Universidade Federal de Juiz de Fora

Instituto de Ciências Exatas

PROFMAT (Mestrado Profissional em Matemática em Rede Nacional)

Thamara Brassolins Reiff

Programação de computadores: Uma proposta para o 9º ano do Ensino Fundamental.

Juiz de Fora

2017

Thamara Brassolins Reiff

Programação de computadores: Uma proposta para o 9º ano do Ensino Fundamental.

Dissertação apresentada ao PROFMAT (Mestrado Profissional em Matemática em Rede Nacional), da Universidade Federal de Juiz de Fora como requisito parcial a obtenção do grau de Mestre em Matemática. Área de concentração: Ensino de Matemática.

Orientador: Luís Fernando Crocco Afonso

Juiz de Fora

2017

Ficha catalográfica elaborada através do Modelo Latex do CDC da UFJF
com os dados fornecidos pelo(a) autor(a)

Reiff, Thamara Brassolins.
Programação de computadores : Uma proposta para o 9º ano do Ensino Fundamental. / Thamara Brassolins Reiff. – 2017.
65 f. : il.

Orientador: Luís Fernando Crocco Afonso
Dissertação (Mestrado) – Universidade Federal de Juiz de Fora, Instituto de Ciências Exatas. PROFMAT (Mestrado Profissional em Matemática em Rede Nacional), 2017.

1. VisuAlg. 2. Algoritmo. 3. Programação. I. Afonso, Luís Fernando Crocco, orient. II. Título.

Thamara Brassolins Reiff

Programação de computadores: Uma proposta para o 9º ano do Ensino Fundamental.

Dissertação apresentada ao PROFMAT (Mestrado Profissional em Matemática em Rede Nacional), da Universidade Federal de Juiz de Fora como requisito parcial a obtenção do grau de Mestre em Matemática. Área de concentração: Ensino de Matemática.

Aprovada em: 02 de agosto de 2017.

BANCA EXAMINADORA

Professor Dr. Luís Fernando Crocco Afonso
Universidade Federal de Juiz de Fora

Professor Dr. Sandro Rodrigues Mazorche
Universidade Federal de Juiz de Fora

Professor Dr. Jorge Andrés Julca Avila
Universidade Federal de São João del-Rei

AGRADECIMENTOS

Agradeço primeiramente aos meus pais que em todas as etapas da minha vida se fizeram presentes, se dedicaram e renunciaram de tempo e realizações pessoais para que eu tivesse a oportunidade de estudar e ter uma boa formação profissional, além da formação pessoal. Eu devo tudo que sou a vocês, e se sinto orgulho de mim e do lugar onde cheguei é por que me espelho em vocês, não apenas nos ensinamentos, mas no exemplo de homem e mulher que vocês são para mim.

Ao meu irmão pelo apoio, compreensão e por alegrar minhas duras tardes escrevendo este trabalho.

Ao meu orientador, Professor Luís Fernando Crocco Afonso, pelo carinho e paciência. Posso dizer que tive ótimos professores ao longo da minha formação, mas sem dúvidas em você eu vejo o profissional que eu gostaria de ser.

A todos os professores do PROFMAT pelos ensinamentos e dedicação.

À CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) pela concessão da bolsa durante todo o período de realização deste mestrado.

Aos meus amigos, pelas alegrias, tristezas e angústias compartilhadas durante todo o mestrado, o apoio de vocês foi fundamental.

A todos que de alguma forma contribuíram, direta ou indiretamente, para que esse objetivo fosse alcançado.

“Ensinar não é transferir conhecimento, mas criar as possibilidades para a sua própria produção ou a sua construção.”

(Paulo Freire)

RESUMO

Este trabalho é uma proposta para o ensino de algoritmos de programação para alunos do 9º ano do Ensino Fundamental. Ele apresenta uma breve história da disseminação do ensino de algoritmos, especialmente no Brasil, bem como sua definição e principais formas de representação. Além disso, pensado como um projeto a ser desenvolvido no contraturno e na modalidade extraclasse, este trabalho apresenta um estudo teórico sobre VisuAlg e seus principais comandos e sugere 18 atividades que introduzem gradualmente conceitos básicos para a construção de algoritmos. Cada atividade apresenta um problema matemático que deve ser pensado e estruturado sob a forma de pseudocódigo e escrito diretamente em VisuAlg.

