UM ESTUDO SOBRE O ENSINO-APRENDIZAGEM DE LÓGICA DE PROGRAMAÇÃO

A STUDY ON THE TEACHING-LEARNING OF LOGIC PROGRAMMING

Henryethe Valentim
Orientador: Prof. Dr. André Koscianski

Universidade Tecnológica Federal do Paraná /Programa de Pós-Graduação em Educação Científica e Tecnológica/ henryethe@gmail.com

Resumo

Este estudo foi feito com o intuito de analisar alguns porquês da evasão e repetência das disciplinas introdutórias do ensino de programação. Partindo do histórico das linguagens de programação e a relação existente entre a linguagem e os seus paradigmas. Faz uma breve reflexão sobre as teorias de ensino-aprendizagem e o ensino de lógica de programação. É realizada também uma análise críticas sobre algumas ferramentas utilizadas no ensino de algoritmos, discutindo-se a necessidade de uma metodologia adequada para o ensino destas disciplinas. Deste modo é descrita a proposta de trabalho de um professor para a introdução da disciplina de lógica de programação em um curso de informática em nível subseqüente.

Palavras-chave: Lógica de programação. Metodologia. Ensino-aprendizagem. Algoritmo.

Abstract

This study was done in order to analyze why some of the repetition and dropout of the subjects of teaching introductory programming. On the history of programming languages and the relationship between language and its paradigms. A brief reflection on the theories of teaching and learning and teaching of logic programming. It also performed an analysis on some critical tools used in teaching algorithms, discussing the need for appropriate methodology for teaching these subjects. Thus is described the proposed work of a teacher to introduce the discipline of logic programming in a computer course in subsequent levels.

Keywords: Logic Programming. Methodology. Teaching and learning. Algorithm.

INTRODUÇÃO

O ensino da tecnologia envolve questões relacionadas ao pensamento sistematizado, o qual tem a sua base no ensino das ciências. Ciência e tecnologia são aliadas no processo ensino-aprendizagem. O ensino da matemática pode utilizar da tecnologia de jogos eletrônicos para despertar, memorização e raciocínio rápido, como em jogos que propõem problemas para serem resolvidos, através da digitação de valores, ou até mesmo através de tiros em naves que apresentem a solução correta. A química pode utilizar a tecnologia para demonstrar uma reação, ou a biologia para visualizar uma bactéria; estes dois exemplos exibem apenas algumas das possibilidades do uso da tecnologia como auxiliar no ensino-aprendizagem das ciências.

Tendo em vista o exposto, uma questão que se coloca seria como unir os recursos da informática com o ensino de tecnologia.

Neste trabalho trataremos mais especificamente do ensino de lógica de programação, uma disciplina integrante da grade curricular dos cursos que envolvem programação de computadores nos níveis técnico e superior. Faz parte desta reflexão analisar as metodologias seguidas pelos professores da disciplina, que possam proporcionar melhor aproveitamento no ensino-aprendizagem.

Este trabalho tem por objetivo refletir sobre estas questões enquanto descreve a forma de abordagem da disciplina de lógica de programação pelo professor de um curso técnico em informática.

1 HISTÓRICO

O computador de programa armazenado de Von Neumann criado na final dos anos 40 trouxe à necessidade da escrita de programas que fizessem com que os computadores executassem determinadas tarefas. Na história da computação as linguagens foram agrupadas em gerações.

Na década de 50 surgem às linguagens de 1ª Geração ou linguagens de máquina; encontram-se no mais baixo nível de abstração de programação. A 2ª Geração das linguagens de computador aconteceu entre os anos de 1960 e 1974, com o surgimento de linguagens com grande aceitação no mercado, as quais introduziram novas características como a programação multiusuário, sistemas de execução em tempo real e desenvolvimento de gerenciadores de base de dados. São exemplos de linguagem desta geração Fortran, Cobol, Algol e Basic. As linguagens de 3ª Geração são representadas pelas linguagens que surgiram entre 1974 e 1986 e tinham como característica a possibilidade de criar sistemas distribuídos, incorporar recursos mais inteligentes, e exigir um hardware menos robusto.

As linguagens de programação podem ser divididas em duas categorias: linguagens de propósito geral e especializadas. As linguagens de propósito geral baseiam-se na linguagem Algol e podem ser utilizadas em aplicações científicas e comerciais, exemplos: Pascal, PL/1, C e Modula-2. Linguagens especializadas se caracterizam por utilizarem uma forma sintática não usual e serem desenvolvidas para aplicações específicas, exemplos: LISP, Prolog, Smaltallk, APL e Forth. As linguagens de 4ª geração surgiram a partir de 1986, são também conhecidas como linguagens artificiais, combinam características procedurais e não-procedurais, possuem um alto grau de abstração. São classificadas em linguagens de consulta, geradoras de programas e outras linguagens (4GL).

