Ministério da Educação
Universidade Tecnológica Federal do Paraná
Campus de Ponta Grossa

UMA PROPOSTA DE ABORDAGEM PRÁTICA BASEADA EM AUSUBEL PARA O ENSINO DE PROGRAMAÇÃO DE COMPUTADORES

JOÃO HENRIQUE BERSSANETTE

PONTA GROSSA

2016

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

Os cursos da área de computação e informática de nível superior ou técnico tem como uma de suas metas capacitar estudantes a apresentar soluções computadorizadas para diversos problemas do mundo real. Para produzir estas soluções os alunos devem utilizar comandos definidos a partir de uma linguagem de programação. Uma linguagem de programação tem um conjunto de símbolos e regras de sintaxe e semântica que permite descrever um conjunto de processos de forma precisa e que possam ser executadas por um computador.

Portanto, aprendizagem de programação é essencial para todas as carreiras ligadas à computação e a informática. Esta aprendizagem ocorre em uma série de disciplinas como algoritmos, lógica de programação, linguagem de programação, técnicas de programação, estrutura de dados, entre outras.

Estas disciplinas podem ser consideradas fundamentais para formação de alunos que terão no desenvolvimento de softwares o produto final de seu trabalho.

No entanto, o processo de ensino/aprendizagem de programação tem se demonstrado difícil para estudantes e professores, acarretando grandes índices de reprovação, desistência e abandono de cursos em instituições de ensino.

Isto ocorre devido a diversos problemas, dentre esses, a literatura aponta causas como a falta de competências na resolução de problemas, poucas habilidades matemáticas, baixo nível de abstração, dificuldades de interpretação do problema e compreensão de texto por parte dos estudantes.

Visando minimizar o impacto das dificuldades, diversas alternativas são propostas na literatura, existindo três caminhos gerais: (i) ferramentas, (ii) metodologias ou estratégias, (iii) ferramentas e metodologias associadas.

Este manual se enquadra na segunda categoria (metodologias ou estratégias), e se apoia na teoria da aprendizagem significativa de David Ausubel, para propor uma abordagem prática baseada em Ausubel para a apresentação dos conteúdos de programação aos estudantes.

Ausubel considera que o fator mais importante para que a aprendizagem ocorra, é determinar aquilo que o estudante já sabe ou conhece, para que estes conhecimentos prévios sirvam de ponto de ancoragem para os novos conhecimentos a serem adquiridos. Se considerarmos que para a maioria dos estudantes iniciantes em informática e computação a programação é um assunto completamente novo, a

ausência de conhecimentos prévios pode tornar o processo de ensino/aprendizagem mais delicado.

Assim, ao se trabalhar com um assunto desconhecido é importante resgatar alguma referência, para que sirvam de ponto de ancoragem para os novos conhecimentos, nesta proposta indicamos a utilização do computador como referência, visando tornar o conteúdo menos estranho aos estudantes.

O objetivo desta proposta é expor os alunos mais cedo ao uso prático do computador, enfatizando/valorizando a interação com a máquina para o ensino de programação.

Esta proposta se destina aos cursos da área da informática e computação, que conforme as Diretrizes Curriculares de Cursos da área de Computação e Informática:

> Os cursos da área de computação e informática tem como objetivos a formação de recursos humanos para o desenvolvimento científico e tecnológico da computação (hardware e software), para atuação na área de educação em computação em geral e para o desenvolvimento de ferramentas de informática que atendam a determinadas necessidades humanas. (BRASIL, 2001, p.4)

Sendo assim, visando atender ao objetivo de desenvolvimento de ferramentas de informática, a Programação de Computadores faz parte da área de formação básica dos cursos de Informática e Computação.

Os conteúdos presentes nas disciplinas de Programação, devem conter o ensino de linguagens de programação, seus conceitos, os princípios e os modelos de programação e o estudo de estruturas de dados e de métodos de classificação e pesquisa de dados (Santos & Costa, 2005).

Dessa forma, as disciplinas de programação de computadores, são caracterizadas como uma atividade voltada à solução de problemas, sendo relacionada com uma variada gama de outras atividades como especificação, projeto, validação, modelagem e estruturação de programas e dados, utilizando linguagens de programação como ferramentas. (Brasil, 2001)

Conforme o Quadro 1, todos os cursos de informática e computação devem cobrir os conceitos relacionados a programação de computadores, sendo que alguns cursos priorizam certos conteúdos ao detrimento de outros.

Matéria Programação	
Curso	Conteúdos
Bacharelado em Ciência da Computação Engenharia de Computação Bacharelado	As disciplinas devem cobrir, com abrangência e profundidade, pelo menos uma linguagem de programação desta matéria (primeira linguagem de programação). Devem cobrir também com abrangência e profundidade paradigmas de linguagens de programação, estrutura de dados e pesquisa e ordenação de dados.
Bacharelado em Sistemas de Informação Licenciatura em Computação	As disciplinas devem cobrir todas as principais linguagens de programação com abrangência e profundidade. Devem cobrir também com abrangência e profundidade estrutura de dados e pesquisa e ordenação de dados.

Quadro 1: Matéria Programação Cursos X Conteúdos
Fonte: Adaptado de Diretrizes Curriculares de Cursos da área de Computação e Informática (Brasil, 2001)

Com relação aos cursos técnicos na área de informática e Computação o Ministério da Educação (MEC), classifica os cursos de acordo com Eixos Tecnológicos, sendo que os cursos relacionados a esta pesquisa estão presentes no Eixo de Informação e Comunicação.

De acordo com o Catálogo Nacional de Cursos Técnicos (CNCT) o eixo de Informação e Comunicação conta com nove cursos, dentre estes cursos são especialmente relevantes para este estudo os cursos presentes no Quadro 2.

Matéria Programação	
Curso	Atividades
Técnico em Informática:	Desenvolve programas de computador, seguindo as especificações e paradigmas da lógica de programação e das linguagens de programação. Utiliza ambientes de desenvolvimento de sistemas, sistemas operacionais e banco de dados. Realiza testes de programas de computador, mantendo registros que possibilitem análises e refinamento dos resultados. Executa manutenção de programas de computadores implantados
Técnico em Informática para Internet:	Desenvolve programas de computador para internet, seguindo as especificações e paradigmas da lógica de programação e das linguagens de programação. Utiliza ferramentas de desenvolvimento de sistemas, para construir soluções que auxiliam o processo de criação de interfaces e aplicativos empregados no comércio e marketing eletrônicos. Desenvolve e realiza a manutenção de sites e portais na internet e na intranet.
Técnico em Programação de Jogos Digitais:	Compõe equipes multidisciplinares na construção dos jogos digitais. Utiliza técnicas e programas de computadores especializados de tratamento de imagens e sons. Desenvolve recursos, ambientes, objetos e modelos a ser utilizados nos jogos digitais. Implementa recursos que possibilitem a interatividade dos jogadores com os programas de computador. Integra os diversos recursos na construção do jogo.

Quadro 2: Cursos técnicos e atividades

Fonte: Adaptado de BRASIL. Mec, Setec. Catálogo Nacional de Cursos Técnicos, 2012.

Assim, os conteúdos como Lógica e Linguagens de Programação devem estar presentes nos currículos destes cursos, visto que os alunos terão na atividade de desenvolvimento de programas de computador as bases para sua formação.

2 REFERENCIAL TEÓRICO

Este capítulo apresenta os pressupostos teóricos que nortearam este trabalho.

Inicialmente apresentam-se alguns conceitos de programação de computadores, visando esclarecer ao leitor o contexto deste trabalho. Em seguida, é apresentada uma visão geral do processo de ensino de programação.

Por fim, discute-se a teoria da aprendizagem significativa de David Ausubel e como ela poderia contribuir no ensino/aprendizagem de programação de computadores.

2.1 PROGRAMAÇÃO DE COMPUTADORES

Os computadores podem ser considerados como uma ferramenta indispensável a vida moderna, estando presente nos mais variados segmentos. Esta máquina vem alterando significativamente nossas vidas nos mais variados contextos.

Hoje, para a maioria dos usuários o computador é algo simples, isso é possível devido aos avanços que o tornou cada vez mais intuitivo e de fácil utilização.

Um *software* ou programa de computador, é o resultado da compilação ou interpretação de um conjunto de instruções elaboradas para realização de uma tarefa ou resolução de um problema. Estas instruções devem seguir padrões de sintaxe para que possam ser compreendidas e executadas pelo computador.

O desenvolvimento da solução e implementação por meio da codificação engloba criar ou deduzir o raciocínio necessário para resolução do problema pelo computador.

Há diversas linguagens de programação com características específicas. A escolha da linguagem a ser utilizada está associada a diversos fatores, dentre os mais importantes pode-se citar o paradigma de desenvolvimento do *software.*

Um paradigma pode ser definido como um modelo de programação, ou seja, é um padrão utilizado pelo programador para orientar a maneira como o *software* será desenvolvido.

Geralmente o paradigma procedural é o mais comum para o ensino dos fundamentos de programação de computadores e será adotado nesta dissertação (Baranauskas, 1993).

Assim, programar nos diferentes paradigmas significa representar segundo padrões diferentes a solução do problema a ser resolvido. Pode-se dizer que um mesmo problema pode ser abordado empregando qualquer um dos paradigmas existentes, resultando em maior ou menor esforço do programador.