Palavras-chave: VisuAlg. Algoritmo. Programação.

ABSTRACT

This work is a proposal for the teaching of a programming algorithm for 9th grade students. It presents a brief history of the dissemination of algorithm teaching, especially in Brazil, as well as its definition and main forms of representation. In addition, thought as a project to be developed in a after/before school program and in a extra shift modality, this work presents a theoretical study on VisuAlg and its main commands and suggests 18 activities that gradually introduce basic concepts for the construction of algorithms. Each activity presents a mathematical problem that must be thought and structured in the form of pseudocode and written directly in VisuAlg.

Key-words: VisuAlg. Algorithms. Programming.

SUMÁRIO

1 INTRODUÇÃO

Se analisarmos o ambiente escolar contemporâneo, é possível identificar uma nova geração emergente, a dos "nativos digitais" habituados com a World Wide Web, com os smartphones, tablets, e o melhor da tecnologia.

No entanto, perante a chamada "geração Z", que engloba os nascidos entre os anos de 1992 a 2010 e está intimamente ligada à expansão extraordinária da internet e de demais aparatos tecnológicos, a prática docente, especialmente no campo da matemática, permanece estacionada em seus métodos expositivos tradicionais.

Os recursos didáticos comumente utilizados em sala, como giz, quadro-negro e livro didático, são ultrapassados para uma realidade na qual os alunos estão super conectados tecnologicamente e transitam o tempo todo conectados em seus celulares. Isso reflete diretamente na dinâmica da aprendizagem, uma vez que essa tradicionalidade não desperta o interesse dos alunos, reproduzindo uma sala de aula literalmente apática. A tecnologia faz parte da vida dos jovens e eles querem utilizá-la.

Consoante essa realidade, este trabalho propõe a inserção da tecnologia, através da construção de algoritmos de programação, contrapondo ao simples uso de aplicativos ou softwares educativos. O objetivo é que os alunos não se limitem a consumir tecnologia, mas a aprendam a construí-la.

Dessa forma, o intuito é promover um aprendizado que ultrapasse os limites dos planejamentos básicos para o ensino da Matemática propriamente ditos, focalizado no raciocínio lógico e dedutivo que é imprescindível na construção dos mais diversos saberes escolares, de modo a incentivar a criatividade e, sobretudo, o protagonismo no processo de aprendizagem. Assim, o aluno que antes era mero receptor e reprodutor de informações, poderá valer-se da tecnologia para construir o conhecimento e, portanto, ter um recurso com o qual possa criar, pensar, manipular a informação.

O objetivo, no entanto, não é abandonar o propósito da educação básica e moldar futuros programadores, mas investir em uma tentativa válida de interligar os conhecimentos fundamentais de tecnologias computacionais, incorporados no cotidiano de todos, com o conteúdo matemático ministrado nas escolas de ensino básico da rede pública de ensino.

Todavia, é bem-aceito que alguns alunos acabem se identificando como programadores no futuro. Assim, a educação básica, que pretende garantir o desenvolvimento social do educando, estará cumprindo o disposto nos artigos 22 da Lei de Diretrizes e Bases da Educação nacional - LDB 9384/96 e 205 da Constituição Federal de 1988, que garantem ao jovem o direito de acesso à educação promovida de forma a desenvolver e preparar o discente para o exercício da cidadania e para a qualificação profissional.

Para tanto, a proposta a aplicar-se no contraturno e na modalidade extraclasse

compenetrou-se no 9° ano do Ensino Fundamental, período no qual os conceitos matemáticos estão mais abstratos e displicência dos alunos com tais conceitos ficam bastante evidentes. É nesse período também que muitos jovens procuram tracejar caminhos mais concretos, como cursos técnicos profissionalizantes.

Dividido em 4 capítulos, o presente trabalho objetiva o aprendizado da construção de algoritmos de programação, e para isso os alunos precisam entender e organizar os processos matemáticos na resolução prática de problemas cotidianos.

No capítulo primeiro, a abordagem recai sobre a história da informática educativa no Brasil desde seu início nas universidades, na década de 70, projetos educacionais e sobre a inserção da construção de algoritmos de programação nas escolas.

