encontrar manuais que depois do famoso "*hello word*", partem para exercícios com os operadores de soma(+), subtração(-), divisão(/) e multiplicação(*) sobre tipos numéricos em exercícios que lembram muito os de matemática conhecidos como "arme e efetue".

O normativismo e a gramatiquice, para Faraco (2006b, p.25), seriam "partes intrínseca de todo um conjunto de conceitos, atitudes e valores fundamentalmente autoritários, muito

adequados ao funcionamento de uma sociedade profundamente marcada pela divisão e exclusão social. É sobre este tipo de prática pedagógica descontextualizada, formalista e até autoritária que apresentamos três relatos de experiência no resto deste capítulo.

Os três relatos têm como função mostrar como é possível articular o corpo teórico apresentado até agora. As atividades foram aplicadas nos anos de 2017 e 2018 e no primeiro semestre 2019 em turmas de disciplina de Programação Orientada a Objetos. Em 2017, foram turmas de Ensino Médio e em 2018 em diante, turmas de Ensino Superior.

O primeiro relato de experiência é de uma atividade chamada Super Trunfo, que usa um jogo de cartas com mesmo nome para trabalhar com o conceito de classe e de objeto. Durante este período, foram feitas diversas atividades envolvendo este jogo de cartas, mas aqui encontra-se descrita somente uma aplicada nas turmas de Ensino Superior.

A segunda atividade tem como base um tabuleiro de xadrez e tem como objetivo trabalhar os conceitos de polimorfismo e herança. Esta atividade também tem duas versões, uma aplicada no Ensino Médio e uma aplicada no Ensino Superior. Escolhemos descrever a atividade feita no primeiro contexto.

O último relato é uma atividade que usa filmes como base para a introdução dos conceitos iniciais de OO: classe, objeto, atributos e valores. Assim, como as anteriores, filmes e similares (séries de tv) foram a temática de diversas atividades ao longo destes anos, servindo como base para o ensino de banco de dados relacional, programação para dispositivos móveis e, o foco deste texto, orientação a objetos.

Nas três atividades, oferecemos como material de análise, os códigos entregues durante e ao final das atividades e/ou trocas de e-mail entre as e os participantes e o professor da disciplina. Com isso, entendemos que estes materiais são enunciados concretos e apontamos alguns pontos de articulação entre a teoria apresentada até aqui e o dia a dia das atividades.

É importante lembrar que estes relatos se encaixam nesta tese da forma como experimentos didáticos (HIELE-GELDOF; HIELE, 1984). Ou seja, nenhuma destas atividades foi feita levando em conta outros processos alheios à situação escolar; não foram planejadas a fim de serem mensuráveis e/ou fornecerem respostas a uma hipótese bem definida, fazendo com que sua análise seja totalmente *a posteori*.

Por fim, podemos enumerar os dois principais pontos de partida destas atividades: 1) Entender abstração como um processo de mediação entre duas concretudes da forma como Meksenas (1992) propõe. Isto permite, principalmente, o uso dos conceitos associados a teoria

de Vigotski, em especial, o possível papel do professor ou da professora nesse processo. E, 2) Encarar a escrita de código-fonte de um programa sob regras similares da escrita de outros tipos de texto, dentro de uma perspectiva social e histórica, com o apoio das considerações sobre linguagens do Círculo de Bakhtin.

4.1 O ENUNCIADO CONCRETO NO CÍRCULO DE BAKHTIN

A noção de enunciado[6] para Círculo de Bakhtin tem um papel central no entendimento da linguagem como atividade histórica, cultural e social (BRAIT; MELLO, 2005). Em diversos momentos distintos, o enunciado é construído em oposição ao que seria uma frase/texto, sendo entendido como algo que vai além desses (SOBRAL, 2009, p.92). Uma frase ou texto devem ser entendidos aqui como elementos parte de uma prática linguística descontextualizada e que entende a linguagem como um sistema autônomo da história, da cultura e da sociedade. Em linhas gerais, o enunciado no entendimento do Círculo "implica muito mais do que aquilo que está incluído dentro dos fatores estritamente linguísticos, o que, vale dizer, solicita um olhar para os outros elementos que o constituem" (BRAIT; MELLO, 2005, p.67).

Por exemplo, Volochinov (2019 [1926]), preocupado com o entendimento do sentido integral de uma conversa, comenta sobre o contexto verbal e extra verbal da palavra. O contexto verbal se refere a palavra somente como fenômeno linguístico enquanto a perpectiva defendida por ele, a sociológica, leva em conta também o contexto extra verbal. Este, seria formado por três aspectos: 1) horizonte espacial comum dos falantes; 2) o conhecimento e a compreensão comum aos dois e 3) a avaliação comum da situação (VOLOCHINOV, 2019 [1926], p.118).

O exemplo usado por Volochinov (2019 [1926], p.118) tem duas pessoas em um quarto e, de repente, uma diz: "Puxa", sem uma resposta da segunda. O horizonte espacial comum seria o ambiente em que a interação ocorre, no caso o quarto, janela etc. Sobre o conhecimento e a compreensão comum, Volochinov (2019 [1926], p.118) complementa o exemplo ao colocar que as duas pessoas viram um floco de neve pela janela, estão cansadas do inverno que demora em acabar e que existe um descontentamento quanto à distância que a primavera parece ter, uma vez que ainda está nevando. Ou seja, o significado compartilhado por ambas as pessoas quanto o floco de neve que cai.

[6] Ou enunciação ou enunciado concreto. Neste texto estes termos serão usado para ser referir ao mesmo conceito dentro da filosofia do Círculo de Bakhtin. Para mais detalhes, ver Brait e Mello (2005) e a nota de rodapé da página 11 em Bakhtin (2016 [19XX]).

Para Volochinov (2019 [1926], p.119), a avaliação faz a ligação entre a o contexto real e o enunciado. A avaliação seria uma espécide de julgamento de valor, qualquer seja sua guia (cognitiva, política etc) e é uma das partes subentendidas do enunciado. É importante destacar o caráter partilhado destas avaliações, uma vez que elas não seriam “emoções individuais, mas os atos socialmente lógicos e necessários” (VOLOCHINOV, 2019 [1926], p.121). Ou seja, a avaliação entendida é conhecida somente por “aqueles que pertencem ao mesmo horizonte social” (VOLOCHINOV, 2019 [1926], p.121).

No caso deste trabalho, as trocas de e-mail que serão apresentadas são enunciados concretos e como tal têm um contexto verbal e extraverbal. Também há locutores e interlocutores que, dentro do processo de diálogo, trocam de posição quanto à autoria e possuem um projeto e uma execução enunciativa. Também julgamos importante identificar que existe um terceiro autor neste processo, um autor-criador. Este é diferente do autor-pessoa, funcionando como uma figura discursiva que surge desta monografia e que dá um acabamento e uma entonação para o processo analisado.

Por exemplo, a Tabela 8 mostra uma troca simples de e-mails. Neste caso, uma dupla de estudantes estava com um problema em um exercício da disciplina de Estrutura de Dados de um curso de Tecnologia em Análise e Desenvolvimento de Sistemas (este contexto é detalhado em uma seção mais a frente). Ambos os estudantes eram do curso de Ciência da Computação, matriculados na disciplina. Neste dia, a aula era justamente para tirar dúvidas e ajudar a turma a implementar um código que consiga percorrer os itens de uma lista encadeada.

Tabela 8 – Textos da troca de e-mails com um estudante na formação de grupos para um trabalho em equipe.

Remetente	**texto**
Professor 25/04/2018 23:27	Indo embora, lembrei que o temp->proximo->proximo pode ficar no while inclusive, uma vez que nunca vai existir um caso em que o temp->proximo é nulo e o temp->proximo->proximo, não.
Estudante 25/04/2018 23:47	Obrigado professor :D

Fonte: Autoria própria

O exercício em questão era percorrer uma lista encadeada (uma tarefa típica associada ao conteúdo de Estrutura de Dados) e a linguagem utilizada na disciplina era a linguagem C (mais especificamente notação de ponteiros). Com esta informação, é possível entender que

“temp->proximo->proximo” e “temp->proximo” são códigos em linguagem C. Menos evidente é o termo “while” que também faz parte das palavras reservadas do C.

Ainda sobre a questão do exercício, “ficar no while” remete a possibilidade do programa entrar no que é chamado de *loop* infinito, condição no qual um algoritmo nunca termina sua execução. Terminar, ou seja, ter uma quantidade de etapas finitas é uma condição necessária para que um algoritmo seja válido computacionalmente[7].

O dia em que foram trocadas estas mensagens, era um dos dias da semana (quarta-feira) que esta disciplina acontecia relativamente tarde (começava às 20:50 e terminava às 22:30). Em aulas expositivas, isto não é muito relevante, estretanto em momentos que exige uma abordagem mais ativa, como era o caso desta aula, às vezes existiam impactos na habilidade de se “achar *bugs*” em códigos alheios.

Assim, dado o cansaço acumulado do dia, nem sempre questões que em outros momentos são resolvidas com facilidade, eram resolvidas em sala de aula. Mais de uma vez a resposta para um problema enfrentado por estudantes ficava claro somente foram dos momentos em sala de aula, após um certo período de relaxamento. No caso deste e-mail, a resposta veio após um banho e uma refeição. Por isso, a primeira mensagem foi enviada cerca de uma hora depois do término da aula.

Quanto às intenções por de trás de um *feedback* rápido (uma hora depois), sendo que seria relativamente aceitável responder na aula seguinte (somente na próxima semana), destacamos a importância do componente institucional. O primeiro semestre de 2018 era também o primeiro semestre na instituição como professor temporário e a mesma matéria de Estrutura de Dados era dada para o curso de Ciência da Computação por uma professor com muito mais tempo de casa. Desta maneira, não conseguir achar as causas de um problema na implementação do código referente a um exercício típico da disciplina pode parecer uma falta de domínio do conteúdo, independente do motivo e por comparação.

Tanto o porquê do e-mail no mesmo dia quanto o próprio ato de enviar o email podem ser entendidos como parte daquilo que, para Bakhtin (2016 [1924], p.73), transforma um texto em enunciado: “a sua ideia (intenção) e a realização dessa intenção” (BAKHTIN, 2016 [1924], p.73). Quanto as intenções do segundo enunciado, podemos apensa presumir e especular sobre:

[7] Neste caso trabalhamos com a definição mais informal de algoritmo: Uma sequência de passos bem definidos com o objetivo de resolver um problema qualquer. Esta definição parece mais útil para quem está a aprender a lidar com linguagens de programação do que uma definição mais robusta e precisa, logo que exige um maior domínio de conceitos computacionais para ser colocada em uso, como as que tem por base máquinas de Turing.

a resposta 20 minutos após pode ser por cordialidade, por consideração ao tempo de resposta, ou até mesmo compulsória, já que a mensagem foi enviada por quem irá o avaliar.

A existência da intenção e a sua realização são importantes para configurar a concretude (em oposição ao carácter abstrato da linguística a qual o Círculo de Bakhtin se opõe) de um enunciado dentro do posicionamento do Círculo, pois implicam na existência de um autor no enunciado. Sobral (2009, p.91) identifica que nesta mesma obra, mais à frente, Bakhtin (2016 [1924]) diz que para um texto tornar-se um enunciado também precisa ter um autor. No caso da troca de mensagens da Tabela 8, existe uma alterância na enunciação, mudando a posição entre locutor e interlocutor em cada uma das mensagens, trocando a posição de autoria.

Sobral (2009, p.92) comenta sobre dois críterios estruturais que diferem um enunciado em comparação com uma frase para o Círculo: O enunciado implica em alternância, ou seja, tem sempre dentro de suas estrutura tanto o locutor quanto o interlocutor, mesmo que nem sempre de forma material e direta. E segundo, o enunciado é um todo, ou seja, ele tem um certo acabamento que exige uma ação responsiva de outro sujeito.

A forma como um e-mail funciona tem em si mecânicas que permitem responder e dar um certo acabamento ao enviá-lo. Entretanto, é preciso deixar claro que o para o Círculo, as duas questões colocadas por Sobral (2009, p.92) têm importância estrutural e implicam necessariamente a presença do outro no processo enunciativo. Em outras palavras, a alteridade é parte fundamental do projeto de linguagem do Círculo de Bakhtin e existe mesmo quando o a alternância não ocorre de fato, como seria o caso de e-mails não respondidos.

Levando em contas o que foi exposto até então sobre o que é o enunciado de um ponto de vista bakhtiniano, Sobral (2009, p.98) oferece uma definição compacta do conceito:

> uma unidade discursiva mediante a qual o locutor busca realizar um dado projeto enunciativo, de acordo com a interação em que está envolvido (e que o leva a alterar esse projeto enunciativo ao longo de sua execução), tendo por material as formas da língua e imprimindo ao que é dito um tom avaliativo que leva em conta a resposta ativa presumida do interlocutor a quem o locutor se dirige. (SOBRAL, 2009, p.98)

Neste caso temos algo parecido com a questão da autobiografia discutida por Bakhtin ([1969] 2011), pois nas trocas de e-mail, um dos interlocutores representados no texto é a mesma pessoa que escreveu esta monografia. Ambos são irredutíveis, já que a "coincidência pessoal 'na vida' da pessoa de quem se fala com a pessoa que fala não elimina a diferença ente esses elementos no interior do todo artístico" (BAKHTIN, [1969] 2011, p.139).

Mesmo que as trocas de e-mail sejam transcrições dos originais, o autor-criador seria o "centro artístico e axiológico que dá unidade ao objeto estético. Continua igualmente ocupando uma posição de exterioridade, ou seja, continua tendo um excedente de visão e conhecimento" (FARACO, 2006a, p.49-50). Isto implica em reconhecer, mais uma vez a dimensão subjetiva deste texto, mas desta vez dentro do quadro conceitual de Bakthin.

Sobral (2009, p.33) coloca que, em certa medida, tentar superar teorias e métodos que tomam a parte pelo todo ou que acreditam no todo como uma soma simples de suas partes é uma constante na obra do Círculo. Volochinov (2019 [1926], p.155) explicita esta visão de forma clara ao consolidar algumas de suas críticas ao método formal e ao psiquismo no estudo da literatura: "Em um balanço final, ambos [psiquismo e formalismo] os pontos de vista pecam pelo mesmo defeito: eles tentar encontrar o todo na parte, isto é, a estrutura da parte isolada de modo abstrato é apresentada como a estrutura do todo".

De forma similar, Bakhtin (2017 [1919/1921], p.43) constrói sua filosofia do ato responsável entendendo que existem dois mundos "absolutamente incomunicáveis e mutuamente impenetráveis: o mundo da cultura e o mundo da vida" (BAKHTIN, 2017 [1919/1921], p.43). De forma geral, ele argumenta sobre a incapacidade do mundo da cultura (teoricista) em dar conta do mundo da vida, de forma independente deste segundo. Ou como coloca Faraco (2017, p.152), Bakhtin vê problemas na pretensão universalista em sistemas do mundo da cultura, especialmente, no "pressuposto de que de um enunciado universal se deduz necessariamente a minha ação" (FARACO, 2017, p.152).

Como o foco desta parte são os e-mails, é importante lembrar que este meio é apenas uma das formas de comunicação usadas em alguns contextos específicos de ensino e aprendizagem. Entretanto, entendemos que apresentar uma parte de um processo maior é compatível com as visões do Círculo, pois não temos uma pretensão de generalização e temos plena consciência de que a parte não é o todo.

A título de organizar o processo de análise iremos usar os quatro níveis que Sobral (2009, p.41) identifica no processo interativo dialógico proposto nos escritos do Círculo de Bakhtin. Estes níveis são integrados e estão organizados de acordo com grau de amplitude e abrangência. Também levam em conta as instituições sociais e a conjuntura da situação.

O primeiro nível seria o dos aspectos físicos e materiais na interação entre os sujeitos que fazem parte do processo enunciativo. Sobral (2009, p.42) comenta que assistir a uma palestra

ao vivo e por um telão são situações diferentes, pois o palestrante pode modular sua apresentação de acordo com as reações da plateia.

O segundo nível trata do imaginário e das práticas sociais associadas ao contexto imediato. Neste nível, são acionados os papéis sociais no processo interativo. Um palestrante tem o "direito" durante sua palestra, mesmo que na plateia tenha colegas com conhecimento igual ou superior do seu assunto (SOBRAL, 2009, p.42).

Sobral (2009) não oferece exemplos para explicar o terceiro e quarto níveis. O terceiro nível engloba as questões ligadas aos aspectos históricos e culturais da interação junto com suas instituições formais ou informais. Já o quarto nível, Sobral (2009, p.44) coloca o "espírito de época" para tentar lidar com questões mais temporais tais como cortes geracionais.

Com isso, consideramos que a introdução desta tese junto com o segundo capítulo tratam, de forma parcial, o quarto e terceiro níveis descritos por Sobral (2009). Na próxima seção temos a descrição das duas instituições em que foram feitas as atividades que motivaram as trocas de e-mail que oferece um complemento ao terceiro nível.

Após descrever algumas das questões ligadas aos contextos, apresentamos de forma geral o corpus formado pelos e-mails trocados entre estudantes e o professor com o objetivo de apresentar as questões ligadas ao processo de ensino e aprendizagem, mas um pouco mais independentes das atividades.

Já nas descrições das atividades, estamos tentanto lidar com o segundo e primeiro níveis ao mostrar parte da materialidade que fez parte do processo de interação. Junto com elas estão algumas análises de e-mails trocados e nesses tentamos identificar, principalmente, os elementos usados por Volochinov (2019 [1926]) para articular o enunciado concreto. Além disso, também usamos as teorias de Vygotski para tecer comentários sobre o papel do professor neste processo.

4.2 OS CONTEXTOS INSTITUCIONAIS: ESCOLA E FACULDADE

4.2.1 A Escola

As atividades descritas a seguir foram feitas em dois contextos diferentes. Vamos chamar o primeiro de Escola, pois foram feitas com turmas do Ensino Médio e o segundo de Faculdade, pois foram feitas em disciplinas de Ensino Superior. As atividades do primeiro contexto foram feitas durante o anos de 2017 e as da segunda, no ano de 2018 e no primeiro semestre de 2019.

As atividades do primeiro contexto foram feitas em um campus de uma instituição da Rede de Ensino Federal localizada em uma cidade da região metropolitânea de Curitiba (Paraná). O campus começou a funcionar em 2015 com dois cursos técnicos integrados ao Ensino Médio: O Curso Técnico em Administração e o Curso Técnico em Informática.

No ano de 2017, o Curso Técnico em Informática ainda não tinha nenhuma turma de Ensino Médio formada. O curso tinha três turmas regulares, cada uma em um dos três primeiros anos dos quatro que o curso tem.

Os dois cursos eram organizados por bimestres (quatro no total) e o processo de avaliação era feito por conceitos (A, B, C ou D) atribuidos ao final de cada bimestre em cada unidade curricular. Caso o ou a estudante não obtivesse 75% de presença ou ao final do ano letivo tivesse atribuído o conceito D (insuficiente), ele ou ela deveria fazer no ano seguinte a disciplina novamente como dependência. Se o ou a estudante tivesse quatro ou mais conceitos D, teria que refazer todas as unidades curriculares do ano.

A turma do primeiro ano iniciou o ano com 46 estudantes, a do segundo com 39 e a do terceiro ano 31. Todas as disciplinas tinham encontros semanais, feitas em um sala com computadores (única do campus) alvo de disputas[8]. A preferência de uso era para as unidades do eixo técnico de informática, mas em várias ocasiões outros professores pediam para usar a sala, uma vez que suas atividades também precisavam de computadores.

Todas as unidades curriculares do eixo técnico de informática tinham um encontro por semana ao longo do ano. As atividades do primeiro ano foram feitas na unidade curricular de Arquitetura e Sistemas Operacionais, que ocorria nas segundas (11:20 às 13:00). As dos segundos anos foram feitas dentro da unidade Programação Orientada a Objetos 1, que tinha encontros semanais às sextas (7:40 até 9:20). As do terceiro ano foram feitas em três unidades: Programação Orientada a Objetos 2, Redes de Computadores e Internet e Banco de Dados 2. Respectivamente, elas tinham encontros semanais às segundas (9:20 até as 11:20, com 20 minutos de intervalo), quintas (9:20 até as 11:20) e sextas (11:20 até 13:00), como mostram os Quadros 2, 3 e 4.

Durante o ano de 2017, existiam três estudantes matriculados na dependência de Programação Orientada a Objetos. Dado o formato da grade (das 7:40 até 13:00, todos os ou as estudantes estão em alguma aula), as dependências deveriam ser feitas no contraturno (neste

[8] Professoras e professores de outras disciplinas tinham propostas que envolviam o uso de computadores para pequisa na internet ou uso de algum tipo de *software* específico. Como existia somente uma sala, os horários em que não existia nenhuma aula de alguma disciplina de informática eram escassos. Quando possível, os professores de informática cediam a sala, mesmo tendo preferência.

Quadro 2 – Horários das disciplinas do primeiro ano.

horários	Segunda	Terça	Quarta	Quinta	Sexta
7:40 - 8:30	Artes	História	Português	Espanhol	Geografia
8:30 - 9:20	Artes	História	Português	Espanhol	Geografia
9:20 - 10:10	Inglês	Int. Informática	Sociologia	Matemática	Algoritmos
10:10 - 10:30	Intervalo	Intervalo	Intervalo	Intervalo	Intervalo
10:30 - 11:20	Inglês	Int. Informática	Matemática	Filosofia	Algoritmos
11:20 - 12:10	Arquitetura e S.O.	Ed. Física	Espanhol	Lógica e LP	Inglês
12:10 - 13:00	Arquitetura e S.O.	Ed. Física	Espanhol	Lógica e LP	Inglês

Fonte: Arquivo pessoal

Quadro 3 – Horários das disciplinas do segundo ano.

horários	Segunda	Terça	Quarta	Quinta	Sexta
7:40 - 8:30	Portugues	Banco de Dados	Biologia	Filosofia	Programação OO
8:30 - 9:20	Portugues	Banco de Dados	Biologia	Inglês	Programação OO
9:20 - 10:10	Matemática	Inglês	Física	Inglês	Geografia
10:10 - 10:30	Intervalo	Intervalo	Intervalo	Intervalo	Intervalo
10:30 - 11:20	Sociologia	Inglês	Física	Matemática	Geografia
11:20 - 12:10	História	Quimica	Espanhol	Espanhol	Ed. Física
12:10 - 13:00	História	Quimica	Espanhol	Espanhol	Ed. Física

Fonte: Arquivo pessoal

Quadro 4 – Horários das disciplinas do terceiro ano.

horários	Segunda	Terça	Quarta	Quinta	Sexta
7:40 - 8:30	Sociologia	Português	História	Biologia	Geografia
8:30 - 9:20	Matemática	Português	História	Biologia	Geografia
9:20 - 10:10	Programação OO	Ed. Física	Matemática	Redes de Computadores e Internet	Física
10:10 - 10:30	Intervalo	Intervalo	Intervalo	Intervalo	Intervalo
10:30 - 11:20	Programação OO	Ed. Física	Espanhol	Redes de Computadores e Internet	Física
11:20 - 12:10	Quimica	Espanhol	Inglês	Filosofia	Banco de Dados
12:10 - 13:00	Quimica	Espanhol	Inglês	Inglês	Banco de Dados

Fonte: Arquivo pessoal

caso, no período da tarde). Os encontros com os estudantes ocorriam de forma flexível a cada duas semanas, nas sextas-feiras.

Ao todo, nas turmas de informática do ano de 2017 existiam mais estudantes do sexo masculino matriculados do que do sexo feminino, como mostra a Tabela 9

Tabela 9 – Quantidade de estudantes matriculados na escola divididos por sexo.

Ano	sexo	quantidade
Primeiro	Mulheres	13
Primeiro	Homens	33
Segundo	Mulheres	11
Segundo	Homens	26
Terceiro	Mulheres	12
Terceiro	Homens	19

Fonte: Autoria própria

A escola faz parte da Rede Federal de Ensino, cujo o ingresso é feito mediante concurso público para docentes efetivos ou processo seletivo para docentes temporários. Neste caso, foi feito um contrato com vigência de 1 ano com 40 horas sem dedicação exclusiva. Por lei, este contrato pode ser renovado por no máximo dois anos consectivos e, ao deixar o cargo de docente temporário, a pessoa não pode ingressar no serviço público federal no mesmo cargo pelos próximos dois anos. As vagas para docentes temporários têm regras específicas para serem abertas, sendo que a ocupada neste contexto foi criada para cobrir o afastamento para o doutorado de um professor concursado. Como o professsor afastado deveria voltar em um ano, o contrato temporário não foi renovado, pois a vaga deixaria de existir.

4.2.2 A Faculdade

As atividades do segundo contexto foram feitas em um dos centros de uma universidade estadual da região sul do Brasil. Este centro fica em uma cidade com a população estimada em quase 600.000 habitantes e lá funcionam somente cursos ligados às ciências exatas[9]: Tecnologia em Análise e Desenvolvimento de Sistemas, Ciência da Computação, Engenharia Civil,

[9] A cidade em questão tem uma grande quantidade de indústrias.

Engenharia de Produção, Engenharia Elétrica, Engenharia Mecânica, Licenciatura em Física, Licenciatura em Matemática e Licenciatura em Química.

Os cursos de Ciência da Computação e de Tecnologia em Análise e Desenvolvimento de Sistemas fazem parte do Departamento de Ciência da Computação. O curso de Ciência da Computação é um curso integral com duração de quatro anos e o de Análise de Sistemas tem duração de três anos e ocorre no período noturno.

As atividades foram feitas nas disciplinas de Estrutura de Dados e Programação Orientada a Objetos, ambas parte da terceira fase (de um total de seis) do curso de Tecnologia em Análise e Desenvolvimento de Sistemas. Entretanto, como estas disciplinas também são ofertadas no Curso de Ciência da Computação, as turmas sempre tinham alguns estudantes matriculados em Ciência da Computação. As disciplinas ofertadas pelo departamento de Ciência da Computação são semestrais e exigiam que o estudante tivesse pelo menos 75% de presença e nota maior que 7,0 ao final da disciplina.

Nos três semestres (primeiro e segundo de 2018 e primeiro de 2019), a disciplina de Programação Orientada a Objetos teve 30 estudantes matriculados. Os encontros ocorriam duas vezes por semana, nas quintas-feiras (19:00 até as 20:40) e sextas-feiras (20:50 até as 22:30).

No Ensino Superior, a quantidade de estudantes do sexo masculino também era maior que a de estudantes do sexo feminino. Se comparado com a escola, temos uma acentuação desta diferença, como é possível ver na Tabela 10. Mesmo que dois exemplos não permitam qualquer tipo de generalização, é interessante ver que ambos não entram em conflito direto com o panorama apresentado na introdução da tese.

Tabela 10 – Quantidade de estudantes matriculados na disciplina de Programação Orientada a Objetos divididos por sexo.

Período	sexo	quantidade
2018/1	Mulheres	8
2018/1	Homens	22
2018/2	Mulheres	6
2018/2	Homens	24
2019/1	Mulheres	10
2019/1	Homens	20

Fonte: Autoria própria

O ingresso como docente na universidade estadual também tem as duas modalidades mediante a concurso para professores efetivos e processo seletivo para docentes temporários. Entratanto, como a vigência dos cursos de graduação é semetral, o contrato tem periodicidade igual, mas podendo ser renovado por até 4 anos. Diferente da Escola, o regime de trabalho é por hora, ou seja, salário e dedicação dependem da quantidade de aulas por semestre. O contrato foi renovado duas vezes e encerrado a pedido do docente por motivos pessoais.

4.3 AS CARACTERÍSTICAS DO CORPUS DE ANÁLISE

O corpus apresentado neste capítulo é composto em sua maioria por troca de e-mails[10] e representa uma das formas de comunicação usadas nas atividades. O e-mail foi escolhido como uma das formas de comunicação usada nas atividades pelos seguintes motivos: 1) É uma forma de comunicação relativamente simples e espera-se que poucas pessoas tenham dificuldade em operá-lo.

2) E-mails também permitem "fugir" de plataformas como Moodle, que tem duas características não desejáveis: Primeiro, mesmo sabendo que as formas de controle social vão além das mecânicas de um *software*, estas plataformas são imbutidas de mecanismos de controle (horário limite de entregas, visão de acessos etc) que, se possível, queremos evitar. E segundo, em especial no caso da Escola, durante o ano de 2017, o campus não tinha nenhum suporte de TI para os professores[11]. Ou seja, a carga de trabalho de usar uma plataforma para suporte de ensino seria toda do docente.

3) O e-mail permite uma comunicação assíncrona, facilitando avaliações (no sentido de parecer ou opinião) de código que tomam mais tempo e que poderiam ser feitas fora do contexto de sala de aula. Esta necessidade de um "tempo" para o processo também é o motivo da preferência pelo e-mail no lugar de outros tipos de comunicação instantânea tais como *whatsapp* ou *telegram*. 4) Nesta linha ainda, o tipo de comunicação baseada em e-mails também permite uma troca de informações entre os estudantes e o professor fora da periodicidade dos encontros presenciais[12].

5) De um ponto de vista burocrático, a caixa de e-mail permitia controlar com facilidade os trabalhos entregues, corrigidos e com *feedback* já enviado.

[10] Toda e qualquer identificação dos e das estudantes foi removida a fim de preservar a privacidade destes. Isto não tem muito impacto nas análises apresentadas, pois o foco delas é apresentar uma dimensão deste processo.

[11] Inclusive, durante o recesso escolar do meio do ano, fiz a manutenção do laboratório de informática.

[12] Esta mecânica não é necessariamente benéfica, pois ela também permite comunicação em horários indesejáveis.

6) As caixas de e-mail também permitiam rastrear o histórico das interações, permitindo um certo grau de auditoria (revisão de nota, em que data os trabalhos foram entregues etc), caso necessário.

Quanto ao arquivo em que se encontram estas mensagens, ele é um serviço de e-mail online, ou seja, é um arquivo privado em formato digital. Como esta é uma caixa de e-mail de uso geral, em um primeiro momento, foram separados somente os e-mails trocados com estudantes. Em um segundo momento, foram excluídos e-mails sobre notas e/ou avisos resultando em um total de 3727 e-mails separados em 1.680 discussões por e-mail. A Tabela 11 mostra a quantidade de discussões agrupadas por quantidade de e-mails contidas nelas:

Tabela 11 – Discussões de e-mail agrupadas por quantidade de e-mails.

qty. de e-mails	qtde. de Discussões
01	611
02	732
03	119
04	93
05	37
06	32
07	21
08	9
09	7
10	4
11	4
12	4
13	1
14	2
16	1
17	1
19	1
39	1

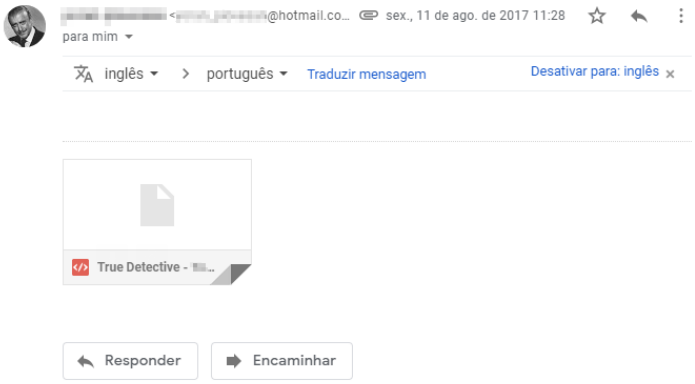
Fonte: Autoria própria

É interessante notar que uma grande quantidade de e-mails está agregado em discussões com um ou dois e-mails somente. As discussões com um e-mail, quando o e-mail veio de algum estudante, basicamente, são entregas de trabalhos, notificações, como a Figura 1.

O e-mail da Figura 1 foi enviado por um estudante do terceiro ano do Ensino Médio da Escola, apenas “entregando” um dos exercícios que estavam a ser feitos dentro de sala de aula. A forma de entrega era variada, como mostra a Figura 2, em que outra atividade foi entregue com o código no corpo do e-mail, enquanto na primeira, foi enviado um arquivo anexado com o código fonte.

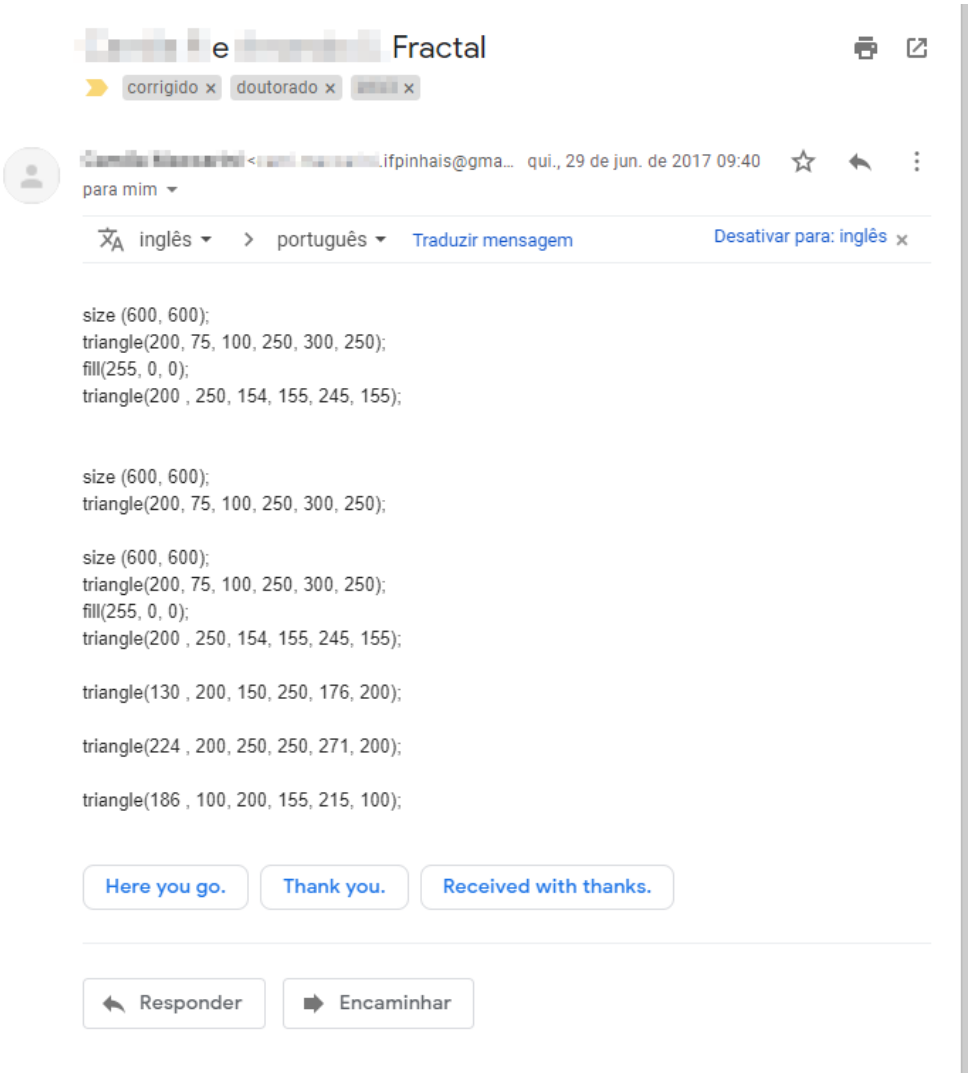
A estratégia de envio dependia muito do tipo de atividade. Entretanto, quando o ou a estudante tinha mais experiência com um ambiente profissional ligado à computação, estratégias

Figura 1 – Exemplo 01 de e-mail típico em discussões com somente um e-mail.



Fonte: Autoria própria

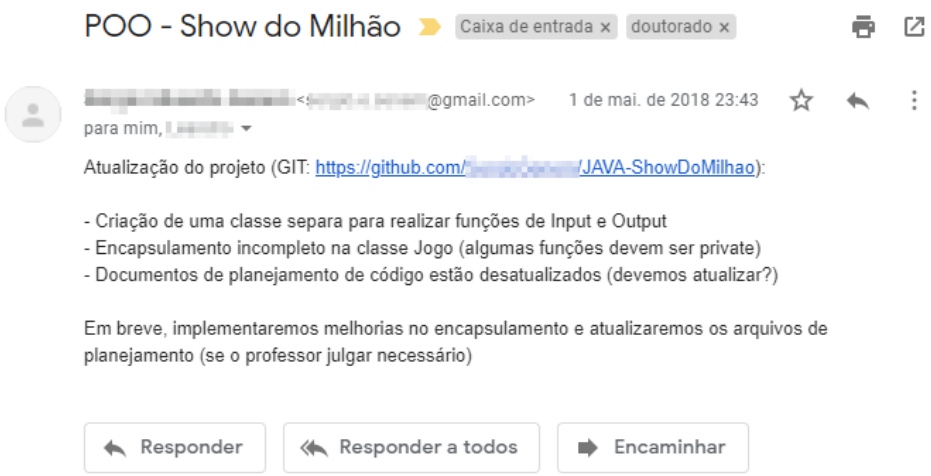
Figura 2 – Exemplo 02 de e-mail típico em discussões com somente um e-mail.



Fonte: Autoria própria

mais específicas eram usadas. Este é o caso da Figura 3, em que um estudante da Faculdade, que já trabalha como estagiário em uma empresa, envia apenas o *link* de um repositório do tipo git[13] em que estava disponível a versão do código mais recente.

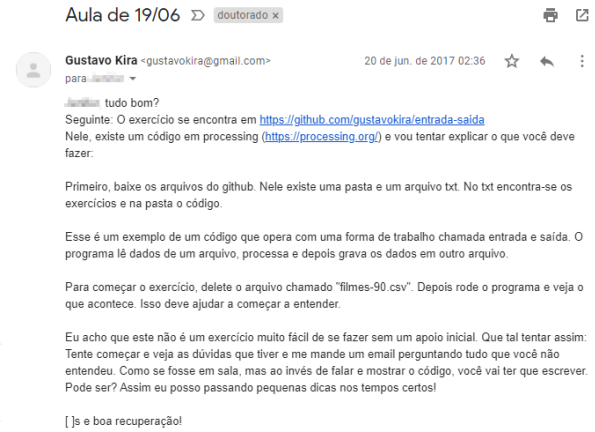
Figura 3 – Exemplo 03 de e-mail típico em discussões com somente um e-mail.



Fonte: Autoria própria

Junto com estes tipos de mensagem, as discussões com somente um e-mail também continham avisos do lado do docente com relação a notas, exemplos usados em sala para auxílio de atividades ou recuperação de conteúdo, como é o caso da Figura 4.

Figura 4 – Exemplo de e-mail do docente para um estudante que não pode comparecer em uma aula.



Fonte: Autoria própria

A Figura 4 mostra um pouco como a troca de e-mails é uma tentativa de extender a dinâmica do diálogo que acontece em sala de aula. Neste caso, um estudante estava “de atestado

[13] Git (https://git-scm.com/) é um sistema de controle de versão de código distribuido e livre. É implementado como serviço (grátis e pago) por empresas como Github, Gitlab e Atlassian (Bitbucket). Seu uso como controle de versionamento de código é bem difundido na industria.

médico" e não compareceu na aula em que foi feita uma atividade. Ele também mostra que o e-mail é parte de uma dinâmica maior e, neste caso, não pode ser entendido fora de seu contexto.

De forma geral, o tempo médio entre o primeiro e segundo e-mails das discussões com mais de um e-mail foi de 5,7 dias, com desvio padrão de mais ou menos 7,4 dias. Isto mostra que boa parte dos e-mails eram respondidos dentro do tempo de uma semana, algo que fica perto da periodicidade dos encontros de cada disciplina. Quanto à frequência das trocas, a Tabela 12 mostra a média do tempo entre o envio de dois e-mails consecutivos.

Tabela 12 – Discussões agrupadas por quantidade de e-mails e média do tempo de resposta entre dois e-mails seguidos.

qtd. de e-mails	média do tempo de resposta entre dois e-mails seguidos em dias
2	7,1088
3	3,7649
4	1,9297
5	2,1078
6	1,8094
7	1,1809
8	1,4199
9	0,2033
10	1,7188
11	0,2918
12	0,6021
13	3,6246
14	0,9605
16	0,0029
17	3,1306
19	5,2651
39	0,3971

Fonte: Autoria própria

A Tabela 12 mostra que existe uma tendência de que quanto mais intensa era a troca, menor era o tempo entre dois e-mails consecutivos. Em certa medida, o e-mail permite recuperar alguns elementos que as convesas em sala de aula têm, pois dado o tempo limitado e a quantidade de estudantes, nem sempre era possível conversar e tirar a dúvidas específicas de todos.

Algumas das interações por e-mail ocorriam após a entrega de uma atividade, como o exemplo da Tabela 13[14] que ocorreu após uma entrega de uma atividade sobre o conceito de pilha da disciplina de Estrutura de Dados da Faculdade. Nesta atividade deveria ser implementado um programa usando a estrutura de dados pilha (fornecida pronta já) usando a linguagem C e usá-la para verificar se uma palavra qualquer é um palíndromo.

[14] As trocas de mensagens serão reproduzidas em tabelas para facilitar a leitura.

Tabela 13 – Textos da troca de e-mails com um grupo em uma atividade da disciplina de Estrutura de dados.
.

Remetente	**texto**
Grupo 04/04/2018	Boa noite, professor! Segue em anexo os fontes desenvolvidos para a resolução do problema do Palíndromo. O trabalho foi desenvolvido por mim e pelo -. Att, - (3 anexos)
Professor 09/04/2018	Olá: 1) scanf tem problemas com espaços mesmo. 2) Aquilo que falei: não é preciso tirar tudo e contar para responder se é palíndromo ou não, mas ainda assim funciona. Se a gente quisesse saber exatamente o que tem de diferente, daí a abordagem de vcs ta mais no caminho certo. 3) Uma coisinha pequena e chata, mas é bom comentar: façam tudo na mesma língua! a função palindrome é a única que destoa. 4) Tem um string maiúsculo nos cabeçalhos.
Grupo 09/04/2018	Bom dia, professor: Acho que agora está certo. Já comparei direto ao tirar um letra, se ela é diferente da letra na posição correspondente. Assim não preciso armazenar em uma string e nem contar quantos iguais tem.

Professor
09/04/2018
	Agora que eu notei uma coisa:
	if(pilha_array_vazia(p)){
	printf(”A pilha esta vazia!\n”);
	return -1;
	}
	Esse p que vc passa não é uma pilha! É o ponteiro do array de char. Vai funcionar legal várias vezes, mas em certos casos vai ter um comportamento não esperado.
	Outra coisa, o flush pra funcionar melhor tem que ser fflush(stdout);
	Tomei a liberdade de fazer umas alterações no seu programa. Nada na lógica só na organização de arquivos. Veja se entende o que muda! Até
	(1 anexo)

Professor 09/04/2018	*Mudei só para mostrar uma coisa para vc! O seu programa, tirando o que comentei agora, está ok!

Fonte: Autoria própria

De forma resumida, o ambiente compartilhado pelos participantes do diálogo é tanto a sala de aula quando o espaço virtual dos e-mails, mesmo que a Tabela 13 faça somente referência ao ambinete virtual. O conhecimento comum da situação, neste caso, é a atividade e seus materiais, o conceito de pilha em estrutura de dados e a noção de palíndromo.

A seguir, no trecho de código 4.1, temos a função principal da solução para o problema do palíndromo entregue no primeiro e-mail da Tabela 13:

Algoritmo 4.1 – Exemplo de trecho de uma solução de uma atividade sobre o conceito de pilha.

```
1
2  int palindrome(char *p, int k){
3      PilhaArrayChar* pilha = pilha_array_criar();
4      int i, cont=0;
5      char inv[PILHA_ARRAY_SIZE];//sera o array que vai guardar o a palavra invertida
6
```

```
7         for(i=0;i<k;i++){
8               pilha_array_colocar (pilha, p[i]); // coloca a palavra recebida na pilha
9         }
10  /*
11        criei uma variavel chamada inv
12        que vai armazenar o palavra invertida
13  */
14        for(i=0;i<k;i++){
15              inv[i] = pilha_array_retirar (pilha);
16        }
17
18        printf ("Ao contrario \n");
19
20        for(i=0;i<k;i++){
21              printf ("%c", inv[i]);
22        }
23
24  /*
25        Para saber se e palindrome e so conta quantas letras iguais os
26        dois vetores vao ter, se todas forem iguais entao e palindrome
27  */
28        for(i=0;i<k;i++){
29              if(p[i] == inv[i]){
30                    cont++;
31              }
32        }
33  /*
34        Se a quantidade de palavras iguais (no caso o cont)
35        for igual ao tamanho do vetor (no caso k) entao retorna 1
36  */
37          pilha_array_destruir (pilha); // destruir essa pilha para nao gastar memoria
38
39        if(cont == k){
40              return 1;
41        }else{
42              return 0;
43        }
44  }
```

Fonte: Arquivo pessoal. Todos os direitos reservados.

O algoritmo implementado dentro da função *palindrome* funciona da seguinte maneira: 1) Recebe uma palavra qualquer e a sua quantidade de caracteres. 2) Criar uma pilha. 3) Cria um *array* vazio com o mesmo tamanho da palavra. 4) “Empilha” cada caracter da palavra a partir da sua primeira letra. 5) Retira da pilha os caracteres e guarda-os no *array* criado no passo 3. 6) Conta quantos caracteres da palavra original são iguais com os caracteres na mesma posição no array criado no passo 3. 7) Se a quantidade obtida no passo 6 for igual ao tamanho da palavra original, ela é um palíndromo.

A solução proposta pelos estudantes cumpre o objetivo do exercício. No *feedback* foi levantada a possibilidade de uma implementação alternativa da solução do problema e mostrada as diferenças entre ela e a usada pelo grupo. Em resumo, a solução alternativa consiste em eliminar a etapa de contar a quantidade de letras e decidir se a palavra é um palíndromo ou não na comparação da palavra original com o conteúdo do *array*. Por exemplo, a *string* “abcda” ao ser colocada e retirada de uma pilha é automaticamente invertida, ou seja “sai” da pilha a *string* “adcba”. Se compararmos caracter a caracter por sua posição (primeira letra com primeira letra,

segunda com a segunda etc) e se em uma das comparações os caracteres na mesma posição não forem iguais, não é um palíndromo.

De um ponto de vista computacional, a solução alternativa "gasta" menos processamento por dois motivos: Primeiro, não faz a contagem e segundo, não precisa ver a palavra invertida inteira para decidir que não é um palíndromo. Estas não são questões associadas diretamente com o conceito de pilha, mas muito relevantes no uso de estrutura de dados, pois a escolha e forma de processamento de uma estrutura é parte relevante de seu uso.

Assim, podemos fazer algumas considerações quanto à intenção de mostrar tal solução. Do ponto de vista da teoria vigotskiana, apresentada anteriormente, podemos dizer que esta proposta é uma atividade autônoma com uma participação ativa de dicentes e que visa uma apropriação de produtos da Ciência da Computação, parte da cultura tratada por Vigotski. Identificamos como instrumentos os códigos-fonte, o computador, e-mails e os elementos comuns de uma sala de aula.

A troca de e-mails apresentada na Tabela 13 mostra uma parte da dimensão interpessoal do processo de mediação, no qual espera-se estar trabalhando dentro da zona de desenvolvimento iminente. Com isto, esperamos que, possivelmente, os integrantes do grupo venham a conseguir fazer o mesmo tipo de considerações e análises sem uma ajuda igual a demostrada na troca de e-mails.

Este exemplo também serve para discutir o entendimento do que seria o processo de abstração que apresentamos. O conceito de palíndromo, mesmo sendo um tanto artificial, tem relações com conceitos já apresentados para os e as estudantes no Ensino Médio. Inverter uma palavra não é um processo difícil dentro do ambiente universitário, mas inverter usando ferramentas e conceitos da computação obrigatoriamente é consideravelmente mais complexo.

Também temos um indício de que o processo de abstração precisa levar em conta a "rede" de conceitos associados à atividade. A ideia de uma pilha não é algo difícil de ser entendido, mas sua implementação depende de conceitos ligados à programação de computadores. Além disso, a troca de mensagens mostra que um foco mais amplo é proveitoso, como é o caso da implementação alternativa do reconhecimento ao palíndromo. Isto é relevante ao lembrar que a disciplina de Estrutura de Dados é uma das disciplinas do começo dos cursos ligados à Ciência da Computação, ou seja, existem diversos espaços de conteúdo parte de um curso superior ainda a serem explorados com pouco sombreamento se comparado aos disponíveis ao final do curso.

Por fim, vale a pena citar, apenas a título de contexto, que nestas atividades foram poucos os casos de indicações ou referências a outros conteúdos como vídeos e textos de terceiros. As exceções foram dois casos específicos. Primeiro, um estudante tinha interesse em processamento de imagens e perguntou sobre um problema que envolve rotação de matriz e álgebra vetorial. Partiu do próprio estudante o envio de um vídeo que mostravam os conceitos e foi possível apenas explicar sobre o que seria.

O segundo caso era de um estudante com deficiência. Dado que as atividades regulares propostas não tinham nenhum efeito, foram feitas algumas atividades extracurriculares que envolviam assistir a vídeos. Porém, neste caso em especial, o estudante tinha um grande interesse sobre vídeos de internet, o que facilitou o uso de tal mídia.

4.4 FILMES E PROGRAMAÇÃO DE COMPUTADORES

Sempre que assumo uma disciplina, além de executar o protocolo institucional (normalmente, apresentar a ementa, plano de ensino e planos de aula), também faço uma apresentação pessoal e peço para que as pessoas na disciplina se apresentem.

Nada de novo até aqui. Entretanto, além de explicar quem eu sou, minha formação e minhas experiências, eu também apresento um filme que eu gostei de ver recentemente. Também peço para cada participante fazer o mesmo. Este é um jeito que encontrei de ter algo em comum para conversar com as e os estudantes. Nas disciplinas de OO, serve como um contexto para iniciar as explicações do que são classes, objetos, atributos e valores. Em outras disciplinas, como banco de dados, é possível usar filmes ou seriados como conteúdo do próprio banco, o que significa imcorporá-los na modelagem do banco também.

O uso de filmes como matéria prima serve de suporte para uma concretude inicial nas atividades didáticas. O objetivo é aproveitar um conhecimento de domínio bem estruturado (todos os e as participantes de todas as atividades tinham plena consciência do que é um filme), o que significa que as pessoas não vão ter dúvidas sobre o conteúdo daquilo que deve virar um código-fonte. Assim, é possível focar nas questões ligadas ao processo de transformação deste domínio concreto para o domínio referente aos métodos e conceitos ligados à Computação.

No caso do ensino de programação orientada a objetos, também temos uma questão histórica, uma vez que o projeto de uma locadora de filmes foi, durante um certo tempo, um dos exemplos relativamente difundidos para o ensino de Orientação a Objetos, UML e Sistemas de

Informação. Se algum dia as locadoras fizeram parte da realidade das pessoas, durante o período em que esta tese foi formulada, isso não é mais verdade.

Primeiro, é feita a explicação introdutória sobre o que é uma classe e para que serve com relação à organização de um código-fonte. Em um segundo momento, com a ajuda do grupo, são identificados possíveis atributos da classe e, em seguida, discutimos se o resultado deste "protótipo" é adequado a qualquer contexto, dando exemplos e perguntando se a classe proposta permite fazer o que é preciso com ela.

Dois exemplos simples, que mostram como uma classe Filme pode ser diferente dependendo do seu contexto, mesmo que parecidos, são um cinema e uma plataforma de *streaming*. Esperamos que as e os participantes identifiquem que cada um dos contextos permite diferentes possibilidades de atributos. Por exemplo, no caso do cinema, sala e horário são seriam possibilidades, enquanto para um plataforma de *streaming*, não.

Neste ponto, também é possível comentar a possibilidade de outras classes como atributos, mas preferencialmente deixamos a questão para ser tratada mais à frente, por outras atividades planejadas para trabalhar com Composição e Agregação.

O próximo passo é explicar a diferença entre classe e objetos, conceitos muito próximos, mas de importante distinção. Ainda fazemos proveito dos filmes, agora tentando deixar clara a diferença entre atributos e valores.

Com este contexto, é pedido para as e os participantes criarem um "top 20" de filmes. Eles devem criar uma classe Filme e criar 20 objetos do tipo filme, com no mínimo 5 atributos. Como é uma atividade introdutória, escolhemos usar algo já trabalhado em sala para que exista uma referência de apoio ao processo de abstração que deve ser empregado aqui.

A escolha do número 20 também é proposital, para que fique clara uma questão básica quanto à diferença entre classes e objetos, em especial, no caso comum de um código-fonte em que é possível existir diversas instâncias (objetos) de uma classe, mas não é válida a existência de mais de uma classe com mesmo nome.

Os trechos de código a seguir são parte da disciplina de Programação Orientada a Objetos ofertada no Ensino Superior. De maneira geral, os códigos entregues mostraram poucos problemas quanto ao conceito de classe e o de objeto. Como a forma de trabalho proposta não vê sentido em apenas atribuir uma nota ao que foi entregue ou "descontar" um valor por aquilo que se considera desadequado com o que foi pedido, ao mesmo tempo que tenta levar em conta as

condições concretas iniciais das e dos participantes, apresentamos, a seguir, alguns trechos de código para discutir como se dá este processo.

No trecho de código 4.2 é possível ver algumas das características do que apresentamos sobre o processo de abstração contextualizado. Por exemplo, o "tornar mais concreto o concreto" de Althusser (2019) aparece ao contrastar a definição de um filme em um "top 20", com uma outras para contextos diferentes, tais como os exemplos citados da plataforma de *streaming* e do cinema.

Se analisarmos cada uma destas definições isoladas, é possível entender como a abstração pode ser vista como uma processo de "escolha" daquilo que é importante ou de subtração de características desnecessárias. Esta visão tem seu foco naquilo que sofre a abstração e deixa de lado quem está a fazer a abstração. Já nas propostas que apresentamos, como a pessoa que esta a fazer o processo de abstração deve ser levada em conta no processo, argumentamos que as definições de filmes em diferentes problemas tem seu nexo nela. Ou seja, aquilo que é um filme tem sua concretude enrriquecida, pois todas estas visões do que é um filme coexistem em quem trabalhou com no processo de abstração, não existindo uma substituição do conceito original de filme.

Algoritmo 4.2 – Trecho de código entregue para a atividade de filmes

```
1
2  class Filmes{
3    int posicao;
4    String nome;
5    String genero;
6    int ano_lancamento;
7    String  class_indicativa ;
8  }
9
10 class Main {
11 public static void main (String [] args) {
12   Filmes a = new Filmes ();
13   a.nome="Orgulho e Preconceito";
14   a.posicao=1;
15   a.genero="Drama/Romance";
16   a.ano_lancamento=2005;
17   a. class_indicativa ="Livre";
18
19   Filmes b = new Filmes ();
20   b.nome="As Branquelas";
21   b.posicao=2;
22   b.genero="Comédia";
23   b.ano_lancamento=2004;
24   b. class_indicativa ="+12";
25
26    [...]
27
28   Filmes s = new Filmes ();
29   s.nome="Treinando o Papai";
30   s.posicao=19;
31   s.genero="Comédia";
32   s.ano_lancamento=2007;
33   s. class_indicativa ="Livre";
34
```

```
35      Filmes t = new Filmes ();
36      t.nome="Jogos Vorazes";
37      t.posicao=20;
38      t.genero="Ação/Drama";
39      t.ano_lancamento=2012;
40      t. class_indicativa ="+12";
41
42      System.out. println ("Posição: "+a.posicao+" Nome: "+a.nome+" Genero: "+a.genero+" Ano de
            Lançamento: "+a.ano_lancamento+" Classifica çao Indicativa : "+a. class_indicativa );
43      System.out. println ("Posição: "+b.posicao+" Nome: "+b.nome+" Genero: "+b.genero+" Ano de
            Lançamento: "+b.ano_lancamento+" Classifica çao Indicativa : "+b. class_indicativa );
44
45       [...]
46
47      System.out. println ("Posição: "+s.posicao+" Nome: "+s.nome+" Genero: "+s.genero+" Ano de
            Lançamento: "+s.ano_lancamento+" Classifica çao Indicativa : "+s. class_indicativa );
48      System.out. println ("Posição: "+t.posicao+" Nome: "+t.nome+" Genero: "+t.genero+" Ano de
            Lançamento: "+t.ano_lancamento+" Classifica çao Indicativa : "+t. class_indicativa );
49  }
50  }
```

Fonte: Arquivo pessoal. Todos os direitos reservados.

O trecho 4.2 também permite discutir a questão da mediação entre duas concretudes. Neste caso, a primeira concretude seria o conhecimento prévio sobre filmes, cinema e afins. A concretude almejada, mais próxima de conceitos ligados à computação, pode ser vista na forma como é feita a descrição da classe Filmes[15]. Ao definir que o nome de um filme deve ser uma *String* estamos a fazer com que exista uma relação entre aquilo que já era conhecido como filme com um modelo de filme que depende de conceitos ligados à computação.

Temos também algumas questões mais específicas, como o uso de *snake case*[16] nos atributos ano_lancamento e class_indicativa e o atributo gênero sendo tratado como multivalorado. Sintaticamente, não existe nada de errado com o código entregue (ele compila). Mas, como nesta disciplina de OO foi usada a linguagem Java e o seu padrão é o uso de *camel case*, foi indicada para a pessoa que entregou o código esta questão.

O mesmo foi feito sobre o uso de atributos multivalorados no código, pois sua validade depende muito do contexto em que o código é usado. Por exemplo, faz parte da disciplina de Banco de Dados[17] discutir atributos multivalorados e, dependendo do tipo de aplicação, seu uso pode tornar inviável certos tipos de funcionalidade. Por outro lado, se estas funcionalidades não fazem parte do escopo, seu uso pode simplificar o processo de produção da aplicação.

[15] É mais comum que o nome de uma classe seja no sigular, algo que é trabalhado ao longo da disciplina, pois com exemplos as motivações ficam mais claras.

[16] *Snake case* é um padrão de escrita de programação que substitui os espaços na declaração de nomes de variáveis e funções/métodos pelo carácter sublinhado (*underscore*). Este é o mesmo caso do *lower_case_with_underscores* do Python (https://www.python.org/dev/peps/pep-0008/)

[17] Neste caso estamos nos referindo somente aos bancos de dados relacionais.

O próximo trecho (4.3), um outro pedaço de um código entregue para a mesma atividade, também ajuda a levantar questões similares, dando destaque ao uso de caixa baixa para a primeira letra do nome de uma classe.

Algoritmo 4.3 – Trecho de código entregue para a atividade de filmes

```
1
2  class filme {
3     String Nome;
4     int ano;
5     int posicao;
6     int nota;
7     String genero;
8  }
9
10 class Main {
11
12 public static void main (String [] args) {
13
14 filme a = new filme ();
15 a.posicao = 1;
16 a.Nome = "A Hospedeira";
17 a.genero = "Ficção";
18 a.nota = 10;
19 a.ano =2013;
20 System.out.println ("POSIÇÃO: "+a.posicao+"  NOME: "+a.Nome+" GENERO: "+a.genero+" NOTA:
       "+a.nota+" ANO: "+a.ano);
21
22 filme b = new filme ();
23 b.posicao = 2;
24 b.Nome = "Perdido em marte";
25 b.genero ="Ficção";
26 b.nota = 10;
27 b.ano =2016;
28 System.out.println ("POSIÇÃO: "+b.posicao+"  NOME: "+b.Nome+" GENERO: "+b.genero+" NOTA:
       "+b.nota+" ANO: "+b.ano);
29
30  [...]
31
32 filme u = new filme ();
33 u.posicao = 19;
34 u.Nome = "A nova cinderela";
35 u.genero = "Romance";
36 u.nota = 9;
37 u.ano =2015;
38 System.out.println ("POSIÇÃO: "+u.posicao+"  NOME: "+u.Nome+" GENERO: "+u.genero+" NOTA:
       "+u.nota+" ANO:"+u.ano);
39
40 filme v = new filme ();
41 v.posicao = 20;
42 v.Nome = "O incrivel Hulk";
43 v.genero = "Ficção";
44 v.nota = 9;
45 v.ano =2008;
46 System.out.println ("POSIÇÃO: "+v.posicao+"  NOME: "+v.Nome+" GENERO: "+v.genero+" NOTA:
       "+v.nota+" ANO:"+v.ano);
47
48 }
49
50 }
```

Fonte: Arquivo pessoal. Todos os direitos reservados.

Comparado com o exemplo anterior, é possível ver indícios que o conceito inicial de filme, ou aquilo que não deve ser “cortado” varia, uma vez que os atributos escolhidos para representar um filme em um “top 20” são diferentes. A maior diferença está no fato de que

o primeiro tem um atributo para a classificação indicativa (linha 7), enquanto o segundo um atributo para uma nota do filme (linha 6).

O trecho seguinte (4.4) mostra um código entregue com o uso de diversos elementos específicos característicos da linguagem Java.

Algoritmo 4.4 – Trecho de código entregue para a atividade de filmes

```
1  // Declaracao da classe
2
3  package br.com.poo.utils;
4
5  import java.util.Date;
6
7  public class Filme implements Comparable<Filme>{
8      String nome;
9      int duracao;
10     double avaliacao;
11     Date dataDeLancamento;
12     String genero;
13
14
15     public Filme(String nome, int duracao, double avaliacao, Date dataDeLancamento, String
           genero) {
16         this.nome = nome;
17         this.duracao = duracao;
18         this.avaliacao = avaliacao;
19         this.dataDeLancamento = dataDeLancamento;
20         this.genero = genero;
21     }
22
23     @Override
24     public int compareTo(Filme filme) {
25         if (this.avaliacao > filme.getAvaliacao()) {
26             return -1;
27         } else if (this.avaliacao < filme.getAvaliacao()) {
28             return 1;
29         }
30         return 0;
31     }
32
33     @Override
34     public String toString() {
35         return "["+this.nome+" - "+this.avaliacao+"]";
36     }
37
38     private double getAvaliacao() {
39         return this.avaliacao;
40     }
41
42 }
43
44
45 package br.com.poo.utils;
46
47 import java.util.ArrayList;
48 import java.util.Collections;
49 import java.util.List;
50
51 public class RankingDeFilmes {
52     List<Filme> filmes = new ArrayList<Filme>();
53
54     public RankingDeFilmes() {
55
56     }
57
58     public void adiciona(Filme filme) {
59         this.filmes.add(filme);
60     }
```

```
61
62      public void top(int quant) {
63          Collections.sort(filmes);
64          int size = filmes.size();
65          quant = size < quant ? size : quant;
66          for ( int i = 0 ; i < quant ; i++ ) {
67              String posicao = (i+1) < 10 ? "0"+(i+1) : String.valueOf((i+1));
68              System.out.println(posicao+" "+filmes.get(i));
69          }
70      }
71  }
72
73
74  package br.com.poo.main;
75
76  import java.sql.Date;
77
78  import br.com.poo.utils.Filme;
79  import br.com.poo.utils.RankingDeFilmes;
80
81  public class Main {
82      public static void main(String[] args) {
83
84          RankingDeFilmes ranking = new RankingDeFilmes();
85          ranking.adiciona(new Filme("Sempre ao Seu Lado",93, 4.0, new Date(2009, 11, 25),
                "Drama"));
86          ranking.adiciona(new Filme("Homens de Honra",128, 4.7, new Date(2001, 2, 23), "Drama"));
87          ranking.adiciona(new Filme("Django Livre",93, 2.9, new Date(2013, 0, 18), "Faroeste"));
88          ranking.adiciona(new Filme("Até o Último Homem",140, 3.5, new Date(2007, 0, 26),
                "Biografia"));
89          ranking.adiciona(new Filme("Viva - A Vida é uma Festa",105, 4.1, new Date(2018, 0, 4),
                "Animação"));
90          ranking.adiciona(new Filme("A Bela e a Fera",87, 3.8, new Date(2012, 1, 3),
                "Animação"));
91          ranking.adiciona(new Filme("Divertida Mente",95, 4.0, new Date(2015, 5, 18),
                "Animação"));
92          ranking.adiciona(new Filme("O Resgate do Soldado Ryan",163, 5.0, new Date(1998, 8, 11),
                "Guerra"));
93          ranking.adiciona(new Filme("Orgulho e Preconceito",127, 2.1, new Date(2006, 1, 10),
                "Romance"));
94          ranking.adiciona(new Filme("Gladiador",155, 4.8, new Date(100, 4, 19), "Épico"));
95          ranking.adiciona(new Filme("Clube da Luta",139, 5.0, new Date(1999, 9, 29),
                "Suspense"));
96          ranking.adiciona(new Filme("O Retorno de Jedi",133, 4.0, new Date(1983, 9, 6),
                "Aventura"));
97          ranking.adiciona(new Filme("Toy Story - Um Mundo de Aventuras",77, 3.6, new Date(1995,
                11, 22), "Animação"));
98          ranking.adiciona(new Filme("Pulp Fiction - Tempo de Violência",143, 2.0, new Date(1995,
                1, 18), "Suspense"));
99          ranking.adiciona(new Filme("Diário de uma Paixão",121, 4.6, new Date(2004, 7, 13),
                "Romance"));
100         ranking.adiciona(new Filme("A Viagem de Chihiro",125, 3.4, new Date(2003, 6, 18),
                "Animação"));
101         ranking.adiciona(new Filme("Harry Potter e o Prisioneiro de Azkaban",140, 4.0, new
                Date(2004, 5, 4), "Fantasia"));
102         ranking.adiciona(new Filme("Guerra nas Estrelas",121, 4.0, new Date(1978, 0, 30),
                "Aventura"));
103         ranking.adiciona(new Filme("À Procura da Felicidade",118, 4.9, new Date(2007, 1, 2),
                "Biografia"));
104         ranking.adiciona(new Filme("O Silêncio dos Inocentes",118, 4.7, new Date(1991, 4, 17),
                "Suspense"));
105
106         ranking.top(20);
107     }
108 }
```

Fonte: Arquivo pessoal. Todos os direitos reservados.

Existe uma grande diferença na forma de implementação deste código se comparado com os dois primeiros códigos. Neste, temos o uso de métodos construtores (linha 15 e linha 54),

implementação de Interface (linha 7), uso de anotações (@*Override*), criação de mais de uma classe, laço de repetição (linha 66) etc. Estes são elementos mais específicos da linguagem Java, mas também compartilhados com outras linguagens similares.

As seguir, na Tabela 14, as considerações sobre o código feitas e enviadas por e-mail para a pessoa que fez o código.

