PROPOSTA DE 'FRAMEWORK' CONCEITUAL PARA O ENSINO DE ALGORITMOS EM CURSOS TECNOLÓGICOS

PROPOSED CONCEPTUAL 'FRAMEWORK' FOR TEACHING TECHNOLOGY COURSES IN ALGORITHMS

ROBERTO CORREIA DE MELO[1]

Recebido em Janeiro de 2013. Aceito em Março de 2013.

[1]Bacharel em Matemática Aplicada pela Fundação Santo André-SP, Mestre em Ciências pela Universidade São Marcos-SP, Doutor em Ciências Tecnologia Nuclear Aplicações pelo IPEN-USP-SP, Docente do Centro Paula Souza de São Paulo. Av. Antonia Rosa Fioravante, 804. Vila Fausto Morelli. Fatec-Mauá. Mauá. CEP 09751-000. Fone: 011-4543-3221. Fax: 011-4519-5899. www.fatecmaua.com.br. e-mail: robertomelo2006@uol.com.br Fone: 011-99747-6823.

PROPOSTA DE "FRAMEWORK" CONCEITUAL PARA O ENSINO DE ALGORITMOS EM CURSOS TECNOLÓGICOS

RESUMO

Docentes de disciplinas de cursos de Tecnologias de Informação (T.I.) cujos objetivos são apresentar conceitos referentes a algoritmos e lógica de programação enfrentam com frequência dificuldades em suas atividades por não existir um *framework* conceitual para a abordagem dessas disciplinas, que tenha sido testado, seja relativamente simples de implementar, esteja ao alcance do entendimento dos discentes e que seja um roteiro prático para utilização pelos docentes. Este artigo propõe a criação e aplicação de um *framework* conceitual a ser seguido pelos docentes e discentes nas atividades presenciais dessas disciplinas, proporcionando-lhes melhores condições para desenvolverem suas atividades durante a abordagem desses assuntos. O *framework* se baseia na combinação de três fatores: 1) princípios das teorias cognitivas de Inteligências Múltiplas, de Howard Gardner, e *Mindful Learning*, de Ellen Langer; 2) utilização de conceitos heurísticos sobre como resolver problemas propostos por George Pólya; e 3) nas experiências em docência do autor deste artigo em combinar os primeiros dois fatores com técnicas de desenvolvimento de programas, tais como fluxogramas, testes de mesa, pseudocódigo e código-fonte em linguagem "C" em sala de aula e laboratórios.

PALAVRAS-CHAVE: Algoritmos. Lógica de Programação. *Framework* Conceitual. Metodologia de Ensino de Programação. Ensino Superior Tecnológico.

PROPOSED CONCEPTUAL "FRAMEWORK" FOR TEACHING TECHNOLOGY COURSES IN ALGORITHMS

ABSTRACT

Teachers of Information Technology (I.T.) disciplines which have the aim to introduce students to concepts of algorithms, programming logic, and similar, have to face frequently deep dificulties due the non-existence of a conceptual framework which allow teach them. This paper proposes a way to fill this lack of actions by means of creating and applying a "conceptual framework" to be used by teachers and students when they are involved in face-to-face activities. This will construct to them better ways to their activities for one or two semesters of this kind of discipline. The proposed "conceptual framework" is based on three main ideas: 1) over concepts from cognitive theories as Multiple Intelligences, by Howard Gardner, and Mindful Learning, by Ellen Langer, 2) using heuristic concepts about how to solve problems, proposed by George Pólya, and 3) over the teaching experience of this article author in mixing the two first ideas with programming developing techniques as flowcharts, desk tests, pseudocode and source code written in "C" programming language in classes and labs.

KEY WORDS: Algorithms. Programming Logic. Conceptual Framework. Teaching Programming Methodology. Higher Technological Education.

1 INTRODUÇÃO

A noção de algoritmo está presente na história do pensamento humano e em quase tudo o que se faz cotidianamente. Um algoritmo é, segundo Barbosa (2001), uma descrição passo a passo de como um problema computacional deve ser solucionado.

O bom aproveitamento das atividades de ensino e de aprendizagem referentes às disciplinas de cursos de T.I., em geral, denominadas *Algoritmos, Lógica de Programação* ou *Linguagem de Programação I*, constitui-se em um pré-requisito essencial para as disciplinas subsequentes que abordam linguagens de programação, e enfrentam algumas especificidades e dificuldades que necessitam ser analisadas e superadas para que se possa otimizar sua abordagem e preparar melhor discentes para a continuidade de seus estudos nesse ramo essencial das Ciências da Computação.

Muitos processos cognitivos complexos estão envolvidos no desenvolvimento de algoritmos, tais como: comunicar, analisar, sintetizar, avaliar, deduzir, induzir e, aqui, para resumir, optou-se por destacar que o desenvolvimento de algoritmos envolve vários tipos de *tarefas epistêmicas*, que tendem a ocorrer de modo mais completo quando realizadas pelos discentes se esses trabalham colaborativamente e executando-as em pequenos grupos supervisionados pelos docentes.