No segundo capítulo, o conteúdo está focalizado no conceito e na aplicabilidade do algoritmo, assim como as diversas formas de representá-lo.

No terceiro capítulo, é exposto o estudo sobre o VisuAlg, software utilizado para criar, editar, interpretar e também executar os algoritmos. Este estudo engloba os tipos de variáveis, os principais comandos e funções, e também as estruturas condicionais e de repetição.

No quarto e último capítulo, é proposta uma sequência de atividades que envolvem a construção de algoritmos em situações que contemplam os conteúdos básicos da matemática, compreendendo declaração de variáveis, comando de entrada, saída e atribuição, funções do VisuAlg, estrutura condicional e estrutura de repetição.

1.1 HISTÓRIA DA INFORMÁTICA EDUCATIVA NAS ESCOLAS BRASILEIRAS

No Brasil, assim como aconteceu em diversos outros países, a utilização do computador como uma ferramenta para a educação teve início com algumas experiências em universidades. Diversos autores, dentre eles Maria Candida Moraes [13], João Kerginaldo Firmino do Nascimento [14], Ramon de Oliveira [16] e José Armando Valente e Fernando José de Almeida [21] relatam a trajetória da informática educativa no Brasil desde a década de 70, quando começou a ser introduzida com foco no meio acadêmico, até o final da década de 90. O relato a seguir é baseado nos autores e referências citados acima.

Segundo os autores, a informática educativa se apresentou pela primeira vez ao ser discutida em um seminário sobre o uso de computadores no ensino de fisica, na USP de São Carlos, em colaboração com a Universidade de Dartmouth dos Estados Unidos. No entanto, a Universidade Federal do Rio de Janeiro é apontada pelos autores como instituição pioneira na utilização do computador em atividades acadêmicas. Apesar de inicialmente voltado para o ensino da informática, em 1973 seu uso se dava no desenvolvimento de simulações com alunos da disciplina de química.

Em 1975, na Universidade Estadual de Campinas (Unicamp), um grupo de pesquisadores coordenado pelo professor Ubiratan d'Ambrósio escreveu o documento "Introdução de Computadores nas Escolas de 2^o Grau", financiado pelo acordo do Ministério da Educação (MEC) com o Banco Interamericano de Desenvolvimento (BID), mediante convênio com o Programa de Reformulação do Ensino (Premen – MEC). E, no mesmo ano, a Unicamp recebereu a visita para cooperação técnica internacional dos cientistas Seymour Papert e Marvin Minsky, criadores de uma nova perspectiva em inteligência artificial. Visita essa que refletiu na qualidade dos trabalhos desenvolvidos pela Unicamp.

No ano seguinte, um grupo de pesquisadores da Unicamp visitou Instituto de Tecnologia de Massachusetts nos Estados Unidos (MIT-USA), e seu retorno permitiu a criação de um grupo interdisciplinar envolvendo especialistas das áreas de computação, linguística e psicologia educacional, dando origem às primeiras investigações sobre o uso de computadores na educação, utilizando a linguagem Logo[1].

No início de 1983, foi instituído o Núcleo Interdisciplinar de Informática Aplicada à Educação (Nied) da Unicamp, que com apoio do MEC, teve durante muitos anos o Projeto Logo como grande referencial, envolvendo crianças.

As décadas de 80 e 90 tiveram grande atenção do MEC quanto à preocupação com o uso de programas que associassem educação e informática. Em 1981, na Universidade de Brasília, foi realizado um importante evento, o "I Seminário Nacional de Informática na Educação", que contou com a participação de especialistas nacionais e internacionais, constituindo-se no primeiro fórum a pesquisar o uso do computador como ferramenta auxiliar do processo de ensino-aprendizagem.