Tabela 14 – Textos da resposta de email em uma atividade de Filmes

Texto
Olá: Legal que vc usou pacotes. Só o nome utils não é muito comum para esse tipo de classe. Já que vc foi além do que pedi, não sei o que vc escolheu fazer desse jeito ou não quis fazer pois já é bastante coisa. Mesmo assim, vou comentar. Ignore se quiser! 1) O construtor Date que vc usou foi aposentado. Dá para usar a classe Calendar para fazer a mesma coisa. Ou java 8 tem um pacote que dá para usar também. 2) getAvaliacao() da classe Filme, neste caso, não é necessário. Os atributos não são privados, então podemos acessá-los diretamente. Mas, que essa forma que vc fez é o "mais correto"em java. 3) Curiosidade: porque usar ternário no top do Ranking e não no compareTo? =) Ali é um dos lugares que o ternário deixa o código "bonito". 4) Outra coisa interessante de pensar, não necessariamente melhor ou pior, é que o tratamento de colocar o zero na frente dos números da posição, poderia ser responsabilidade de um método da classe Filme, ou outra até. Pessoalmente, gosto da sua abordagem mais. 5) Outra coisa legal de pensar é onde instanciar o ArrayList. No ranking, você criou direto no espaço do atributo. Não sei se vc sabe, acho que sim, mas Java permite ter mais de um contrutor. Imagine que eu queria criar o arraylist de filmes fora e passar para a classe ranking. Do jeito que está a gente sempre vai instanciar um arraylist independente de usá-lo ou não. Curiosa sua lista. Silencio dos Inocentes e Bela e a Fera na mesma lista diz muito sobre a pessoa! =) Não preciso comentar, mas legal usar @overrides, implements e etc. Até!

Fonte: Autoria Própria

Este terceiro exemplo ajuda a mostrar o quanto pode variar o grau de conhecimento das pessoas que participam da disciplina. Tentar organizar a atividade pelo nível mais alto ou pelo mais baixo torna a atividade sem função para uma parte das pessoas envolvidas. Inclusive, procurar um critério “médio” a fim de dar uma isonomia também traz perdas ao processo de ensino e aprendizagem, uma vez que não seriam atividades adequadas para ninguém.

Os três trechos de código também mostram que mesmo em um exercício simples, existe espaço para se levar em conta a concretude inicial das pessoas envolvidas. Se no primeiro (4.2) e segundo (4.3) eram necessários clarificar alguns padrões sociais quanto à escrita de código em Java, no terceiro (4.4), são feitas considerações específicas sobre Java diferentes das feitas para os dois primeiros grupos.

Se nos dois primeiros casos os conceitos articulados ligados à computação parecem ser os mesmos ou muito próximos, o terceiro mostra algo diferente. Em outras palavras, a Zona de Desenvolvimento Iminente do terceiro caso não trabalha com os mesmos conceitos das zonas

dos dois primeiros. Todas estão orientadas para a Computação, mas em níveis e intensidades diferentes.

Inclusive é possivel ver indícios de que cada Zona de Desenvolvimento Iminente é única e dependente do contexto e das pessoas envolvidas. A Tabela 15 mostra uma troca de e-mails sobre a mesma atividade que dá alguns subsídios para tal afirmação.

Tabela 15 – Textos da troca de emails com um estudante sobre a atividade de filmes.

Remetente	**texto**
Estudante 15/08/2018	Olá Professor, preciso de sua ajuda, estou com duvida em relação ao exercício 02. para eu poder fazer um top 20 eu devo criar um array ordenando meus filmes pelos maiores votos do meu atributo “num_votos”? estou com duvida para montar esse exercício, se você puder me passar algumas orientações. 1. Top 20 filmes Criar uma classe filme com pelo menos 5 atributos e fazer um top 20 mostrar na tela Public Class Filme { String Titulo; Int duracao; int num_votos; String genero; String diretor; Att, -

Professor
15/08/2018
	Se quiser uma classe Ranking, seria bom.
	sintaxe de array em java é assim:
	Tipo[] nomeDaVar = new Tipo[tamanhoDoArray];
	ex:
	int[] numeros = new int[10]; //Array de inteiros com 10 espaços.
	Por hora, em vez de ordenar, coloque na ordem já.

Estudante 16/08/2018	Olá Professor, Boa Tarde. em anexo segue o exercício 02. Sou sincero em dizer que não consegui montar a ordenação na atividade 01, montei somente as classes. no exercício 02 montei os objetos em negrito mas fiquei na duvida se é assim ou não.... mas tentei realizar o exercicio... no aguardo da sua correção.. Obrigado (arquivo em anexo)

(continua)

(continuação)

Remetente	**texto**
Professor 16/08/2018	-, faz o mesmo que vc fez no 2 para a classe filme! Filme f1 = new Filme(); f1.nome ... f1.duracao ... Filme f2 = new Filme (); f2.nome ... Até 20....
Estudante 16/08/2028	Pronto :) mas é preciso fazer um println para mostrar em tela? ou isso vai ficar para um segundo momento? (arquivo em anexo)
Professor 16/08/2018	Outra coisa que eu não tinha visto, String precisa estar dentro de aspas para ser válido. Por exemplo: f1.genero = "drama"; Note o ponto e vírgula ao final de cada instrução. Fazemos isso rodar na aula!
Estudante 16/08/2018	Olá Professor, Boa Noite. Segue o exercício revisado conforme a orientação em aula. (arquivo em anexo)
Professor 28/08/2018	-, mesmo que eu tenha falado para mandar de qualquer jeito, acostume mandar em arquivos do tipo texto plano. Código neles dá menos chance de dar problemas!

Fonte: Autoria própria

Nesta troca temos novamente como ambiente compartillhado o espaço da sala de aula e o virtual. O exercício em questão é o descrito anteriormente sobre filmes e este estudante também entregou um outro exercício sobre classes. Aquilo que ele chamou de “exercício 02” no primeiro e-mail, na verdade era o 01, o exercício do “Top 20” de filmes.

Uma primeira diferença da mensagem da Tabela 14 com relação à Tabela 15 está na troca contínua de mensagens da segunda. Enquanto na primeira resposta ao grupo foi por meio de sugestões, já que o código entrege não tinha problemas de sintaxe. No segundo, existe uma troca direta de mensagens inclusive com uma espectativa de uma resposta. Por exemplo, na terceira linha, o/a estudante escreve “no aguardo da sua correção..”. Na quinta linha, temos diversas perguntas diretas que mostram a mesma espectativa.

A correção de um exercício faz parte das funções esperadas de um professor ou uma professora, mas isto não necessariamente implica em uma relação dialógica. É bem possível e não necessariamente incomum que a correção de um exercício seja focada apenas no objeto ocorrendo como resolução em sala em formato explanatório ou retroalimentação com um valor numérico).

A frase “Fazemos isso rodar na aula!” na linha 6, dá indícios que a solução da questão continuaria na sala de aula e reforça o papel destas trocas de e-mails como parte de um processo mais amplo.

As duas Tabelas (14 e 15) ajudam a mostrar o porquê é importante entender a linguagem como uma atividade e não um sistema abstrato de signos no ensino. Sendo uma atividade, seu aprendizado passa obrigatoriamente por uma série de interações concretas entre pessoas que, neste caso, estão na posição de estudante e de professor. Em uma perspectiva que o dialogismo não é predominante, esta dinânica não é necessariamente central e abre espaço para métodos e práticas que trabalham com uma perspectiva que aprendizagem é uma transferência de conhecimento entre as partes.

Em outras palavras, fugir da metáfora de sistema de signos ajuda a lembrar que o processo de ensino e aprendizagem não pode deixar de lado as pessoas envolvidas no processo concreto de interação. Ter o dialogismo como referência permite um olhar crítico sobre certas práticas corriqueiras como lista de exercícios e exposição de conteúdos.

Quanto ao exercício dos filmes, é possível ver que o estudante entendia que a lista poderia ser uma entidade separada, mas encontrava dificuldades em conseguir especificá-la em código. O trecho 4.5 mostra a classe *Ranking* enviada no primeiro anexo:

Algoritmo 4.5 – Trecho de código entregue para a atividade de filmes que representa uma tentativa de criar uma classe Ranking

```
1
2  Class  Ranking {
3        String   Titulo ;
4        Int  num_visualizacao;
5        Int   avaliacao ;
6  }
```

Fonte: Arquivo pessoal. Todos os direitos reservados.

A classe proposta pelo estudante se assemelha mais a uma relação entre o “top 20” e um filme ou uma mescla entre filme e posição no ranking. O arquivo anexo era do tipo docx que auto formata o que considera o início de uma frase com caixa alta. Por isso, “Int” está com a letra “i” em caixa alta.

Porém, como ele estava ainda entendendo a diferença entre uma classe e um objeto do mesmo tipo, foi julgado que seria um pouco mais complicado tentar trabalhar com mais de uma classe e *arrays*. Em sala, foi feita uma solução de compromisso e entregue no penúltimo e-mail:

Algoritmo 4.6 – Trecho de código entregue para a atividade de filmes que representa um Ranking

```
1
2  Filme  f1  = new Filme();
3  f1.nome = ‘‘O Poderoso Chefão’’;
4  f1.duracao  = 175;
```

```
5  f1. faixa_etaria = 14;
6  f1.genero = ''Drama'';
7  f1. diretor = '' Francis Ford Cappola'';
8
9  Filme f2 = new Filme();
10 f2.nome = ''A Lista de Schindler'';
11 f2.duracao = 195;
12 f2. faixa_etaria = 14;
13 f2.genero = ''Guerra'';
14 f2. diretor = ''Liam Neeson'';
15
16  [...]
17
18 Filme f20 = new Filme();
19 f20.nome = ''Velozes e Furiosos 7'';
20 f20.duracao = 137;
21 f20. faixa_etaria = 14;
22 f20.genero = ''Ação'';
23 f20. diretor = ''James Wan'';
24
25 TopFilmes top = new TopFilmes();
26 top. primeiro = f1 ;
27 top.segundo = f2;
28 top. terceiro = f3;
29 top.quarto = f4;
30 top.quinto = f5;
31 top.sexto = f6;
32 top.setimo = f7;
33 top. oitavo = f8;
34 top.nono = f9;
35 top.decimo = f10;
36 top.decimo_primeiro = f11;
37 top.decimo_segundo = f12;
38 top. decimo_terceiro = f13;
39 top.decimo_quarto = f14;
40 top.decimo_quinto = f15;
41 top.decimo_sexto = f16;
42 top.decimo_setimo = f17;
43 top.decimo_oitavo = f18;
44 top.decimo_nono = f19;
45 top.vigesimo = f20;
```

Fonte: Arquivo pessoal. Todos os direitos reservados.

De um ponto de vista profissional, este código tem um problema claro: Ele não consegue lidar facilmente com uma mudança de tamanho na lista, pois cada posição é identificada por um valor nominal. A forma como está definida a classe TopFilmes exige que seja mexida em sua estrutura para que dê conta de, por exemplo, 21 ou 22 filmes. Caso fossem usadas outras estruturas tais como laço de repetição, listas ou arrays, esta adaptação seria mais fácil, dado um poder maior de generalização para a solução.

Mas, com o aporte da teoria articulada anteriormente, este é um código importante, pois tem um papel pedagógico no processo de mediação. Esta solução articula mais de uma classe, algo que não era exigido pelo exercício e propõe uma solução que, por hora, não envolve outras estruturas tais como um *array* ou um *ArrayList*. Dada a dificuldade inicial do estudante em lidar com classes e objetos, somente trabalhar com estes dois conceitos pareceu ser mais proveitoso no momento.

Este exemplo, se comparado com os apresentados anteriormente, mostra o quanto díficil é estabelecer de uma forma mais estruturada o que pode ser a concretude inicial de cada estudante. Também revela uma clara dependência contextual do código produzido, pois se fosse um estudante diferente orientado por uma outra pessoa, ou as mesmas pessoas em um outro ambiente, provavelmente, o código resultante seria outro. Em uma perspectiva mais restritiva, um exercício simples assim não deveria nem permitir uma amplitude tão grande de soluções.

Isso mostra duas coisas: Primeiro, que a zona de desenvolvimento iminente tem "formatos" diferentes dependendo dos envolvidos. Em outras palavras, ela não é um processo linear em que é possível estabelecer uma série de grupos de conceitos e práticas que sirvam como degraus até as formas "mais adequadas" ligadas à Computação. Nestes casos a zona é situada e, mesmo em estudantes considerados iniciantes, incluem conceitos e práticas diferentes.

Em segundo lugar, o exemplo do trecho 4.6 mostra a importância de entender a escrita de um programa de computador não como a resolução de uma equação. Uma equação admite uma gama estreita e restrita de soluções que não vão levar em conta a pessoa que está a resolvê-la. Neste ponto de vista, normalmente o "correto" é a forma que o professor sabe ou prefere, algo que nem sempre está dentro da zona de desenvolvimento iminente de quem está na posição de aprendiz[18].

Um outro fator interessante que esta atividade permite é uma avaliação subjetiva do nível de conhecimento das pessoas matriculadas na disciplina. Como ela foi inserida no semestre como uma das primeiras atividades, é possível ter uma ideia ao analisar os códigos entregues, dado suas idiossincrasias, ou seja, nas escolhas feitas para a escrita do código-fonte.

A seguir, um código que não aparenta ser muito diferente dos dois primeiros para alguém sem nenhuma vivência em programação de computadores, mas com muitos indícios de que foi escrito por alguém com um nível, pelo menos, razoável de programação em linguagem Java:

Algoritmo 4.7 – Trecho de código entregue para a atividade de filmes

```
1  package filme;
2  import java.util.ArrayList;
3
4  public class Main {
5
6      public static void main(String[] args) {
7
8          int i, j;
```

[18] Quanto à comparação com equações, mais de uma vez ela foi contestada com casos de ensino de matemática básica que divergem do formato "arme e efetue". Entretanto, dada a falta de tradição que o ensino de progrmação tem comparado ao ensino de matemática, argumentamos que a visão que prevalece sobre a equação é a visão das disciplinas de cálculo das engenharias.

```
9         ArrayList<Filme> rankingFilme = new ArrayList();
10
11        Filme f1  = new Filme();
12        Filme f2  = new Filme();
13        Filme f3  = new Filme();
14        Filme f4  = new Filme();
15        Filme f5  = new Filme();
16        Filme f6  = new Filme();
17        Filme f7  = new Filme();
18        Filme f8  = new Filme();
19        Filme f9  = new Filme();
20        Filme f10 = new Filme();
21        Filme f11 = new Filme();
22        Filme f12 = new Filme();
23        Filme f13 = new Filme();
24        Filme f14 = new Filme();
25        Filme f15 = new Filme();
26        Filme f16 = new Filme();
27        Filme f17 = new Filme();
28        Filme f18 = new Filme();
29        Filme f19 = new Filme();
30        Filme f20 = new Filme();
31
32        f1.titulo   = "Lagoa Azul";
33        f1.duracao = 104;
34        f1.genero   = "Drama";
35        f1.direcao  = "Randa Kleiser";
36
37        f2.titulo   = "Star Wars: Episódio VI - O Retorno de Jedi";
38        f2.duracao = 134;
39        f2.genero   = "Aventura";
40        f2.direcao  = "Richar Marquand";
41
42        f3.titulo   = "Star Wars: Eposódio V - O Império Contra-Ataca";
43        f3.duracao = 124;
44        f3.genero   = "Aventura";
45        f3.direcao  = "Irvin kershner";
46
47        f4.titulo   = "Star Wars: Episódio IV - Uma Nova Esperança";
48        f4.duracao = 121;
49        f4.genero   = "Aventura";
50        f4.direcao  = "George Lucas";
51
52        f5.titulo   = "O Senhor dos Anéis: A Sociedade do Anel";
53        f5.duracao = 178;
54        f5.genero   = "Fantasia";
55        f5.direcao  = "Peter Jackson";
56
57        f6.titulo   = "O Senhor dos Anéis: As Duas Torres";
58        f6.duracao = 179;
59        f6.genero   = "Fantasia";
60        f6.direcao  = "Peter Jackson";
61
62        f7.titulo   = "O Senhor dos Anéis: O Retorno do Rei";
63        f7.duracao = 201;
64        f7.genero   = "Fantasia";
65        f7.direcao  = "Peter Jackson";
66
67        f8.titulo   = "Um Contratempo";
68        f8.duracao = 106;
69        f8.genero   = "Suspense";
70        f8.direcao  = "Oriol Paulo";
71
72        f9.titulo   = "Ilha do Medo";
73        f9.duracao = 137;
74        f9.genero   = "Suspense";
75        f9.direcao  = "Martin Scorsese";
76
77        f10.titulo   = "A Praia";
78        f10.duracao = 119;
79        f10.genero   = "Aventura";
80        f10.direcao  = "Danny Boyle";
```

```
81
82          f11. titulo   = "A Origem";
83          f11.duracao = 148;
84          f11.genero   = "Suspense";
85          f11. direcao  = "Christopher  Nolan";
86
87          f12. titulo   = "Star  Wars: Episódio  III  – A Vingança dos  Sith";
88          f12.duracao = 140;
89          f12.genero   = "Aventura";
90          f12. direcao  = "George Lucas";
91
92          f13. titulo   = "O Bar";
93          f13.duracao = 102;
94          f13.genero   = "Suspense";
95          f13. direcao  = "Álex de  la  Iglesia ";
96
97          f14. titulo   = "Me Chame Pelo Seu Nome";
98          f14.duracao = 131;
99          f14.genero   = "Drama";
100         f14. direcao  = "Luca Guadagnino";
101
102         f15. titulo   = " Cloverfield  – Monstro";
103         f15.duracao = 85;
104         f15.genero   = "Terror";
105         f15. direcao  = "Matt Reeves";
106
107         f16. titulo   = "A Volta  dos  Mortos Vivos";
108         f16.duracao = 91;
109         f16.genero   = "Terror";
110         f16. direcao  = "Dan O'Bannon";
111
112         f17. titulo   = "Todo Mundo Quase Morto";
113         f17.duracao = 99;
114         f17.genero   = "Comédia";
115         f17. direcao  = "Edgar Wright";
116
117         f18. titulo   = "Uma Noite de Crime";
118         f18.duracao = 88;
119         f18.genero   = "Terror";
120         f18. direcao  = "James DeMonaco";
121
122         f19. titulo   = "Harry  Potter  e  o  Enigma do Príncipe";
123         f19.duracao = 153;
124         f19.genero   = "Fantasia";
125         f19. direcao  = "David  Yates";
126
127         f20. titulo   = "Harry  Potter  e  as  Relíquias  da  Morte";
128         f20.duracao = 146;
129         f20.genero   = "Fantasia";
130         f20. direcao  = "David  Yates";
131
132         rankingFilme.add(f1);
133         rankingFilme.add(f6);
134         rankingFilme.add(f8);
135         rankingFilme.add(f4);
136         rankingFilme.add(f16);
137         rankingFilme.add(f2);
138         rankingFilme.add(f18);
139         rankingFilme.add(f3);
140         rankingFilme.add(f9);
141         rankingFilme.add(f14);
142         rankingFilme.add(f11);
143         rankingFilme.add(f19);
144         rankingFilme.add(f13);
145         rankingFilme.add(f10);
146         rankingFilme.add(f15);
147         rankingFilme.add(f5);
148         rankingFilme.add(f17);
149         rankingFilme.add(f7);
150         rankingFilme.add(f12);
151         rankingFilme.add(f20);
152
```

```
153             System.out. println ("### RANKING DE FILMES ###");
154
155             j = rankingFilme. size ();
156             for(i = 0; i < j; i++){
157
158                 Filme filmeAux = rankingFilme. get(i);
159
160                 System.out. println (i + 1 + "° - " + filmeAux. titulo );
161             }
162
163         }
164     }
```

Fonte: Arquivo pessoal. Todos os direitos reservados.

Primeiro, tanto o trecho 4.4 quanto o 4.7 foram entregues em arquivos anexos e não no corpo do e-mail, como é comum no começo da disciplina[19], mostrando um conhecimento da organização de uma aplicação Java (existe uma série de regras para além da sintaxe da linguagem, como o nome do arquivo deve ser o mesmo da única classe pública existente dentro dele, a declaração de pacote *package* deve refletir a estrutura de pastas desde a raiz do código-fonte, etc.).

Segundo, o uso de um *ArrayList*, uma classe parte do pacote de utilidades do Java, para resolver o problema de uma lista e o uso de um laço de repetição ao final para "mostrar" na tela os resultados em vez de escrever cada "System.out.println" necessário para mostrar informações dos 20 filmes escolhidos.

Dada a perspectiva de mediação, a atividade mostra algumas formas que o papel de professor pode assumir em uma atividade de escrita de código-fonte. Nas atividades descritas nesta seção, a principal função do professor foi trabalhar dentro da Zona de Desenvolvimento Iminente dos e das participantes e através das interações no dia a dia e por e-mail, tentando auxiliar no processo de "mudança" de concretude. Se, ao começar a atividade as pessoas tinham uma certa relação com conceitos computacionais, neste caso, com aqueles ligados à OO e linguagem Java, espera-se que ao seu final elas cheguem a um novo nível (seja ele qual for), com relações mais fortes com a esfera da computação.

4.5 JOGOS DE CARTAS E PROGRAMAÇÃO DE COMPUTADORES

Super Trunfo é um jogo de cartas comercializado pela Grow, com funcionamento similar ao jogo britânico *Top Trumps* (WIKIPEDIA, 2022). O jogo consiste em um baralho com 32 cartas divididas em oito grupos que vão de 1 até 8 e cada carta dentro do grupo possui uma letra

[19] No começo da disciplina não é usada nenhuma IDE em especial, incentivando o uso de IDEs online, para não haver problemas de versão de sistema operacional e afins. Mais para frente no curso, algumas atividades exigem o uso de IDEs para que sejam feitas de forma satisfatória.

(A, B, C ou D). Entretanto, cada baralho tem um tema (aviões, personagens de séries animadas, países etc.) que define as características que uma carta pode ter. Por exemplo, na Figura 5, temos duas cartas que fazem parte do baralho da série animada "Hora da Aventura"[20] composto pelos personagens da série e cada carta tem as seguintes características: poderes, inteligência, força, espírito aventureiro e amizade.

Figura 5 – Exemplo de duas cartas de Super Trunfo.

Fonte: Autoria própria

A forma mais simples de jogo funciona assim: As cartas são embaralhadas e distribuídas. Cada jogador forma um monte em suas mãos, de tal modo que possa ver apenas a carta de cima. Começa o jogo quem estiver à esquerda do jogador que distribuiu as cartas. Ele escolhe uma das características da sua carta de cima e a lê em voz alta. Depois os jogadores leem, cada um na sua vez, o valor que está na sua carta de cima. Ganha aquele que tiver o maior valor. Caso haja características indicadas com este símbolo (triângulo com a base em cima), vence quem tiver o menor valor. O vencedor da rodada recebe as cartas dos outros jogadores, coloca-as atrás do seu monte de cartas e escolhe uma característica que está na carta seguinte. Empate: Se dois ou mais jogadores possuírem cartas com o mesmo valor, os demais deixam suas cartas na mesa e a vitória é decidida entre aqueles que empataram. Para isso, quem escolheu inicialmente, escolhe outra característica da próxima carta. Ganha todas as cartas da rodada quem tiver o valor mais alto. SuperTrunfo: A carta Super Trunfo é embaralhada com as demais. Suas informações superam as características de todas as cartas marcadas com B, C e D, sem levar em consideração os valores.

[20] *Adventure Time*, no original.

Ela perde apenas se um dos jogadores tiver uma carta marcada com a letra A. Vencedor: Vence o jogador que ficar com todas as cartas do baralho[21].

Em todas as atividades que envolveram a escrita de código, foram disponibilizados baralhos verdadeiros (Figura 6) para que cada grupo pudesse manipular as cartas. Esta etapa cria uma espécie de concretude e contexto inicial compartilhado entre todos envolvidos no processo, um primeiro nível de concretude, nos termos de Meksenas (1992).

Figura 6 – Quatro dos baralhos usados para as atividades.

Fonte: Autoria própria

Existem pelo menos duas questões teóricas importantes sobre esta etapa. Primeiro, ao trabalhar com aprendizagem de leitura e escrita contextualizada, é importante localizar algumas limitações desta abordagem, em especial, com relação às abordagens críticas, como a de Freire (2014). Por exemplo, a questão do "partir do saber" (FREIRE; VASCONCELOS; BRITO, 2006, p.148) em que o educador deve "conseguir entender a leitura do mundo feita pelo educando e, a partir desta leitura, ampliar o seu conhecimento, levando o educando a ter uma visão mais crítica" (FREIRE; VASCONCELOS; BRITO, 2006, p.148) não necessariamente está contemplada neste caso, dado que uma parte considerável do nível concreto para o início da mediação não vem necessariamente do universo dos educandos.

Além desta primeira concretude "artificial", neste caso especificamente, a dimensão crítica também é prejudicada. Na Figura 5 e Figura 6 existem baralhos com conteúdos licenciados (Hora da Aventura, personagens da Pixar e super-heróis da DC) emparelhados com outros temas ligados de forma muito mais indireta a questões caras a Freire (2014), tais como mudança de realidade e consciência crítica.

[21] Texto retirado de uma das cartas auxiliares que vem com um baralho do jogo.

Entretanto, é importante contrastar esta prática com práticas consideradas comuns no ensino de disciplinas técnicas. Bazzo (2011, p.13) coloca que "como regra geral, são considerados habilitados a seguir as carreiras docentes aqueles que possuem um título superior, qualquer que seja ele" com um respaldo no entendimento de que "o domínio dos saberes técnicos da profissão é suficiente para transformar um indivíduo legalmente diplomado num professor" (BAZZO, 2011, p.13).

Bazzo (2011, p.13) ainda complementa que, com isso, estes professores "perpetuam não só os aspectos positivos necessários à manutenção do estilo de pensamento da comunidade profissional, mas também os seus desacertos".

No capítulo 2, mostramos um pouco daquilo que "consideramos" desacertos do campo de ensino em computação, dando destaque para o entendimento raso sobre abstração e pensamento computacional. Ainda, dado o recorte dos textos usados no capítulo 2, as práticas apresentadas nestes textos não necessariamente refletem as práticas correntes nas disciplinas do eixo técnológico de cursos ligados à computação.

Tento isso em vista, por mais que esta atividade não avance muito quanto a questões caras para Freire, Vasconcelos e Brito (2006), ela tenta promover uma mudança quanto às práticas comuns na introdução dos conceitos de OO.

Assim, a programação de um jogo simples de Super Trunfo é uma atividade baseada em projeto que tem como um de seus principais resultados materiais, a escrita de um programa de computador que permita um jogo de Super Trunfo contra o computador[22].

Para alcançar este objetivo, é necessário escrever um código-fonte (nas atividades apresentadas foram usadas a linguagem Java, pois era a linguagem padrão de cada curso) e, como descrito até agora, este processo será encarado sob regras similares à escrita de um texto, em especial na forma como o Círculo de Bakhtin entende o processo.

Quanto ao processo de abstração, existem três apontamentos: 1) As cartas do baralho fornecem um apoio de sentido da mesma forma que Machado (2011) descreve a função da língua materna no aprendizado da matemática, no qual el afunciona como um suporte de significado para a sintaxe sem fala das expressões. 2) Também criam um ponte entre o "dentro" do computador e "fora", deixando evidente que o programa não pode ser entendido sem suas relações com outras entidades materiais. 3) Como colocado anteriormente, fornecem uma parte importante do

[22] Computador deve ser entendido como uma série de regras escritas a fim de simular um outro jogador.

"primeiro" nível de abstração do qual parte a atividade. A outra parte, seriam as próprias pessoas envolvidas no processo.

Esta atividade foi executada três vezes com turmas do Ensino Superior. Uma versão simplificada foi executada em uma turma de terceiro ano da Escola[23]. Em todas as turmas, o objetivo da atividade era trabalhar com os conceitos iniciais de orientação a objetos: Classe, Objeto, Atributos e Valores. Todas as aplicações da atividade foram feitas depois de aulas expositivas e exercícios isolados sobres os quatro conceitos.

São estes quatro conceitos o motivo da escolha do Super Trunfo como base da atividade, uma vez que o jogo oferece relações bem diretas entre suas propriedades e estes conceitos em si. Por exemplo, ao modelar a questão dentro do ponto de vista da Orientação a Objetos, é muito difícil escapar de propor uma classe chamada Carta que seja a representação do conjunto de cartas e entender que cada carta em si é uma instância da classe Carta, ou seja, um objeto que por sua vez possui valores associados aos atributos definidos na classe Carta[24].

No caso do baralho da Figura 5, uma classe Carta poderia ter como atributos: nome, grupo, poderes, inteligência, força, espírito aventureiro e amizade. Também podemos derivar da figura dois objetos, um para cada carta. A Tabela 16 mostra os valores que teriam os objetos que representam as cartas da Figura 5:

Tabela 16 – Relação entre atributos e valores das cartas da Figura 5 .

Atributo	**Valor carta da esquerda**	**Valor carta da direita**
Nome	Billy	Fiona
Grupo	4C	8c
Poderes	20	15
Inteligência	100	85
Força	10	4
Espírito aventureiro	5	5
Amizade	15	15

Fonte: Autoria própria

A primeira coluna da Tabela 16 mostra as propriedades que uma classe Carta poderia conter, enquanto a segunda e terceira colunas mostram os valores que estas propriedades assumem nos objetos em questão. A célula da primeira coluna mais uma célula de outra coluna qualquer representa a relação que existe entre a propriedade da classe com o valor do objeto.

[23] Antes disso, existiu uma versão reduzida. A folha entregue aos estudantes junto com as cartas está disponível em: https://arcaz.ct.utfpr.edu.br/items/show/60

[24] Caso a Orientação a Objetos não seja obrigatória, é possível modelar as cartas como um conjunto de *arrays* em que cada um representa um atributo da carta (nome, grupo, propriedades etc.) e é identificado pelo índice de acesso ao *array.*

Com isso, as cartas podem servir como um apoio para os conceitos de programação orientada a objetos. Por exemplo, caso alguém encontre dificuldade em entender qual a diferença entre um objeto e a classe da qual faz parte ou dificuldade em separar os conceitos de atributo e valor,[25] podemos usar as cartas como uma forma de apoio. Isto faz com que as cartas possam ser usadas como um meio de comunicação concreta entre todas as pessoas envolvidas na atividade, tanto entre estudantes com estudantes quanto entre professores e estudantes.

De forma intencional, esta atividade ignora questões importantes sobre encapsulamento e uniformidade de acesso através de *getters* e *setters*. *Getters* e *setters* são métodos que controlam o acesso às propriedades privadas de um objeto. Em algumas linguagens como Dart e Javascript, estes métodos possuem palavras reservadas próprias para definí-los (*get* e *set*). No caso de Java, são apenas métodos com convenções de nomenclatura, sem nenhum tratamento específico dado pelo processo de compilação.

Getters e *setters* foram excluídos desta atividade, pois em um contexto mais amplo entendemos que ambos exigem um certo domínio de outros conceitos, tais como os trabalhados nesta atividade. Em outras palavras, são mais acessíveis a partir de níveis de concretude diferentes daqueles imaginados para esta atividade. Além disso, no caso de Java, o uso de *getters* e *setters* torna a sintaxe confusa para quem não está acostumado com este "costume" em linguagens de programação[26]. Por exemplo, no trecho de código 4.8, temos uma classe pessoa com dois atributos (nome e sobreNome) com seus respectivos *getters* e *setters*:

Algoritmo 4.8 – Exemplo de getter e setter em Java

```
1
2  class Pessoa {
3    private String nome;
4    private String sobreNome;
5
6    public Pessoa(){
7    }
8
9    public void setNome(String nome){
10     this.nome = nome;
11   }
12   public void getNome(){
13     return this.nome;
14   }
15   public void setSobreNome(String sobreNome){
16     this.sobreNome = sobreNome;
17   }
18   public void getSobreNome(){
19     return this.sobreNome;
20   }
21 }
```

Fonte: Autoria própria. Gustavo Kira, 2021. CC BY-NC-SA 4.0

[25] Ambas percebidas como dúvidas recorrentes, mas ainda carecem de um estudo empírico.
[26] Esta é uma percepção que também carece de uma investigação mais sistemática.

Ao olhar apenas os trechos que trabalham com o atributo nome da classe Pessoa, temos 7 ocorrências da palavra: uma vez na linha 3, duas na linha 9, duas na linha 10, uma na linha 12 e uma na linha 13. Ainda existe o uso do *this* em referência de acesso ao próprio objeto, um conceito nada trivial.

Também existe o fato de que outras linguagens lidam com o problema de formas diferentes. *Python*, por exemplo, não oferece nenhuma forma de encapsulamento obrigatório, ficando a cargo de convenção entre programadores ou aproveitando-se da própria forma como o interpretador funciona para dificultar o acesso aos atributos considerados internos. Sem contar a necessidade de lidar com os modificadores de visibilidade (*public* e *private*, no exemplo).

Do ponto de vista de planejamento da atividade, o Super Trunfo é a parte simples de lidar da primeira concretude. A pessoas envolvidas no processo de ensino e aprendizagem, a outra parte da primeira concretude, são uma questão mais complexa, com destaque para a assimetria quanto ao conhecimento sobre programação de *software*.

Nas três turmas do curso superior, existiam estudantes que nunca tiveram contato com programação antes de entrarem nesta modalidade. Estudantes que fizeram Ensino Médio Técnico em Informática (com no mínimo o conhecimento básico de programação) e pessoas que já trabalhavam em empresas da área com conhecimento consolidado de, pelo menos, metade do currículo da disciplina.

Ignorar esta questão é entrar em contradição direta com o princípio de partir de um nível de concretude para auxiliar os e as estudantes a chegar em um outro nível mais próximo da esfera da computação. Reconhecer esta questão implica entender que cada pessoa está em um nível de concretude diferente e através desta atividade mediadora deveria alcançar um outro nível de concretude, mais próximo da computação. Ou seja, não existe um nível de concretude inicial da atividade, mas níveis de concretude dos participantes.

Com isso, neste caso em especial, o planejamento da atividade consegue, no máximo, especular um nível de concretude de acordo com o currículo, sem levar em conta as pessoas envolvidas de fato, tendo somente um significado pontencial.

Desta maneira, só é possível lidar com as idiossincrasias dos grupos ao longo da atividade, pois são nas interações do cotidiano que elas ficam evidentes. Como esta é uma atividade aberta em que o objetivo não é reproduzir *ipsis litteris* algum tipo de conhecimento (como uma prova de teorema ou identificar a aplicação de uma regra pré-estabelecida), é possível adequar a proposta a cada grupo, de acordo com o seu domínio do conteúdo e interesse.

Para dar uma estrutura básica às pessoas e enquadrar a atividade dentro da disciplina de um curso de Ensino Superior, ela foi dividida em duas partes: primeiro, os grupos deveriam entender o jogo e entregar um documento escrito (nada de código) com suas regras, ações possíveis e enumerar as classes que julgavam necessárias para que o programa funcionasse. Esta é uma etapa importante para aqueles que não tiveram muito contato com programação, pois obriga[27] um contato cedo com questões ligadas à OO e permite o docente tirar dúvidas.

As entregas deveriam ser feitas por email, pois desta forma seria relativamente fácil dar *feedback* para as equipes, conforme já afirmado.

A seguir, a Tabela 17 traz um trecho de como seguiu a troca de mensagens com um grupo (graduação) em que existia um integrante com um conhecimento sobre OO bem avançado, considerando a ementa da disciplina. A primeira mensagem redirecionada ao grupo foi em resposta à primeira versão de código enviado cerca de 5 dias antes.

[27] A questão de obrigar alguém a fazer algo por imposição vai de encontro de diversas práticas educacionais de cunho progressista, as quais esta tese se filia. Inicialmente, esta contradição é aceita dado uma percepção pessoal de que ela ajuda os ou as participantes. Este sentimento ainda carece de validação científica.

Tabela 17 – Textos da troca de e-mails com um primeiro grupo na atividade Super Trunfo. .

Remetente	**texto**
	Tem cara de código Java! Até Demais! Algumas coisas: 1) EmbaralharCartas, RetornarPrimeiraCarta... é mais comum começar com letras minúsculas o nome de métodos. 2) Atributo e AtributoCarta. Essa parte do código tem bem cara de código Java mais avançado, uma vez que parece desnecessariamente complicado, mas se for pensar em expandir ou algo do tipo é relativamente fácil fazer sem criar novas classes! **3) Já que vocês usaram comparator e viram que é algo meio padrão, é normal usar um ternário meio ilegível para fazer o retorno correto. Ex:** **@Override** **public int compareTo(Atributo o) {** **return this.valor > o.valor ? 1: this.valor < o.valor: -1: 0;** **}**
Professor 11/04/2018	4) Jogada. Eu gosto bastante de ter uma classe para a Jogada, mas não sei se deixaria o tirar do topo dentro dela. Do jeito que está, para criar um novo tipo de jogada, preciso necessariamente extender a classe para isso. Enfim, nada para mudar, apenas para levarem em consideração. 5) Rodada: Tem sobrecarga de método! =) Coisas legais para pensar: 6) Tudo que imprime info na tela, poderia estar em classes só de apresentação de das entidades. A gente pode discutir melhor isso em sala alguma hora. 7) Na dificuldade, em vez de usar um switch, é possível usar várias classes que herdam de uma comum. Ex: class IA com um metodo Atributo EscolherAtributo(Carta carta) class IAFacil extends IA e dá override no EscolherAtributo com o random dentro class IAMedio extends IA e dá override no EscolherAtributo.... Com isso, você instancia o objeto correto na hora que sabe qual dificuldade vai rodar, e joga dentro do CPU o objeto. Enfim, isso é coisa pro final da matéria! heheheh. obs: não cheguei a rodar ainda. Abraços!

(continua)

(continuação)

remetente	texto
Grupo 11/04/2018	Bom dia, professor. Algumas questões sobre esses tópicos: **3) Como o compareTo só espera que o resultado seja positivo ou negativo, não seria mais fácil nesse caso específico retornar só a diferença entre os valores? Claro que também precisamos verificar se quem vence é o maior ou menor.** **@Override public int compareTo(Atributo o) {** **return this.atributoCarta.venceMaior ? this.valor - o.valor :** **o.valor - this.valor;** **}** Assim usamos um ternário mais legível e ainda tratamos pra empate em ambos os casos. 6) Até daria pra tentar alguma coisa, mas não consegui uma classe que faria isso. Ainda dá pra tentar implementar alguma coisa. 7) Nesse caso não seria melhor ter uma interface IA pra obrigar quem for implementar um novo nível de dificuldade por exemplo a ter a regra? Esse código não deve compilar, ainda estamos trabalhando nele. Atenciosamente, -
Professor 11/04/2018	Primeiro, não precisam implementar o que eu falei! =) Minhas considerações são para que vcs pensem no processo e vejam o que é possível ou não. **Sobre a 3:** **De acordo com os docs do Java, esse seu ternário esta certo. Eu tenho costume de retornar só -1, 0 e 1, mas não é obrigatório. Entretanto, não sei dizer se tem alguma diferença nas duas formas, sem pesquisar um pouco. Mas acho que deve funcionar.** **Uma coisa legal disso é pensar que vcs poderiam ter duas subclasses de atributos, tipo AtributosDesc e AtributosAsc, cada uma com um compareTo implementado. Um igual ao de vcs e o outro ao contrário. (Não precisa fazer!)** Sobre a 6: Sim, não é algo fácil de imaginar do nada. Eu aprendi isso melhor vendo o funcionamento de alguns programas e frameworks. Mas eu vou tentar passar a ideia básica para vcs. Sobre a 7: Sim e não. Não quis falar em interface porque é algo que nunca discutimos. Como você sabe o que é, sim eu acho que uma interface seria mais interessante. Entretanto, como esta parte do código é um pouco especialista, uma classe base para o problema não seria tão diferente de uma interface. Inclusive, daria até para usar uma classe abstrata (uma classe que não pode ser instanciada). Bom, como estrutura de código, está bem interessante. Vai bem além do que eu espero para um mês de aulas, inclusive usando conceitos que eu só vou entrar daqui algumas semanas. Se precisar de alguma ajuda para fazer compilar me avisem! Até!

(continua)

(continuação)

remetente	texto
	Claro, não vamos sair implementando tudo simplesmente porque você falou, mas são ideias a se considerar com certeza. A questão de uma classe de apresentação é uma coisa que já passou pelo código, mas aumentou um tanto a complexidade. Já a entrada de dados está dentro de uma classe helper para facilitar em todos os usos. Segue em anexo o código-fonte atual (as alterações que ainda não finalizei estão comentadas).
grupo	**Essa ideia de AtributosDesc e Asc é genial e resolve já a questão de encapsulamento das propriedades do atributo sem precisar de muito esforço, então com certeza vai ser implementada.** Sobre a questão de interface, é só mais uma entre várias maneiras de implementar pelo que eu vejo. Por mim tanto faz como é feito isso, não tenho ideia de como pode afetar o desempenho do programa usar classe ou interface. A data de entrega é para essa quinta-feira?

Fonte: Autoria própria

Sobre esta interação, o uso do *Comparable*[28] (em negrito na tabela 17) mostra bem a questão do conhecimento acima do esperado durante o projeto da atividade e o papel que o professor pode desempenhar levando em conta a atividade como mediação e entendendo o que seria a Zona de Desenvolvimento Iminente de Vygotsky (2007).

O uso do conceito de *Interface* na linguagem Java junto com junto com a discussão de sub classes para os atributos exigem uma base bem consolidade em conhecimentos sobre Programação Orientada a Objetos. Se ignorarmos a Zona de Desenvolvimento Iminente, em outras palavras, considerar as pessoas envolvidas no processo como secundárias, a discussão por e-mail não seria necessária, pois o grupo em questão cumpriu o que o projeto exigia. Desta perspectiva, não necessariamente é papel do docente na mediação abordar os indícios de que o grupo já domina os conhecimentos propostos para o projeto.

Aqui também temos um exemplo de uma ação monologizante dentro do processo na sugestão do docente: “1) EmbaralharCartas, RetornarPrimeiraCarta... é mais comum começar com letras minúsculas o nome de métodos.”. Esta recomendação não tem nenhuma implicação quanto à validade do programa de um ponto de vista sintático, mas apenas apresentar o grupo ao uso de uma forma pré-estabelecida.

Na Tabela 17 também temos exemplos de julgamentos de valor de ambos os lados. Por exemplo, no item 2 da primeira linha existe um comentário sobre como um certo trecho do

[28] Comparable é uma interface da linguagem Java que permite trabalhar com uma “ordem natural” dos objetos, parte da classe a implementar. Mais detalhes em: https://docs.oracle.com/javase/8/docs/api/java/lang/Comparable.html

código está complicado sem necessidade. Na última linha da tabela também temos o aceite de uma das sugestões feitas por parte do grupo.

A seguir, apresentamos os trechos de código que levaram a esta troca de e-mails (4.9) e o resultado final (4.10).

Algoritmo 4.9 – Exemplo da classe Atributo na primeira apresentação

```
1
2  package pkgsuper.entidades;
3
4  public class Atributo implements Comparable<Atributo> {
5
6      final int id;
7      private final float valor;
8      private final AtributoCarta atributoCarta;
9
10     public Atributo(int id, float valor, AtributoCarta atributoCarta) {
11         this.id = id;
12         this.valor = valor;
13         this.atributoCarta = atributoCarta;
14
15     }
16
17     public void MostrarAtributo() {
18         atributoCarta.MostrarAtributo(valor);
19     }
20
21     @Override
22     public int compareTo(Atributo o) {
23         float diferencaValor = this.valor - o.valor;
24         if (diferencaValor == 0) {
25             return 0;
26         }
27
28         if (atributoCarta.VenceMaior()) {
29             return diferencaValor > 0 ? 1 : -1;
30         } else {
31             return diferencaValor < 0 ? 1 : -1;
32         }
33     }
34 }
```

Fonte: Arquivo pessoal. Todos os direitos reservados.

Na linha 21, é possível ver o que motivou a discussão sobre o *compareTo*. Além disso, já no primeiro código, existem diversos conceitos aplicados que ainda não tinham sido trabalhados na disciplina, tais como visibilidade (*private* e *public*), sobrecarga de método (*@Override*) e uso de interfaces (*implements*).

Algoritmo 4.10 – Exemplo da classe Atributo e AtributoAsc na entrega final

```
1
2  package pkgsuper.entidades.Atributo;
3
4  public class Atributo implements Comparable<Atributo> {
5
6      public final int id;
7      private final float valor;
8      private final AtributoCarta atributoCarta;
9
10     public Atributo(int id, float valor, AtributoCarta atributoCarta) {
11         this.id = id;
12         this.valor = valor;
13         this.atributoCarta = atributoCarta;
```

```
14      }
15
16      public String mostrarAtributo () {
17          return atributoCarta . mostrarAtributo ( valor );
18      }
19
20      @Override
21      public int compareTo(Atributo o) {
22
23          if (this . valor == o. valor ) {
24              return 0;
25          }
26
27          return atributoCarta . retornarVencedor ( this . valor , o. valor );
28      }
29
30      public int ordenarAtributo ( Atributo o) {
31          return this . atributoCarta . ordenarAtributo (o. atributoCarta );
32      }
33
34      public int retornaIndice () {
35          return atributoCarta . retornaIndice ();
36      }
37  }
38
39  package pkgsuper. entidades . Atributo ;
40
41  public class AtributoAsc extends AtributoCarta {
42
43      public AtributoAsc( int indiceAtributo , String descricao , String unidadeValor) {
44          super( indiceAtributo , descricao , unidadeValor);
45      }
46
47      @Override
48      int retornarVencedor ( float valor1 , float valor2 ) {
49          float diferenca = valor2 - valor1 ;
50          return diferenca > 0 ? 1 : -1;
51      }
52  }
```

Fonte: Arquivo pessoal. Todos os direitos reservados.

Já no trecho retirado do código final entregue, é possível ver que os métodos não começam mais com caixa alta (o integrante com mais experiência em programação do grupo trabalhava com C#) e foi criada uma classe chamada “AtributoDesc” (apresentada acima) e outra “AtributoAsc”.

É possível argumentar quanto à necessidade da classe “AtributoCarta” no modelo apresentado. Mas como foi comentado, entender a escrita de um programa como a escrita de texto e não a resolução de uma equação, implica aceitar que o produto final pode ter divergências. É justamente este ponto que permite dar conta dos diversos perfis.

Gostaríamos de chamar a atenção para duas questões sobre trocas de e-mail e a escrita dos códigos-fonte. A primeira é a existência de uma intextextualidade entre o código-fonte e o e-mail feita pelo uso de “retalhos de código”[29] como referência direta ao código fonte. Além disso, a discussão feita no e-mail é fundamental para entender o processo pelo qual o código-fonte é

[29] *Code snipets.*

alterado. Em segundo lugar, é possível identificar no código entregue (4.10) indícios materiais da dialogicidade do processo, uma vez que a discussão do e-mail resulta em uma alteração direta do código.

Uma questão ainda não muito bem trabalhada é a assimetria de conhecimento dentro das equipes. Neste caso, um dos integrantes possuía um conhecimento muito maior que os outros dois. No final deste capítulo, retomamos esta discussão com um pouco mais de profundidade.

Na Tabela 18, temos outro trecho da troca de e-mails com um segundo grupo sobre a mesma atividade:

Tabela 18 – Textos da troca de e-mails com um segundo grupo (B) na atividade Super Trunfo

Remetente	**texto**

Grupo
03/04/2018
	Beleza, outra coisa,
	estou tendo problemas ao impedir que o usuário digite uma letra ao invés do inteiro dos atributos da carta,
	estou usando try, catch, para evitar essa exception,
	Erro Exception in thread “main” java.util.InputMismatchException ,
	ele não dá o erro, mas também não dá o print dentro do catch,
	Segue segmentos do código com o método escolherAtributo(),
	`public void escolherAtributo() {`
	`Scanner scan = new Scanner(System.in);`
	`try`
	`{`
	`System.out.println("\n0 - Altura");`
	`System.out.println("1 - Comprimento");`
	`System.out.println("2 - Peso");`
	`System.out.println("3 - Viveu há");`
	`System.out.println("4 - Sair do jogo");`
	`System.out.println("Escolha o atributo: ");`
	`escolha = scan.nextInt();`
	`}`
	`catch(InputMismatchException exception)`
	`{`
	`System.out.println("Insira um valor entre 0 e 4 para continuar!");`
	`}`
	`do {`
	`jogador.escolherAtributo();`
	`} while (jogador.escolha != 0 && jogador.escolha != 1 &&`
	`jogador.escolha != 2 && jogador.escolha != 3 && jogador.escolha != 4);`

Professor 03/04/2018	Preciso do código todo pra ver melhor... Mas tem que ver se é essa excessão que tá sendo jogada mesmo, e se não tá sendo pegada em outro lugar...
Grupo 03/04/2018	(código fonte como anexo no email)
Professor 03/04/2018	Pra mim ele está imprimindo o catch. O problema é que quando você "pega"uma Exception, o programa não morre, pois você, teoricamente, está tratando ela. Pelo que vi, falta você integrar essa parte do do, while com a exception.

Fonte: Autoria própria

No caso deste grupo, também com um conhecimento relativamente mais alto do que o esperado, as dúvidas giraram em torno do uso do *try . . . catch* em Java. O primeiro e o segundo grupos mostram a amplitude de formas que o código-fonte da atividade pode assumir e deixa claro o quão diferentes podem ser os diversos níveis de concretude.

A seguir, na Tabela 19, temos outra troca de e-mails com outro grupo em que os estudantes não tinham um conhecimento tão avançado quanto aos dois primeiros apresentados.

Tabela 19 – Textos da troca de emails com um terceiro grupo na atividade Super Trunfo

Remetente	**texto**
Grupo 02/04/2018	Oi, professor... eu tô tentando entender algumas coisas que ainda tá abstrato. A princípio, as 3 mais importantes dúvidas para me tirar dessa bagunça seria: 1- como eu posso pegar a primeira carta a ser exibida no deck do usuário (que considerei o topo) para puxar as informações que ele optou comparar? Ex.: usuário quis comparar velocidade máxima com a primeira carta a ser exibida no deck do computador. Como puxo essas informações? Usando ii = 0 do for do usuário e comparar com o jj = 0 do computador? Minha cabeça está algo tipo assim: if(ii[0].cartas.get(ii).velocidademax > jj[0].cartas.get(jj).velocidademax){ . . . } 2 - Como eu posso apagar essa posição ii[0] que é o topo do usuário e passar para baixo do monte dele mesmo? Seria algo como ii[0] passasse a ser ii[15] e ii[1] caisse para 0, virando o topo? 3 - Me socorre
Grupo 02/04/2018	(código fonte no corpo do email)
Professor 02/04/2018	-, estou escrendo no celular, então vai se tornar partes. Mais pro fim da tarde eu estarei na frente de um PC, daí posso ver melhor. 1 para pegar a carta do topo: Carta c = cartas.get(0); Para pegar a carta do topo é retirá-la do arraylist: Carta c = cartas.remove(0); Se entendi direito, cada jogador de vcs não têm um baralho próprio, certo? Pq se tivesse era soh executar a segunda operação no arraylist do jogador.
Professor 02/04/2018	que que eh ii[0]? 2 com arraylist lista g soh usar o remove. Se for arraylist, vc precisa de uma variável para indicar o topo. Tipo internet topo = 0; se tirar uma carta, topo++; array[topo].

(continua)

(continuação)

Remetente	texto
Grupo 02/04/2018	acho q foi o q fizemos no exercício de EDA, que era indicar o topo e tal... Eu fiz o programa mas n sei se ta correto.. eu fucei na net e achei esse .remove. Funcionou.. mas preciso criar condições do tipo: até quando é pra remover? eu usei: cartas.remove(2) no final do for que usei pra repetir o programa, para somar as pontuações e ir eliminando as cartas. Ficou meio confuso a ideia do topo lá... mas vo dar uma olhada no meu programa de pilha ashufas vai dar uma luz. dps te mando o programa para vc analisar, se vc quiser
Professor 02/04/2018	Remove funciona com arraylist! Array lista eh uma pilha/fila/lista pronta! Vc só tem que usar. Com array, fica parecido com eda mesmo, pois você tem que fazer os mecanismos de funcionamento de pilha. Usar o cartas.remove(0) remove o primeiro valor. cartas.remove(2), remove o terceiro e assim por diante. Se tiver dúvidas, manda!
Professor 02/04/2018	Até quando remover, ou quando remover. Normalmente precisa de condições sim. Sobre a comparação, Olha o exemplo que tem no repositório onde tem as informações do trabalho. Tem um código lah que deve ajudar.
Grupo 02/04/2018	Oi, professor. Eu to tentando entender como criar essa referencia ao topo de cada monte... pq acho q isso ta implicando na remoção. quando vo remover a 8ª carta de cada monte, ele da erros desse tipo: Exception in thread "main"java.lang.IndexOutOfBoundsException: Index: 16, Size: 16 at java.util.ArrayList.rangeCheck(ArrayList.java:657) at java.util.ArrayList.get(ArrayList.java:433) at Main.main(Main.java:388)
Professor 02/04/2018	O erro ta dizendo do que vc tá tentando remover o item com id 16(17o item) e ele não existe! Entrei no ônibus agora e só vou conseguir responder mais tarde!

Fonte: Autoria própria

O terceiro grupo apresenta dificuldades quanto à sintaxe e ainda tem dúvidas de como implementar as ações que foram identificadas no jogo de Super Trunfo. Nesta troca de email, é interessante notar que existia uma dificuldade em diferenciar um *array* de um *ArrayList*[30], estruturas muito parecidas de um ponto de vista semântico, mas com características diferentes dada suas implementações.

[30] *Array* é a mesma estrutura de dados encontrada na linguagem C, em que variáveis de mesmo tipo são guardadas de forma sequencial na memória do computador, enquanto *ArrayList* é uma das implementações da estrutura de dados do tipo lista na linguagem Java.

As duas últimas mensagens mostram um exemplo do que pode ser trabalhar dentro da Zona de Desenvolvimento Iminente em atividades que envolvem a escrita de código-fonte: O auxílio na interpretação do que são os erros em tempo de execução ou compilação. No trecho da Tabela 19, temos um erro de acesso a uma referência inexistente na memória do programa. Para alguém com pouca experiência na escrita de códigos, é importante ter a certeza de onde está o erro para descobrir como consertá-lo e evitar que um pequeno erro trave qualquer outro desenvolvimento.

Em algumas situações, as trocas de e-mail foram importantes para que os grupos não ficassem parados com erros de compilação e ou execução, uma vez que as aulas presenciais ocorriam duas vezes por semana em dias consecutivos. Ou seja, as conversas pessoais com os grupos ocorriam com intervalos de uma semana e um erro como o do código do terceiro grupo poderia fazer com que o grupo ficasse parado por dias.

É importante colocar que por mais que estes e-mails e os códigos sejam o principal suporte concreto deste estudo de caso, eles são auxiliares ao processo de ensino e aprendizagem. Por exemplo, foi relativamente comum dar o *feedback* por e-mail e discutir o código entregue e possibilidade de mudanças em sala na aula seguinte. A Tabela 20 e a 21 mostram exemplos destes casos.

Tabela 20 – Textos da troca de emails com um quarto grupo na atividade Super Trunfo

Remetente	**texto**
Grupo 02/05/2019	Segue o link para acesso do meu repositório do super trunfo: https://github.com/-/-

(continua)

(continuação)

Remetente	texto

	Seria mais legal se a sua classe Jogo não tivesse o "public static void main"nela.
	
	Nunca usou a classe Jogador?
	
	método cartaAleatória: já que vcs passam um array, porque não retornar a carta já?
	
	Isto está engraçado. Não esta errado, mas poderia ser mais fácil:
	
	`Carta objJogador = new Carta(`
	`this.baralhoTotal.get(indice1).grupo,`
	`this.baralhoTotal.get(indice1).nome,`
	`this.baralhoTotal.get(indice1).altura,`
	`this.baralhoTotal.get(indice1).comprimento,`
	`this.baralhoTotal.get(indice1).peso,`
	`this.baralhoTotal.get(indice1).viveuHa,`
	`this.baralhoTotal.get(indice1).superTrunfo`
	`);`
	`this.baralhoTotal.remove(indice1);`

Professor
08/05/2019
	O método remove do ArrayList retorna o objeto removido, o que significa que poderia ser feito assim:
	
	`Carta objJogador = this.baralhoTotal.remove(indice1);`
	
	Mas, salvo alguns casos especiais, a forma de vcs funciona.
	
	o método "escolhePropriedade(int i)"funciona, mas poderia usar algo mais claro do que 0 ou 1 para decidir se pede algo pro usuário ou sorteia randomico.
	
	"comparaCartas": faltou tempo ou não sabiam como resolver?
	
	cartasToJogador, mesma coisa de antes:
	
	`this.jogador.add(obj1);`
	
	funcionaria de forma similar.
	
	De maneira geral, mesmo usando métodos, o código de vcs ficou um pouco com cara de estruturado com métodos. Mas para um primeiro trabalho eu acho que ta ok até!

Fonte: Autoria própria

Tabela 21 – Textos da troca de emails com um quinto grupo na atividade Super trunfo.

Remetente	**texto**
Grupo 17/04/2019	Boa noite professor, Segue trabalho do Super trunfo finalizado. Grupo: - - – Atenciosamente, - Tel.: -

(continua)

(continuação)

Remetente	texto

Professor
25/05/2019
	1) Sobre o método de embaralhar, vejam se o trecho de código abaixo faz sentido:
	`int quantidade = 24;`
	`int trocas = 100;`
	`int[] ordem = new int[quantidade];`
	`for(int i = 0;i< quantidade;i++){`
	`ordem[i] = i;`
	`}`
	`Random gerador = new Random();`
	`for(int i = 0;i<trocas;i++){`
	`int a = gerador.nextInt(quantidade);`
	`int b = gerador.nextInt(quantidade);`
	`int temp = ordem[a];`
	`ordem[a] = ordem[b];`
	`ordem[b] = temp;`
	`}`
	
	Notem que no final temos um array com valores trocados de posição, que pode ser usado para o array de cartas de vcs. Isso evita os diversos whiles, o que significa que o código é mais escalável do que a forma de vcs.
	
	2) if (this.jogador1.cartasDoJogador[0].nome == "Leão").... Porque não usar o atributo supertrunfo?
	
	3) String comecinho;
	Eu ri, mas não é um bom nome para as strings que vem do terminal....
	
	4) jogo.jogador1.numerodecartas != 24 && jogo.jogador2.numerodecartas != 24 poderia ser um método, tipo:
	
	`public boolean continuarJogo(){`
	`return this..jogador1.numerodecartas != 24 &&`
	`this..jogador2.numerodecartas != 24;`
	`}`
	
	5) Existem diversas partes do código q podem ficar um pouco mais claros, trocando algumas coisas de lugar. Por exemplo, EU faria algo assim com os "ifs"do método jogadacomputador
	
	`String atributo = ;`
	`float valor2 = 0;`
	`float valor1 = 0;`
	`switch (this.opcaodajogada){`
	`case 1:`
	`atributo = "Altura";`
	`valor1 = this.jogador1.cartasDoJogador[0].altura;`
	`valor2 = this.jogador2.cartasDoJogador[0].altura;`
	`break;`
	`case 2:`
	`atributo = "Peso";`
	`valor1 = this.jogador1.cartasDoJogador[0].peso;`
	`valor2 = this.jogador2.cartasDoJogador[0].peso;`
	`break;`
	`.....`
	`}`
	`System.out.println("O oponente escolheu "+atributo+":");`
	`System.out.println();`
	`System.out.println("A sua "+atributo+ "vale: "+valor1);`
	`System.out.println("A do seu adversário vale: "+valor2);`

(continua)

(continuação)

Remetente	texto

Professor
25/05/2019
	6) Também acho que a classe jogador poderia ter métodos que ajudem a separar o código em partes. Por exemplo, o trecho:
	
	//reorganiza as cartas do j1
	for(int trocador=0; trocador < 26; trocador++){
	
	jogador1.cartasDoJogador[trocador]=jogador1.cartasDoJogador[trocador+1];
	}
	
	poderia ser um método da classe jogador. Algo do tipo:
	
	void reorganizarCartas(){
	for(int trocador=0; trocador < 26; trocador++){
	this.cartasDoJogador[trocador]=
	this.cartasDoJogador[trocador+1];
	}
	}
	
	o que faria a gente poder chamar esses métodos para cada jogador dentro do método reorganizarcartas do jogo:
	
	jogador1.reorganizarCartas(); jogador2.reorganizarCartas();
	
	7) método hack: mesmo deletando o conteúdo, ainda sobrou as chamadas no código.

Fonte: Autoria própria

De forma geral, o uso do Super Trunfo é um exemplo de como articular as teorias apresentadas até aqui em uma atividade educacional. Nele, temos o entendimento que a abstração é um processo de mediação entre duas concretudes (de mesma forma que Meksenas (1992) entende o processo), deixando de lado o seu valor quanto ao oposto ao concreto. Isso tem implicações importantes para o ensino e aprendizagem de fundamentos da computação, em especial, no uso de linguagens computacionais.

4.6 TABULEIROS E PROGRAMAÇÃO DE COMPUTADORES

Nesta seção, será descrita uma das atividades que tiveram como mote o jogo de xadrez, chamada de “Mini Chess”. Ela foi aplicada para uma turma de segundo ano da Escola. Também existiu uma segunda, chamada somente de “Xadrez”, que foi aplicada para três turmas na Faculdade.

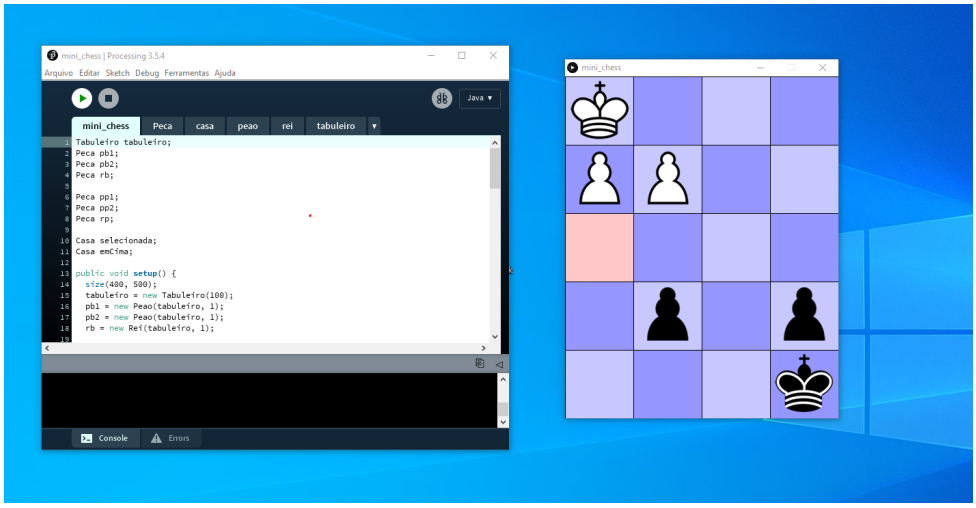
As duas atividades partem de uma premissa um pouco diferente das atividades de Super Trunfo, pois oferecem um código-fonte inicial como ponto de partida do processo. Na escrita de programas de computador, não é raro interagir com códigos prontos, escrito por um outro grupo

de pessoas a que se tem acesso ou não. Dado o nosso entendimento de texto indissociável de seu processo de escrita e contexto, julgamos esta uma modalidade importante de ser trabalhada no processo de ensino de fundamentos de programação.

Além disso, se pensarmos na questão das concretudes, esta atividade oferece um conjunto de códigos-fonte como instrumentos e um ponto de partida concreto. Os materiais fornecidos seriam uma primeira concretude, esta já mais próxima dos conceitos de informática ou computação do que os usados, por exemplo, nos jogos de cartas.

O "Mini Chess" consistia em um programa escrito para o ambiente *Processing*[31] usando orientação a objetos. A escolha do ambiente se deu por, principalmente, dois motivos: 1) A orientação visual do *Processing* é um recurso que permite trabalhar com um nível de concretude maior, permitindo que as e os estudantes possam ver os resultados de seus algoritmos diretamente no tabuleiro (virtual), além disso o *feedback* rico que a interface visual oferece comparada com um terminal, como mostra a Figura 7. 2) *Processing* é um ambiente que usa uma linguagem derivada de Java, a linguagem usada como base durante todo o processo de ensino desta turma.

Figura 7 – Interface gráfica do programa no início do projeto.



Fonte: Autoria própria

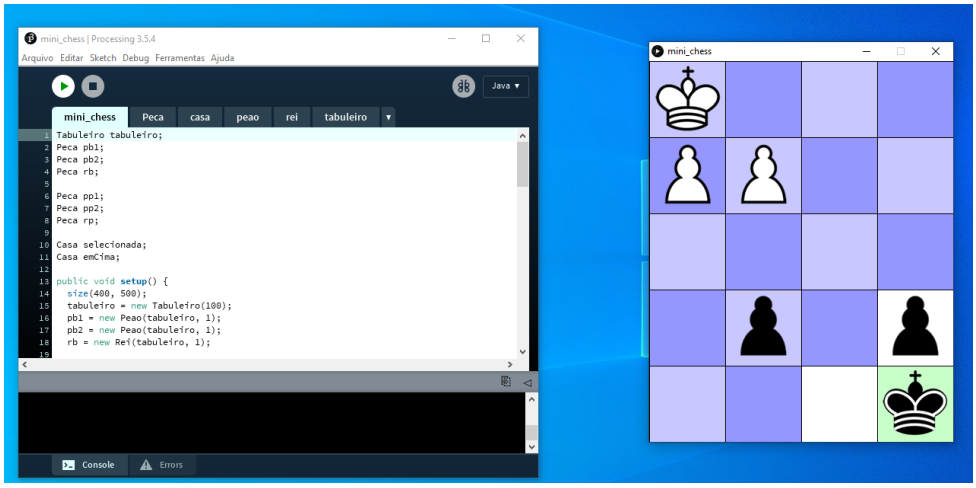
Esta atividade teve como objetivo trabalhar os conceitos de herança e polimorfismo. Ela se aproveita da própria estrutura do jogo de xadrez em que existem diferentes peças que se movimentam no tabuleiro de acordo com regras pré-estabelecidas. De um ponto de vista da OO, podemos dizer que existem 32 peças com propriedades em comum (podem ter uma posição no tabuleiro, uma cor e possuem uma série de casas para as quais podem se mover em um dado

[31] *Processing* é um ambiente de programação criado para fazer com que seja mais fácil o desenvolvimento de aplicações visuais (FRY; REAS, 2021).

instante), mas dependendo do seu tipo (peão, rei, bispo, torre, cavalo e rainha), podem se mover para casas diferentes do tabuleiro.