Portanto, programar computadores é uma atividade que requer o conhecimento do conteúdo que está sendo tratado, o domínio de uma linguagem de programação onde a solução do problema deve ser representada, e criatividade, uma vez que há sempre inúmeras maneiras de se chegar a uma solução por meio da programação. A programação de computadores é uma atividade exigente, que requer do programador certas habilidades a fim de que possa implementar soluções para um determinado problema e representa-las no ambiente computacional. A programação está relacionada com outras atividades como especificação, projeto e modelagem (Brasil, 2001).

2.2 ENSINO DE PROGRAMAÇÃO

No Brasil, o ensino de programação ocorre principalmente por cursos superiores nas áreas de Computação e Informática. Esse conteúdo também é trabalhado em certos cursos técnicos profissionalizantes subsequentes ou na modalidade integrada ao ensino médio.

Segundo as diretrizes curriculares para os cursos de informática e computação do MEC, a matéria de Programação faz parte da área de formação básica em Computação e Informática, juntamente com as matérias de Computação, Algoritmos e Arquitetura de computadores (Brasil, 2001).

Geralmente o processo inicial de ensino/aprendizagem de programação ocorre por meio de um conjunto de disciplinas introdutórias que são identificadas por nomes como: Algoritmos, Lógica de programação, Linguagem de programação, Técnicas de programação entre outras.

Estas disciplinas têm como objetivo fornecer aos alunos os conceitos básicos de programação, isto representa um pequeno conjunto de comandos e conceitos, dos quais os alunos devem utilizar para implementar soluções para um determinado problema e representa-las num ambiente computacional.

A programação é possivelmente uma das únicas matérias que tenta ensinar métodos de resolução de problemas gerais, visto que em cursos como filosofia, matemática e física, o estudante lê e aprende conceitos para se resolver problemas específicos não sendo capaz de generalizá-los (GRIES, 1974).

Alguns aspectos fundamentais em um curso introdutório seriam: 1º Como resolver problemas; 2º Como descrever uma solução algorítmica; 3º Como verificar se um algoritmo está correto (GRIES, 1974; GOMES; HENRIQUES; MENDES, 2008).

De modo geral, nestas disciplinas introdutórias, o conteúdo tratado inclui principalmente a descrição dos passos necessários para se solucionar um problema, entradas e saídas, constantes e variáveis, tipos primitivos de dados, instrução de atribuição, operadores aritméticos, relacionais e lógicos, estruturas simples de controle de fluxo, decisão e repetição. Em algumas salas de aula evita-se o contato com uma linguagem de programação nesse instante.

Neste contexto, atividades práticas podem ter uma importante função, pois conforme apontam (HABERMAN; MULLER, 2008; BENNEDSSEN; CASPERSEN, 2008), um dos entraves no ensino de programação e a forte carga de conceitos abstratos presentes no processo da elaboração e implementação da solução do problema no ambiente computacional.

Sendo assim, e importante oportunizar ao estudante que observe a execução de um programa e seus resultados, o que lhe oferece mais um caminho para tratar dificuldades encontradas em aspectos teóricos, uma vez que outra limitação encontrada em algumas salas de aula e justamente deixar de enfatizar a solução de problemas para se concentrar em conceitos teóricos (NOBRE; MENEZES, 2002; KOLIVER; DORNELES; CASA, 2004; GOMES; MENDES, 2007).

Outra característica interessante relacionada aos alunos, é que percebe-se que num determinado momento alguns deles apresentam um ganho significativo de aprendizagem e avançam sem problemas durante os conteúdos, e no entanto outros alunos mesmo se esforçando não conseguem atingir os objetivos propostos pela disciplina (Ambrósio, Almeida, Macedo, Santos, & Franco, 2011).

Dunican (2002), identifica três categorias de estudantes iniciantes em programação:

I. Alunos que não têm a aptidão para compreender os conceitos básicos, este é muitas vezes resultado de uma escolha equivocada do curso;
II. Alunos que podem captar os conceitos essenciais se expostos a abordagens de ensino eficazes; e
III. Alunos que são totalmente confortáveis com a natureza abstrata de conceitos de programação.

Seguindo essa classificação e assumindo que esteja correta o professor deve ficar atento, e se possível identificar os alunos pertencentes à segunda categoria, e assegurar condições adequadas de ensino/aprendizagem para que a maioria destes possam progredir. Entretanto, esperar que o processo de ensino/aprendizagem de programação, possa ser considerado apenas uma receita didática é seguramente um erro.

Conforme apresentado durante esta seção, é possível observar que o aprendizado de programação de computadores é uma das bases na formação de estudantes dos cursos de informática e computação. Entretanto aprender a programar computadores não é uma tarefa simples, tampouco trivial (Jenkins, 2002; Robins, Rountree, & Rountree, 2003).

Ao longo dos anos, o processo de ensino/aprendizagem dos fundamentos de programação de computadores, tem se mostrado difícil para estudantes e professores, estas dificuldades levaram muitos professores e pesquisadores a estudar suas causas e propor soluções variadas que visam de alguma maneira atenuar estes problemas. Grande parte destas pesquisas são voltadas especialmente para estudantes novatos em programação, como em Brusilovsky et al. (1994), Soloway et al. (1983), Delgado et al. (2004), Pereira Júnior; Rapkiewicz (2004), apenas para citar alguns trabalhos.

O grande volume de literatura referente à programação introdutória é reflexo das dificuldades relacionadas ao tema (Sheard, Simon, Hamilton, & Lönnberg, 2009) que faz com que o ensino de programação de computadores seja considerado um dos sete grandes desafios na educação em informática (Sleeman, 1986).

2.3 APRENDIZAGEM SIGNIFICATIVA

Aprendizagem significativa é o conceito central da teoria da aprendizagem de David Ausubel, proposta na década de 60, (Ausubel, Novak, & Hanesian, 1968; Ausubel, 1963), nesta teoria o autor destaca que a aprendizagem significativa é o mecanismo humano para adquirir e armazenar a vasta quantidade de ideias e informações representadas em qualquer campo de conhecimento.

Ausubel (2003, p.vi), destaca que:

> O conhecimento é significativo por definição. É o produto significativo de um processo psicológico cognitivo ("saber") que envolve a interação entre ideias "logicamente" (culturalmente) significativas, ideias anteriores ("ancoradas") relevantes da estrutura cognitiva particular do aprendiz (ou estrutura dos conhecimentos deste) e o "mecanismo" mental do mesmo para aprender de forma significativa ou para adquirir e reter conhecimentos.

Essa teoria preconiza que o conhecimento se organiza em estruturas cognitivas, que são conjuntos de conhecimentos que o indivíduo possui sobre um determinado assunto. A aprendizagem torna-se significativa quando os conhecimentos anteriores são inter-relacionados ao novo conteúdo a ser estudado o qual passa a ser incorporado às estruturas de conhecimento, adquirindo significado especial.

Este conceito não consiste numa simples associação e sim uma interação relevante entre os conhecimentos que se possui e os que será conhecido (Moreira, 1999).

Sendo assim, a aprendizagem significativa é um processo de mudança do conhecimento alterando a estrutura cognitiva do aprendiz, modificando os conceitos pré-existentes e criando novas conexões. É um processo onde a configuração da estrutura cognitiva passa de um estado a outro.

Nessa perspectiva, novos conhecimentos são construídos à medida que o aprendiz se movimenta no sentido de articular novos saberes aos que já possui, assim, aprendizagem precisa ser ancorada a uma outra já existente na estrutura cognitiva do sujeito para que possa ser então assimilada.

Portanto, um dos pontos fundamentais desta teoria consiste em determinar aquilo que o aprendiz já sabe ou conhece (MASINI & Moreira, 2001), ou seja, o estado atual da sua estrutura, para que a proposta de ensino seja baseada nestes conhecimentos.

No caso do aprendiz não possuir conhecimentos prévios sobre o novo conceito a aprendizagem não ocorrerá de maneira significativa, pois não resulta na aquisição de significados para o sujeito. Quando isso ocorre dá-se o nome de aprendizagem mecânica ou automática.

A aprendizagem mecânica, ou automática é o contrário da aprendizagem significativa: a aprendizagem de novas informações ocorre com pouca ou nenhuma associação com conceitos relevantes existentes na estrutura cognitiva, o que fará com que estes conhecimentos sejam esquecidos com maior facilidade (MASINI & Moreira, 2001) .

Para evitar isso, Ausubel sugere deliberadamente manipular a estrutura cognitiva usando organizadores prévios (Moreira & Sousa, 1996). Organizadores prévios tem a função de ponte entre o que o aprendiz sabe e o que deve saber. Eles visam estabelecer relações entre ideias, proposições e conceitos já existentes na estrutura cognitiva, a fim de que a aprendizagem possa ser significativa. Além disso, também podem ser usados para “reativar” significados obliterados, para “buscar” na estrutura cognitiva do aluno significados que já existiam, mas que não eram usados há algum tempo.

Para que a aprendizagem seja significativa há três condições: predisposição do indivíduo; material potencialmente significativo e estrutura cognitiva capaz de assimilar a nova informação (AusubeL, Novak, & Hanesian, 1980).

A predisposição para aprender está intimamente relacionada com a experiência afetiva que o aprendiz tem no evento educativo (NOVAK, 2000). Além disso, a aprendizagem significativa propõe a participação ativa do aluno na aquisição de conhecimento, de maneira a evitar-se uma mera reprodução de conceitos formulados pelo professor ou pelo livro-texto, mas uma reelaboração do aluno (Pelizzari & Kriegl, 2002).