Nesse seminário ficou clara a preocupação para que a aquisição e o uso de programas fossem baseados na realidade brasileira. Segundo Moraes:

> Entre as recomendações, destacavam-se aquelas relacionadas à importância de que as atividades de informática na educação fossem balizadas por valores culturais, sociopolíticos e pedagógicos da realidade brasileira, bem como a necessidade do prevalecimento da questão pedagógica sobre as questões tecnológicas no planejamento de ações. O computador foi reconhecido como um meio de ampliação das funções do professor e jamais como ferramenta para substituí-lo. ([13], p. 4)

[1] A linguagem Logo foi criada no final da década de 60, com finalidades educacionais, por um grupo de pesquisadores do MIT-USA e liderados por Seymour Papert, este que trabalhou com Piaget e foi influenciado pelas teorias do construtivismo. A Logo tem como proposta colocar a criança para realizar ações de comandando de um robô (ou uma representação de robô na tela do computador). A ideia era que a criança deveria ensinar o computador a realizar alguma coisa por meio dos comandos. Ela poderia interagir livremente e o conhecimento seria construído a partir da reflexão sobre as reações decorrentes dos comandos dados.

O "II Seminário Nacional de Informática na Educação" aconteceu no ano seguinte e buscava coletar novos subsídios para a criação dos projetos piloto, a partir de reflexões dos especialistas das áreas de educação, psicologia, informática e sociologia. Dentre as recomendações, destaca-se a necessidade de que a presença do computador na escola fosse encarada como um recurso auxiliar ao processo educacional e jamais como um fim em si mesmo. Ainda, havia a recomendação de que as aplicações do computador não deveriam se restringir ao 2º grau.

A partir das recomendações dos seminários, em 1983 a Comissão Especial de Informática na Educação (CE/IE) apresentou o projeto Educação com computadores (Educom). Dentre as metas do projeto Educom, destaca-se a pesquisa do uso educacional da informática. Segundo ele foram implantados centros-piloto na Universidade Federal de Pernambuco (UFPE), Universidade Federal do Rio de Janeiro (UFRJ), Universidade Federal de Minas Gerais (UFMG), Universidade Federa do Rio Grande do Sul (UFRGS) e Universidade Estadual de Campinas (Unicamp). Esses centros-piloto tinham como principal atividade o desenvolvimento de pesquisa sobre o uso da informática na educação, além de dedicarem-se à formação de recursos humanos e à produção de softwares educativos.

Criado por recomendação do Comitê Assessor de Informática e Educação do Ministério da Educação (CAIE/MEC) e implementado em 1987, o Projeto Formar foi um curso de especialização de 360 horas para professores e técnicos das redes municipais e estaduais, com o objetivo de capacitá-los para implantação e uso dos computadores nas secretarias de Educação às quais estavam vinculados, criando os Centros de Informática Educativa (Cied).

Diversos outros projetos e programas de incentivo a utilização do computador na educação surgiram no decorrer dos anos. No entanto a tecnologia evolui a uma velocidade muito superior aos estudos e pesquisas de utilização com fins pedagógicos. A computação se propaga nas escolas, porém ainda muito distante do desejável. Conforme destacam Valente e Almeida em [21],

> A História da Informática na Educação no Brasil data de mais de 20 anos. Nasceu no início dos anos 70 a partir de algumas experiências na UFRJ, UFRGS e UNICAMP. Nos anos 80 se estabeleceu através de diversas atividades que permitiram que essa área hoje tenha uma identidade própria, raízes sólidas e relativa maturidade. Apesar dos fortes apelos da mídia e das qualidades inerentes ao computador, a sua disseminação nas escolas está hoje muito aquém do que se anunciava e se desejava. A Informática na Educação ainda não impregnou as idéias dos educadores e, por isto, não está consolidada no nosso sistema educacional.

1.2 PROGRAMAÇÃO INSERIDA NA EDUCAÇÃO BÁSICA

Nos dias de hoje, aprender a programar é um diferencial comparado ao que foi aprender inglês há alguns anos. Rafael Carvalho cita em [4] cita em seu texto que o diretor do grupo Lifelong Kindergarten do MIT Media Lab, Mitchel Resnick, defende a programação como uma das habilidades do século 21 e deveria ser tão importante quanto ler ou escrever. Para Mitchel Resnick essa importância não se limita apenas às oportunidades de trabalho, ela possibilita ver o mundo de novas maneiras.