O programa fornecido às e aos estudante era composto de cinco classes iniciais: Tabuleiro, Peça, Casa, Peão e Rei. Tanto o peão quanto o rei foram implementados como especializações da classe Peça, entretanto seus movimentos não estavam finalizados. Além disso, também já estava implementada a visualização do tabuleiro na tela, assim como uma forma de visualizar os movimentos de cada peça do tabuleiro, conforme Figura 8.

Figura 8 – Rei selecionado (fundo verde) e opções de movimentação (fundo branco). O movimento ilustrado não está de acordo com as regras do xadrez.



Fonte: Autoria própria

Com isso, foram pedidos para as e os estudantes que tentassem implementar os seguintes itens: 1) Conserte os movimentos do rei e dos peões. 2) Adicione uma torre de cada cor. 3) Adicione um bispo de cada cor. 4) Adicione uma rainha de cada cor. 5) Adicione um cavalo de cada cor. 6) Faça as opções de movimento da torre funcionar. 7) Faça as opções de movimento do bispo funcionar. 8) Faça as opções de movimento da rainha funcionar. 9) Faça as opções de movimento do cavalo funcionar.

O primeiro item tinha como função direcionar o foco das pessoas para como os conceitos de herança e polimorfismo foram usados neste programa. A classe Peça tem um método chamado "movimentosPermitidos" que deve retornar um *ArrayList* de objetos do tipo Casa. Toda vez que uma peça é selecionada na interface gráfica, são mostradas todas as casas que este método retorna. Ou seja, cada peça deve ter uma lógica para decidir quais casas são acessíveis em um dado momento. Nesta versão da atividade não é necessário fazer o jogo "funcionar". Apenas fazer com que as peças mostrem as casas corretas ao serem selecionadas.

Nos conjunto de itens que vai do 2 ao 5, era preciso criar novas classes para as peças aparecerem no tabuleiro. Os arquivos de imagens já existiam no programa, apenas era preciso criar as novas peças que deveriam estender a classe Peça para funcionar.

Nos itens 6, 7, 8 e 9 trabalham com a especialização de comportamento da classe estendida de fato. O desafio para as e os estudantes não era conseguir escrever o código referente ao polimorfismo, mas trabalhar em cima do programa que as classes filhas tenham, de fato, um comportamento específico.

Este exercício teve duração de duas semanas, com um dia em sala de aula em cada semana (normalmente são quatro aulas de 50 minuto por semana, mas neste caso, o feriado de 8 de setembro caiu em um dos dias). Dada a baixa quantidade de horas em sala com a turma e a grande quantidade de estudantes (36 no total), o atendimento em sala nem sempre foi ideal.

Esta atividade foi feita na disciplina de Programação Orientada a Objetos na Escola com uma turma do segundo ano do curso técnico em informática. Como a Escola tinha alguns tabuleiros de xadrez foi levantada a possibilidade de usá-los como parte da atividade, mas por motivos circunstanciais, não foi possível o seu uso.

Nesta atividade podemos considerar que o tabuleiro de xadrez, seja ele físico ou o virtual, é também uma concretude inicial. Esta concretude está mais próxima do cotidiano dos estudantes, principalmente por lidar com questões espaciais e de configuração. Não é um jogo simples, mas suas regras e configurações são de acesso relativamente fácil, o que ajuda neste caso.

De forma geral, esta atividade teve uma alta taxa de desistência: de 35 estudantes, 16 não entregaram. Entretanto, esta quantidade não foi muito diferente das demais atividades do ano[32]. Os exemplos a seguir são de estudantes que tiveram um bom desempenho na matéria ao longo de todo o ano letivo[33].

A seguir, na Tabela 22, temos uma troca de e-mails com um grupo que tinha um integrante com um alto interesse pelo conteúdo da disciplina. A tabela começa com uma resposta à análise do código enviado.

[32] Tentamos discutir um pouco esta questão ao final deste capítulo.

[33] É importante colocar que o método de avaliação da instituição era a atribuição de conceitos, como comentado anteriormente. Por bom desempenho consideramos estudantes que entregaram quase todos os trabalhos no ano e que pareceram conseguir usar conceitos computacionais para resolver os problemas propostos

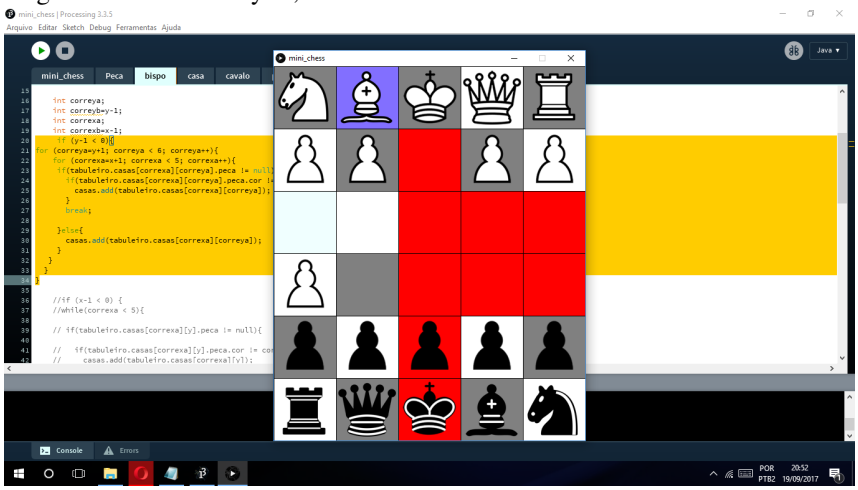
Tabela 22 – Textos da troca de e-mails com um primeiro grupo na atividade Mini Chess .

Remetente	**texto**
Professor 05/09/2017	Olá: Vou tentar responder com a lógica e deixo a implementação com vocês, pode ser? Se tiverem muitas dificuldades, avisem. 1) A torre deve selecionar todos os quadrados nos DOIS lados dela e na FRENTE e ATRÁS, certo? 2) Em cada direção, vocês precisam ir selecionando cada quadrado ENQUANTO ele não tiver uma peça nele ou ele não for o último. 3) Entendem que devem fazer esse processo para cada um dos lados? Com isso vcs devem conseguir começar. Lembrem-se que é através dos índices que vcs "pegam"outras casas... [ ]s
Grupo 14/09/2017	Professor... Já consegui fazer os movimentos principais da torre. Mas tenho uma pergunta, tem que fazer o movimento das peças de qualquer local do tabuleiro? Por exemplo a torre, estou com um problema, quando tento colocá-lo em um local fora de sua posição inicial/bordas do tabuleiro, ele não executa meu while. Estou tentando usar o tabuleiro.casas.lenght, mas não entendi direito como usá-lo, a nova versão do meu tabuleiro está disponível no drive, se o senhor puder me ajudar, já poderei aplicar o conceito da torre no bispo e rainha. Senão, posso tentar terminar amanhã na aula, mas uma ajuda agora seria bem vinda.

Professor
14/09/2017
	Seria bom se ela se mexesse de qualquer lugar, mas se não der, ok.
	
	Sugestão de como fazer a torre pegar todas as casas acima dela, acho que vcs conseguem replicar a lógica para o resto!:
	
	//crio uma variavel que vai aumentar de acordo com os indices y acima da posicão da torre.
	//yf é igual a posicao da minha peça -1, isto é um em cima. int yf = y-1;
	
	//enquanto o yf for maior que -1, ou seja, esteja dentro do tabuleiro
	
	while(yf > -1){
	//tabuleiro.casas[x][yf].peca -> mesmo x da peca, e o yf, na primeira iteracao é um acima, na segunda dois... etc
	// se existir uma peca na casa sendo vista
	if(tabuleiro.casas[x][yf].peca != null){
	//tem a peca e a peca é de cor diferente, adiciono a casa como possibilidade de movimentacao
	if(tabuleiro.casas[x][yf].peca.cor != cor){
	casas.add(tabuleiro.casas[x][yf]); }
	//sempre que tem uma peca, é a ultima casa, por isso saio do while
	break;
	//se não tem peca, é uma possibilidade de movimento, logo coloco a casa como possibilidade
	}else{
	casas.add(tabuleiro.casas[x][yf]);
	}
	//subo um acima do atual o yf.
	yf–;
	[ ]s

(continua)

(continuação)

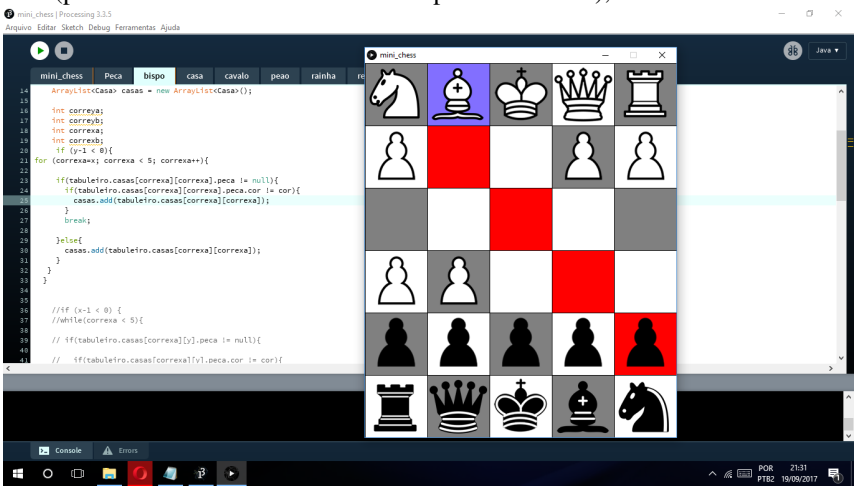
Remetente	**texto**
	Professor, meu código para o bispo não tá dando muito certo. Era para pegar a diagonal usando x+1 e y+1, mas ele fica assim: 

Grupo
19/09/2017
	O código é:
	`int correya;`
	`int correyb=y-1;`
	`int correxa;`
	`int correxb=x-1;`
	`if (y-1 < 0){`
	`for (correya=y+1; correya < 6; correya++){`
	`for (correxa=x+1; correxa < 5; correxa++){`
	`if(tabuleiro.casas[correxa][correya].peca != null){`
	`if(tabuleiro.casas[correxa][correya].peca.cor != cor){`
	`casas.add(tabuleiro.casas[correxa][correya]);`
	`}`
	`break;`
	`}else{`
	`casas.add(tabuleiro.casas[correxa][correya]);`
	`}`
	`}`
	`}`
	`}`
	Acho que estou tendo problemas com a matriz, mas não sei como resolver...

Professor 19/09/2017	Conseguem pensar que, na verdade, precisam de somente um for para andar na diagonal? Veja se isso ajuda, se não der, dou mais dicas! [ ]s
Professor 19/09/2017	Pensem que o valor de x e de y sempre é igual. No de vcs, o for começa na diagonal, mas anda em todos depois. O for não é no x ou no y, é nesse valor igual!

(continua)

(continuação)

Remetente	**texto**
Grupo 19/09/2017	Professor, o valor de x e y não é igual. O y=0 e x=1, quando tentei aplicar sua dica (provavelmente deve ter sido incompetência minha), mas deu isso: 
Professor 19/09/2017	Hauahuahuaha, nem é incompetência! Primeiro, parece muito mais diagonal agora, não? Falta fazer começar na casa certa! Se começar na casa do lado direito, parece que arruma! Programar é isso mesmo. A gente vai resolvendo problemas aos poucos. Se ainda não conseguir, me avise que melhoro a dica! hehehe [ ]s

(continua)

(continuação)

Remetente	**texto**

Grupo
21/09/2017
	Professor, no bispo de baixo consegui movê-lo para as duas direções. Mas o de cima só consigo para uma
	 
	`if (y-1 < 0){`
	`  for (correxa=x; correxa < 4; correxa++){`
	`  if(tabuleiro.casas[correxa+1][correxa].peca != null){`
	`  if(tabuleiro.casas[correxa+1][correxa].peca.cor != cor){`
	`  casas.add(tabuleiro.casas[correxa+1][correxa]);`
	`  }`
	`  break;`
	`  }else{`
	`  casas.add(tabuleiro.casas[correxa+1][correxa]);`
	`  }`
	`  }`
	`  for (correxa=x; correxa < 4; correxa–){`
	`  if(tabuleiro.casas[correxa-1][correxa+1].peca != null){`
	`  if(tabuleiro.casas[correxa-1][correxa+1].peca.cor != cor){`
	`  casas.add(tabuleiro.casas[correxa-1][correxa+1]);`
	`  }`
	`  break;`
	`  }else{`
	`  casas.add(tabuleiro.casas[correxa-1][correxa+1]);`
	`  }`
	`  }`
	`}`

Grupo 22/09/2017	Tem algo estranho, -. Depois que eu explicar o que eu preciso hj, a gente vê isso.
Grupo 22/09/2017	Professor, a segunda condição da classe bispo. Ele não tá entrando lá, para poder usar o bispo preto.

Fonte: Autoria própria

Duas questões surgem desta troca, primeiro o uso de *prints* de tela com auxílio à comunicação e a parte do código em que estavam as condições que não levavam ao comportamento esperado pelos estudantes.

Sobre a primeira questão, é interessante notar que o grupo mudou a cor das casas que representam os movimentos possíveis de uma peça, ou seja, executou uma alteração em um trecho de código já existente a fim de resolver uma questão pessoal. Também mostra que o grupo fez proveito do *feedback* visual oferecido para identificar os problemas que estavam ocorrendo com o seu algoritmo, dando indícios que o uso de interface gráfica pode cumprir um papel de ferramenta que auxilia o processo de mediação entre a concretude daquilo que os estudantes conhecem com aquela mais próxima dos conceitos computacionais.

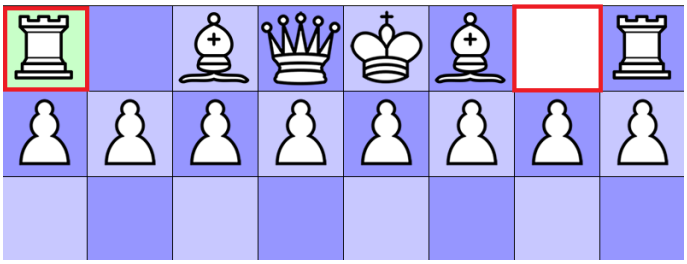
Nesta troca de emails, o grupo precisa de auxílio com questões ligadas ao processamento de matrizes com o uso de laço de repetição. Entretanto, é justamente a possibilidade de tratar cada peça de uma forma diferente que estamos efetivamente trabalhando com a herança e polimorfismo.

Este é um exemplo interessante de como funciona o processo mediado por instrumentos e signos. O grupo, ao alterar o código fornecido, está a usar de instrumentos para alterar sua realidade material. De forma concomitante, os signos são articulados e, através do trabalho inter pessoal, espera-se que sejam incorporados aos processos mentais, ao longo do tempo.

O trecho também mostra que o grupo não teve dificuldades em lidar com o conceito de herança e polimorfismo (conseguiu criar classes que representam as outras peça e as colocou no tabuleiro virtual), mas teve mais dificuldade em lidar com os laços de repetição dentro de arrays multidimensionais.

A relação entre os conceitos específicos de OO e outros mais difusos associados à lógica de programação ficam evidentes aqui. Neste trecho e no próximo, as tabelas mostram que nos tipos de atividades que propomos, as dúvidas e questionamentos que as e os estudantes encontram não estão restritos aos conceitos propostos pela atividade.

Tabela 23 – Textos da troca de e-mails com um segundo grupo na atividade Mini Chess

Remetente	**texto**
Grupo 22/09/2017	profe, agora que eu coloquei mais uma torre ta acontecendo isso aqui:  A outra torre ta funcionando de boas. mandei por arquivo de texto pra ficar mais fácil Vou tentar fazer o cavalo aqui e mando quando conseguir ou precisar de ajuda kk (5 anexos)
Professor 22/09/2017	-, não abri os arquivos ainda, pois não estou na frente do pc. Veja se vc não tá usando o mesmo objeto torre duas vezes. Igual o peão lah. Se não for, vejo mais tarde!
Grupo 22/09/2017	No caso seria se eu estivesse usando tb e tb, isso? Se for isso eu criei o tb e tb1 pra ficar diferentes Pode ver isso mais tarde, de boas kkkkkkkkkkkk

Professor 22/09/2017	Oi: você criou tb e tb1, mas colocou no tabuleiro duas vezes tb. Acha essas linhas!
	tabuleiro.addPeca(0, 0, tb);
	tabuleiro.addPeca(7, 0, tb);
	[]s

Fonte: Autoria própria

Este outro grupo também gastou um bom tempo fazendo com que os movimentos das peças estivessem de acordo, mas o trecho acima mostra um problema quanto ao uso de objetos. Estava claro para o grupo que existia uma diferença entre os dois objetos em questão, existia o domínio de que era necessário criar um objeto diferente para cada torre, mas o problema ocorreu na hora de associar os objetos (tb, tb1) a outro (tabuleiro).

Os questionamentos feitos pelos dois grupos eram problemas da “lógica” do algoritmo. A sintaxe do programa estava correta (o compilador/interpretador reconhece como um programa válido), mas o algoritmo não fazia aquilo que quem escreveu o código queria que fizesse.

Nos dois casos, as intervenções do professor são muito similares aquelas feitas sobre problemas de sintaxe: Ajudar os e as estudantes a entenderem o que acontece, o porquê acontece e fornecer formas de resolver o problema. Ou seja, deve atuar na Zona de Desenvolvimento Iminente, organizando o processo a fim de auxiliar as pessoas envolvidas a desenvolver os processos psicológicos associados às tarefas executadas.

Apenas a título de mostrar a diversidade da turma, temos um terceiro grupo que teve dificuldades com questões anteriores à lógica de programação:

Tabela 24 – Textos da troca de e-mails com um terceiro grupo na atividade Mini Chess

Remetente	**texto**
Grupo 20/09/2017	Professor como que eu faço pra abrir o arquivo program pra fazer o exercício do xadrez no meu pc?
Professor 20/09/2017	Desculpe a demora -, mas cheguei agora em casa. Vai nesta página (https://processing.org/download/), baixe o processing que seja compatível com o seus sistema operacional e é só abrir, como é feito nas aulas. Se tiver algum problema, me avise! [ ]s
Grupo 20/09/2017	Professor não abre o site no meu computador
Professor 20/09/2017	Tenta com: https://processing.org/ Se não der, testa: https://23.239.19.146/ O segundo, o chrome vai dizer que não é seguro...mas ta certo. [ ]s

Fonte: Autoria própria

O grupo da Tabela 24 foi um que começou relativamente tarde a atividade, deixando para fazer o processo inicial (baixar o programa adequado) para o final. Na troca de e-mails, por um motivo desconhecido, a página oficial do processing não estava disponível. Por isso a recomendação de acesso usando o endereço de ip.

Como esta atividade foi feita desta maneira, somente uma vez, é difícil apontar causas de alguns problemas, com destaque em especial a alta quantidade de abstenções. Existem diversas questões relevantes fora do escopo desta discussão, tais como a quantidade de disciplinas simultâneas que as e os estudantes tinham (diferente de um curso superior, em um curso técnico integrado as matrículas são obrigatórias de acordo com o ano) e, por consequência, a grande quantidade de trabalhos e atividades.

Em 2017, o segundo ano do Ensino Médio Técnico em Informática tinha 14 disciplinas regulares: Banco de Dados, Sociologia, Matemática, Português, Química, Educação Física, História, Biologia, Inglês, Filosofia, Geografia, Espanhol, Programação Orientada a Objetos e Física. A instituição trabalhava com avaliações bimestrais, o que implica que em certos períodos do ano, existe uma pressão maior sobre os e as estudantes com relação ao seu desempenho escolar.

As aulas de Orientação a Objetos eram ofertadas nas sextas e o ano de 2017 teve diversos feriados que caíram na quinta ou na sexta. Comparada com a turma do terceiro ano em

que fui responsável por três disciplinas ligadas à informática/computação, minha única disciplina para o segundo ano era a de Programação Orientada a Objetos.

Estas duas questões são importantes para entender a influência da problemática estrutural dentro da sala de aula. Primeiro, o grande volume de disciplinas com avaliações feitas a cada final de bimestre. Para tentar evitar contribuir com esta questão, o ano da disciplina de Programação Orientada a Objetos foi planejado para que os trabalhos fossem entregues de forma contínua e a avaliação era composta pelos trabalhos do bimestre. Entretanto, isto não contribui para aliviar a quantidade de conteúdo que o corpo discente deveria lidar. Segundo, o pouco contato com a turma também não ajuda em termos gerais.

Por exemplo, na turma do terceiro ano, às vezes era possível tirar dúvidas de uma disciplina como Redes de Computadores no horário da disciplina de Banco de Dados. Mesmo não sendo ideal, o contato maior era importante para que dúvidas não ficassem muito tempo sem serem abordadas. Isto leva a entender que a troca de e-mails não deve ser considerada como substituta a qualquer outro tipo de abordagem somente pelo que estamos apresentando.

Outra questão é a organização do próprio currículo do curso, cuja ementa da disciplina introdutória de OO é praticamente igual a uma de Ensino Superior, como mostra a Tabela 25[34], além das estratégias do corpo discente para lidar com tudo isso.

Tabela 25 – Ementas das disciplinas de Programação Orientada a Objeto dos dois contextos educacinais

Curso	Ementa
Ensino Superior	Conceitos de orientação a objetos. Decomposição de programas. Generalização e especialização. Agregação e composição. Herança e polimorfismo. Projeto orientado a objetos. Estudo de uma linguagem.
Ensino Médio	Histórico e Paradigmas de Programação. Conceitos básicos de abstração e orientação a objetos (Objetos, classes, atributos e métodos). Herança e polimorfismo. Classes abstratas. Diagrama de classes. Conceito de exceções. A linguagem Java. Integração Java e Banco de Dados. Temas transversais conforme Res. CNE/CEB nº 02/2012. Exibição de filmes de produção nacional no contexto da disciplina conforme § 8º do artigo 26 da Lei nº 9.394/1996.

Fonte: Autoria própria

Mesmo que ementas não definam o que ocorre em sala de aula e sejam sensíveis ao corpo docente que a escreveu no projeto do curso, é no mínimo curioso notar que a ementa de Ensino Médio é muito maior e mais específica que a de uma disciplina equivalente de Ensino Superior. Além disso, se as matérias seguintes forem planejadas levando em conta que os

[34] O texto completo dos componentes curriculares encontra-se nos anexos.

conceitos presentes nestas ementas foram vencidos, haverá sempre um descompasso entre a as pessoas, a estrutura e o que é feito em sala de aula.

Nos dois primeiros grupos, as trocas de e-mails mostradas foram sempre com um integrante do grupo somente que tomou a frente no trabalho, enquanto em outras disciplinas podia ocorrer o inverso[35].

É possível questionar se esta é a atividade mais adequada para os conceitos de polimorfismo e herança, pois a dificuldade não se encontra na aplicação dos conceitos em si, mas em outras questões. De um ponto de vista extremamente idealista sobre currículo, dado que quem cursa OO já fez a disciplina de fundamentos de programação, o uso de condicionais e laços de repetição deveria ser algo resolvido. Ou seja, deveríamos esperar menos dificuldade com laços de repetição do que com a criação de classes e objetos[36].

Entretanto, esta atividade mostra que não necessariamente isso que ocorre. Primeiro, mesmo que seja importante dominar as estruturas básicas típicas de programação, elas não são pré-requisitos para entender os conceitos ligados à orientação a objetos. Mas, existe uma forte relação entre as estruturas básicas e a orientação a objetos, o que faz ignorar por completo sua influência algo complicado.

Por exemplo, os trechos mostrados permitem ver uma relação complexa entre laços e classes, sem uma linearidade específica. É possível entender claramente o processo de herança sem dominar por completo o uso de lógica de programação para manipular arrays e vice-versa.

4.7 CONSIDERAÇÕES GERAIS

Neste capítulo, foram apresentadas três atividades para discutir o ensino e aprendizagem de programação de computadores. De forma geral, apontamos quatro problemáticas que o conjunto ajuda a ilustrar. Primeiro, sobre pensamento computacional. Estes exemplos ilustram a questão que Tedre e Denning (2016, p.125) colocam sobre como o pensamento computacional seria uma habilidade e, como tal, "adquirida com o tempo através da prática – não um conhecimento sobre fatos ou informação"[37] (TEDRE; DENNING, 2016, p.125) contribuindo para

[35] Todas estas são percepções sem um estudo concreto com base em conversas informais com os estudantes.

[36] Apenas a título de complementação, o mesmo ocorreu com alguns grupos na atividade aplicada para o Ensino Superior.

[37] *A skill is an ability acquired over time with practice— not knowledge of facts or information.*

o entendimento de que "dominar um conjunto de conhecimentos de um campo específico não necessariamente dá a habilidade de atuar bem no campo[38] (TEDRE; DENNING, 2016, p.126).

As trocas de e-mail sobre a atividade "Mini chess" são um bom ponto de partida para este tópico. Nestas trocas, é possível identificar que os estudantes não têm dificuldades de entender os conceitos de Herança e Polimorfismo. Eles precisaram recorrer à ajuda em outras questões, como uso de laços para percorrer uma matriz ou depurrar comportamentos não esperados de seus códigos. É importante entender que são dúvidas sobre o efetivo uso destes conceitos em um programa de computador. Do nosso ponto de vista, fazem parte da prática e por consequência devem ser levadas em conta no processo de ensino e aprendizagem destes conceitos ligados a Ciência da Computação.

A segunda problemática é, de certa maneira, um desdobramento da primeira: o peso e função das instituições no processo de ensino e aprendizagem de programação de computadores. Nos dois contextos, entendemos que a estrutura organizacional dos cursos privilegia a visão de que os conceitos de Computação, para usar as palavras de Tedre e Denning (2016), são sobre fatos ou informação. A dimensão mais clara disto é a separação comum entre uma disciplina de algoritmos, fundamentos de programação e orientação a objetos. Mesmo que tacitamente seja reconhecida a continuidade entre os conceitos e práticas destas disciplinas, esta divisão tem consequências e influenciam as práticas de sala de aula.

Todas as atividades descritas foram feitas em cursos voltados à formação profissional e, inclusive, no caso das disciplinas de Programação Orientada a Objetos, o domínio deste "paradigma" é quase um pré-requisito para o trabalho como desenvolvedor ou cargos afins. Os exemplos apresentados são formas de tentar lidar com estes contextos e, nestes casos, as amarras institucionais apresentaram limitadores importantes, principalmente em um trabalho que consiga levar em conta a subjetividade e o contexto concreto dos participantes. No caso da Escola, este problema é mais acentuado, pois diferente da Faculdade, a formação em Computação era compulsória junto com a formação geral. Para agravar, a decisão de estudar em um curso técnico integrado muitas vezes é tomada pelo responsável.

Nisto temos uma espécie de inconsistência estrutural: Se os conceitos sobre computação são conhecimentos sobre fatos ou informação, então podemos tratá-los como física ou química, disciplinas que são introduzidas mais tarde na formação básica, ou seja, pensamento computacional, nesta perspectiva, é diferente de programação de computadores, uma vez que é

[38] *The realization that mastering a domain's body of knowledge need not confer skill at performing well in the domain.*

claro que programar um computador é uma habilidade. Mas o que fazer então se o "mercado"[39] demanda conhecimento em programação? E, mais curioso: Por que muitas das iniciativas que se filiam a esta visão usam a programação para articular estes conceitos que não deveriam precisar de programação?

Entretanto, se os conceitos ligados à Computação forem habilidades que precisam de tempo e prática para desenvolvimento, talvez seja necessário tratá-las como português e matemática, disciplinas que têm uma presença muito mais constante na trajetória escolar de um indivíduo no processo de escolarização. Deste lado, o problema está no fato de que este "tempo" de amadurecimento não existiu dentro da estrutura escolar para estas pessoas que chegam ao curso técnico no Ensino Médio ou faculdade. Ou seja, como ensinar e aprender algo que demanda tempo e prática quando não se teve este tempo e prática? Além disso, como dissemos no início deste trabalho, entre estes estudantes, temos vários que por diversos motivos têm algum tipo de contato com computação.

Podemos resumir estas problemáticas na questão de que a Ciência da Computação não parece ter clareza sobre o que é formar um cientista da computação (ou alguém considerado como parte da comunidade da Ciência da Computação de forma geral) e informar pessoas sobre conceitos ligados a suas práticas. Apenas a título de analogia, parece claro que no Ensino Médio, a Biologia ou a Química não estão preocupadas em ensinar as pessoas a pensar como um/a Biólogo/a um/a Químico/a, mas apresentar seus conceitos, modelos e teorias que julgam serem importantes para a formação de um indivíduo.

Entendemos que as duas perpectivas fazem parte da Computação. Existem conceitos que podem ser tratados como fatos e informação e existem práticas que demandam tempo e habilidades. Estas duas dimensões coexistem e não deveriam ser tratadas de forma separada. Pois com o aporte da teoria de Vigotski, a Computação é parte de uma cultura e, como tal, tem práticas e conceitos que foram desenvolvidos ao longo da história. A programação de computadores e suas variações é uma destas práticas, e coisas como Programação Orientada a Objeto são conhecimentos desenvolvidos sobre tais práticas.

Como comentado anteriormente, a questão estrutural dos cursos também tem influência direta sobre as práticas de sala de aula. A organização em disciplinas acaba por criar divisões artificiais sobre o continuum de conhecimentos ligados à Computação e restringe o espaço de

[39] Pessoalmente não acho que esta entidade abstrata e presente mais como uma retórica dentro da escola deva ditar os processos escolares, mas como ignorar esta questão em um curso cujo o objetivo seria disponibilizar mais rápidamente mão de obra para o "mercado"?

atividades que podem ser exploradas, deixando a cargo da pessoa no papel de professor fazer "desvios" do que é a ementa de uma disciplina. Pior, se a ou o estudante tem necessidade ou vontade de explorar algo diferente, não existe nada na estrutura que leve isso em conta.

Por exemplo, em um outro trabalho que usava um jogo de tabuleiro na mesma turma do "Minichess", foi proposto para duas estudantes trabalhar com os desenhos do tabuleiro. Esta não é uma atividade diretamente ligada aos conceitos de Orientação a Objetos, mas é um tipo de trabalho típico de desenvolvimento de *software*. Além disso, de forma indireta trabalha com conceitos ligados a classes, objetos, herança e polimorfismo, uma vez que, da mesma maneira que os laços de repetição em matriz eram necessários para implementar de fato a herança e polimorfismo, as imagens são uma das partes constitutivas das classes e objetos. Conseguimos fazer as etapas iniciais (estudo de referências, esboços e entendimento de como funciona a imagem no computador), mas devido ao volume de trabalho e outras questões circunstanciais, não pude dar a atenção devida às estudantes para efetivar o processo.

Um segundo ponto sobre as instituições é a própria organização da sala de aula, em que uma pessoa é responsável pela mediação de diversas que, em teoria, têm um mesmo nível de conhecimento sobre o tema da máteria. Esta organização é uma muito diferente da típica no desenvolvimento de *software*, em que existem diversas pessoas com níveis diferentes atuando juntas.

Era relativamente comum que os grupos de estudantes tivessem alguém mais interessado que tomava a frente nas discussões. A curto prazo, isso gera preocupações sobre o quanto os demais estudantes do grupo estão a participar efetivamente da atividade. Entretanto, esta é uma forma de mediação válida quanto ao uso da Zona de Desenvolvimento Iminente e por isso não deveria ser descartada tão facilmente. O problema, de um ponto de vista educacional, está na possível falta de preparo didático do colega mais experiente ou capacitado (FINO, 2001, p.9).

Uma tentativa de lidar com isto foi criar atividades com diversos tipos de tarefas diferentes. Estas atividades partiam de uma versão "alpha" de um programa escrito especificamente para a atividade. As tarefas são separadas em fáceis, médias e difíceis. As fáceis são mudanças simples e diretas tais como mudar a forma como uma informação é apresentada ou adicionar algo novo, mas que obedeça a um padrão existente no *software*. As médias consistiam em tarefas mais complicadas, que podem ser consideradas integrações de 3 a cinco tarefas fáceis. Por fim, as difíceis, era dado somente o objetivo da tarefa, cabendo a quem implementar toda a decisão do como fazer.

Com isso, cada equipe deveria ficar responsável por uma quantidade de pontos. Desta maneira, uma equipe com um integrante mais capacitado poderia resolver tarefas médias e ou difíceis, enquanto os menos experientes, as mais fáceis. A quantidade de pontos exigida é defina de uma forma que não seja possível entregar o trabalho somente resolvendo questões fáceis, ou só difíceis.

Em terceiro lugar, nas propostas do Círculo de Bakhtin, a noção de tema e de significação parecem elucidar certas práticas descritas nas atividades. Por significação, Volochinov (2019 [1926], p.228) entende como "aqueles aspectos do enunciado que são repetíveis e idênticos a si mesmos em todas as ocorrências" (VOLOCHINOV, 2019 [1926], p.228). Enquanto tema, seria "definido não apenas pelas formas linguísticas que o constituem – palavras, formas morfológicas e sintáticas, sons, entonação – mas também pelos aspectos extraverbais da situação" (VOLOCHINOV, 2019 [1926], p.228). O tema está diretamente ligado ao enunciado e é "individual e irrepetível como o próprio enunciado" (VOLOCHINOV, 2019 [1926], p.228).

Nas sitações descritas, podemos considerar os conceitos da Ciência da Computação como algo similar à significação. Objetos, classes, variáveis, métodos são os aspectos idênticos e repetíveis em atividades ligadas ao ensino e aprendizagem de linguagens de programação. Para Volochinov (2019 [1926], p.229), o "tema deve apoiar-se em alguma significação estável, caso contrário ele perderá a sua conexão com aquilo que veio antes e que veio depois, ou seja, perderá totalmente seu sentido".

Já os "desvios" feitos nas atividades resultantes das interações entre as pessoas envolvidas são parte da dimensão individual e irrepetível dos enunciados. Isto envolve aquelas questões que não estão necessariamente visíveis nos documentos gerados nas atividades a fim de suprir demandas institucionais. Por exemplo, as trocas de e-mail sobre laços de repetição em exercícios de herança ou as escolhas feitas pelos grupos em como implementar as regras de um super trunfo.

Assim, o tema seria uma manifestação concreta, uma forma de acesso indireto daquilo que Vigotski chamou de Zona de Desenvolvimento Iminente. Por exemplo, na Tabela 22, como já afirmamos, mostra que os conceitos mobilizados não foram somente aqueles planejados, mas outros que, de um ponto de vista reducionista, deveriam ter sido incorporados anteriormente. De certa maneira, se olhar a atividade sem as entonações é possível entender que o seu conteúdo é o que está no planejamento: Herança e Polimorfismo. Entretanto, as trocas de e-mail mostram que além destes conceitos, cada estudante também articula outros. Os casos apresentados na Tabela 22 e na 23 mostram que estas atividades devem ser entendidas em sua dimensão enunciativa,

pois do contrário, elementos importantes quanto ao processo de ensino e aprendizagem podem escapar.

De maneira prática, entender que o tema só existe na concretude da enunciação faz com que estas atividades devam ser planejadas levando em conta tais características. Devem ser pensadas para que possibilitem tais "desvios" que, dado o que foi apresentado aqui, parecem ser tão fundamentais para o ensino e aprendizagem quanto os caminhos originais desenhados. Esta seria uma maneira de levar em conta as pessoas envolvidas no processo.

Quanto ao entendimento do processos de ensino e aprendizagem de programação de computadores através do dialogismo, gostaria de argumentar quanto à existência de práticas que podem ser mais dialógicas ou mais monológicas. Ambas as perspectivas trabalham com as questões levantadas por Faraco (2013) (perspectiva avaliativa entre sujeito e mundo, relação entre eu/outro e unicidade de eventos do mundo da vida), mas de formas diferentes.

A título de reforçar questões que o dialogismo permite observar, vamos comentar de forma genérica e grosseira sobre como uma visão mais monológica entenderia as questões apresentas na tese. É importante salientar que entendemos que a realidade é mais complexa e que tanto os exemplo mais dialógicos quanto os mais monológicos se misturam e, até mesmo, podem ser contraditórios. Também não custa reforçar que as práticas apresentadas neste texto são tentativas de se aproximar de uma perpectiva mais dialógica e não necessariamente defendemos que elas sejam totalmente dialógicas em um sentido contrário as práticas monológicas.

As práticas mais monológicas entendem a linguagem como um sistema de signos que basta o entendimento de suas regras para que exista um domínio. Esta perspectiva é muito similar ao modelo bancário de Freire (2014), em especial, quanto à metáfora do depósito. Um planejamento que acredite que apenas aulas expositivas sobre sintaxe ou regras de uma linguagem de programação são o suficiente, ou de forma mais geral, as práticas que têm seu foco na dimensão do ensino e deixam de lado a aprendizagem têm tendências mais monologizantes.

Do ponto de vista dialógico, a linguagem é uma atividade, logo necessita de interação. De maneira similar, seu ensino e aprendizagem também devem levar em conta tais características. Ou seja, não basta apresentar o conteúdo, mas um código deve ser escrito, julgado, defendido e mudado. Nestas trocas e manipulações que se daria o processo de ensino e aprendizagem. É importante entender que em ambos os casos, a heterodiscursividade existe, o que difere é a forma como as "vozes" dentro do processo interagem entre si.

Quanto às "vozes", ou seja, a relação entre eu e o outro no processo de ensino e aprendizagem de programação de computadores, a prática monologizante é direcionada a ignorar ou subestimar o outro. Consideramos monologizantes as práticas que têm seu foco no objeto de estudo (linguagem de programação) e deixam em segundo plano as e os estudantes. Por exemplo, provas ou avaliações independentes de contexto ou pior, avaliações que têm como objetivo somente minimizar o trabalho docente[40]. Também é monologizante o uso de um perfil de estudante ideal como base para estabelecer a dificuldade do conteúdo que mais diz sobre as espectativas de quem o estabelece do que representa efetivamente aqueles que frequentam as disciplinas.

Em uma perspectiva mais dialógica, estudantes são pessoas com trajetórias de vida e espectativas variadas. Com isso, é preciso estabelecer formatos que estudantes sejam levados em conta como sujeitos ao mesmo tempo que o discurso base da Computação também faça parte da discussão. O dialogismo, ao reconhecer a importância do encontro destes discursos como parte central do processo, permite propor práticas que sejam menos monológicas, como dar liberdade para os ou as estudantes escolherem temas de trabalhos ou propor exercícios com diversos graus de dificuldade. Nas atividades apresentadas, foi trabalhada uma correção personalizada de códigos escritos junto com discussões sobre a sua forma e estilo, como uma tentativa de se aproximar de algo mais dialógico.

Sobre o não-alibi, o monologismo tende a ter um álibi para com aquilo que não consegue dar conta. Em disciplinas de Computação, muitas vezes equipamentos são importantes para que alguns conceitos tornem-se concretos para os e as estudantes. Não ter equipamentos ou condições, muitas vezes, são usados como álibi para problemas de contexto. Mais grave quando são apresentadas condições adversas, em especial com relação a estudantes que por algum motivo precisam de algum tratamento especial, a falta de experiência ou questões burocraticas legais são usadas como forma de justificar uma inanição por parte de docentes[41].

Quanto a entender o processo de abstração como medição concreta, as atividades apresentadas mostram que existe uma variedade de níveis, formas e possibilidades. No caso das cartas ela é feita através de objetos, no caso do jogo de xadrez através de uma interface gráfica e no caso dos filmes através do próprio conteúdo comum. Entratanto, independente do meio usado

[40] Entendo que muitas vezes estas são estratégias de sobrevivência de docentes e que, dado certos contextos instuticionais, é o que mantém viável o trabalho. Mas, ainda assim, são práticas que ignoram o outro (estudantes), mesmo que sejam justificadas e/ou necessárias.

[41] Novamente, o simplismo que tratamos esta questão é evidente. Nem sempre as instituções possibilitam que estas questões sejam adereçadas pelos docentes ou muitas vezes são morosas quanto ao trato de exceções.

na atividade, as trocas de e-mail deixam claro que o reconhecimento das pessoas como parte do processo é indispensável para uma mediação efetivamente concreta.

Além disso, é importante colocar mais uma vez que estas atividades têm limitações quanto a conseguir lidar com os interesses ou demandas diretas das e dos estudantes. Mas, no planejamento da disciplina, elas foram importantes para preparar um terreno para atividades mais abertas, no sentido de que a concretude inicial fosse algo proposto pelos próprios estudantes.

Na disciplina de Orientação a Objetos, na Faculdade, o último trabalho era desenvolver um sistema ou uma parte usando conceitos da disciplina. Com um tema em aberto, as estratégias de escolha dos grupos foi bem variada. Desde grupos usando um mesmo trabalho feito para outra disciplina para "ganhar tempo", estudantes implementando jogos virtuais ou simulações de partes de jogos de tabuleiro, sistemas de gerenciamento comercial e até propostas que usavam outras linguagens de programação que eram de interesse dos e das estudantes.

Por fim, todas estas problemáticas mostram o quanto é importante levar em conta a crítica feita por Vieira Pinto (2005a) sobre a necessidade de contextualização da tecnologia. Acreditar que conceitos da Ciência da Computação devem ser ensinados para todos e todas, logo deveriam fazer parte do ensino básico, ganha novas dimensões quando se leva em conta o contexto concreto. No caso deste trabalho, isto fica evidente nas diversas peculiaridades encontradas no curso de Ensino Médio, como o inchaço da grade horária causado pelas disciplinas técnicas ou no peso e obrigatoriedade das disciplinas ligadas a Computação dentro do ensino básico. Este formato é algo muito diferente de outras formas mais pontuais, tais como *workshops* ou atividades eletivas.

5 EPÍLOGO

O final desta tese foi escrita quase dois anos após a última vez que entrei em uma sala de aula com a responsabilidade de ser o professor. Decidi por pausar minha carreira docente alguns meses antes da pandemia do coronavírus se espalhar pelo mundo.

Voltei a tentar escrevê-la somente meses depois da pandemia ter avançado no Brasil, quando percebi que não era algo que iria "passar" em um ou dois meses. Uma vez que não atuei mais como docente, não tive que experienciar a escola e a universidade em tempos de pandemia, pelo menos no papel de professor.

De certa maneira, a troca de e-mails parece ser uma forma de trabalhar compatível com as exigências impostas pela pandemia às escolas e às universidades. Permite distanciamento social, além de manter algumas características da interação que ocorre em sala de aula. Mas, como colocado ao longo do texto, as trocas de e-mails são uma parte de um processo maior de interação. São a parte do trabalho que deixou rastros e esses, por sua vez, permitem a aplicação dos processos característicos de pesquisa e análise monográfica/acadêmica.

Acredito que as bases do processo descrito nesta tese estão interditados, pelo menos provisoriamente. Mesmo sabendo que o processo dialógico vai além de uma resposta face a face, o apagamento do encontro em sala de aula (um mesmo espaço físico) torna muito difícil um processo mais dialógico, principalmente com a perda das entonações mais concretas. O ambiente virtual tem suas entonações, mas a discussões feitas aqui têm uma dependência grande do contexto concreto da sala de aula. Enfim, o uso do e-mail *per se* como parte do processo não é o suficiente para dar destaque a dimensão dialógia do ensino e aprendizagem de programação de computadores. Naquilo que foi apresentado neste trabalho, se retirada a base epstemológica voltada para uma perspectiva materilialista, resta apenas uma visão mecanicista do processo dialógico. E, como bem exemplifica Amorim (2001, p.142), "no modelo mecanicista, a linguagem dos engenheiros se faz presente e a troca verbal assemelha-se ao trabalho dos telégrafos". Esperamos ter dado indícios de que somente a mecânica da troca não parece ser o suficiente para lidar com ensino e aprendizagem de programação de computadores.

Quanto a três perguntas do início do texto, esperamos ter apresentado o papel que as estruturas tanto intitucionais quanto conceituais têm no processo de ensino e aprendizagem formal de programação de computadores. Sobre a primeira questão, "Como lidar com os diferentes perfis de estudantes no ensino de programação dentro de uma estrutura como uma disciplina ?",

a forma de trabalho apresentada deposita muito do trabalho na figura do professor. Quanto maior a quantidade de estudantes, mais vezes o papel de mediador ou de locutor no diálogo é acionado. Este processo personalista parece ter seus frutos, mas a longo prazo, tem um desgaste que não pode ser ignorado.

Com relação à segunda questão, “Como lidar com a assimetria de conhecimentos sobre Computação ?”, entender o processo de ensino e aprendizagem como dialógico e que uma disciplina não deve fazer com que todo que saiam dela tenham um certo nível de conhecimento sobre seu conteúdo, mas auxiliar as pessoas em atingir novos “lugares” de conhecimento em que possam articular com mais facilidade os conhecimentos da disciplina, parece ser um bom encaminhamento. As atividades apresentadas tem em si características que propiciam este entendimento.

Enfim, se “É possível pensar em uma forma de ensino de programação que dê conta destas duas questões?”, acreditamos que esta tese permite responder que sim. A forma proposta é específica dos contextos apresentados e dependente das pessoas envolvidas, mas mostra que é possível pensar uma forma de ensino de programação que não tenha foco em apenas medir ou laurear quem deve seguir a diante na Informática ou Ciência da Computação.

REFERÊNCIAS

ABRAMS, Brad. **History around Pascal Casing and Camel Casing**. 2004. Disponível em: https://blogs.msdn.microsoft.com/brada/2004/02/03/history-around-pascal-casing-and-camel-casing/. Acesso em: 15 jul. 2018.

AGUIAR, Janderson. Experiência baseada em gamificação no ensino sobre herança em programação orientada a objetos. *In*: . Sociedade Brasileira de Computação – SBC, 2015. Disponível em: https://doi.org/10.5753/cbie.wcbie.2015.1444.

AGUIAR, Ygor Quadros de; CECOTTI, Letícia; CAETANO, Lucas; MARCOS, Pedro De Botelho; AZAMBUJA, Jose Rodrigo. Pré-comp: introduzindo os fundamentos da computação e contribuindo com a motivação e aproveitamento acadêmico. *In*: . Sociedade Brasileira de Computação – SBC, 2015. Disponível em: https://doi.org/10.5753/cbie.wcbie.2015.1340.

ALTHUSSER, L. **Iniciação À Filosofia Para Os Não-filósofos**. *[S.l.]*: WMF MARTINS FONTES, 2019. ISBN 9788546901647.

AMARAL, Marília Abrahão; BIM, Sílvia Amélia; BOSCARIOLI, Clodis; MACIEL, Cristiano. Introducing computer science to brazilian girls in elementary school through HCI concepts. *In*: **Design, User Experience, and Usability: Users and Interactions**. Springer International Publishing, 2015. p. 141–152. Disponível em: https://doi.org/10.1007/978-3-319-20898-5_14.

AMARAL, Marília Abrahão; EMER, Maria Claudia Figueiredo Pereira; BIM, Silvia Amélia; SETTI, Mariangela Gomes; GONÇALVES, Marcelo Mikosz. Investigando questões de gênero em um curso da área de computação. **Revista Estudos Feministas**, FapUNIFESP (SciELO), v. 25, n. 2, p. 857–874, ago. 2017. Disponível em: https://doi.org/10.1590/1806-9584.2017v25n2p857.

AMORIM, M. **O pesquisador e seu outro: Bakhtin nas ciências humanas**. *[S.l.]*: Musa, 2001. (Cultura educação letras e lingüística). ISBN 9788585653590.

ANDRADE, Raul; MENDONçA, Jonas; OLIVEIRA, Wesley; ARAUJO, Ana Liz; SOUZA, Flávia. Uma proposta de oficina de desenvolvimento de jogos digitais para ensino de programação. *In*: . Sociedade Brasileira de Computação – SBC, 2016. Disponível em: https://doi.org/10.5753/cbie.wcbie.2016.1127.

ANDRIOLA, Wagner Bandeira; CAVALCANTE, Luanna Rodrigues. Avaliação do raciocínio abstrato em estudantes do ensino médio. **Estudos de Psicologia (Natal)**, scielo, v. 4, p. 23 – 37, 6 1999.

ANGWIN, Laura Castaneda Julia. The digital divide / high-tech boom a bust for blacks, latinos. 1998. Disponível em: https://www.sfgate.com/news/article/The-Digital-Divide-High-tech-boom-a-bust-for-3007911.php.

ANJOS JR., José; SANTIAGO, Levy; VIANA, Hellan; ABIJAUDE, Jauberth; SOBREIRA, Pericles. Avaliação de linguagens visuais de programação no ensino médio a partir da utilização do conceito de robótica pedagógica. *In*: . Brazilian Computer Society (Sociedade Brasileira de Computação – SBC), 2017. Disponível em: https://doi.org/10.5753/cbie.wcbie.2017.962.

ARANTES, Flávia; FERREIRA, José Michael Leandro da Silva. Uma dinâmica para ensino de conceitos fundamentais de programação. *In*: . Sociedade Brasileira de Computação – SBC, 2015. Disponível em: https://doi.org/10.5753/cbie.wcbie.2015.1218.

ARAUJO, Ana Liz; ANDRADE, Wilkerson; GUERRERO, Dalton. Um mapeamento sistemático sobre a avaliação do pensamento computacional no brasil. *In*: . Sociedade Brasileira de Computação – SBC, 2016. Disponível em: https://doi.org/10.5753/cbie.wcbie.2016.1147.

ARAUJO, Ana Liz; ANDRADE, Wilkerson; SEREY, Dalton. Pensamento computacional sob a visão dos profissionais da computação: uma discussão sobre conceitos e habilidades. *In*: . Sociedade Brasileira de Computação – SBC, 2015. Disponível em: https://doi.org/10.5753/cbie.wcbie.2015.1454.

ASPRAY, William. **Participation in Computing: The National Science Foundation's Expansionary Programs**. 1st. ed. *[S.l.]*: Springer Publishing Company, Incorporated, 2016. ISBN 3319248308.

AURELIANO, Viviane Cristina Oliveira; TEDESCO, Patrícia Cabral de Azevedo Restelli; GIRAFFA, Lúcia Maria Martins. Desafios e oportunidades aos processos de ensino e de aprendizagem de programação para iniciantes. *In*: . *[S.l.: s.n.]*, 2016.

AVILA, Christiano; CAVALHEIRO, Simone. Robótica educacional como estratégia de promoção do pensamento computacional – uma proposta de metodologia baseada em taxonomias de aprendizagem. *In*: . Brazilian Computer Society (Sociedade Brasileira de Computação – SBC), 2017. Disponível em: https://doi.org/10.5753/cbie.wcbie.2017.1192.

BACHELARD, G. **A epistemologia**. *[S.l.]*: Edições 70, 2001. ISBN 9789724402321.

BACHELARD, Gaston. **A Formação do Espírito Científico**. *[S.l.: s.n.]*, 2016.

BAKHTIN, Mikhail. O discurso no romance. *In*: **Teoria do Romance I: A estilística**. *[S.l.]*: Editora 34, [1930] 2015. p. 19–242.

BAKHTIN, M. **Estética Da Criação Verbal**. *[S.l.]*: WMF MARTINS FONTES, [1969] 2011. ISBN 9788578274702.

BAKHTIN, Mikhail. **Marxismo e filosofia da linguagem: problemas fundamentais do método sociológico na ciência da linguagem**. 16. ed. *[S.l.]*: Hucitec, 2014. (Linguagem e Cultura).

BAKHTIN, Mikhail. O texto n alinguística, na filologia e outras ciências humanas. *In*: **Os Gêneros Do Discurso**. *[S.l.]*: EDITORA 34, 2016 [1924].

BAKHTIN, Mikhail. **Os Gêneros Do Discurso**. *[S.l.]*: EDITORA 34, 2016 [19XX].

BAKHTIN, M. **Notas Sobre Literatura, Cultura E Ciências Humanas**. *[S.l.]*: EDITORA 34, 2017. ISBN 9788573266665.

BAKHTIN, M. **Para uma filosofia do ato responsável**. *[S.l.]*: Pedro & João Ed., 2017 [1919/1921]. ISBN 9788579930096.

BAR-ILAN, Judit. Web of science with the conference proceedings citation indexes: the case of computer science. **Scientometrics**, Springer Science and Business Media LLC, v. 83, n. 3, p. 809–824, jan. 2010. Disponível em: https://doi.org/10.1007/s11192-009-0145-4.

BARCELOS, Thiago; BORTOLETTO, Rodrigo; ANDRIOLI, Mary. Formação online para o desenvolvimento do pensamento computacional em professores de matemática. *In*: . Sociedade Brasileira de Computação – SBC, 2016. Disponível em: https://doi.org/10.5753/cbie.wcbie.2016.1228.

BARCELOS, Thiago; MUñOZ, Roberto; ACEVEDO, Rodolfo Villarroel; SILVEIRA, Ismar Frango. Relações entre o pensamento computacional e a matemática: uma revisão sistemática da literatura. *In*: . Sociedade Brasileira de Computação – SBC, 2015. Disponível em: https://doi.org/10.5753/cbie.wcbie.2015.1369.

BARCELOS, Thiago; NASCIMENTO, Luiz Felipe Silva do; SOUZA, Alexandra; SILVA, Leandro; MUñOZ, Roberto. Análise automatizada do discurso de aprendizes de programação: relações entre emoções e nível de experiência. *In*: . Brazilian Computer Society (Sociedade Brasileira de Computação – SBC), 2017. Disponível em: https://doi.org/10.5753/cbie.wcbie.2017.1202.

BARCELOS, Thiago; SOUZA, Alexandra; SILVA, Leandro; MUñOZ, Roberto; ACEVEDO, Rodolfo Villarroel. Mensurando o desenvolvimento do pensamento computacional por meio de mapas auto-organizáveis: um estudo preliminar em uma oficina de jogos digitais. *In*: . Brazilian Computer Society (Sociedade Brasileira de Computação – SBC), 2017. Disponível em: https://doi.org/10.5753/cbie.wcbie.2017.932.

BARROS, J.D.A. **O campo da história: Especialidades e abordagens**. *[S.l.]*: Editora Vozes, 2012. ISBN 9788532643179.

BATHKE, Julia; RAABE, André. Pensamento computacional na educação de jovens e adultos: Lições aprendidas. *In*: . Sociedade Brasileira de Computação – SBC, 2016. Disponível em: https://doi.org/10.5753/cbie.wcbie.2016.1087.

BAUER, Rudieri; FLORES, Gian Luca Motta; CRESTANI, Angelo V.; MOMBACH, Jaline. Projeto codific@r: Oficinas de programação em dispositivos móveis no ensino fundamental. *In*: . Brazilian Computer Society (Sociedade Brasileira de Computação – SBC), 2017. Disponível em: https://doi.org/10.5753/cbie.wcbie.2017.1210.

BAZZO, Walter Antonio. **Ciência, tecnologia e sociedade: e o contexto da educação tecnológica**. *[S.l.]*: Editora da UFSC, 2011. ISBN 9788532805508.

BEHESHTI, Mohsen; GUNAWARDENA, Ananda D. Teaching programming paradigms using a laboratory approach. **J. Comput. Sci. Coll.**, Consortium for Computing Sciences in Colleges, USA, v. 16, n. 3, p. 144–148, 3 2001. ISSN 1937-4771. Disponível em: http://dl.acm.org/citation.cfm?id=374685.374736.

BERARDI, Rita; KOZIEVITCH, Nádia; BIM, Silvia Amelia; AUCELI, Pedro. Oficina de banco de dados com aprendizado cinestésico para meninas do ensino médio. *In*: **Anais do XXV Workshop de Informática na Escola (WIE 2019)**. Sociedade Brasileira de Computação (SBC), 2019. Disponível em: https://doi.org/10.5753/cbie.wie.2019.345.

BEZERRA, Paulo. Breve glossário de alguns conceitos-chave. *In*: **Teoria do Romance I: A estilística**. *[S.l.]*: Editora 34, 2015. p. 243–250.

BIJKER, W.E.; HUGHES, T.P.; PINCH, T.; DOUGLAS, D.G. **The Social Construction of Technological Systems: New Directions in the Sociology and History of Technology**. *[S.l.]*: MIT Press, 2012. (The MIT Press). ISBN 9780262300872.

BLAHA, M.; RUMBAUGH, J. **Modelagem e projetos baseados em objetos**. *[S.l.]*: Campus, 1994.

BLIKSTEIN, Paulo; WORSLEY, Marcelo; PIECH, Chris; SAHAMI, Mehran; COOPER, Steven; KOLLER, Daphne. Programming pluralism: Using learning analytics to detect patterns in the learning of computer programming. **Journal of the Learning Sciences**, Informa UK Limited, v. 23, n. 4, p. 561–599, out. 2014. Disponível em: https://doi.org/10.1080/10508406.2014.954750.

BLOCH, M. **Apologia da história: Ou o ofício de historiador**. *[S.l.]*: Zahar, 2002. ISBN 9788537805732.

BOOCH, Grady. **Oral History of Charles Simonyi**. Computer History Museum, 2008. Disponível em: http://archive.computerhistory.org/resources/access/text/2015/06/102702232-05-01-acc.pdf.

BORGES, Rosemary; MARQUES, Carla; LIMA, Rommel; SOUZA, Jorge Allende Bonamigo Maia de. Tutor inteligente para recomendação de atividades de programação em um ambiente virtual de aprendizagem. *In*: . Brazilian Computer Society (Sociedade Brasileira de Computação – SBC), 2017. Disponível em: https://doi.org/10.5753/cbie.wcbie.2017.922.

BOURDIEU, P.; CATANI, A.M. **Escritos de educação**. *[S.l.]*: Editora Vozes, 1999. (Ciências sociais da educação). ISBN 9788532620538.

BOURDIEU, P.; CHAMBOREDON, J.C.; PASSERON. **Ofício de Sociólogo: Metodologia da pesquisa na sociologia**. *[S.l.]*: Vozes, 2015.

BRACKMANN, Christian; BOUCINHA, Rafael Marimon; ROMAáN-GONZAáLEZ, Marcos; BARONE, Dante Augusto Couto; CASALI, Ana. Pensamento computacional desplugado: Ensino e avaliação na educação primária espanhola. *In*: . Brazilian Computer Society (Sociedade Brasileira de Computação – SBC), 2017. Disponível em: https://doi.org/10.5753/cbie.wcbie.2017.982.

BRAIT, B.; MELLO, R. Enunciado/enunciado concreto/enunciaćão. *In*: **Bakhtin: conceitos-chave**. *[S.l.]*: Contexto, 2005. p. 61–78. ISBN 9788572442909.

BRILLIANT, Susan S.; WISEMAN, Timothy R. The first programming paradigm and language dilemma. **SIGCSE Bull.**, ACM, New York, NY, USA, v. 28, n. 1, p. 338–342, 3 1996. ISSN 0097-8418. Disponível em: http://doi.acm.org/10.1145/236462.236572.

BRUN, Eric. Interdisciplinarity in french social sciences scientific journals. **Dados**, FapUNIFESP (SciELO), v. 60, n. 3, p. 867–894, 9 2017. Disponível em: https://doi.org/10.1590/001152582017137.

BUCCI, Paolo; LONG, Timothy J.; WEIDE, Bruce W. Do we really teach abstraction? *In*: **Proceedings of the Thirty-Second SIGCSE Technical Symposium on Computer Science Education**. New York, NY, USA: Association for Computing Machinery, 2001. (SIGCSE '01), p. 26–30. ISBN 1581133294. Disponível em: https://doi.org/10.1145/364447.364531.

BURKE, Peter. **A escrita da história**. *[S.l.]*: UNESP, 1992. ISBN 9788571390270.

BURKE, P. **O Renascimento**. *[S.l.]*: Texto & Grafia, 2008. ISBN 9789898285843.

CACACE, F.; CERI, S.; CRESPI-REGHIZZI, S.; TANCA, L.; ZICARI, R. Integrating object-oriented data modelling with a rule-based programming paradigm. **SIGMOD Rec.**, ACM, New York, NY, USA, v. 19, n. 2, p. 225–236, 5 1990. ISSN 0163-5808. Disponível em: http://doi.acm.org/10.1145/93605.98732.

CAVALCANTE, Ahemenson; COSTA, Leonardo Dos Santos; ARAUJO, Ana Liz. Um estudo de caso sobre competências do pensamento computacional desenvolvidas na programação em blocos no code.org. *In*: . Sociedade Brasileira de Computação – SBC, 2016. Disponível em: https://doi.org/10.5753/cbie.wcbie.2016.1117.

CERUZZI, P.E.; E, P.; CERUZZI, C.A.E.C.P.E.; MISA, T.J.; ASPRAY, W. **A History of Modern Computing**. *[S.l.]*: MIT Press, 2003. (History of computing). ISBN 9780262532037.

CODE.ORG. **President Obama asks America to learn computer science**. *[s.n.]*, 2013. Disponível em: https://www.youtube.com/watch?v=6XvmhE1J9PY.

CODE.ORG. **Code.org and Diversity in Computer Science**. *[s.n.]*, 2018. Disponível em: https://code.org/diversity.

COMMISSION, Broadband. **Connecting Africa Through Broadband**. UNESCO, 2019. Disponível em: https://www.broadbandcommission.org/Documents/working-groups/DigitalMoonshotforAfrica_Report.pdf.

COMPUTERSCIENCE.ORG. **The Current State of Women in Computer Science**. *[s.n.]*, 2018. Disponível em: https://www.computerscience.org/resources/women-in-computer-science/.

CORRÊA, Diogo; PETERS, Gabriel. Somos bourdieusianos? **Cadernos do Sociofilo**, v. 1, p. 1 – 5, 03 2011.

COSTA, Leonardo Dos Santos; CAVALCANTE, Ahemenson; ARAUJO, Ana Liz Souto O.; ANDRADE, Wilkerson; GUERRERO, Dalton. Um estudo exploratório da aplicação de pensamento computacional baseado nas perspectivas de professores do ensino médio. *In*: . Brazilian Computer Society (Sociedade Brasileira de Computação – SBC), 2017. Disponível em: https://doi.org/10.5753/cbie.wcbie.2017.992.

CRAIG, Michelle; PETERSEN, Andrew. Student difficulties with pointer concepts in c. *In*: . New York, NY, USA: ACM, 2016. (ACSW '16), p. 8:1–8:10. ISBN 978-1-4503-4042-7. Disponível em: http://doi.acm.org/10.1145/2843043.2843348.

CURRICULA, Association for Computing Machinery (ACM) Joint Task Force on Computing; SOCIETY, IEEE Computer. **Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science**. New York, NY, USA: Association for Computing Machinery, 2013. ISBN 9781450323093.

CURRICULA, CORPORATE The Joint Task Force on Computing. Computing curricula 2001. **J. Educ. Resour. Comput.**, Association for Computing Machinery, New York, NY, USA, v. 1, n. 3es, p. 1–es, set. 2001. ISSN 1531-4278. Disponível em: https://doi.org/10.1145/384274.384275.

DAGNINO, R. **Neutralidade da ciência e determinismo tecnológico: um debate sobre a tecnolciência**. *[S.l.]*: UNICAMP, 2008. ISBN 9788526807891.

DENZIN, N.K.; LINCOLN, Y.S.; NETZ, S.R. **O planejamento da pesquisa qualitativa: teorias e abordagens**. *[S.l.]*: Artmed, 2006. ISBN 9788536306636.

DESLAURIERS, Jean-Pierre; KÉRISIT, Michèle. Novas perspectivas do ensino de língua portuguesa:implicações para a alfabetização. *In*: POUPART, J.; DESLAURIERS, Jean-Pierre; GROULX, Lionel-H; LAPERRIÈRE; MAYER, Robert; ÁLVARO, Pires (Ed.). **A pesquisa qualitativa: enfoques epistemológicos e metodológicos**. *[S.l.]*: Vozes, 2008, (Coleção sociologia). ISBN 9788532636812.

DOSSE, François. **A história**. *[S.l.]*: UNESP, 2012.

DVORAK, Wolfgang; GOTTLOB, Georg; PICHLER, Reinhard; WOLTRAN, Stefan. Alternation as a programming paradigm. *In*: . New York, NY, USA: ACM, 2009. (PPDP '09), p. 61–72. ISBN 978-1-60558-568-0. Disponível em: http://doi.acm.org/10.1145/1599410.1599419.

ECO, U. **A busca da língua perfeita na cultura européia**. *[S.l.]*: EDUSC, 2001. (Coleção Signum). ISBN 9788574601090.

EVANS, Will; RANGARAJAN, Sinduja. Hidden figures: How silicon valley keeps diversity data secret. 2017. Disponível em: https://www.revealnews.org/article/hidden-figures-how-silicon-valley-keeps-diversity-data-secret/.

FARACO, Carlos Alberto. Autor e autoria. *In*: BRAIT, Beth (Ed.). **Bakhtin : outros conceitos-chave**. *[S.l.]*: Contexto, 2006.

FARACO, Carlos Alberto. Ensinar x não ensinar gramática: ainda cabe essa questão? **Calidoscópio**, v. 4, n. 1, p. 15–26, 2006.

FARACO, Carlos Alberto. **Linguagem e Diálogo – As ideias linguísticas do círculo de Bakhtin**. *[S.l.]*: PARABOLA, 2013.

FARACO, Carlos Alberto. Um pósfacio meio impertinente. *In*: **Para uma filosofia do Ato responsável**. São Carlos: Pedro & João, 2017. p. 9 – 40.

FARIAS, Adelito; ANDRADE, Wilkerson; ALENCAR, Rayana. Pensamento computacional em sala de aula: Desafios, possibilidades e a formação docente. *In*: . Sociedade Brasileira de Computação – SBC, 2015. Disponível em: https://doi.org/10.5753/cbie.wcbie.2015.1226.

FERNANDES, Hugo Batista; SILVEIRA, Ismar Frango. Pensamento computacional: iniciativas para o seu desenvolvimento por meio da modalidade de ensino a distância. *In*: . Sociedade Brasileira de Computação – SBC, 2016. Disponível em: https://doi.org/10.5753/cbie.wcbie.2016.1070.