2 EXEMPLO

Para ilustrar as etapas que um aluno deve seguir ao produzir um programa de computador, descreveremos um exemplo: Suponha o cálculo do fatorial de um número N inteiro. O cálculo do fatorial de um número N é obtido através da multiplicação de N pelos seus antecessores até se chegar ao número 1. Sendo assim, o fatorial de 5 é obtido por 5 x 4 x 3 x 2 x 1.

Observa-se que há uma ação que se repete dentro do procedimento de cálculo: a multiplicação é realizada várias vezes. Para solucionarmos esta questão será empregado um laço. Na programação de computadores toda vez que precisamos repetir N vezes uma determinada seqüência de comandos, utilizamos uma laço ou estrutura de repetição. O laço possibilita ao programador repetir uma parte do programa quantas vezes forem necessárias. Exemplo: exiba na tela os números inteiros de 1 a 100:

```
Para N:= 1 até 100 faça
    Escreva (N)
```

Voltando ao exemplo do cálculo do fatorial de 5, uma forma de solução é apresentada na figura 1:

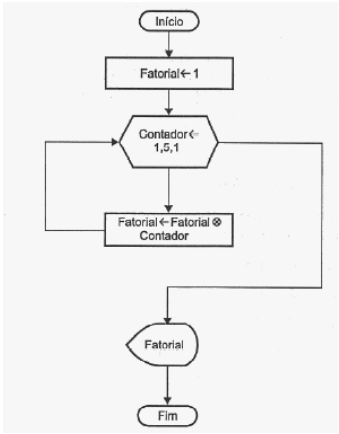

Figura 1 – Fluxograma: cálculo do fatorial de 5

O desenho apresentado na figura 1 é chamado de fluxograma e é usado para mostrar de forma gráfica, a lógica utilizada para solucionar um determinado problema, o mesmo enfatiza passos individuais e o fluxo de execução. Entretanto, computadores não são programados usando desenhos. Existem algumas linguagens gráficas como a "G" do LabVIEW[1], mas são raras e pouco utilizadas.

No ensino de programação são usados textos que descrevem o processo de solução de um determinado problema, na figura 2 é exibido o algoritmo do cálculo do fatorial:

```
    fat, i, n: INTEIRO;
ESCREVA ("DIGITE UM NÚMERO: ");
LEIA (n);
Fat := n;
PARA i DE 1 ATÉ (n – 1)  FAÇA
      fat := fat * i;
FIM_PARA;
ESCREVA (fat);
```

Figura 2 – Algoritmo cálculo do fatorial de N.

```
Program FATORIAL;
var
   I, N, FAT: integer;
Begin
      Writeln('Digite um número:');
      readln (N);
      FAT := N;
      For i:=1 to  (N-1)  do
          FAT := FAT * I;
       writeln('O fatorial de  ', N, ' equivale a:  ', FAT);
end;
```

Figura 3 – Código na Linguagem Pascal fatorial de N.

Na figura 3 pode-se observar que no programa escrito em Pascal há várias palavras adicionais, tais como "**program**", "**var**", "**begin**", "**end**". Elas são uma exigência do compilador, que nada mais é do que um programa ou conjunto de programas que tem por

[1] **LabVIEW** (acrónimo para *Laboratory Virtual Instrument Engineering Workbench*) é uma linguagem de programação gráfica originária da National Instruments. A primeira versão surgiu em 1986 para o Macintosh e atualmente existem também ambientes de desenvolvimento integrados para os Sistemas Operacionais Windows, Linux e Solaris.

objetivo traduzir um programa escrito em uma linguagem de alto nível (figura 3), o programa-fonte, em um programa expresso em uma linguagem de baixo nível (linguagem que somente o computador entende - binário) o programa-objeto. Esta tradução envolve a análise sintática, a qual tem por objetivo verificar se o programa está escrito dentro das regras da linguagem de programação do compilador, por exemplo, na Linguagem Pascal todo comando deve terminar com um ponto e vírgula, caso contrário o compilador deverá emitir uma mensagem de erro. Após, realizada esta análise, o compilador partirá para a análise semântica, que envolve a verificação de contexto, por exemplo, em Pascal não podemos somar o número 3 com o valor booleano true, ou usar uma variável X que não esteja declarada, somente após a passagem pelas 2 verificações o programa será convertido para a linguagem de máquina, e poderá ser executado pelo computador.

Tendo em vista as exigências que envolvem a compilação de um programa, o aprendizado da linguagem torna-se difícil, devido à sobrecarga de sintaxe da linguagem que deve ser assimilada pelo aluno fator este que precisa ser levado em consideração quando se fala sobre o ensino de programação.