Para Ausubel, cada disciplina tem uma estrutura articulada e hierarquicamente organizada de conceitos (MASINI & Moreira, 2001). No entanto, a ordem em que os principais conceitos e ideias da matéria de ensino são apresentadas muitas vezes não é a mais adequada para facilitar a interação com o conhecimento prévio do aluno.

Por isso, é essencial uma análise crítica da matéria de ensino a ser apresentada ao estudante; o conteúdo precisa ter boa organização lógica, cronológica

e epistemológica, mas além disso é indispensável uma análise com foco no estudante e em particular seu conhecimento prévio.

Além disso, é importante também não sobrecarregar o aluno de informações desnecessárias, dificultando a organização cognitiva. É preciso buscar a melhor maneira de relacionar, explicitamente, os aspectos mais importantes do conteúdo da matéria de ensino aos aspectos especificamente relevantes de estrutura cognitiva do aprendiz. Este relacionamento é imprescindível para a aprendizagem significativa.

O adiamento da experiência de aprendizagem para além da maturidade do estudante desperdiça oportunidades de aprendizagens valiosas e, muitas vezes, reduzindo de forma desnecessária, a quantidade e complexidade dos conteúdos que se pode dominar num determinado período da aprendizagem escolar. Por outro lado, quando um aluno é exposto, prematuramente, a uma tarefa de aprendizagem, antes de estar preparado de forma adequada para a mesma, não só não aprende a tarefa em questão (ou aprende-a com muitas dificuldades), como também aprende com esta experiência a temer, desgostar e evitar a tarefa (Ausubel, 2003).

Neste sentido, se faz necessário determinar continuamente o que o aprendiz conhece para ensiná-lo de acordo. Para tal, faz-se necessário mecanismos que visem identificar o estado mental de cada aprendiz, relativo ao domínio de conhecimento em questão, para que possam auxiliar os professores nesta tarefa.

Ausubel enfoca a linguagem como um facilitador importante para a ocorrência da aprendizagem significativa (Moreira, 1983). Desta forma, é possível observar que os conceitos abordados serão realmente assimilados, se eles forem apresentados numa linguagem coerente e que faça sentido para o aprendiz.

3 PROPOSTA DE ABORDAGEM

Antes da apresentação da descrição da proposta de abordagem elaborada, cabe compará-la à abordagem tradicional e pontuar alguns aspectos, para isso é apresentado a seguir o

Elementos	Abordagem Tradicional	Abordagem Proposta
Exposição conteúdos	Frequentemente teórica-conceitual	Prática; informação teórica surge para explicar mecanismos.
Diferentes assuntos	Exposição de comandos isolados.	Integração (FOR usando IF, etc..)
Sequenciamento	Fortemente linear, conforme atestam diários de classe	Cíclico: mesmo comando é discutido várias vezes em diferentes contextos.
Atividades práticas em laboratório	Poucas	Muitas
Representação de soluções	Fluxogramas e pseudocódigo	Linguagens de programação
Resolução de problemas	Apresentação de soluções prontas	Estimulo a proposição de soluções
Atividades para o desenvolvimento de experiências em programação	Listas de exercícios	Desafios semanais
Perspectiva do aluno em relação ao conteúdo apresentado	Tendência a vê-lo de forma mais Abstrata	Atividades práticas para introduzir ou confirmar conceitos teóricos.

Quadro 3.

Elementos	Abordagem Tradicional	Abordagem Proposta
Exposição conteúdos	Frequentemente teórica-conceitual	Prática; informação teórica surge para explicar mecanismos.
Diferentes assuntos	Exposição de comandos isolados.	Integração (FOR usando IF, etc..)
Sequenciamento	Fortemente linear, conforme atestam diários de classe	Cíclico: mesmo comando é discutido várias vezes em diferentes contextos.
Atividades práticas em laboratório	Poucas	Muitas
Representação de soluções	Fluxogramas e pseudocódigo	Linguagens de programação
Resolução de problemas	Apresentação de soluções prontas	Estimulo a proposição de soluções

Atividades para o desenvolvimento de experiências em programação	Listas de exercícios	Desafios semanais
Perspectiva do aluno em relação ao conteúdo apresentado	Tendência a vê-lo de forma mais Abstrata	Atividades práticas para introduzir ou confirmar conceitos teóricos.

Quadro 3: Abordagem Tradicional X Abordagem Proposta
Fonte: Autoria própria

Na abordagem tradicional geralmente ocorre a teorização dos conteúdos introdutórios de programação de computadores, aula após aula o professor apresenta conceitos aos estudantes. No entanto, a teorização dos conteúdos introdutórios de programação de computadores num primeiro momento é pouco relevante para os alunos, visto que para grande maioria dos alunos a programação de computadores é um assunto novo, portanto os mesmos não possuem os subsunçores necessários.

Deste modo, nesta etapa do processo, a exposição insistente a conceitos e teorias pode levar a um aprendizado mecânico. Um forte indício disso é a dificuldade dos alunos de construírem programas, embora em sala declarem ter entendido e demonstrem saber ler códigos.

Em nossa proposta de abordagem, durante as primeiras semanas, em nenhum momento esses conceitos são expostos de forma teórica, os estudantes são expostos a situações práticas que oportunizam a eles desenvolver seus próprios conceitos.

Outro aspecto diz respeito a como os conteúdos geralmente são trabalhados na abordagem tradicional. Nesta abordagem geralmente o professor separa os conteúdos em caixas, e apresenta aos estudantes de maneira isolada, normalmente a estrutura básica, a sequência, os comandos de entrada e saída, variáveis, estrutura de decisão, repetição e etc.

Nossa proposta visa apresentar os mesmos conteúdos os mesmos conteúdos descritos anteriormente mas de maneira integrada. Pois é assim que a programação acontece, ou seja, na hora de se solucionar um problema se faz necessário que os estudantes relacionem esses elementos e isto acaba por se tornar um problema.

Além disso, na abordagem tradicional os conteúdos possuem uma sequência, ou seja, tem uma estrutura articulada e hierarquicamente organizada de conceitos, no entanto, a ordem em que os principais conceitos e ideias da matéria de ensino são

apresentadas muitas vezes não é a mais adequada para facilitar a interação com o conhecimento prévio do aluno.

Assim, a apresentação dos conteúdos presente na proposta de abordagem, se dá de forma cíclica, isto se faz necessário uma vez que cada aluno irá formar sua própria estrutura cognitiva e progredir em uma velocidade diferente. Muitos alunos não possuem os subsunçores, maturidade ou as competências necessárias para se aprimorar estes conteúdos na primeira vez que estes são apresentados, de forma cíclica os conteúdos são apresentados por diversas vezes, em momentos e contexto diferentes o que pode favorecer o processo.

É possível observar que as pessoas constroem e reconstroem o seu conhecimento ao longo da vida, e os estudantes que não aprendem satisfatoriamente o que tentamos ensinar, muito provavelmente não o fazem por não terem o conhecimento prévio necessário para uma aprendizagem significativa (Braathen, 2003).

Com relação as atividades práticas, normalmente na abordagem tradicional o professor faz pouco uso dos laboratórios, seja porque a instituição não oferece um suporte adequado ou pela grande quantidade de alunos presentes em sala de aula no início da disciplina.

A proposta de abordagem preza por grande quantidade de atividades práticas em laboratório (se possível todas as aulas), visto que ao se utilizar o computador não se exige do estudante imaginar o funcionamento de algum comando, a execução de um comando pela máquina adquire um caráter quase concreto, possibilitando o estudante a experimentar hipóteses e sedimentar seus conceitos. Comparando com abordagem tradicional o estudante só irá confirmar o que viu mais tarde ou outro dia em laboratório.

Na abordagem tradicional inicialmente é solicitado aos estudantes que descrevam processos que em muitos casos não tem nenhuma relação com a programação de computadores, (exemplo: algoritmo para trocar lâmpada, vir de casa até a escola, etc.), na sequência geralmente o professor passa a utilizar pseudocódigo ou fluxograma para representar pequenos programas.

Deve-se tomar cuidado também com relação a escolha do ambiente e linguagem programação, tendo em vista que o foco principal é a aprendizagem dos conteúdos e não a sintaxe de uma determinada linguagem específica.

Em nossa proposta inicialmente utilizamos linguagens de sintaxe simples como o *BASIC* e *PYTHON*, como sempre enfatizando os conteúdos da programação de computadores, recomendando que a partir do momento em que seja detectado que as aprendizagens destes conteúdos sejam satisfatórias, seja apresentado aos alunos ambientes e linguagens de sintaxe mais complexa, como C.

É comum também na abordagem tradicional a apresentação de soluções prontas, como por exemplo sequência Fibonacci, calculo fatorial e outros, nestes exemplos o professor descreve os processos da solução, ficando o estudante limitado a apresentar um único tipo de solução a correta e na maioria das vezes seguindo apenas o raciocínio do professor, a abordagem proposta estimula a proposição de soluções, desta forma o aluno fica livre para criar sua própria proposta de solução e até mesmo formular outros problemas derivados do problema inicial.