FERREIRA, Ana Carolina; SANTOS, Juliana; SILVA, Raul; OLIVEIRA, Allan Thales Ramos; ZABOT, Diego; ABDALLA, Débora; MATOS, Ecivaldo. Hello world: relato de experiência de um curso de iniciação à programação. *In*: . Sociedade Brasileira de Computação – SBC, 2016. Disponível em: https://doi.org/10.5753/cbie.wcbie.2016.1306.

FIALA, Dalibor; TUTOKY, Gabriel. Computer science papers in web of science: A bibliometric analysis. **Publications**, v. 5, n. 4, 2017. ISSN 2304-6775. Disponível em: https://www.mdpi.com/2304-6775/5/4/23.

FINO, Carlos Nogueira. Vygotsky e a zona de desenvolvimento proximal (zdp): três implicações pedagógicas. **Revista Portuguesa de educação**, v. 14, p. 273–291, 2001.

FIORIN, J.L. **Introdução ao pensamento de Bakhtin**. *[S.l.]*: Contexto, 2016.

FLEISSNER, Sebastian; BANIASSAD, Elisa. Towards harmony-oriented programming. *In*: . *[S.l.: s.n.]*, 2008. p. 819–822.

FLOYD, Robert W. The paradigms of programming. **Commun. ACM**, ACM, New York, NY, USA, v. 22, n. 8, p. 455–460, 8 1979. ISSN 0001-0782. Disponível em: http://doi.acm.org/10.1145/359138.359140.

FONSECA FILHO, Cléuzio. **História da computação: teoria e tecnologia**. *[S.l.]*: EDIPUCRS, 2007.

FRANCHI, C. **Mas o que é mesmo "gramática"?** *[S.l.]*: Parábola Ed., 2008. (Na ponta da língua). ISBN 9788588456556.

FRANçA, Rozelma; TEDESCO, Patrícia. Desafios e oportunidades ao ensino do pensamento computacional na educação básica no brasil. *In*: . Sociedade Brasileira de Computação – SBC, 2015. Disponível em: https://doi.org/10.5753/cbie.wcbie.2015.1464.

FREIRE, P. **Pedagogia da esperança: um reencontro com a Pedagogia do oprimido**. *[S.l.]*: Paz e Terra, 2014. ISBN 9788521900108.

FREIRE, P.; VASCONCELOS, M.L.M.C.; BRITO, R.H.P. de. **Conceitos de educação em Paulo Freire: glossário**. *[S.l.]*: Fundo Mackenzie de Pesquisa, 2006. ISBN 9788532633224.

FREITAS, Marcos Cezar de. O conceito de tecnologia: O quarto quadrante do círculo de Álvaro vieira pinto. *In*: . *[S.l.: s.n.]*, 2005.

FRITZ, W. B. The women of eniac. **IEEE Annals of the History of Computing**, v. 18, n. NaN, p. 13–28, 9 1996. Disponível em: doi.ieeecomputersociety.org/10.1109/85.511940.

FRY, Ben; REAS, Casey. **Processing Overview**. 2021. Disponível em: https://processing.org/tutorials/overview/.

G1. Dona do google se torna quarta empresa nos eua a atingir us$1 trilhão em valor de mercado. Último acesso em 22 de set. de 2020, 2020. Disponível em: https://g1.globo.com/economia/tecnologia/noticia/2020/01/16/dona-do-google-se-torna-4a-empresa-a-atingir-us-1-trilhao-em-valor-de-mercado.ghtml.

GABRIEL, Richard P. The structure of a programming language revolution. *In*: . New York, NY, USA: ACM, 2012. (Onward! 2012), p. 195–214. ISBN 978-1-4503-1562-3. Disponível em: http://doi.acm.org/10.1145/2384592.2384611.

GERALDES, Wendell Bento; FERNEDA, Edilson; MARIZ, Ricardo; ALONSO, Luiza. O pensamento computacional no ensino profissional e tecnológico. *In*: . Brazilian

Computer Society (Sociedade Brasileira de Computação – SBC), 2017. Disponível em: https://doi.org/10.5753/cbie.wcbie.2017.902.

GIRLS, Rails. **Press Kit**. 2021. Disponível em: http://railsgirls.com/press.html.

GOMES, Anabela; HENRIQUES, Joana; MENDES, António José. Uma proposta para ajudar alunos com dificuldades na aprendizagem inicial de programação de computadores. **Educação, Formação & Tecnologias**, v. 1, n. 1, p. 93–103, 2008. ISSN 1646-933X.

GOMES, Anabela; MENDES, Antonio. A teacher's view about introductory programming teaching and learning: Difficulties, strategies and motivations. *In*: **2014 IEEE Frontiers in Education Conference (FIE) Proceedings**. IEEE, 2014. Disponível em: https://doi.org/10.1109/fie.2014.7044086.

GOMES, Tancicleide; BARRETO, Pedro; LIMA, Isabella Rocha Albuquerque; FALCãO, Taciana Pontual. Avaliação de um jogo educativo para o desenvolvimento do pensamento computacional na educação infantil. *In*: . Sociedade Brasileira de Computação – SBC, 2015. Disponível em: https://doi.org/10.5753/cbie.wcbie.2015.1349.

GOMES, Tancicleide CS; MELO, Jeane CB de. App inventor for android: Uma nova possibilidade para o ensino de lógica de programaçao. *In*: . *[S.l.: s.n.]*, 2013. v. 2, n. 1.

GONZATTO, Rodrigo Freese; MERKLE, Luiz Ernesto. Vida e obra de álvaro vieira pinto: um levantamento biobibliográfico. **Revista HISTEDBR On-line**, Universidade Estadual de Campinas, v. 16, n. 69, p. 286, feb 2017. Disponível em: https://doi.org/10.20396/rho.v16i69.8644246.

GOOGLE. **Google Java Style Guide**. *[s.n.]*, 2018. Disponível em: https://google.github.io/styleguide/javaguide.html.

GRENFELL, M. Metodologia. *In*: **Pierre Bourdieu: conceitos fundamentais**. *[S.l.]*: Editora Vozes, 2018. p. ??–?? ISBN 9788532657848.

GRENFELL, M. **Pierre Bourdieu: conceitos fundamentais**. *[S.l.]*: Editora Vozes, 2018. ISBN 9788532657848.

GROSSI, Miriam. Trabalho de campo & subjetividade. *In*: . *[S.l.: s.n.]*, 1992.

GRUDIN, Jonathan; WILLIAMS, Gayna. Two women who pioneered user-centered design. **interactions**, ACM, New York, NY, USA, v. 20, n. 6, p. 15–20, 11 2013. ISSN 1072-5520. Disponível em: http://doi.acm.org/10.1145/2530538.

GUTTAG, John; HORNING, Jim; WING, Jeannette. Some notes on putting formal specifications to productive use. **Science of Computer Programming**, Elsevier BV, v. 2, n. 1, p. 53–68, out. 1982. Disponível em: https://doi.org/10.1016/0167-6423(82)90004-1.

HARTEL, Pieter H.; HERTZBERGER, L. O. Paradigms and laboratories in the core computer science curriculum: An overview. **SIGCSE Bull.**, ACM, New York, NY, USA, v. 27, n. 4, p. 13–20, 12 1995. ISSN 0097-8418. Disponível em: http://doi.acm.org/10.1145/216511.216521.

HENDERSON, Peter B.; CORTINA, Thomas J.; WING, Jeannette M. Computational thinking. **SIGCSE Bull.**, Association for Computing Machinery, New York, NY, USA, v. 39, n. 1, p. 195–196, mar. 2007. ISSN 0097-8418. Disponível em: https://doi.org/10.1145/1227504.1227378.

HENN, Steve. When women stopped coding. 2014. Disponível em: https://www.npr.org/sections/money/2014/10/21/357629765/when-women-stopped-coding.

HIELE-GELDOF, Dina van; HIELE, Pierre .M van. **English Translation of Selected Writings of Dina Van Hiele-Geldof and Pierre M. Van Hiele**. *[S.l.]*: publisher not identified, 1984.

HOLANDA, Aurélio Buarque de. **Mini Dicionario Aurelio Lingua Portuguesa**. *[S.l.]*: Positivo livros, 2010. ISBN 9788538580812.

JEFFREY, R.C.; BOOLOS, G.S.; BURGESS, J.; MORTARI, C.A. **Computabilidade E Lógica**. *[S.l.]*: UNESP, 2013. ISBN 9788539303663.

KAMPFF, Adriana Justin Cerveira; LOPES, Tiago; ALVES, Isa Mara; SOUZA, Vinicius Costa de; RIGO, Sandro; MARSON, Fernando. Pensamento computacional no ensino superior: Relato de uma oficina com professores da universidade do vale do rio dos sinos. *In*: . Sociedade Brasileira de Computação – SBC, 2016. Disponível em: https://doi.org/10.5753/cbie.wcbie.2016.1316.

KAY, Alan C. Computer software. **Scientific American**, v. 251, n. 3, p. 52–59 (Intl. ed. 40–47), 9 1984.

KIRA, Gustavo; MERKLE, Luiz Ernesto. Ponteiros em papel: O ensino e a aprendizagem de ponteiros em linguagem de programação c com base em envelopes coloridos. *In*: . Porto Alegre, RS, Brasil: SBC, 2018. ISSN 2595-6175. Disponível em: https://sol.sbc.org.br/index.php/wei/article/view/3529.

KRAMER, Jeff. Is abstraction the key to computing? **Commun. ACM**, Association for Computing Machinery, New York, NY, USA, v. 50, n. 4, p. 36–42, abr. 2007. ISSN 0001-0782. Disponível em: https://doi.org/10.1145/1232743.1232745.

KUHN, T.S. **A estrutura das revoluções cientificas**. *[S.l.]*: Perspectiva, 2007. (Coleção Debates).

LATOUR, Bruno; WOOLGAR, Steve. **Laboratory life: The construction of scientific facts**. *[S.l.]*: Princeton University Press, 1986.

LEFEBVRE, Henri. **Lógica formal/ lógica dialéctica**. Rio de Janeiro: Editora Civilização Brasileira S.A., 1991.

LEFRANCOIS, G.R. **Teorias da aprendizagem: o que o professor disse**. *[S.l.]*: Cengage Learning, 2016. ISBN 9788522106226.

LEITE, Maici; REINALDO, Francisco; MASCHIO, Eleandro; MARCZAL, Diego; OLIVEIRA, Carolina Moreira. Pensamento computacional nas escolas: Limitado pela tecnologia, infraestrutura ou prática docente? *In*: . Brazilian Computer Society (Sociedade Brasileira de Computação – SBC), 2017. Disponível em: https://doi.org/10.5753/cbie.wcbie.2017.1002.

LEMOS, Guilherme Augusto Rezende. A RETIFICAÇÃo DE ERROS e o PROBLEMA DA VERDADE NA EPISTEMOLOGIA DE GASTON BACHELARD. **Linguagens, Educação e Sociedade**, Universidade Federal do Piaui, v. 1, n. 40, p. 138, 11 2018. Disponível em: https://doi.org/10.26694/les.v1i40.7659.

LÉVÉNEZ, Éric. **History of programming languages**. *[S.l.: s.n.]*, 2004.

LIMA, Árllon Chaves; SOUSA, Decíola Fernandes de. Desenvolvimento do raciocínio lógico e algoritmo através do programa institucional de bolsas de iniciação à docência no ensino fundamental. *In*: . Sociedade Brasileira de Computação – SBC, 2015. Disponível em: https://doi.org/10.5753/cbie.wcbie.2015.1379.

LIMA, Árllon Chaves; SOUSA, Decíola Fernandes de. Experiência no programa institucional de bolsas de iniciação à docência (PIBID): Desenvolvimento do raciocínio lógico e algoritmo na educação básica. *In*: . Sociedade Brasileira de Computação – SBC, 2015. Disponível em: https://doi.org/10.5753/cbie.wcbie.2015.1290.

LIMA, Michelle Pinto. As mulheres na ciência da computação. **Revista Estudos Feministas**, FapUNIFESP (SciELO), v. 21, n. 3, p. 793–816, dez. 2013. Disponível em: https://doi.org/10.1590/s0104-026x2013000300003.

LIPPERT, Eric. References are not addresses. Último acesso em 31 de mar. de 2018, 2018. Disponível em: https://blogs.msdn.microsoft.com/ericlippert/2009/02/17/references-are-not-addresses.

LISKOV, Barbara H.; WING, Jeannette M. A behavioral notion of subtyping. **ACM Transactions on Programming Languages and Systems**, Association for Computing Machinery (ACM), v. 16, n. 6, p. 1811–1841, nov. 1994. Disponível em: https://doi.org/10.1145/197320.197383.

LéVéNEZ, Éric. Computer languages timeline. Último acesso em 21 de nov. de 2021, 2021. Disponível em: https://www.levenez.com/lang/.

MACHADO, Nilson José. **Matemática e língua materna**. São Paulo: Cortez: Autores Associados, 2011.

MAHESHWARI, Piyush. Teaching programming paradigms and languages for qualitative learning. *In*: . New York, NY, USA: ACM, 1996. (ACSE '97), p. 32–39. ISBN 0-89791-958-0. Disponível em: http://doi.acm.org/10.1145/299359.299365.

MAIA, Marcel Maggion. Limites de gênero e presença feminina nos cursos superiores brasileiros do campo da computação. **Cadernos Pagu**, FapUNIFESP (SciELO), n. 46, p. 223–244, abr. 2016. Disponível em: https://doi.org/10.1590/18094449201600460223.

MALERBA, J. **A história escrita: teoria e história da historiografia**. *[S.l.]*: Contexto, 2006. ISBN 9788572443036.

MARGOLIN, Victor. **Políticas do artificial: ensaio e estudos sobre design**. *[S.l.]*: Record, 2014.

MARGOLIS, J.; ESTRELLA, R.; GOODE, J.; HOLME, J.J.; NAO, K. **Stuck in the Shallow End: Education, Race, and Computing**. *[S.l.]*: MIT Press, 2010. (The MIT Press). ISBN 9780262260961.

MARGOLIS, J.; FISHER, A. **Unlocking the Clubhouse: Women in Computing**. *[S.l.]*: MIT Press, 2002. (Mit Press). ISBN 9780262632690.

MARGOLIS, Jane; GOODE, Joanna; BINNING, Kevin R. Exploring computer science: Active learning for broadening participation in computing.

Último acesso em 13 de nov. de 2018, 2018. Disponível em: https://cra.org/crn/2015/10/expanding-the-pipeline-exploring-computer-science-active-learning-for-broadening-participation-in-comput

MEKSENAS, Paulo. As noções de concreto e abstrato: sua relação com as práticas de ensino. **Revista da Faculdade de Educação**, v. 18, n. 1, p. 92–98, 1992.

MENDONçA, André Luis de Oliveira. O legado de thomas kuhn após cinquenta anos. **Scientiae Studia**, scielo, v. 10, p. 535 – 560, 0 2012.

MERKLE, Luiz Ernesto. **Disciplinary and Semiotic Relations across Human-Computer Interaction**. 2002. Tese (Doutorado) — The University of Western Ontario (Canada), 2002. AAINQ68095.

MESTRE, Palloma; ANDRADE, Wilkerson; GUERRERO, Dalton; SAMPAIO, Livia; RODRIGUES, Rivanilson Da Silva; COSTA, Erick. Pensamento computacional: Um estudo empírico sobre as questoes de matemática do PISA. *In*: . Sociedade Brasileira de Computação – SBC, 2015. Disponível em: https://doi.org/10.5753/cbie.wcbie.2015.1281.

MICROSOFT. General naming conventions. Último acesso em 21 de nov. de 2021, 2021. Disponível em: https://docs.microsoft.com/en-us/dotnet/standard/design-guidelines/general-naming-conventionsredirectedfrom=MSDN.

MICROSYSTEMS, Sun. **Java Code Conventions**. *[s.n.]*, 1997. Disponível em: http://www.oracle.com/technetwork/java/codeconventions-150003.pdf.

MILLER, Gary. The spreadsheet paradigm: A basis for powerful and accessible programming. *In*: . New York, NY, USA: ACM, 2015. (SPLASH Companion 2015), p. 33–35. ISBN 978-1-4503-3722-9. Disponível em: http://doi.acm.org/10.1145/2814189.2814201.

MILNE, Iain; ROWE, Glenn. Difficulties in learning and teaching programming—views of students and tutors. **Education and Information Technologies**, v. 7, n. 1, p. 55–66, 3 2002. ISSN 1573-7608. Disponível em: https://doi.org/10.1023/A:1015362608943.

MONTEIRO, M.A. **Introdução À Organização de Computadores (5a. Ed.).** *[S.l.]*: Grupo Gen – LTC, 2000. ISBN 9788521618492.

MYCROFT, Alan. On integration of programming paradigms. **ACM Comput. Surv.**, ACM, New York, NY, USA, v. 28, n. 2, p. 309–311, 6 1996. ISSN 0360-0300. Disponível em: http://doi.acm.org/10.1145/234528.234735.

NCSES. Science and engineering degrees, by race and ethnicity of recipients: 2008–18. 2021. Disponível em: https://ncsesdata.nsf.gov/sere/2018/#tabs-1.

NUNES, Daltro José. **Ciência da Computação na Educação Básica**. 2011. Disponível em: http://www.adufrgs.org.br/artigos/ciencia-da-computacao-na-educacao-basica/.

NUNES, Jordão Horta. Gênero e raça no trabalho em tecnologia da informação (TI). **Ciências Sociais Unisinos**, UNISINOS - Universidade do Vale do Rio Dos Sinos, v. 52, n. 3, jul. 2016. Disponível em: https://doi.org/10.4013/csu.2016.52.3.09.

OBAMA WHITE HOUSE, THE. **President Obama Meets with Students at an "Hour of Code" Event**. *[s.n.]*, 2014. Disponível em: https://www.youtube.com/watch?v=EJEBGf_uS_M.

OLIVEIRA, José; BARBOSA, Alexandre. Perfis de jogadores em contextos de ensino/aprendizagem em disciplinas de programação. *In*: . Sociedade Brasileira de Computação – SBC, 2016. Disponível em: https://doi.org/10.5753/cbie.wcbie.2016.1137.

ORACLE. **Naming Conventions**. *[s.n.]*, 1999. Disponível em: http://www.oracle.com/technetwork/java/codeconventions-135099.html.

ORTIZ, Julia; PEREIRA, Roberto. Pensamento computacional na educação de jovens e adultos: desafios e oportunidades. *In*: . Brazilian Computer Society (Sociedade Brasileira de Computação – SBC), 2017. Disponível em: https://doi.org/10.5753/cbie.wcbie.2017.1069.

OXFORD LEARNERS DICTIONARIES. K-12: adjective. 2021. Disponível em: https://www.oxfordlearnersdictionaries.com/definition/english/k-12.

PAIVA, Luiz Fernando de; FERREIRA, Ana Carolina; ROCHA, Caio; BARRETO, Jandiaci; MELHOR, André; LOPES, Randerson; MATOS, Ecivaldo. Uma experiência piloto de integração curricular do raciocínio computacional na educação básica. *In*: . Sociedade Brasileira de Computação – SBC, 2015. Disponível em: https://doi.org/10.5753/cbie.wcbie.2015.1300.

PAPERT, Seymour. **A Máquina das Crianças**. Porto Alegre: Artes Médicas, 1994.

PAUL, Annie Murphy. Are college lectures unfair? Último acesso em 13 de nov. de 2018, 2015. Disponível em: https://www.nytimes.com/2015/09/13/opinion/sunday/are-college-lectures-unfair.html.

PIMENTEL, Edson; FRANçA, Vilma; NORONHA, Robinson; OMAR, Nizam. Avaliação contínua da aprendizagem, das competências e habilidades em programação de computadores. **Anais do Workshop de Informática na Escola**, v. 1, n. 1, p. 533–544, 2003. ISSN 2316-6541. Disponível em: https://www.br-ie.org/pub/index.php/wie/article/view/819.

PINHEIRO, Andreia; FRANCO, João; LEITE, Jorge. Desenvolvimento do pensamento computacional e discussões sobre representação feminina na computação: um estudo de caso. *In*: . Brazilian Computer Society (Sociedade Brasileira de Computação – SBC), 2017. Disponível em: https://doi.org/10.5753/cbie.wcbie.2017.1111.

PPGTE. Mediações e culturas. 2018. Disponível em: http://www.utfpr.edu.br/curitiba/estrutura-universitaria/diretorias/dirppg/programas/ppgte/areas-pesquisa/tecnologia-e-interacao.

PPGTE. Área de concentração. 2018. Disponível em: http://www.utfpr.edu.br/curitiba/estrutura-universitaria/diretorias/dirppg/programas/ppgte/area-de-concentracao.

PRESTES, Z. **Quando não é quase a mesma coisa: traduções de Lev Semionovitch Vigotski no Brasil**. *[S.l.]*: Autores Associados, 2021. ISBN 9786588717288.

PRETA, Info. **Press Kit**. 2021. Disponível em: https://infopreta.com.br/quem-somos/.

PRETALAB. **PretaLab**. 2021. Disponível em: https://www.pretalab.com/.

QUEIROZ, Rubens; SAMPAIO, Fábio Ferrentini; SANTOS, Mônica Pereira dos. DuinoBlocks4kids : Ensinando conceitos básicos de programação a crianças do ensino fundamental i por meio da robótica educacional. *In*: . Sociedade Brasileira de Computação – SBC, 2016. Disponível em: https://doi.org/10.5753/cbie.wcbie.2016.1169.

RAABE, André; SANTANA, André Luiz Maciel; ELLERY, Natália; GONçALVES, Filipe. Um instrumento para diagnóstico do pensamento computacional. *In*: . Brazilian Computer Society (Sociedade Brasileira de Computação – SBC), 2017. Disponível em: https://doi.org/10.5753/cbie.wcbie.2017.1172.

RAABE, André; ZANCHETT, Guilherme; VAHLDICK, Adilson. Jogos de programar como uma abordagem para os primeiros contatos dos estudantes com à programação. *In*: . Sociedade Brasileira de Computação – SBC, 2015. Disponível em: https://doi.org/10.5753/cbie.wcbie.2015.1485.

RAEDER, Meteus; PY, Mônica; RIGO, Sandro; PINHEIRO, Josaine. L2pm: relato de uma experiência sobre o ensino integrado de lógica, programação e matemática para computação. *In*: . *[S.l.: s.n.]*, 2016.

RAGONIS, Noa; HABERMAN, Bruria. Linking different programming paradigms: Thoughts about instructional design. *In*: . New York, NY, USA: ACM, 2010. (ITiCSE '10), p. 310–310. ISBN 978-1-60558-820-9. Disponível em: http://doi.acm.org/10.1145/1822090.1822187.

RAIOL, Alberto A. C.; SARGER, João; SOUZA, Aline; SIVALDO, Silva; FáBIO, Bezerra. Resgatando a linguagem de programação logo: Uma experiência com calouros no ensino superior. *In*: . *[S.l.: s.n.]*, 2015.

REINFELDS, Juris. A three paradigm first course for cs majors. *In*: . New York, NY, USA: ACM, 1995. (SIGCSE '95), p. 223–227. ISBN 0-89791-693-X. Disponível em: http://doi.acm.org/10.1145/199688.199792.

RESNICK, Mitchel; MALONEY, John; MONROY-HERNáNDEZ, Andrés; RUSK, Natalie; EASTMOND, Evelyn; BRENNAN, Karen; MILLNER, Amon; ROSENBAUM, Eric; SILVER, Jay; SILVERMAN, Brian; KAFAI, Yasmin. Scratch: Programming for all. **Commun. ACM**, ACM, New York, NY, USA, v. 52, n. 11, p. 60–67, 11 2009. ISSN 0001-0782. Disponível em: http://doi.acm.org/10.1145/1592761.1592779.

ROSSUM, Nick Coghlan Guido van. **Style Guide for Python Code**. 2001. Disponível em: https://legacy.python.org/dev/peps/pep-0008/. Acesso em: 15 jul. 2018.

ROWLEY, Michael. Guidance trees: A new programming paradigm for non-programmers. *In*: . New York, NY, USA: ACM, 2011. (OOPSLA '11), p. 33–34. ISBN 978-1-4503-0942-4. Disponível em: http://doi.acm.org/10.1145/2048147.2048164.

SANTOS, Boaventura de Sousa. Para uma sociologia das ausências e uma sociologia das emergências. **Revista Crítica de Ciências Sociais**, OpenEdition, n. 63, p. 237–280, out. 2002. Disponível em: https://doi.org/10.4000/rccs.1285.

SANTOS, Carolina Marins. Por que as mulheres "desapareceram" dos cursos de computação? 2016. Disponível em: https://jornal.usp.br/universidade/por-que-as-mulheres-desapareceram-dos-cursos-de-computacao/.

SCHOEFFEL, Pablo; MOSER, Paolo; VARELA, Geraldo; DURIGON, Letícia; ALBUQUERQUE, Gustavo Cibils de; NIQUELATTI, Matheus. Uma experiência no ensino de pensamento computacional para alunos do ensino fundamental. *In*: . Sociedade Brasileira de Computação – SBC, 2015. Disponível em: https://doi.org/10.5753/cbie.wcbie.2015.1474.

SELBY, Cynthia; WOOLLARD, John. Computational thinking: the developing definition. University of Southampton (E-prints), 2013.

SELBY, Cynthia; WOOLLARD, John. Refining an understanding of computational thinking. Unpublished. 2014. Disponível em: https://eprints.soton.ac.uk/372410/.

SILVA, Débora Priscilla da; SIDNEI, Simone; JESUS, Ângelo; SILVA, Carlos Eduardo Paulino. Aplicação de robótica na educação de forma gradual para o estímulo do pensamento computacional. *In*: . Sociedade Brasileira de Computação – SBC, 2016. Disponível em: https://doi.org/10.5753/cbie.wcbie.2016.1188.

SILVA, Guilherme Alves da. **Uma perspectiva crítica para as políticas públicas de inclusão digital no Brasil: estudo de caso sobre não-usos e não-usuários de internet**. 2020. Dissertação (Mestrado) — UTFPR, 2020.

SILVA, Gercineide Torres da; SOUZA, José Luziel de; SILVA, Luiz Augusto Matos da. Aplicação da ferramenta scratch para o aprendizado de programação no ensino fundamental i. *In*: . Sociedade Brasileira de Computação – SBC, 2016. Disponível em: https://doi.org/10.5753/cbie.wcbie.2016.1285.

SILVA, J.J. da. **Filosofias da matemática**. *[S.l.]*: UNESP, 2007. ISBN 9788571397514.

SILVA, Joseli Maria; SILVA, Edson Armando; JUNCKES, Ivan Jairo. **Construindo a ciência: elaboração crítica de projetos de pesquisa**. Curitiba: Pós Escrito, 2009.

SILVA, Leuson M. P. Da; BONFIM, Brendo C.; SILVA, Rogério C.; SILVA, Jefferson B. da. Poogame: Um jogo sériopar ao ensino de programação orientada a objetos. *In*: . *[S.l.: s.n.]*, 2016.

SILVA, Nyara Cardoso; LIMA, Ana Carina; SOUZA, Niellen De; SOUSA, Decíola Fernandes de. Raciocínio lógico nas escolas: Uma introdução ao ensino de algoritmos de programação. *In*: . Brazilian Computer Society (Sociedade Brasileira de Computação – SBC), 2017. Disponível em: https://doi.org/10.5753/cbie.wcbie.2017.1011.

SILVA, Thiago Reis. Desenvolvendo a programação de jogos digitais no ensino médio: um relato de experiência utilizando a ferramenta construct2. *In*: . Brazilian Computer Society (Sociedade Brasileira de Computação – SBC), 2017. Disponível em: https://doi.org/10.5753/cbie.wcbie.2017.1142.

SILVA, Tomaz Tadeu da. A produção social da identidade e da diferença. *In*: SILVA, Tomaz Tadeu da (Ed.). **Identidade e diferença: a perspectiva dos estudos culturais**. Rio de Janeiro: Editora Vozes, 2000. p. 73–102.

SIMON, Herbert A. **The Sciences of the Artificial (3rd Ed.)**. Cambridge, MA, USA: MIT Press, 1996. ISBN 0-262-69191-4.

SIMONYI, Charles. **Hungarian Notation**. Microsoft, 1999. Disponível em: https://docs.microsoft.com/en-us/previous-versions/visualstudio/visual-studio-6.0/aa260976(v=vs.60)?redirectedfrom=MSDN.

SMITH, M.R.; MARX, L. **Does Technology Drive History?: The Dilemma of Technological Determinism**. *[S.l.]*: PAPERBACKSHOP UK IMPORT, 1994. (Mit Press). ISBN 9780262691673.

SOBRAL, A. **Do dialogismo ao gênero: as bases do pensamento do círculo de Bakhtin**. *[S.l.]*: Mercado de Letras, 2009. (Idéias sobre linguagem). ISBN 9788575911228.

SOBRAL, A. **A filosofia primeira de Bakhtin**. *[S.l.]*: Mercado de Letras, 2019. (Idéias sobre linguagem). ISBN 9788575911228.

SOUTO, Mychelline; TEDESCO, Patrícia. Uma revisão sistemática da literatura sobre conhecimentos, habilidades, atitudes e competências desejáveis para auxiliar a aprendizagem de programação. *In*: . Brazilian Computer Society (Sociedade Brasileira de Computação – SBC), 2017. Disponível em: https://doi.org/10.5753/cbie.wcbie.2017.1162.

SOUZA, Isabelle Maria Lima de; RODRIGUES, Rivanilson Da Silva; ANDRADE, Wilkerson. Introdução do pensamento computacional na formação docente para ensino de robótica educacional. *In*: . Sociedade Brasileira de Computação – SBC, 2016. Disponível em: https://doi.org/10.5753/cbie.wcbie.2016.1265.

SOUZA, Saymon; CASTRO, Thais. Investigação em programação com scratch para crianças: uma revisão sistemática da literatura. *In*: . Sociedade Brasileira de Computação – SBC, 2016. Disponível em: https://doi.org/10.5753/cbie.wcbie.2016.1078.

SOUZA, Solange Jobim e; ALBUQUERQUE, Elaine Deccache Porto e. A pesquisa em ciências humanas: uma leitura bakhtiniana. **Bakhtiniana: Revista de Estudos do Discurso**, scielo, v. 7, p. 109 – 122, 12 2012.

SOUZA, Tatiele Pereira de; TOSTA, Tania Ludmila Dias. Imagens de gênero e raça na tecnologia da informação: invisibilidades negras, territérios brancos; mulheres ocultas, espaços masculinos. **Cadernos de Gênero e Tecnologia**, v. 13, n. 42, 2020.

SPRAGUE, Peter; SCHAHCZENSKI, Celia. Abstraction the key to cs1. **J. Comput. Sci. Coll.**, Consortium for Computing Sciences in Colleges, Evansville, IN, USA, v. 17, n. 3, p. 211–218, fev. 2002. ISSN 1937-4771.

STOLIN, Yuila; HAZZAN, Orit. Students' understanding of computer science soft ideas: the case of programming paradigm. **ACM SIGCSE Bulletin**, ACM, v. 39, n. 2, p. 65–69, 2007.

SUNG, Kelvin; SNYDER, Lawrence. A case of computer science principles with traditional text-based programming languages. **J. Comput. Sci. Coll.**, Consortium for Computing Sciences in Colleges, USA, v. 30, n. 1, p. 161–172, 10 2014. ISSN 1937-4771. Disponível em: http://dl.acm.org/citation.cfm?id=2667369.2667399.

TAFT, S. Tucker; BLOCH, Joshua; BOCCHINO, Robert; BURCKHARDT, Sebastian; CHAFI, Hassan; COX, Russ; GASTER, Benedict; STEELE, Guy; UNGAR, David. Multicore, manycore, and cloud computing: Is a new programming language paradigm required? *In*: . New York, NY, USA: ACM, 2011. (OOPSLA '11), p. 165–170. ISBN 978-1-4503-0942-4. Disponível em: http://doi.acm.org/10.1145/2048147.2048192.

TEDRE, Matti; DENNING, Peter J. The long quest for computational thinking. *In*: **Proceedings of the 16th Koli Calling International Conference on Computing Education Research**. *[S.l.: s.n.]*, 2016. p. 120–129.

TIMMERMANN, Glaucia Keidann; GONZáLEZ, Fernando. Mediações que os professores e alunos estabelecem com o conteúdo da disciplina de algoritmos de cursos superiores:

estudo de caso. *In*: . Sociedade Brasileira de Computação – SBC, 2016. Disponível em: https://doi.org/10.5753/cbie.wcbie.2016.1295.

TOKARNIA, Mariana. Um em cada 4 brasileiros não tem acesso à internet, mostra pesquisa. Último acesso em 22 de set. de 2020, 2020. Disponível em: https://agenciabrasil.ebc.com.br/economia/noticia/2020-04/um-em-cada-quatro-brasileiros-nao-tem-acesso-internet.

TRIPP, David. Pesquisa-ação: uma introdução metodológica. **Educação e Pesquisa**, scielo, v. 31, p. 443 – 466, 12 2005.

TURKLE, Sherry; PAPERT, Seymour. Epistemological pluralism and the revaluation of the concrete. **Journal of Mathematical Behavior**, v. 11, n. 1, p. 3–33, 1992.

UNION, ITU International Telecommunications. **Percentage of Individuals using the Internet**. ITU, 2012. Disponível em: http://www.itu.int/en/ITU-D/Statistics/Documents/statistics/2013/Individuals_Internet_2000-2012.xls.

U.S. Equal Employment Opportunity Commission. **EEO-1 Data Collection**. 2021. Disponível em: https://www.eeoc.gov/employers/eeo-1-data-collection. Acesso em: 15 jun. 2021.

VAN ROY, Peter. Programming paradigms for dummies: What every programmer should know. 2009.

VASSALLO, Trae; LEVY, Ellen; MADANSKY, Michele; MICKELL, Hillary; PORTER, Bennett; LEAS, Monica. Elefant in the valley. 2018. Disponível em: https://www.elephantinthevalley.com/.

VEE, Annette. Understanding computer programming as a literacy. **Literacy in Composition Studies**, v. 1, n. 2, p. 42–64, 2013.

VEER, R. van der; VALSINER, J. **Vygotsky - Uma síntese**. *[S.l.]*: Loyola, 2014.

VELÁZQUEZ-ITURBIDE, J. Ángel. A programming languages course for freshmen. **SIGCSE Bull.**, ACM, New York, NY, USA, v. 37, n. 3, p. 271–275, 6 2005. ISSN 0097-8418. Disponível em: http://doi.acm.org/10.1145/1151954.1067519.

VERA, William Fabian Machado; MELHOR, André; COSTA, Wallyson Nascimento; MATOS, Ecivaldo. Dó, ré, mergesort: um relato de experiência interdisciplinar de ensino de computação com matemática e música. *In*: . Sociedade Brasileira de Computação – SBC, 2016. Disponível em: https://doi.org/10.5753/cbie.wcbie.2016.1275.

VIEIRA PINTO, Álvaro. **O conceito de tecnologia**. Rio de Janeiro, RJ: Contraponto, 2005.

VIEIRA PINTO, Álvaro. **O conceito de tecnologia**. Rio de Janeiro, RJ: Contraponto, 2005.

VIGOTSKI, L.S. **Psicologia pedagógica**. *[S.l.]*: Martins Fontes, 2004. ISBN 9788533620728.

VIGOTSKI, L.S. **Psirrologuia razvitia rebionka**. *[S.l.]*: Eksmo, 2004.

VOLÓCHINOV, V. **Marxismo E Filosofia Da Linguagem: PROBLEMAS FUNDAMENTAIS DO MÉTODO SOCIOLÓGICO NA CIÊNCIA DA LINGUAGEM**. *[S.l.]*: EDITORA 34, 2017 [1929]. ISBN 9788573266610.

VOLOCHINOV, V. A palavra na vida e a palavra na poesia: para uma poética sociológica. *In*: **A palavra na vida e a palavra na poesia**. *[S.l.]*: EDITORA 34, 2019 [1926]. p. 109–146.

VRETTAS, George; SANDERSON, Mark. Conferences versus journals in computer science. **Journal of the Association for Information Science and Technology**, v. 66, n. 12, p. 2674–2684, 2015. Disponível em: https://asistdl.onlinelibrary.wiley.com/doi/abs/10.1002/asi.23349.

VYGOTSKY, L.S. **A formação social da mente: o desenvolvimento dos processos psicológicos superiores**. *[S.l.]*: Martins Fontes, 2007. (Psicologia e Pedagogia). ISBN 9788533600249.

VYGOTSKY, L. S. La prehistoria del desarrollo del lenguaje escrito. *In*: VYGOTSKY, L. S. (Ed.). **Vygotsky, Lev S. Obras Escogidas III**. Madrid: *[s.n.]*, 1995.

WALGPROG. **I Workshop de Ensino em Pensamento Computacional, Algoritmos e Programação**. *[s.n.]*, 2015. Disponível em: http://walgprog.gp.utfpr.edu.br/2015/.

WATT, D.A. **Programming Language Design Concepts**. *[S.l.]*: Wiley India Pvt. Limited, 2004. ISBN 9788126505272.

WEISUL, Kimberly. **Half of This College's STEM Graduates Are Women. Here's What It Did Differently**. 2017. Disponível em: https://www.inc.com/kimberly-weisul/how-harvey-mudd-college-achieved-gender-parity-computer-science-engineering-physics.html. Acesso em: 07 mar. 2021.

WELLS, Mark B; KURTZ, Barry L. Teaching multiple programming paradigms: a proposal for a paradigm general pseudocode. **ACM SIGCSE Bulletin**, ACM, v. 21, n. 1, p. 246–251, 1989.

WERTSCH, James V; SOHMER, Richard. Vygotsky on learning and development. **Human Development**, S. Karger AG, v. 38, n. 6, p. 332–337, 1995. Disponível em: https://doi.org/10.1159/000278339.

WIESE, Eliane S.; YEN, Michael; CHEN, Antares; SANTOS, Lucas A.; FOX, Armando. Teaching students to recognize and implement good coding style. *In*: . New York, NY, USA: ACM, 2017. (L@S '17), p. 41–50. ISBN 978-1-4503-4450-0. Disponível em: http://doi.acm.org/10.1145/3051457.3051469.

WIKIPEDIA. **Super Trunfo**. 2022. Disponível em: https://pt.wikipedia.org/wiki/Super_Trunfo. Acesso em: 27 jan. 2022.

WING, Jeannette M. Computational thinking. **Commun. ACM**, ACM, New York, NY, USA, v. 49, n. 3, p. 33–35, 3 2006. ISSN 0001-0782. Disponível em: http://doi.acm.org/10.1145/1118178.1118215.

WING, Jeannette M. Computational thinking and thinking about computing. *In*: . New York, NY, USA: IEEE, 2008. p. 1–1.

WING, Jeannette M. Research notebook: Computational thinking–what and why? Último acesso em 24 de nov. de 2021, 2011. Disponível em: https://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why.

WING, Jeannette M. Computational thinking benefits society. Último acesso em 24 de nov. de 2021, 2021. Disponível em: http://socialissues.cs.toronto.edu/index.html%3Fp=279.html.

WING, Jeannette M.; STANZIONE, Dan. Progress in computational thinking, and expanding the HPC community. **Communications of the ACM**, Association for Computing Machinery (ACM), v. 59, n. 7, p. 10–11, jun. 2016. Disponível em: https://doi.org/10.1145/2933410.

WINNER, L. **The Whale and the Reactor: A Search for Limits in an Age of High Technology**. *[S.l.]*: University of Chicago Press, 2010. ISBN 9780226902098.

ZANETTI, Humberto; OLIVEIRA, Claudio. Práticas de ensino de programação de computadores com robótica pedagógica e aplicação de pensamento computacional. *In*: . Sociedade Brasileira de Computação – SBC, 2015. Disponível em: https://doi.org/10.5753/cbie.wcbie.2015.1236.

ZEHETMEIER, Daniela; BÖTTCHER, Axel; BRÜGGEMANN-KLEIN, Anne; THURNER, Veronika. Defining the competence of abstract thinking and evaluating CS-students' level of abstraction. *In*: **Proceedings of the 52nd Hawaii International Conference on System Sciences**. Hawaii International Conference on System Sciences, 2019. Disponível em: https://doi.org/10.24251/hicss.2019.921.

ZHANG, Lin; GLäNZEL, Wolfgang. Proceeding papers in journals versus the “regular” journal publications. **Journal of Informetrics**, v. 6, n. 1, p. 88–96, 2012. ISSN 1751-1577. Disponível em: https://www.sciencedirect.com/science/article/pii/S175115771100071X.

ZIMMERMANN, Jussara; WATANABE, Andreia; BARCELOS, Thiago; MANCINI, Felipe. Proposta de aplicação e avaliação de conceitos do pensamento computacional em crianças hospitalizadas. *In*: . Sociedade Brasileira de Computação – SBC, 2016. Disponível em: https://doi.org/10.5753/cbie.wcbie.2016.1249.

ZUHUD, Daeng Ahmad Zuhri. Some prospective approaches for the shift of programming paradigms. *In*: . New York, NY, USA: ACM, 2013. (ISDOC ’13), p. 87–93. ISBN 978-1-4503-2299-7. Disponível em: http://doi.acm.org/10.1145/2503859.2503873.

APÊNDICES

APÊNDICE A – PENSAMENTO COMPUTACIONAL: TEXTOS DO WALGPROG (2015-2017) QUE USAM O TERMO "PENSAMENTO COMPUTACIONAL" COM NO MÁXIMO 10 OCORRÊNCIAS

Tabela 26 – Textos do WAlgProg (2015-2017) que usam o termo "pensamento computacional" no máximo dez vezes

quant.	autor	parágrafo
1	Arantes e Ferreira (2015)	Projetos envolvendo a programação de computadores aplicada à educação tiveram o seu auge na década de 80, com a linguagem Logo [Papert 1980]. O uso do Logo pregava uma modificação estrutural quanto ao papel da tecnologia, colocando o computador como um poderoso aliado na construção e materialização do conhecimento. Atualmente, muitos voltaram a defender a ideia de que saber programar é uma habilidade importante na sociedade contemporânea. Não basta saber criar textos, usar redes sociais e usar a Internet, é preciso saber também como os computadores funcionam e estimular o desenvolvimento do **pensamento computacional** [Resnick et al. 2009].
1	Lima e Sousa (2015b)	Resumo. O artigo tem como objetivo apresentar a experiência adquirida por um bolsista do Programa Institucional de Bolsas de Iniciação à Docência (PIBID) através de um curso realizado na escola E.E.E.F.M. Barão de Igarapé Miri em Belém-PA, destinado ao ensino da construção de algoritmo. Trabalhou-se o **pensamento computacional**, o raciocínio lógico e o próprio algoritmo, buscando-se alcançar o desempenho e compreensão dos alunos do conteúdo ministrado, colocando em prática os conhecimentos adquiridos por meio da utilização dos computadores da sala de informática na escola, através da pseudo-linguagem "Portugol" e de exercícios relacionados ao cotidiano, para facilitar o processo de ensino e aprendizagem.
2	Lima e Sousa (2015b)	Existem outros projetos com **pensamento computacional** e Lógica de Programação que atuam no ensino superior como o que [Barcelos 2013] explora em seu projeto de pesquisa as relações entre o conhecimento matemático prévio dos alunos ingressantes em cursos superiores na área de Computação e o desenvolvimento de habilidades e competências básicas para o domínio da tecnologia, denominadas como **pensamento computacional**, e as possíveis interações entre ambos no contexto de uma oficina de desenvolvimento de jogos digitais, elaborada e oferecida dentro da estrutura curricular de um curso técnico em Informática. Para o autor a Lógica de Programação é indispensável para a continuidade com sucesso dos estudos na área. Dessa forma, identificar como os novos conhecimentos em programação se ancoram nos conceitos matemáticos já presentes (ou que deveriam estar presentes) pode constituir-se em uma contribuição para aperfeiçoar o ensino de Computação e Matemática.
1	Lima e Sousa (2015b)	Barcelos, T S. (2013). Relações entre o **Pensamento Computacional** e a Matemática através da construção de Jogos Digitais. In SBC – Proceedings of SBGames 2013. Cruzeiro do Sul, Brasil.

(continua)

Tabela 26 – Textos do WAlgProg (2015-2017) que usam o termo "pensamento computacional" no máximo dez vezes

(continuação)

quant.	autor	parágrafo
1	Aguiar *et al.* (2015)	Resumo. O ensino em computação, assim como o desenvolvimento de competências e habilidades ligadas à tecnologia, é um fator muito importante para o desenvolvimento do **pensamento computacional** e o raciocínio lógico, contribuindo com a capacidade de resolução de problemas. No entanto, o número de interessados em computação vem sofrendo redução e, por este motivo, o desempenho dos estudantes nas disciplinas introdutórias demonstra ser uma preocupação para os educadores na área. Esse artigo apresenta o Pré-Comp, um projeto que além de contribuir com a fundamentação dos princípios básicos da computação também contribui com a familiarização com o meio acadêmico e perspectivas no campo profissional, assim como para o desenvolvimento da criatividade, interesse e motivação dos estudantes.
2	Aguiar *et al.* (2015)	Numa terceira etapa, todos os calouros são divididos em grupos de até sete pessoas, sob a tutoria de alunos veteranos previamente selecionados dos cursos de graduação em questão. Cada grupo é individualmente treinado pelos seus tutores, ou como comumente chamados "treinadores", e é indispensável a diversidade dos grupos que contêm alunos dos três diferentes cursos. O principal objetivo das aulas é apresentar uma fundamentação da computação, principalmente trabalhar o **pensamento computacional** e o raciocínio lógico. Como ementa das aulas foi atribuído o conceito e tipos de representação de algoritmos (descrição narrativa, fluxograma e pseudocódigo), e conceitos como arranjo, vetor, matriz, ponteiro, sub-rotina e operadores (lógicos, aritméticos e relacionais). Nenhuma linguagem de programação é ensinada durante o Pré-Comp visto que o foco principal é construir o **pensamento computacional** e oferecer ferramentas para que posteriormente o algoritmo para solução de um problema possa ser facilmente traduzido para uma linguagem. Ao fim de cada treinamento é realizada uma prova prática a fim de avaliar cada grupo.
2	Gomes *et al.* (2015)	Resumo. Os jogos digitais têm estado cada vez mais presentes nos contextos educacionais, e suas características intrínsecas despontam como possibilidades inovadoras para apresentar conteúdos de maneira mais atraente e motivadora. O presente artigo versa sobre a avaliação de um jogo educativo para o ensino do **pensamento computacional** sob dois vieses: o da avaliação formativa e o da avaliação objetiva. Os resultados demonstram que o jogo pode ser uma possibilidade adequada para o ensino do **pensamento computacional**, mas se faz necessário o acompanhamento de um professor.

(continua)

Tabela 26 – Textos do WAlgProg (2015-2017) que usam o termo "pensamento computacional" no máximo dez vezes

(continuação)

quant.	autor	parágrafo
1	Gomes *et al.* (2015)	Von Wangenheim e Von Wangenheim (2012) afirmam que os jogos proporcionam um ambiente que fomenta a experimentação e inclui a visualização das consequências dos erros cometidos, permitindo que o estudante possa aprender na prática, além de obter um feedback instantâneo e customizado de acordo com sua performance. Os jogos educativos, em específico, podem ser definidos como jogos que possuem uma proposta pedagógica inserida em seu conteúdo e contribuem para que o processo de ensino-aprendizagem ocorra de forma mais natural, prazerosa e dinâmica [Nicoletti e Filho 2004]. A partir deste cenário, e considerando as possibilidades oferecidas pelos jogos digitais que favorecem a motivação, o entretenimento e a ludicidade, emerge a busca por jogos educativos adequados para auxiliar no desenvolvimento do **pensamento computacional**, que Von Wangenheim et al. (2014) definem como:
1	Gomes *et al.* (2015)	Neste sentido, este artigo apresenta a avaliação de um jogo educacional voltado ao desenvolvimento do **pensamento computacional** na educação infantil (The Foos), a partir de duas abordagens: objetiva e formativa.
1	Gomes *et al.* (2015)	The Foos consiste em um jogo criado para auxiliar na disseminação do **pensamento computacional** para crianças a partir dos cinco anos de idade. Nele, o jogador deve guiar os personagens para realizar pequenas tarefas, e à medida que o jogador avança de fase, novos recursos e botões de comando são introduzidos.
1	Gomes *et al.* (2015)	Já a avaliação formativa permitiu observar aspectos que abrangem desde a motivação das crianças ao longo do jogo, até mesmo a compreensão de elementos da interface e a compreensão de alguns conceitos elementares associados ao **pensamento computacional**. As crianças demonstraram um envolvimento intenso com o jogo, mesmo diante de desafios para os quais elas não conseguiram desenvolver a solução de imediato. Comumente, quando as crianças se deparam com situações nos jogos em que elas não conseguem avançar com facilidade, a maioria delas solicita mudar de jogo sob a premissa de que o jogo é chato, quando na verdade querem expressar que consideram o jogo difícil. E, embora as crianças expressassem que achavam o jogo chato nos momentos em que se depararam com atividades mais difíceis, não solicitaram mudar de jogo.
1	Gomes *et al.* (2015)	Com as duas avaliações tornou-se possível a observação de diferentes aspectos referentes ao jogo The Foos e sua forma de ensino do **pensamento computacional** para crianças. Enquanto com a avaliação formativa, foi possível avaliar especificamente a resposta do público-alvo em relação às funcionalidades do jogo, na avaliação objetiva foi feita uma análise do jogo de acordo com 10 parâmetros que abordaram vários critérios de avaliação que vão desde facilidade de uso até mesmo afetividade e motivação.

(continua)

Tabela 26 – Textos do WAlgProg (2015-2017) que usam o termo "pensamento computacional" no máximo dez vezes

(continuação)

quant.	autor	parágrafo
1	Lima e Sousa (2015a)	Resumo: O presente artigo relata a experiência de um bolsista do Programa Institucional de Bolsas de Iniciação à Docência por meio de um curso, no qual se trabalhou a introdução ao raciocínio lógico e algoritmo com alunos do 5º ano do ensino fundamental de uma escola pública. Buscou-se alcançar a compreensão do conteúdo ministrado, assim como a construção do conhecimento dos alunos no âmbito do **pensamento computacional** por meio da utilização dos computadores da sala de informática da escola com auxílio da pseudo-linguagem "Portugol IDE" e de forma lúdica utilizando recursos didático-pedagógicos como objetos de aprendizagem.
1	Lima e Sousa (2015a)	O objetivo do artigo é apresentar a experiência dos bolsistas do PIBID através do curso onde foi trabalhado o **pensamento computacional**, a introdução ao raciocínio lógico, os conceitos de algoritmo, estrutura sequencial e estrutura de seleção, ministrado para alunos da educação básica do 5º ano do ensino fundamental de uma escola pública. A sala de informática da escola foi utilizada para o desenvolvimento do curso, para execução de atividades com objetos de aprendizagem e construção dos algoritmos.
2	Lima e Sousa (2015a)	Uma proposta metodológica para a atividade com Números Binários em que tem como objetivo desenvolver habilidades necessárias para prática do **Pensamento Computacional** em crianças do quarto ano do Ensino Fundamental é apresentada por [Campos et al. 2014]. Os autores afirmam que as escolas do ensino fundamental são veículos relevantes para implantação do uso do PC (**pensamento computacional**), no sentido de incentivar habilidades para desenvolvimento pleno e efetivo do raciocínio lógico-dedutível, principalmente nos primeiros anos escolares.
2	Lima e Sousa (2015a)	Existem outros trabalhos que estimulam o **pensamento computacional** na educação básica, como a "Computação Desplugada". Sem a utilização do computador, [Bezerra 2014] em sua experiência em uma escola pública da Região Metropolitana de Belém através do PIBID da Universidade Federal Rural da Amazônia (UFRA) do curso de Licenciatura em Computação, propôs o ensino da Computação Desplugada no ensino fundamental, explorando também o universo dos algoritmos, enfatizando ser a parte fundamental da computação, por serem os algoritmos que "dizem" como os computadores devem trabalhar. O autor relata em sua experiência que o contato com a computação desde o ensino fundamental é de grande importância por permitir o desenvolvimento do **pensamento computacional**, contribuindo para a possibilidade de futuros profissionais na área de computação.

(continua)

Tabela 26 – Textos do WAlgProg (2015-2017) que usam o termo "pensamento computacional" no máximo dez vezes

(continuação)

quant.	autor	parágrafo
1	Lima e Sousa (2015a)	A importância do ensino de Computação na Educação Básica é evidenciada em [França et al. 2012], decorrente das atividades desenvolvidas nos Estágios Curriculares Obrigatórios no curso de Licenciatura em Computação da Universidade de Pernambuco (UPE), no ano de 2011. A metodologia, bem como os resultados obtidos, foram baseados na realização de atividades lúdicas para facilitar o processo de ensino-aprendizagem, através da Computação Unplugged com a finalidade de promover a disseminação do **pensamento computacional** na educação básica, desenvolvendo as atividades com a implementação de algoritmos. Utilizaram a ferramenta Scratch composta por uma linguagem gráfica de programação, que possibilita a criação de histórias interativas, animações, jogos e o compartilhamento dessas criações na Web, envolvendo os alunos do 9º ano do Ensino Fundamental da Escola de Aplicação Professora Ivonita Alves Guerra da Universidade de Pernambuco, localizada no agreste meridional do estado de Pernambuco, na cidade de Garanhuns.
1	Lima e Sousa (2015a)	Apesar das dificuldades encontradas pelos bolsistas em adaptar as aulas, conseguiu-se através do ensino da lógica e algoritmo, assim como na construção do **pensamento computacional**, que os alunos pudessem no final do curso adquirir notas melhores, resultando um bom desempenho na 2ª Avaliação, principalmente nas matérias de lógica e matemática.
1	Lima e Sousa (2015a)	Para licenciados em computação ainda são grandes os desafios a enfrentar. Muitas escolas principalmente as públicas, os responsáveis pelos laboratórios de informática não realizam atividades que possam contribuir para o crescimento do rendimento escolar dos alunos e muito menos para disseminação do **pensamento computacional**, pois basicamente não são conhecedores da área, deixando de utilizar um espaço que pode ser de tanto valor para o processo de ensino e de aprendizagem. A sala de informática da escola que se realizou o curso, por exemplo, a maioria dos computadores estava parados por falta de manutenção e por estarem muito tempo sem serem utilizados, sendo necessário que os próprios bolsistas do PIBID formassem um mutirão para concertá-los para que fosse possível realizar as atividades na escola.
1	Raabe, Zanchett e Vahldick (2015)	Foram testados três jogos de programar com diferentes enredos e notações de programação, buscando conhecer a opinião dos estudantes sobre: (i) jogo preferido, (ii) engajamento e diversão, (iii) estilo tutorial (forma de propor os problemas aos jogadores); e (iv) percepção sobre programação e conceitos envolvidos. A compreensão sobre a forma como estes jogos são percebidos pelos estudantes auxilia no planejamento de atividades de introdução ao **pensamento computacional**, possibilitando também delinear diretrizes para a concepção e o projeto de novos jogos.
1	Raabe, Zanchett e Vahldick (2015)	Os sujeitos participantes da pesquisa foram 14 estudantes (6 meninas e 8 meninos) que fazem parte de um projeto de extensão voltado à introdução do **pensamento computacional** envolvendo ensino de programação e robótica. Os estudantes são do 1º e 2º anos do ensino médio de uma escola pública e foram selecionados pela direção da escola devido ao bom desempenho acadêmico.

(continua)

Tabela 26 – Textos do WAlgProg (2015-2017) que usam o termo "pensamento computacional" no máximo dez vezes

(continuação)

quant.	autor	parágrafo
1	Souza e Castro (2016)	Nesse contexto, o ensino de conceitos de programação ainda na educação básica pode ser um grande facilitador para o processo de aprendizagem. Várias iniciativas vem sendo tomadas para que o **pensamento computacional** e o raciocínio lógico sejam desenvolvidos desde cedo, uma delas é o uso de ferramentas para o auxílio do ensino desses conceitos. A ferramenta mais utilizada para o ensino e pesquisa atualmente é o Scratch[1].
1	Andrade *et al.* (2016)	São encontradas na literatura algumas propostas para ensinar a desenvolver jogos. Porém, dentre elas são identificados diferentes objetivos para a utilização dessa prática. Andrade (2013) propõe a criação de jogos com intuito de abordar conteúdos de matemática no ensino médio. Ainda com esse público alvo, Rebouças et al (2010) propõem formas de aumentar o interesse dos alunos pela área de informática e Scaico et al (2012) propõe desenvolver jogos para trabalhar o ensino de programação, assim como estimular o **Pensamento Computacional**.
1	Andrade *et al.* (2016)	Na literatura são encontradas diferentes propostas nas quais o desenvolvimento de jogos é abordado para finalidades pedagógicas. O trabalho de Andrade et al. (2013) teve como objetivo promover o ensino da matemática por meio do desenvolvimento de jogos usando o Scratch. No mesmo contexto, Barcelos e Silveira (2013) realizaram atividades práticas de desenvolvimento de jogos digitas e investigaram habilidades do **Pensamento Computacional** relacionadas à matemática.
1	Andrade *et al.* (2016)	Foi observado que parte dos alunos, por mais que tenham se interessado pela programação, não tem vontade de seguir na área da TI. Entretanto, as habilidades trabalhadas nessa oficina podem complementar a formação dos alunos, independente da carreira que venham a seguir. A metodologia desenvolvida na oficina permitiu aos alunos estimular habilidades do **Pensamento Computacional** relacionadas à construção de algoritmos, abstração, simulação, paralelismo e automatização de soluções.
1	Andrade *et al.* (2016)	Como trabalhos futuros, no tocante à programação, planejamos inserir recomendações de boas práticas de organização e reutilização de código. Também pesquisaremos aspectos para avaliar a documentação produzida pelos alunos (storyboard e diagrama de elementos do jogo). Além disso, avaliaremos outros motores de jogos que permitam explorar mais funcionalidades para produção de jogos. Por último, investigaremos como habilidades do **Pensamento Computacional** podem ser estimuladas e avaliadas através de oficinas de desenvolvimento de jogos.
1	Oliveira e Barbosa (2016)	Realizando uma pesquisa apenas nos anais do Workshop de Ensino em **Pensamento Computacional**, Algoritmos e Programação do ano de 2015, é possível encontrar 6 trabalhos (16% dos trabalhos publicados), tais como, [Aguiar 2015] e [Raabe et al. 2015], relacionados ao uso de jogos ou a gameficação em ambientes de ensino. Desta forma, observando o crescente interesse em pesquisas relacionadas ao uso de jogos na área de educação, nos motivou a formulação da seguinte questão de pesquisa "um perfil de jogador pode ser utilizado em um ambiente educacional?". Caso um perfil de jogador seja adequado para estes ambientes, todo o esforço da área de identificação de perfis de jogadores pode ser transportado para contextos educacionais.

(continua)

Tabela 26 – Textos do WAlgProg (2015-2017) que usam o termo "pensamento computacional" no máximo dez vezes

(continuação)

quant.	autor	parágrafo
1	Queiroz, Sampaio e Santos (2016)	Muito dessa dificuldade existe em decorrência da não introdução da Ciência da Computação como parte das ciências básicas, o que faz com que ela normalmente não seja abordada nas séries fundamentais [Barcelos 2012]. Por outro lado, diferentes autores como Mitchel Resnick (2009), vêm defendendo a necessidade de incorporar o ensino de programação já nos primeiros anos do Ensino Fundamental (K-12 nos EUA), como forma de desenvolver, nos alunos, competências e habilidades relacionadas ao **Pensamento Computacional** [Wing 2006]
1	Queiroz, Sampaio e Santos (2016)	Para Hemmendinger (2010), o objetivo do desenvolvimento do **Pensamento Computacional** não é o de fazer com que todos passem a pensar como cientistas da computação, mas sim, habilitar as pessoas a aplicarem esta maneira específica de raciocinar na busca por novos questionamentos e na solução de diversos tipos de problemas nas mais variadas áreas do conhecimento [Barr & Stepherson 2011].
1	Queiroz, Sampaio e Santos (2016)	Os resultados parciais aqui apresentados já nos permitem afirmar que a utilização do DB4K, em associação com as atividades didáticas e materiais de robótica desenvolvidos e utilizados nesta pesquisa, é um caminho viável para o desenvolvimento de algumas habilidades relacionadas ao **pensamento computacional** em crianças dos anos iniciais do Ensino Fundamental I por meio da programação de computadores via Robótica Educacional.
1	Vera *et al.* (2016)	Freire (1996) lembra que a abordagem didática deve considerar a realidade dos estudantes, incentivando-os a se apropriarem do conhecimento e desenvolver a curiosidade epistemológica. É essa forma especial de curiosidade que direciona o estudante a tentar conhecer e reconhecer os elementos do mundo, sintoniza -se com um dos elementos mais atuais de discussão sobre o ensino de computação: o raciocínio **(ou pensamento) computacional**.
1	Silva, Souza e Silva (2016)	Resumo. Iniciativas que buscam incluir o estudo de lógica de programação nas séries iniciais faz com que professores e especialistas em educação ampliem a aplicação de ferramentas que permitam aos estudantes alcançar os benefícios obtidos com o aprendizado de programação. Este trabalho apresenta uma experiência de aplicação da ferramenta Scratch para o ensino de programação a 26 alunos do 5º. ano do Ensino Fundamental I. Foram realizadas oficinas compostas de aulas expositivas e práticas baseadas na elaboração e desenvolvimento de um quiz sobre meio ambiente, e da exposição dos projetos para a comunidade escolar. Como principais resultados, destacamos a imersão do **pensamento computacional** a esse grupo de participantes e a disseminação do ensino de programação na escola.
1	Silva, Souza e Silva (2016)	É importante ressaltar que, encontrou-se na escola um ambiente favorável para a realização deste trabalho, pois, a mesma possui em seu Plano Pedagógico um pilar destinado à inovação, com o ensino de robótica educativa, sendo daí selecionados os alunos que participaram do projeto para aplicar com o Scratch o **pensamento computacional** e o ensino de programação na escola.

(continua)

Tabela 26 – Textos do WAlgProg (2015-2017) que usam o termo "pensamento computacional" no máximo dez vezes

(continuação)

quant.	autor	parágrafo
1	Timmermann e González (2016)	Seu conteúdo está diretamente relacionado com o desenvolvimento do **pensamento lógico ou computacional** (GIRAFFA, MULLER e MORAES, 2015) necessário para a elaboração de soluções algorítmicas que, mais tarde ou em concomitante, poderão se tornar programas escritos em diversas linguagens de programação. Além disso, conforme as Diretrizes Curriculares Nacionais dos cursos de Computação, na seção 3 – Projetos Pedagógicos, Organização do Curso e Conteúdos Curriculares, esta explicito que o estudante deve desenvolver a capacidade de compreender os problemas que motivam a escrita de tais soluções automatizadas. Ao mesmo tempo em que se evidência a relevância da aprendizagem de seu conteúdo, estudos revelam que o componente curricular Algoritmos constitui um dos grandes problemas enfrentados pelos alunos da Computação. Conforme Barcelos, Tarouco, Bercht (2009, p.1) seu índice de reprovação é um dos maiores entre as demais disciplinas destes cursos.
1	Ferreira *et al.* (2016)	Araújo, Andrade e Guerrero (2015) afirmam que o **pensamento computacional** é um conjunto de conceitos, habilidades e práticas da computação que podem ser aplicados tanto em atividades do cotidiano como em outras áreas do conhecimento.
1	Ferreira *et al.* (2016)	Embora muitas literaturas refiram-se a esse conjunto de habilidades como **pensamento computacional**, Paiva et al. (2015) e Matos, Paiva e Corlett (2016) apresentam discussões acerca do termo e afirmam que tais características estão relacionadas ao pensamento analítico, que por conseguinte fariam parte do que chamam de raciocínio computacional. De acordo com Ribeiro et al. (2013), o raciocínio computacional pode ser utilizado para resolver problemas de naturezas diversas e em diferentes contextos, não ficando restrito apenas aos problemas matemáticos.
1	Anjos Jr. *et al.* (2017)	Com base nisto, a implementação do d**pensamento computacional** em estudantes do EM poderia representar uma nova forma de se desenvolver, neste público, competências e habilidades que facilitassem a construção de soluções de problemas nas mais variadas áreas do conhecimento. Entretanto,formação do adolescente no ensino de lógica e da programação computacional se apresenta como um desafio, visto que há uma grande complexidade nos termos que circundam este conteúdo [Barr and Stephenson 2011].
1	Silva *et al.* (2017)	Resumo. O ensino do **pensamento computacional** tem sido de grande importância na expansão das capacidades cognitivas dos alunos em diferentes etapas de seu desenvolvimento. O presente artigo relata a experiência de discentes do curso de Licenciatura em Computação, intitulada "Oficina de Raciocínio Lógico e Programação", com objetivo de auxiliar os alunos do 8º e 9º ano na Olimpíada Brasileira de Matemática das Escolas Públicas (OBMEP), e introduzir a Computação na Educação Básica. Como resultado notou-se nos alunos uma mudança positiva quanto ao raciocínio e à capacidade de solucionar problemas, além do aumento do interesse e curiosidade para as áreas relacionadas à Computação.