Outro ponto importante a ser analisado no ensino de programação é que existem infinitas formas para solucionar um determinado problema. Isto pode ser verificado na figura 4, que apresenta a resolução do cálculo do fatorial com o laço enquanto ... faça.

```
    fat, i, num : INTEIRO;
ESCREVA (“DIGITE UM NÚMERO: ”);
LEIA (num);
i:= 1;
fat := num;
ENQUANTO  i < num FAÇA
    fat := fat * i;
    i:= i+1;
FIM_ENQUANTO;
ESCREVA (fat);
```

Figura 4 – Algoritmo com o laço enquanto ... faça do Fatorial de N

Como a construção de algoritmos envolve raciocínio, e pessoas entendem um mesmo problema de forma diferente, os dois exemplos acima apresentam duas maneiras de resolver um cálculo simples, como o do fatorial. Levando-se isso em consideração os professores desta disciplina precisam dar conta de questões de ensino (teoria), avaliativas (correções) e aprendizagem onde teoria e prática necessitam unir-se na compilação de um programa.

3 LINGUAGENS DE PROGRAMAÇÃO E PARADIGMAS

O ensino de lógica de programação está ligado ao conceito de algoritmos coerentes e válidos. Algoritmos são seqüencias lógicas para resolver problemas. Na computação a linguagem de programação é utilizada para traduzir um algoritmo de deforma que o computador possa executá-lo. Assim como o pensamento humano pode ser expresso em diversas línguas, o algoritmo pode ser escrito em várias linguagens de programação.

A linguagem de programação pode ser definida como um conjunto de regras e comandos com sintaxe e semântica próprias, utilizada pelo programador para traduzir os algoritmos, e tem como objetivo enviar comandos para serem executados pelo processador e memória do computador. A proximidade com a linguagem humana define o nível da linguagem de programação

As linguagens de programação são classificadas em linguagens de baixo nível e de alto nível. Linguagens de baixo nível se caracterizam por se aproximarem muito das instruções usadas pela CPU. As instruções de máquina são representadas por bits na memória.

A proximidade com a máquina torna complicado programar usando apenas as linguagens de baixo nível.

As linguagens de alto nível caracterizam-se por serem linguagens mais próximas do homem e com elas são desenvolvidas os softwares comerciais. São linguagens de alto nível: BASIC, COBOL, PASCAL, C.

Quando se está ensinando alguém a programar, tem-se como objetivo ensinar uma linguagem de alto nível. Um exemplo de soma entre dois números e multiplicação entre dois números na Linguagem COBOL:

```
ADD Num1, Num2 GIVING Result

MULTIPLY Num1 BY Num2 GIVING Result
```

Além da classificação apresentada acima, as linguagens de programação apresentam diferenças quanto à sintaxe (vocabulário - "forma de escrita") e a semântica (que refere-se ao significado de cada elemento da linguagem). Porém a diferença mais significativa entre as linguagens de programação ocorre com relação a seus paradigmas.

Os paradigmas determinam como o uso da linguagem determinará a resolução de problemas. A literatura nos oferece diversas definições de paradigma de programação aqui usaremos a definição de Papert (1985):

> um paradigma de programação é como um "quadro estruturador" subjacente à atividade de programar e a escolha do paradigma de programação pode mudar notavelmente *"a maneira como o programador pensa sobre a tarefa de programar"*.

A atividade de programar no paradigma Imperativo (Procedural) significa propor tarefas seqüenciais ao computador de modo a solucionar determinado problema. Este modelo representa o estilo de programação estruturada onde os programas são decompostos, modularizados em rotinas (procedimentos) que serão executados seqüencialmente pelo computador. São linguagens Imperativas: FORTRAN, COBOL, ALGOL 60, APL, BASIC, PL/I, SIMULA 67, ALGOL 68, PASCAL, C etc.

Programar no Paradigma Funcional é escrever funções, e verificar como as mesmas reagem após serem processadas pela máquina. Neste paradigma o resultado de uma função será o mesmo para um dado conjunto de valores não importando onde, ou quando, seja executa. Este paradigma é utilizado pela Inteligência Artificial. Linguagens Funcionais: LISP, Haskell, Scheme ML, etc.

O paradigma Orientado a Objetos baseia-se no fato de que programas simulam o mundo real, o qual é composto por objetos. Na programação Orientado a Objetos os objetos trocam mensagens entre si. A primeira linguagem Orientada a Objeto foi SIMULA, desenvolvida em 1966 e depois refinada em SMALLTALK. Geralmente as linguagens que se utilizam do paradigma Orientado a Objetos como: C++, JAVA, Delphi e Visual Basic, são apresentadas para os alunos após eles terem assimilado os conceitos do paradigma Procedural.