Desta forma deve-se tomar cuidado em algumas etapas do processo, visto que alguns alunos podem ter dificuldades na hora de propor soluções, uma atenção especial neste contexto deve ser dada a motivação e ao relacionamento professor-aluno, pois caso contrário, o aluno pode se sentir desestimulado e ficar alheio ao processo.

Portanto, o professor deve ser muito flexível e tentar estimular ao máximo a proposição de soluções mesmo que estas não sejam as corretas, o aluno deve ser instigado a realizar operações e atividades de alteração de códigos, e partir destas o professor deve questiona-los sobre o que o programa produz.

Dessa forma, espera-se que os estudantes possam desenvolver seu próprio repertório de conhecimentos e habilidades, podendo assim resolver as atividades propostas.

Na abordagem tradicional é comum professores se utilizarem de listas de exercícios visando incentivar os alunos a adquirirem uma maior experiência em programação, entretanto, vivemos em uma época em que tudo é compartilhado e que é difícil não se encontrar a solução para um problema de uma lista de exercícios pela internet, além disso, observa-se que em alguns caso um único estudante faz a lista de exercícios e compartilha com os demais colegas, isto acontece devido a vários motivos. A proposta de abordagem estimula os alunos a adquirirem essa experiência em programação por meio de desafios semanais, que na grande maioria das vezes é a solução para um único problema desenvolvido em laboratório, cujo não é dada a

solução, e isso estimula a curiosidades dos alunos em relação a resolução do problema e a competividade fazendo com que eles não compartilhem suas soluções.

Por fim, sintetizando a proposta de abordagem, esta consiste na apresentação de pequenos trechos de códigos simples, que devem ser examinados e modificados pelos alunos, usando uma linguagem de programação.

Dessa forma, primeiramente o professor realiza a exposição de um comando ou trecho de um programa no projetor. Em seguida questiona os alunos sobre como obter um resultado ligeiramente diferente na tela por meio de modificação do comando ou trecho. Neste momento o professor deve ter atenção para não podar as soluções propostas pelos alunos mesmo que estas sejam erradas, após breve discussão, os alunos deveriam sugerir modificações nestes comandos ou programas, os alunos eram instigados a modificar os programas e perceberem o que acontecia, visando chegar a uma solução satisfatória, neste momento os alunos também eram incentivados a compartilhar suas soluções e explicar o que eles fizeram.

4 PLANOS DE AULA

Os planos de aula que serão apresentados no decorrer desta seção, foram desenvolvidos a partir da proposta abordagem objeto deste estudo, para cada aula sugere-se uma carga horária de duas horas aula em laboratório de informática.

A estrutura dos roteiros foi inspirada no trabalho de (FELDER; BRENT, 2003). Os roteiros contém os objetivos da aula, conteúdos e atividades a serem realizadas. Cada aula não trata de um assunto específico devido a integração dos conteúdos, também não se estabelece rigidamente o tempo que o professor irá usar para sua aplicação, pois depende de uma série de fatores como por exemplo a turma, laboratório, entre outros.

Os conteúdos abordados nestes roteiros com a proposta de abordagem são: Comandos de entrada e saída; Variáveis; Estrutura de decisão; Estrutura de repetição; Implementação de problemas numa linguagem de programação.

A seguir é apresentado os roteiros de cinco aulas com a proposta de abordagem.

<table>
<tr><th colspan="2">Aula 01</th></tr>
<tr><td colspan="2">Objetivos:
Identificar os conhecimentos prévios dos alunos referente a programação de computadores;
Apresentar a área de trabalho da ferramenta Decimal Basic
Apresentar por meio de atividades práticas os comandos de entrada e saída; Variáveis; Estrutura de repetição;
Implementar problemas numa linguagem de programação.</td></tr>
<tr><th>Conteúdos</th><th>Atividades</th></tr>
<tr><td>Conhecimentos prévios</td><td>Fazer levantamentos dos conhecimentos em programação de cada aluno.</td></tr>
<tr><td>Download da ferramenta</td><td>Fazer download da ferramenta Decimal Basic por meio do link http://www.geocities.jp/thinking_math_education/EnglishWindows.htm</td></tr>
<tr><td>Ferramenta</td><td>Apresentar a área de trabalho da ferramenta.</td></tr>
<tr><td rowspan="5">PRINT</td><td>Mostrar comando PRINT “nome”</td></tr>
<tr><td>Solicitar aos alunos que modifiquem o comando PRINT para imprimir outra coisa PRINT “outra coisa”</td></tr>
<tr><td>Apresentar o seguinte código:<pre>PRINT 123
PRINT 100 + 200
PRINT “100 + 200”
END</pre></td></tr>
<tr><td>Questionar aos alunos o resultado da execução do código.</td></tr>
<tr><td>Explicar diferença entre caracteres e números ‘de verdade’</td></tr>
<tr><td rowspan="2">Variáveis, PRINT</td><td>Apresentar o seguinte código:<pre>N = 10
PRINT N
PRINT N * 2
END</pre></td></tr>
<tr><td>Explicar aos alunos o que é uma variável.</td></tr>
<tr><td rowspan="3">Laço FOR, Variáveis, PRINT</td><td>Apresentar o seguinte código:<pre>FOR i=0 to 10
   PRINT “NOME”
NEXT i
END</pre></td></tr>
<tr><td>Questionar aos alunos o resultado da execução do código.</td></tr>
<tr><td>Solicitar aos alunos que modifiquem o código para imprimir 15 vezes, 5 vezes e 100 vezes</td></tr>
</table>

<table>
<tr><th colspan="2">Aula 01 – continuação</th></tr>
<tr><th>Conteúdos</th><th>Atividades</th></tr>
<tr><td rowspan="5">Laço FOR, Variáveis, PRINT</td><td>Apresentar o seguinte código:
<pre>FOR J = 1 to 10
   PRINT J
NEXT J
END</pre></td></tr>
<tr><td>Questionar aos alunos o resultado da execução do código.</td></tr>
<tr><td>Solicitar aos alunos que modifiquem o código para imprimir J de 5 até 10 e para imprimir de 10 até 50</td></tr>
<tr><td>Solicitar aos alunos que modifiquem o código para imprimir de 10 até 1, grande parte dos alunos irá propor a solução FOR J = 10 to 1 (o que não irá funcionar), apresentar o código para solução do problema proposto:
<pre>FOR I = 1 TO 10
   PRINT 10 - I
NEXT I
END</pre></td></tr>
<tr><td>Explicar aos alunos que em programação, muitas coisas têm que ser ‘ajustadas’.</td></tr>
<tr><td rowspan="4">Laço FOR, Variáveis, PRINT, Variante STEP</td><td>Explicar que isso se aprende com o tempo e experiência em programação.</td></tr>
<tr><td>Solicitar aos alunos que modifiquem o código para imprimir todos os números impares de 1 a 33, e na sequência todos os números pares de 0 a 44.</td></tr>
<tr><td>Apresentar o seguinte código:
<pre>FOR J = 39 to 0 step -2
   PRINT J
NEXT J
END</pre></td></tr>
<tr><td>Questionar aos alunos o resultado da execução do código.</td></tr>
<tr><td>Solicitar aos alunos que modifiquem os códigos feitos anteriormente para imprimir todos os números impares de 1 a 33, e na sequência todos os números pares de 0 a 44, incluindo a variante STEP</td></tr>
<tr><td>Desafio Semanal I</td><td>Solicitar os alunos que desenvolvam o código que mostre na tela o seguinte resultado:
<pre>11
22
33
44
55
66
77
88
99</pre></td></tr>
</table>

Quadro 4: Aula 01 com a abordagem proposta

Fonte: Autoria própria.

<table>
<tr><th colspan="2">Aula 02</th></tr>
<tr><td colspan="2">Objetivos:
Avaliar os conhecimentos dos alunos referente aos conteúdos apresentados na aula anterior;
Reforçar os conceitos por meio de atividades práticas os comandos de entrada e saída; Variáveis; Estrutura de repetição;
Apresentar por meio de atividades práticas a Estrutura de decisão;
Implementar problemas numa linguagem de programação.</td></tr>
<tr><th>Conteúdos</th><th>Atividades</th></tr>
<tr><td>Conhecimentos prévios</td><td>Questionar aos alunos quais comandos foram apresentados até o momento e anotar em local visível para os alunos.</td></tr>
<tr><td>Desafio Semanal I</td><td>Verificar quais alunos conseguiram resolver o Desafio Semanal I, com base nas diferentes propostas de solução para problema desenvolvidas pelos alunos, apresentar aos demais estudantes e pedir que eles comentem as soluções propostas</td></tr>
<tr><td rowspan="5">PRINT</td><td>Apresentar o seguinte código:<pre>PRINT 1
PRINT 2,
PRINT 3;
PRINT 4;
PRINT 5,
PRINT 6
PRINT 7
END</pre></td></tr>
<tr><td>Questionar aos alunos o resultado da execução do código.<pre>1
 2   3 4 5   6
 7</pre></td></tr>
<tr><td>Explicar aos estudantes que muitas coisas em computação acabam sendo deduzidas, observando os códigos e resultados apresentados pela máquina.</td></tr>
<tr><td>Explicar que é interessante fazer pequenos testes para aprender coisas novas.</td></tr>
<tr><td>Solicitar aos alunos que modifiquem os códigos feitos anteriormente para que o código mostre na tela o seguinte resultado:<pre>1 2 3 4 5 6 7 8 9</pre></td></tr>
<tr><td>Laço FOR, Variáveis, PRINT</td><td>Solicitar aos alunos que modifiquem os códigos feitos anteriormente para que o código mostre na tela o mesmo resultado, mas com a seguinte restrição: Usar apenas um comando PRINT N</td></tr>
<tr><td rowspan="2">Variáveis, INPUT, PRINT</td><td>Apresentar o seguinte código:<pre>INPUT nome$
PRINT nome$
END</pre></td></tr>
<tr><td>Questionar aos alunos o resultado da execução do código.</td></tr>
</table>