(continua)

Tabela 26 – Textos do WAlgProg (2015-2017) que usam o termo "pensamento computacional" no máximo dez vezes

(continuação)

quant.	autor	parágrafo
1	Silva *et al.* (2017)	O ensino da lógica surge como uma maneira de exercitar o conhecimento dos alunos em matemática e estimular o **pensamento computacional**. A representação algorítmica de um determinado problema, aliada à narração verbal, pode ocasionar "uma transição mais 'suave' para a compreensão da linguagem matemática" [BARCELLOS e SILVEIRA 2012], trabalhando a interpretação de texto e conhecimentos técnicos da área da computação.
1	Silva *et al.* (2017)	O **pensamento computacional** envolve o uso do computador como um recurso capaz de ampliar as capacidades cognitivas humanas, bem como a criatividade, a produtividade, a operacionalidade e a inventividade França et al. [2012]. Mas, também pode ser trabalhado através de representações nas quais associa-se o funcionamento do processamento de um computador às práticas cotidianas, como ocorre na Computação Desplugada.
1	Silva *et al.* (2017)	A turma do 8º ano, apesar de não ter realizado o teste final, demonstrou-se bastante participativa durante as atividades, com melhora significativa na escrita, na formulação de respostas às questões subjetivas e na interpretação de textos. Obteve, assim como a turma do 9º ano, uma mudança positiva em resolução de expressões numéricas e desafios de lógica, bem como no modo de ver a Computação e o raciocínio lógico no dia a dia, reconhecendo a importância de estudar lógica nas escolas e de utilizar o **pensamento computacional** em situações diversas.
1	Silva *et al.* (2017)	O artigo apresentou a experiência realizada no Estágio Supervisionado Obrigatório do Curso de Licenciatura em Computação, através do **pensamento computacional** para auxiliar os alunos da Educação Básica na OBMEP. Após a olimpíada, não foi possível obter as notas para verificar se houve bom rendimento por meio das intervenções e comparar com as edições anteriores.
1	Silva (2017)	A aprendizagem de programação, por exemplo, no ensino médio pode não apenas atrair profissionais com aptidão e talento para a computação, como também ajudar estudantes de outras áreas em sua educação ou em suas futuras profissões. Nunes (2011) exemplifica a situação dos advogados que, na medida em que leem um texto, usam o **pensamento computacional** para extrair dele fatos e regras que justifiquem um parecer conclusivo. Outros procedimentos, como organizar processos, também podem ser apresentados de forma algorítmica.
1	Silva (2017)	De acordo com os resultados evidenciados, fica claro que a aplicação e o desenvolvimento de jogos pelos próprios alunos é um importante meio de auxílio ao processo de ensino e aprendizagem. Além desse modo de aprendizado fazendo com que os alunos solucionem o problema – o desenvolvimento do jogo – envolver alunos e professores em atividades que estimulam conhecimento sobre o **pensamento computacional** e uso de jogos digitais no ensino básico, pode servir como fator motivacional para os alunos aprenderem outros conteúdos.
1	Souto e Tedesco (2017)	Workshop de Ensino em **Pensamento Computacional**,
1	Souto e Tedesco (2017)	Artigos que investigam competências relacionadas ao **pensamento computacional** desenvolvidas durante a aprendizagem de programação

(continua)

Tabela 26 – Textos do WAlgProg (2015-2017) que usam o termo "pensamento computacional" no máximo dez vezes

(continuação)

quant.	autor	parágrafo
1	Avila e Cavalheiro (2017)	Este artigo apresenta uma proposta de metodologia para engajamento gradativo de aprendizes em conceitos relacionados à robótica educacional. A robótica é utilizada como estratégia para disseminação de conceitos relacionados ao **Pensamento Computacional**. A metodologia também incorpora duas taxonomias de aprendizagem. A primeira, uma adaptação da taxonomia de Bloom revisada, orienta o nível de exigência cognitiva dos conteúdos e atividades que serão realizadas em futuras intervenções baseadas na metodologia. A outra, conhecida por taxonomia SOLO, é utilizada para mensurar o nível de entendimento estrutural dos alunos em relação às respostas de exercícios/atividades.
1	Avila e Cavalheiro (2017)	A criação do termo "**Pensamento Computacional**" (PC) é comumente atribuído à pesquisadora Jeannette Wing [Barr et al. 2011, Lye and Koh 2014, Perkovìc et al. 2010]. Certamente em função do artigo escrito por [Wing 2006] onde ela introduz PC como um processo que envolve a "resolução de problemas, a capacidade de projetar sistemas e a compreensão do comportamento humano recorrendo aos conceitos fundamentais da Ciência da Computação".
1	Barcelos *et al.* (2017a)	No entanto, é necessário destacar que essas técnicas ainda não estão muito populares no ramo de análise do aprendizado. No contexto do desenvolvimento do **Pensamento Computacional**, atividades didáticas que envolvem a colaboração e a construção de artefatos são comuns (RODE et al., 2015); ainda, muitas vezes as atividades propostas demandam dos alunos o desenvolvimento de competências e habilidades relacionadas à programação. A análise automatizada de dados provenientes da fala de alunos que descrevem suas estratégias cognitivas para a resolução de problemas tem se mostrado como uma estratégia promissora, dado que trabalhos anteriores (HAIDER et al., 2016; WORSLEY; BLIKSTEIN, 2010b) mostraram ser possível inferir o estado emocional do entrevistado bem como suas dúvidas e incertezas. Dessa forma, este trabalho tem como objetivo apresentar, dentro da perspectiva de uma Análise Multimodal da Aprendizagem, os resultados preliminares de um experimento com alunos de cursos na área de Computação com diferentes níveis de experiência em programação onde as ocorrências de emoções na fala foram correlacionadas com o nível de experiência prévio dos participantes.
1	Barcelos *et al.* (2017a)	Considerando especificamente o desenvolvimento do **Pensamento Computacional** e de competências relacionadas à programação, as emoções identificadas durante o processo de aprendizagem podem trazer indícios de uma maior ou menor efetividade desse processo. Bosch e D'Mello (2017) reportam um estudo com 99 estudantes que tiveram sua primeira experiência com programação de computadores registrada em vídeo e áudio; suas conclusões indicam que a transição entre estados de engajamento e confusão, e entre estados de tédio e engajamento estavam, respectivamente, relacionados a eventos inesperados ("bugs" no código) e orientação do instrutor para sanar dúvidas. No entanto, no estudo os estados emocionais foram identificados a posteriori por meio da rotulação de momentos das gravações pelos próprios participantes.

(continua)

Tabela 26 – Textos do WAlgProg (2015-2017) que usam o termo "pensamento computacional" no máximo dez vezes

(continuação)

quant.	autor	parágrafo
2	Barcelos *et al.* (2017a)	É importante destacar que Bosch e D'Mello (2017) postulam que o estado emocional de engajamento relacionado ao "fluxo" (flow) (CSIKSZENTMIHALYI, 1990) seria o ideal em um processo de aprendizagem de programação. No tocante ao desenvolvimento do **Pensamento Computacional**, Basawapatna et al. (2013) também trabalham com essa mesma hipótese na proposta de um modelo para aprendizagem escalonada. Por outro lado, Brennan e Resnick (2012) indicaram que a análise do discurso de aprendizes relacionado às estratégias de construção de artefatos computacionais pode trazer evidências do desenvolvimento do **Pensamento Computacional**. Dessa forma, esta pesquisa se situa na confluência entre os trabalhos anteriores que buscaram a análise automatizada do discurso, porém com o enfoque na identificação automatizada das emoções expressadas durante uma atividade de programação de computadores.
1	Barcelos *et al.* (2017a)	O desenvolvimento de competências e habilidades relacionadas ao **Pensamento Computacional** e à programação de computadores envolve cada vez mais atividades colaborativas e complexas, o que cria novos desafios à avaliação. Técnicas relacionadas a um paradigma de Analítica Multimodal da Aprendizagem, como a análise automatizada do discurso, podem permitir a obtenção e análise de mais fatos relevantes relacionados ao processo de aprendizagem.

Fonte: Autoria própria

APÊNDICE B – PENSAMENTO COMPUTACIONAL: TEXTOS DO WALGPROG (2015-2017) QUE USAM O TERMO “PENSAMENTO COMPUTACIONAL” DE 11 A 30 OCORRÊNCIAS

Tabela 27 – Textos do WAlgProg (2015-2017) que usam o termo “pensamento computacional” entre onze e vinte e nove vezes

quant.	autor	parágrafo
1	Zanetti e Oliveira (2015)	Resumo. Este artigo discute sobre uma prática pedagógica para ensino de programação de computadores por meio de Robótica Pedagógica e Pensamento Computacional. Tem como objetivo trazer uma proposta de ensino que possa amenizar as principais dificuldades de alunos iniciantes em programação, auxiliando nos aspectos relacionados a construção da solução de problemas.
2	Zanetti e Oliveira (2015)	Em relação ao desenvolvimento do raciocínio computacional (algorítmico) necessário aos alunos em disciplinas de programação, é relevante adotar um método no qual seja possível apresentar e traduzir as noções de algoritmo dentro da disciplina. A capacidade de abstração é algo fundamental para o sucesso na aprendizagem de programação, principalmente para compreender problemas e propor soluções. A adoção do Pensamento Computacional pode orientar de forma ampla a atividade mental de abstrair problemas e formular soluções descritas em algoritmos. O termo Pensamento Computacional (em inglês Computational Thinking) foi introduzido por Wing (2008), com o objetivo de englobar desde a estruturação do raciocínio lógico, até o comportamento humano para a ação de resolução de problemas.
2	Zanetti e Oliveira (2015)	Assim, este trabalho traz uma proposta metodológica que delineia um contexto de aprendizagem significativa de programação de computadores suportada por uma linguagem de programação visual para o desenvolvimento do Pensamento Computacional e aliada a ferramentas de RP. Também apresenta um estudo realizado através de uma oficina e prática com alunos matriculados em um curso de ensino médio integrado ao técnico de Informática, utilizando uma ferramenta de programação visual S4A (Scratch 4 Arduino) e um robô controlado pela plataforma eletrônica Arduino. Além do estudo, o artigo ainda apresenta uma revisão bibliográfica sobre o uso de RP e Pensamento Computacional no contexto do ensino de programação de computadores.
1	Zanetti e Oliveira (2015)	O artigo apresenta na Seção 2 uma revisão bibliográfica sobre os desafios no ensino de programação e apresenta uma breve descrição sobre RP e suas aplicações; na Seção 3 é discutida a abordagem pedagógica baseada em Pensamento Computacional para resolução de problemas; a Seção 4 apresenta a prática e maiores detalhes da oficina com os alunos; a Seção 5 apresenta a análise dos resultados; e por fim, na Seção 6 são apresentadas as considerações finais.
1	Zanetti e Oliveira (2015)	3. Pensamento Computacional como modelo de solução de problemas

(continua)

Tabela 27 – Textos do WAlgProg (2015-2017) que usam o termo "pensamento computacional" entre onze e vinte e nove

(continuação)

quant.	autor	parágrafo
1	Zanetti e Oliveira (2015)	Disciplinas como a programação de computadores ou Lógica de Programação permite realizar um encadeamento coerente de uma sequência lógica de instruções para resolução de problemas, constituindo um dos saberes elementares para desenvolver o Pensamento Computacional. Como supracitado, usualmente o conteúdo relacionado à disciplina é ensinado de forma isolada, não se preocupando com a contextualização dessas instruções ou a correlação com outras áreas de conhecimento. Nesse sentido, o aluno não compreende a aplicabilidade do conhecimento que é ensinado, tornando o processo de aprendizagem não siginificativo.
1	Zanetti e Oliveira (2015)	O Pensamento Computacional não se limita às ferramentas, como se fosse totalmente dependente das tecnologias [de Paula, Valente e Burn 2014]. Está mais próximo a um pensamento analítico, que compartilha características com o pensamento matemático (habilidades necessárias para soluções de problemas), pensamento sistêmico (projetar e avaliar sistemas complexos que operam sob restrições específicas) e pensamento científico (conhecimento teórico sobre o assunto, análise de resultados e senso crítico). Portanto, se trata de uma maneira específica de se pensar e de se analisar uma situação ou artefato, sendo influenciado por todas as áreas de conhecimento e diferentes tecnologias [Wing 2008; de Paula, Valente e Burn 2014].
2	Zanetti e Oliveira (2015)	Nesse contexto, é válido afirmar que o Pensamento Computacional é altamente requerido no repertório de habilidades dos alunos em disciplinas relacionadas à programação de computadores. O termo "computacional" não pode ser compreendido como sinônimo de "programação", mas sim como uma maneira de se compreender os sistemas computacionais, como eles funcionam e como são projetados e programados. A programação não é o fim, mas sim um dos meios para se atingir os objetivos [de Paula, Valente e Burn 2014]. Estudiosos na década de 1980, como Seymour Papert [Papert 1985], já afirmavam que era necessário promover o desenvolvimento do Pensamento Computacional junto aos alunos, argumentando sobre o pensamento analítico (chamado em seu trabalho de "pensamento procedimental"), utilizando a linguagem de programação Logo, que precedeu várias ferramentas pedagógicas de ensino de programação, incluindo ferramentas de RP.
1	Zanetti e Oliveira (2015)	Adotando esse argumento, nota-se que o Pensamento Computacional direciona a aprendizagem para algo mais amplo, o que favorece a apresentação de toda estrutura curricular e a abrangência dos saberes necessários nas disciplinas de programação de computadores. Se alinhado com o ferramental da RP, torna-se possível através desse meio menos abstrato que o tradicional, estimular a construção do conhecimento e o engajamento dos alunos, como uma maneira de se fomentar o raciocínio lógico, a habilidade cognitiva e a prática de solução de problemas. Também é importante ressaltar o argumento Construcionista atrelado à RP, que de acordo com essa teoria, o processo de aprendizagem é mais significativo e profundo, já que os alunos podem trabalham em projetos e desafios que lhes são pessoalmente mais atrativos e, consequentemente, mais significativos [Rusk et al. 2008].

(continua)

Tabela 27 – Textos do WAlgProg (2015-2017) que usam o termo "pensamento computacional" entre onze e vinte e nove

(continuação)

quant.	autor	parágrafo
2	Zanetti e Oliveira (2015)	Para esse trabalho foi realizado uma oficina e, posteriormente, uma prática com um grupo de alunos com o intuito de analisar o uso de RP e, para promover e estimular o Pensamento Computacional, direcionando a resolução de problemas, construção de conhecimento e autoavaliação. Há quatro objetivos bem definidos alinhados às seguintes dificuldades comumente encontradas em alunos iniciantes em programação: (i) avaliar o próprio conhecimento; (ii) compreender e interpretar problemas; (iii) resolver problemas aplicando o Pensamento Computacional e; (iv) depuração de erros e correção dos mesmos.
1	Zanetti e Oliveira (2015)	Na prática adotada nesse trabalho, o aluno deve resolver o problema proposto refletindo sobre quais são as ações corretas, quais estruturas de programação são necessárias. E quando surgir problemas, o aluno pode testar e depurar um novo procedimento em busca de um resultado melhor. O termo "depurar" ou "depuração" relacionado ao problema se baseia em Valente (2005), que afirma que a depuração implica uma nova descrição da solução e, assim, sucessivamente, repetindo o ciclo "descrição-execução-reflexão-depuração-descrição", como vemos na Figura 4. O uso desse "ciclo de depuração" alinha-se com as diretrizes fundamentais do Pensamento Computacional, e é facilitada pela presença do ambiente de programação e do robô. Tais instrumentos facilitam a análise do programa de modo que o aluno possa achar seus erros. Este processo de achar e corrigir o erro constitui em uma oportunidade única para o aluno aprender sobre um determinado conceito envolvido na solução do problema ou sobre estratégias de resolução de problemas, além de poder, através da programação, relacionar seu pensamento em um nível metacognitivo [Valente, 2005].
1	Zanetti e Oliveira (2015)	Este trabalho teve como objetivo mostrar a potencialidade do uso de ferramentas da RP aliadas ao Pensamento Computacional para resolução de problemas e aprendizagem de conceitos de programação de computadores. A prática proposta apresenta uma possibilidade de obter um meio menos abstrato e mais motivador do que o modelo tradicional comumente apresentado em cursos de Computação. Com esse novo cenário de ensino é possível engajar o aluno como elemento ativo na construção do conhecimento e no fomento do raciocínio lógico.
1	Zanetti e Oliveira (2015)	Entretanto, a prática apresentada não esgota o problema relacionado ao ensino de programação de computadores em absoluto, mas contribui na exploração do ferramental disponível na RP e a construção cognitiva orientada ao Pensamento Computacional. Como trabalho futuro é esperado a adição de novos elementos que possam contribuir com a abstração e a solução de problemas, assim como meios eficazes de avaliar o desempenho dos alunos na prática pedagógica. Para tanto, será realizado experimentos com um grupo maior e mais heterogêneo de alunos.
1	Mestre *et al.* (2015)	Pensamento Computacional nas Escolas: Limitado pela Tecnologia, Infraestrutura ou Prática Docente?

(continua)

Tabela 27 – Textos do WAlgProg (2015-2017) que usam o termo "pensamento computacional" entre onze e vinte e nove

(continuação)

quant.	autor	parágrafo
1	Mestre *et al.* (2015)	A informática converge para inovações e automatização de processos na resolução de problemas aplicada a quaisquer áreas. O Pensamento Computacional incorpora inovações tecnológicas no meio pedagógico e promove o dinamismo nas técnicas e métodos tradicionais. O artigo identificou as demandas das escolas municipais de Francisco Beltrão – Paraná, nos aspectos que envolvem o uso da tecnologia como meio de transmissão e aquisição de conhecimento, seja por parte do aprendiz ou do professor. Os resultados mostram reconstrução dos processos de ensino e aprendizagem. A intervenção de acadêmicos da Licenciatura em Informática é necessária no exercício docente relacionado à Computação nas escolas municipais.
1	Mestre *et al.* (2015)	A explicação para a resistência dos professores ao uso da tecnologia pode estar fundamentada no paradigma dos currículos tradicionalistas. Trata-se de uma vertente contrária à vaga curricular para professor graduado em Licenciatura em Informática. Desta maneira, o conceito de Pensamento Computacional, proposto por Wing (2006), emergiu como incentivador aos recursos tecnológicos utilizados por professores das escolas públicas municipais da região de Francisco Beltrão – Paraná, que não tem a formação tecnológica inserida em seus currículos.
1	Mestre *et al.* (2015)	O objetivo deste trabalho foi identificar e mapear itens presentes no espaço de intervenção para compreender se a aplicação da proposta do Pensamento Computacional nas escolas municipais está limitada à tecnologia ou ao currículo dos professores. Para isso, os alunos da disciplina de Didática Aplicada à Informática estruturaram um survey que foi aplicado em escolas municipais de Francisco Beltrão – Paraná. Esta atividade fez parte dos Estágios I, II e III de experimentação à Prática Docente dos acadêmicos. Os resultados apontaram para um colegiado ainda não preparado e sem vistas de interesse tecnológico.
1	Mestre *et al.* (2015)	2. Pensamento Computacional e o Curso de Licenciatura em Informática
2	Mestre *et al.* (2015)	Pensamento Computacional é um tema que atrai pesquisadores e parece estar impactando no contexto escolar. Em 2006, Jeannette M. Wing, no artigo Computational Thinking, apresenta o conceito para Pensamento Computacional e a relevância de abordar tal assunto dentro do contexto educacional de forma tão elementar quanto o aprendizado da leitura, escrita ou de operações matemáticas. Deste ponto em diante, diversas pesquisas e aplicações do Pensamento Computacional foram estudadas e aplicadas.
1	Mestre *et al.* (2015)	O termo Pensamento Computacional, termo cunhado pela proposta de Jeannette Wing, está também associado às ideias de resolução de problemas, projeto de sistemas e compreensão do comportamento humano norteados por conceitos fundamentais da Ciência da Computação (Wing, 2006). Isto se revela como um método utilizado para solucionar problemas, conceber sistemas pelo tal como preconiza a Teoria Geral de Sistemas pela habilidade de abstrair e generalizar, e compreender o comportamento humano inspirado em conceitos fundamentais da Ciência da Computação.

(continua)

Tabela 27 – Textos do WAlgProg (2015-2017) que usam o termo "pensamento computacional" entre onze e vinte e nove

(continuação)

quant.	autor	parágrafo
1	Mestre *et al.* (2015)	Araujo, Andrade e Serey (2015) se posicionam a partir da definição dada pela International Society for Technology in Education (ISTE) e pela Computer Science Teachers Association (CSTA) como uma abordagem para resolução de problemas, incorporando processos mentais e ferramentas que utilizam habilidades como organização e análise de dados, construção de algoritmos, abstração, criação de modelos, simulação, automatização de soluções e paralelização, considerando como uma lista de habilidades. Nessa mesma perspectiva, França e Tedesco (2015) defendem a inserção do Pensamento Computacional desde a educação básica como forma de melhorar o aprendizado lógico em nível escolar dos alunos e possibilitar o uso mais eficaz dessas tecnologias móveis em benefício da sociedade.
1	Mestre *et al.* (2015)	É importante lembrar que o Pensamento Computacional não se fundamenta em saber navegar na Internet, acessar e-mails, editar um texto, utilizar planilhas eletrônicas, elaborar uma apresentação ou manipular um equipamento eletrônico. Sua importância está para o processo de resolução de problemas lógicos nos diversos contextos da sociedade, permitindo que se possa aplicar a Computação nas suas ações cotidianas(GOMES e MELO, 2013).
2	Mestre *et al.* (2015)	Segundo Mestre et. al. (2015), as habilidades estimuladas pelo Pensamento Computacional estão diretamente relacionadas à resolução de problemas, que envolvem as capacidades de ler, interpretar textos, compreender situações reais propostas e transpor informações para modelos matemáticos, científicos ou sociais. Para os autores Araújo, Andrade, Guerrero (2015), o Pensamento Computacional é um conjunto de conceitos, habilidades e práticas da Computação que podem ser aplicados tanto em atividades do cotidiano como em outras áreas do conhecimento. Uma abordagem de resolução de problemas incorporando processos mentais e ferramentas que utilizam habilidades, tais como organização e análise de dados, construção de algoritmos, abstração, decomposição, simulação, automatização e paralelização.
1	Mestre *et al.* (2015)	O Pensamento Computacional está centrado na perspectiva da resolução de problemas mediados ou não pelo uso computador. Essa abordagem se expande por todos os contextos na busca de soluções mais eficientes para resolver problemas e atividades complexas.
1	Mestre *et al.* (2015)	Nunes (2010) esclarece que o ensino de aplicativos, tais como editores de texto, planilhas de cálculo, linguagens de programação não influencia pensamentos de resolução de problemas na educação básica. Por exemplo, para datilografar não é necessário o uso de um editor de texto, nem calculadoras para o ensino da Matemática. O papel do professor está em tornar o aluno confiante e capaz de pensar computacionalmente. Para Blikstein (2008), o uso dos aparatos digitais (gadget) também é um facilitador a desenvolver a habilidade do Pensamento Computacional.

(continua)

Tabela 27 – Textos do WAlgProg (2015-2017) que usam o termo "pensamento computacional" entre onze e vinte e nove

(continuação)

quant.	autor	parágrafo
1	Mestre *et al.* (2015)	Os cursos de Licenciatura em Informática têm a responsabilidade de formar professores para disseminar o Pensamento Computacional, assim se pode confirmar a demanda em relação as competências e habilidades previamente definidas no Processo nº 23001.000026/2012-95 do tratado pelas Diretrizes Curriculares Nacionais para os cursos de graduação (bacharelado e licenciatura) em Computação necessárias para aquisição de conhecimentos de forma autônoma e efetiva (NUNES, 2011). O aluno deve atingir competências para reconhecer tarefas a serem realizadas de maneira mais rápida e eficientemente mediadas por computador. A partir dessas competências, o aluno também será capaz de desenvolver habilidades para programar o computador a realizar tais tarefas.
1	Mestre *et al.* (2015)	Para elucidar se o uso do Pensamento Computacional nas escolas municipais está limitado à tecnologia, infraestrutura ou ao currículo dos professores das escolas da rede pública municipal, utilizou-se do survey exploratório supervisionado, como metodologia de coleta de dados amplamente empregada por Bauer e Gaskell (2002).
1	Mestre *et al.* (2015)	No que tange à primeira dimensão, intitulada Infraestrutura e Equipamentos Informáticos, ver Figura 1, nota-se que somente 60% das escolas apresentam computadores em condições para a prática computacional. Este resultado permite aplicar o Pensamento Computacional uma vez que a infraestrutura se encontra coerente.
1	Mestre *et al.* (2015)	Para a segunda dimensão, intitulada Acesso à Internet nas Salas de Aulas, a Figura 2 apresenta um resultado que favorece a argumentação daqueles professores tradicionalistas que usam quadro e giz. Embora dados revelaram que mais de 50% das escolas não possuem acesso à Internet, ainda assim é possível reverter a situação e planejar ações que permitam explorar o Pensamento Computacional sem o uso dos artefatos computacionais.
1	Mestre *et al.* (2015)	Já a terceira dimensão, chamada de Ferramentas e Programas mais utilizados nas Salas de Informática, revelou que embora as escolas não possuírem acesso constante à Internet, alguns professores utilizaram do Pensamento Computacional no uso de softwares educativos e utilitários, conforme apresenta a Figura 3.
1	Mestre *et al.* (2015)	Por fim, a última dimensão, intitulada Internet vs. Atividades Escolares, apresenta dados que informam o desinteresse dos alunos frente às atividades desenvolvidas nas salas de informática com metodologias tradicionalistas, ver Figura 4. Estes dados também revelaram que os softwares educativos não estão sendo trabalhados de maneira se fazer presente nas atividades cotidianas dos alunos. Infelizmente infere-se que o uso das salas de informática está somente pela manipulação de objetos, sem um motivo ou aplicação direta ao Pensamento Computacional em seu cotidiano escolar.
1	Mestre *et al.* (2015)	Os resultados encontrados apontam evidências de atividades abordadas com pouca complexidade, tais como ler notícias, copiar conteúdos, visualizar mapas, desenhar, escrever com editores de texto, calcular com calculadora. A partir desses dados é possível afirmar que a forma como a tecnologia foi explorada por estas escolas não satisfaz a perspectiva do Pensamento Computacional.

(continua)

Tabela 27 – Textos do WAlgProg (2015-2017) que usam o termo "pensamento computacional" entre onze e vinte e nove

(continuação)

quant.	autor	parágrafo
2	Mestre *et al.* (2015)	O estudo abordou a questão fundamental sobre a preparação dos professores e das estruturas educacionais para a implementação do ensino de computação. Também descreveu os resultados obtidos com a aplicação de um survey em escolas públicas com o objetivo de compreender como o Pensamento Computacional está sendo tratado nas escolas e quais as limitações existentes. Basicamente, o survey contemplou as seguintes dimensões: infraestrutura, acesso à Internet, aplicativos e atividades escolares. Os resultados mostraram um corpo docente despreparado para atuar com a prática do pensamento computacional.
2	Mestre *et al.* (2015)	Com o objetivo de identificar se a aplicação do Pensamento Computacional estava limitada pela infraestrutura ou pelos professores, percebeu-se que todo o processo de ensino-aprendizagem precisa ser considerado. Embora se tenha espreitado apenas as atividades escolares que usam computador ou dispositivos móveis, é sabido que a aplicação do Pensamento Computacional vai além. Nessa vertente, o mapeamento nas escolas revelou que o esforço do governo em investir em formação de professores para uso de recursos tecnológicos é válido, mas parece frágil. Assim, cognição e comportamento podem ser relacionados, extraíndo as melhores características, com o intuito de revelar um método para solucionar problemas. De posse dessa hibridização, é possível conceber sistemas tal como preconiza a Teoria Geral de Sistemas e sua habilidade de abstrair, generalizar e compreender o comportamento humano inspirado em conceitos fundamentais da Ciência da Computação. No caso das escolas presentes neste estudo, ainda há muito aspectos a serem reforçados, validados pelos resultados encontrados na região Sudoeste do Paraná. Possivelmente seja a consequência do currículo tradicionalista destes professores em não pensar computacionalmente, de maneira lúdica ou ainda, da proposta não ter sido esclarecida junto às escolas, através de cursos de formação de professores, ou da demanda de um licenciado em informática estar presente nestes espaços.
1	Mestre *et al.* (2015)	Vale destacar que o uso dos recursos tecnológicos no ensino, não exige apenas seu manuseio, mas um interesse intrínseco em fazer a diferença na prática pedagógica reflexiva, uma vez que o uso de tablets e smartphones não garante, por si só, uma melhor qualidade do ensino. Em se tratando de tecnologia, a maneira como está sendo utilizada com os alunos mostra que é preciso uma grande intervenção dos profissionais de licenciatura para potencializar o uso do pensamento computacional (não necessariamente apenas com a internet, mas com outros recursos). Assim, a formação continuada de professores para a utilização de tecnologias afins pode vir a contribuir para o aprimoramento da prática educativa, se pautada pela compreensão das possibilidades e limites desse instrumento e na concretização do papel educativo escolar.

(continua)

Tabela 27 – Textos do WAlgProg (2015-2017) que usam o termo "pensamento computacional" entre onze e vinte e nove

(continuação)

quant.	autor	parágrafo
1	Mestre *et al.* (2015)	A análise dos gráficos também revelou um contexto escolar público bimodal. De um lado, professores e suas disciplinas específicas devem receber formação para explorar os recursos tecnológicos em sala de aula, uma vez que o governo não optou pela computação desplugada por Bell, Witten e Fellows (2011). De outro, os profissionais licenciados em informática buscam solidificar sua prática e sua participação no contexto escolar envoltos no Pensamento Computacional produto de uma formação preocupada com a pesquisa que emerge neste meio.
1	Paiva *et al.* (2015)	2. Raciocínio ou Pensamento Computacional? – Dosado com interdisciplinaridade
2	Paiva *et al.* (2015)	A disseminação do raciocínio computacional, por meio do pensamento computacional, vem crescendo entre os estudos relacionados ao ensino da CC. Por isso faz-se necessário estabelecer uma breve discussão acerca desses conceitos, para que possamos utilizá-los adequadamente. A diferenciação conceitual entre raciocínio computacional e pensamento computacional não está apenas nos termos utilizados, mas sobretudo nas diferenciadas compreensões acerca do tema.
2	Paiva *et al.* (2015)	Como podemos ver, a autora apresenta o que chama de computational thinking, como um conjunto de ferramentas mentais usado para raciocínio heurístico (no cotidiano, para além dos cientistas). A tradução livre de interpretação, reduz o conceito explicado por Wing ao senso comum. Neste artigo preferimos utilizar a expressão "raciocínio computacional", por considerá-lo mais adequado ao que de fato reconhecemos como computational thinking. Wing (2006) ao trazer algum exemplo explicando o que seria pensamento computacional, faz uso do termo "raciocínio" para citar algo específico da CC. A maioria dos autores que utilizam sua obra como base seguem a mesma ideia, como podemos ver em Ribeiro et al. (2013), quando apresentam um exemplo onde explica de forma detalhada a aplicação do algoritmo no cotidiano:
1	Paiva *et al.* (2015)	Os estudos de algoritmos envolvem conceitos como abstração, refinamento, a modularização, recursão, etc. Aprender esses conceitos melhora a capacidade de raciocínio e resolução de problemas por meio de processos de aprendizagem meta-cognitiva, considerado essencial para a inteligência.[...] Para ilustrar esta ideia, podemos pensar em um cenário computacional consiste em uma pessoa A, com sua linguagem LA, uma pessoa B, com sua linguagem LB, e uma máquina M (um computador, por exemplo) com o seu LM idioma. A recebe um problema P para resolver (por exemplo, para extrair a raiz quadrada de um número). Depois de analisar o problema, A escolhe uma máquina, por exemplo M, em que a solução pode ser adequadamente aplicado. A solução é então escrita numa linguagem LM compreensível pela máquina M. A solução do problema é conhecido como algoritmo. O processo de resolução de problemas (envolvendo análise de problemas, selecionando máquinas adequadas, construção e algoritmos de execução) é chamado pensamento computacional. (p. 22-23) (tradução nossa)

(continua)

Tabela 27 – Textos do WAlgProg (2015-2017) que usam o termo "pensamento computacional" entre onze e vinte e nove

(continuação)

quant.	autor	parágrafo
2	Paiva *et al.* (2015)	Podemos compreender que esse processo específico de resolução de um problema seria o raciocínio computacional, pois essas características apresentadas fazem parte do pensamento analítico, que consequentemente faz parte do pensamento computacional.
2	Paiva *et al.* (2015)	Portanto, neste texto, consideramos que ao tratar sobre pensamento computacional, o que estão sendo envolvidos de fato são os mecanismos e elementos epistemológicos de raciocínio computacional. Esse raciocínio se expressa quando o pensamento computacional está relacionado ao pensamento analítico e ao raciocínio inferencial (abdutivo, indutivo ou dedutivo) – com envolvimento da lógica e/ou da matemática. Dessa forma, podemos assimilar o raciocínio computacional como a capacidade de resolução de problemas de forma sistemática, usando inferências lógicas e abstrações, habilidades importantes para a CC.
3	Paiva *et al.* (2015)	Na área de Computação, algumas práticas interdisciplinares isoladas já têm sido realizadas, como aquelas que descrevem experiências de integração curricular envolvendo resolução de problemas no ensino superior (Pinto et al., 2010; Matos, 2013). Todavia, poucos são os relatos e/ou estudos de experiências interdisciplinares de integração curricular envolvendo a CC na educação básica. França et al. (2014) apresentam em seu trabalho a importância do ensino dos fundamentos da CC com outras áreas no intuito de disseminar o pensamento computacional na educação básica e pontua a promoção do pensamento computacional de modo interdisciplinar, ressaltando que "a interdisciplinaridade do pensamento computacional também tem sido considerada em ações que exacerbam a necessidade de conhecimentos em Computação, na educação básica." (França et al., 2014, p. 1510).
1	Araujo, Andrade e Serey (2015)	Pensamento Computacional (PC) consiste em uma abordagem de resolução de problemas que explora conceitos da computação. O cerne da questão encontrase nos processos mentais envolvidos na maneira como um profissional da computação age quando coloca em ação técnicas e práticas da Ciência da Computação para solucionar problemas. Neste trabalho, realizamos um survey com profissionais da computação para capturar a compreensão que eles têm de PC. Nossos resultados apontam que 64% dos pesquisados desconhecem o PC e sugerem que o termo e habilidades relevantes são pouco conhecidos e compreendidos pelos profissionais, tanto na indústria como na academia.
1	Araujo, Andrade e Serey (2015)	Pensamento Computacional (PC) foi apresentado em 2006 por Jeannette Wing associado às ideias de resolução de problemas, design de sistemas e compreensão do comportamento humano norteados por conceitos fundamentais da Ciência da Computação (Wing, 2006). Foi difundido que se trata de um conjunto de conceitos, habilidades e práticas da computação que podem ser aplicados tanto em atividades do cotidiano como em outras áreas do conhecimento. A importância do PC foi comparada às competências de ler, escrever e calcular.

(continua)

Tabela 27 – Textos do WAlgProg (2015-2017) que usam o termo "pensamento computacional" entre onze e vinte e nove

(continuação)

quant.	autor	parágrafo
1	Araujo, Andrade e Serey (2015)	A comunidade de Educação em Computação acolheu a ideia inicial e nos últimos anos inúmeros trabalhos foram publicados. Esses versam sobre propostas de atividades, relatos de experiências e pesquisas empíricas sobre adoção e avaliação do pensamento computacional nos diversos níveis de ensino (Mannila et al, 2014) (Walden et al, 2013) (Barr e Stephenson, 2009). Outros investigam a respeito da definição formal para o PC (Selby e Woollard, 2014)(Hu, 2011).
1	Araujo, Andrade e Serey (2015)	Nesse contexto, nós levantamos as seguintes questões de pesquisa: QP1 Qual a percepção do termo Pensamento Computacional para os profissionais da computação?; QP2 Existe diferença significativa percepção do PC quando comparados os profissionais da academia (professores, estudantes de pósgraduação e pesquisadores) aos profissionais da indústria (desenvolvedores, analistas, engenheiros de software)?; QP3 Os profissionais da computação reconhecem que exploram habilidades associadas ao PC nas suas atividades laborais? QP4 Os profissionais da computação reconhecem que exploram habilidades associadas ao PC em suas atividades do cotidiano, fora do seu ambiente de trabalho?
1	Araujo, Andrade e Serey (2015)	2. Pensamento Computacional: resolução de problemas explorando habilidades
1	Araujo, Andrade e Serey (2015)	Pensamento Computacional é uma abordagem focada na resolução de problemas explorando processos cognitivos, técnicas e ferramentas comuns na Ciência da Computação. Embora o termo faça uma alusão direta à Computação, nem todos os processos e técnicas associados são exclusivos da Computação. Mesmo assim, eles foram incorporados e são amplamente explorados no contexto de buscar soluções mais eficientes para resolver problemas ou atividades complexas.
1	Araujo, Andrade e Serey (2015)	Nosso questionário foi planejado, em sua primeira parte, para capturar a visão pessoal sobre o termo PC dos respondentes, com a pergunta discursiva "Qual seu entendimento sobre Pensamento Computacional?". E assim, responder as questões de pesquisa 1 e 2:
1	Araujo, Andrade e Serey (2015)	QP1 – Qual a percepção do termo Pensamento Computacional para os profissionais da computação?
1	Araujo, Andrade e Serey (2015)	QP2 Há diferença significativa na proporção de respostas convergentes para a definição do termo Pensamento Computacional quando comparados os profissionais da academia aos profissionais da indústria?
1	Araujo, Andrade e Serey (2015)	5.1. Análise da definição do termo "Pensamento Computacional"
1	Araujo, Andrade e Serey (2015)	Examinamos as respostas da questão discursiva "Qual seu entendimento sobre Pensamento Computacional?" no intuito de responder a questão de pesquisa 1:
1	Araujo, Andrade e Serey (2015)	5.2. Análise das habilidades associadas ao Pensamento Computacional
1	Araujo, Andrade e Serey (2015)	Nesse trabalho realizamos um survey com profissionais da computação os quais atuam na academia e na indústria, para capturar a compreensão que esses profissionais têm de Pensamento Computacional em suas atividades laborais e cotidianas. Nossos resultados

(continua)

Tabela 27 – Textos do WAlgProg (2015-2017) que usam o termo "pensamento computacional" entre onze e vinte e nove

(continuação)

quant.	autor	parágrafo
1	Schoeffel *et al.* (2015)	Resumo. Este artigo relata a experiência do ensino de pensamento computacional para alunos do ensino fundamental, por meio de um curso, utilizando ambientes lúdicos de programação, jogos, gamification e computação desplugada. O público atingido foi de 34 alunos dos 7º, 8º e 9º anos do Ensino Fundamental do município de Ibirama – Santa Catarina e a avaliação mostrou resultados positivos relacionados à satisfação e diversão com o curso. Além desses resultados, foi avaliada a participação dos alunos do curso na Olimpíada Brasileira de Informática. Foram encontradas evidências de um desempenho significativamente superior em comparação ao desempenho de alunos que não participaram dessa atividade.
1	Schoeffel *et al.* (2015)	A utilização de programação de computadores nas escolas, como meio de melhorar o raciocínio lógico, pensar de forma mais crítica e propor novas soluções são como ferramentas que podem melhorar esse cenário (BARCELOS; SILVEIRA, 2012; WILSON; MOFFAT, 2010; TSALAPATAS ET AT., 2012; LIN ET AL., 2005; PEA; KURLAND, 2007; ACKAY, 2009; SERIN; SERIN; SAELI ET AL., 2011) Segundo Pereira (2013), a codificação de programas de computadores deveria estar lado a lado com matérias tradicionais como biologia, química e física. Um dos objetivos de ensinar programação para crianças é o desenvolvimento do Pensamento Computacional, que caracteriza-se pelo uso de princípios da computação para ajudar a separar elementos de um problema em outras áreas, determinar seus relacionamentos e desenvolver passos lógicos para chegar a soluções automatizadas (KAFAI; BURKE, 2013).
1	Schoeffel *et al.* (2015)	O objetivo desse artigo é relatar uma experiência do ensino d pensamento computacional utilizando ferramentas lúdicas para alunos o sétimo, oitavo e novo ano do ensino fundamental do município de Ibirama – Santa Caarina, analisando os resultados das avaliações realizadas e os resultados da participação desses alunos na Olimpíada Brasileira de Informática (OBI).
1	Schoeffel *et al.* (2015)	Esse artigo está estruturado em cinco capítulos, sendo que o segundo capítulo contextualiza pensamento computacional e descreve trabalhos correlatos, o terceiro descreve as estratégias de ensino adotadas para os cursos realizados e a metodologia utilizada no experimento, o quarto apresenta e discute os resultados alcançados e, no quinto capítulo, são descritas as considerações finais.
1	Schoeffel *et al.* (2015)	2. Pensamento Computacional

(continua)

Tabela 27 – Textos do WAlgProg (2015-2017) que usam o termo "pensamento computacional" entre onze e vinte e nove

(continuação)

quant.	autor	parágrafo
2	Schoeffel *et al.* (2015)	Segundo Atmatzidou e Demetriadis (2014), o Pensamento Computacional tem recebido muito atenção nos últimos anos como sendo uma habilidade fundamental que promove mudanças no paradigma cognitivo dos estudantes em todas as áreas da ciência. Pensamento Computacional, conforme definido por Ray et al. (2011), é um processo de resolução de problemas que inclui: i) a formulação de problemas de forma que permita a utilização do computador ou de outras ferramentas para auxílio na resolução; ii) organização e análise lógica dos dados; iii) representação de dados por meio de abstração, como modelos e simulações; iv) solução automática por meio de pensamento algorítmico; v) identificação, análise e implementação de possíveis soluções com objetivo de alcançar a combinação mais eficiente e efetiva de passos e recursos; vi) generalização e transferência desse processo de resolução de problemas para uma grande variedade de questões.
2	Schoeffel *et al.* (2015)	Neste contexto, o pensamento computacional deve ser tratado como uma habilidade fundamental para todos, não só para cientistas da computação. Assim como a leitura, a escrita e a matemática, deve-se acrescentar pensamento computacional a todas as habilidades analíticas das crianças (WING, 2006). O desenvolvimento do raciocínio lógico é peça essencial para a resolução de problemas, pois assim o aprendiz deixa de reproduzir conceitos pré-definidos, e passa a ser mentor de suas tomadas de decisão, processando dados e informações e gerando resoluções baseadas no que aprende (VARGAS et al., 2012).
1	Schoeffel *et al.* (2015)	Diversas iniciativas têm sido desenvolvidas ao redor do mundo para o ensino de lógica e pensamento computacional para crianças desde a Educação Infantil, Ensino Fundamental e Médio. Alguns desses relatos serão descritos na próxima seção.
1	Schoeffel *et al.* (2015)	Foram encontrados dois tipos de trabalhos julgados correlatos ao presente artigo: i) relatos de experiência do ensino de lógica e pensamento computacional a alunos do ensino fundamental; ii) relatos de experiência na preparação e execução da Olimpíada Brasileira de Informática.

(continua)

Tabela 27 – Textos do WAlgProg (2015-2017) que usam o termo "pensamento computacional" entre onze e vinte e nove

(continuação)

quant.	autor	parágrafo
1	Schoeffel *et al.* (2015)	No primeiro conjunto de trabalhos, existem diversos relatos de cursos de lógica e pensamento computacional para alunos do ensino fundamental e médio, no Brasil e em diversas partes do mundo. Como exemplos, cita-se o trabalho de Wangenheim, Nunes e Santos (2014), o qual propôs e aplicou uma unidade instrucional para ensino de computação no Ensino Fundamental utilizando o Scratch em uma turma do primeiro ano em uma escola de Florianópolis-SC. Outro trabalho relacionado é o de Wilson e Moffat (2010), que avaliou a utilização do Scratch no aprendizado de programação em 21 alunos de oito e nove anos de idade, em Glasgow na Escócia. Lin et al. (2005) descreve a experiência de ensino de programação para 81 alunos do quinto, sexto e sétimo ano do Ensino Fundamental em Taiwan. Kalelioğlu e Gülbahar (2014) relatam a participação de 49 alunos do quinto ano do ensino fundamental num curso de programação com Scratch na Turquia. De forma geral, os resultados mostram que o aprendizado de computação é atrativo para os alunos e é viável para alunos desde o primeiro ano do ensino fundamental. Os trabalhos avaliam aspectos diferentes, como: adequação de ferramentas, satisfação dos alunos, aprendizagem e percepção dos professores. Porém, nenhum dos trabalhos citados apresentou tratamento estatístico para tais resultados.
2	Schoeffel *et al.* (2015)	Com relação aos relatos de ensino de pensamento computacional e raciocínio lógico por meio da programação, diversos trabalhos foram encontrados, porém a maioria deles descreve ações isoladas e de curta duração. Com relação ao incentivo e participação na OBI, nenhum dos artigos encontrados descreve resultados de forma sistemática, além de não terem sido encontrados artigos relacionando um curso para o ensino de pensamento computacional com resultados na OBI, como é o caso deste trabalho.
1	Schoeffel *et al.* (2015)	Estado de Santa Catarina, que visa o ensino lúdico de tecnologia para jovens, o qual está dividido em três ações: i) lógica computacional: ensino de raciocínio lógico e pensamento computacional de forma lúdica e prática; ii) programação de computadores: introdução ao ensino de programação, com introdução de linguagens de programação; iii) robótica: ensino e prática de montagem e programação de computadores, utilizando os kits LEGO Mindstorms®.
1	Schoeffel *et al.* (2015)	A estratégia de ensino consistiu em dividir o curso em dois momentos, sendo que o primeiro correspondeu às três primeiras semanas e incluiu atividades de preparação para a OBI, em conjunto com atividades de introdução à lógica e pensamento computacional; o segundo teve foco na prática de lógica e programação de algoritmos por meio de ferramentas lúdicas.
1	Schoeffel *et al.* (2015)	Esse trabalho descreveu uma iniciativa de ensino de pensamento computacional para alunos do ensino fundamental. Para buscar maior envolvimento e engajamento dos alunos, foram utilizados aspectos lúdicos com diferentes técnicas e ferramentas de ensino.

(continua)

Tabela 27 – Textos do WAlgProg (2015-2017) que usam o termo "pensamento computacional" entre onze e vinte e nove

(continuação)

quant.	autor	parágrafo
3	Fernandes e Silveira (2016)	Resumo. O Pensamento Computacional traz um conjunto de processos cognitivos, técnicas e conceitos para resoluções de problemas que podem ser aplicados em várias áreas do campo de conhecimento e acredita-se que a Educação a Distância pode ter grande importância para a sua disseminação e desenvolvimento. Nesse contexto, o presente artigo propõe uma investigação por meio de uma revisão narrativa da literatura, investigar os conceitos do Pensamento Computacional e as iniciativas existentes para a sua disseminação e desenvolvimento por meio da modalidade de ensino a distância. Como resultado, identificaram-se quatro plataformas e algumas lacunas que podem guiar futuras pesquisas sobre plataformas online para a disseminação e o desenvolvimento do Pensamento Computacional.
1	Fernandes e Silveira (2016)	Nesse contexto, o ensino a distância pode contribuir com a disseminação do pensamento computacional e este artigo tem como objetivo investigar por meio da revisão narrativa da literatura os conceitos do PC e pesquisar plataformas que na modalidade a distância colaborem com seu desenvolvimento e disseminação.
1	Fernandes e Silveira (2016)	2 Pensamento Computacional (PC)
1	Fernandes e Silveira (2016)	Pensamento Computacional (PC) é o processo de reconhecimento de aspectos da Ciência da Computação a partir da aplicação e utilização de ferramentas e técnicas de computação. Segundo Nunes (2011), a introdução desse conceito se justifica pelo seu caráter transversal, pois em um mundo cada vez mais globalizado é necessário dominar suas aplicações tornando o país mais rico e competitivo nas diversas áreas de aplicação da computação e da tecnologia da informação. Como afirma Bundy (2007), para entender o século XXI deve-se primeiro entender a computação. Podemos tentar definir o PC como um conjunto de habilidades e competências que prestam subsídios para resoluções de problemas, aplicável às diversas áreas e campos de atuação.
1	Fernandes e Silveira (2016)	PC, segundo Wing (2006), é uma habilidade imprescindível para todas as pessoas, assim como as habilidades de ler, escrever e fazer cálculos, por essa razão deve ser adicionado ao pensamento analítico de cada criança. Em síntese, é pensar como um cientista da computação quando confrontado por um problema desde os primeiros anos de alfabetização.
1	Fernandes e Silveira (2016)	Automatizar soluções por meio de pensamento algorítmico (uma série de passos ordenados);
1	Fernandes e Silveira (2016)	Quadro 1. Pensamento Computacional – Conceitos

(continua)

Tabela 27 – Textos do WAlgProg (2015-2017) que usam o termo "pensamento computacional" entre onze e vinte e nove

(continuação)

quant.	autor	parágrafo
1	Fernandes e Silveira (2016)	A presente pesquisa segue os moldes de uma revisão narrativa da literatura que, segundo Rother (2007), são publicações amplas apropriadas para descrever e discutir o desenvolvimento ou o "estado da arte" de um determinado assunto, sob ponto de vista teórico ou conceitual. Apanhando-se nessa premissa, foram utilizados, para revisão, artigos científicos publicados em periódicos e revistas, relatórios e documentos técnicos, material de divulgação e documentos institucionais. Assim, a busca dos trabalhos foi realizada nas bases de dados eletrônicas ScienceDirect , ScIELO2, Capes e ERIC e por fim em Artigos publicados nos Anais dos Workshops do CBIE/ WAlgProg 2015. A pesquisa sobre o PC e estratégias para o seu desenvolvimento é razoavelmente recente e, por isso, optamos por uma string de busca abrangente: Pensamento Computacional em português e Computational Thinking em inglês.
1	Fernandes e Silveira (2016)	Tabela 2. Plataformas para o desenvolvimento do pensamento computacional Plataformar ONLINE Utilizada Apenas citada
1	Fernandes e Silveira (2016)	4.2 Como é abordado o Pensamento Computacional
1	Fernandes e Silveira (2016)	É possível identificar que ainda são poucos os estudos sobre plataformas online para o desenvolvimento do PC. A maioria dos trabalhos revisados utilizou a plataforma Scratch o que revela a falta de estudos sobre outras plataformas como a Code.org ou mesmo a Computational Thinking for Educators que não foi citada nos trabalhos revisados. Por meio da identificação dessas limitações, esperamos que novos estudos venham a suprir essas lacunas. Em trabalhos futuros, pretende-se desenvolver uma pesquisa qualitativa e documental sobre as plataformas Code.org e a Computational Thinking for Educators com o intuito de analisar de que modo estas plataformas podem contribuir para o desenvolvimento e a disseminação do pensamento computacional.
1	Silva *et al.* (2016)	Resumo. Este artigo descreve uma metodologia de ensino de lógica de programação através da robótica e descreve um relato de experiência de sua aplicação em uma escola pública de um pequeno município. O objetivo da abordagem é de estimular o Pensamento Computacional nos estudantes envolvidos nas atividades. Como diferencial, a metodologia proposta considera a progressão dos alunos entre as tecnologias Lego Mindstorms e Arduino. Além disso, a abordagem leva em consideração a utilização de poucas unidades de kits de robótica para se adequar a realidade das escolas públicas brasileiras.
1	Silva *et al.* (2016)	Do mercado, indústrias, medicina e até mesmo afazeres domésticos para sala de aula, a robótica vem ganhando grande espaço e interesse por parte de pesquisadores e professores de todo o mundo. Nos últimos tempos, a robótica se mostrou como uma ferramenta importante no processo de desenvolvimento cognitivo e de habilidades sociais dos alunos, desde a pré-escola ao ensino médio [ALIMISIS 2013]. Por isso, a robótica educacional pode ser um importante meio para motivar e envolver estudantes em atividades de programação e, consequentemente, trabalhar de forma lúdica o Pensamento Computacional. O estudante poderá ser motivado a criar um algoritmo pois terá a oportunidade de testar sua solução através das ações e interações de um robô.

(continua)

Tabela 27 – Textos do WAlgProg (2015-2017) que usam o termo "pensamento computacional" entre onze e vinte e nove

(continuação)

quant.	autor	parágrafo
3	Silva *et al.* (2016)	Segundo Barr e Stephenson (2011) o Pensamento Computacional, no contexto do ensino básico, é uma abordagem de solução de problemas que pode ser automatizada, transferida e aplicada em diferentes matérias. Para os autores, os estudantes se tornam desenvolvedores das soluções. Barr e Stephenson (2011) também mostram como o Pensamento Computacional pode ser utilizado para trazer benefícios no aprendizado de diferentes disciplinas do ensino básico, como Matemática, Ciências, Estudos Sociais, Linguagem e Artes. Portanto, o estímulo do Pensamento Computacional pode trazer grandes benefícios para a educação básica brasileira, como um todo, uma vez que pode melhorar o desempenho dos alunos em várias disciplinas do currículo básico.
1	Silva *et al.* (2016)	O presente trabalho se propõe a estimular o Pensamento Computacional de forma gradual em alunos do ensino fundamental, através de atividades de robótica realizadas em equipe. Para tal, a abordagem trabalha inicialmente conceitos teóricos de robótica e algoritmos, programação visual na tecnologia Lego EV3 e, finalmente, programação e desenvolvimento de projetos na plataforma Arduino. Isso em um contexto social de estudantes que pertencem a uma realidade local de uma escola pública de um pequeno município.
1	Silva *et al.* (2016)	O ensino de conceitos de lógica de programação e algoritmos através da robótica tem sido explorado por pesquisadores e educadores de diferentes formas. No entanto, pode ser notado no trabalho de Neto et al. (2015) que ainda é necessário maiores esforços, por parte de pesquisadores, para que se tenha um maior número de trabalhos focados no tema Pensamento Computacional e Robótica Pedagógica. Ainda existem relativamente poucas publicações científicas envolvendo estes conceitos. Neste sentido, o uso da robótica no âmbito educacional é mostrado a seguir através de algumas iniciativas recentes com contextos aproximados ao presente trabalho.
1	Silva *et al.* (2016)	Cardoso e Antonello (2015) relatam a experiência de utilização de programação visual e robótica, através de ferramentas de codificação por blocos e kits Arduino. O objetivo do trabalho foi ensinar conceitos de programação para alunos ingressantes no curso de Bacharelado de Sistemas da Informação. Os autores relatam que os resultados alcançados foram positivos. De França et al. (2014) apresentam as lições aprendidas com experiências de licenciandos em computação na disseminação do Pensamento Computacional na educação básica. Entre as experiências, os autores descrevem a realização de oficinas de robótica com o apoio do projeto ROBUCA: Inserção da Robótica Educativa no UCA e da Plataforma Robô Livre. Uma revisão sistemática da literatura a respeito da Robótica Pedagógica aplicada ao ensino de programação, através de laboratórios remotos, pode ser encontrada em Almeida e Netto (2015).

(continua)

Tabela 27 – Textos do WAlgProg (2015-2017) que usam o termo "pensamento computacional" entre onze e vinte e nove

(continuação)

quant.	autor	parágrafo
1	Silva *et al.* (2016)	É importante observar que as etapas seguem uma sequência de evolução gradual do conhecimento dos alunos a respeito de lógica de programação e das tecnologias a serem utilizadas (Figura 1). No entanto, as etapas de trabalho também respeitam a autonomia dos estudantes para experimentarem e testarem diferentes soluções para os desafios propostos. Neste contexto, o Pensamento Computacional é trabalhado de forma que os alunos possam propor soluções lógicas que vão evoluindo a cada etapa.
1	Silva *et al.* (2016)	Através da utilização da robótica, pôde-se observar que o projeto despertou nos alunos o interesse científico e os incentivou na solução de desafios que envolviam decomposição de problemas e realização de testes interativos através do robô, que são exercícios essenciais para estimular o Pensamento Computacional. Também foi possível observar a importância do trabalho em equipe na solução dos desafios propostos para os estudantes.
1	Silva *et al.* (2016)	Foi possível observar, durante a aplicação dos conceitos passados na oficina, que a robótica é um tema muito envolvente e efetivamente desperta o interesse dos alunos. Experiências têm indicado que o uso da Robótica Educacional nas escolas, para ensino de lógica de programação, possibilita que esses estudantes aprendam e aprimorem o Pensamento Computacional e desta forma tenham ganhos também no aprendizado de outras disciplinas.
1	Silva *et al.* (2016)	O projeto é um grande sucesso entre os alunos. Portanto, como trabalhos futuros a proposta é sua continuação na escola, não apenas aplicando conceitos de programação e robótica, mas também a integração destes conceitos, principalmente no que diz respeito ao Pensamento Computacional, com os conteúdos de outras disciplinas. Desta forma, pode-se reforçar e ampliar o que já foi aprendido em sala de aula. Haverá também a aplicação de um questionário antes, durante e depois da oficina para avaliar quantitativamente seu aproveitamento e a evolução por parte dos alunos em outras disciplinas.
3	Barcelos, Bortoletto e Andrioli (2016)	Resumo. No contexto das discussões sobre a incorporação do Pensamento Computacional na educação básica um dos principais desafios atuais é a adequada formação de professores para desenvolver atividades relacionadas. Neste artigo apresentamos um curso para formação inicial e continuada de professores de Matemática, baseado na construção de jogos digitais e oferecido em uma plataforma online, visando capacitá-los a criar atividades que envolvam tópicos matemáticos juntamente com o desenvolvimento de competências do Pensamento Computacional. Os resultados preliminares indicam que os participantes atingiram um nível avançado de competência em alguns tópicos do Pensamento Computacional e tiveram maior interesse por atividades diretamente relacionadas com a construção de jogos.

(continua)

Tabela 27 – Textos do WAlgProg (2015-2017) que usam o termo "pensamento computacional" entre onze e vinte e nove

(continuação)

quant.	autor	parágrafo
1	Barcelos, Bortoletto e Andrioli (2016)	Desde a proposta inicial de Jeanette Wing de classificar como Pensamento Computacional (PC) um conjunto de competências e habilidades do cientista da computação que poderiam ser desenvolvidas em estudantes da educação básica, uma série de iniciativas vem sendo desenvolvidas para efetivar a difusão de tais competências e habilidades junto a esse público. Dentre elas, podemos destacar a incorporação de diretrizes curriculares para o desenvolvimento do PC ao currículo de referência para o ensino de Computação na educação básica norte-americana (CSTA, 2011) e a inclusão pelo Departamento de Educação do Reino Unido de aspectos do PC relacionadas à programação de computadores na educação básica para turmas de alunos a partir de cinco anos (CELLAN-JONES, 2014).
1	Barcelos, Bortoletto e Andrioli (2016)	No entanto, uma limitação não trivial à difusão do Pensamento Computacional é a capacitação de professores para a criação e aplicação de atividades de desenvolvimento do PC (BARCELOS et al., 2015; CURZON et al., 2014; YADAV et al., 2014). Essa tarefa pode ser mais desafiadora ao se considerar a carência de professores especializados no ensino de Computação na educação básica e a necessidade de articular o PC com o currículo tradicional obrigatório, como mencionado anteriormente.
1	Barcelos, Bortoletto e Andrioli (2016)	Frente ao exposto, este artigo apresenta o desenvolvimento e aplicação piloto de um curso online para a formação inicial e continuada de docentes de Matemática que desejem incorporar temas relacionados ao Pensamento Computacional às suas atividades didáticas. O curso utilizou como tema motivador o desenvolvimento de jogos digitais, explicitando e discutindo a cada atividade proposta os tópicos e conceitos matemáticos presentes. A sequência deste artigo é organizada da seguinte forma: na seção 2 é apresentada a fundamentação teórica para a necessidade de iniciativas de formação docente na área de PC, bem como os pressupostos pedagógicos para a estruturação da formação proposta neste artigo. Na seção 3 a estrutura e atividades do curso online são apresentados. Na seção 4 é descrito um primeiro oferecimento da formação, realizado no segundo semestre de 2015, e uma análise preliminar dos resultados obtidos pelos participantes. Por fim, na seção 5 são apresentadas as conclusões e perspectivas de trabalhos futuros.
2	Barcelos, Bortoletto e Andrioli (2016)	Segundo Yadav et al. (2014), a formação de professores visando a compreensão e aplicação de conceitos do Pensamento Computacional deve envolver não apenas os professores de Computação mas também os professores de áreas específicas. Os autores utilizaram um quase-experimento1 cuja conclusão indicou que professores em formação que participaram de uma oficina de conceitos relacionados ao Pensamento Computacional passaram a conceituá-lo como uma estratégia para resolução de problemas, aplicável a diversas áreas. Outros trabalhos que também visam identificar a mudança de percepção de professores em formação sobre o PC foram relatados por Morreale e Joiner (2011) e Blum e Cortina (2007).

(continua)

Tabela 27 – Textos do WAlgProg (2015-2017) que usam o termo "pensamento computacional" entre onze e vinte e nove

(continuação)

quant.	autor	parágrafo
2	Barcelos, Bortoletto e Andrioli (2016)	Barcelos et al. (2015) apresentaram uma revisão sistemática da literatura de trabalhos em língua inglesa, na qual concluem a carência de estudos que envolvam a incorporação do Pensamento Computacional na formação de professores de Matemática. No contexto brasileiro, Farias, Andrade e Alencar (2015) identificaram, por meio da aplicação de um questionário, que egressos de um curso de Licenciatura em Computação tem pouco conhecimento teórico ou experiência prática em relação ao conceito e desenvolvimento do Pensamento Computacional na educação básica.
2	Barcelos, Bortoletto e Andrioli (2016)	Por outro lado, encontram-se evidências da literatura de possíveis aproximações entre as competências e habilidades normalmente associadas à Matemática e aquelas associadas ao Pensamento Computacional. Ke (2014) discute que a utilização do ambiente de programação Scratch para construção de jogos permite que crianças tenham contato com conceitos matemáticos tanto simbólicos quanto concretos, e que a modificação do código de programas demanda um raciocínio abstrato e a interação com o problema por meio de símbolos e expressões matemáticas. Isso vem a complementar o trabalho de Mor e Noss (2008), onde se discute que a linguagem algorítmica pode se constituir como uma "transição suave" entre a imprecisão da linguagem coloquial e o formalismo da linguagem matemática. Comparando as diretrizes curriculares para o ensino de Matemática do Brasil, Estados Unidos e Chile, Barcelos e Silveira (2014) identificaram competências comuns entre as propostas no âmbito do Pensamento Computacional e aquelas definidas como resultado do ensino de Matemática na educação básica.
1	Barcelos, Bortoletto e Andrioli (2016)	Após desenvolver as atividades dos nove módulos os participantes devem desenvolver um jogo de sua própria autoria como projeto final do curso. Essa atividade tem como objetivo a consolidação dos conhecimentos adquiridos ao longo do curso, bem como permitir a contextualização das relações entre a Matemática que faz parte da prática docente do participante e os conceitos do Pensamento Computacional adquiridos. Para tanto, o projeto deve ser desenvolvido com base em três diretrizes: utilizar atores (sprites) em movimento; utilizar três dentre as estruturas de programação aprendidas (variáveis, estruturas condicionais, estruturas de repetição, sincronização via envio de mensagens); o jogo pode ser pensado para ser construído pelos alunos do participante ou para ser jogado por eles. Neste oferecimento do curso a apresentação dos projetos finas foi presencial e teve como aspecto motivador uma premiação para o projeto mais criativo e com melhor jogabilidade.

(continua)

Tabela 27 – Textos do WAlgProg (2015-2017) que usam o termo "pensamento computacional" entre onze e vinte e nove

(continuação)

quant.	autor	parágrafo
1	Barcelos, Bortoletto e Andrioli (2016)	No entanto, o acesso aos materiais foi maior do que a entrega de atividades pelos participantes, vinculada às tarefas propostas nos vídeos dos módulos 1 a 5, 8 e 10 (projeto final). Cada participante deveria efetuar uma entrega por módulo, e verificou-se uma média de 7,42 entregas por módulo (ou 37,1% dos participantes inicialmente inscritos), sendo que 5 participantes (25% dos inicialmente inscritos) entregaram o projeto final. Considerando a temática aberta do projeto final, optou-se por realizar uma análise dos artefatos produzidos pelos participantes (BRENNAN; RESNICK, 2012) para evidenciar o desenvolvimento de competências e habilidades do Pensamento Computacional por eles. Para tanto, utilizou-se a rubrica proposta por Moreno-León e Robles (2015), que propõe identificar o uso de conceitos em nível básico, intermediário e avançado em sete categorias. Na Tabela 2 são apresentados os critérios dessa rubrica.
1	Barcelos, Bortoletto e Andrioli (2016)	Tabela 2. Rubrica de avaliação da utilização de con ceitos do Pensamento Computacional (MORENO-LEÓN; ROBLES, 2015)
1	Barcelos, Bortoletto e Andrioli (2016)	Figura 3. Cobertura dos conceitos do Pensamento Com putacional nos projetos finais desenvolvidos pelos participantes
2	Barcelos, Bortoletto e Andrioli (2016)	A relevância do desenvolvimento do Pensamento Computacional na educação básica e o potencial de desenvolvimento conjunto com outras disciplinas vem sendo discutido ao longo dos anos. No entanto, a adequada formação de professores para trabalhar tais conceitos ainda se constitui como um desafio. Dessa forma, neste artigo é apresentada a estrutura e o oferecimento preliminar de um curso online para o desenvolvimento do Pensamento Computacional em professores de Matemática e estudantes de licenciatura em Matemática por meio do desenvolvimento de jogos digitais com a plataforma Scratch.
1	Barcelos, Bortoletto e Andrioli (2016)	Os resultados preliminares indicam que o envolvimento dos participantes foi maior em atividades do curso que envolviam diretamente a construção de jogos, o que é um indício do aspecto motivador da temática escolhida. Ainda, os concluintes do curso demostraram por intermédio dos projetos finais um domínio intermediário ou avançado de aspectos do Pensamento Computacional relacionados à programação de computadores. Dessa forma, se tornaram autores de suas próprias atividades didáticas baseadas em jogos incluindo conceitos matemáticos.
2	Zimmermann *et al.* (2016)	Resumo. O estudo objetiva avaliar o impacto da utilização do software ScratchJr para a aprendizagem de conceitos do Pensamento Computacional e sua influência sobre o desempenho cognitivo em crianças hospitalizadas. Desenvolveremos oficinas, onde sujeitos da pesquisa utilizarão a arquitetura computacional para a criação de jogos e animações. Para verificar o aprimoramento sobre conceitos do Pensamento Computacional serão utilizados análise de artefatos e entrevista ao final das oficinas. Para avaliar o nível intelectual dos participantes utilizaremos a escala WISC-III pré e pós realização das oficinas.