As linguagens Ada, Linda ou Occam, representam o paradigma Concorrente, o qual é pouco utilizado em sistemas de alto nível e em ambientes comerciais.

O paradigma Lógico se utiliza da lógica matemática na programação, John MacCarthy foi o precursor deste paradigma, o qual tem por base um conjunto de fórmulas em um fragmento da lógica de primeira ordem, herdando o modelo dessa teoria. Exemplos de linguagens lógicas: Planner, Prolog, QA-4, Popler, Mercury, Visual Prolog, Oz etc.

Faz-se necessário uma análise criteriosa dos paradigmas de programação, tendo em vista a sua adequação ao meio educacional. Embora a marca do primeiro paradigma aprendido pelo iniciante em programação seja algo a ser também transposto por ele no

decorrer da aprendizagem de outras linguagens com outros paradigmas, no ensino técnico subseqüente, a linguagem Pascal foi a escolhida.

A linguagem Pascal utiliza o Paradigma Procedural, o qual está mais próximo dos conteúdos estruturantes da disciplina de lógica de programação, esta foi à linguagem escolhida para iniciar o processo de ensino-aprendizagem dos alunos. Algumas características desta linguagem corroboram a sua escolha dentre elas destaca-se o fato de ser uma linguagem tipada e altamente estruturada, a qual oferece um ambiente amigável aos alunos iniciantes, proporcionado uma boa relação entre teoria e prática.

4 TEORIAS DE ENSINO-APRENDIZAGEM

A aprendizagem é um processo fundamental da vida. O homem aprende e através desta aprendizagem desenvolve comportamentos que o permitem viver. A condução do processo de ensino requer a compreensão do processo de aprendizagem, neste sentido desde a antiguidade, filósofos e pensadores estudam a aprendizagem e os fenômenos que a envolvem.

Para o professor compreender como ocorre o processo ensino-aprendizagem, é fundamental. Este entendimento dará a ele suporte para a criação de novos meios (exercícios, formas de apresentar o conteúdo, como avaliar) que trarão melhorias a sua prática diária. Sócrates defendia a idéia de que o conhecimento preexiste no espírito do homem e a aprendizagem consistia no despertar desses conhecimentos inatos e adormecidos. Para Platão em sua teoria dualista o corpo (as coisas) era separado da alma (ou idéias), definia aprendizagem como uma reminiscência. Aristóteles com um ponto de vista mais científico rejeita a preexistência das idéias em nosso espírito, desenvolveu a idéia do ensino intuitivo e utilizou os métodos dedutivo e indutivo e aplicou-os em suas observações, experiências e hipóteses. Santo Agostinho adotou o método indutivo, e se utilizou da introspecção para realizar suas experiências mentais. São Tomás de Aquino definia a aprendizagem como processo inteligente, dinâmico, cujo seu principal agente é a atividade de quem aprende. Apesar dos esforços dos filósofos até a Idade Média a ênfase na educação permaneceu teológica e teórica. Com o advento da ciência moderna, Bacon, Descartes e Locke disseminaram a fé no conhecimento, baseada no senso-percepção e no raciocínio lógico, neste contexto surgiram às modernas teorias de ensino-aprendizagem.

As teorias da de ensino-aprendizagem são numerosas e divergentes, devido a isso usaremos a abordagem de Campos em seu livro Psicologia da Aprendizagem.

Edward Lee Thorndike (1874-1949) foi o organizador da Psicologia Educacional, esta teoria recebeu diferentes nomes durante a história, alguns são: associonistas modernos, funcionalistas modernos, psicólogos de estímulo-e-resposta, entre outros. O conceito básico desta teoria é a conexão que "[...] resulta da associação entre as impressões dos sentidos e os impulsos para a ação, isto é, associação entre a verdadeira situação (S) e resposta (R)". (CAMPOS, p.168-169, 1979)

Na teoria conexionista a aprendizagem baseia-se na Lei do Exercício, onde a conexão diminui quando não é exercitada. Alguns fatores afetam o exercício, a intensidade do estímulo, que se refere à atenção ao se executar um exercício, quanto mais recente o estímulo mais fácil será sua execução, sendo assim a aprendizagem é definida como um processo criador que não exige de quem aprende a compreensão do que gerou o conhecimento a ser assimilado, levando apenas em consideração a formação de ligações entre os neurônios.

Thorndike diz que a aprendizagem é o resultado da seleção de reações já existentes no sistema nervoso, através de ensaio e erro, e do acerto acidental nas quais o prazer do acerto gera conexões nervosas e a insatisfação do erro elimina-as.