<table>
<tr><th colspan="2">Aula 02 – continuação 1</th></tr>
<tr><th>Conteúdos</th><th>Atividades</th></tr>
<tr><td rowspan="4">Variáveis, INPUT, PRINT</td><td>Apresentar o seguinte código:
<pre>INPUT x
INPUT y
PRINT X+Y
END</pre></td></tr>
<tr><td>Questionar aos alunos o resultado da execução do código.</td></tr>
<tr><td>Explicar aos estudantes as diferenças no armazenamento de dados de variáveis Números X Caracteres</td></tr>
<tr><td>Solicitar aos alunos que modifiquem os códigos apresentados para que seja feita a soma de 5 números</td></tr>
<tr><td rowspan="2">Variáveis, Laço For, PRINT, INPUT, Acumulador</td><td>Solicitar aos alunos que modifiquem os códigos apresentados para que seja feita a leitura e soma de 10 números, utilizando apenas um comando INPUT e 2 variáveis, os alunos têm 90 segundos para fazer isso, quem conseguir ganha um doce</td></tr>
<tr><td>Apresentar a solução para o problema proposto, explicar o que é um acumulador:
<pre>LET soma = 0

FOR i = 1 TO 5

   PRINT "diga um número"
   INPUT n

   LET soma = soma + n

NEXT i

PRINT "total = "; soma

END</pre></td></tr>
<tr><td rowspan="4">Variáveis, INPUT, IF, PRINT, Operadores relacionais</td><td>Apresentar o seguinte código:
<pre>INPUT N

IF (N>10) THEN
   PRINT "Maior que 10"
END IF

IF (N<10) THEN
   PRINT "Menor que 10"
END IF
END</pre></td></tr>
<tr><td>Questionar aos alunos o resultado da execução do código.</td></tr>
<tr><td>Explicar que o IF só executará o código que tem dentro dele caso a condição seja verdadeira.</td></tr>
<tr><td>Solicitar aos alunos que modifiquem os códigos para que seja apresentada a mensagem Igual a 10.</td></tr>
</table>

<table>
<tr><th colspan="2">Aula 02 – continuação 2</th></tr>
<tr><th>Conteúdos</th><th>Atividades</th></tr>
<tr><td>Variáveis, INPUT, IF, ELSE, PRINT</td><td>Apresentar o seguinte código:
<pre>INPUT N

IF (N>10) THEN
   PRINT "Maior que 10"
ELSE
   PRINT "Menor que 10"
END IF

END</pre>Questionar aos alunos o resultado da execução do código.</td></tr>
<tr><td rowspan="2">Variáveis, INPUT, IF, ELSE, PRINT</td><td>Explicar que o ELSE só executará o código que tem dentro dele caso a condição a condição do IF seja falsa. E que um ELSE por receber outros IF dentro dele.</td></tr>
<tr><td>Solicitar aos alunos que modifiquem os códigos para que seja apresentada também a mensagem Igual a 10.</td></tr>
<tr><td>Desafio Semanal II</td><td>Solicitar os alunos que desenvolvam o código que mostre na tela o seguinte resultado:
<pre>XXX
XXX
XXX</pre>Usando apenas um único comando PRINT “X”</td></tr>
</table>

Quadro 5: Aula 02 com a abordagem proposta

Fonte: Autoria própria

<table>
<tr><th colspan="2">Aula 03</th></tr>
<tr><td colspan="2">Objetivos:
Avaliar os conhecimentos dos alunos referente aos conteúdos apresentados até o momento;
Reforçar os conceitos por meio de atividades práticas os comandos de entrada e saída; Variáveis; Estrutura de repetição; Estrutura de decisão;
Apresentar por meio de atividades práticas o conceito de Identação do código e os comandos RANDOMIZE e EXIT FOR;
Implementar problemas numa linguagem de programação.</td></tr>
<tr><th>Conteúdos</th><th>Atividades</th></tr>
<tr><td>Conhecimentos prévios</td><td>Questionar aos alunos quais comandos foram apresentados até o momento e anotar em local visível para os alunos.</td></tr>
<tr><td>Desafio Semanal II</td><td>Verificar quais alunos conseguiram resolver o Desafio Semanal II, com base nas diferentes propostas de solução para problema desenvolvidas pelos alunos, apresentar aos demais estudantes e pedir que eles comentem as soluções propostas</td></tr>
<tr><td>Estrutura do programa e identação</td><td>A partir desta aula, enfatizar uso de espaços em branco entre linhas de comandos e TAB para comandos que estejam dentro das estruturas FOR, IF, ELSE.</td></tr>
<tr><td rowspan="4">RANDOMIZE, PRINT, RND</td><td>Apresentar o seguinte código:<pre>RANDOMIZE
PRINT RND
END</pre></td></tr>
<tr><td>Questionar aos alunos o resultado da execução do código, solicitar que os alunos executem o código diversas vezes até eles perceberem que o RND gera um número aleatório.</td></tr>
<tr><td>Solicitar aos alunos que modifiquem os códigos para que mostre na tela um número entre 0 e 10. Lembra-los que em programação, muitas coisas têm que ser ‘ajustadas’.</td></tr>
<tr><td>Apresentar a solução para o problema proposto:<pre>RANDOMIZE
PRINT 10*RND
END</pre></td></tr>
<tr><td rowspan="2">RANDOMIZE, PRINT, INT, RND</td><td>Solicitar aos alunos que modifiquem os códigos para que mostre na tela um número entre 7 e 10. Os alunos têm 90 segundos para fazer isso, quem conseguir ganha um doce</td></tr>
<tr><td>Apresentar a solução para o problema proposto:<pre>RANDOMIZE
PRINT INT (7+3*RND)
END</pre></td></tr>
<tr><td>Questionar aos alunos o resultado da execução do código, explicar que o comando INT transforma um valor real num valor inteiro</td></tr>
</table>

<table>
<tr><th colspan="2">Aula 03 – continuação 1</th></tr>
<tr><th>Conteúdos</th><th>Atividades</th></tr>
<tr><td>Variáveis, RANDOMIZE, INT, RND, Laço FOR, IF, ELSE</td><td>Solicitar aos alunos que desenvolvam um programa com as seguintes características:
Escrever um programa que sorteia um número inteiro aleatório X entre 1 e 10 e guarda na variável X
Depois o programa lê um número na variável N cinco vezes, em um laço
Cada vez que o programa lê um número, ele mostra a diferença entre N e X.
Se aparecer o número zero no vídeo, significa que o usuário descobriu o valor de X
Dar um bom tempo para que os alunos desenvolvam suas soluções.</td></tr>
<tr><td>Variáveis, RANDOMIZE, INT, RND, Laço FOR, PRINT, INPUT</td><td>Apresentar a solução para o problema proposto:
<pre>RANDOMIZE
LET X = INT (10*RND)

FOR i = 1 TO 5
   PRINT "digite um número";
   INPUT n
   PRINT x-n
NEXT i

END</pre></td></tr>
<tr><td rowspan="2">Variáveis, RANDOMIZE, INT, RND, Laço FOR, PRINT, INPUT, IF, ELSE</td><td>Solicitar aos alunos que reescrevam a solução para o problema anterior, acrescentando os comandos IF e ELSE, para informar ao usuário as seguintes mensagens: Em caso de acerto “Parabéns! Que lindo!”, em caso de erro “Errou! Tente novamente”.
Dar um bom tempo para que os alunos desenvolvam suas soluções.</td></tr>
<tr><td>Apresentar a solução para o problema proposto:
<pre>RANDOMIZE
LET X = INT (10*RND)

FOR i = 1 TO 5
   PRINT "digite um número";
   INPUT n

   IF n = x THEN
      PRINT "Parabéns! Que lindo!"
   ELSE
      PRINT "Errou! Tente novamente."
   END IF
NEXT i

END</pre></td></tr>
</table>

<table>
<tr><th colspan="2">Aula 03 – continuação 2</th></tr>
<tr><th>Conteúdos</th><th>Atividades</th></tr>
<tr><td rowspan="2">Variáveis, RANDOMIZE, INT, RND, Laço FOR, PRINT, INPUT, IF, ELSE, EXIT FOR</td><td>Apresentar o comando EXIT FOR (talvez seja pedido pelos alunos no programa anterior)
<pre>RANDOMIZE
LET X = INT (10*RND)

FOR i = 1 TO 5
   PRINT "digite um número";
   INPUT n

   IF n = x THEN
      PRINT "Parabéns! Que lindo!"
      EXIT FOR
   ELSE
      PRINT "Errou! Tente novamente."
   END IF
NEXT i

END</pre></td></tr>
<tr><td rowspan="2">Variáveis, RANDOMIZE, INT, RND, Laço FOR, PRINT, INPUT, IF, ELSE</td><td>Solicitar aos alunos que modifiquem o programa anterior para a cada vez que o usuário tenta acertar o número, mas errar, o programa deverá ‘zoar’ com o jogador, cada vez apresentando uma frase diferente. Dar um bom tempo para que os alunos desenvolvam suas soluções.</td></tr>
<tr><td>Apresentar a solução para o problema proposto:
<pre>RANDOMIZE
LET X = INT(10*RND)

FOR i = 1 TO 5

   PRINT "digite um número";
   INPUT n

   IF n = x THEN
      PRINT "Parabéns! Que lindo!"
      EXIT FOR
   ELSE
      IF i = 1 then
          PRINT "Errou!"
      end if
      IF i = 2 then
          PRINT "não, criatura!"
      end if
      IF i = 3 then
          PRINT "ai que tristeza"
      end if

   END IF
NEXT i
END</pre></td></tr>
</table>

Aula 03 – continuação 3	
Conteúdos	**Atividades**

Desafio Semanal III	Solicitar os alunos que desenvolvam o código que mostre na tela o seguinte resultado:
	`x.`
	`xx.`
	`xxx.`
	`xxxx.`
	`xxxxx.`
	Usando apenas um único comando PRINT “X” e PRINT “.”