De acordo com Goodyear (2001), são sete as tarefas epistêmicas envolvidas em processos de aprendizagem colaborativa, agrupadas na "Taxonomia de Ohlsson", resumidas no Quadro 1:

Quadro 1 – Taxonomia de Ohlsson

Tarefa epistêmica	Significado
Descrevendo	Desenvolvendo um discurso referenciando um objeto ou evento tal que uma pessoa que participe do discurso adquira uma concepção acurada do objeto ou evento.
Explanando	Desenvolvendo um discurso em relação a um evento tal que uma pessoa que participe do discurso entenda por que aquele evento aconteceu.
Prevendo	Desenvolvendo um discurso tal que uma pessoa que participe dele entenda que tal evento acontece ou acontecerá sob determinadas circunstâncias, ou seja, entenda as relações de causa e efeito envolvidas no evento.
Arguindo	Argumentando a favor (ou contra) uma posição ou questão em particular, portanto aumentando (ou diminuindo) no receptor a certeza de que a posição ou questão esteja certa.
Criticando	Desenvolvendo um discurso sobre algo (especialmente um produto cultural) tal

(avaliando)	que uma pessoa que participe do discurso fique ciente dos aspectos positivos e negativos envolvidos no assunto.
Explicando	Explicar um conceito é desenvolver um discurso sobre ele tal que uma pessoa que participe do discurso adquira um claro entendimento de seu significado.
Definindo	Definir um termo é propor um uso para ele.

Fonte: Goodyear, 2001, adaptado pelo autor.

Partindo-se das tarefas epistêmicas citadas, e buscando construir o *framework* conceitual que se propõe, tem-se, dentre as especificidades e dificuldades a serem enfrentadas no ensino de algoritmos, que a primeira delas é o docente compreender e utilizar nas práticas do dia a dia os princípios propostos em duas das mais destacadas teorias cognitivas das últimas décadas: a teoria das Inteligências Múltiplas, de Howard Gardner, mundialmente conhecida, e a teoria Mindful Learning *(*até agora sem tradução para o português), de Ellen Langer, também perspicaz e profunda, porém menos conhecida do que a primeira tanto no meio acadêmico quanto fora dele.

A segunda das especificidades e dificuldades a ser superada pelo docente é fazer com que os discentes assimilem um "modo de pensar algorítmico", imprescindível para que eles entendam problemas computacionais e depois desenvolvam algoritmos e programas de computador para solucioná-los. Um esquema conceitual que se considera adequado a ser usado para esse fim é o proposto no trabalho clássico "How to Solve Problems", de Pólya (1957).

A terceira e última das especificidades e dificuldades é capacitar os discentes na utilização de meios práticos e objetivos para realizar o projeto, a criação e a implementação de programas de computador necessários para testar as soluções que criaram, e que são compostos por sete ações: 1) especificar as variáveis envolvidas na solução do problema, 2) desenhar fluxograma, 3) desenvolver teste de mesa, 4) escrever pseudocódigo, 5) escrever o código-fonte na linguagem "C", 6) compilar e executar o módulo resultante do programa fonte "C" criado, e 7) examinar os resultados obtidos.

Essa sequência de ações práticas não tem a abrangência e nem pretende substituir os paradigmas ou métodos de desenvolvimento de software amplamente utilizados atualmente, mas pode encaixar-se neles, em especial nas fases denominadas "implementação" e "testes" dos métodos (ou modelos) denominados "Clássicos (ou cascata ou *waterfall*)" e "Interativo".

A junção de todas as especificidades mencionadas constitui-se no que se denomina aqui "framework conceitual", a ser aplicado nas atividades de ensino e aprendizagem, na forma proposta.

2 ETAPA 1 DO "FRAMEWORK" – APLICAR PRINCÍPIOS DE TEORIAS COGNITIVAS

Uma das teorias cognitivas que tem causado ampla repercussão mundial junto a profissionais de educação em suas atividades de ensino nas últimas décadas é a teoria denominada "Inteligências Múltiplas", originada pelos trabalhos do cientista norte-americano Howard Gardner. Esta teoria considera que existem oito tipos predominantes de inteligências nas pessoas, que precisam ser conhecidas pelos docentes para que eles possam se beneficiar desse conhecimento em suas disciplinas e cujas características estão resumidas no Quadro 2:

Quadro 2 – Os oito tipos de inteligências propostas por Gardner

Inteligência	Pontos fortes	Preferências	Aprende melhor por meio de	Necessita de
VERBAL/ LINGUÍSTICA	Escrita, leitura, Memorização de datas, contar estórias.	Escrever, ler, contar estórias, conversar, memorizar.	Ouvir e ver palavras, falando, lendo, escrevendo.	Livros, cadernos, instrumentos de escrita, diálogos, discussões.
LÓGICO/ MATEMÁTICA	Matemática, lógica, resolução de problemas, raciocínio, padrões	Questões, trabalhos com números, experimentos, resolver problemas.	Relacionamentos e padrões, classificando, abstrações.	Coisas para pensar e explorar, materiais científicos, museus.
VISUAL/ESPACIAL	Mapas, leitura, gráficos, desenhos, visualizações, abstrações.	Desenhar, construir, desenvolver, criar, apreciar pinturas.	Trabalho com cores e pinturas, visualizações, imaginação.	LEGO, vídeos, filmes, slides, arte, jogos, quebra-cabeças.
CORPO-REOCINESTÉSICA	Esportes, dança, competições, uso de ferramentas, atuação.	Mover-se, tocar e falar, linguagem corporal.	Toque, movimento, conhecimento por sensações.	Atuação, coisas para construir, movimento, esportes.
MUSICAL	Sons, lembrar-se de melodias, ritmos, canto.	Cantar, tocar um instrumento, ouvir música.	Ritmo, melodia, ouvindo músicas e melodias.	Tempo para cantar concertos, instrumentos.
INTERPESSOAL	Liderança, organização, resolver conflitos, vender.	Falar a pessoas, ter amigos, unir grupos.	Comparando, relacionando, compartilhando.	Amigos, jogos coletivos, eventos sociais, clubes.