Ivan Petrovitch Pavlov (1849-1939) foi o autor teoria do condicionamento reflexológico ou condicionamento clássico. Utiliza a abordagem conexionista que trata do

estímulo e reação. Esta teoria não tem por objetivo explicar a aprendizagem, mas sim relatar as condições e os fatores fundamentais na formação de hábitos. Esta teoria é própria para explicar condições de aprendizagem que envolvem reações emocionais, falhando na explicação da aprendizagem intelectual, que envolve o pensamento e o raciocínio lógico.

A teoria de Skinner baseia-se no conexionismo de Thorndike e no behaviorismo de Watson, sendo seu princípio norteador o estímulo-resposta, que "[...] significa que o comportamento opera no ambiente, para gerar conseqüências". (CAMPOS, p.190, 1979)

Na teoria do Condicionamento Operante a aprendizagem é definida como uma mudança na probabilidade de resposta, ou seja, o reforço fortalecido para gerar uma determinada resposta, aumenta a certeza de se alcançar determinado objetivo.

Os criadores da teoria Clássica de Gestalt são Wetheimer (1890-1943) e seus discípulos Köhler e Koffka. A base das teorias gestaltistas ou de campo é a percepção. Nesta teoria a aprendizagem não se dá através do estímulo-resposta como nas teorias conexionistas, sendo a resposta um sinal da aprendizagem não fazendo parte do processo. Na visão da gestalt o todo é maior do que a soma de suas partes, ou seja, o que envolve o fenômeno aprendizagem não pode ser aprendido apenas com estímulos-respostas particulares. O seu foco está no insight e a sua relação com a formação de conceitos. Esta teoria propõe uma aprendizagem significativa, que necessita de aquisições de estruturas e formas, com discernimento e compreensão da situação.

A educação do século XXI tem como objetivo primeiro o aprender a aprender, vivemos em um tempo de aceleração do ritmo das coisas, a sociedade apresenta mudanças em todas as áreas: política, tecnológica, social, cultural, econômica. Aprender neste contexto "requer assimilar informação, ter conhecimentos, realizar operações, exercitar procedimentos e estratégias para tirar o melhor partido do que se conhece, conhecer mais, resolver problemas, tomar decisões". (GARCIA & VEIGA, p.92, 2007)

As teorias de ensino-aprendizagem buscam compreender como se dá a aprendizagem, o construtivismo de Vygotsky e de seus seguidores - Luria, por exemplo -, ao lado de outras teorias como o interacionismo de Piaget, alteraram profundamente o modo de ver todo o processo do ensino e aprendizagem (POZO, 2002).

O professor como ator do processo ensino-aprendizagem precisa estar ciente de seu papel, e é necessário que ele possua a percepção do que o aluno não consegue assimilar. Mais do que ensinar é necessário refletir sobre sua prática e refazer suas metodologias para que possa atingir o seu objetivo maior, que é o de fazer com que o aluno aprenda.

5 FERRAMENTAS NO ENSINO DE PROGRAMAÇÃO

A revisão de literatura realizada mostrou uma grande quantidade de trabalhos que propõem ferramentas, isto é: programas que possam auxiliar o ensino da disciplina. Foi encontrada uma proporção muito menor de material que se incline sobre metodologias de trabalho.

Uma das primeiras ferramentas auxiliares no ensino-aprendizagem de algoritmos foi BALSA, desenvolvida na Brown University (BROWN, 1987) e tinha como objetivo servir como um laboratório de testes dinâmicos para algoritmos. Outro exemplo foi ASTRAL, desenvolvida em 1996 pela UNICAMP; trabalhava com o conceito de animar a execução de um algoritmo.

Santos & Costa (2005) descrevem as ferramentas EDDL (Estruturas de Dados Dinâmicas Lineares), TED (Tutorial de Estruturas de Dados) e uma ferramenta que possibilita ensino a distância, a TBC-EAD/WEB. Algumas características importantes apontadas são as seguintes:

> i) *links* explicativos, evitando a necessidade de aprendizagem via tutorial; ii) a usabilidade da interface gráfica é razoável, possibilitando ao professor apresentar conceitos iniciais, conteúdos teóricos e práticos aos poucos, como apresentaria em transparências; iii) conteúdo teórico simples, de forma a familiarizar melhor o aluno com o assunto; iv) processo gráfico passo a passo, com elementos numéricos, o que melhora a visualização e o entendimento; e v) legendas explicativas, que ilustram as etapas do processo de apresentação de algoritmos. (SANTOS & COSTA, 2005)

Outros exemplos recentes de ferramentas apontadas na literatura são Cooper et al. (2003) e Maloney et al. (2008), ambos enfatizando visualização e com um forte apelo lúdico.