(continua)

Tabela 27 – Textos do WAlgProg (2015-2017) que usam o termo "pensamento computacional" entre onze e vinte e nove

(continuação)

quant.	autor	parágrafo
1	Zimmermann *et al.* (2016)	Em 2006, Wing sistematizou uma abordagem para resolução de problemas que combinou o pensamento crítico com os fundamentos da computação, definida como Pensamento Computacional. Esse método utiliza os fundamentos e técnicas da Ciência da Computação para a solução de problemas. A base dessa metodologia consiste na utilização do computador como um instrumento para melhor compreensão do mundo atual, permeado por dispositivos computacionais, refletindo positivamente em na produtividade, inventividade e criatividade do usuário (Wing, 2006).
2	Zimmermann *et al.* (2016)	O conceito de Pensamento Computacional se refere ao domínio de competências e habilidades da Computação que podem ser aplicadas à compreensão de conteúdos de outras áreas da ciência. A tecnologia computacional, que permeia nosso mundo e rotinas, é vista como importante recurso provedor de subsídios para que os indivíduos possam não apenas utilizar a tecnologia, mas também compreendê-la e serem capazes de implementar soluções para problemas utilizando recursos computacionais. Especificamente, o pensamento computacional pode auxiliar na resolução de problemas das mais diversas áreas, através de conceitos como abstração, decomposição, entre outros. Além disso, pode liderar a busca pelo aperfeiçoamento das tecnologias ligadas à informação e comunicação (CSTA, 2011a).
3	Zimmermann *et al.* (2016)	Como o Pensamento Computacional amplia a capacidade de dedução e resolução de problemas, muitos estudiosos acreditam que seus conceitos e metodologia deveriam ser ensinados desde cedo em escolas primárias (Sica, 2008). Além da competência para a leitura, escrita e aritmética, o Pensamento Computacional deveria ser adicionado ao currículo escolar para que a função de ampliação da capacidade analítica-computacional durante a formação das crianças seja alcançada de forma mais efetiva. A Microsoft e a Universidade de Carnegie Mellon criaram o Centro de Pensamento Computacional em 2007 e a Google tem se empenhado em promover esta metodologia em todo o currículo do ensino primário e secundário nos Estados Unidos (Carvalho e Moro, 2013).
1	Zimmermann *et al.* (2016)	No Brasil, atualmente, esses conceitos são aprendidos somente por aqueles que optam por cursos técnicos ou de graduação na área da computação. Esse fato se contrapõe às exigências atuais do mercado de trabalho, onde grande parte das profissões exige uma compreensão da Ciência da Computação. Esse é o caso das áreas ligadas à arte e entretenimento, comunicação, saúde, entre outros. Assim, o Pensamento Computacional mostra-se como uma das habilidades mais importantes e menos compreendidas que se tornam necessárias para o pleno exercício da cidadania neste século (Blikstein, 2008).

(continua)

Tabela 27 – Textos do WAlgProg (2015-2017) que usam o termo "pensamento computacional" entre onze e vinte e nove

(continuação)

quant.	autor	parágrafo
1	Zimmermann *et al.* (2016)	Com o Pensamento Computacional os usuários passam a ter não só a capacidade de desenvolver os seus próprios sistemas, como o reforço de competências adjacentes, sendo elas: o pensamento abstrato (utilização de diferentes níveis de abstração para perceber os problemas e, passo a passo, soluciona-los), o pensamento algorítmico (expressão de soluções em diferentes passos de forma a encontrar a forma mais eficaz e eficiente de resolver um problema), o pensamento lógico (formulação e exclusão de hipóteses) e o pensamento dimensionável (decomposição de um grande problema em pequenas partes ou composição de pequenas partes para formular uma solução mais complexa) (CSTA, 2011b).
1	Zimmermann *et al.* (2016)	Assim, pretendemos utilizar um ambiente de programação disponível para dispositivos móveis para avaliar o impacto de uma sequência didática baseada na construção de jogos digitais que contribua com o desenvolvimento de elementos do Pensamento Computacional e com o desempenho cognitivo em crianças hospitalizadas. Na sequência do artigo são apresentadas a metodologia do estudo, na Seção 2, e os resultados esperados na Seção 3.
1	Zimmermann *et al.* (2016)	O cronograma de atividades das oficinas foi dividido em 10 encontros individuais, onde os pacientes realizarão desafios relacionados com a aprendizagem de conceitos do Pensamento Computacional. Para tal utilizaremos o software ScratchJr como ferramenta computacional para o desenvolvimento das tarefas propostas. Por meio da utilização desse ambiente de programação é possível desenvolver histórias interativas, animações, jogos, músicas e artes. O ScratchJr constitui-se de uma linguagem de programação e ambiente de desenvolvimento de software gratuito, desenvolvido pelo Instituto de Tecnologia de Massachusetts (Manzano e Oliveira, 2008). A equipe do ScratchJr desenhou o idioma da interface e programação para combinar, o desenvolvimento pessoal, social, emocional e cognitivo de crianças acima de 5 (cinco) anos (Flannery, 2009).
1	Zimmermann *et al.* (2016)	Em cada atividade da oficina, realizada 2 vezes por semana, os pacientes receberão instruções sobre os objetivos propostos para as tarefas dos encontros. Serão apresentados a um exemplo da tarefa proposta sendo executada e a partir daí iniciam o trabalho. O pesquisador atuará como um facilitador, observando o trabalho e intervendo quando os pacientes solicitarem esclarecimentos. Na Tabela 1 apresentamos um resumo dos conceitos do Pensamento Computacional a serem abordados durante os encontros das Oficinas.
1	Zimmermann *et al.* (2016)	Atividade Conceitos do Pensamento Computacional a serem abordados Oficina 1 Familiarização do ambiente e sequência

(continua)

Tabela 27 – Textos do WAlgProg (2015-2017) que usam o termo "pensamento computacional" entre onze e vinte e nove

(continuação)

quant.	autor	parágrafo
1	Zimmermann *et al.* (2016)	Ao final dos 10 encontros aplicaremos a avaliação do experimento utilizando novamente a escala WISC-III, além de uma Avaliação Baseada em Evidências e uma Entrevista Baseada em Evidências e Artefatos. A Avaliação Baseada em Evidências preconiza a observação, registro e análise da utilização das ferramentas que apontam o uso de conceitos do Pensamento Computacional durante a elaboração dos projetos dos usuários no ScratchJr. No entanto, essa ferramenta de avaliação é totalmente orientada para produto e por esse motivo não revela a intencionalidade por trás do processo de desenvolvimento de projeto, e consequentemente oculta informações sobre conceitos do pensamento computacional que podem ter sido utilizados (Brennan e Resnick, 2012).
1	Zimmermann *et al.* (2016)	durante todo o processo e como ele lidou com esses problemas. Assim, a Entrevista Baseadas em Evidências e Artefatos permite o melhor entendimento sobre a fluência do usuário sobre conceitos particulares, a expansão do foco do produto exclusivamente para incluir processo permitindo a identificação da utilização de práticas pensamento computacional durante o desenvolvimento de projetos (Brennan e Resnick, 2012).
1	Zimmermann *et al.* (2016)	Neste estudo apontamos como contribuições científicas e tecnológicas o levantamento de informações sobre perfil sociodemográfico, utilização de dispositivos computacionais, os interesses e fluências relacionados à jogos digitais dos sujeitos de pesquisa e a avaliação do impacto da utilização de um ambiente de programação para dispositivos móveis na aprendizagem de conceitos do Pensamento Computacional e o impacto dessa aprendizagem sobre o desempenho cognitivo de crianças hospitalizadas participantes do estudo.
1	Zimmermann *et al.* (2016)	3 - Motivação das crianças que, ao brincarem, estarão desenvolvendo aprendizagens relativas aos conceitos do Pensamento Computacional e da resolução de problemas que contribuirão para o desenvolvimento e formação de conhecimentos importantes que serão levados ao longo da vida dos pacientes.
1	Souza, Rodrigues e Andrade (2016)	Resumo. A robótica educativa visa aproximar o que se ensina à realidade do aluno utilizando atividades baseadas em problemas. Contudo, o ensino é demasiadamente dificultado diante da deficiência na formação docente e discente em competências do Pensamento Computacional (PC). Nesse sentido, com o intuito de melhorar o ensino-aprendizagem da robótica, foi realizado um curso com ênfase em PC aplicado à robótica para docentes do Ensino Médio atuantes no ensino de robótica das Escolas SESI Paraíba. Esse trabalho apresenta uma análise dos efeitos deste curso sob a formação docente e os resultados obtidos apontam para um efeito positivo significativo na prática docente, o que se refletiu no desempenho estudantil.
1	Souza, Rodrigues e Andrade (2016)	Para Rodrigues et al. (2015), incluir o Pensamento Computacional (PC) no EB pode favorecer o desenvolvimento dos alunos em disciplinas relacionadas à resolução de problemas. Em complemento, Wing (2006) destaca que ao trabalhar com PC é necessário tratar fundamentos de estruturas computacionais como, lógica, algoritmos, resolução de problemas, dentre outras. Conforme Aguiar et al. (2015), o PC pode ser estimulado quando trabalhado em união com a robótica educacional.

(continua)

Tabela 27 – Textos do WAlgProg (2015-2017) que usam o termo "pensamento computacional" entre onze e vinte e nove

(continuação)

quant.	autor	parágrafo
1	Souza, Rodrigues e Andrade (2016)	Nesse aspecto, a presente pesquisa objetiva analisar os efeitos da introdução de PC na formação dos docentes de robótica do SESI-PB. Para isso, buscamos responder a seguinte pergunta: (RQ1) A introdução do Pensamento Computacional na formação de professores no Modelo LEGO® de Educação Tecnológica beneficia o domínio tecnológico da Robótica aplicada no Ensino Médio?
1	Souza, Rodrigues e Andrade (2016)	2.2. Ensino do Pensamento Computacional na Educação Básica
1	Souza, Rodrigues e Andrade (2016)	A ACM Model Curriculum for K-12 Computer Science (CSTA, 2016) defende a imersão do ensino de computação nos ambientes escolares, uma vez que os conceitos intrínsecos nesse segmento podem aperfeiçoar a capacidade de resolução de problemas, além do suporte nas demais ciências do currículo. Segundo Nunes (2008), para entender a computação é imprescindível explorar conceitos de "noções de modelos computacionais, algoritmos, complexidade, autômatos, entre outros conteúdos", capacidades designadas por Wing (2006) de Pensamento Computacional.
1	Souza, Rodrigues e Andrade (2016)	2.3. Pensamento Computacional aliado ao Ensino com Robótica
1	Souza, Rodrigues e Andrade (2016)	Inserir o Pensamento Computacional no contexto do EB é algo que requer planejamento (Zanetti, 2015). O uso da Robótica no Ensino Básico pode favorecer a construção de práticas e métodos para ensino do PC, pois usar robôs como instrumento pedagógico proporciona um ambiente benéfico ao aprendizado na escolar (Papert, 1985). Benitti (2012), ao discutir sobre os modos de inserir o PC no EB, apresenta a Robótica como uma ferramenta pedagógica em virtude das diversas possibilidades e recursos envolvidos que estimulam e impulsionam a efetivação do conhecimento nos alunos.
1	Souza, Rodrigues e Andrade (2016)	Este trabalho buscou responder uma questão de pesquisa (RQ1) relacionada a análise do efeito da introdução do PC na formação de professores da oficina de robótica, avaliando o desempenho de alunos antes e após a alteração da disciplina. Além disso, também foi levado em consideração a percepção dos professores referente ao impacto do Pensamento Computacional no ensino de Robótica no Ensino Básico.
1	Souza, Rodrigues e Andrade (2016)	RQ1: A introdução do Pensamento Computacional na formação de professores no Modelo LEGO® de Educação Tecnológica beneficia o domínio tecnológico da Robótica aplicada no Ensino Médio?
1	Souza, Rodrigues e Andrade (2016)	Para responder à questão de pesquisa RQ1 (A introdução do Pensamento Computacional na formação de professores no Modelo LEGO® de Educação Tecnológica beneficia o domínio tecnológico da Robótica aplicada no Ensino Médio?) analisamos a opinião dos professores que lecionam a disciplina Oficina Tecnológica de Robótica através de um questionário com 16 perguntas que considerou informações gerais sobre o perfil, além de questões específicas para a análise dos efeitos da inserção do PC na formação dos mesmos. Em paralelo, foi realizada uma análise do desempenho dos alunos em busca de indícios para a resposta da RQ1.
1	Souza, Rodrigues e Andrade (2016)	2. A metodologia aplicada nas capacitações da ZOOM Education considerou o ensino do Pensamento Computacional como pré-requisito para o aprendizado da robótica?

(continua)

Tabela 27 – Textos do WAlgProg (2015-2017) que usam o termo "pensamento computacional" entre onze e vinte e nove

(continuação)

quant.	autor	parágrafo
1	Souza, Rodrigues e Andrade (2016)	No âmbito profissional da docência, 50% concordaram parcialmente, enquanto que 50% concordaram totalmente quando foram questionados se o trabalho realizado com o PC melhorou o desempenho no ensino, dados que se reafirmam na pergunta 8 quando se faz uma comparação entre o trabalho realizado com e sem o PC. Por fim, a inserção do PC no ensino melhora o desempenho dos alunos é concordada parcialmente por 25% e totalmente por 75%, sugerindo que mesmo havendo uma falta de entendimento quanto a definição do PC, a essência técnica foi absorvida e indicada através da confirmação do Pensamento Computacional ter impactado positivamente no desempenho dos alunos. Diante dos dados discutidos, se fez necessário uma análise mais criteriosa do desempenho dos alunos na Oficina Tecnológica de Robótica considerando alunos da 1ª Série do SESI-PB de 2014 e 2015, sem e com o PC respectivamente. A priori, foi analisada a média e desvio padrão dos grupos experimental e de controle na Oficina Tecnológica de Robótica. Na Tabela 2, a média do grupo experimental é 9,8% maior em relação à média do grupo de controle. Além disso, o desvio padrão do desempenho dos alunos do grupo experimental é menor em relação a do grupo de controle, inferindo que ocorreu menor variação nas médias dos alunos do grupo experimental em relação ao grupo de controle. Os resultados obtidos deixam evidente a diferença efetiva no desempenho dos grupos, destacando um impacto positivo do PC no ensino.
1	Souza, Rodrigues e Andrade (2016)	O estudo, explicitou que a presença do PC na formação dos professores favorece o desenvolvimento técnico na área de robótica e no desempenho profissional em sala de aula, evidenciando que o desempenho dos mesmos no uso aplicado da robótica foi satisfatório, os tornando autônomos e facilitando a programação de robôs customizados, além de terem assimilado a essência do PC, o que conjecturou a inserção do Pensamento Computacional no ensino da robótica dos alunos da 1ª Série de 2015.
2	Kampff *et al.* (2016)	Resumo. O artigo relata a realização de oficina sobre Pensamento Computacional destinada a professores do ensino superior. A oficina teve como principal objetivo sensibilizá-los sobre as possibilidades de aplicação deste modelo de pensamento em atividades educacionais de diferentes áreas do conhecimento. O conceito de Pensamento Computacional é apresentado segundo a definição de Wing (2006 e 2008). Em seguida, são descritos os objetivos da oficina, os tópicos abordados e a atividade prática realizada com os participantes, baseada em um framework visual de organização de procedimentos para a solução de problemas, criado especialmente para a ocasião do curso. Os resultados da oficina, observados com base em uma pesquisa com os participantes, são apresentados na última seção.

(continua)

Tabela 27 – Textos do WAlgProg (2015-2017) que usam o termo "pensamento computacional" entre onze e vinte e nove

(continuação)

quant.	autor	parágrafo
1	Kampff *et al.* (2016)	Na última década, os avanços da web, a emergência dos dispositivos móveis, o crescimento da indústria dos games e o ressurgimento da cultura faça-você-mesmo (do- it-yourself), dentre outros fenômenos contemporâneos, evidenciam as carências de conhecimento sobre os modos de funcionamento e mesmo sobre os impactos dos meios computacionais na sociedade e na cultura [Horizon Report, 2016]. Neste contexto, segundo Berrocos et al. (2015), cresce o interesse de governos e instituições públicas e privadas em introduzir e fomentar a aprendizagem de temas relacionados ao ensino de programação em todos os níveis escolares. Associado a este movimento em prol da aprendizagem de linguagens de programação propriamente ditas, observa-se também o esforço protagonizado por pesquisadores, educadores e organizações de diversos tipos para fomentar um conjunto de competências, habilidades e disposições relacionadas ao universo da informática, as quais dão forma ao conceito de Pensamento Computacional [Blikstein, 2013].
1	Kampff *et al.* (2016)	Para autores como Wing (2006 e 2008), Pensamento Computacional é uma competência analítica fundamental que todas as pessoas, e não somente profissionais de computação, podem usar para resolver problemas em distintos contextos, para projetar sistemas e mesmo para entender o comportamento humano.
1	Kampff *et al.* (2016)	Neste contexto, o presente artigo apresenta os resultados alcançados em uma experiência de oficina sobre Pensamento Computacional destinada a professores do ensino superior, realizada durante a Semana Pedagógica da Universidade do Vale do Rio dos Sinos (UNISINOS). O principal objetivo da oficina foi sensibilizar os professores sobre as possibilidades de aplicação deste modelo de pensamento em atividades educacionais relacionadas a variados contextos de aprendizagem. Um dos interesses que motivaram a realização desta oficina foi despertar o interesse e promover o envolvimento de professores de áreas de conhecimento não relacionadas a cursos de Ciências da Computação. Como contribuição adicional, foi desenvolvido um framework visual para facilitar aos participantes a experimentação do processo.
3	Kampff *et al.* (2016)	O texto apresenta-se dividido em três seções. Na primeira delas, é descrito o contexto acadêmico e sociocultural em que o conceito de Pensamento Computacional toma forma. Já a segunda seção é destinada à narração do processo que conduziu à realização da oficina de Pensamento Computacional para os professores da UNISINOS. Na terceira parte, é descrita a atividade prática realizada durante a oficina, através da qual buscou-se aprofundar a compreensão do conceito de Pensamento Computacional, aplicando-se procedimentos orientados à resolução de variados tipos de problemas. Nesta mesma seção é apresentado o "Canvas Algorítmico", um framework visual desenvolvido especialmente para a oficina e que tem como função organizar o processo de criação de procedimentos orientados à resolução de problemas em distintos contextos. Finalmente, na seção de considerações finais, são encaminhadas algumas sugestões de ajuste a partir da experiência relatada.
1	Kampff *et al.* (2016)	2. Pensamento Computacional

(continua)

Tabela 27 – Textos do WAlgProg (2015-2017) que usam o termo "pensamento computacional" entre onze e vinte e nove

(continuação)

quant.	autor	parágrafo
1	Kampff *et al.* (2016)	Na abertura de um artigo inaugural sobre o tema, Jannette Wing (2006), uma pioneira nos estudos sobre Pensamento Computacional, chamava a atenção sobre a importância da aprendizagem de conceitos computacionais. Destacava que não se tratava somente de uma competência cognitiva útil para a escrita de programas informáticos, mas de repertório de conhecimentos favoráveis à resolução de diferentes tipos de problemas, ao estabelecimento de formas de comunicação social e à gestão de múltiplos aspectos da vida cotidiana.
2	Kampff *et al.* (2016)	Perspectivas como estas apresentadas por Berrocoso et al. (2015) e Wing (2006) sustentam a emergência de um tipo específico de abordagem analítica relacionado aos estudos de computação que vem se popularizando através do conceito de Pensamento Computacional. Para Wing (2006) Pensamento Computacional é um termo que designa um conjunto de processos de resolução de diferentes tipos de problemas que incluem características tais como sistematização e análise de dados e criação de soluções que utilizam uma série de passos ordenados (algoritmos).
1	Kampff *et al.* (2016)	Segundo Berrocoso et al. (2015), o conceito de Pensamento Computacional diz respeito a uma competência complexa relacionada a um modelo de desenvolvimento de ideias que se aplicam em múltiplos aspectos da vida cotidiana e que possuem como principal característica relacionarem-se simultaneamente com o pensamento abstrato- matemático e com o pensamento pragmático-engenheiro. Trata-se, ainda, de uma competência voltada para a resolução de problemas de forma inteligente e imaginativa (qualidades humanas as quais os computadores não detêm), que todo cidadão deveria conhecer para melhor se desenvolver na sociedade digital.
2	Kampff *et al.* (2016)	O Pensamento Computacional não diz respeito à atividade de programação de computadores propriamente dita, mas sim à capacidade de pensar abstratamente em diversos níveis, independente dos dispositivos. Pode-se desenvolver o Pensamento Computacional sem computadores - com lápis e papel, por exemplo - ainda que, evidentemente, os computadores auxiliem na resolução de problemas os quais, sem eles, a trajetória até as soluções seria muito mais difícil e, em vários casos, humanamente impossível de se alcançar.
1	Kampff *et al.* (2016)	O Pensamento Computacional permite a solução de problemas, decompondo-os em elementos e encontrando algoritmos que os resolvam. Implica em um variado conjunto de procedimentos, dentre os quais destacam-se: decomposição, reconhecimento de padrões, abstração e desenho algorítmico (Figura 1). Os procedimentos são descritos na sequência:
2	Kampff *et al.* (2016)	Tomando como base os debates em torno do conceito de Pensamento Computacional e as características que lhe conferem propriedades de processo analítico para resolução de problemas em diferentes áreas, um grupo de professores vinculados à gestão dos cursos de graduação da UNISINOS e ao Núcleo de Inovação em Práticas Educacionais (NIPE) propuseram uma oficina de capacitação para o corpo docente da instituição. O objetivo era iniciar um diálogo sobre a importância de adotar alguns dos pressupostos relacionados ao Pensamento Computacional em atividades de sala de aula.

(continua)

Tabela 27 – Textos do WAlgProg (2015-2017) que usam o termo "pensamento computacional" entre onze e vinte e nove

(continuação)

quant.	autor	parágrafo
1	Kampff *et al.* (2016)	3. Oficina de Pensamento Computacional
1	Kampff *et al.* (2016)	A UNISINOS é uma instituição de ensino superior privada localizada na cidade de São Leopoldo, Região Metropolitana de Porto Alegre, no Rio Grande do Sul. No período de abertura de todos os semestres letivos, a Universidade promove um conjunto de atividades de formação e capacitação docentes com temáticas variadas, as quais ocorrem durante o período de uma semana e reúnem professores de diversas áreas de conhecimento. Na edição de agosto de 2016, o Pensamento Computacional foi o conceito escolhido como tema central do evento, servindo de base de reflexão e orientação das atividades propostas.
1	Kampff *et al.* (2016)	Nesta seção, portanto, é descrito o processo que conduziu à proposição de uma oficina de Pensamento Computacional ofertada para professores de todos os cursos da instituição, na qual reuniram-se mais de cem participantes para debaterem e praticarem conceitos relacionados ao tema central do evento.
1	Kampff *et al.* (2016)	O primeiro desafio encontrado pela equipe de Formação Docente, que trabalhou em parceria com o NIPE, foi encontrar uma forma de projetar uma atividade de Pensamento Computacional que conseguisse engajar professores de áreas de conhecimento não relacionadas à Ciência da Computação. Foram realizadas várias reuniões para debater o planejamento da oficina, bem como as possibilidades de estratégias adotadas para tornar a abordagem sobre o tema ao mesmo tempo acessível e familiar para todo o grupo de professores. Em determinado momento do processo, optou-se por realizar um teste de apresentação dos conteúdos selecionados para serem trabalhados durante a oficina com professores representantes das diferentes áreas de conhecimento, tendo em vista verificar a receptividade às discussões teóricas, exemplos apresentados e atividades propostas. Este encontro foi fundamental para que algumas decisões fossem tomadas, descritas na sequência.
2	Kampff *et al.* (2016)	Por fim, o grupo de professores responsáveis pelo planejamento da oficina definiu que o planejamento deveria contemplar três etapas: a) debate sobre a presença ubíqua da computação em nossa sociedade a partir de exemplos selecionados por áreas de conhecimento; b) apresentação do conceito de Pensamento Computacional; c) realização de uma atividade prática com os participantes da oficina. Sobre esta última etapa, um ponto de consenso foi que a atividade deveria aprofundar alguns dos aspectos conceituais relacionados ao Pensamento Computacional por meio de um exercício que pudesse ser realizado em pequenos grupos e sem o uso de computadores. A partir destes pressupostos, foi desenvolvido um mapa visual de organização de procedimentos para a solução de problemas, o Canvas Algorítmico, que será descrito na próxima seção.

(continua)

Tabela 27 – Textos do WAlgProg (2015-2017) que usam o termo "pensamento computacional" entre onze e vinte e nove

(continuação)

quant.	autor	parágrafo
2	Kampff *et al.* (2016)	Partindo-se de um dado problema, o canvas tem como propósito principal orientar aqueles que o utilizam através de uma série de passos que conduzam até sua resolução. Os passos retomam os principais procedimentos relacionados ao Pensamento Computacional, são eles: decomposição, padrões, abstração e desenho algorítmico. Assim, após serem discutidos exemplos de recursos computacionais utilizados no meio acadêmico e profissional, seguido da discussão sobre os processos presentes no Pensamento Computacional, apresentou-se o Canvas Algorítmico e exemplos em diversas áreas do conhecimento. A título de exemplo, apresentamos, na Figura 3, uma forma de como o canvas poderia ser preenchido a partir do problema "Como melhorar a coleta de lixo de uma cidade"?
1	Kampff *et al.* (2016)	Após a discussão dos exemplos, em grupos de 4 a 5 pessoas, os docentes pensaram em problemas complexos de suas áreas de atuação e utilizaram o Canvas Algorítmico para propor soluções. O material gerado foi partilhado com os demais grupos presentes em cada uma das oficinas. Como as quatro oficinas ocorreram simultaneamente, no primeiro dia da Semana Pedagógica da UNISINOS, os professores seguiram discutindo suas possibilidades para uma nova cultura universitária, ratificando a importância de oportunizar formação aos estudantes universitários sobre Pensamento Computacional, o que já ocorrerá neste semestre por meio da oferta de cursos de extensão.
1	Kampff *et al.* (2016)	Ao avaliarmos os dados da pesquisa, verifica-se que os professores foram bastante receptivos à proposta, reconheceram o alto grau de aplicabilidade dos conceitos explorados e demonstraram compreender a importância do desenvolvimento do Pensamento Computacional na Universidade. O engajamento dos docentes durante as oficinas ratificou o compromisso do NIPE em oportunizar novas formações sobre o tema, aprofundando conceitos e aplicações.
1	Geraldes *et al.* (2017)	Resumo. Pensamento Computacional (PC) diz respeito à resolução de problemas baseada em conceitos da computação e que podem ser utilizado em diversas atividades cotidianas. Sobre este aspecto, é preciso compreender como as habilidades relacionadas ao PC podem ser disseminadas no ambiente educacional pelas diversas disciplinas curriculares dos mais diversos cursos de formação. Este artigo apresenta uma pesquisa com professores da educação profissional e tecnológica com objetivo de analisar a percepção destes com relação à utilização do PC em suas práticas pedagógicas. Concluiu-se que, nesse contexto, PC é ainda associada ao uso do computador como ferramenta de apoio às atividades pedagógicas restritas a tarefas operacionais.

(continua)

Tabela 27 – Textos do WAlgProg (2015-2017) que usam o termo "pensamento computacional" entre onze e vinte e nove

(continuação)

quant.	autor	parágrafo
1	Geraldes *et al.* (2017)	O termo Pensamento Computacional (PC) foi utilizado pela primeira vez por Seymour Papert em seu livro "Mindstorms: crianças, computadores e ideias poderosas" (Papert, 1980). A professora Jeannett Wing, do Departamento de Ciência da Computação da Universidade Carnegie Mellon afirma que o PC pode ser definido como a capacidade humana em compreender e resolver problemas utilizando os conceitos fundamentais da computação [Wing, 2006; Aho, 2012; The Royal Society, 2012; Groover e Pea, 2013]. Nos últimos 10 anos, várias experiências com a inclusão da Ciência da Computação nos currículos escolares têm sido realizadas com o objetivo de disseminar o PC.
1	Geraldes *et al.* (2017)	2. O Pensamento Computacional
2	Geraldes *et al.* (2017)	O instrumento de coleta de dados foi um questionário online contendo questões fechadas e abertas (Quadro 1), adaptadas do questionário aplicado por Mannila et al. (2014). A primeira parte do questionário foi planejada para capturar a visão pessoal sobre o termo PC dos respondentes. Para isso, foi utilizada uma pergunta discursiva "O que você entende sobre o termo Pensamento Computacional?". O objetivo desta pergunta foi responder a primeira questão de pesquisa (Qual a percepção dos professores da educação profissional e tecnológica sobre o termo Pensamento Computacional?).
1	Geraldes *et al.* (2017)	1. O que você entende sobre o termo Pensamento Computacional?
1	Geraldes *et al.* (2017)	A seguir é apresentada: (i) uma análise da percepção dos professores sobre o termo "Pensamento Computacional" em relação à definição da CSTA, (ii) as atividades pedagógicas realizadas pelos professores que eles reconhecem como habilidades associadas ao PC (iii) as ferramentas ou softwares utilizadas nas atividades pedagógicas em sala de aula em relação ao estudo realizado por Mannila et al. (2014).
1	Geraldes *et al.* (2017)	5.1 Análise do entendimento do termo "Pensamento Computacional"
2	Geraldes *et al.* (2017)	Foram examinadas as respostas da questão discursiva "O que você entende sobre o termo Pensamento Computacional?" no intuito de responder a questão de pesquisa 1: Qual a percepção dos professores da educação profissional e tecnológica sobre o termo Pensamento Computacional?
1	Geraldes *et al.* (2017)	Este trabalho assumiu como objetivo conhecer a percepção dos professores da educação profissional e tecnológica sobre o Pensamento Computacional, seus conceitos e habilidades em relação a sua prática pedagógica diária. Este objetivo foi subdividido em duas questões de pesquisa: (i) Qual a percepção dos professores da educação profissional e tecnológica sobre o termo PC? (ii) Os professores da educação profissional e tecnológica reconhecem que exploram habilidades associadas ao PC nas suas atividades pedagógicas?

(continua)

Tabela 27 – Textos do WAlgProg (2015-2017) que usam o termo "pensamento computacional" entre onze e vinte e nove

(continuação)

quant.	autor	parágrafo
2	Barcelos *et al.* (2017b)	Resumo. A análise automatizada de código desenvolvido em atividades de desenvolvimento do Pensamento Computacional vem mostrando potencial em identificar estratégias utilizadas para a resolução dos problemas propostos. No entanto, ainda não tem sido explorada em detalhes a evolução do código e sua relação com a aquisição de competências do Pensamento Computacional. Neste estudo foram empregados mapas auto-organizáveis na análise de jogos produzidos por alunos em uma Oficina de Produção de Jogos Digitais nos anos de 2013 e 2017. O resultado do processo de treinamento produziu agrupamentos no mapa que apresentaram coerência com estratégias utilizadas pelos alunos e previamente observadas em sala de aula.
1	Barcelos *et al.* (2017b)	Da mesma forma que em outros contextos do processo de ensino-aprendizagem, avaliar adequadamente o desenvolvimento de competências e habilidades relacionadas ao Pensamento Computacional se constitui como um desafio. A avaliação da aprendizagem em contextos experimentais tem sido frequentemente realizada com a utilização de técnicas de pesquisa relacionadas ao paradigma qualitativo, tais como a observação etnográfica e as entrevistas, bem como testes e questionários específicos (ARAÚJO; ANDRADE; GUERRERO, 2016). Por outro lado, outras iniciativas para assegurar o desenvolvimento de um conjunto mínimo de competências e habilidades incluem a definição de currículos de referência (CSTA, 2011) e rubricas educacionais para contextos mais específicos (MORENO-LEÓN; ROBLES, 2015).
1	Barcelos *et al.* (2017b)	Rubricas educacionais podem estar associadas a ambientes de desenvolvimento utilizados para fomentar o Pensamento Computacional. O Scratch concentra algumas dessas iniciativas devido a sua ampla utilização, chegando inclusive a ser citado recentemente dentre as 20 linguagens de programação mais populares no índice TIOBE (TIOBE, 2017). Sistemas para análise do código produzido com o Scratch visam identificar desde estatísticas básicas de utilização das estruturas de programação (WOLZ; HALLBERG; TAYLOR, 2011) até análises estruturais mais elaboradas (BOE et al., 2013; MORENO-LEÓN; ROBLES, 2015).
1	Barcelos *et al.* (2017b)	entanto, ainda há poucos trabalhos de pesquisa que explorem a utilização de métricas obtidas a partir de código Scratch para obter inferências de nível mais alto sobre o desenvolvimento de competências ao longo do tempo de forma semi-automatizada. A utilização de técnicas de Analítica de Aprendizagem (SILVA; PERES; BOSCARIOLI, 2016) poderia, por exemplo, permitir que um professor visualizasse o avanço de seus alunos de forma individualizada e ao mesmo tempo comparada com uma base histórica de resultados, permitindo um suporte adequado a um aluno que eventualmente apresentasse dificuldades em uma determinada etapa do desenvolvimento do Pensamento Computacional.

(continua)

Tabela 27 – Textos do WAlgProg (2015-2017) que usam o termo "pensamento computacional" entre onze e vinte e nove

(continuação)

quant.	autor	parágrafo
1	Barcelos *et al.* (2017b)	Dessa forma, o objetivo deste artigo é apresentar uma validação preliminar da viabilidade do uso de técnicas de aprendizado de máquina, em específico os Mapas Auto-Organizáveis, para identificar o desenvolvimento ao longo do tempo de competências e habilidades do Pensamento Computacional em alunos participantes de uma Oficina de Desenvolvimento de Jogos Digitais. A sequência deste artigo é organizada da seguinte forma: na seção 2 é apresentada uma revisão da literatura sobre a utilização de métricas obtidas a partir de código Scratch para avaliar a aprendizagem e o funcionamento dos mapas auto-organizáveis; na seção 3 são descritos os métodos e técnicas utilizados na pesquisa, bem como a estrutura e instâncias de oferecimento da Oficina de Jogos Digitais cujos dados foram utilizados; os resultados e discussão são apresentados na seção 4 e na seção 5 são apresentadas as conclusões preliminares da pesquisa e os possíveis trabalhos futuros.
1	Barcelos *et al.* (2017b)	que Brennan e Resnick (2012) propuseram que a aquisição de competências do Pensamento Computacional poderia ser evidenciada pela complexidade dos artefatos computacionais produzidos por alunos, tais como o código de programas, algumas estratégias para a análise de códigos produzidos no ambiente Scratch vêm sendo desenvolvidas e avaliadas. Wolz et al. (2011) propuseram o Scrape, um sistema para análise e apresentação visual do tipo e frequência de utilização das estruturas de programação em um programa Scratch. Uma abordagem mais sofisticada foi descrita por Boe et al. (2013) com o Hairball, um sistema baseado em plug-ins com o objetivo de extrair diversas características por meio da análise do código Scratch, tais como o emprego de práticas não recomendadas (bad smells).
2	Barcelos *et al.* (2017b)	Moreno-León e Robles (2015) propuseram uma rubrica que associa o nível de complexidade e frequência das estruturas de programação utilizadas a sete categorias conceituais relacionadas ao Pensamento Computacional: abstração, paralelismo, lógica, sincronização, controle de fluxo, interação com o usuário e representação de dados. A cada categoria é atribuída uma pontuação 1, 2 ou 3 em função da complexidade das estruturas e estratégias de programação utilizadas. A rubrica foi implementada em um sistema web denominado Dr. Scratch que se baseia em uma extensão do funcionamento do Hairball. A partir da análise automatizada de código tem sido possível obter resultados relacionados às boas e más práticas de programação utilizando Scratch (AIVALOGLOU; HERMANS, 2016; TECHAPALOKUL, 2017). No entanto, ainda não tem sido explorada em detalhes a evolução do código produzido por alunos e sua relação com a aquisição de competências do Pensamento Computacional.
1	Barcelos *et al.* (2017b)	No contexto desta pesquisa a utilização do SOM se justifica pela variada quantidade de métricas que podem ser obtidas a partir de código Scratch produzido por alunos e a necessidade de exploração visual da correlação com outras variáveis cujo efeito no desenvolvimento do Pensamento Computacional ainda não é bem esclarecida.

(continua)

Tabela 27 – Textos do WAlgProg (2015-2017) que usam o termo "pensamento computacional" entre onze e vinte e nove

(continuação)

quant.	autor	parágrafo
1	Barcelos *et al.* (2017b)	A Oficina de Produção de Jogos Digitais, descrita anteriormente em (MUÑOZ et al., 2015; BARCELOS et al., 2014) foi criada com o objetivo de promover o desenvolvimento de competências do Pensamento Computacional relacionadas à automação de processos e construção de algoritmos por meio da construção de jogos digitais no ambiente Scratch. Sua estrutura, inspirada em princípios do construcionismo e da Aprendizagem Baseada em Problemas (ABP), propõe desafios de complexidade crescente a partir dos quais os participantes são convidados a explorar conceitos relacionados ao desenvolvimento de jogos (animação de sprites, colisão, controles por teclado e mouse) e a fundamentos de programação (variáveis, estruturas condicionais, laços e mensagens). Os desafios estão, na maior parte do tempo, relacionados à construção de jogos digitais reais.
1	Barcelos *et al.* (2017b)	Os projetos finais (jogo 9), apesar de estarem localizados em nodos próximos no mapa, apresentam duas diferenciações importantes: a presença de um peso maior para o "Controle de fluxo" no cluster mais à direita. Em análise conjunta com o ano de oferecimento da Oficina, foi possível verificar que todos os jogos do cluster à direita foram oriundos da oficina de 2017. Considerando a similaridade dos pesos das demais características do Pensamento Computacional nos dois clusters, é possível inferir que a estratégia de utilizar um projeto com temática mais "aberta", conforme mencionado anteriormente, pode ter possibilitado aos alunos exercitar de forma mais efetiva algumas competências em relação à edição de 2013.
1	Barcelos *et al.* (2017b)	Quanto à definição da rubrica e sua relação com o processo de agrupamento do algoritmo para treinamento da rede, verificou-se uma uniformidade nos pesos atribuídos à característica "Interação com o usuário". A rubrica considera que o nível mais alto de competência nessa característica relaciona-se ao uso dos novos recursos de reconhecimento de vídeo e áudio presentes no Scratch 2.0. Como a Oficina não previa a utilização desses recursos, é esperado que a avaliação dos jogos dos participantes seja sempre abaixo do máximo nesse critério, provavelmente o tornando pouco determinante no processo de treinamento. Devido ao fato desses novos recursos não envolverem nenhuma complexidade para sua modelagem ou uso, pois conceitualmente tratam-se apenas de variáveis pré-definidas que indicam a maior ou menor presença de movimentação no vídeo ou volume no áudio capturado no microfone, seria possível questionar se seu uso realmente denotaria um nível mais alto de competência no Pensamento Computacional.
1	Barcelos *et al.* (2017b)	A análise automatizada do código produzido por alunos é uma técnica promissora para identificar e monitorar as estratégias de resolução de problemas empregadas e sua relação com o Pensamento Computacional. No entanto, técnicas mais sofisticadas de análise podem ser necessárias para avaliar o avanço da sofisticação das estratégias utilizadas por um aluno ao longo do tempo e mesmo a comparação dos resultados obtidos por ele no desenvolvimento de um programa em relação a soluções de referência previamente coletadas.

(continua)

Tabela 27 – Textos do WAlgProg (2015-2017) que usam o termo "pensamento computacional" entre onze e vinte e nove

(continuação)

quant.	autor	parágrafo
1	Brackmann *et al.* (2017)	Resumo. O Pensamento Computacional (PC) vem gerando um novo foco educacional nas escolas mundiais como um conjunto de aptidões para a solução de problemas. Até o momento, não há um consenso de metodologia de ensino e disponibilidade de material para atender as expectativas dos professores. Para sanar essa incerteza, realizou-se uma avaliação de estudantes da educação primária espanhola com uma abordagem Quase- Experimental no intuito de beneficiar crianças em regiões/escolas onde não há dispositivos eletrônicos, Internet e até mesmo energia elétrica. Os resultados apresentam dados estatísticos que comprovam uma melhoria significativa no desempenho dos estudantes que tiveram atividades de PC Desplugado.
1	Brackmann *et al.* (2017)	Vivemos em tempos marcados pela fluidez da informação e pela valorização do conhecimento. Mais do que nunca, para lidar com a informação, processá-la e transformá-la em aptidões para a vida, exige-se, em primeiro lugar, o domínio de uma série de ferramentas e recursos tecnológicos que devem ser acessíveis a todos, sem distinção de qualquer natureza. Nos tempos atuais, o desafio que se impõe aos usuários é criar seus próprios sistemas (por exemplo, programas e jogos) ou modificar os existentes de acordo com sua necessidade pessoal. É neste contexto que surge a habilidade que é vista como crucial no século 21: o Pensamento Computacional (PC) (Kologeski et al., 2016). Devido a essa tendência mundial, o PC vem sendo adotado em escolas da educação básica em diversos países (Brackmann et al., 2016).
1	Brackmann *et al.* (2017)	(2006) define o PC como uma atividade mental para a formulação de um problema que é possível ser resolvido computacionalmente, ou seja, processos de pensamentos envolvidos na identificação de um problema e expressar a sua solução de forma eficaz, de tal forma que uma máquina ou uma pessoa possa executar. O PC utiliza quatro dimensões (também conhecido como pilares) para atingir o objetivo principal dessa abordagem: solução de problemas. Pesquisas lideradas pela Code.Org (2015), Liukas (2015) e BBC Learning (2015) mesclaram os elementos citados por Grover e Pea (2013) e resumiram nos chamados "Quatro Pilares do Pensamento Computacional", sendo eles: Decomposição, Reconhecimento de Padrões, Abstração e Algoritmos. Todos os Quatro Pilares têm grande importância e são interdependentes durante o processo de formulação de soluções computacionalmente viáveis.
1	Brackmann *et al.* (2017)	O Pensamento Computacional envolve identificar um problema complexo e quebrá-lo em pedaços menores e mais fáceis de gerenciar (Decomposição). Cada um desses problemas menores pode ser analisado individualmente com maior profundidade, identificando problemas parecidos que já foram solucionados anteriormente (Reconhecimento de padrões), focando apenas nos detalhes que são importantes, enquanto informações irrelevantes são ignoradas (Abstração). Por último, passos ou regras simples podem ser criados para resolver cada um dos subproblemas encontrados (Algoritmos). Seguindo os passos ou regras utilizadas para criar um código, é possível também ser compreendido por sistemas computacionais e, consequentemente, utilizado na resolução de problemas complexos de forma eficiente, independentemente da carreira profissional que o estudante deseja seguir.

(continua)

Tabela 27 – Textos do WAlgProg (2015-2017) que usam o termo “pensamento computacional” entre onze e vinte e nove

(continuação)

quant.	autor	parágrafo
1	Brackmann *et al.* (2017)	contexto, foram criadas e adaptadas diferentes atividades para professores e alunos de maneira que possam usar e replicar esse material para suas aulas sem a necessidade de equipamentos eletrônicos, Internet ou eletricidade para que crianças possam estudar conceitos da Computação em escolas que não possuem equipamentos apropriados (e.g. estragados, ultrapassados ou na ausência destes) ou localizadas em áreas geográficas distantes (e.g. áreas rurais ou florestais). Acredita-se que com o uso dessas atividades desplugadas (sem a necessidade de máquinas) é possível ensinar Pensamento Computacional de maneira mais acessível, ou seja, usando basicamente papel, tesoura, canetas, lápis de colorir, cola e demais materiais escolares de uso comum.
2	Brackmann *et al.* (2017)	Na literatura é raro encontrar pesquisas a respeito da aplicação e avaliação de estudantes de maneira desplugada. Para preencher esse vazio, o presente trabalho apresenta uma pesquisa realizada em duas escolas primárias na Espanha para avaliar se as crianças têm uma alteração no desempenho das habilidades relacionadas ao Pensamento Computacional através de atividades sem computadores. Sendo assim, esta pesquisa tem como objetivo verificar se aulas de Pensamento Computacional Desplugado na educação primária são eficazes através da aplicação de um pré e pós teste, realizados antes e após as aulas de PC Desplugado.
1	Brackmann *et al.* (2017)	O surgimento do Pensamento Computacional Desplugado não é muito claro, pois a necessidade de abstração para a criação de qualquer software e hardware é essencial. É importante constar também que o uso de exemplos físicos e materiais escolares são comuns para simular o comportamento de máquinas até os dias atuais em cursos de graduação. Quando se trata de salas de aulas da educação básica, os primeiros registros são encontrados a partir de 1997 quando Bell et al. (1997) lançou um rascunho de um livro em formato digital denominado "Computer Science Unplugged... Off-line activities and games for all ages"para professores de todos os níveis escolares que queriam dar um diferencial a seus alunos. A ideia foi muito bem recebida pelos demais professores e a própria academia. Devido à qualidade do material publicado, a Association for Computing Machinery (ACM) recomendou que as atividades contidas no livro fizessem parte do currículo proposto pela Computer Science Teachers Association (CSTA) dos Estados Unidos da América (Association for Computing Machinery, 2003). Até a publicação deste artigo, o livro CS Unplugged se encontra na versão 3.1 e pode ser acessado no site do projeto (Bell et al., 2015).

(continua)

Tabela 27 – Textos do WAlgProg (2015-2017) que usam o termo "pensamento computacional" entre onze e vinte e nove

(continuação)

quant.	autor	parágrafo
1	Brackmann *et al.* (2017)	Existem diversos estudos que investigam a eficiência de linguagens de programação (visual e código) com crianças Román-Gonzáles et al. (2015), Román-Gonzáles (2016), Grover et al. (2017), Franklin et al. (2017), porém carece de abordagens desplugadas. Outras pesquisas tentam padronizar a avaliação e o ensino de atividade de PC Desplugado, como por exemplo: 1) Nishida et al. (2009) onde apresenta uma proposta de padrão de design, 2) uma avaliação do PC de maneira transversal em uma escola de ensino médio (Feaster et al., 2011); 3) estudos de casos no processo de adoção do PC na sala de aula (Curzon, 2013); 4) avaliação dos pontos de vista de estudantes a respeito da CC antes e após aulas de PC (Taub et al., 2009); e também 5) sugestões de como professores podem avaliar o progresso dos estudantes ao realizar atividades de PC (Curzon et al., 2014). Lambert et al. (2009) realizou uma tentativa similar, porém na identificação de um aumento no interesse nas áreas da Computação ou Matemática, sem verificar um aumento/decremento nas habilidades relacionadas ao Pensamento Computacional.
1	Brackmann *et al.* (2017)	3.2. Instrumento Avaliativo: Teste de Pensamento Computacional
2	Brackmann *et al.* (2017)	O teste utilizado na pesquisa para medir o desenvolvimento das habilidades que fazem parte do Pensamento Computacional foi desenvolvido pelo pesquisador espanhol Román- González et al. (2015). Este teste tenta identificar a habilidade de formação e solução de problemas, baseando-se nos conceitos fundamentais da Computação, além de utilizar sintaxes lógicas usadas nas linguagens de programação. É o único teste encontrado na literatura com essas características. As questões que compõem o instrumento avaliativo incluem conceitos dos quatro pilares do Pensamento Computacional: abstração, decomposição, reconhecimento de padrões e algoritmos. Devido ao fato de o teste ter passado por um rigoroso processo de validação, o autor entende que os resultados são confiáveis (Román-González, 2015). O instrumento é composto por 28 questões de múltipla escolha, sendo que cada questão possui quatro alternativas de resposta e somente uma é válida. Um exemplo de questão é apresentado na Figura 1. Sua aplicação ocorre em navegadores (e.g. Chrome, Firefox, Edge) e pode ser acessado de qualquer dispositivo. Nesta pesquisa utilizaram-se exclusivamente os computadores do laboratório de informática das escolas.
1	Brackmann *et al.* (2017)	Figura 1. Exemplo de questão do Teste de Pensamento Computacional: "Qual sequência leva o 'Pac-Man' até o fantasma pelo caminho indicado?"

(continua)

Tabela 27 – Textos do WAlgProg (2015-2017) que usam o termo "pensamento computacional" entre onze e vinte e nove

(continuação)

quant.	autor	parágrafo
1	Brackmann *et al.* (2017)	Na primeira semana da pesquisa, os alunos das quatro turmas foram convidados a participar do experimento, como parte das suas aulas regulares durante o primeiro semestre de 2017. Para aplicação dos pré-testes, os alunos foram acompanhados de suas professoras ao laboratório de informática da escola, onde permaneceram por até 60 minutos para a realização de um teste individual de Pensamento Computacional desenvolvido por Román-González et al. (2015). Durante as próximas cinco semanas, foram ministradas aproximadamente duas horas semanais de aula de PC Desplugado no grupo experimental. Alguns dos materiais utilizados durante as aulas foram apresentados na Seção 3.3. Durante cada uma das sessões semanais, era possível trabalhar em média duas atividades. Na sétima semana, os estudantes de ambos os grupos (experimental e controle) foram novamente levados até o laboratório de informática para nova realização do teste. As respostas de todos os estudantes foram registradas na Google Cloud para que, posteriormente, pudessem ser visualizadas, recuperadas e convertidas. As respostas foram então tabuladas e analisadas estatisticamente. Os resultados e discussões sobre os dados coletados encontram-se disponíveis na próxima seção.
1	Brackmann *et al.* (2017)	Esta seção apresenta os resultados dos Testes de Pensamento Computacional e uma breve discussão dos mesmos. Relembra-se que nenhum dos alunos teve contato com aulas formais de programação. A pontuação dos testes de PC é calculada de acordo com a quantidade de questões respondidas corretamente, lembrando que o teste é composto de 28 questões (pontuação máxima). O teste foi aplicado tanto no Grupo Experimental como no de Grupo Controle. Na Tabela 2 são apresentados os resultados das médias e desvios padrão (DP) das duas turmas e na Figura 3 o gráfico compara as pontuações entre o pré e pós-teste.
1	Brackmann *et al.* (2017)	Este artigo apresentou uma breve introdução do Pensamento Computacional, seu histórico e o estado-da-arte. Em seguida foram explanadas todas as etapas da pesquisa, os grupos participantes, o teste utilizado, os materiais desenvolvidos e os resultados. Baseado na experiência adquirida durante o processo, pode-se concluir o seguinte:
1	Brackmann *et al.* (2017)	pode ser visto na Tabela 2, houve uma melhora considerável na pontuação dos estudantes com resultados estatísticos altamente significativos no grupo experimental após apenas 10 horas de aula de Pensamento Computacional
1	Brackmann *et al.* (2017)	abordagem desplugada tem suas limitações e, por isso, recomenda-se seu uso na introdução do Pensamento Computacional. Como trabalho futuro, propõe-se uma pesquisa mais detalhada para identificar o ponto de convergência da abordagem plugada e desplugada ou quando a abordagem desplugada perde sua eficácia e se torna necessário migrar para as máquinas. Outra proposta seria uma validação do Teste de PC no formato impresso.

(continua)

Tabela 27 – Textos do WAlgProg (2015-2017) que usam o termo "pensamento computacional" entre onze e vinte e nove

(continuação)

quant.	autor	parágrafo
1	Ortiz e Pereira (2017)	Resumo: Diferentes iniciativas para promover o desenvolvimento do pensamento computacional estão surgindo no Brasil, porém pouca atenção ainda é direcionada ao público da educação de jovens e adultos. Este é um público que busca completar a educação básica e se atualizar perante os contextos da sociedade moderna, possuindo desafios particulares de sua realidade socioeconômica. Este artigo apresenta uma discussão sobre o desenvolvimento do pensamento computacional no ensino de jovens e adultos com o intuito de sensibilizar para a importância de se investigar, propor e experimentar atividades, respeitando as particularidades deste público.
1	Ortiz e Pereira (2017)	Wing (2006) argumenta que o pensamento computacional envolve a resolução de problemas, a concepção de sistemas e a compreensão do comportamento humano, baseando-se nos conceitos fundamentais da Ciência da Computação. A National Research Council (2010) diz que uma pessoa pensando computacionalmente percebe que modelos computacionais podem ajudar a resolver e entender problemas de várias áreas, como mudanças climáticas, economia, entre outros.
1	Ortiz e Pereira (2017)	Recentemente, diversas iniciativas têm surgido no Brasil com o intuito de ensinar algoritmos, programação e outros fundamentos computacionais para desenvolver habilidades e aptidões relacionadas ao pensamento computacional. Públicos bastante visados para estas pesquisas são os alunos do próprio ensino superior em Ciência da Computação, alunos dos ensinos fundamental e médio, e professores. O público da educação de jovens e adultos (EJA), entretanto, não tem sido contemplado por estas iniciativas, sendo um contexto passível de exploração e com características diferenciadas, pois possuem diversos contextos sociais, culturais e econômicos desafiadores, muitas vezes gerados pelo abandono dos estudos.
1	Ortiz e Pereira (2017)	O fato de estarem retomando os estudos após a vida adulta, sugere o desejo do público da EJA de se desenvolver. Assim, o desenvolvimento do pensamento computacional nesse público poderá tanto beneficiá-los, por ser uma ferramenta capaz de ajudar na compreensão e resolução de diversos problemas, quanto trazer desafios de pesquisa, demandando a elaboração de estratégias adequadas para tal desenvolvimento.
1	Ortiz e Pereira (2017)	Para se propor e experimentar formas eficientes de despertar o pensamento computacional no público de alunos da EJA e fazer com que este conhecimento tenha sentido em seus contextos, é preciso compreender a realidade sociocultural desse público. Neste artigo desenvolvemos mais essa discussão, apresentamos um mapeamento de literatura sobre as iniciativas no cenário brasileiro e apresentamos desafios e oportunidades de pesquisa neste contexto.
1	Ortiz e Pereira (2017)	2. Pensamento Computacional

(continua)

Tabela 27 – Textos do WAlgProg (2015-2017) que usam o termo "pensamento computacional" entre onze e vinte e nove

(continuação)

quant.	autor	parágrafo
1	Ortiz e Pereira (2017)	documento "Referencias de Formação em Educação" (ZORZO et al., 2017, p.7), aprovado pela Comissão de Educação, cita como habilidades relacionadas à computação e ao pensamento computacional propostas para a educação infantil: "Compreender uma situação problema criando e identificando sequências de passos de uma tarefa para sua solução."; "Representar os passos de uma tarefa através de uma notação pictórica, de forma organizada e relacional"; "Criar passos para solução de problemas relacionados ao movimento do corpo e trajetórias espaciais.". Para o ensino fundamental: "Representar em experiências concretas as principais abstrações para descrever dados: registros, listas e grafos."; "Identificar as principais abstrações para construir processos: escolha, composição e repetição, simulando e definindo algoritmos simples que representem situações do cotidiano infantil."; "Utilizar linguagem lúdica visual para representar algoritmos."; e "Compreender a técnica de decompor um problema para solucioná-lo.".
1	Ortiz e Pereira (2017)	Os artigos mapeados foram classificados nas categorias: "Nova Estratégia" — trabalhos que relatam apenas a utilização de diferentes ferramentas ou estratégias para o ensino de conteúdos de computação junto aos públicos normalmente abordados (e.g., alunos da graduação de Ciência da Computação e cursos afins); e "Novo Público" — trabalhos que, além de relatarem uso de diferentes ferramentas/estratégias, relatam também o ensino de fundamentos da computação e pensamento computacional para públicos diferenciados, como os alunos da EJA, por exemplo. Como critérios de exclusão, foram desconsideradas as pesquisas que relatavam análise/proposta de ferramentas, análises de oportunidades e revisões sistemáticas, pois tais artigos não fazem parte do escopo da pesquisa. Ao término desta etapa, os trabalhos relacionados somavam 31, e foram classificados da seguinte maneira: 12 relatos de Nova Estratégia, e 19 relatos de Novo Público. Do total de trabalhos relacionados, 17 foram publicados em 2015 e 14 em 2016.
1	Ortiz e Pereira (2017)	característica comum do público da EJA é a presença de diversos contextos econômicos, sociais e culturais dentre os alunos. A falta de vivência com a tecnologia costuma ser bastante observada neste público, sendo, portanto, uma área de conhecimento não dominada por eles. Desta forma, desenvolver um projeto de ensino de fundamentos computacionais visando a construção do pensamento computacional neste público é um desafio.
1	Ortiz e Pereira (2017)	Conhecimento não se ensina, se constrói. Do ponto de vista pedagógico, de acordo com Paulo Freire (1996, p.24) "ensinar não é transferir conhecimento, mas criar possibilidades para a sua produção ou a sua construção", portanto, este parece um objetivo de pesquisa promissor: criar possibilidades para a construção de entendimento sobre o pensamento computacional, de forma individual e socialmente compartilhada, levando em consideração as particularidades deste público.

(continua)

Tabela 27 – Textos do WAlgProg (2015-2017) que usam o termo "pensamento computacional" entre onze e vinte e nove

(continuação)

quant.	autor	parágrafo
1	Ortiz e Pereira (2017)	O pensamento computacional auxilia na compreensão e resolução de diversos problemas, relacionados ou não com a computação. Embora muitas pesquisas estejam sendo feitas nesta área no Brasil, o mapeamento apresentado neste artigo mostra que essas pesquisas não têm contemplado o público da educação de jovens e adultos. Esse público possui a necessidade de se atualizar em diversos contextos da sociedade moderna. É um público que, geralmente, precisa trabalhar para o sustento de sua casa e família e, por meio dos estudos, busca obter uma melhor posição social.
3	Ortiz e Pereira (2017)	Por ser um público que possui características diferenciadas e contextos socioculturais diferenciados, o desenvolvimento de uma pesquisa neste público revela muitos desafios, demandando a elaboração de estratégias para cumprir o objetivo proposto. O desenvolvimento do pensamento computacional neste público significa, além de aquisição de conhecimento e habilidades, uma chance de se atualizar sobre os recursos tecnológicos disponíveis, tornando os alunos mais preparados para lidar com eles, preparando-os para uma sociedade mediada por tecnologia. Este artigo apresentou uma discussão sobre o desenvolvimento do pensamento computacional no ensino de jovens e adultos com o intuito de sensibilizar para a importância de se investigar, propor e experimentar formas de criar possibilidades para a construção do pensamento computacional, expondo também desafios e oportunidades oriundas das limitações e particularidades deste público.
4	Pinheiro, Franco e Leite (2017)	Neste sentido, diversas metodologias vêm sendo empregadas com intuito de aproximar crianças e adolescentes com a área tecnológica por intermédio da aprendizagem de programação associada a ferramentas, ambientes de aprendizagem e metodologias, como Scratch [Bezerra & Dias, 2014], Lightbot [Cassenote & Antoniazzi, 2015] e Computação Desplugada [BELL et al, 2011]. Estes destacam-se por associar elementos lúdicos ao ensino de programação, permitindo que pessoas de diferentes idades experimentem o ato de programar. Desse modo, a promoção de iniciativas que desmistificam a ideia de que a programação é algo restrito e específico para estudiosos da área de Computação, reforçando a ideia de que o ato de programar é uma habilidade inerente ao mundo contemporâneo, considerando as contribuições dos avanços tecnológicos para o desenvolvimento da sociedade. Sobre o Pensamento Computacional, sobretudo, cabe destacar que Pensamento Computacional é uma habilidade fundamental para todos, não somente para cientistas da computação. À leitura, escrita e aritmética, deveríamos incluir Pensamento Computacional na habilidade analítica de todas as crianças. (...) envolve a resolução de problemas, projeção de sistemas, e compreensão do comportamento humano, através da extração de conceitos fundamentais da Ciência da Computação. O Pensamento Computacional inclui uma série de ferramentas mentais que refletem a vastidão do campo da Ciência da Computação [WING, 2016].

(continua)

Tabela 27 – Textos do WAlgProg (2015-2017) que usam o termo "pensamento computacional" entre onze e vinte e nove

(continuação)

quant.	autor	parágrafo
2	Pinheiro, Franco e Leite (2017)	Neste sentido, este artigo é resultado da experiência de estudantes do curso de Licenciatura em Computação em um projeto de extensão, cuja principal objetivo foi disseminar conhecimentos sobre a Computação a estudantes da rede pública na faixa etária de 11 a 18 anos, principalmente o público feminino. As propostas de ação foram articuladas para despertar a curiosidade acerca das possibilidades de uso das tecnologias, ao trabalhar conceitos basilares de programação, evidenciando as diferentes vertentes da área. Deste modo, buscou-se promover o desenvolvimento do raciocínio lógico, Pensamento Computacional e abstração para resolução de problemas. Suscitadas as discussões sobre a masculinização das ciências exatas e suas tecnologias, sobretudo, da Computação, foi traçado um caminho para superação da instrumentalização do ambiente educativo como espaço de mera formação de mão de obra, numa perspectiva de educação transformadora, além de possibilitar não apenas o consumo da tecnologia, mas a participação dos sujeitos no processo de desenvolvimento tecnológico e na construção do Pensamento Computacional.
7	Pinheiro, Franco e Leite (2017)	Ainda nesta perspectiva, Cavalcante e Costa [2016] trazem contribuições voltadas às competências utilizadas em um curso de programação, que propiciam o desenvolvimento do Pensamento Computacional. Como elemento de investigação, os autores utilizam um framework responsável por avaliar a eficácia de ambientes de programação em blocos no desenvolvimento do Pensamento Computacional. Nos resultados obtidos, os autores apresentam a eficiência do framework e as práticas elencadas no ambiente relacionadas ao Pensamento Computacional não reconhecidas pelo framework. Por fim, os autores propõem a análise de outros cursos e outros frameworks relacionados ao Pensamento Computacional. França e Tedesco [2015] corroboram os ideais de Cavalcante e Costa quando discutem os desafios que circundam o ensino do Pensamento Computacional na educação básica brasileira; evidenciam diferentes pesquisas na área da Computação sobre o Pensamento Computacional, além de apresentar abordagens que possibilitam a utilização da programação enquanto meio de desenvolver o Pensamento Computacional no ensino médio, trazendo como referencial pesquisas desenvolvidas em diferentes estados do Brasil.
1	Pinheiro, Franco e Leite (2017)	Desenvolvido pela instituição sem fins lucrativos Code.org (http://www.code.org), o Hora do Código é um ambiente virtual cujo intuito é permitir que professores, estudantes e pais, tenham o primeiro contato com o mundo da programação de forma lúdica e eficaz, desmistificando a ideia de que programar é algo difícil. Além de viabilizar o ensino de programação, as atividades disponíveis no ambiente apresentam intervenções subsidiadas pela técnica da Computação Desplugada – "atividades desenvolvidas com o objetivo de ensinar os fundamentos da Ciência da Computação sem a necessidade de computadores" [BELL et al, 2011], visando o desenvolvimento de habilidades como o Pensamento Computacional, raciocínio lógico e abstração.

(continua)

Tabela 27 – Textos do WAlgProg (2015-2017) que usam o termo "pensamento computacional" entre onze e vinte e nove

(continuação)

quant.	autor	parágrafo
1	Pinheiro, Franco e Leite (2017)	Ficou evidenciado que a inserção das tecnologias no ambiente escolar pode também estimular o desenvolvimento do raciocínio lógico-matemático, a criatividade, a abstração e o Pensamento Computacional. Neste sentido, deve-se destacar a necessidade da base metodológica, sem a qual não seria possível utilizar, de maneira eficaz, as ferramentas escolhidas para as atividades de programação. Corrobora esta perspectiva as observações sobre o uso inicial e familiarização com o ambiente Hora do Código, quando foi notado que os participantes buscaram meios de solucionar os problemas apresentados por experimentação, via tentativa e erro. Diante deste cenário, foi necessário intervir com provocações aos participantes sobre as resoluções por eles apresentada.
1	Pinheiro, Franco e Leite (2017)	Com este trabalho, promoveu-se o incentivo à participação de grupos pouco representados na área da Computação, evidenciando as vantagens no desenvolvimento de crianças e adolescentes propiciados pela aprendizagem de conceitos da área e a construção do Pensamento Computacional. Por fim, deve-se destacar a importância do conjunto de atividades desenvolvidas no projeto para os participantes, bem como salientar a necessidade de mais iniciativas semelhantes. O despertar do interesse dos estudantes, em especial, as meninas, pode fazer toda diferença na transformação das suas realidades.
1	Bauer *et al.* (2017)	A maioria dos pesquisadores descrevem os benefícios do ensino de programação em escolas primárias. Mas, muitas escolas brasileiras não oferecem esse conhecimento regularmente. Então, este artigo reporta a experiência em um projeto de extensão para ensino de programação à adolescentes. Pensamento Computacional, Práticas Colaborativas e Aprendizagem Baseada em Problemas usando a plataforma MIT App Inventor, foram usadas como metodologia. As avaliações preliminares indicam que a prática apresenta resultados satisfatórios e pode ser oferecida para outras escolas e cidades para melhorar o desenvolvimento do raciocínio lógico e interesse na área computacional.
1	Bauer *et al.* (2017)	O pensamento computacional consiste na união de fundamentos e conceitos da Ciência da Computação com o pensamento crítico, utilizando técnicas e metodologias presentes dessa área. Wing (2006) descreve diferentes abordagens, entre elas, o ensino de programação nas escolas. Atualmente, o acesso a este conhecimento não está popularizado nas instituições brasileiras, principalmente nas cidades interioranas. Percebe-se que o contato regular dos jovens com programação é restrito somente aos cursos de informática, seja nível técnico ou superior.
1	Bauer *et al.* (2017)	O restante deste trabalho está organizado como segue. Na Seção 2, apresentam-se conceitos relacionados ao ensino de programação e pensamento computacional nas escolas. Tendo visto isso, a Seção 3 descreve as metodologias que vieram a ser aplicadas neste projeto. A Seção 4 é reservada ao relato das experiências das atividades realizadas. Por fim, na Seção 5, são discorridas as avaliações finais do projeto e a Seção 6 apresenta as considerações finais.

(continua)

Tabela 27 – Textos do WAlgProg (2015-2017) que usam o termo "pensamento computacional" entre onze e vinte e nove

(continuação)

quant.	autor	parágrafo
2	Bauer *et al.* (2017)	Nesse mesmo sentido, a Sociedade Brasileira de Computação (2017) incentiva o ensino do pensamento computacional desde a educação básica, já que evolui consideravelmente a capacidade de dedução e conclusão de problemas (COSTA et al., 2016). Recentemente, a SBC elaborou um documento que propõe referenciais de formação em computação para a educação básica, tendo como eixos norteadores o Pensamento Computacional, o Mundo Digital e a Cultura Digital. Nesse documento o ensino de programação surge como uma das atividades essenciais para desenvolver habilidades de resolução de problemas e a fluência no mundo digital.
2	Bauer *et al.* (2017)	A metodologia empregada reúne conceitos do Pensamento Computacional, assim como a realização de atividades com Aprendizagem Baseada em Problemas (ABP) e Práticas Colaborativas. Segundo Andrade et al. (2013), pensamento computacional é um processo de raciocínio que se origina da combinação do pensamento crítico com os fundamentos da computação envolvido na formulação de problemas e das suas soluções, estando diretamente vinculada à Aprendizagem Baseada em Problemas.
1	Bauer *et al.* (2017)	O primeiro encontro teve por objetivo a apresentação dos discentes ministrantes e dos alunos participantes, além de descrever o Projeto codIFic@r e seu propósito. Para aproximação foi feita uma roda de conversa, em que cada pessoa declarou seu nome, idade, ano que estudava, o que esperava do curso, qual seu nível de conhecimento em informática e o nível de utilização de aplicativos em celulares. Após as devidas apresentações foi mostrado vídeos de motivação do movimento Code.Org, no qual celebridades mundiais relatam a relevância do ensino de programação e pensamento computacional nas escolas e como o desenvolvimento de software pode mudar o mundo.
1	Bauer *et al.* (2017)	A metodologia, baseada no Pensamento Computacional, engloba desafios para resolução em grupo e também um concurso final para promoção de criatividade, coautoria e autonomia aos alunos. As limitações descritas são quanto aos equipamentos do laboratório de informática da escola.
1	Bauer *et al.* (2017)	A comunidade escolar, assim como o corpo diretivo da escola participante do projeto avaliaram de forma positiva a ação uma vez que se pôde observar resultados como o ingresso de alunos participantes do projeto em cursos técnicos a nível médio oferecidos pelo IFFar no ano seguinte. Dessa forma, além da exploração do pensamento computacional nas escolas, o projeto propiciou a divulgação dos cursos de Informática e consequentemente o interesse de mais jovens pela área.