INTRAPESSOAL	Reconhecer pontos fortes e pontos fracos, estabelecer metas, compreender-se.	Trabalhar sozinho, refletir interesses.	Trabalhando sozinho, refletindo, criando projetos.	Lugares secretos, tempo sozinho, projetos próprios, escolhas.
NATURALÍSTICA	Compreender a natureza, fazer distinções, identificar fauna e flora.	Estar envolvido com a natureza, tomar decisões.	Trabalhando com a natureza, explorando coisas vivas.	Ordem, padrões de iguais ou diferentes, conexões com a vida real.

Fonte: Gardner, 1995, adaptado pelo autor.

Três dentre as inteligências apontadas por Gardner aparentam estar mais diretamente ligadas a características envolvidas na resolução de problemas algorítmicos: Verbal/Linguística, Lógico/Matemática e Visual/Espacial. E três outros tipos de inteligência também apresentam potencial de poder auxiliar na implementação e testes das soluções encontradas: Corpóreo-cinestésica, Interpessoal e Intrapessoal. (Sendo assim, em princípio, parece que somente os tipos de inteligências Musical e Naturalística não ajudariam diretamente a entender, projetar e resolver algoritmos).

O docente das disciplinas Algoritmos e suas correlatas, mediante o reconhecimento das características desses seis tipos de inteligências, pode/deve utilizar os pontos fortes e preferências de cada grupo para planejar o modo de ministrar e maximizar o aproveitamento dessas disciplinas.

Outra teoria cognitiva, essa de repercussão menor do que a teoria das Inteligências Múltiplas junto a profissionais de educação nas últimas décadas, mas igualmente profunda e perspicaz em sua percepção e seus conceitos, é a teoria *Mindful Learning*, de Ellen Langer. Nela propõe-se a existência de sete mitos penetrantes (*pervasive*, no original) prejudiciais às atividades cognitivas, frequentemente presentes nas atividades de ensino e de aprendizagem de todos os níveis, e que precisam ser combatidos. O Quadro 3 relaciona os sete mitos penetrantes prejudiciais à cognição propostos por Langer:

Quadro 3 – Sete mitos penetrantes prejudiciais à cognição

Mito	Característica
1	Os assuntos básicos têm que ser aprendidos tão bem a ponto de tornarem-se automáticos para a pessoa.
2	Prestar atenção significa permanecer focado em uma coisa de cada vez.
3	É importante retardar as gratificações.
4	Memorizar é necessário em educação.
5	Esquecer é um problema.
6	Inteligência é conhecer "o que já existe".
7	Só há respostas corretas ou erradas.

Fonte: Langer, 1997, adaptado pelo autor.

A teoria também destaca que há formas desatentas de pensar (*mindlessness*), que prevalecem na maioria das pessoas e que são caracterizadas pela presença de três fatores observáveis nos indivíduos, apontados no Quadro4:

Quadro 4 – Fatores do pensamento desatento (*mindlessness*)

Fator *mindlessness*	Característica
1	Comprometimento com categorias pré-existentes. Ficam-se rigidamente comprometidos com categorias criadas no passado.
2	Comportamento automático. Compreende a tendência em repetir as mesmas ações na presença dos mesmos estímulos.
3	Agir a partir de uma perspectiva única. Compreende responder somente à causa apresentada, sem considerar outras variáveis que possam enriquecê-la.

Fonte: Langer, 1989, adaptado pelo autor.

E a teoria também aponta que há formas atentas de pensar (*mindfulness*), cujo desenvolvimento é desejável na maioria das pessoas, e que são caracterizadas pela presença de três fatores observáveis nos indivíduos, indicados no Quadro 5:

Quadro 5 – Fatores do pensamento atento (*mindfulness*)

Fator *mindfulness*	Característica
1	Criação de novas categorias. Processo cognitivo natural, espontâneo, adaptativo e evolutivo que as crianças têm.
2	Abertura para novas informações. Processo de recepção de novos estímulos, melhorando a aprendizagem.
3	Percepção de mais de uma perspectiva simultaneamente; amplia a criatividade e estimula diferentes reações.

Fonte: Langer, 1989, adaptado pelo autor.

Embora se tenha apresentado apenas superficialmente princípios e características dessas duas teorias cognitivas, é importante que os docentes sejam capacitados a fundo nelas – por meio de cursos ou *workshops* ministrados por especialistas – a fim de utilizá-las com proficiência.