Quadro 6: Aula 03 com a abordagem proposta

Fonte: Autoria própria

<table>
<tr><th colspan="2">Aula 04</th></tr>
<tr><td colspan="2">Objetivos:
Avaliar os conhecimentos dos alunos referente aos conteúdos apresentados até o momento;
Reforçar os conceitos por meio de atividades práticas os comandos de entrada e saída; Variáveis; Estrutura de repetição; Estrutura de decisão;
Apresentar por meio de atividades práticas os conceitos de contador e acumulador;
Implementar problemas numa linguagem de programação.</td></tr>
<tr><th>Conteúdos</th><th>Atividades</th></tr>
<tr><td>Conhecimentos prévios</td><td>Questionar aos alunos quais comandos foram apresentados até o momento e anotar em local visível para os alunos.</td></tr>
<tr><td>Desafio Semanal III</td><td>Verificar quais alunos conseguiram resolver o Desafio Semanal III, com base nas diferentes propostas de solução para problema desenvolvidas pelos alunos, apresentar aos demais estudantes e pedir que eles comentem as soluções propostas</td></tr>
<tr><td rowspan="2">Variáveis, PRINT, INPUT, IF, ELSE</td><td>Solicitar aos alunos que desenvolvam um programa que leia dois valores e mostre na tela uma das três mensagens a seguir:
‘Números iguais’, caso os números sejam iguais;
‘Primeiro é maior’, caso o primeiro seja maior que o segundo;
‘Segundo maior’, caso o segundo seja maior que o primeiro.
Os alunos têm 5 minutos para desenvolver a solução para este problema, aquele que terminar primeiro ganha um doce.</td></tr>
<tr><td>Apresentar a solução para o problema proposto:<pre>
LET n1=0
LET n2=0

PRINT "Digite um valor para n1..: "
INPUT n1

PRINT "Digite um valor para n1..: "
INPUT n2

IF n1=n2 THEN
   PRINT "Números iguais"
ELSE
   IF n1>n2 THEN
      PRINT "Primeiro é maior"
   ELSE
      PRINT "Segundo é maior"
   END IF
END IF

END
</pre></td></tr>
</table>

<table>
<tr><th colspan="2">Aula 04 – continuação 1</th></tr>
<tr><th>Conteúdos</th><th>Atividades</th></tr>
<tr><td rowspan="2">Variáveis, PRINT, INPUT</td><td>Solicitar aos alunos que desenvolvam um programa que leia dois valores e ao término do programa mostre a soma destes valores. Os alunos têm 5 minutos para desenvolver a solução para este problema, aquele que terminar primeiro ganha um doce.</td></tr>
<tr><td>Apresentar a solução para o problema proposto:
<pre>LET soma=0
LET n1=0
LET n2=0

PRINT "Digite um valor..: "
INPUT n1
PRINT "Digite outro valor..: "
INPUT n2
LET soma = n1 + n2

PRINT "A soma dos valores é..: "; soma
END</pre></td></tr>
<tr><td rowspan="2">Variáveis, Laço FOR, PRINT, INPUT</td><td>Solicitar aos alunos que alterem o programa anterior para que leia uma quantidade de valores informada pelo usuário e ao término do programa mostre a soma destes valores e a média. Os alunos têm 5 minutos para desenvolver a solução para este problema, aquele que terminar primeiro ganha um doce.</td></tr>
<tr><td>Apresentar a solução para o problema proposto:
<pre>LET soma=0
LET n=0
LET q=0

INPUT q

FOR i=1 TO q
   PRINT "Digite um valor..: "
   INPUT n

   LET soma = soma + n
NEXT i

PRINT "A soma dos valores é..: "; soma
PRINT "A média dos valores é..: "; soma/q
END</pre></td></tr>
</table>

<table>
<tr><th colspan="2">Aula 04 – continuação 2</th></tr>
<tr><th>Conteúdos</th><th>Atividades</th></tr>
<tr><td rowspan="2">Variáveis, DO ~ LOOP, PRINT</td><td>Apresentar o seguinte código:
<pre>LET n=0

DO
   LET n=n+1
   PRINT n
LOOP

END</pre></td></tr>
<tr><td>Questionar aos alunos o resultado da execução do código, explicar o que é um laço infinito</td></tr>
<tr><td rowspan="2">Variáveis, DO ~ WHILE LOOP, PRINT</td><td>Apresentar o seguinte código:
<pre>LET n=0

DO WHILE n<100
   LET n=n+1
   PRINT n
LOOP

END</pre></td></tr>
<tr><td>Questionar aos alunos o resultado da execução do código, explicar as diferenças entre DO ~ LOOP e DO ~ WHILE LOOP</td></tr>
<tr><td rowspan="2">Variáveis, DO ~ WHILE LOOP, PRINT, INPUT, IF, ELSE, EXIT DO</td><td>Apresentar o seguinte código:
<pre>LET n=0

DO
   PRINT "Informe um valor: "
   INPUT n

   IF n=0 THEN
      EXIT DO
   ELSE
      PRINT n
   END IF

LOOP

END</pre></td></tr>
<tr><td>Questionar aos alunos o resultado da execução do código, e o que acontece quando o usuário informa o valor 0 para n, associar ao EXIT FOR</td></tr>
</table>

<table>
<tr><th colspan="2">Aula 04 – continuação 3</th></tr>
<tr><th>Conteúdos</th><th>Atividades</th></tr>
<tr><td rowspan="3">Variáveis, DO, PRINT, INPUT, IF, ELSE
EXIT DO, CONTADOR, ACUMULADOR</td><td>Solicitar aos alunos que desenvolvam um programa que leia a idade de várias pessoas, a leitura deverá parar quando a idade digitada for zero. O programa deverá mostrar a soma das idades digitadas; o programa deverá contar quantas idades foram digitadas; o programa deverá mostrar a média das idades digitadas. Os alunos têm 10 minutos para desenvolver a solução para este problema, aquele que terminar primeiro ganha um doce.</td></tr>
<tr><td>Apresentar a solução para o problema proposto:
<pre>LET idade=0
LET soma=0
LET qtd=0

DO
   PRINT "Informe a Idade: "
   INPUT idade

   IF idade=0 THEN
      EXIT DO
   ELSE
      LET soma = soma + idade
      LET qtd = qtd + 1
   END IF

LOOP
PRINT "A média de Idade é...:"; soma/qtd

END</pre></td></tr>
<tr><td>Solicitar aos alunos que alterem o programa anterior para que para que o programa mostre além da média de idade, a menor e maior idade. Os alunos têm 3 minutos para desenvolver a solução para este problema, aquele que terminar primeiro ganha um doce.</td></tr>
</table>

<table>
<tr><th colspan="2">Aula 04 – continuação 4</th></tr>
<tr><th>Conteúdos</th><th>Atividades</th></tr>
<tr><td>Variáveis, DO ~ LOOP, EXIT DO, PRINT, INPUT CONTADOR, ACUMULADOR</td><td>Apresentar a solução para o problema proposto:
<pre>
LET idade=0
LET soma=0
LET qtd=0
LET maior=0
LET menor=0

DO
   PRINT "Informe a Idade: "
   INPUT idade

   IF idade=0 THEN
      EXIT DO
   ELSE
      LET soma = soma + idade
      LET qtd = qtd + 1

      IF (menor>idade) OR (menor=0)THEN
         LET menor = idade
      END IF

      IF (maior<idade) THEN
         LET maior = idade
      END IF

   END IF

LOOP

PRINT "A média de Idade é...:"; soma/qtd
PRINT "A maior Idade é...:"; maior
PRINT "A menor Idade é...:"; menor
END
</pre></td></tr>
<tr><td>Desafio Semanal IV</td><td>Apresentar aos alunos os seguintes códigos:
<pre>
SET WINDOW -10,10, -10,10
DRAW GRID

PLOT LINES: 1,1;1,4

END
</pre>Usando o comando PLOT LINES e as coordenadas do plano cartesiano os alunos devem tentar mostrar na tela um quadrado, um retângulo e um triangulo.</td></tr>
</table>