Em geral as ferramentas de ensino-aprendizagem de programação pretendem propiciar uma aprendizagem fácil e intuitiva, mas não existe uma unanimidade com relação à seu uso. Elas podem ser consideradas uma alternativa mas não metodologias em si.

Os autores que se interessam ao aspecto metodológico geralmente enfatizam dois aspectos: o uso de heurísticas, como em Wirth (1971) e Polya (2004); e o desenvolvimento da habilidade de resolver problemas, como em Papert (1985) e Falker e Palmer (2009).

6 DESCRIÇÃO

Aqui descreveremos em linhas gerais a disciplina de Lógica de Programação do Curso Técnico em Informática de um Colégio Estadual, no período noturno, na cidade de Ponta Grossa no Estado do Paraná, no nível subseqüente. Como este Curso é ofertado pela rede pública de ensino, não existem critérios no momento da inscrição, a única exigência é que o aluno já tenha terminado o Ensino Médio. Isto trás para a sala de aula uma diversidade de alunos com características próprias, alguns vindos do EJA, outros há muito tempo afastados dos bancos escolares, e ainda aqueles que acabaram de sair do Ensino Médio.

Lindeman (1926) identificou cinco pressupostos-chave para a educação de adultos os quais fazem parte dos fundamentos da moderna teoria de aprendizagem de adultos (Andragogia[2]):

> 1. Adultos são motivados a aprender à medida que experimentam que suas necessidades e interesses serão satisfeitos.
> 2. A orientação de aprendizagem do adulto está centrada na vida; [...] para se organizar seu programa de aprendizagem são as situações de vida e não disciplinas.
> 3. A experiência é a mais rica fonte para o adulto aprender.
> 4. Adultos têm uma profunda necessidade de serem auto-dirigidos; [...]
> 5. As diferenças individuais crescem com a idade; por isto, a educação de adultos deve considerar as diferenças de estilo, tempo, lugar e ritmo de aprendizagem.

Emenda da disciplina de Lógica de Programação:

Conteúdos:

- Etapas para a resolução de um problema via computador;
- Conceitos básicos;
- Seqüência lógica;
- Conceitos de tipos de dados e instruções primitivas;
- Operadores;
- Representações (Diagrama de bloco, Narração descritiva e Linguagem algorítmica);
- Pseudocódigo;
- Regras para a construção de algoritmos;

---

[2] **Andragogia** é a arte ou ciência de orientar adultos a aprender, segunda a definição creditada a Malcolm Knowles, na década de 1970. O termo remete a um conceito de educação voltada para o adulto, em contraposição à pedagogia, que se refere à educação de crianças (do grego *paidós*, criança). Fonte: http://pt.wikipedia.org/wiki/Andragogia

- Comandos de entrada e saída;
- Estrutura Seqüencial;
- Estrutura de Desvio;
- Estrutura de Repetição;
- Teste de mesa;
- Conjuntos Homogêneos:
    - Vetores;
    - Matrizes;
- Implementação de algoritmos;
- Arquivos de dados:
    - Conceitos;
    - Abertura/Leitura/Fechamento de arquivos;
    - Movimentação de registros;
    - Gravação de arquivos.

Na matriz curricular do Curso Técnico em Informática, a disciplina de Lógica de Programação localiza-se no 1º semestre com uma carga horária de 80 horas, divida em 4 aulas semanais. Um problema identificado tanto pelos alunos como pelo professor, é que a carga horária não é compatível com a ementa extremamente extensa, os alunos sentem a necessidade de mais aulas para praticar o que aprenderam.

Como esta disciplina serve de base para a de Linguagem de Programação, o paradigma estruturada foi o escolhido pelo professor que optou pela linguagem Pascal (PZIM), por ser mais simples e ser fortemente estrutura, tendo em vista que a carga horária é reduzida.

Como a ferramenta da disciplina de Linguagem de Programação é o Delphi 7.0, a disciplina de Lógica de Programação trabalha com a linguagem Pascal, esta escolha foi feita para facilitar o processo, mesmo que exista mudança de paradigma, do Pascal para o Object Pascal, a sintaxe e a semântica não apresentam diferenças significativas, o que neste caso favorece a evolução da aprendizagem do aluno em um pequeno período de tempo, que é o caso deste curso de duração de um ano e meio, que tem por objetivo formar desenvolvedores de softwares comerciais.

6.1 PROPOSTA DE TRABALHO E APLICAÇÃO PRÁTICA

Tendo em vista o contexto apresentado o professor sentiu a necessidade de modificar sua proposta de trabalho procurando utilizar meios para tornar sua disciplina interessante para os alunos. Pois em seu entendimento, quando despertado o interesse, o próprio aluno envolve-se no processo de aprendizagem, o que o facilita, e propicia menor desistência e reprovação.