O docente das disciplinas Algoritmos e similares, mediante o reconhecimento dos oito tipos de inteligências – em especial dos seis destacados – e das características dos fatores dos pensamentos atentos e desatentos, deve utilizar os pontos fortes de cada uma para maximizar o aproveitamento dos conteúdos ministrados.

3 ETAPA 2 DO "FRAMEWORK" TEÓRICO – ESTRATÉGIAS HEURÍSTICAS

Na literatura técnica sobre programação de computadores, há inúmeras referências à necessidade de divisão de um problema complexo em suas subpartes a fim de facilitar a resolução completa do problema. Essa abordagem ou método é denominado de "Refinamentos Sucessivos" e é citada, por exemplo, por Ynoguti (2005). Nela, divide-se a busca de solução para um problema computacional em quatro etapas, sendo que a primeira destaca a importância de se entender o problema, a segunda fala da criação de um plano, a terceira aborda a resolução do problema propriamente dito e a quarta fala de aspectos da verificação e validação da solução.

De acordo com Barbosa (2001) os métodos usados para desenvolver algoritmos são baseados no conceito de *programação estruturada*, em que se utiliza uma quantidade limitada de construtores lógicos, que são as estruturas de repetição (E.R.) disponibilizadas pelas próprias linguagens de programação estruturadas, como é o caso das linguagens "C", Java e VB.

Nesse tipo de programação não se usa instruções de desvio para qualquer outra parte do programa, mas apenas desvios internos dentro das E.R., o que é realizado automaticamente dependendo do resultado de testes de condições.

Outro conceito frequentemente presente quando se trata de resolução de algoritmos é a denominada abordagem *top-down*, que consiste em se efetuar refinamentos sucessivos da solução obtida e de evidenciá-la de diferentes formas tão logo seja obtido o refinamento.

Vários modos de raciocínios são importantes para a resolução de problemas computacionais, sendo dois dos principais a *dedução* (forma lógica que garante que uma conclusão é verdadeira quando as premissas forem verdadeiras) e a *indução* (processo de descoberta de leis gerais por meio da observação e da combinação de exemplos particulares), mas também é vital para o sucesso do elaborador de um algoritmo que ele conheça minimamente a arquitetura do computador aonde seu algoritmo/programa será processado.

Pólya (1957) propôs um conjunto de ações lógicas a serem usadas na resolução de problemas computacionais, e as denominou de "Estratégias Heurísticas". *Heurística* é uma palavra grega que encerra um conceito amplo, significa "descubro" ou "encontro". Como substantivo significa "arte ou ciência do descobrimento"; já como adjetivo, como em "estratégias heurísticas" ou "regras heurísticas", refere-se à aplicação prática de procedimentos visando descobrir soluções para os problemas sob estudo.

Heurística é mais do que simplesmente a adoção do pensamento cartesiano de se dividir um problema complexo em várias partes menores; vai além do raciocínio do tipo "entrada-processamento-saída", tentando-se obter cada parte de resolução mais simples. Até porque, quase sempre, além da dificuldade de realizar-se *a priori* essa segmentação, essa subdivisão, quando realizada, não ajuda o discente a efetivamente resolver os problemas envolvidos na etapa de "processamento".

Como é essencial haver uma visão metódica, devem-se utilizar as ações propostas por Pólya, mas apenas como o passo inicial de um framework conceitual a ser usado para atingir os objetivos envolvidos na resolução de algoritmos. E outras ações devem ser tomadas na sequência.

As estratégias heurísticas propostas por ele, resumidas no Quadro 6, dividem-se em quatro grandes fases.

Quadro 6 – As quatro fases propostas por Pólya

Fase	Objetivo
1	Entender o problema
2	Criar um plano de ataque
3	Executar o plano de ataque
4	Analisar os resultados

Fonte: Pólya, 1957, adaptado pelo autor.

Ao mesmo tempo em que apontou esses passos para resolver problemas computacionais, ele alertou para o fato de que eles representam guias gerais (ou *guidelines*) e não regras rígidas, e defendeu que é inerente à natureza de *guidelines* que eles podem não funcionar se forem seguidos literalmente. E, ele alerta, deve-se interpretá-los (e, por extensão, a qualquer conjunto de regras) por meio do que ele classifica como "olhos da experiência".

Dentre os problemas computacionais, a resolução de algoritmos ocupa um lugar de destaque, uma vez que ele envolve aspectos bastante claros e importantes da área de T.I. ou de Ciência da Computação, tais como: compreensão abstrata de problemas, raciocínio lógico, dedução, indução, análise, síntese, avaliação, capacidade de planejamento, capacidade de projeção e criação de soluções, sua implementação e testes quanto ao seu funcionamento.

O Quadro 7 resume características essenciais das quatro fases propostas por Pólya para a resolução de problemas computacionais e que se propõe utilizar e transmitir aos discentes das disciplinas de Algoritmos e suas congêneres. (Na coluna "Ação dos Métodos e Meios Práticos" faz-se uma correlação entre o que foi proposto por Pólya e os passos indicados dentro da terceira etapa do "framework conceitual" esboçado aqui).