Quadro 7: Aula 04 com a abordagem proposta

Fonte: Autoria própria

<table>
<tr><th colspan="2">Aula 05</th></tr>
<tr><td colspan="2">Objetivos:
Avaliar os conhecimentos dos alunos referente aos conteúdos apresentados até o momento;
Reforçar os conceitos por meio de atividades práticas os comandos de entrada e saída; Variáveis; Estrutura de repetição; Estrutura de decisão;
Apresentar por meio de atividades práticas o conceito de sub-rotinas e área de plotagem do ambiente;
Implementar problemas numa linguagem de programação.</td></tr>
<tr><th>Conteúdos</th><th>Atividades</th></tr>
<tr><td>Conhecimentos prévios</td><td>Questionar aos alunos quais comandos foram apresentados até o momento e anotar em local visível para os alunos.</td></tr>
<tr><td>Desafio Semanal IV</td><td>Verificar quais alunos conseguiram resolver o Desafio Semanal IV, com base nas diferentes propostas de solução para problema desenvolvidas pelos alunos, apresentar aos demais estudantes e pedir que eles comentem as soluções propostas</td></tr>
<tr><td>Variáveis, Laço FOR, CALL SUB, PLOT LINES</td><td>Apresentar aos alunos os seguintes códigos:<pre>
SET WINDOW -10,10, -10,10
DRAW GRID

CALL teste (0,0,1)
CALL teste (-3, -2, 0.5)
CALL teste (2,2,1)

SUB teste (x,y,raio)
   OPTION ANGLE DEGREES
   FOR t=0 TO 360
      PLOT LINES: x+raio*COS(t),y+raio*SIN(t);
   NEXT t
   PLOT LINES

END SUB
END
</pre></td></tr>
<tr><td rowspan="3">Variáveis, PRINT, INPUT, Laço FOR, IF e ELSE, CALL e SUB, PLOT LINES</td><td>Questionar aos alunos o resultado da execução do código, solicitar que os alunos modifiquem os valores das coordenadas da sub-rotina teste.</td></tr>
<tr><td>Solicitar aos alunos que reescrevam o desafio semanal IV utilizando os comandos CALL e SUB.</td></tr>
<tr><td>Solicitar aos alunos que utilizando os comandos CALL e SUB, tentem desenhar um boneco palito, e uma forca. Os alunos têm 10 minutos para desenvolverem a solução o primeiro que terminar ganha um doce.</td></tr>
</table>

Aula 05 – continuação	
Conteúdos	**Atividades**
Variáveis, PRINT, INPUT, Laço FOR, IF e ELSE, CALL e SUB, PLOT LINES	Solicitar aos alunos que desenvolvam um programa utilizando CALL e SUB com as seguintes características: Escrever um programa que sorteia um número inteiro aleatório X entre 1 e 10 e guarda na variável X; Depois o programa lê um número na variável N seis vezes, em um laço; Cada vez que o usuário tenta acertar o número, mas errar, o programa deverá mostrar na área de plotagem uma parte do boneco palito na forca. Caso o usuário acerte o número o programa, deve emitir a seguinte mensagem “Parabéns que Lindo! ” e encerrar o programa. Dar um bom tempo para que os alunos desenvolvam suas soluções. O primeiro que terminar ganha um doce.
Variáveis, PRINT, INPUT, IF ELSE, FOR, DO ~ WHILE, RANDOMIZE, RND, PLOT LINES, CALL SUB	Solicitar aos alunos que alterem o programa anterior para que seja incluído um menu e mais um jogo o de operações matemáticas, com as seguintes características: O menu deve exibir as seguintes opções 0 para sair, 1 para o jogo da forca, 2 para operações matemáticas e 3 para exibir os créditos do desenvolvedor do programa; O jogo de operações matemáticas deve ter as seguintes características: O programa deve sortear dois números aleatórios N1 e N2 entre 1 e 10 e guarda nas variáveis N1 e N2; O programa deve sortear um número entre 1 e 4 e guardar na variável OP; O programa deve mostrar na tela o valor sorteado para N1 uma das seguintes operações soma, subtração, divisão e multiplicação, e N2 =? Depois o programa deve ler um número a Resposta; Se a resposta estiver correta o programa da os parabéns e acumula 10 pontos; Se a resposta estiver incorreta, o programa informa que a resposta está errada e da nova chance ao usuário diminuindo 10 pontos; A pontuação nunca deverá ser inferior a 0.
	Verificar quais alunos conseguiram resolver o problema, e com base nas diferentes propostas de solução para problema desenvolvidas pelos alunos, apresentar aos demais estudantes e pedir que eles comentem as soluções propostas

Quadro 8: Aula 05 com a abordagem proposta

Fonte: Autoria própria

5 CONCLUSÃO

Programar computadores é uma atividade exigente, que requer certas habilidades e competências para implementar soluções de um determinado problema num ambiente computacional.

As dificuldades envolvidas no processo de ensino/aprendizagem de computadores abrangem os quatro integrantes envolvidos no processo a instituição, o professor, o aluno, e os conteúdos, com diversas variáveis para cada um destes.

A literatura disponibiliza um grande e atual acervo de propostas que visam contribuir com o processo de aprendizagem de programação, ou atenuar as dificuldades existentes no processo. Estas propostas geralmente se encaixam em três principais vertentes ferramentas, estratégias e a combinação de ferramentas e estratégias.

No entanto, as propostas apresentadas por pesquisadores da área não revertem o quadro com a presença das dificuldades sentidas por alunos e professores, tampouco suprem as necessidades das diversas situações de ensino e diferentes contextos, conseguindo obter melhores resultados.

O objetivo desta proposta de abordagem prática, baseada na teoria da aprendizagem significativa, enfatizando/valorizando a interação com a máquina e expondo os estudantes mais cedo ao uso prático do computador para o ensino de programação é conduzir os alunos através das dificuldades apresentadas nas disciplinas de programação.

Para validarmos nossa proposta buscou-se comparar a abordagem proposta a uma abordagem tradicional para o ensino dos fundamentos de programação de computadores, por meio de aplicações no ensino técnico e superior. As aplicações indicam que a exposição dos estudantes mais cedo ao uso prático do computador e a assuntos que normalmente são vistos primeiramente de maneira conceitual na abordagem tradicional não interferem negativamente no desempenho dos estudantes.

Desta forma, cabe nos remetermos a Ausubel, que indica que cada disciplina tem uma estrutura articulada e hierarquicamente organizada de conceitos. No entanto, a ordem em que os principais conceitos e ideias da matéria de ensino são apresentadas muitas vezes não é a mais adequada para facilitar a interação com o conhecimento prévio do aluno.

Isso num primeiro momento nos remete a mais questionamentos do que respostas como por exemplo:

Por que ainda reproduzimos métodos de ensino de programação que remetem ao século passado, onde grande parte da escolas e alunos não tinham acesso as ferramentas disponíveis atualmente?

Por que os conteúdos de programação geralmente ainda são apresentados de maneira que a solução do problema não é associada à sua representação em uma linguagem de programação?

Por que apesar de inúmeras pesquisas, o quadro retratado por Dijkstra em 1989, sobre a cruel realidade de ensinar ciência da computação ainda persiste?

Como ensinar programação eficazmente para os alunos atuais?

Dos resultados que a abordagem proposta apresenta, destacamos a indicação para mudanças na estrutura articulada e hierarquicamente organizada de conceitos no ensino de programação, ou seja, em nossa abordagem os conteúdos foram apresentados inter-relacionados e de maneira cíclica onde o estudante pode ganhar maturidade para assimilar, o que geralmente não ocorre na abordagem tradicional.

Deste modo, a situação proposta pela abordagem possibilita aos alunos verem os conteúdos mais vezes, o que pode contribuir para a aquisição de experiência em programação.

Além disso, percebeu-se indícios de que a aplicação prática contribuiu para diminuir o grau de abstração presente na matéria de programação, favorecendo entre outros aspectos a motivação dos estudantes.

REFERÊNCIAS

Ambrósio, A. P. L., Almeida, L. S., Macedo, J., Santos, A., & Franco, A. H. (2011). Programação de Computadores: Compreender as dificuldades de aprendizagem dos alunos. *Revista Galego-Portuguesa de Psicoloxía E Educación*, *19*(1, ano 16), 185–197. Retrieved from http://repositorium.sdum.uminho.pt/handle/1822/15554

Ausubel, D. (1963). The psychology of meaningful verbal learning. Retrieved from http://psycnet.apa.org/psycinfo/1964-10399-000

Ausubel, D. (2003). Aquisição e retenção de conhecimentos: uma perspectiva cognitiva. *Lisboa: Plátano*. Retrieved from http://files.mestrado-em-ensino-de-ciencias.webnode.com/200000007-610f46208a/ausebel.pdf

Ausubel, D., Novak, J., & Hanesian, H. (1968). Educational psychology: A cognitive view. Retrieved from http://www.spbkbd.com/english/art_english/art_51_030211.pdf

AusubeL, D., Novak, J., & Hanesian, H. (1980). Psicologia educacional. Retrieved from https://scholar.google.com.br/scholar?q=Psicologia+educacional&btnG=&hl=pt-BR&as_sdt=0%2C5#3

Baranauskas, M. C. C. (1993). Procedimento, função, objeto ou lógica? linguagens de programação vistas pelos seus paradigmas. *Computadores E Conhecimento:*, 1–15. Retrieved from http://www.nied.unicamp.br/publicacoes/arquivos/3XaQ1o8Nmk\nhttp://disciplinas.dcc.ufba.br/pub/MATA56/Exercicios/(Leitura_e_Resenha)_ArtigoDiscussaoParadigmas.pdf