Na disciplina de lógica de programação, assim como na matemática, o desenvolvimento do raciocínio é crucial para que os objetivos de ensinar e aprender sejam atingidos. Como então “despertar”, “motivar” os alunos?

A primeira abordagem é a motivacional, o professor desperta a curiosidade no aluno, explica como funciona internamente o caixa-eletrônico do banco, por exemplo: Você já parou para pensar, o porquê de todo o processo que você realiza durante o saque de R$100,00 no caixa-eletrônico? E através deste pequeno exemplo, alguns conceitos de programação são repassados aos alunos, como o armazenamento (banco de dados), variáveis, e o próprio funcionamento algorítmico do computador (seqüencial). Outros exemplos são dados como o do caixa do mercado, da loja. Com este novo modo de perceber o cotidiano o aluno começa a entender como o computador “pensa”.

São propostos aos alunos problemas de lógica, os quais devem ser feitos sem o auxílio do professor. O aluno deverá ao final da disciplina ser capaz de raciocinar sem o modelo do professor. Raciocinar a partir de seu próprio conhecimento é o que se propõe ao aluno. No exercício 1 não é dada nenhuma explicação, no primeiro momento os alunos ficam

revoltados, “o professor na explica”, “eu não estou entendendo nada”, “eu vim para aprender”, porém quando alguns colegas começam a solucionar os exercícios, inicia-se o processo de integração e trocas, e o aluno percebe que cada colega encontrou a solução de forma diferente, e ele mesmo começa o seu processo de solução.

1. Exercício[3] para preencher a seqüência numérica, encontre a lógica das seqüências abaixo:

a. Série linhas simples

1	8	27	64	

b. Três linhas e três colunas

6	7	8
7	9	11
8	11	

c. Três linhas e três colunas:

1	25	9	46	11
10	34	18	55	

O objetivo deste exercício é que o aluno raciocine para solucionar cada problema e descreva o seu pensamento. As respostas aqui são: a) números elevados ao cubo, b) a primeira linha soma 1, a segunda 2 e a última 3; c) a linha de baixo é igual à linha de cima mais 9. No exercício 1 o professor utilizada a abordagem de Vygostsky, onde o outro auxilia no processo de conhecimento, a interação, torna possível, ver muito do que fica oculto durante a explicação do professor, pois quando o professor explica, a maioria é capaz de repetir o processo explicado. Contudo, quando alguns colegas possuem formas diferentes de entender um exercício, o aluno percebe que o raciocínio não é linear e previsível.

O exercício 2 partilha da mesma concepção do exercício 1, porém, agora além de ser dado o exercício em anexo o aluno terá uma folha explicativa de como resolve-lo. Pretende-se assim verificar o nível de compreensão do texto (interpretação) do aluno.

2. Exercício da revista Coquetel[4] de raciocínio lógico de nível fácil: Três casais vivem felizes numa cidade. Com base nas dicas abaixo, tente descobrir o nome de cada marido, a profissão de cada um e o nome de suas respectivas esposas.

1. O médico é casado com Maria.
2. Paulo é advogado.
3. Patrícia não é casada com Paulo.
4. Carlos não é médico.

	Médico	Engenheiro	Advogado	Lúcia	Patrícia	Maria
Carlos			N			
Luís			N			
Paulo	N	N	S			
Lúcia	N					
Patrícia	N					
Maria	S	N	N			

Somente após esta introdução “ao pensar logicamente”, são apresentados aos alunos conceitos de seqüencia lógica, algoritmos e estrutura de construção (condicionais e de repetição).

---

[3] Fonte: http://www.testonline.com.br/teslog01.htm

[4] Revista Coquetel. Desafios de Lógica. n. 48, p. 4-7, set. 2006.

Na ementa a implementação de algoritmos encontra-se praticamente no final dos conteúdos, como o algoritmo é algo abstrato, é importante que a partir do conteúdo de estrutura seqüencial, inicie-se o processo de concretização, pois o aluno tem dificuldade em compreender o algoritmo, a menos que o professor o traduza em uma linguagem de programação e ele posso então visualizar (concretizar) o seu funcionamento. Aliar a teoria com a prática no momento que se ensina as estruturas de construção dos algoritmos facilita a sua compreensão pelos os alunos.