Quadro 7 – As quatro fases para resolver um problema

Fase	Título	O que fazer nela (perguntas, respostas e ações)	Ação dos "Métodos e Meios Práticos"
1	ENTENDENDO O PROBLEMA	O que é desconhecido? Quais são os dados? Qual é a condição? É possível satisfazer a condição? A condição é suficiente para se determinar o que é desconhecido? Ou é insuficiente? Ou é redundante? Ou é contraditória? Desenhe uma figura. Introduza uma notação adequada. Separe as várias partes da condição. Você pode escrevê-las?	1) ESPECIFICAR AS VARIÁVEIS
2	CRIANDO UM PLANO	Você já resolveu problema parecido antes? Você conhece um problema semelhante a esse? Em havendo um problema semelhante já resolvido antes, você pode usá-lo agora? Você pode usar seu método? Pode fazer adaptações agora para usar um método anterior? Pode reformular o problema?	2) DESENHAR O FLUXO-GRAMA, 3) EXECUTAR O TESTE DE MESA, 4) ESCREVER O PSEUDO-CÓDIGO, 5) ESCREVER PROGRAMA "C"
3	EXECUTANDO	Execute o plano em busca de soluções.	6) COMPILAR E EXECUTAR

	O PLANO	Verifique cada passo. Você pode confirmar claramente se cada passo executado está correto? Você pode provar que cada passo está correto?	O PROGRAMA "C"
4	EXAMINANDO A SOLUÇÃO OBTIDA	Você é capaz de confirmar o resultado? Você tem condições de checar os argumentos? Você consegue usar os resultados ou os métodos utilizados em outros problemas.	7) EXAMINAR OS RESULTADOS OBTIDOS E CONFRONTÁ-LOS COM AS NECESSIDADES INICIAIS

Fonte: Pólya, 1957, adaptado pelo autor.

Pólya destaca que, para a resolução de problemas computacionais, tão importante quanto ter um plano, é o professor ajudar seus alunos, e que esta missão não é fácil, pois demanda tempo, prática, devoção e princípios. Ele lembra que alunos, deixados sem ajuda, podem não apresentar progresso algum com o decorrer do tempo. Por outro lado, se o professor ajudar demasiado, pouco pode restar como desafio para os alunos. Então ele conclui que o professor deve ajudar, mas não em demasia nem tão pouco, de modo que os estudantes tenham que realizar a parte mais significativa dos esforços envolvidos.

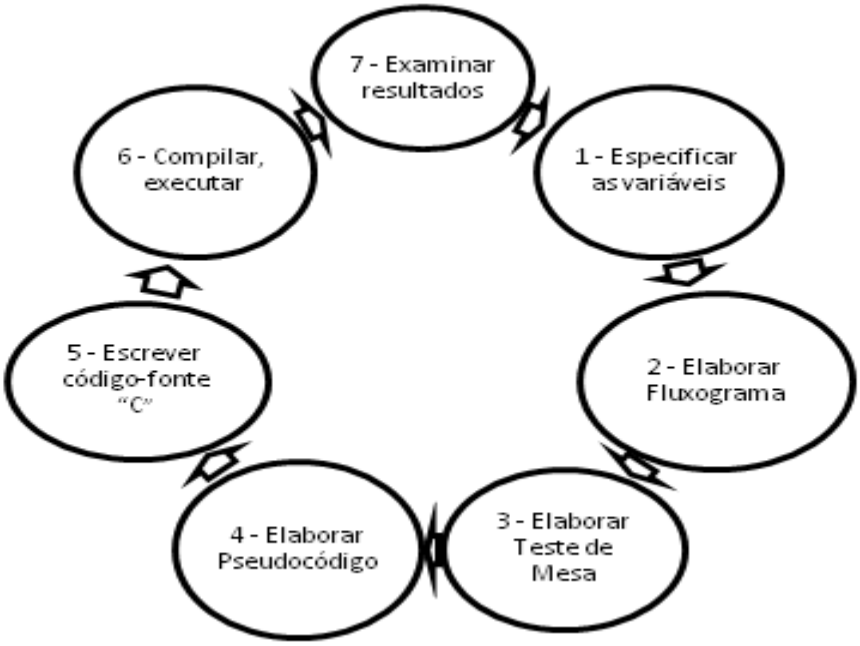
4 ETAPA 3 DO "FRAMEWORK" – MÉTODOS E MEIOS PRÁTICOS

A terceira e última das especificidades é prover métodos e meios práticos e objetivos aos docentes e discentes para maximizar o aproveitamento da disciplina Algoritmos e suas similares, que serão usados na criação e implantação de programas de computador e para testarem as soluções que criaram.

Nessa abordagem, essa etapa é composta por sete ações interligadas e que se reforçam mutuamente: 1) especificação das variáveis envolvidas na solução do problema, 2) desenho do fluxograma, 3) criação do teste de mesa, 4) escrita do pseudocódigo, 5) elaboração do código-fonte da linguagem "C", 6) compilação e execução do programa fonte criado e 7) exame dos resultados.

Cabe ressaltar que é de fundamental importância que os discentes trabalhem em pequenos grupos, de 3 ou 4 integrantes, supervisionados pelo docente, já que o trabalho colaborativo é um dos elementos-chave para se atingir mais rápida e eficazmente os objetivos em quaisquer atividades de ensino e aprendizagem. A Figura 1 exibe as ações envolvidas para resolver problemas computacionais:

Figura 1 – As sete ações para resolver um problema propostas



Fonte: Elaborado pelo autor.