Braathen, P. (2003). O processo ensino aprendizagem em disciplinas básicas do terceiro grau. *Educação & Tecnologia*. Retrieved from http://www.revista.cefetmg.br/index.php/revista-et/article/view/53

Brasil, M. (2001). Diretrizes Curriculares de Cursos da área de Computação e Informática. *… Especialistas de Ensino de Computação E Informática. …*, 1–23. Retrieved from http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:DIRETRIZES+CURRICULARES+DE+CURSOS+DA+ÁREA+DE+COMPUTAÇÃO+E+INFORMÁTICA#4

Brusilovsky, P., Kouchnirenko, a., Miller, P., & Tomek, I. (1994). Teaching Programming to Novices: A Review of Approaches and Tools. *Proceedings of ED-MEDIA 94--World Conference on Educational Multimedia and Hypermedia*, 103–110. Retrieved from http://www.computer.org/portal/web/csdl/doi/10.1109/FIE.1999.839268

Delgado, C., XEXEO, J. a. M., Souza, I. F., Campos, M., & Rapkiewicz, C. E. (2004). Uma Abordagem Pedagógica para a Iniciação ao Estudo de Algoritmos. *XII Workshop de …*. Retrieved from http://www.lbd.dcc.ufmg.br/colecoes/wei/2004/0024.pdf

Dijkstra, E. W. (1988). On the cruelty of really teaching computing science. *Unpublished Manuscript EWD*. Retrieved from http://www.smaldone.com.ar/documentos/ewd/EWD1036_pretty.pdf

Dunican, E. (2002). Making The Analogy : Alternative Delivery Techniques for First Year Programming Courses. *Technology*, (June). Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.115.8440&rep=rep1&type=pdf

Gomes, A., Henriques, J., & Mendes, A. J. (2008). Uma proposta para ajudar alunos com dificuldades na aprendizagem inicial de programação de computadores. *Educação, Formação & Tecnologias*, *1*(1), 93–103, Maio. Retrieved from http://www.eft.educom.pt/index.php/eft/article/view/23

Gries, D. (1974). What should we teach in an introductory programming course? *ACM SIGCSE Bulletin*, *6*(1), 81–89. http://doi.org/10.1145/953057.810447

Jenkins, T. (2002). On the Difficulty of Learning to Program. *Proceedings of the 3rd Annual Conference of the LTSN …*, *4*, 53–58. Retrieved from http://78.158.56.101/archive/ics/events/conf2002/tjenkins.pdf

MASINI, E., & Moreira, M. (2001). Aprendizagem significativa: a teoria de David Ausubel. *São Paulo: Centauro*. Retrieved from https://scholar.google.com.br/scholar?q=Aprendizagem+Significativa+-+A+Teoria+de+David+Ausubel.&btnG=&hl=pt-BR&as_sdt=0%2C5#0

Moreira, M. (1983). Uma abordagem cognitivista ao ensino da Física. Retrieved from https://scholar.google.com.br/scholar?q=Uma+abordagem+cognitivista+ao+ensino+da+F%C3%ADsica&btnG=&hl=pt-BR&as_sdt=0%2C5#0

Moreira, M. (1999). Teorias de aprendizagem. Retrieved from https://scholar.google.com.br/scholar?q=Aprendizagem+significativa+moreira+1999&btnG=&hl=pt-BR&as_sdt=0%2C5#0

Moreira, M., & Sousa, C. (1996). Organizadores prévios como recurso didático. *Porto Alegre, RS, Instituto de Física Da UFRGS, …*. Retrieved from https://scholar.google.com.br/scholar?q=Organizadores+pr%C3%A9vios+como+recurso+did%C3%A1tico.&btnG=&hl=pt-BR&as_sdt=0%2C5#0

Novak, J., Rabaça, A., & Valadares, J. (2000). Aprender criar e utilizar o conhecimento: Mapas conceptuais TM como ferramentas de facilitação nas escolas e empresas. Retrieved from https://scholar.google.com.br/scholar?q=Aprender%2C+criar+e+utilizar+o+conhecimento.+Mapas+conceituais+como+ferramentas+de+facilita%C3%A7%C3%A3o+nas+escolas+e+empresas&btnG=&hl=pt-BR&as_sdt=0%2C5#0

Pelizzari, A., & Kriegl, M. (2002). Teoria da aprendizagem significativa segundo Ausubel. *Revista …*. Retrieved from http://files.gpecea-usp.webnode.com.br/200000393-74efd75e9b/MEQII-2013- TEXTOS COMPLEMENTARES- AULA 5.pdf

Pereira Júnior, J. C. R., & Rapkiewicz, C. E. (2004). O Processo de Ensino-Aprendizagem de Fundamentos de Programação : Uma Visão Crítica da Pesquisa no Brasil. *Anais Do XII Workshop Sobre Educação Em Computação (SBC)*. Retrieved from http://www.lbd.dcc.ufmg.br/colecoes/weirjes/2004/003.pdf

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and Teaching Programming : A Review and Discussion. *Computer Science Education*, *13*(2), 137–172. http://doi.org/10.1076/csed.13.2.137.14200

Santos, R. P., & Costa, H. A. X. (2005). TBC-AED e TBC-AED/Web: Um desafio no ensino de algoritmos, estruturas de dados e programação. *IV Workshop de Educação Em Computação E Informática Do Estado de Minas Gerais (WEIMIG' 2005). Varginha, MG, Brasil*. Retrieved from http://www.cos.ufrj.br/~rps/pub/completos/2005/WEIMIG.pdf

Sheard, J., Simon, S., Hamilton, M., & Lönnberg, J. (2009). Analysis of research into the teaching and learning of programming. *Proceedings of the Fifth International Workshop on Computing Education Research Workshop - ICER '09*, 93. http://doi.org/10.1145/1584322.1584334

Sleeman, D. (1986). The challenges of teaching computer programming. *Communications of the ACM*, *29*(9), 840–841. http://doi.org/10.1145/6592.214913

Soloway, E., K., E., Bonar, J., & Greenspan, J. (1983). What do novices know about programming ? *In B. Shneiderman & A. Badre (Eds.), Directions in Human-Computer Interactions, Norwood, NJ: Ablex.*, 27–54. Retrieved from https://scholar.google.com.br/scholar?q=What+do+novices+know+about+programming%3F&btnG=&hl=pt-BR&as_sdt=0%2C5#0

ANEXO A - Sugestão de Avaliações para os conteúdos, referentes aos cinco planos de aula desta proposta de abordagem para o Ensino de Programação de Computadores.

AVALIAÇÃO 1

ALUNO:

DATA:

DISCIPLINA:

PROFESSOR:

CURSO:

CONCEITO / NOTA

Orientações:

- *Avaliação Prática individual.*
- *Crie uma pasta na área de trabalho com seu nome;*
- *Salve cada exercício conforme o modelo ExQ1, ExQ2...*
- *Ao término da avaliação chame o professor.*

1. Escreva um programa que imprima na tela 300 vezes a seguinte frase **"ISHIIIII VOU TER DE APRENDER A PROGRAMAR"**

2. Escreva um programa que leia do teclado dois números e guarde nas variáveis **A** e **B**, troque os valores (**A deve receber B e B deve receber A)**, e imprima na tela os valores de **A** e **B.**

3. Escreva um programa que leia do teclado dois números **X** e **Y**, e realize a soma do intervalo informado pelo usuário, e imprima na tela o valor da soma.

4. Escreva um programa que leia do teclado três valores (a, b, c) e imprima na tela estes valores em ordem crescente.

5. Escreva um programa que leia do teclado a idade de dez pessoas e ao término imprima:

   a. Valor da idade do mais Jovem.
   b. Valor da idade do mais velho

AVALIAÇÃO 2

ALUNO:

DATA:

DISCIPLINA:

PROFESSOR:

CURSO:

CONCEITO/NOTA

Orientações:

- *Avaliação Prática individual.*
- *Crie uma pasta na área de trabalho com seu nome;*
- *Salve cada exercício conforme o modelo ExQ1, ExQ2...*
- *Ao término da avaliação chame o professor.*

1. Escreva um programa que peça dois números inteiros e imprima na tela a soma desses dois números.

2. Escreva um programa que leia um valor em metros e imprima na tela o valor convertido em centímetros.

3. Escreva um programa que leia a quantidade de dias, horas, minutos e segundos do usuário, e calcule e imprima na tela o total em segundos.

4. Escreva um programa que calcule o tempo de uma viagem de carro. O programa deve ler a distância a percorrer e a velocidade média esperada para a viagem. O programa deve imprimir na tela o tempo estimado para viagem.

5.

URI Online Judge | 1142

PUM

Adaptado por Neilor Tonin, URI Brasil

Timelimit: 1

Escreva um programa que leia um valor inteiro N. Este N é a quantidade de linhas de saída que serão apresentadas na execução do programa.

Entrada

O arquivo de entrada contém um número inteiro positivo N.

Saída

Imprima a saída conforme o exemplo fornecido.

Exemplo de Entrada	Exemplo de Saída
7	1 2 3 PUM 5 6 7 PUM 9 10 11 PUM 13 14 15 PUM 17 18 19 PUM 21 22 23 PUM 25 26 27 PUM