Fonte: Autoria própria

APÊNDICE C – PENSAMENTO COMPUTACIONAL: TEXTOS DO WALGPROG (2015-2017) QUE USAM O TERMO "PENSAMENTO COMPUTACIONAL" ACIMA DE 30 OCORRÊNCIAS

Tabela 28 – Textos do WAlgProg que usam o termo "pensamento computacional" mais que trinta vezes

quant.	autor	parágrafo
3	Farias, Andrade e Alencar (2015)	As aceleradas e complexas transformações na sociedade demandam por um perfil profissional diferenciado, detentor de um abrangente conjunto de habilidades e competências alinhadas ao pensamento analítico e estatístico, as quais não se limitam a manipulação avançada de editores de textos ou planilhas eletrônicas, e, neste sentido, o pensamento computacional apresenta subsídios ricos para estimular e fomentar tais habilidades e competências. Considerando este contexto, o presente artigo versa sobre a adequação da formação docente para esta realidade, bem como sobre as possibilidades e os desafios encontrados para a apresentação dos conceitos pertinentes ao pensamento computacional no cotidiano de sala de aula para as novas gerações. Para aferir estes cenários, foi aplicado um questionário direcionado para alunos concluintes do curso de Licenciatura em Computação, visto que estes são oportunos para compreensão do grau de ciência sobre a importância do Pensamento Computacional.
1	Farias, Andrade e Alencar (2015)	A comunidade acadêmica está convergindo cada vez mais para a primordial importância da ampliação do ensino de conceitos da Computação, contemplando alunos desde a tenra idade, tornando assim, uma ciência basilar para a formação de habilidades primárias para bom desempenho de qualquer profissão. Exemplos de iniciativas neste cenário são encontrados em alguns estados brasileiros como: Rio de Janeiro (Souza et al., 2014), Minas Gerais (Carvalho et al., 2013), Paraíba (Scaico et al., 2013), Rio Grande do Sul (Andrade et al., 2013), Amazonas (Vieira et al., 2013), Pernambuco (França et al., 2012) e Bahia (Sousa et al., 2010). Neste aspecto, uma aptidão que vem ganhando robustez na aplicação junto aos alunos do ensino fundamental e médio se dá com o Pensamento Computacional.
3	Farias, Andrade e Alencar (2015)	Corriqueiramente são encontradas várias definições sobre tal estratégia cognitiva. A mais oportuna que mantém consonância com a investigação aqui apresentada, é desenhada por Qin (2009), no qual o pensamento computacional como um arranjo de pensamento que aplica conceitos e metodologias da Computação para solucionar problemas em um amplo espectro de assuntos oferecendo, então, um arcabouço de habilidades essenciais para qualquer ciência contemporânea. Neste aspecto, o Pensamento Computacional seria uma espécie de canivete suíço cognitivo, que analogamente, pertenceria ao kit de sobrevivência intelectual, capaz de ofertar um conjunto de habilidades que potencializa a resolução de problemas complexos e/ou desconhecido aplicando raciocínio computacional. Tamanha é a riqueza educacional do Pensamento Computacional, que variadas publicações estão crescendo nos últimos tempos no cenário internacional, (Barr and Stephenson 2011, Denning 2009, Hu 2011, Wing 2006, Wing 2008), bem como no nacional (França 2014, Gomes 2013, Carvalho 2013, Farias, 2012).

(continua)

Tabela 28 – Textos do WAlgProg que usam o termo “pensamento computacional” mais que trinta vezes

(continuação)

quant.	autor	parágrafo
2	Farias, Andrade e Alencar (2015)	Partindo desta necessidade, a presente investigação surgiu da inquietação sobre o brio, a visão diferenciada que os formandos em Licenciatura em Ciência da Computação (LCC) têm para acompanhar atividades de Pensamento Computacional junto aos alunos de formação do Ensino Fundamental e Média. Portanto, este trabalho tem como objetivo delinear a formação docente atual, e para isto, foi realizada uma pesquisa com alunos concluintes do curso de LCC da Universidade Federal Rural de Pernambuco. Acredita- se que esta análise contribuirá para que a comunidade acadêmica reflita com mais profundidade sobre a qualidade da formação docente nos cursos de LCC, no tocante ao aparato conceitual que estes futuros profissionais deverão possuir para ter capacidade de fomentar atividades imersas no Pensamento Computacional.
1	Farias, Andrade e Alencar (2015)	Castro (2013) faz uma observação salutar, ao afirmar categoricamente, que a informática em uma escola, carece de um profissional formado em Licenciatura em Computação, sendo necessário mudança nestes cenários com celeridade. Aqui é válido destacar quão basilar é a formação destes profissionais no tocante ao componente de Pensamento Computacional, uma vez que estes conceitos contribuem para a adoção de um profissional habilitado para lecionar com propriedade os conceitos Computacionais.
2	Farias, Andrade e Alencar (2015)	Como objeto de estudo dessa pesquisa, foram analisadas algumas turmas concluintes do curso de Licenciatura em Computação da modalidade a distância da Universidade Federal Rural de Pernambuco, que cursaram a disciplina optativa de Informática na Educação. Este curso possui carga horária de 2925h, e está distribuído entre os 15 polos nas seguintes cidades do estado de Pernambuco: Afrânio, Gravatá, Limoeiro, Pesqueira, Surubim, Cabrobó, Ipojuca, Olinda, Petrolina, Tabira, Carpina, Jaboatão dos Guararapes, Palmares, Recife e Trindade. A análise realizada incluiu a coleta e análise de dados realizados a partir das respostas de um questionário aplicado com estes alunos. O questionário abrangeu 13 perguntas que contemplaram o conhecimento do aluno sobre os conceitos que circundam o Pensamento Computacional, a identificação da experiência profissional e as limitações para implantar atividades voltadas para o desenvolvimento deste conceito. As perguntas são detalhadas a seguir: O conhecimento do aluno sobre os conceitos que circundam o Pensamento Computacional
1	Farias, Andrade e Alencar (2015)	1. Você já havia escutado este termo antes, Pensamento Computacional? Se sua resposta for "não", por favor, passe para a questão 4.
2	Farias, Andrade e Alencar (2015)	3. Para você o que significa pensamento computacional na prática? Para você o que significa pensamento computacional?
1	Farias, Andrade e Alencar (2015)	8. Você apresenta conceitos de pensamento computacional para seus alunos?
1	Farias, Andrade e Alencar (2015)	11. Você acredita que apresentar os conceitos de pensamento computacional despertam maior interesse nos alunos?
1	Farias, Andrade e Alencar (2015)	O questionário tem duas perguntas abertas, desta forma é possível compreender o sentimento do aluno com mais precisão. Elas são as seguintes: (i) para você o que significa pensamento computacional na prática? (ii) você enfrenta alguma dificuldade para inserção desses conceitos na sala de aula? Se sua resposta foi sim, o que motiva estas dificuldades?

(continua)

Tabela 28 – Textos do WAlgProg que usam o termo "pensamento computacional" mais que trinta vezes

(continuação)

quant.	autor	parágrafo
1	Farias, Andrade e Alencar (2015)	Após a coleta e compilação das respostas do questionário, foram realizadas algumas análises com o objetivo de traçar o perfil dos alunos que estão concluindo o curso superior em Licenciatura em Computação, a adequação do curso e da visão do estudante sobre a compreensão dos conceitos de Pensamento Computacional e as expectativas da aplicação em sala de aula. As subseções a seguir irão detalhar essa análise.
1	Farias, Andrade e Alencar (2015)	Ficou evidente desde a primeira pergunta que, quase metade dos alunos entrevistados (43,8
1	Farias, Andrade e Alencar (2015)	Em relação aos conceitos sobre Pensamento Computacional (Pergunta 3), 31,2
1	Farias, Andrade e Alencar (2015)	Aluno A: Compreende que Pensamento Computacional é tão simplesmente pensar nas tecnologias contemporâneas com o objetivo de ser inserido na atual dinâmica mercadológica.
1	Farias, Andrade e Alencar (2015)	Aluno B: Entende que abstrair a realidade por meio de uma interação lógica e objetiva para auxiliar as atividades que são necessárias, se traduz em Pensamento Computacional.
1	Farias, Andrade e Alencar (2015)	Aluno C: Evolução de todos os sentidos sensoriais e cognitivos caracteriza Pensamento Computacional
1	Farias, Andrade e Alencar (2015)	Aluno D: Tem a clara função da ferramenta computacional na sociedade, é Pensamento Computacional
1	Farias, Andrade e Alencar (2015)	Aluno E: Usar a ferramenta computacional para solucionar problemas é compreendido como Pensamento Computacional
1	Farias, Andrade e Alencar (2015)	Em relação aos alunos que não conhecem o termo "Pensamento Computacional", foi direcionada a pergunta para compreender atuação como docente em escolas (Pergunta 4), 75por opção, 25
3	Farias, Andrade e Alencar (2015)	A pergunta 6 só foi respondida pelos entrevistados que já atuam na docência em escolas públicas ou privadas, dos que responderam 12,5
1	Farias, Andrade e Alencar (2015)	Acentua-se aqui, quão necessário é o amadurecimento da compreensão sobre o que de fato inviabiliza a prática docente com o pensamento computacional. Aliado com a argumentação de Perris (2013), o desafio não é a utilização de novas tecnologias para reproduzir os sistemas tradicionais de ensino, mas fomentar novos espaços de aprendizagem, proporcionando melhorias para professores e alunos, e, por conseguinte melhorar a qualidade da educação.
2	Farias, Andrade e Alencar (2015)	Na pergunta 10, com o objetivo de compreender de quem partiu o estímulo fomentador das atividades de Pensamento Computacional, foi possível compreender que 20
2	Farias, Andrade e Alencar (2015)	Nesta investigação, aspectos que compõe a realidade vivida pelos estudantes concluintes do curso de LCC foram coletados e analisados para averiguar qual o poder de compreensão e formação que estes possuem sobre tópicos de Pensamento Computacional. Notou-se que os estudantes possuem clara limitação conceitual sobre o componente, tendo respostas fundamentadas em crenças, destoando do que de fato a literatura instrui. Existe pouca atividade por parte destes concluintes, na promoção do Pensamento Computacional com estudantes do ensino fundamental e médio.

(continua)

Tabela 28 – Textos do WAlgProg que usam o termo "pensamento computacional" mais que trinta vezes

(continuação)

quant.	autor	parágrafo
2	Farias, Andrade e Alencar (2015)	Esta investigação não tem o propósito de ficar apenas limitada nas falhas metodológicas, culminando no singelo dinamismo na promoção do Pensamento Computacional. Ao observar que as habilidades de uma turma de alunos concluintes estão carentes em seu fundamento basilar, acredita-se que tal cenário não foi estimado quando formularam os cursos de LCC no Brasil. Por tal, fazer uma profunda consideração sobre o que de fato espera-se de um profissional em LCC poderá ser determinante para a propagação do Pensamento Computacional, ou capaz deste marasmo ceifar o potencial cognitivo de uma geração que tende em atingir patamares mais sofisticados que as gerações contemporâneas.
1	Farias, Andrade e Alencar (2015)	Como meta futura, pretende-se aprimorar o questionário aplicado nesta investigação, para ser mais abrangente, pois se tem o intuito de coletar mais informações de outros estados do Nordeste, e caso exista viabilidade, cogita-se aplicar em cursos do Brasil. Assim, buscaremos ter um comparativo mais holístico da real formação dos docentes em Licenciatura em Computação no tocante ao Pensamento Computacional.
1	Farias, Andrade e Alencar (2015)	Andrade, D., Carvalho, T., Silveira, J., Cavalheiro, Foss, L., Fleischmann, A. M., Aguiar, M., Reiser, R. (2013). Proposta de Atividades para o Desenvolvimento do Pensamento Computacional no Ensino Fundamental. In: XVI WIE, SBC.<
1	Farias, Andrade e Alencar (2015)	Carvalho, M. L. B.; Chaimowicz, L.; Moro, M.. M. (2013) Pensamento Computacional no Ensino Médio Mineiro. In: pp. 640-649, Anais do XXI WEI, CSBC, SBC.</reference>
1	Farias, Andrade e Alencar (2015)	França, R. S, Ferreira, V.A.S., Almeida, L.C.F., Amaral, H.J.C.(2014) A disseminação do pensamento computacional na educação básica: lições aprendidas com experiências de licenciandos em computação. CSBC -WEI, SBC.
1	Farias, Andrade e Alencar (2015)	Gomes, T. C. S., Melo, J. C. B. (2013). O Pensamento Computacional no Ensino Médio: Uma Abordagem Blended Learning. In: XXI WEI, SBC.
1	Barcelos *et al.* (2015)	Relações entre o Pensamento Computacional e a Matemática: uma Revisão Sistemática da Literatura
2	Barcelos *et al.* (2015)	Uma melhor compreensão das relações entre o Pensamento Computacional e disciplinas já presentes no currículo da educação básica pode contribuir para a identificação de possíveis benefícios educacionais. Dessa forma, este artigo apresenta uma Revisão Sistemática da Literatura (RSL), incluindo 48 estudos publicados em língua inglesa entre 2006 e 2014 que apresentam atividades didáticas desenvolvendo o Pensamento Computacional e competências, habilidades ou conteúdos da Matemática. Vários tópicos matemáticos são desenvolvidos, com predominância da Álgebra, Cálculo e habilidades cognitivas de alto nível. Verifica-se ainda, nos últimos dois anos, um aumento do desenvolvimento de experiências na educação básica e um maior rigor metodológico na avaliação dos efeitos de aprendizagem. Por outro lado, há uma carência de estudos relacionados à Modelagem Matemática e à formação de professores.

(continua)

Tabela 28 – Textos do WAlgProg que usam o termo “pensamento computacional” mais que trinta vezes

(continuação)

quant.	autor	parágrafo
1	Barcelos *et al.* (2015)	Hood e Hood afirmam que a chave para obter a fluência dos estudantes em Tecnologia da Informação é incorporá-la ao currículo escolar (HOOD; HOOD, 2005). Porém tal fluência não deveria ocorrer por intermédio do domínio de um conjunto de técnicas, mas sim por uma organização do pensamento voltada à resolução de problemas. Por essa razão, um conjunto de competências e habilidades associadas à Ciência da Computação deveria ser desenvolvido pelos estudantes desde os primeiros anos da educação básica. Wing nomeou tal conjunto como Pensamento Computacional (WING, 2006). Esse termo é atualmente utilizado para descrever os processos cognitivos relacionados à abstração e decomposição para permitir a resolução de problemas utilizando recursos computacionais e estratégias algorítmicas, dentre outras habilidades.
2	Barcelos *et al.* (2015)	Para incorporar atividades de desenvolvimento do Pensamento Computacional na educação básica é necessário considerar qual seria o impacto de tais atividades na aprendizagem dos conteúdos escolares tradicionais. Isso é particularmente importante por ser o ensino básico, em vários países, um ambiente no qual várias prioridades, ideologias e filosofias lutam por atenção (CSTA, 2011). Relações entre a Matemática e a Ciência da Computação no ambiente educacional já foram discutidas anteriormente (KE, 2014; KRONE; SITARAMAN; HALLSTROM, 2011). Dessa forma, é razoável inferir que o processo de ensino-aprendizagem de Matemática seria beneficiado de alguma forma pela incorporação do Pensamento Computacional ao currículo do ensino básico.
2	Barcelos *et al.* (2015)	Muitas atividades visando o desenvolvimento do Pensamento Computacional associado a competências, habilidades e conteúdos matemáticos foram reportadas na literatura nos últimos anos, o que motiva uma análise detalhada dos resultados educacionais obtidos por intermédio dessas iniciativas. Dessa forma, este artigo apresenta uma revisão sistemática da literatura com o objetivo de identificar como as relações entre a Matemática e o Pensamento Computacional foram demonstradas por meio do desenvolvimento de atividades didáticas descritas na literatura. A revisão foi guiada pelas seguintes perguntas de pesquisa:
2	Barcelos *et al.* (2015)	(Q1) Como atividades didáticas relacionadas ao Pensamento Computacional e à Matemática foram desenvolvidas e qual seu público-alvo? (Q2) Que competências, habilidades e conteúdos do Pensamento Computacional e da Matemática são ensinados nas atividades didáticas? (Q3) Que métodos e técnicas de pesquisa são empregados para identificar possíveis efeitos de aprendizagem?
1	Barcelos *et al.* (2015)	O passo seguinte está relacionado com a definição do protocolo da revisão (Atividade 1.3). Segundo Wohlin et al. (2012) o protocolo deve indicar a estratégia de pesquisa, que inclui os repositórios de pesquisa dos quais serão extraídos os estudos, definição de strings de busca e critérios de inclusão e exclusão. Cinco repositórios foram incluídos nesta revisão: ACM, IEEE Xplore, ERIC, ScienceDirect e SpringerLink. A pesquisa sobre as características e atividades do Pensamento Computacional é razoavelmente recente e, por isso, optamos por usar uma string de busca abrangente: "computational thinking"and ("math"or "mathematics"). O objetivo foi identificar a maior quantidade de estudos possível, disponíveis em língua inglesa, mesmo à custa de uma seleção manual mais trabalhosa.

(continua)

Tabela 28 – Textos do WAlgProg que usam o termo “pensamento computacional” mais que trinta vezes

(continuação)

quant.	autor	parágrafo
1	Barcelos *et al.* (2015)	Foram definidos dois critérios de inclusão. Para ser incluído na revisão, o estudo deveria (CI1) indicar alguma relação entre o Pensamento Computacional e alguma competência, habilidade ou tópico relacionado à Matemática. Ainda, para ser incluído na revisão um estudo deveria (CI2) descrever uma atividade didática e apresentar os resultados da sua avaliação. Como optamos por definir uma estratégia de busca inclusiva, foram incluídos tanto estudos que apresentaram resultados de experimentos controlados quanto estudos que apresentavam apenas evidências empíricas. Essa estratégia foi adotada para identificar quão precisa foi a identificação dos resultados educacionais. Da mesma forma, foram excluídos estudos que atendiam a algum dos critérios de exclusão abaixo:
1	Barcelos *et al.* (2015)	(CE2) Pensamento Computacional (ou o ensino de Computação) é o tema do artigo, no entanto não são apresentadas relações com a Matemática;
1	Barcelos *et al.* (2015)	Os 77 artigos incluídos foram então separados em dois grupos: o primeiro grupo (EXP), com 48 artigos, incluiu os estudos que descreveram a aplicação de uma atividade didática acompanhada de uma avaliação experimental dos resultados de aprendizagem. O segundo grupo (DI) foi formado por 29 artigos nos quais os autores apresentavam discussões conceituais das relações entre o Pensamento Computacional e a Matemática com base em seu ponto de vista, análise documental ou outro tipo de estudo teórico. Também foram incluídos neste grupo os artigos que descreviam propostas de atividades didáticas sem nenhuma descrição de sua aplicação ou avaliação. Por limitação de espaço, a partir deste ponto serão apresentadas apenas as análises realizadas nos 48 artigos do primeiro grupo.
2	Barcelos *et al.* (2015)	É possível identificar que uma expressiva quantidade de estudos (26 artigos) apresentam experiências voltadas a um ou mais níveis da educação básica. Por outro lado, foram identificados 16 artigos que tem como alvo exclusivamente o ensino de graduação, ou tiveram sua validação experimental realizada neste ambiente. Apesar da definição original de Pensamento Computacional (WING, 2006) considerar o seu desenvolvimento como uma proposta voltada à educação básica, muitos pesquisadores vem também propondo atividades voltadas ao desenvolvimento do Pensamento Computacional de forma aliada com a Matemática para alunos de graduação. Tais atividades são predominantemente voltadas para disciplinas introdutórias, visando atenuar os altos índices de evasão e reprovação em tais disciplinas. No entanto, também é necessário ressaltar que o interesse da comunidade em realizar experimentos na educação básica vem crescendo: dos 26 artigos que relatam tais experiências, 10 (38.4

(continua)

Tabela 28 – Textos do WAlgProg que usam o termo "pensamento computacional" mais que trinta vezes

(continuação)

quant.	autor	parágrafo
1	Barcelos *et al.* (2015)	As experiências didáticas que desenvolvem a Matemática juntamente com o Pensamento Computacional vêm utilizando, predominantemente, ferramentas computacionais durante o seu desenvolvimento. A partir do mapeamento das ferramentas e materiais utilizados nos artigos, foram identificadas 42 ocorrências do uso de ferramentas computacionais e apenas 10 ocorrências do uso de atividades que não dependem do computador. Na Tabela 3 são apresentados as ferramentas de software utilizadas e os materiais físicos utilizados, bem como as respectivas frequências de utilização. É preciso ressaltar que um estudo pode apresentar mais de uma ocorrência de uso de ferramentas de software e materiais.
1	Barcelos *et al.* (2015)	A seguir foram analisadas as habilidades e conteúdos relacionados à Matemática que vem sendo desenvolvidos em conjunto com o Pensamento Computacional nos estudos realizados. A partir do agrupamento dos estudos que desenvolvem temas semelhantes foi possível identificar oito grupos de habilidades e conteúdos desenvolvidos:
1	Barcelos *et al.* (2015)	A partir da análise dos estudos incluídos nesta revisão é possível inferir que há um crescente interesse da comunidade científica em explorar as relações entre o Pensamento Computacional e a Matemática. O aumento significativo, nos últimos anos, na quantidade de estudos nos quais atividades didáticas são apresentadas e testadas empiricamente contribui para essa conclusão.
4	Barcelos *et al.* (2015)	Apesar de, originalmente, o conceito de Pensamento Computacional referir-se a habilidades que deveriam ser desenvolvidas por estudantes da educação básica, verificou-se que as atividades didáticas descritas não se restringem apenas esse nível de ensino: cerca de 33% das atividades didáticas relatadas foram oferecidas a estudantes de graduação. Por um lado, isso indica que o desenvolvimento do Pensamento Computacional tem se mostrado útil na organização das pesquisas que tem como finalidade resolver os problemas de evasão e reprovação em cursos de graduação em Computação. Entretanto, isso não deixa de ser um "desvio" nos objetivos originais do Pensamento Computacional, já que o ensino de graduação é tipicamente o campo de aplicação mais próximo da realidade da pesquisa nas universidades. Algumas experiências didáticas apresentadas nos estudos têm como objetivo a formação de professores para disseminar o desenvolvimento do Pensamento Computacional; no entanto, tais iniciativas ainda são muito preliminares (apenas 4 estudos em um universo de 48) e necessitam de uma maior sistematização.

(continua)

Tabela 28 – Textos do WAlgProg que usam o termo "pensamento computacional" mais que trinta vezes

(continuação)

quant.	autor	parágrafo
2	Barcelos *et al.* (2015)	Verifica-se um aumento do interesse da comunidade pela condução de experimentos didáticos na educação básica nos últimos dois anos (2013 e 2014) abrangidos pela revisão. Da mesma forma, o rigor metodológico empregado na avaliação da aprendizagem também parece ter aumentado nos estudos publicados nos últimos dois anos. As atividades didáticas desenvolvidas se relacionam a uma grande variedade de conteúdos matemáticos, utilizando ferramentas computacionais também bastante diversas. Esse é um indicativo da flexibilidade e potencial dos conceitos e ferramentais computacionais como suporte para ensinar e contextualizar a Matemática. Essa evidência contradiz parcialmente a revisão da literatura apresentada por (GROVER; PEA, 2013), que menciona que o Pensamento Computacional não vinha sendo utilizado para ensinar outras disciplinas. Entretanto, deve-se notar a carência de trabalhos que utilizam a estratégia da Modelagem Matemática, visto que a construção e interpretação de modelos podem ser consideradas como uma habilidade comum à Matemática e ao Pensamento Computacional (BARCELOS; SILVEIRA, 2012; LEE et al., 2011).
3	Barcelos *et al.* (2015)	O Pensamento Computacional representa um conjunto de habilidades relacionadas à Ciência da Computação que deveriam ser desenvolvidas pelos estudantes da educação básica. No entanto, se mostra necessário compreender as relações do Pensamento Computacional com as disciplinas tradicionais do currículo escolar e quais os possíveis benefícios do desenvolvimento de estratégias didáticas conjuntas. Neste artigo apresentamos uma revisão sistemática da literatura de estudos publicados entre 2006 e 2014 que abordam as relações entre o Pensamento Computacional e a Matemática e cujos resultados permitem identificar os avanços e limitações da pesquisa nessa área.
1	Barcelos *et al.* (2015)	A maioria das experiências didáticas descritas tem como público-alvo os alunos da educação básica, mas uma parcela relevante das experiências foi desenvolvida com alunos de graduação. Por outro lado, há relatos insuficientes de experiências desenvolvidas para formação inicial e continuada de professores. Uma grande variedade de tópicos da Matemática vem sendo abordados, com alguma predominância para a Álgebra e o Cálculo. Vários estudos procuram desenvolver conjuntamente o Pensamento Computacional a Matemática por meio de habilidades de alto nível compartilhadas entre os dois paradigmas de pensamento, mas poucos estudos utilizam a construção e avaliação de modelos matemáticos e computacionais.
1	Barcelos *et al.* (2015)	É possível identificar um claro avanço na disponibilidade e variedade de atividades didáticas que envolvem o Pensamento Computacional e a Matemática. No entanto, ainda há públicos-alvo e habilidades matemáticas que vem sendo pouco exploradas pela comunidade. Por meio da identificação dessas limitações, esperamos que novos estudos venham a suprir essas lacunas. Em trabalhos futuros, pretende-se desenvolver uma análise mais aprofundada das técnicas de pesquisa empregadas nos estudos incluídos na revisão, bem como estendê-la incluindo também estudos publicados em português e espanhol.

(continua)

Tabela 28 – Textos do WAlgProg que usam o termo "pensamento computacional" mais que trinta vezes

(continuação)

quant.	autor	parágrafo
1	Barcelos *et al.* (2015)	BARCELOS, T. S.; SILVEIRA, I. F. Pensamento Computacional e Educação Matemática: relações para o ensino de Computação na educação básica. XX Workshop sobre Educação em Computação. Anais do XXXII Congresso da Sociedade Brasileira de Computação. Curitiba: SBC, 2012.
1	França e Tedesco (2015)	Desafios e oportunidades ao ensino do pensamento computacional na educação básica no Brasil
2	França e Tedesco (2015)	Resumo. Atualmente é requerido dos estudantes desenvolver diversas habilidades, dentre elas o pensamento computacional. Contudo, no Brasil o ensino de tal habilidade não integra o currículo escolar. Nesse sentido, este artigo discute desafios ao ensino do pensamento computacional na educação básica brasileira, apresentando oportunidades de pesquisa na área. Além disso, é apresentada uma proposta para minimizar alguns dos problemas apontados, a qual demonstrou contribuir com a aprendizagem de estudantes do nível médio durante um curso de desenvolvimento de jogos digitais.
1	França e Tedesco (2015)	Ao falar-se de Computação enquanto saber necessário na educação básica deve-se ressaltar que seu ensino passa por noções de modelos computacionais, algoritmos, complexidade, autômatos, entre outros conteúdos (NUNES, 2008). Assim, o ensino à manipulação de softwares, tais como editores de texto, não é suficiente na atualidade. A habilidade em questão tem sido chamada de pensamento computacional (WING, 2006) e deve integrar a formação básica dos cidadãos do século XXI possibilitando que futuros jornalistas, músicos, advogados, dentre outros, resolvam problemas de suas respectivas áreas de atuação.
2	França e Tedesco (2015)	Ensinar habilidades computacionais na educação básica no Brasil pode, portanto, configurar-se um desafio e apresentar-se como um cenário repleto de oportunidades aos educadores, pesquisadores e comunidade escolar. Nesse cenário, este trabalho discute desafios enfrentados no ensino do pensamento computacional na educação básica brasileira apontando para oportunidades de pesquisa na área. Além disso, é apresentada uma proposta passível de ser integrada na sala de aula que promove a aprendizagem do pensamento computacional pela prática de programação no contexto do ensino médio.
1	França e Tedesco (2015)	O restante do artigo está organizado como segue: na seção 2 são apresentadas experiências de ensino do pensamento computacional na educação básica brasileira e na seção 3 são apontados desafios ao ensino de tal habilidade, considerando-se o atual currículo escolar. Ainda, na seção 4 é apresentado um caminho para superar alguns dos desafios descritos e, por fim, na seção 5 são feitas algumas considerações finais.
1	França e Tedesco (2015)	2. Pensamento computacional em contextos escolares brasileiros
1	França e Tedesco (2015)	O interesse pela disseminação do pensamento computacional no ambiente escolar é crescente tendo envolvido a participação de pesquisadores e educadores de diferentes países, incluindo o Brasil.

(continua)

Tabela 28 – Textos do WAlgProg que usam o termo "pensamento computacional" mais que trinta vezes

(continuação)

quant.	autor	parágrafo
1	França e Tedesco (2015)	Em Pernambuco, França et al. (2012) e França et al. (2014) relatam experiências vivenciadas por licenciandos em Computação na busca pela disseminação de pensamento computacional na escola, bem como na divulgação dos cursos de Licenciatura em Computação e na desmistificação do papel desses profissionais na sociedade. Já Gomes e Melo (2013) descrevem um curso de lógica de programação ofertado a estudantes do ensino médio no qual foi utilizado o ambiente de programação visual App Inventor (www.appinventor.mit.edu) para criar aplicativos para dispositivos Android.
1	França e Tedesco (2015)	No Rio Grande do Sul, Andrade et al. (2013) propuseram atividades direcionadas ao ensino fundamental que envolvem os princípios do pensamento computacional. Já Vieira et al. (2014) realizaram peças teatrais no Amazonas possibilitando aos participantes compreender os fundamentos da Computação de forma clara e divertida. Por outro lado, de Souza et al. (2014) têm utilizado o AgentSheets (www.agentsheets.com) em suas pesquisas em escolas do Rio de Janeiro possibilitando que os educandos expressem seus interesses e capacidades de comunicação por meio da programação de jogos e simulações.
1	França e Tedesco (2015)	Já no trabalho de Barcelos e Silveira (2013) os autores investigaram, em uma instituição de São Paulo, de que forma as competências relacionadas à Matemática são mobilizadas por estudantes do ensino médio no processo de desenvolvimento do pensamento computacional através da construção de jogos digitais. No mesmo Estado, Garcia et al. (2008) apresentam a experiência de execução de um projeto que visou proporcionar a estudantes do ensino médio a oportunidade de adquirir competências relacionadas ao desenvolvimento do raciocínio lógico, especificamente na resolução de problemas por meio de algoritmos e estruturas de dados; e, como efeito secundário, desejava-se motivar os participantes a continuar seus estudos na área de Computação. Os resultados apontam para o interesse os estudantes nos conteúdos do curso, havendo ainda participação deles em atividades da OBI e o ingresso em cursos superiores da área de Computação.
1	França e Tedesco (2015)	No trabalho de Carvalho et al. (2013) os autores apresentam uma iniciativa para a inserção do ensino do pensamento computacional e de conceitos básicos de tecnologia da informação no ensino médio de escolas de Minas Gerais. O projeto, voltado à empregabilidade e que envolve diversas áreas incluindo Tecnologia da Informação, foi implementado em 2012 em onze escolas da zona norte de Belo Horizonte, abrangendo um total de 5.979 alunos. A expectativa era que em 2014 todas as 2.167 escolas estaduais fizessem parte do projeto, alcançando os 678.684 estudantes do ensino médio estadual.
1	França e Tedesco (2015)	As atividades desenvolvidas, descritas na seção anterior, são, em sua maioria, decorrentes de projetos de pesquisa, ensino e extensão executados por graduandos e pós-graduandos de cursos de Computação. Considerando-se os benefícios que o pensamento computacional pode trazer para a formação dos estudantes da atualidade, discutem-se alguns desafios a serem enfrentados na busca pelo o seu ensino na educação básica no Brasil.

(continua)

Tabela 28 – Textos do WAlgProg que usam o termo "pensamento computacional" mais que trinta vezes

(continuação)

quant.	autor	parágrafo
1	França e Tedesco (2015)	Uma das principais questões a serem consideradas no ensino do pensamento computacional é o currículo escolar. Nesse contexto, enxergam-se duas possibilidades que podem ser abordadas em futuras pesquisas. A primeira diz respeito à criação de uma disciplina obrigatória, assim como Matemática, que trate dos fundamentos da Computação nos diferentes anos da educação básica. Nessa conjectura, deve-se atentar para diversos aspectos, dentre eles, os possíveis efeitos da introdução de uma nova disciplina no currículo escolar na motivação e na aprendizagem dos estudantes.
1	França e Tedesco (2015)	A segunda possibilidade tem uma perspectiva interdisciplinar onde o pensamento computacional é trabalhado atrelado às disciplinas já existentes no currículo escolar. Nesse contexto, a promoção da aprendizagem de conceitos computacionais é feita aliada à construção de conhecimento de conteúdos curriculares dos diferentes anos da educação básica.
2	França e Tedesco (2015)	Além disso, outra questão que deve ser considerada diz respeito ao quê e quando ensinar o pensamento computacional na educação básica. Nesse contexto, é preciso definir diretrizes curriculares para o ensino de tal habilidade nas escolas brasileiras, enfatizando que conceitos devem ser introduzidos e em quais anos escolares. Isto deve levar em consideração que o ensino do pensamento computacional não deve cobrir apenas a manipulação de recursos digitais, mas sim os fundamentos da Computação, enquanto ciência.
2	França e Tedesco (2015)	O ensino do pensamento computacional na educação básica brasileira irá requerer a formação de professores especializados para atuarem nas escolas. Nesse contexto, segundo Nunes (2010), os cursos de Licenciatura em Computação têm uma enorme responsabilidade de formar professores para introduzir a Ciência da Computação na educação básica, disseminando assim o pensamento computacional.
1	França e Tedesco (2015)	Há de se considerar que tais cursos não devem apenas concentrar suas atividades na construção de softwares educativos e no uso de tecnologias aplicadas à educação, mas preparar os futuros licenciados em Computação para que atuem no ensino do pensamento computacional. Isto irá requerer que os cursos ofertados pelas instituições de Educação possibilitem a tais profissionais uma sólida formação em Ciência da Computação, Matemática e Educação, como definido nas Diretrizes Curriculares Nacionais para os cursos de graduação em Computação.
2	França e Tedesco (2015)	Ademais, acredita-se que os outros cursos de licenciatura também poderiam beneficiar-se de uma formação em pensamento computacional. Isto poderia prover aos futuros professores de Matemática, Biologia, e outros, uma formação em Computação que os ajudaria na resolução de problemas de suas respectivas áreas de atuação. Além disso, o desenvolvimento de ações educativas envolvendo o pensamento computacional de modo interdisciplinar também poderia ser favorecido nas escolas brasileiras.

(continua)

Tabela 28 – Textos do WAlgProg que usam o termo "pensamento computacional" mais que trinta vezes

(continuação)

quant.	autor	parágrafo
1	França e Tedesco (2015)	Tem-se observado que diferentes abordagens de ensino têm sido empregadas na educação básica vislumbrando-se o ensino do pensamento computacional. Quando se tem como objetivo promover essa habilidade por meio da lógica de programação, por exemplo, é recorrente o uso de ambientes visuais de programação tais como Scratch (www.scratch.mit.edu/), App Inventor (www.appinventor.mit.edu), AgentSheets (www.agentsheets.com/) e RoboMind (www.robomind.net/pt). Como produtos dessas atividades tem-se, por exemplo, jogos, simulações e animações, criados a partir dos interesses dos estudantes ou a partir de recomendações dos instrutores dos cursos.
1	França e Tedesco (2015)	Ainda constata-se que os princípios da Computação podem ser ensinados aos estudantes sem depender do uso de uma tecnologia específica. Nesse contexto, destaca-se o trabalho de Bell et al. (2011) que propuseram um conjunto de atividades lúdicas envolvendo fundamentos da Ciência da Computação, publicadas no livro Computer Science Unplugged. Tais atividades estimulam o pensamento computacional, sem o uso do computador, e têm despertado o interesse de pesquisadores e professores e empregadas em diversos países, entre eles o Brasil.
1	França e Tedesco (2015)	Assim, em atividades que busquem promover a aprendizagem do pensamento computacional, diversos fatores devem ser considerados, dentre eles os interesses dos aprendizes o que resulta em um desafio que educadores e pesquisadores devem se preocupar: como ensinar conceitos fundamentais da Computação considerando o público-alvo e suas peculiaridades provendo mecanismos para uma aprendizagem significativa?
1	França e Tedesco (2015)	Tendo em vista os desafios relacionados aos processos de ensino e aprendizagem de Computação, uma abordagem é apresentada para minimizar tais problemas. Trata-se de um modelo para a aprendizagem do pensamento computacional, intitulado penC
1	França e Tedesco (2015)	Ainda na última fase, os estudantes podem monitorar seu processo de aprendizagem do pensamento computacional através de diferentes atividades, baseadas na precisão no monitoramento do conhecimento (Knowledge Monitoring Accuracy – KMA) definido por Tobias e Everson (2002) e no viés no monitoramento do conhecimento (Knowledge Monitoring Bias – KMB) definido por Gama (2004).
1	França e Tedesco (2015)	Como forma identificar o impacto do modelo penC na autorregulação e na aprendizagem do pensamento computacional, um quasi-experimento foi realizado em um curso de desenvolvimento de jogos digitais que introduziu conceitos de lógica de programação em iniciantes na área.
1	França e Tedesco (2015)	Nas atividades propostas em tal curso era possível aos estudantes construir jogos de acordo com seus interesses pessoais, tendo, no entanto, que atender a requisitos mínimos, envolvendo conteúdos de lógica de programação. Buscou-se, com isso, manter o engajamento dos educandos nas atividades, ao mesmo tempo em que aprendiam conceitos norteadores do pensamento computacional. Na Figura 2 são exibidos alguns jogos digitais desenvolvidos pelos estudantes com o uso do Stencyl (www.stencyl.com).

(continua)

Tabela 28 – Textos do WAlgProg que usam o termo "pensamento computacional" mais que trinta vezes

(continuação)

quant.	autor	parágrafo
1	França e Tedesco (2015)	A partir de tal experimento foi possível observar resultados positivos da abordagem proposta, frente à tradicional. Em relação à aprendizagem do pensamento computacional, o modelo penC trouxe contribuições na formação dos estudantes, sendo observadas melhorias em seu desempenho em habilidades de depuração e criação de códigos, consideradas importantes para os programadores. Na variável autorregulação da aprendizagem também se constatou efeito positivo no grupo experimental, sugerindo que o penC pode contribuir para o desenvolvimento de tal competência.
1	França e Tedesco (2015)	O pensamento computacional é uma das habilidades a serem desenvolvidas pelos estudantes do século XXI. Contudo, seu ensino ainda não integra o currículo escolar brasileiro, resultando em diversos desafios a serem enfrentados por pesquisadores e comunidade escolar.
2	França e Tedesco (2015)	Nesse contexto, este trabalho apresentou iniciativas realizadas em diversos Estados, enfatizando desafios e oportunidades para o ensino do pensamento computacional no Brasil. Frente aos desafios discutidos foi apresentada uma proposta que pode ajudar a superar problemas identificados, ajudando estudantes da educação básica em sua formação em conceitos computacionais. O quasi-experimento descrito foi realizado no contexto de um curso de desenvolvimento de jogos digitais. Contudo, isto não limita a utilização do penC em intervenções didáticas que tratem o pensamento computacional de modo interdisciplinar.
1	França e Tedesco (2015)	Espera-se que as questões aqui expostas corroborem para a educação em Computação no cenário brasileiro, provendo insights para realização de futuras pesquisas sobre a disseminação do pensamento computacional nos níveis fundamental e médio.
1	França e Tedesco (2015)	ANDRADE, Daiane et al. Proposta de Atividades para o Desenvolvimento do Pensamento Computacional no Ensino Fundamental. In: Anais do XVI Workshop de Informática na Escola. SBC, 2013.
1	França e Tedesco (2015)	BARCELOS, T. S.; SILVEIRA, I. F. Relações entre o pensamento computacional e a matemática através da construção de jogos digitais. In: Anais do XII Simpósio Brasileiro de Jogos e Entretenimento Digital, 2013.
1	França e Tedesco (2015)	CARVALHO, M. L. B.; CHAIMOWICZ, L.; MORO, M. M. Pensamento Computacional no Ensino Médio Mineiro. In: Anais do XXI Workshop sobre Educação em Computação. 2013.
1	França e Tedesco (2015)	FRANÇA, R. S. ; FERREIRA, V. F. S.; ALMEIDA, L. C. F. ; AMARAL, H. J. C. A disseminação do pensamento computacional na educação básica: lições aprendidas com experiências de licenciandos em computação. In: Anais do XXII Workshop sobre Educação em Computação. SBC, 2014.
1	França e Tedesco (2015)	FRANÇA, R. S. Um modelo para a aprendizagem do pensamento computacional aliado à autorregulação. 2015. Dissertação (Mestrado em Ciência da Computação) – Centro de Informática, Universidade Federal de Pernambuco, Recife. 2015.
1	França e Tedesco (2015)	GOMES, T. C. S.; MELO, J. C. B. O Pensamento Computacional no Ensino Médio: Uma Abordagem Blended Learning In: Anais do XXI Workshop sobre Educação em Computação. SBC, 2013.
1	Cavalcante, Costa e Araujo (2016)	Um Estudo de Caso Sobre Competências do Pensamento Computacional Estimuladas na Programação em Blocos no Code.Org

(continua)

Tabela 28 – Textos do WAlgProg que usam o termo "pensamento computacional" mais que trinta vezes

(continuação)

quant.	autor	parágrafo
1	Cavalcante, Costa e Araujo (2016)	Resumo. Este trabalho apresenta um estudo de caso qualitativo sobre as competências do pensamento computacional exploradas em um curso de programação introdutória da plataforma Code.org. Os resultados demonstram quais as competências relacionadas aos conceitos, práticas e perspectivas computacionais modeladas em um framework puderam ser encontradas no curso.
2	Cavalcante, Costa e Araujo (2016)	O pensamento computacional se caracteriza como um processo com vista à resolução de problemas por meio de conceitos, recursos e ferramentas computacionais. O pensamento computacional baseia-se em fundamentos da computação, envolvendo a resolução de problemas, a capacidade de projetar sistemas e a compreensão do comportamento humano [Wing, 2006]. É uma forma de aplicar conceitos trabalhados na computação, mas que não são exclusivos somente desta área, pois podem ser aplicados na resolução de problemas dos mais variados campos do saber. A aplicabilidade transversal e multidisciplinar ressignifica o "pensar computacionalmente" como uma competência fundamental para todas as pessoas, não apenas para profissionais da computação, despontando como um requisito elementar para a formação básica dos profissionais de todas as áreas nos próximos anos.
1	Cavalcante, Costa e Araujo (2016)	O ensino de programação introdutória é apontado como uma abordagem para estimular o pensamento computacional no contexto escolar. Nos últimos anos foram criadas plataformas de ensino de programação que adotam uma estratégia composta por traços lúdicos, ambientação amigável e programação através de blocos. Scratch (https://scratch.mit.edu/), App Inventor (http://appinventor.mit.edu/), Alice (http://www.alice.org/index.php), Blockly Games (https://blockly-games.appspot.com) são exemplos dessas plataformas. Um outro projeto que vem se destacando é a plataforma de ensino de programação Code.org (http://www.code.org). O projeto tem como perspectiva trazer os conceitos básicos da Ciência da Computação, principalmente da programação, para o ambiente escolar.
2	Cavalcante, Costa e Araujo (2016)	Apesar do grande número de usuários, são poucas as pesquisas nacionais sobre as vantagens do ensino de programação ofertada pela plataforma [Dantas e Costa, 2013]. Sobretudo, são escassos os trabalhos que apontam evidências sobre o estímulo do pensamento computacional por meio dos cursos da plataforma Code.org [Kalelioğlu, 2015] . Assim, desejamos responder a seguinte questão de pesquisa: quais competências do pensamento computacional são estimuladas por meio de cursos de introdução à programação ofertados pela plataforma Code.org?
2	Cavalcante, Costa e Araujo (2016)	Nesse contexto, este trabalho exploratório visa identificar e analisar competências do pensamento computacional estimuladas através de um curso de programação oferecido pela plataforma Code.org. Para atingir esse objetivo, selecionamos um framework que avalia competências do pensamento computacional no contexto de ambientes de programação em blocos e identificamos e analisamos seus elementos em um curso.

(continua)

Tabela 28 – Textos do WAlgProg que usam o termo "pensamento computacional" mais que trinta vezes

(continuação)

quant.	autor	parágrafo
2	Cavalcante, Costa e Araujo (2016)	O presente trabalho está organizado da seguinte maneira: a Seção 2 apresenta brevemente competências do pensamento computacional; a Seção 3 descreve o framework que avalia competências do pensamento computacional no contexto de ambientes de programação em blocos; a Seção 4 cita trabalhos relacionados; a Seção 5 descreve os procedimentos adotados; a Seção 6 analisa os resultados; a Seção 7 conclui o trabalho e apresenta trabalhos futuros.
1	Cavalcante, Costa e Araujo (2016)	2. Competências do Pensamento Computacional
1	Cavalcante, Costa e Araujo (2016)	Competência é a capacidade de utilizar mais de um recurso para resolver algo de forma inovadora, criativa e no momento necessário. Já a habilidade pode ser entendida como uma capacidade individual de resolver problemas utilizando conhecimentos previamente estabelecidos, na qual, a partir deles, podemos realizar induções e deduções com o objetivo de resolvê-los [Bonotto e Felicette, 2014]. Assim, competência consiste em um conceito mais abrangente no qual há um conjunto de habilidades e atitudes vinculadas na realização de uma ação. Por isso, usaremos o termo competências do pensamento computacional no restante do texto.
2	Cavalcante, Costa e Araujo (2016)	A Computer Science Teacher Association1 (CSTA) e a International Society for Technology in Education (ISTE) organizaram uma definição operacional que auxilia na identificação de competências do pensamento computacional. Desta forma, o pensamento computacional poderá ser desenvolvido a partir de atividades de: Formular problemas de forma que nos permita usar um computador e outras ferramentas para ajudar a resolvê-los; Organizar e analisar dados logicamente; Representar dados através de abstrações tais como modelos e simulações; Propor soluções através do pensamento algorítmico; Identificar, analisar e implementar as soluções combinando recursos de forma eficiente e eficaz; Generalizar e transferir um processo de solução de um problema para outros.
1	Cavalcante, Costa e Araujo (2016)	Essa definição operacional é suportada e aprimorada por um conjunto de disposições ou atitudes que são essenciais para o pensamento computacional. Tais atitudes, conforme , incluem1: Confiança em lidar com a complexidade; Persistência ao trabalhar com problemas difíceis; Tolerância em lidar com ambiguidade; Capacidade de lidar com problemas em aberto; Capacidade de se comunicar e trabalhar em grupo para atingir um objetivo ou solução em comum.
1	Cavalcante, Costa e Araujo (2016)	3. Framework para identificar competências do pensamento computacional na programação introdutória
1	Cavalcante, Costa e Araujo (2016)	O framework proposto por Brennan e Resnick (2012) permite a compreensão das competências do pensamento computacional que podem ser exploradas em ambiente de programação introdutória como o Scratch. O framework está dividido em três elementos: conceitos computacionais, práticas computacionais e perspectivas computacionais. A Figura 1 resume as competências exploradas em cada parte.

(continua)

Tabela 28 – Textos do WAlgProg que usam o termo "pensamento computacional" mais que trinta vezes

(continuação)

quant.	autor	parágrafo
2	Cavalcante, Costa e Araujo (2016)	Kalelioğlu (2015) investigou os efeitos que oficinas code.org provocaram na habilidade de resolução de problemas em estudantes com 10 anos de idade. Os resultados apontaram que as meninas tiveram maior impacto positivo na resolução de problema em comparação ao meninos. Já Israel et al (2015) investigou como professores da educação básica com pouca experiência em computação poderiam integrar o pensamento computacional em sua prática pedagógica. Os autores utilizaram oficinas do Code.org e outras ferramentas como instrumento para introduzir conceitos de programação e pensamento computacional aos professores.
2	Cavalcante, Costa e Araujo (2016)	Iniciamos o estudo com uma pesquisa bibliográfica no intuito de identificar as habilidades e competências do pensamento computacional estimuladas pelos ambientes de programação introdutória em blocos. Em seguida, pesquisamos modelos de avaliação de competências do pensamento computacional no contexto de programação introdutória em blocos.
2	Cavalcante, Costa e Araujo (2016)	Dentre os frameworks para avaliação do pensamento computacional, o de Brennan e Resnick (2012) apresentou-se como o mais abrangente para avaliação de competências do pensamento computacional em programação em blocos. Embora ele tenha sido concebido sob a perspectiva do ambiente Scratch, avaliamos sua aplicabilidade e limitações no contexto de um curso da plataforma Code.org. Além disso, não encontramos na literatura um framework adequado ao contexto dos cursos do Code.org.
1	Cavalcante, Costa e Araujo (2016)	6. Análise das competências do pensamento computacional no Code.org
1	Cavalcante, Costa e Araujo (2016)	Discutimos nesta seção as competências do pensamento computacional no curso 2 da plataforma Code.org sob a ótica do framework de Brennan e Resnick (2012).
1	Cavalcante, Costa e Araujo (2016)	Este trabalho apresentou um estudo exploratório sobre as competências do pensamento computacional estimuladas em um curso de programação introdutória da plataforma Code.org. Os resultados demonstram que as competências modeladas no framework de Brennan e Resnick (2012) puderam ser aplicadas no curso, com possibilidade de expansão nos quesitos de eficiência e reconhecimento de padrões.
1	Cavalcante, Costa e Araujo (2016)	Destacamos que o estímulo à eficiência (utilizar a menor quantidade de blocos necessários para solução) é uma prática presente no curso. Isso contribui para que o aluno planeje melhor a construção de sua abordagem de resolução do problema. O reconhecimento de padrões é outra prática estimulada no curso durante a introdução do conceito de repetição. Ambas as práticas não são destacadas no framework. Assim, elas poderão ser incorporadas em outro modelo de avaliação de pensamento computacional em cursos de programação introdutória baseados em blocos.
1	Cavalcante, Costa e Araujo (2016)	Em trabalhos futuros, planejamos analisar outros cursos da plataforma, sob outras perspectivas do pensamento computacional e utilizando outros frameworks como modelo. Em outra vertente, também planejamos aplicar e coletar dados de alunos durante a execução do curso, em busca de evidências empíricas das perspectivas computacionais, bem como identificar outras habilidades específicas que são estimuladas no desenvolvimento do curso.

(continua)

Tabela 28 – Textos do WAlgProg que usam o termo "pensamento computacional" mais que trinta vezes

(continuação)

quant.	autor	parágrafo
1	Cavalcante, Costa e Araujo (2016)	2. Competências do Pensamento Computacional
1	Cavalcante, Costa e Araujo (2016)	3. Framework para identificar competências do pensamento computacional na programação introdutória
1	Cavalcante, Costa e Araujo (2016)	6. Análise das competências do pensamento computacional no Code.org
1	Araujo, Andrade e Guerrero (2016)	Um Mapeamento Sistemático sobre a Avaliação do Pensamento Computacional no Brasil
2	Araujo, Andrade e Guerrero (2016)	Resumo. Após dez anos da disseminação do termo Pensamento Computacional pela pesquisadora Jannette Wing, são diversas as iniciativas no intuito de estimulá-lo e avaliá-lo. Neste trabalho realizamos um mapeamento sistemático de literatura com o objetivo de identificar como o pensamento computacional é estimulado e avaliado no Brasil. Os resultados apontaram que programação, testes e códigos são as abordagens e instrumentos mais utilizados. Discutimos ainda sobre a granularidade das avaliações e apontamos lacunas de pesquisa na área.
6	Araujo, Andrade e Guerrero (2016)	Pensamento computacional é compreendido como um processo de resolução de problemas, projeto de sistemas e compreensão do comportamento humano norteados por conceitos fundamentais da Ciência da Computação [Wing, 2006]. Ao longo desta década, no âmbito nacional, as iniciativas em discutir e promover pensamento computacional estão principalmente concentradas nos trabalhos de pesquisa e extensão das universidades e faculdades. Encontramos pesquisas interessadas em saber quais ferramentas têm contribuído para disseminação do pensamento computacional [Bombasar et al., 2015] e como o pensamento computacional tem sido relacionado a outras disciplinas, como a matemática [Barcelos et al., 2015; Mestre et al., 2015]. Outros estudos têm apontado os desafios e possibilidades do ensino de pensamento computacional nas escolas e a formação de licenciados em computação [França e Tedesco, 2015; Farias et al., 2015]. Entretanto, poucas pesquisas têm focado na forma de avaliação do pensamento computacional.
1	Araujo, Andrade e Guerrero (2016)	Ademais, no pensamento computacional desejamos avaliar construtos. Construtos são variáveis latentes as quais não podem ser observadas diretamente. No nosso contexto, são habilidades cognitivas que permitem resolver problemas de forma sistemática. Por isso, precisamos de instrumentos ou artefatos adequados capazes de fornecer indícios que nos permitam mensurar o progresso dessas habilidades cognitivas.
3	Araujo, Andrade e Guerrero (2016)	Este trabalho tem o objetivo de identificar o estado da arte no Brasil sobre avaliação do pensamento computacional. Para ter uma visão abrangente desse assunto, desejamos responder as seguintes questões de pesquisa: QP1 - Quais são as abordagens de estímulo ao pensamento computacional identificadas no Brasil? QP2 - Quais são as habilidades avaliadas no Brasil? QP3 - Quais são os instrumentos e/ou artefatos selecionados para avaliar o progresso do pensamento computacional no Brasil?
1	Araujo, Andrade e Guerrero (2016)	Para responder essas questões de pesquisa, realizamos um mapeamento sistemático de literatura nos principais eventos e revistas nacionais na área de Informática na Educação. Sumarizamos os resultados e discutimos (i) a programação como estímulo e avaliação do pensamento computacional, (ii) a granularidade das avaliações e (iii) a replicabilidade de estudos empíricos.

(continua)

Tabela 28 – Textos do WAlgProg que usam o termo "pensamento computacional" mais que trinta vezes

(continuação)

quant.	autor	parágrafo
2	Araujo, Andrade e Guerrero (2016)	Selby e Woollard (2012) propuseram uma definição para o pensamento computacional associada às principais habilidades baseada em uma revisão de literatura. Para eles, o pensamento computacional é um processo cognitivo que reflete as habilidades de pensar de forma abstrata e algorítmica, saber decompor atividades complexas, bem como realizar avaliações e generalizações.
1	Araujo, Andrade e Guerrero (2016)	Bombasar et al. (2015) executaram uma revisão sistemática de literatura para evidenciar as ferramentas utilizadas no ensino do pensamento computacional. Os resultados apontaram que as ferramentas de linguagem de programação visual, como o Scratch, são as mais empregadas nesse contexto.
1	Araujo, Andrade e Guerrero (2016)	Barcelos et al. (2015) realizaram uma revisão sistemática no intuito de conhecer as atividades pedagógicas envolvendo pensamento computacional e matemática. Eles descobriram que dentre os diversos assuntos da matemática, há predominância de Álgebra, Cálculo e habilidades cognitivas de alto nível nos trabalhos reportados. Eles também relataram que, com relação à avaliação dos efeitos da aprendizagem, 69
1	Araujo, Andrade e Guerrero (2016)	Encontramos diversos trabalhos que promovem o pensamento computacional principalmente em alunos da educação básica. No intuito de ter uma visão abrangente e compreender melhor as abordagens, as habilidades mensuradas e a maneira como elas são avaliadas, organizamos as seguintes questões de pesquisa (QP):
1	Araujo, Andrade e Guerrero (2016)	QP1: Quais são as abordagens de estímulo ao pensamento computacional identificadas no Brasil? O propósito é identificar as abordagens pedagógicas que promovem o pensamento computacional para apontar as habilidades avaliadas em cada abordagem.
1	Araujo, Andrade e Guerrero (2016)	QP3: Quais são os instrumentos e/ou artefatos selecionados para avaliar o progresso do pensamento computacional no Brasil? O objetivo é apontar os instrumentos e artefatos empregados pelos pesquisadores para avaliar habilidades nos sujeitos.
1	Araujo, Andrade e Guerrero (2016)	Uma vez definidas as questões de pesquisa, seguimos para a próxima etapa do protocolo que define a condução da pesquisa. A chave de busca escolhida para selecionar os artigos foi "pensamento computacional" e "raciocínio computacional" (entre aspas). Optamos por usar os termos que caracterizam essa área no intuito de abranger o maior número possível de trabalhos publicados e ter uma visão ampla do estado da arte.
1	Araujo, Andrade e Guerrero (2016)	Selecionamos os cinco principais eventos e revista de publicação nacional na área de informática na educação: Simpósio Brasileiro de Informática na Educação (SBIE), Workshop de Informática na Escola (WIE), Workshop sobre Educação em Computação (WEI), Workshop de Ensino de Pensamento Computacional, Algoritmos e Programação (WAlgProg) e a Revista Novas Tecnologias na Educação (RENOTE). Pesquisamos a chave de busca nos motores de busca disponíveis nos anais online desses eventos/revista nos campos de busca de título e resumo. A exceção ocorreu no WEI, em que foi realizada uma busca manual devido a ausência de motores de busca nos anais online. Esta etapa resultou 32 artigos. Desses, 11 artigos são do WAlgProg, 11 do WIE, 5 do SBIE, 4 do WEI e 1 da RENOTE.

(continua)

Tabela 28 – Textos do WAlgProg que usam o termo "pensamento computacional" mais que trinta vezes

(continuação)

quant.	autor	parágrafo
3	Araujo, Andrade e Guerrero (2016)	Para selecionar os artigos relevantes para as questões de pesquisa, foram definidos os critérios de inclusão e exclusão. Como critério de inclusão utilizamos artigos (completos ou curtos) que apresentem um método, proposta ou prática de avaliação do pensamento computacional. Os critérios de exclusão utilizados foram: 1) Artigos que não descrevem uma forma de avaliação do pensamento computacional; 2) Artigos que apresentem avaliação de ferramenta e não de habilidades do pensamento computacional estimuladas pela ferramenta; 3) Artigos com abordagem/modelo repetido; 4) Artigos com revisões de literatura.
1	Araujo, Andrade e Guerrero (2016)	Em seguida, todos os 32 artigos foram lidos e aplicados os critérios de inclusão e exclusão. No caso de artigos com repetição do modelo proposto, foi considerado o estudo mais completo. Já os artigos que apresentam proposta de intervenção curricular, histórico de abordagem de ensino sem avaliação, ou que realizaram a análise de conhecimento de estudantes/profissionais sobre o pensamento computacional e revisões de literatura foram excluídos, pois não apresentavam uma forma de avaliação de habilidades. Após essa etapa, 22 artigos foram selecionados e analisados. A última etapa do protocolo de pesquisa foi a extração e o mapeamento dos dados, detalhados na Seção 4.
1	Araujo, Andrade e Guerrero (2016)	O resultado do mapeamento sistemático apontou crescimento de interesse em pensamento computacional nos últimos anos, conforme mostra a Figura 1(a). Em 2015 foram publicados 15 artigos e não encontramos nenhuma publicação antes de 2011.
2	Araujo, Andrade e Guerrero (2016)	4.2. QP1: Quais são as abordagens de estímulo ao pensamento computacional? As abordagens de estímulo ao pensamento computacional encontradas nos artigos resultantes deste mapeamento sistemático foram sumarizadas no Quadro 1.
1	Araujo, Andrade e Guerrero (2016)	Quadro 1. Abordagens de estímulo ao Pensamento Computacional
2	Araujo, Andrade e Guerrero (2016)	O resultado do nosso estudo apontou que programação é a abordagem mais empregada para estimular o pensamento computacional (11/22 artigos). Scratch se destacou como a mais usada tanto para oficinas de introdução à programação como para o desenvolvimento de jogos e animações (7/11 artigos de programação). Atividades desplugadas foi a segunda abordagem mais popular (8/22 artigos), tanto na proposta de novas atividades como no relato de experiências de aplicação com crianças. Ainda observamos três artigos abordando ferramenta e jogos, um com robótica, um com provas do PISA (Programme for International Student Assessment) e uma proposta de modelo de avaliação do pensamento computacional em contexto de programação. Ressaltamos que o mesmo artigo pode apresentar duas ou mais abordagens diferentes.
1	Araujo, Andrade e Guerrero (2016)	Nosso estudo apontou para ampla variedade de habilidades associadas ao pensamento computacional, sobretudo a conceitos e práticas de programação. Na extração das habilidades explícitas em cada artigo, buscamos mapeá-las como estavam descritas e detalhamos no Quadro 2.
1	Araujo, Andrade e Guerrero (2016)	Lógica, Sequenciamento, Reconhecimento de Padrões, Controle de Fluxo, Pensamento Algorítmico e Solução de Problemas
1	Araujo, Andrade e Guerrero (2016)	4.4. QP3: Quais são os instrumentos e/ou artefatos para avaliar o progresso do pensamento computacional?

(continua)

Tabela 28 – Textos do WAlgProg que usam o termo "pensamento computacional" mais que trinta vezes

(continuação)

quant.	autor	parágrafo
3	Araujo, Andrade e Guerrero (2016)	Nosso estudo apontou que teste foi o instrumento mais utilizado pelos pesquisados para avaliar o pensamento computacional (9/22 artigos). Teste inclui questionários, pré e pós testes elaborados pelos próprios autores, bem como provas como ENEM e PISA. Em segundo lugar, códigos e projetos foram os artefatos mais usados para avaliar pensamento computacional, considerando análise das estruturas de programação presentes/ausentes ou o processo de desenvolvimentos. Encontramos ainda avaliação observacional, qualitativas (com discussão das atividades e análise dos artefatos produzidos pelos alunos), grupo focal e avaliação de jogos que estimulam o pensamento computacional (tanto observacional como teórico). O Quadro 3 mostra o mapeamento dos instrumentos e/ou artefatos retornados pela pesquisa.
3	Araujo, Andrade e Guerrero (2016)	Verificamos o crescente interesse nacional em iniciativas que estimulem o pensamento computacional. Só em 2015 foram publicados 15 artigos, provavelmente encorajados pelo WAlgProg, um workshop para discussão de ensino de pensamento computacional e programação. Esse workshop pode se configurar como um importante evento nacional para discussão sobre como promover e avaliar o pensamento computacional.
1	Araujo, Andrade e Guerrero (2016)	5.1. Programação como estímulo e avaliação do pensamento computacional
1	Araujo, Andrade e Guerrero (2016)	Os resultados apontaram que programação tem se revelado uma abordagem habitual tanto para disseminar o pensamento computacional como para avaliar a aprendizagem de pensar computacionalmente por parte do aluno. Dentre as linguagens/plataformas mais frequentemente selecionada para ensinar programação, Scratch tem se destacado.
1	Araujo, Andrade e Guerrero (2016)	Para mensurar o pensamento computacional nos projetos Scratch, os artigos têm empregado avaliação manual de código. Mesmo os artigos que usam frameworks de avaliação de outros trabalhos, o principal critério de avaliação é a ausência/presença de estruturas de programação para determinar o nível de fluência do aluno em determinado
1	Araujo, Andrade e Guerrero (2016)	Uma ferramenta que pode auxiliar na avaliação automática nesse contexto é o Dr. Scratch1 [Moreno, 2014]. Dr. Scratch é uma ferramenta web gratuita, automática e em português para avaliar pensamento computacional em projetos Scratch. Uma vez carregado o projeto na ferramenta via navegador, o Dr. Scratch fornece uma pontuação no tocante a abstração, decomposição de problema, paralelismo, pensamento lógico, sincronismo, controle de fluxo, iteratividade do usuário e representação de dados. Essa ferramenta não foi utilizada por nenhum artigo resultante do mapeamento sistemático, provavelmente por ser recente (lançamento em 2014). Entretanto, ela pode ser útil para trabalhos futuros que desejem uma avaliação automática de projetos Scratch.
1	Araujo, Andrade e Guerrero (2016)	Reconhecemos que programação é uma abordagem promissora para estimular e avaliar o desenvolvimento de habilidades do pensamento computacional, mas indagamos a reflexão: quais habilidades ficam de fora do escopo de programação introdutória? Quais habilidades não podem ser medidas por meio de programação? Quais outras abordagens poderiam complementar o progresso dessas habilidades cognitivas no contexto de resoluções de problemas do mundo real?