Cada ação proposta está detalhada a seguir:

1) ESPECIFICAR AS VARIÁVEIS

É essencial que o discente que está aprendendo algoritmos consiga entender claramente o enunciado do problema e o docente detecte se esse objetivo foi atingido por meio da clareza com que o discente isola e define as variáveis envolvidas na solução.

Dado um enunciado, espera-se que o discente identifique e isole: a) quais são as variáveis de entrada (*input*); b) quais são as variáveis de saída (*output*); c) quais são as variáveis auxiliares (ou de "controle da solução") e, se possível, as identifique *a priori*, d) quais as variáveis de cálculo (ou de "meio") a serem usadas na solução. Se a identificação de todas as variáveis não ocorrer antes de o discente iniciar a resolução do algoritmo, ele pode fazê-lo no decorrer da solução, mas isso pode reduzir a qualidade da solução encontrada.

Observando-se um enunciado-exemplo, como o que segue, do qual um discente teria que "extrair" as variáveis envolvidas: *Desenvolver fluxograma para solicitar ao usuário os seguintes dados de até 50 alunos: a) seu RM (inteiro; 0=fim dos dados; positivo (fazer consistência dos dados de entrada)); b) sua ALTURA (em cm; real; até 210 cm; fazer consistência dos dados de entrada). Ao final, exibir na tela: a) a maior altura encontrada; b) a média das alturas informadas.*

Da leitura desse enunciado espera-se que o discente identifique e separe:

a) Variáveis de entrada: *RM* de cada aluno; *ALTURA* de cada aluno;

b) Variáveis de saída: A *maior altura* encontrada; A *média das alturas* dos alunos;

c) Variáveis auxiliares (ou de "controle"): *I* (controlador de repetições, já que essa solução precisa das estruturas de repetição existentes. *Para* ou *For*, nesse caso.)

d) Variáveis de cálculo (ou de "meio"): após identificar a média das alturas como sendo um dos resultados solicitados, tem-se que obter o *somatório das alturas* encontradas – cujo valor será dividido pela quantidade de alunos para se obter a média procurada.

O Quadro 8 resume as variáveis necessárias para a resolução do enunciado acima.

Quadro 8 – Especificação das variáveis envolvidas na solução.

Tipo da variável	Nome da variável	Significado
ENTRADA	RM	É o RM de cada aluno
ENTRADA	ALT	É a ALTURA de cada aluno
SAIDA	MALT	É a maior altura encontrada entre as alturas informadas
SAIDA	MED	É a média das alturas informadas
CALCULO	I	É o controlador de repetições (e o contador de alunos)
CALCULO	SALT	É a soma das alturas informadas

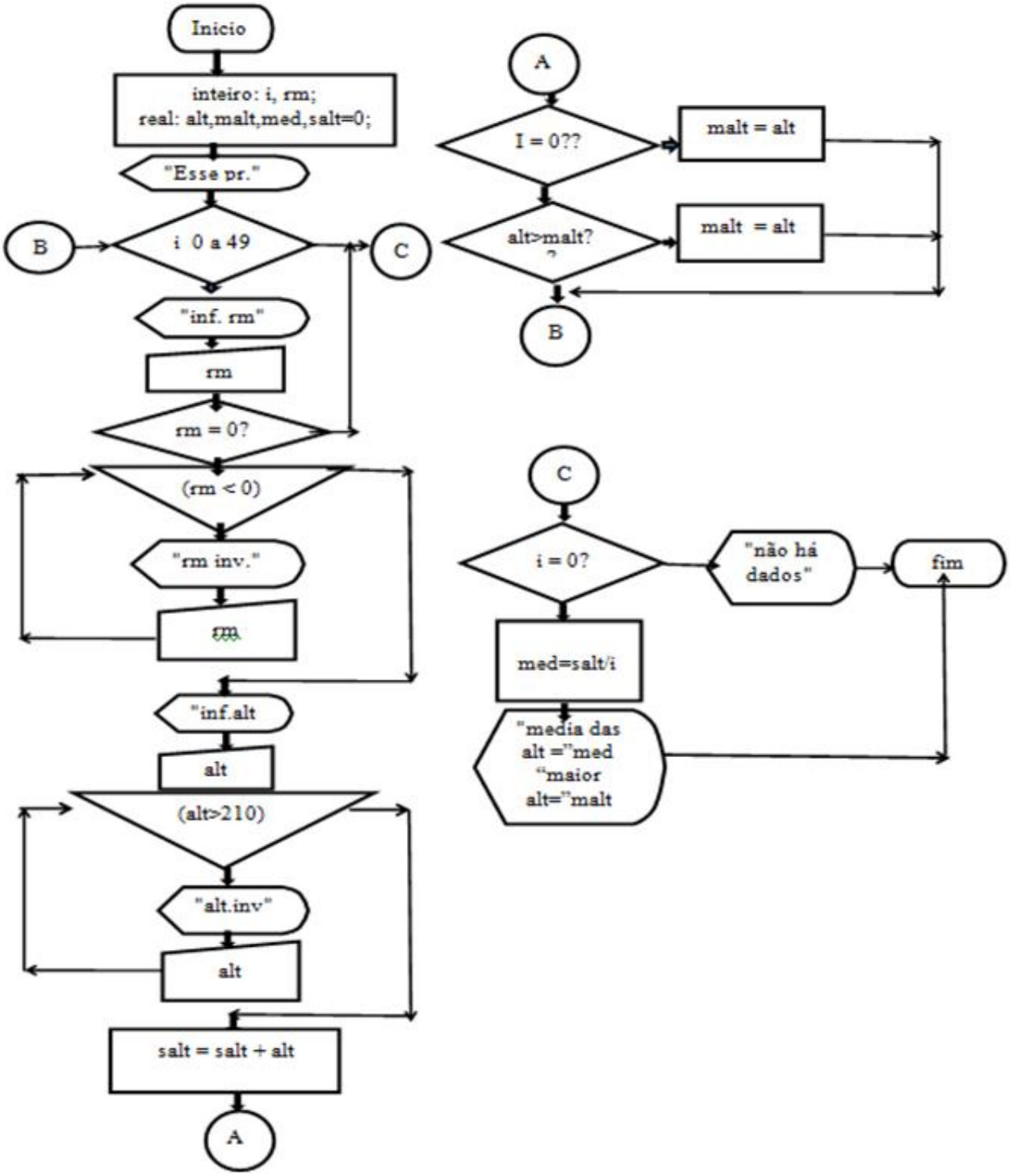
Fonte: elaborado pelo autor.