6.2 RESULTADOS OBSERVADOS

A construção de algoritmos e a programação de computadores são atividades abstratas; tornar concreto por meio do papel (exercícios) e implementação dos algoritmos, serve de ponte entre o palpável (exercício), programa e o pensamento (abstrato), auxiliando no processo de pensar logicamente. Sendo assim é importante conceituar pensamento e abstração. Ainda não existe um consenso sobre o conceito de pensamento, aqui utilizaremos o de Jolivet (p. 43, 1972): "[...] pensamento é a capacidade que tem o ser humano de conhecer em que consistem as coisas e as relações que elas têm entre si". Outro conceito importante é o de abstração. Para Oliveira & Amaral (2001), abstração "é um conceito no qual não se leva em conta um valor específico determinado e sim qualquer entre todos os valores possíveis daquilo com que estamos lidando ou ao que estamos nos referindo".

O exposto acima mostra como os exercícios iniciais são fundamentais, o professor ao utilizar estes exercícios, diminui o impacto causado pela apresentação do algoritmo, algo desprovido de significado para o aluno.

Cabe destacar que para Piaget existem dois tipos de abstração: a abstração empírica, que depende do concreto, ou seja, corresponde a atividade mental capaz de abstrair as propriedades dos objetos e a abstração reflexiva a qual tem relação com as operações formais relacionadas à matemática pura. Observamos que a lógica de programação trabalha mais com a abstração empírica de Piaget do que com a abstração propriamente dita, sendo assim o professor que trabalho com esta disciplina terá mais facilidade em sua condução, no momento que trouxer para a realidade do aluno exemplos e exercícios, abordando os conteúdos e ao mesmo tempo resgatando os conceitos conhecidos pelos alunos.

CONCLUSÃO

Por meio do levantamento bibliográfico observaram-se vários esforços para "facilitar" o entendimento das disciplinas introdutórias a programação de computadores, visto que apresentam grande número de evasão e repetência, tanto em universidades como em cursos técnicos. Há várias propostas de ferramentas para auxiliar o ensino-aprendizagem, mas detalhe importante, embora existam ferramentas, não existe uma padronização na metodologia de ensino ou estudos que demonstrem completamente que a utilização destas ferramentas efetive a aprendizagem dos alunos.

O que fica claro é que se está dando muita importância a ferramentas e pouco importância a metodologia de ensino. A literatura recente começa a se deslocar nesse sentido. Buscando métodos que auxiliem os estudantes, neste trabalho foram propostos exemplos motivadores de aplicações, para evitar a abstração pura da programação de computadores; e exercícios de raciocínio lógico, para incentivar o desenvolvimento dessa habilidade. Observou-se boa aceitação pelos alunos e uma melhoria de interesse e participação deles na sala de aula.

REFERÊNCIAS

BROWN, M. H. **Algorithm Animation.** The MIT Press, 1987.

CAMPOS, Dinah M. de S. **Psicologia da Aprendizagem**. 8 ed. Petrópolis: Vozes, 1979.

COOPER, S. , DANN, W. , PAUSCH, R.: Teaching objects first in introductory computer science, In: **Proceedings of the 34th Technical Symposium on Computer Science Education**, Reno, Nevada, USA (2003).

FALKNER, K., PALMER, M. Developing authentic problem solving skills in introductory computing classes, In: **SIGCSE'09**, Chattanooga,Tennessee, USA, 2009.

GARCÍA, E. G.; VEIGA, E. C. da. O construtivismo e as funções mentais. **Diálogo Educacional**, Curitiba, v. 7, n. 20, jan./abr. de 2007.

JOLIVET, R. **Curso de Filosofia**. Edição Brasileira. Rio de Janeiro: Agir, 1972.

LINDEMAN, Eduard C. **The meaning of adult education**. New York, New Republic, 1926.

MALONEY, J., PEPPLER, K., KAFAI, Y. B., RESNICK, M., RUSK, N.: Programming by choice: urban youth learning programming with Scratch, **ACM SIGCSE Bulletin Archive**, vol. 40, pp. 367-371 (2008).

OLIVEIRA, J. M. de; AMARAL, J. R. do. O pensamento abstrato. **Cérebro & Mente**, Campinas, n. 12, fev./abr. 2001.

PAPERT, S. **LOGO: computadores e educação.** São Paulo: Ed Brasiliense, 1985.

POLYA, George. **How to solve it: a new aspect of mathematical method**. Princeton University Press, 2004.

POZO, J. I. **Teorias cognitivas da aprendizagem**. 3 ed. Porto Alegre: Artes Médicas, 2002.

SANTIAGO, R. de; DAZZI, R. L. S. **Ferramenta de apoio ao ensino de algoritmos.** In: SEMINÁRIO DE COMPUTAÇÃO, 2004, Blumenau.

SANTOS, R. P.; COSTA, H. A. X. 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.

WIRTH, Niklaus. Program Development by Stepwise Refinement, **Communications of the ACM**, vol. 14, pp. 221-227, 1971.