(continua)

Tabela 28 – Textos do WAlgProg que usam o termo "pensamento computacional" mais que trinta vezes

(continuação)

quant.	autor	parágrafo
2	Araujo, Andrade e Guerrero (2016)	Percebemos que os artigos reportam diferentes granularidades no tocante a avaliação das habilidades associadas ao pensamento computacional. Quando o artigo considera o desempenho final (ou nota) em um teste ou um programa, podemos considerar que ele está usando granularidade alta para avaliar o pensamento computacional. Já quando o artigo considera as habilidades avaliadas separadamente, podemos considerar granularidade fina.
1	Araujo, Andrade e Guerrero (2016)	Ambos os casos devem ser considerados para avaliação do pensamento computacional, ressaltando os benefícios e limitações de cada um. Para o exemplo de desempenho final (granularidade alta), os resultados são mais diretos para serem sumarizados e aplicados testes estatísticos. Já para casos em que são avaliados habilidades específicas (granularidade fina), a avaliação pode ser mais custosa, considerando a seleção de ferramentas e critérios estabelecidos para realizar a mesuração de cada habilidade.
1	Araujo, Andrade e Guerrero (2016)	Uma avaliação que considere ambos os casos de granularidade seria ideal para considera o resultado geral (conjunto de habilidades) bem como as habilidades específicas que se deseja mensurar. Entretanto, na prática, construir instrumentos que avaliem construtos do pensamento computacional (separadamente e em conjunto), e simultaneamente, sejam empiricamente validados é uma questão em aberto.
2	Araujo, Andrade e Guerrero (2016)	Avaliar construtos é uma atividade complexa, uma vez que não podemos medi-los diretamente. No pensamento computacional, os construtos são habilidades cognitivas de alto nível que auxiliam na resolução de problemas. Nesse contexto, este trabalho realizou um mapeamento sistemático de literatura no intuito de conhecer como os pesquisadores e professores avaliam pensamento computacional no Brasil.
2	Araujo, Andrade e Guerrero (2016)	Programação foi a abordagem de estímulo ao pensamento computacional mais retornado, seguido de atividades desplugadas (QP1). Encontramos uma ampla gama de habilidades para avaliar o pensamento computacional, com destaque aos conceitos e práticas de programação (QP2). Os principais instrumentos e artefatos que avaliam essas habilidades são testes e códigos/projetos (QP3).
1	Araujo, Andrade e Guerrero (2016)	O fato de programação ter sido apontada como principal abordagem de estímulo ao pensamento computacional foi um resultado esperado, visto que os estudos foram realizados por pesquisadores de computação. Em decorrência, tem-se o impacto tanto nas habilidades que são avaliadas, ou seja, habilidades diretamente relacionadas ao ato de programar, bem como ao código como artefato de avaliação. Destacamos que mesmo sendo teste o instrumento mais utilizado para avaliação, alguns deles não são passíveis de serem reutilizados em replicação de experimento ou em novos estudos, pois não estão disponíveis na íntegra.
4	Araujo, Andrade e Guerrero (2016)	primeira é a possibilidade de pesquisas que explorem oficinas de pensamento computacional para licenciandos, explorando as habilidades do pensamento computacional em casos reais de suas respectivas disciplinas. A segunda é o desenvolvimento de jogos ou aplicativos que pudessem estimular e ao mesmo tempo avaliar o progresso do pensamento computacional. E a terceira é a criação e validação de instrumentos para avaliar o pensamento computacional os quais não fossem dependentes de programação.

(continua)

Tabela 28 – Textos do WAlgProg que usam o termo "pensamento computacional" mais que trinta vezes

(continuação)

quant.	autor	parágrafo
2	Araujo, Andrade e Guerrero (2016)	Por último, indagamos as seguintes questões: considerando a complexidade de avaliar construtos, seria possível construir instrumentos capazes de avaliar as habilidades de pensamento computacional no contexto de resolução de problemas? Seria possível avaliar as habilidades separadamente? Quais seriam as métricas para determinar o nível de proficiência em pensamento computacional?
1	Araujo, Andrade e Guerrero (2016)	Andrade, D., Carvalho, T., Silveira, J., Cavalheiro, S., Foss, L., Fleischmann, A. M., & Reiser, R. (2013). Proposta de atividades para o desenvolvimento do pensamento computacional no ensino fundamental. In: Anais do XXI Workshop de Informática na Escola – WEI. p. 169.
1	Araujo, Andrade e Guerrero (2016)	Barcelos, T., Muñoz, R., Acevedo, R. V., & Silveira, I. F. (2015). Relações entre o Pensamento Computacional e a Matemática: uma Revisão Sistemática da Literatura. In: Anais dos Workshops do IV Congresso Brasileiro de Informática na Escola.
1	Araujo, Andrade e Guerrero (2016)	Bombasar, J.; Raabe, A.; Miranda, E. M. e Santiago, R. (2015). Ferramentas para o ensino-aprendizagem do pensamento computacional: onde está Alan Turing? In: Anais do XXVI Simpósio Brasileiro de Informática na Escola. p. 81.
1	Araujo, Andrade e Guerrero (2016)	Campos, G. M., Cavalheiro, S., Foss, L., Pernas, A. M., de Brum Piana, C. F., Aguiar, M., ... & Reiser, R. (2014). Organização de Informações via Pensamento Computacional: Relato de Atividade Aplicada no Ensino Fundamental. In Anais do XXII Workshop de Informática na Escola - WEI, p. 390.
1	Araujo, Andrade e Guerrero (2016)	Farias, A., Andrade, W., & Alencar, R. (2015). Pensamento Computacional em Sala de Aula: Desafios, Possibilidades e a Formação Docente. In: Anais dos Workshops do IV Congresso Brasileiro de Informática na Educação.
1	Araujo, Andrade e Guerrero (2016)	França, R. S., & do Amaral, H. J. C. (2013). Proposta metodológica de ensino e avaliaçao para o desenvolvimento do pensamento computacional com o uso do scratch. In Anais do XIX Workshop de Informática na Escola. p. 179.
1	Araujo, Andrade e Guerrero (2016)	França, R. S., & Tedesco, P. C. D. A. R. (2014). Um modelo colaborativo para a aprendizagem do pensamento computacional aliado à autorregulação. In XXV Anais do Simpósio Brasileiro de Informática na Educação. p. 1133.
1	Araujo, Andrade e Guerrero (2016)	França, R., & Tedesco, P. (2015). Desafios e oportunidades ao ensino do pensamento computacional na educação básica no Brasil. In: Anais dos Workshops do IV Congresso Brasileiro de Informática na Educação. p. 1464.
1	Araujo, Andrade e Guerrero (2016)	França, R. S., & Tedesco, P. (2015b). Explorando o pensamento computacional no ensino médio: do design à avaliação de jogos digitais. In Anais do XXI Workshop de Informática na Escola – WIE.
1	Araujo, Andrade e Guerrero (2016)	Gomes, T., & Alencar, A. (2015). Análise Empírica de Jogos Educativos para Dispositivos Móveis voltados a Disseminação do Pensamento Computacional na Educação Básica. In Anais dos Workshops do IV Congresso Brasileiro de Informática na Educação. p. 731.

(continua)

Tabela 28 – Textos do WAlgProg que usam o termo "pensamento computacional" mais que trinta vezes

(continuação)

quant.	autor	parágrafo
1	Araujo, Andrade e Guerrero (2016)	Gomes, T., Barreto, P., Lima, I. R. A., & Falcão, T. P. (2015). Avaliação de um Jogo Educativo para o Desenvolvimento do Pensamento Computacional na Educação Infantil. In Anais dos Workshops do IV Congresso Brasileiro de Informática na Educação. p. 1349.
1	Araujo, Andrade e Guerrero (2016)	Hinterholz, L., & da Cruz, M. K. (2015). Desenvolvimento do Pensamento Computacional: um relato de atividade junto ao Ensino Médio, através do Estágio Supervisionado em Computação III. In: XXI Anais do Workshop de Informática na Escola. p. 137.
1	Araujo, Andrade e Guerrero (2016)	Mestre, P., Andrade, W., Guerrero, D., Sampaio, L., da Silva Rodrigues, R., & Costa, E. (2015). Pensamento Computacional: Um estudo empírico sobre as questões de matemática do PISA. In: Anais dos Workshops do IV Congresso Brasileiro de Informática na Escola.
1	Araujo, Andrade e Guerrero (2016)	Ramos, F., & da Silva Teixeira, L. (2015). Significação da Aprendizagem Através do Pensamento Computacional no Ensino Médio: uma Experiência com Scratch. In Anais do XXI Workshop de Informática na Escola. p. 217.
1	Araujo, Andrade e Guerrero (2016)	Rodrigues, R., Andrade, W., Guerrero, D., & Sampaio, L. (2015). Análise dos efeitos do Pensamento Computacional nas habilidades de estudantes no ensino básico: um estudo sob a perspectiva da programação de computadores. In: XXVI Anais do Simpósio Brasileiro de Informática na Educação. p. 121.
1	Araujo, Andrade e Guerrero (2016)	Rodriguez, C., Zem-Lopes, A. M., Marques, L., & Isotani, S. (2015). Pensamento Computacional: transformando ideias em jogos digitais usando o Scratch. In: Anais do XXI Workshop de Informática na Escola. p. 62.
1	Araujo, Andrade e Guerrero (2016)	Schoeffel, P., Moser, P., Varela, G., Durigon, L., de Albuquerque, G. C., & Niquelatti, M. (2015). Uma Experiência no Ensino de Pensamento Computacional para Alunos do Ensino Fundamental. In Anais dos Workshops do IV Congresso Brasileiro de Informática na Educação. p. 1474.
1	Araujo, Andrade e Guerrero (2016)	Zanetti, H., & Oliveira, C. (2015). Práticas de ensino de Programação de Computadores com Robótica Pedagógica e aplicação de Pensamento Computacional. In: Anais dos Workshops do IV Congresso Brasileiro de Informática na Educação. p. 1236.
1	Araujo, Andrade e Guerrero (2016)	4.2. QP1: Quais são as abordagens de estímulo ao pensamento computacional?
1	Araujo, Andrade e Guerrero (2016)	4.4. QP3: Quais são os instrumentos e/ou artefatos para avaliar o progresso do pensamento computacional?
1	Araujo, Andrade e Guerrero (2016)	5.1. Programação como estímulo e avaliação do pensamento computacional
1	Costa *et al.* (2017)	Um Estudo Exploratório da Aplicação de Pensamento Computacional Baseado nas Perspectivas de Professores do Ensino Médio

(continua)

Tabela 28 – Textos do WAlgProg que usam o termo "pensamento computacional" mais que trinta vezes

(continuação)

quant.	autor	parágrafo
2	Costa *et al.* (2017)	Resumo. Este trabalho tem como objetivo conhecer a percepção de professores do Ensino Médio a respeito da aplicação de habilidades do pensamento computacional em sua prática docente. Realizamos uma pesquisa de campo com professores de uma escola pública, na qual aplicamos questionários (survey) e realizamos entrevistas. Os dados coletados foram analisados qualitativamente, comparando as definições da literatura e as respostas do survey e da entrevista. Com base nos resultados, temos indícios de que as habilidades do pensamento computacional ainda são pouco exploradas no grupo de professores pesquisados.
1	Costa *et al.* (2017)	Os impactos causados pela tecnologia na sociedade fizeram requerer da escola muito mais do que o ensino de conteúdos. A fez repensar como trabalhar diversas competências que são exigidas neste século, entre elas, as competências relacionadas à computação. Blinkstein (2008) acredita que a lista de habilidades exigidas para este século é bem extensa. Porém, ele enfatiza o pensamento computacional com sendo uma das mais significativas, como também a menos compreendida.
1	Costa *et al.* (2017)	Pensamento computacional pode ser compreendido como um processo de resolução de problemas que inclui conceitos, habilidades e práticas da Ciência da Computação (Wing, 2006). Wing (2006) argumenta ainda que a sua utilização direciona a solucionar problemas nos mais diversos campos do conhecimento, não sendo exclusiva para profissionais da computação.
1	Costa *et al.* (2017)	Diante dessas premissas, diversos trabalhos vêm desenvolvendo o pensamento computacional nas escolas do Brasil. Scaico et al. (2012) buscaram propagar conceitos da computação nas escolas públicas, utilizando de minicursos para abordarem tais fundamentos com e sem o uso do computador. Já Vieira, Passos e Barreto (2013) visaram inovar na forma de inserção desses fundamentos da computação, se utilizando de técnicas teatrais junto com computação desplugada para propagar os fundamentais da computação nas escolas públicas. Campos et al. (2014) utilizaram atividades com números binários cujo o objetivo foi trabalhar conceitos de abstração e representação de dados.
3	Costa *et al.* (2017)	Essas iniciativas, no intuito de explorar esses conceitos e competências da computação, partiram do âmbito de projetos universitários. São escassos os trabalhos que nos apresentam iniciativas feitas por professores da educação básica. Mesmo sendo possível a inserção de conceitos e práticas da computação, como o pensamento computacional, em sala de aula, com ou sem o uso do computador, são poucas as pesquisas que investigam iniciativas que sejam originados desses professores. Nesse sentindo, desconhecemos se os professores da educação básica possuem ciência dessas habilidades ou se as exploram em suas práticas pedagógicas; também não sabemos qual a visão deles acerca desses conceitos. Assim, é importante investigar quais são as suas percepções sobre o pensamento computacional, uma vez que, os professores da área da computação defendem que as habilidades desenvolvidas com o pensamento computacional podem ser trabalhadas nas mais diferentes áreas do conhecimento, possibilitando assim, uma maior integração sobre diversas disciplinas em sala de aula.

(continua)

Tabela 28 – Textos do WAlgProg que usam o termo "pensamento computacional" mais que trinta vezes

(continuação)

quant.	autor	parágrafo
3	Costa *et al.* (2017)	Este trabalho inicial e exploratório tem o objetivo de conhecer a percepção dos professores do Ensino Médio sobre o pensamento computacional e analisar se as atividades desenvolvidas por eles auxiliam no desenvolvimento dessas habilidades nos alunos. Levantamos as seguintes questões de pesquisa: Qual a percepção dos professores do Ensino Médio sobre o uso das habilidades do pensamento computacional na prática docente? As atividades desenvolvidas pelos professores auxiliam no desenvolvimento do pensamento computacional?
2	Costa *et al.* (2017)	Para responder as questões de pesquisa, iniciamos com uma revisão bibliográfica a respeito do pensamento computacional na educação básica. Em seguinda, planejamos e realizamos um survey com professores do Ensino Médio de uma escola pública no intuito de identificar qual a percepção que eles tinham das habilidades do pensamento computacional em sua prática pedagógica. Por último, entrevistamos seis desses professores com o propósito de identificar como as habilidades eram trabalhadas em suas atividades junto aos alunos. Os dados coletados foram analisados qualitativamente, comparando as definições da literatura e as respostas do survey e da entrevista.
1	Costa *et al.* (2017)	2. Pensamento Computacional
2	Costa *et al.* (2017)	O termo Pensamento Computacional ganhou destaque no artigo seminal de Wing (2006). Ela argumentou que pensamento computacional envolve a resolução de problemas, a concepção de sistemas e compressão do comportamento humano, baseando-se nos fundamentos da Ciência da Computação. Trata-se ainda de um processo que utiliza abstração, recursividade e iteração para se modelar e criar artefatos, bem como outras competências da computação.
2	Costa *et al.* (2017)	A Computer Science Teachers Association (CSTA) possui uma força tarefa que é encarregada de conceituar e propor diretrizes para o ensino do pensamento computacional nas escolas, a CSTA Computational Thinking Task Force (CSTA, 2011). Ela definiu nove habilidades que são primordiais para exercício do pensamento computacional. As nove habilidades são: (i) Coleta de Dados, uma técnica para reunião de dados para se obter informações; (ii) Análise de Dados, compreensão feita sobre o conjunto de informações coletadas; (iii) Representação de Dados, maneira como trabalhamos com os resultados obtidos, ou seja, por meio de representação de gráficos, tabelas, imagens etc; (iv) Decomposição de Problema, um procedimento de decompor os problemas em subpartes menores para se trabalhar com pequenas unidades do problema; (v) Abstração, capacidade de abstrair conceitos amplos, e focar na essência do objeto estudado para tentar compreendê-lo; (vi) Algoritmos e Procedimentos, uma sequência de passos para resolver problemas; (vii) Automação, o processo que inclui um dispositivo (máquinas e/ou ferramentas) que faça a automatização dos processos, seja eles manuais ou mentais; (viii) Paralelização, trabalhar com diversos recursos ao mesmo tempo, para obter um resultado em comum; e (ix) Simulação, representação ou execução de um processo.
1	Costa *et al.* (2017)	2.1. Pensamento Computacional no Ensino Escolar Brasileiro

(continua)

Tabela 28 – Textos do WAlgProg que usam o termo "pensamento computacional" mais que trinta vezes (continuação)

quant.	autor	parágrafo
1	Costa *et al.* (2017)	No Brasil, as iniciativas voltadas à promoção de pensamento computacional nas escolas são provenientes de projetos desenvolvidos nas universidades em muitos dos casos. Vemos ações de pesquisadores e professores de Computação que, através de projetos, desenvolvem tais conteúdos nas escolas. Isso acontece porque os currículos nacionais não contemplam o ensino de computação, mesmo que seus benefícios sejam comprovados sobre as práticas pedagógicas [Cambraia e Scaico, 2013].
1	Costa *et al.* (2017)	Dentre as iniciativas recentes, França et al.(2014) realizaram um relato histórico de atividades desenvolvidas para estimular o ensino de computação no Brasil. Os autores destacaram as iniciativas de: (i) ensino de fundamentos da Ciência da Computação, principalmente através de atividades desplugadas, (ii) ensino de robótica e (iii) promoção do pensamento computacional de modo interdisciplinar por meio do PIBID (Programa Institucional de Bolsa de Iniciação à Docência) das licenciaturas em Pedagogia, Letras, Matemática, Geografia, História.
3	Costa *et al.* (2017)	Mannila et al. (2014) apresentaram a visão que os professores do K-9 (equivante ao ensino fundamental no Brasil) de diferentes países possuem a respeito do pensamento computacional em sua prática pedagógica. Os autores realizaram um survey com professores do K-9 de 6 países com o objetivo de identificar quais aspectos do pensamento computacional já fazem parte das práticas pedagógicas e como elas são realizadas. Os resultados apontaram que algumas práticas já são realizadas, mesmo sem a ciência por parte dos professores de que se trata de habilidade do pensamento computacional.
3	Costa *et al.* (2017)	No Brasil, ainda são escassas as pesquisas sobre a compreenção do pensamento computacional por professores da educação básica, bem como por profissionais e estudantes de Computação. Farias, Andrade e Alencar (2015) aplicaram um questionário a alunos concluintes de um curso de Licenciatura em Computação com propósito de compreender o grau de ciência deles sobre a importância do pensamento computacional. Os resultados apontaram que estudantes possuem limitações conceituais sobre o pensamento computacional, apresentando respostas fundamentadas em crenças, as quais destoam dos conceitos e práticas da literatura.
3	Costa *et al.* (2017)	Araújo, Andrade e Guerrero (2015) realizaram um survey com profissionais da computação no intuito de capturar a compreensão que eles possuem a respeitos do pensamento computacional. Os resultados apontaram que 64% dos pesquisados desconhecem o que o significado de pensamento computacional. Além disso, os resultados sugerem que o termo e as habilidades são pouco conhecidos e compreendidos pelos profissionais, tanto na indústria como na academia. Nesse contexto, vemos que estudantes e profissionais da computação ainda desconhecem pensamento computacional.
1	Costa *et al.* (2017)	A abordagem desta pesquisa pode ser caracterizada como uma pesquisa de campo qualitativa. Para norteá-la, realizamos o levantamento bibliográfico para embasar os conceitos, habilidades e competências do pensamento computacional. Em seguida, planejamos e executamos um survey supervisionado e entrevistas com professores do ensino médio. O restante da seção detalha os participantes e os instrumentos.

(continua)

Tabela 28 – Textos do WAlgProg que usam o termo "pensamento computacional" mais que trinta vezes

(continuação)

quant.	autor	parágrafo
1	Costa *et al.* (2017)	Da entrevista, participaram seis professores que foram selecionados por conveniência. Eles lecionam as seguintes disciplinas: Geografia, Educação Física, História, Biologia, Matemática e Língua Estrangeira. Esses professores selecionados abrangem as área das Ciências Humanas e suas Tecnologias; Linguagens, Códigos e suas Tecnologias e das Ciências da Natureza, Matemática e suas Tecnologias. A escolha dessas áreas se deu pelo fato de tentarmos compreender a visão dos professores em cada grande área, uma vez que as habilidades do pensamento computacional podem ser desenvolvidas em cada uma delas, mas de maneiras distintas. Três professores foram do sexo masculino e três do sexo feminino.
2	Costa *et al.* (2017)	O questionário aplicado no survey foi adaptado de (Manilla et al, 2014). Ele contém 11 questões, sendo 10 questões objetivas e uma subjetiva. Nove dessas questões objetivas abrangem as nove habilidades do pensamento computacional consideradas neste trabalho: coleta de dados, análise de dados, representação de dados, decomposição de problema, abstração, algoritmos, automação, paralelização e simulação. Cada questão apresentava a explicação da habilidade e solicitava que o professor selecionasse a frequência com que julgava explorar essas habilidades do pensamento computacional. A escala de frequência compreendia nas opções de: nem um pouco, um pouco, às vezes, na maioria das aulas, em todas as aulas. A questão subjetiva pedia para o professor descrever um exemplo de como uma das nove habilidades questionadas era utilizada em sala de aula. A décima questão objetiva do questionário pedia para o professor assinalar quais softwares, tecnologias ou outras ferramentas ele usa em sala. As questões do survey podem ser vistas na íntegra no link: https://goo.gl/5Jo5dp
1	Costa *et al.* (2017)	O questionário (survey) teve o objetivo de identificar quais habilidades do pensamento computacional são mais frequentemente utilizadas pelos professores do Ensino Médio durante a realização das atividades dentro ou fora da sala de aula.
1	Costa *et al.* (2017)	A Figura 1 apresenta um gráfico de barras com as respostas dos professores. Percebemos que nenhum dos professores mencionou utilizar as habilidades do pensamento computacional "Em todas as aulas". Porém, vemos indícios de que permite comparar as habilidades utilizadas em outras ocasiões. Ao analisar a resposta "Às vezes", vemos que ela recebeu a maior quantidade de votos em todas as habilidades, com exceção de decomposição de problema, pois é a única entre todas que é utilizada "Na maioria das aulas" pelos professores. Ao analisar a resposta "Um pouco", a quantidade de votos ultrapassou os votos de algumas habilidades que são utilizadas "Na maioria das aulas". Tais habilidades foram: análises de dados, representação de dados, algoritmos e procedimentos, abstração e simulação. Já ao analisarmos os votos de "Nem um pouco" notamos, com base na Figura 1, que houve apenas uma única vez em que a quantidade de votos foi igual para umas das habilidades utilizadas "Na maioria das aulas", a qual foi algoritmos e procedimentos.
1	Costa *et al.* (2017)	As respostas subjetivas foram analisadas e classificadas como "Possui relação" e "Não possui relação" com as habilidades do pensamento computacional. O critério para a classificação foi à identificação de ao menos uma habilidade na resposta do professor.

(continua)

Tabela 28 – Textos do WAlgProg que usam o termo "pensamento computacional" mais que trinta vezes

(continuação)

quant.	autor	parágrafo
1	Costa *et al.* (2017)	Sete respostas foram classificadas como "possui relação" e as habilidades encontradas foram: decomposição de problemas, simulação, abstração, automação, representação de dados. Das respostas que possuem relação com habilidade do pensamento computacional, podemos destacar as seguintes explicações dos professores: a) "Durante os seminários os alunos procuram dividir tarefas grandes em partes menores"; b) "Em trigonometria, utilizo o próprio espaço físico e com auxílio de planilhas ou Geogebra fazemos a demonstração"; c) "Encontrar características para criar modelos que ajudem a resolver alguns problemas enfrentados na sala de aula" , e d) "Dividir tarefas grandes em menores e usar um instrumento tecnológico para ajudar a realizar novas tarefas. Além de tornar a aula menos cansativa, desperta a curiosidade do aluno e facilita a absorção dos conteúdos".
1	Costa *et al.* (2017)	A análise das respostas das entrevistas foi realizada com intuito de conhecer como os professores de fato exploram as habilidades do pensamento computacional durante as atividades didáticas em sala de aula. A análise e classificação das respostas foi baseada nas definições presentes no Computational Thinking Teacher Resources (2011) e em Barr e Stephenson (2011).
1	Costa *et al.* (2017)	Para a habilidade de análise de dados, no survey, os professores responderam que exploram essa habilidade "na maioria das aulas". Entretanto, na entrevista, apenas um professor tentou demonstrar que usava análise de dados. Lendo as outras respostas, temos indícios de que o conceito de análise de dados acaba perdendo o sentido do contexto de pensamento computacional, pois os professores afirmam realizar a interpretação de questões para o aluno. Já para a CSTA (2011, p. 14), análise de dados é visto como "dá sentido aos dados, categorizar, produzir e avaliar gráficos a partir de dados gerados".
1	Costa *et al.* (2017)	Na habilidade de algoritmos e procedimentos, não conseguimos identificar nenhum exemplo nas respostas dos professores. Já para a habilidade de abstração, as respostas dos professores conduzem os alunos a grifarem os dados os quais a questão solicita, ou ainda identificar as palavras-chaves da questão. Três professores associaram a habilidade de simulação à participação de aula de campo, na qual os alunos poderiam observar os assuntos teóricos trabalhos em sala. Entretanto, tal prática não possui relação direta com a definição de simulação apresentado na literatura de pensamento computacional. No tocante a habilidade de automação, o professor de inglês respondeu utilizar junto aos alunos ferramentas de tradução automática de texto. Nas demais respostas, os professores compreenderam que estimular os alunos a usarem o computador, tablet ou celular consistia em uma forma de automatizar tarefas.
1	Costa *et al.* (2017)	Este trabalho exploratório buscou compreender a percepção dos professores do Ensino Médio sobre o pensamento computacional. Os métodos utilizados para atingir esse objetivo foram revisão bibliográfica, aplicação de questionários (survey) e entrevistas. Os dados obtidos por meio dos instrumentos de coleta de dados foram analisados confrontando-os com as definições presentes na literatura.

(continua)

Tabela 28 – Textos do WAlgProg que usam o termo "pensamento computacional" mais que trinta vezes

(continuação)

quant.	autor	parágrafo
4	Costa *et al.* (2017)	Com base nos dados analisados com vistas à literatura, temos indícios que as habilidades do pensamento computacional ainda são pouco exploradas no grupo de professores pesquisados. Esse resultado pode ser ocasionado pela não compreensão do conceito e das habilidades relacionadas ao pensamento computacional. Essa explicação se torna mais evidente quando observamos que alunos concluintes e profissionais da Computação desconhecem pensamento computacional ou possuem conhecimentos equivocados, não condizentes com a literatura, como apontado nos trabalhos relacionados. Somado a isso, a falta de uma formação adequada em pensamento computacional voltada aos professores da educação básica pode ser uma das causas para o desconhecimento sobre como abordá-lo em práticas pedagógicas nas disciplinas.
1	Costa *et al.* (2017)	Outros trabalhos futuros podem conduzidos nas vertentes de: (i) realizar nova coleta de dados em diferentes escolas, envolvendo ensino público e privado; (ii) expandir a coleta de dados para professores do ensino fundamental e comparar os resultados obtidos; (iii) analisar o uso das habilidades do pensamento computacional por disciplinas e comparar se uma área temática explora um maior conjunto de habilidades em comparação a outra.
1	Costa *et al.* (2017)	ARAUJO, A. L., ANDRADE, W., & SEREY, D. (2015). Pensamento Computacional sob a visão dos profissionais da computação: uma discussão sobre conceitos e habilidades. In: IV Anais dos Workshops do Congresso Brasileiro de Informática na Educação.
1	Costa *et al.* (2017)	BLIKSTEIN, Paulo. (2008). O pensamento computacional e a reinvenção do computador na educação. Disponível em <http://www.blikstein.com/paulo/documents/online/ol_-pensamento_computacional.html> Acesso em: 02 de set. 2017.
1	Costa *et al.* (2017)	CAMPOS, Gleider M. et al. (2014) Organização de Informações via Pensamento Computacional: Relato de Atividade Aplicada no Ensino Fundamental. In: Anais do Workshop de Informática na Escola.
1	Costa *et al.* (2017)	FARIAS, A., ANDRADE, W., & ALENCAR, R. (2015). Pensamento Computacional em Sala de Aula: Desafios, Possibilidades e a Formação Docente. In: IV Anais dos Workshops do Congresso Brasileiro de Informática na Educação.
1	Costa *et al.* (2017)	FRANÇA, R. S., FERREIRA, V. F. S., ALMEIDA, L. D., & AMARAL, H. D. (2014). A disseminação do pensamento computacional na educação básica: lições aprendidas com experiências de licenciandos em computação. In: Anais do XXII Workshop sobre Educação em Computação (WEI-CSBC).
1	Costa *et al.* (2017)	2. Pensamento Computacional
1	Costa *et al.* (2017)	2.1. Pensamento Computacional no Ensino Escolar Brasileiro
1	Leite *et al.* (2017)	Pensamento Computacional nas Escolas: Limitado pela Tecnologia, Infraestrutura ou Prática Docente?

(continua)

Tabela 28 – Textos do WAlgProg que usam o termo "pensamento computacional" mais que trinta vezes

(continuação)

quant.	autor	parágrafo
1	Leite *et al.* (2017)	Resumo. A informática converge para inovações e automatização de processosna resolução de problemas aplicada a quaisquer áreas. O PensamentoComputacional incorpora inovações tecnológicas no meio pedagógico epromove o dinamismo nas técnicas e métodos tradicionais. O artigo identificouas demandas das escolas municipais de Francisco Beltrão – Paraná, nosaspectos que envolvem o uso da tecnologia como meio de transmissão eaquisição de conhecimento, seja por parte do aprendiz ou do professor. Osresultados mostram reconstrução dos processos de ensino e aprendizagem. Aintervenção de acadêmicos da Licenciatura em Informática é necessária noexercício docente relacionado à Computação nas escolas municipais.
1	Leite *et al.* (2017)	A explicação para a resistência dos professores ao uso da tecnologia pode estar fundamentada no paradigma dos currículos tradicionalistas. Trata-se de uma vertente contrária à vaga curricular para professor graduado em Licenciatura em Informática. Desta maneira, o conceito de Pensamento Computacional, proposto por Wing (2006), emergiu como incentivador aos recursos tecnológicos utilizados por professores das escolas públicas municipais da região de Francisco Beltrão – Paraná, que não tem a formação tecnológica inserida em seus currículos.
1	Leite *et al.* (2017)	O objetivo deste trabalho foi identificar e mapear itens presentes no espaço de intervenção para compreender se a aplicação da proposta do Pensamento Computacional nas escolas municipais está limitada à tecnologia ou ao currículo dos professores. Para isso, os alunos da disciplina de Didática Aplicada à Informática estruturaram um survey que foi aplicado em escolas municipais de Francisco Beltrão – Paraná. Esta atividade fez parte dos Estágios I, II e III de experimentação à Prática Docente dos acadêmicos. Os resultados apontaram para um colegiado ainda não preparado e sem vistas de interesse tecnológico.
1	Leite *et al.* (2017)	2. Pensamento Computacional e o Curso de Licenciatura em Informática
3	Leite *et al.* (2017)	Pensamento Computacional é um tema que atrai pesquisadores e parece estar impactandono contexto escolar. Em 2006, Jeannette M. Wing, no artigo Computational Thinking,apresenta o conceito para Pensamento Computacional e a relevância de abordar talassunto dentro do contexto educacional de forma tão elementar quanto o aprendizado daleitura, escrita ou de operações matemáticas. Deste ponto em diante, diversas pesquisase aplicações do Pensamento Computacional foram estudadas e aplicadas.
1	Leite *et al.* (2017)	O termo Pensamento Computacional, termo cunhado pela proposta de Jeannette Wing, está também associado às ideias de resolução de problemas, projeto de sistemas ecompreensão do comportamento humano norteados por conceitos fundamentais daCiência da Computação (Wing, 2006). Isto se revela como um método utilizado parasolucionar problemas, conceber sistemas pelo tal como preconiza a Teoria Geral deSistemas pela habilidade de abstrair e generalizar, e compreender o comportamentohumano inspirado em conceitos fundamentais da Ciência da Computação.

(continua)

Tabela 28 – Textos do WAlgProg que usam o termo "pensamento computacional" mais que trinta vezes

(continuação)

quant.	autor	parágrafo
1	Leite *et al.* (2017)	Araujo, Andrade e Serey (2015) se posicionam a partir da definição dada pela International Society for Technology in Education (ISTE) e pela Computer Science Teachers Association (CSTA) como uma abordagem para resolução de problemas, incorporando processos mentais e ferramentas que utilizam habilidades como organização e análise de dados, construção de algoritmos, abstração, criação de modelos, simulação, automatização de soluções e paralelização, considerando como uma lista de habilidades. Nessa mesma perspectiva, França e Tedesco (2015) defendem a inserção do Pensamento Computacional desde a educação básica como forma de melhorar o aprendizado lógico em nível escolar dos alunos e possibilitar o uso mais eficaz dessas tecnologias móveis em benefício da sociedade.
1	Leite *et al.* (2017)	É importante lembrar que o Pensamento Computacional não se fundamenta em saber navegar na Internet, acessar e-mails, editar um texto, utilizar planilhas eletrônicas, elaborar uma apresentação ou manipular um equipamento eletrônico. Sua importância está para o processo de resolução de problemas lógicos nos diversos contextos da sociedade, permitindo que se possa aplicar a Computação nas suas ações cotidianas (GOMES e MELO, 2013).
2	Leite *et al.* (2017)	Segundo Mestre et. al. (2015), as habilidades estimuladas pelo Pensamento Computacional estão diretamente relacionadas à resolução de problemas, que envolvem as capacidades de ler, interpretar textos, compreender situações reais propostas e transpor informações para modelos matemáticos, científicos ou sociais. Para os autores Araújo, Andrade, Guerrero (2015), o Pensamento Computacional é um conjunto de conceitos, habilidades e práticas da Computação que podem ser aplicados tanto em atividades do cotidiano como em outras áreas do conhecimento. Uma abordagem de resolução de problemas incorporando processos mentais e ferramentas que utilizam habilidades, tais como organização e análise de dados, construção de algoritmos, abstração, decomposição, simulação, automatização e paralelização.
1	Leite *et al.* (2017)	O Pensamento Computacional está centrado na perspectiva da resolução de problemas mediados ou não pelo uso computador. Essa abordagem se expande por todos os contextos na busca de soluções mais eficientes para resolver problemas e atividades complexas.
1	Leite *et al.* (2017)	Nunes (2010) esclarece que o ensino de aplicativos, tais como editores de texto, planilhas de cálculo, linguagens de programação não influencia pensamentos de resolução de problemas na educação básica. Por exemplo, para datilografar não é necessário o uso de um editor de texto, nem calculadoras para o ensino da Matemática. O papel do professor está em tornar o aluno confiante e capaz de pensar computacionalmente. Para Blikstein (2008), o uso dos aparatos digitais (gadget) também é um facilitador a desenvolver a habilidade do Pensamento Computacional.

(continua)

Tabela 28 – Textos do WAlgProg que usam o termo “pensamento computacional” mais que trinta vezes

(continuação)

quant.	autor	parágrafo
1	Leite *et al.* (2017)	Os cursos de Licenciatura em Informática têm a responsabilidade de formar professores para disseminar o Pensamento Computacional, assim se pode confirmar a demanda em relação as competências e habilidades previamente definidas no Processo nº 23001.000026/2012-95 do tratado pelas Diretrizes Curriculares Nacionais para os cursos de graduação (bacharelado e licenciatura) em Computação necessárias para aquisição de conhecimentos de forma autônoma e efetiva (NUNES, 2011). O aluno deve atingir competências para reconhecer tarefas a serem realizadas de maneira mais rápida e eficientemente mediadas por computador. A partir dessas competências, o aluno também será capaz de desenvolver habilidades para programar o computador a realizar tais tarefas.
1	Leite *et al.* (2017)	Para elucidar se o uso do Pensamento Computacional nas escolas municipais estálimitado à tecnologia, infraestrutura ou ao currículo dos professores das escolas da redepública municipal, utilizou-se do survey exploratório supervisionado, como metodologiade coleta de dados amplamente empregada por Bauer e Gaskell (2002).
1	Leite *et al.* (2017)	No que tange à primeira dimensão, intitulada Infraestrutura e Equipamentos Informáticos,ver Figura 1, nota-se que somente 60
1	Leite *et al.* (2017)	Para a segunda dimensão, intitulada Acesso à Internet nas Salas de Aulas, a Figura2 apresenta um resultado que favorece a argumentação daqueles professorestradicionalistas que usam quadro e giz. Embora dados revelaram que mais de 50
1	Leite *et al.* (2017)	Já a terceira dimensão, chamada de Ferramentas e Programas mais utilizados nas Salas de Informática, revelou que embora as escolas não possuírem acesso constante à Internet, alguns professores utilizaram do Pensamento Computacional no uso de softwares educativos e utilitários, conforme apresenta a Figura 3.
1	Leite *et al.* (2017)	Por fim, a última dimensão, intitulada Internet vs. Atividades Escolares, apresenta dados que informam o desinteresse dos alunos frente às atividades desenvolvidas nas salas de informática com metodologias tradicionalistas, ver Figura 4. Estes dados também revelaram que os softwares educativos não estão sendo trabalhados de maneira se fazer presente nas atividades cotidianas dos alunos. Infelizmente infere-se que o uso das salas de informática está somente pela manipulação de objetos, sem um motivo ou aplicação direta ao Pensamento Computacional em seu cotidiano escolar.
1	Leite *et al.* (2017)	Os resultados encontrados apontam evidências de atividades abordadas com pouca complexidade, tais como ler notícias, copiar conteúdos, visualizar mapas, desenhar, escrever com editores de texto, calcular com calculadora. A partir desses dados é possível afirmar que a forma como a tecnologia foi explorada por estas escolas não satisfaz a perspectiva do Pensamento Computacional.
2	Leite *et al.* (2017)	O estudo abordou a questão fundamental sobre a preparação dos professores e dasestruturas educacionais para a implementação do ensino de computação. Tambémdescreveu os resultados obtidos com a aplicação de um survey em escolas públicas como objetivo de compreender como o Pensamento Computacional está sendo tratado nasescolas e quais as limitações existentes. Basicamente, o survey contemplou as seguintesdimensões: infraestrutura, acesso à Internet, aplicativos e atividades escolares. Osresultados mostraram um corpo docente despreparado para atuar com a prática dopensamento computacional.

(continua)

Tabela 28 – Textos do WAlgProg que usam o termo “pensamento computacional” mais que trinta vezes

(continuação)

quant.	autor	parágrafo
2	Leite *et al.* (2017)	Com o objetivo de identificar se a aplicação do Pensamento Computacional estava limitada pela infraestrutura ou pelos professores, percebeu-se que todo o processo de ensino-aprendizagem precisa ser considerado. Embora se tenha espreitado apenas as atividades escolares que usam computador ou dispositivos móveis, é sabido que a aplicação do Pensamento Computacional vai além. Nessa vertente, o mapeamento nas escolas revelou que o esforço do governo em investir em formação de professores para uso de recursos tecnológicos é válido, mas parece frágil. Assim, cognição e comportamento podem ser relacionados, extraíndo as melhores características, com o intuito de revelar um método para solucionar problemas. De posse dessa hibridização, é possível conceber sistemas tal como preconiza a Teoria Geral de Sistemas e sua habilidade de abstrair, generalizar e compreender o comportamento humano inspirado em conceitos fundamentais da Ciência da Computação. No caso das escolas presentes neste estudo, ainda há muito aspectos a serem reforçados, validados pelos resultados encontrados na região Sudoeste do Paraná. Possivelmente seja a consequência do currículo tradicionalista destes professores em não pensar computacionalmente, de maneira lúdica ou ainda, da proposta não ter sido esclarecida junto às escolas, através de cursos de formação de professores, ou da demanda de um licenciado em informática estar presente nestes espaços.
1	Leite *et al.* (2017)	Vale destacar que o uso dos recursos tecnológicos no ensino, não exige apenas seu manuseio, mas um interesse intrínseco em fazer a diferença na prática pedagógica reflexiva, uma vez que o uso de tablets e smartphones não garante, por si só, uma melhor qualidade do ensino. Em se tratando de tecnologia, a maneira como está sendo utilizada com os alunos mostra que é preciso uma grande intervenção dos profissionais de licenciatura para potencializar o uso do pensamento computacional (não necessariamente apenas com a internet, mas com outros recursos). Assim, a formação continuada de professores para a utilização de tecnologias afins pode vir a contribuir para o aprimoramento da prática educativa, se pautada pela compreensão das possibilidades e limites desse instrumento e na concretização do papel educativo escolar.
1	Leite *et al.* (2017)	A análise dos gráficos também revelou um contexto escolar público bimodal. De um lado, professores e suas disciplinas específicas devem receber formação para explorar os recursos tecnológicos em sala de aula, uma vez que o governo não optou pela computação desplugada por Bell, Witten e Fellows (2011). De outro, os profissionais licenciados em informática buscam solidificar sua prática e sua participação no contexto escolar envoltos no Pensamento Computacional produto de uma formação preocupada com a pesquisa que emerge neste meio.
1	Leite *et al.* (2017)	Araujo, A. L.; Andrade, W.; Serey, D. (2015). Pensamento Computacional sob a visãodos profissionais da computação: uma discussão sobre conceitos e habilidades. InAnais dos Workshops do Congresso Brasileiro de Informática na Educação (Vol. 4,p. 1454). doi:10.5753/cbie.wcbie.2015.1454Bauer, M. W.; Gaskell, G. (2002). Pesquisa Qualitativa com Texto, Imagem e Som (5thed.).

(continua)

Tabela 28 – Textos do WAlgProg que usam o termo "pensamento computacional" mais que trinta vezes

(continuação)

quant.	autor	parágrafo
2	Leite *et al.* (2017)	Blikstein, P. (2008). O pensamento computacional e a reinvenção do computador naeducação. Disponível em <http://www.blikstein.com/paulo/documents/online/ol_-pensamento_computacional.html > Acesso em: 05 de mai. de 2016.
1	Leite *et al.* (2017)	França, R.; Tedesco, P. (2015). Desafios e oportunidades ao ensino do pensamentocomputacional na educação básica no Brasil. In Anais dos Workshops do CongressoBrasileiro de Informática na Educação (CBIE 2015) (Vol. 4, p. 1464).doi:10.5753/cbie.wcbie.2015.1464
1	Leite *et al.* (2017)	Gomes, T. C. S.; e De Melo, J. C. B. (n.d.). O Pensamento Computacional no EnsinoMédio: Uma Abordagem Blended Learning. In Workshop sobre Educação emComputação (WEI 2013), 2013, Maceió, AL. Anais do XXXIII Congresso daSociedade Brasileira de Computação (CSBC 2013)
1	Leite *et al.* (2017)	Mestre, P., Andrade, W., Guerrero, D., Sampaio, L., Rodrigues, R. D. S., & Costa, E.(2015). Pensamento Computacional: Um estudo empírico sobre as questoes dematemática do PISA. In Anais dos Workshops do Congresso Brasileiro deInformática na Educação (Vol. 4, p. 1281). doi:10.5753/cbie.wcbie.2015.1281
1	Raabe *et al.* (2017)	Um Instrumento para Diagnóstico do PensamentoComputacional
5	Raabe *et al.* (2017)	Resumo. Autores da comunidade científica têm discutido amplamente sobre a possibilidade do desenvolvimento do pensamento computacional melhorar aspectos da capacidade cognitiva do indivíduo. Esta pesquisa tem como objetivo desenvolver um instrumento de diagnóstico do pensamento computacional e aplicar para identificar relações entre a formação dos indivíduos e o pensamento computacional. Busca também validar como os resultados obtidos podem refletir se o pensamento computacional pode ser diagnosticado em um indivíduo. Busca também avaliar o instrumento através dos resultados obtidos em relação a validade do diagnóstico gerado sobre pensamento computacional de um indivíduo. Através da aplicação foram encontrados indícios onde indivíduos que tiveram contato com elementos da computação tiveram melhor desempenho diagnosticado pelo instrumento desenvolvido.
3	Raabe *et al.* (2017)	O Pensamento Computacional é um termo que sintetiza o conjunto de habilidades cognitivas que os profissionais da área de Computação geralmente desenvolvem em seu processo de formação. Wing apresentou este termo através da publicação de um artigo em 2006, defendendo que o Pensamento Computacional deve ser introduzido como uma habilidade na formação da criança. Após esta declaração de Wing, estão surgindo diversas iniciativas promovendo a inserção do Pensamento Computacional como componente do currículo na educação escolar (Wing, 2006).

(continua)

Tabela 28 – Textos do WAlgProg que usam o termo "pensamento computacional" mais que trinta vezes

(continuação)

quant.	autor	parágrafo
1	Raabe *et al.* (2017)	A CSTA define o pensamento computacional como um processo de resolução de problemas que inclui, não exclusivamente, as seguintes características: (i) Formulação de problemas de forma que computadores e outras ferramentas possam ajudar a resolvê-los; (ii) Organização lógica e análise de dados; (iii) Representação de dados através de abstrações como modelos e simulações; (iv) Automatização de soluções através do pensamento algorítmico; (v) Identificação, análise e implementação de soluções visando a combinação mais eficiente e eficaz de etapas e recursos (vi) Generalização e transferência de soluções para uma ampla gama de problemas (CSTA, 2011).
1	Raabe *et al.* (2017)	Um fator motivador para a exploração do pensamento computacional no âmbito da educação básica no Brasil é o baixo desempenho publicado pelo Programme for International Student Assessment (PISA) o qual avalia a performance escolar em matemática, ciências, letramento, resolução de problemas e cognição de alunos de 15 anos de idade. No último relatório no ano de 2014 o Brasil consta na trigésima oitav posição dentre os quarenta e quatro países analisados no teste de resolução de problemas. "Ao longo da última década, o PISA, tornou-se critério mundial para avaliar a qualidade, equidade e eficiência dos sistemas escolares" (OECD, 2014, p. 03).
1	Raabe *et al.* (2017)	Existem algumas abordagens que buscam se aproximar de uma mensuração do Pensamento Computacional. A equipe de Brennan e Resnick utilizou o Scratch como ferramenta para estudantes desenvolverem projetos e avaliaram os resultados através de contagem de comandos de programação, entrevistas e atividades. A conclusão mostra que as avaliações consomem bastante tempo e são realizadas de forma bastante qualitativas (RESNICK, BRENNAN, 2012). Já a SRI criou um framework que descreve quais habilidades devem ser avaliadas a fim de medir o PC com base no currículo d Exploring Science Computer e do Advanced Placement Science Computer Principles (SRI, 2015). O framework apresenta padrões de estratégias que visam avaliar o PC nas práticas: analisar os efeitos do desenvolvimento na computação; projetar e implementar soluções e artefatos criativos; projetar e aplicar abstrações e modelos; analisar o próprio trabalho computacional e o dos outros; comunicação através de processos e resultados; colaborar com outros em atividades computacionais.
1	Raabe *et al.* (2017)	É possível verificar que as abordagens encontradas na literatura são qualitativas, por este motivo não é apresentado um meio de quantificar as habilidades desenvolvidas pelo pensamento computacional nos estudantes, o qual seria de grande ajuda na avaliação do conteúdo para a implantação na formação básica dos indivíduos.

(continua)

Tabela 28 – Textos do WAlgProg que usam o termo "pensamento computacional" mais que trinta vezes

(continuação)

quant.	autor	parágrafo
1	Raabe *et al.* (2017)	Tendo em conta as abordagens já existentes, este trabalho busca contribuir com o desenvolvimento de um método para calcular uma pontuação a fim de quantificar o desenvolvimento da habilidade no indivíduo, além de identificar quais conceitos utilizados em atividades podem ser utilizados para mensurar esta pontuação. Para isso foi desenvolvido um instrumento com diversas atividades em conformidade com definições do pensamento computacional segundo a CSTA. Estas atividades foram aplicadas em amostras considerando diversas faixas etárias para ampliar a validade de constructo e a fidedignidade das mensurações conforme ferramental metodológico tradicional na área de psicometria.
2	Raabe *et al.* (2017)	O instrumento aborda atividades de resolução de problemas, planejadas para que não seja necessário conhecimento prévio de conceitos computacionais, focando nos elementos constituintes do pensamento computacional, os quais são mensurados através dos resultados e desempenho na resolução das atividades. As atividades foram baseadas em testes de QI e Questões do Programa de Enriquecimento Instrumental de Feuerstein (2006), e adaptadas para o formato de jogo puzzle. Este formato foi escolhido pois foi identificado que as iniciativas que abordam atividades gameficadas envolvendo conceitos do pensamento computacional são em sua maioria neste formato, como são os casos verificados em: A hora do código, LightBot, The Foos, Blockly Games e Tynker. Hong et al. (2012) define que jogos de gênero puzzle incentivam os jogadores a terem atenção imediata nos seus erros, facilitando a autoavaliação mais rapidamente do que em outras abordagens.
1	Raabe *et al.* (2017)	Cada uma das questões contidas no instrumento é apoiada em uma ou mais expressões do vocabulário definido em CSTA (2011), o qual sintetiza o pensamento computacional em termos. Foram elaborados nove tipos de questões, onde cada qual corresponde a uma fase do jogo que é composta por mais de um nível. Para melho entendimento, a partir deste momento cada questão do instrumento será abordada como uma fase que compõe o jogo. Na tabela 1 é apresentada a relação entre as questões e o termos da CSTA.
1	Raabe *et al.* (2017)	Tabela 1. Elementos do pensamento computacional presentes em cada fase
1	Raabe *et al.* (2017)	Em cada uma das fases são capturadas as interações para posteriormente construir o diagnóstico do pensamento computacional do indivíduo exposto ao instrumento, então é gerado um escore com base na captura destes dados, o qual é utilizado para identificar o quanto está desenvolvida a habilidade relacionada ao elemento abordado.
2	Raabe *et al.* (2017)	O foco do instrumento é compor um diagnóstico do pensamento computacional do indivíduo, então não é construído um escore final, a finalidade não é gerar uma not de pensamento computacional, mas pontuar em quais elementos o indivíduo é mais ou menos desenvolvido. Para atender este requisito, foi definida a cada questão uma equação matemática que resulta em um escore.

(continua)

Tabela 28 – Textos do WAlgProg que usam o termo "pensamento computacional" mais que trinta vezes

(continuação)

quant.	autor	parágrafo
1	Raabe *et al.* (2017)	A análise dos escores coletados durante a resolução do instrumento possibilita criar diagnósticos individuais que relatam quais elementos do pensamento computacional estão mais fortemente presentes no indivíduo. A elaboração das equações teve como base as interações capturadas durante a resolução do instrumento, por isso foram definidas diferentes equações para cada fase, porém existem definições gerais aplicadas em todas as equações:
1	Raabe *et al.* (2017)	Outras duas aplicações foram utilizadas para a validação concorrente, com estudantes dos cursos de graduação da UNIVALI, sendo 6 de Ciência da Computação, 11 de Engenharia da Computação e 8 de Psicologia. Foi verificada a correlação entre o escore em cada elemento do pensamento computacional e os indivíduos entre as turmas.
3	Raabe *et al.* (2017)	A escolha da utilização das médias da disciplina de matemática se deve ao fato do pensamento computacional ser fortemente conectado aos fundamentos da matemática, conforme Sussman (NRC, 2010) o pensamento computacional e a matemática, ambos são constituídos por uma estrutura linguística onde são necessárias descrições precisas sobre como fazer as coisas. Wing (NRC, 2010) relata que apesar do pensamento computacional estar restrito às limitações físicas do computador, é similar ao pensamento matemático em muitos aspectos.
1	Raabe *et al.* (2017)	Ao interpretar os resultados das correlações utilizando a definição de Cohen (1988) é possível identificar nos resultados obtidos das correlações efetuadas que para 6 elementos do pensamento computacional pode-se obter alta correlação tendo o coeficiente de correlação maior que 0,50, para 10 elementos o coeficiente de correlação se demonstrou moderado e para 2 elementos a correlação foi baixa.
2	Raabe *et al.* (2017)	Apesar do pensamento computacional ainda não ter uma definição única e não estarem definidas quais capacidades cognitivas estão intrínsecas a ele, durante este trabalho foi possível verificar que é possível fazer avaliações quantitativas sobre o seu desenvolvimento através da exploração de questões que abordam determinados elementos do pensamento computacional.
2	Raabe *et al.* (2017)	O fato de explorar os métodos da psicometria na construção do instrumento e utilizar como embasamento a definição operacional do pensamento computacional divulgado em CSTA (2011) possibilitou categorizar as habilidades dos estudantes em habilidade amplamente desenvolvida, habilidade moderadamente desenvolvida e habilidade pouco desenvolvida para cada um dos elementos do pensamento computacional presentes no instrumento.
1	Raabe *et al.* (2017)	As correlações mostram diferenças nos resultados, conforme esperado, os estudantes de Ciência da Computação tiveram melhor desempenho e a correlação obtida entre os estudantes de ensino fundamental e suas médias na disciplina de matemática mostrou que seis elementos do pensamento computacional possuem alta correlação. Mas devido ao pequeno número de participantes da pesquisa o resultado não é estatisticamente significativo.

(continua)

Tabela 28 – Textos do WAlgProg que usam o termo "pensamento computacional" mais que trinta vezes (continuação)

quant.	autor	parágrafo
1	Raabe *et al.* (2017)	O principal experimento a ser realizado no futuro com base neste trabalho é a aplicação do instrumento em uma quantidade maior de indivíduos em que os testes estatísticos possam garantir um resultado mais significante. A elaboração de novas questões pode ainda criar a possibilidade de utilizar este instrumento como pré e pós-testes para avaliação de experimentos que exploram o desenvolvimento pensamento computacional e a resolução de problemas.

Fonte: Autoria própria

APÊNDICE D – ENUNCIADO DA ATIVIDADE SOBRE PILHAS FEITO NA UNIVERSIDADE NO PRIMEIRO SEMESTRE DE 2018

Pilhagem

Observações gerais:

1. Ler este arquivo.

2. Ler novamente este arquivo.

3. Escolher um dos exercícios para resolver.

4. Pode ser feito em até duas pessoas.

5. Enviar por email para o professor até o dia 11/04/2018.

6. Os exemplos em cada exercício são ilustrativos, cabendo as e os dicentes decidir como fazer os seus programas.

7. Recomendação de como abordar um dos exercícios:

1. Entendimento Teórico:

* entender o problema.

* entender o porque do uso da pilha no problema.

* modelar o problema de forma concreta (diagrama, pseudocódigo, texto, ilustração, etc).

2. Escrita do programa:

* escrever e TESTAR o funcionamento da pilha.

* definir trecho do programa que lida com a entrada de dados.

* definir trecho do programa que lida com o uso da pilha.

* definir trecho do programa que lida com a lógica final do problema.

Palíndromo Construa um programa na linguagem c que ao receber uma frase, identifica se ela é ou não um palíndromo. Este programa deve:

1. Usar uma TAD do que implemente métodos de uma pilha.

2. Esta TAD deve estar separada da lógica de entrada e saída de dados (arquivo separado do ponto de entrada do programa).

3. Para facilitar o trabalho, o palíndromo inserido deve ser todo em letras minúsculas (evitar comparação entre maiúsculas e minúsculas. ex: "aA"e "aa").

4. Para facilitar o trabalho, quem digitá-lo deve colocar um ponto final na frase (evitar caracteres em branco ao final da frase; ex: "opa.").

obs: Provavelmente será necessário lidar com finais de string. Pesquisar a importância do charactere ’
0’ para isso.

Dificuldade

* Pilha - média

* Lógica - média

Exemplos de entradas e saídas do programa

“‘ digite uma frase (somente caixa baixa!): termine a frase com um ponto (.) ele. o: ele r: ele eh palindromo //resposta do programa.

digite uma frase (somente caixa baixa!): termine a frase com um ponto (.) qwerty. o: qwerty r: ytrewq nao eh palindromo //resposta do programa. “‘

CtrlZ Construa um programa na linguagem c que dê a opção de compra de dois produtos para a usuária ou o usuário. O programa também deve permitir que seja desfeita a última ação e que seja mostrada a quantidade atual de produtos em estoque:

1. Usar uma TAD do que implemente métodos de uma pilha. 2. Esta TAD deve estar separada da lógica de entrada e saída de dados (arquivo separado do ponto de entrada do programa). 3. O programa deve rodar indefinidamente até que a usuária ou o usuário escolha a opção de sair. 4. O programa deve dar opção de comprar uma quantidade do item A e guardar quantos itens foram comprados desde o início de sua execução. 5. O programa deve dar opção de comprar uma quantidade do item B e guardar quantos itens foram comprados desde o início de sua execução. 6. O programa deve mostrar quanto itens A e B foram comprados até então desde o início da execução do programa. 7. O programa deve conseguir desfazer a última ação de comprar um item. O desfazer não conta como ação.

Dificuldade * Pilha - média * Lógica - fácil

Exemplos de entradas e saídas do programa “‘ escolha uma opcao: 1 - comprar A: 2 - comprar B: 3 - mostrar quantidade de itens: 4 - ctrl Z: 5 - sair: 1

digite uma quantidade: 10 comprar 10 A

escolha uma opcao: 1 - comprar A: 2 - comprar B: 3 - mostrar quantidade de itens: 4 - ctrl Z: 5 - sair: 2

digite uma quantidade: 20 comprar 20 B

escolha uma opcao: 1 - comprar A: 2 - comprar B: 3 - mostrar quantidade de itens: 4 - ctrl Z: 5 - sair: 1

digite uma quantidade: 10 comprar 10 A

escolha uma opcao: 1 - comprar A: 2 - comprar B: 3 - mostrar quantidade de itens: 4 - ctrl Z: 5 - sair: 3 A: 20 B: 20

escolha uma opcao: 1 - comprar A: 2 - comprar B: 3 - mostrar quantidade de itens: 4 - ctrl Z: 5 - sair: 4 defazendo: comprar 10 A

escolha uma opcao: 1 - comprar A: 2 - comprar B: 3 - mostrar quantidade de itens: 4 - ctrl Z: 5 - sair: 3 A: 10 B: 20

escolha uma opcao: 1 - comprar A: 2 - comprar B: 3 - mostrar quantidade de itens: 4 - ctrl Z: 5 - sair: 4 defazendo: comprar 20 B

escolha uma opcao: 1 - comprar A: 2 - comprar B: 3 - mostrar quantidade de itens: 4 - ctrl Z: 5 - sair: 3 A: 10 B: 0

escolha uma opcao: 1 - comprar A: 2 - comprar B: 3 - mostrar quantidade de itens: 4 - ctrl Z: 5 - sair: 5 thau! “‘

Parênteses

Construa um programa na linguagem c que ao receber uma expressão matemática qualquer, verifique se os parênteses estão corretamente colocados:

1. Usar uma TAD do que implemente métodos de uma pilha. 2. Esta TAD deve estar separada da lógica de entrada e saída de dados (arquivo separado do ponto de entrada do programa). 3. O programa deve receber uma expressão matemática qualquer. 4. O programa deve responder se esta espressão é válida ou não com relação SOMENTE aos seus parênteses. 5. Para facilitar o trabalho, o programa não precisa reconhecer se a expressão é válida ou não de acordo com regras de sintaxe. Apenas verificar os parênteses.

Dificuldade * Pilha - fácil * Lógica - difícil

Exemplos de entradas e saídas do programa “‘ digite uma expressao: (10+10)/(20+2) a expressao (10+10)/(20+2) passou na verificacao de parenteses

digite uma expressao: 10+10 a expressao 10+10 passou na verificacao de parenteses

digite uma expressao: ()()() a expressao ()()() passou na verificacao de parenteses

digite uma expressao: (10)(10)((19))(10) a expressao (10)(10)((19))(10) passou na verificacao de parenteses

digite uma expressao:))((()() a expressao NAO))((()() passou na verificacao de parenteses

digite uma expressao: ((()) a expressao NAO ((()) passou na verificacao de parenteses “‘

Caminho de Volta

Construa um programa na linguagem c que possa receber quatro instruções de navegação e uma que mostre o mesmo caminho de volta até sua origem.

1. As instruções irão fazer um elemento mover-se em um plano cartesiano bidimensional. 2. O elemento deve ter uma posição x e uma y. 3. Cada instrução deve ser composta por 1 letra. * "c"faz o elemento andar uma unidade para cima (eixo y). * "b"faz o elemento andar uma unidade para baixo (eixo y). * "e"faz o elemento andar uma unidade para esquerda (eixo x). * "d"faz o elemento andar uma unidade para direita (eixo x). * "v"mostra uma sequencia de comandos que fazem o elemento andar devolta para a sua origem e faz o elemento voltar até a origem. 4. O programa deve rodar indefinidamente. 5. Deve ser mostrado sempre a posição atual do elemento.

Dificuldade * Pilha - médio * Lógica - fácil

Exemplos de entradas e saídas do programa

‘‘‘ x:0 y:0 c: anda para cima b: anda para baixo e: anda para a esquerda d: anda para a direita v: voltar! c

x:0 y:-1 c: anda para cima b: anda para baixo e: anda para a esquerda d: anda para a direita v: voltar! c

x:0 y:-2 c: anda para cima b: anda para baixo e: anda para a esquerda d: anda para a direita v: voltar! c

x:0 y:-3 c: anda para cima b: anda para baixo e: anda para a esquerda d: anda para a direita v: voltar! e

x:-1 y:-3 c: anda para cima b: anda para baixo e: anda para a esquerda d: anda para a direita v: voltar! e

x:-2 y:-3 c: anda para cima b: anda para baixo e: anda para a esquerda d: anda para a direita v: voltar! esq

x:-3 y:-3 c: anda para cima b: anda para baixo e: anda para a esquerda d: anda para a direita v: voltar! b

x:-3 y:-2 c: anda para cima b: anda para baixo e: anda para a esquerda d: anda para a direita v: voltar! b

x:-3 y:-1 c: anda para cima b: anda para baixo e: anda para a esquerda d: anda para a direita v: voltar! e

x:-4 y:-1 c: anda para cima b: anda para baixo e: anda para a esquerda d: anda para a direita v: voltar! e

x:-5 y:-1 c: anda para cima b: anda para baixo e: anda para a esquerda d: anda para a direita v: voltar! c

x:-5 y:-2 c: anda para cima b: anda para baixo e: anda para a esquerda d: anda para a direita v: voltar! d

x:-4 y:-2 c: anda para cima b: anda para baixo e: anda para a esquerda d: anda para a direita v: voltar! v

e b d d c c d d d b b b

x:0 y:0 c: anda para cima b: anda para baixo e: anda para a esquerda d: anda para a direita v: voltar! “‘

Avaliação Pesos de cada item dentro dos parênteses:

1. (0.25) Planejamento teórico da resolução do problema. Inclui, mas não se restringe a: * identificação correta de como usar a pilha, * planejamento do código a ser escrito, * planejamento das entradas e saídas do programa. 2. (0.30) TAD do tipo pilha escrita para este problema. Inclui, mas não se restringe a: * implementação correta dos métodos necessários, * implementação correta da struct ou similar, * implementação da TAD como uma biblioteca.

3. (0.25) Aplicação da pilha no problema proposto. Inclui, mas não se restringe a: * criação e chamada dos métodos da pilha nas situações adequadas. * interação adequada dos elementos que fazem parte do domínio do problema com a pilha.

4. (0.10) Funcionamento adequado do programa. Inclui, mas não se restringe a: * corretude entre as possíveis entradas e saídas do programa.

5. (0.10) Qualidade geral do código. Inclui, mas não se restringe a: * nome de funções adequados, * nome de variáveis adequadas, * clareza no uso de estruturas de repetição e condicionais.

APÊNDICE E – SÍLABOS RESUMIDOS DAS DISCIPLINAS DE PROGRAMAÇÃO ORIENTADA A OBJETOS

Tabela 29 – Sílabo resumido da disciplina de Programação Orientada a Objetos I do Ensino Médio

Curso	Técnico em Informática Integrado ao Ensino Médio
Componente Curricular	Programação Orientada a Objetos I
Carga Horária (hora-aula)	80
Período Letivo	2º ano
Ementa	Histórico e Paradigmas de Programação. Conceitos básicos de abstração e orientação a objetos (Objetos, classes, atributos e métodos). Herança e polimorfismo. Classes abstratas. Diagrama de classes. Conceito de exceções. A linguagem Java. Integração Java e Banco de Dados. Temas transversais conforme Res. CNE/CEB nº 02/2012. Exibição de filmes de produção nacional no contexto da disciplina conforme § 8º do artigo 26 da Lei nº 9.394/1996.
Bibliografia Básica	DEITEL, Harvey M.; DEITEL, Paul J. Java: Como Programar, 6ª Edição. 2005. SIERRA, Kathy; BATES, Bert. Use a cabeça!: Java. Alta Books, 2007. LEITE, Thiago et al. Orientação a Objetos: Aprenda seus conceitos e suas aplicabilidades de forma efetiva. Editora Casa do Código, 2016. ARNOLD, Ken; GOSLING, James. A linguagem de programação Java. Bookman Editora, 2009. BRAUDE, Eric. Projeto de software da programação à arquitetura: uma abordagem baseada em Java. Bookman Editora, 2009.
Bibliografia Complementar	BORATTI, Isaias C. Programação Orientada a Objetos em Java. Florianópolis: VisualBooks. 2007. BEZERRA, Eduardo. Princípios de Análise e Projeto de Sistemas com UML. Rio de Janeiro: Campus, 2003. COELHO, Pedro Alexandre. Programação em Java 2 - Curso Completo. Lisboa: FCA, 2002. HORSTMANN, Cay. Padrões e projetos orientados a objetos. Bookman Editora, 2009. SAMPAIO, Cleuton. SOA e Web services em Java. Brasport, 2006.

Fonte: Autoria própria

Tabela 30 – Sílabo resumido da disciplina de Programação Orientada a Objetos do Ensino Superior

Curso	Tecnologia em Análise e Desenvolvimento de Sistemas
Disciplina	POO0001 - PROGRAMAÇÃO ORIENTADA A OBJETOS
Carga horária	72
Período Letivo	2º ano
Ementa	Conceitos de orientação a objetos. Decomposição de programas. Generalização e especialização. Agregação e composição. Herança e polimorfismo. Projeto orientado a objetos. Estudo de uma linguagem.
Bibliografia Básica	DEITEL, H.M., DEITEL, P.J. Java: como programar. 3a ed. Porto Alegre: Bookman, 2003. KEOGH, J.,GRANNINI, M. OOP Desmistificado - Programação Orientada a Objetos. Alta Books, 2005. ISBN 8576080788. PAGE-JONES, M.; CONSTANTINE, L.L. O que todo programador deveria saber sobre projeto orientado a objeto. São Paulo Makron Books 1997.
Bibliografia Complementar	HORSTMANN, Cay S.; CORNELL, Gary. Core Java 2. São Paulo: Makron Books, c2003. 2 v. ISBN v.1. 8534612250 : v.2. 8 SANTOS, Rafael. Introdução à programação orientada a objetos usando JAVA. Rio de Janeiro: Elsevier, 2003. 319 p. : ISBN 853521206X (broch.)

Fonte: Autoria própria