2) DESENHAR O FLUXOGRAMA

Para demonstrar que compreendeu o fluxo lógico de ações a serem realizadas para solucionar o problema proposto, o discente deve ser capaz de desenvolver um fluxograma detalhando sua proposta de solução. Lembrando-se que cada discente/programador pode ter uma visão distinta para resolver cada enunciado e que não há uma solução única.

Para o enunciado citado, apresenta-se a seguir, na Figura 2, um fluxograma que o resolve:

Figura 2 – Fluxograma para resolver o problema proposto



Fonte: elaborado pelo autor.

3) EXECUTAR O TESTE DE MESA

A fim de fazer um teste lógico para analisar se a solução projetada no fluxograma funciona bem ou não, recomenda-se que o discente elabore um "teste de mesa" baseado nele, que é uma

tabela na qual cada variável do fluxo ocupa uma (ou mais) colunas da tabela, cada variável é colocada na tabela na ordem exata em que aparece no fluxograma, e cada linha da tabela representa uma execução (da estrutura de repetição usada no fluxo que controla o "grande *looping"* do programa).

O desenvolvedor então simula a entrada de dados, executando rigorosamente cada instrução do fluxo, e estabelecendo com o fluxograma um "diálogo" - no qual o fluxo "faz as perguntas e o desenvolvedor as responde" – e vai preenchendo a tabela de valores. (Os valores rejeitados pela consistência obviamente não aparecem na versão final do teste de mesa.) Para o fluxograma apontado, tem-se o teste de mesa conforme consta no Quadro 9:

Quadro 9 – Exemplo de teste de mesa.

I	RM	ALT	SALT=0	MALT	MED
0	123	170	170	170	
1	124	180	350	180	
2	125	160	510		
3	211	175	685		
4	212	165	850		
5	213	185	1035	185	
6	321	165	1200		
7	322	155	1355		
8	0			185	169,375

Fonte: elaborado pelo autor.

4) ESCREVER O PSEUDOCÓDIGO

A fim de preparar a etapa seguinte, que será a elaboração do código-fonte em linguagem "C", elabora-se aqui um pseudocódigo, que é um código simulado, semelhante ao gerado pela linguagem "C" mas ainda em português.

```
INICIO.
DECLARAÇÃO DE VARIÁVEIS.
INTEIRO: I;
REAL: ALT, MALT, MED, SALT=0;
EXIBIR "ESSE PROGRAMA... (COMPLETAR COM O ENUNCIADO)";
PARA I DE 0 A 49 FAÇA
{ EXIBIR "INFORME RM (0=FIM DOS DADOS):"; LER RM;
  SE (RM=0) ABANDONE;
```

```
 ENQUANTO (RM<0) FAÇA
 { EXIBIR "RM NÃO PODE SER NEGATIVO."; LER RM; }
 EXIBIR "INFORME ALTURA: "; LER ALT;
 ENQUANTO (ALT>210)
{EXIBIR "ALT.NÃO PODE SER MAIOR QUE 210:"; LER ALT;}
 SALT = SALT + ALT;
 SE (I=0) MALT = ALT;
 SENÃO SE (ALT>MALT) MALT=ALT;
} // fim do "para"
SE (I>0) {MEDALT=(SALT)/I; EXIBIR "MEDIA DAS ALTURAS = " MED "MAIOR
ALT="MALT;}
SENÃO EXIBIR "NÃO HÁ DADOS.";
FIM.
```

5) ESCREVER O CÓDIGO-FONTE EM LINGUAGEM "C"

A etapa seguinte na solução do problema é compilar o programa-fonte gerado, solucionar eventuais erros e executar o módulo resultante final. A solução projetada nesse caso resulta no código-fonte em linguagem "C" a seguir. A linguagem "C" é a nossa escolhida devido à sua relativa simplicidade, ao seu amplo uso comercial e à ampla disponibilidade de seu ambiente de programação e compilação, passível de ser instalado, gratuitamente, em qualquer computador.

```
#include <iostream>
using namespace std;
// Exemplo de código-fonte em linguagem "C", com o uso de macros de I/O do C++
int main()
{ int i;
  float alt, malt, med, salt=0;
  cout << "\n Esse programa...";
  for (i=0; i < 50; i++)
  { cout << "\n Informe rm (0=fim dos dados): "; cin    >> rm;
    if (rm ==0) break;
    while (rm<0)
    { cout << "\nRM inválido informe novamente: "; cin >> rm; }
      cout << "\nInforme altura: "; cin >> alt;
    while (alt>210)
    { cout << "\nAltura inválida informe novamente: "; cin >> alt; }
     salt = salt + alt;
```

```
    if (i==0) malt = alt;
    else if (alt > malt) malt=alt;
}// fim do for()
if (i >0) { med = float(salt/i); cout << "\nMedia das alturas = " << medalt << "\n\n";
            cout << "\nMaior altura=" malt;}
else cout << "\n Nao há dados." << "\n\n" ;
system ("pause");
return 0;
}//fim do main()
```

6) COMPILAR E EXECUTAR O MÓDULO DO PROGRAMA FONTE "C" CRIADO

A sexta e penúltima etapa na solução do problema é compilar o programa-fonte gerado, solucionar eventuais erros de compilação e executá-lo, verificando se os resultados obtidos são consistentes com o entendimento global do problema.

No ambiente de programação, compilação e execução de programas do C/C++, o Dev-C++, basta abrir a aba "Executar" e acionar a opção "Compilar/Executar". Deve-se digitar os dados exatamente como já simulados no teste de mesa apresentado.

7) EXAMINAR OS RESULTADOS OBTIDOS

A última etapa é confrontar os resultados obtidos com o que foi solicitado e analisar a correção e a aderência dos resultados.

5 CONSIDERAÇÕES FINAIS

Apresenta-se aqui uma proposta de *framework* conceitual a ser adotado nas atividades presenciais que envolvam práticas de ensino e de aprendizagem de disciplinas como Algoritmos e Lógica de Programação, para todos os cursos em que essas disciplinas ocorrem.

Os livros-texto que abordam o assunto Algoritmos e Lógica de Programação, em sua maioria, apenas resolvem os problemas diretamente sem se preocupar em apresentar ou aprofundar um método heurístico a ser adotado para essa finalidade.

Como todo *framework* conceitual, para aperfeiçoar seus resultados, o proposto aqui também necessita ser aplicado, testado e aperfeiçoado exaustivamente. Cabe ressaltar que cada solucionador de problema ou programador de computadores raciocina de um modo particular, mas se constata, ao longo de sua aplicação, que a sequência de atividades propostas aqui pode auxiliar o raciocínio da

maioria dos programadores, porque tem a característica de abordar métodos e meios objetivos a serem utilizados durante a resolução de um problema/algoritmo computacional.

Além dos métodos e meios práticos usados durante a resolução dos problemas, propõem-se aqui adotar ações visando capacitar tanto docentes quanto discentes no conhecimento de duas teorias cognitivas que podem suportar todo o processo, bem como se aponta a necessidade de se manter atenção a aspectos heurísticos para se atingir tal objetivo.

Há alternativas para a ação 6 dos "métodos e meios práticos" aqui proposta (compilar e executar o módulo do programa "C" criado). Uma delas é utilizar a pseudolinguagem de programação VisualG, que permite simular a elaboração e o teste de programas. Outra alternativa é o ASA – Ambiente de Simulação e Animação de Algoritmos, ferramenta que auxilia na criação de fluxogramas e na sua tradução para algumas linguagens de programação reais, mas acredita-se que é mais objetivo, e viável, desde o início, o uso de um ambiente real e profissional de programação, como é o Dev-C++.

REFERÊNCIAS

BARBOSA, L.M. **Ensino de algoritmos em cursos de computação**. São Paulo: Editora EDUC, 2001.

GARDNER, H. E. **Frames of mind, the theory of Multiple Intelligences**. New York: Basic Books, 1983.

GARDNER, H. E. **Inteligências múltiplas**: a teoria na prática. Trad. Maria A. V. Veronese. Porto Alegre: Editora Artes Médicas, 1995.

GOODYEAR, P. Psychological foundations for networked learning. In: **Networked learning**., New York, USA: Springer books, 2001.

LANGER, E.J. **Mindfulness**. Massachusetts, USA: Perseus Books Reading, 1989.

_____. **The power of mindful learning**. Massachusetts, USA: Perseus Books Reading, 1997
PÓLYA, George. **How to solve it**? New York: Penguin Books, 1957.

SCHILDT, H. **C completo e total**. São Paulo: Makron books, 1996.

YNOGUTI, C. A. Uma metodologia para o ensino de algoritmos. **GCETE – Global Congress on Engineering and Technology Education.** São Paulo, Brasil. 2005.