Devido a sua importância, e também pela falta de profissionais com habilidades em programação, diversos países perceberam que este aprendizado deve ser incorporado no currículo das crianças ainda na fase escolar. Dessa forma, desde cedo elas têm acesso a linguagem de programação, o que traz benefícios a outras disciplinas e também para o desenvolvimento de diversas habilidades pessoais como raciocínio lógico, trabalho em equipe, tomada de decisões e criatividade.

À frente do seu tempo, Israel foi pioneiro ao incluir o ensino de programação no currículo do ensino básico, já na década de 1990, revolucionando o currículo da escola básica ao incluir o ensino de algoritmos de programação. Ciência ainda pouco conhecida para muitos países e com poucos estudiosos na época.

Mais de uma década depois, muitos países estão seguindo os passos de Israel, dentre eles a Inglaterra que integrou, em 2014, o ensino de ciência da computação ao currículo do ensino básico como disciplina obrigatória. Com isso, crianças a partir de cinco anos de idade já são capazes de criar e executar programas de computadores nas aulas, além de aprender conceitos sobre a segurança na internet, como manter a privacidade e prever situações de perigo.

O ensino da programação de computadores nas escolas é cada vez mais incentivado, inclusive por personalidades da tecnologia. Segundo Wendel Geraldes em [11], Bill Gates, Mark Zuckerberg, Jack Dorsey, o cantor Will.i.am e políticos como Al Gore e Michael Bloomberg, têm defendido publicamente o ensino de programação nas escolas como forma de inclusão digital. Para eles, interpretar e escrever códigos é tão importante quanto ler e escrever.

Em 2013, o então presidente dos Estados Unidos, Barack Obama, demonstrou seu apoio ao ensino de programação ao dizer “Não compre, apenas, um jogo, crie um. Não se limite a fazer download de uma nova aplicação, ajude a desenvolvê-la. Não jogue no seu celular, programe-o.” [15].

No entanto, há também diversos especialistas e educadores contra essa tendência. Segundo [11], muitos educadores acreditam que o ensino de programação prejudica a infância pois retiram delas a oportunidade de socialização com outras crianças através das brincadeiras e jogos próprios dessa fase. Este mesmo texto aponta ainda a entrevista de

Linus Torvalds, responsável pelo kernel do sistema operacional Linux, ao site Businnes Insider onde afirma: “Na verdade, eu não acredito que todos devem necessariamente tentar aprender a programar [...]. Não é como saber ler e escrever e fazer contas básicas”.

No Brasil, o ensino de programação está mais presente nas escolas técnicas, faculdades e escolas especializadas. No entanto começa a despontar em iniciativas públicas e voluntárias como a da Universidade de Passo Fundo, no Rio Grande do Sul, que em 2014 deu início a uma parceria com a prefeitura do município e implantou a Escola de Hackers. Segundo o professor Adriano Canabarro Teixeira, coordenador do projeto [12]

> (...) alguns dos alunos que participaram estavam mais motivados a participar da escola. Verificamos que eles melhoraram muito a interpretação de texto depois que começaram a programar. E o desdobramento mais claro é o raciocínio lógico e matemático. Todos eles têm um ganho nas disciplinas como matemática e física.

Se difundindo pelo país está também o Code Club, projeto fundado em 2012 no Reino Unido com a ideia de se tornar uma rede mundial de clubes gratuitos de programação, reúne voluntários para ensinar programação como atividade extracurricular em escolas, ou outros locais, para crianças de 9 a 12 anos através da criação de jogos, animações e páginas de internet. Trazido para o Brasil por iniciativa do engenheiro e pesquisador Everton Hermann, o Code Club Brasil possibilita um primeiro contato com a programação de computadores trabalhando não apenas a lógica de programação, mas também a criatividade, resolução de problemas e planejamento.

Ensinar os alunos programar já na educação básica não tem sido tratado somente como uma busca de programadores no futuro. Espera-se que os alunos entendam a lógica por trás da linguagem de programação, e com isso sejam capazes de ampliar o pensamento crítico e também usufruam do poder da computação para se tornarem ususários conscientes.

2 ALGORITMOS

2.1 O QUE SÃO OS ALGORITMOS

Com a tecnologia inserida de forma maciça nas nossas vidas, dificilmente encontra-se pessoas que nunca tenham utilizado um computador ou mesmo um celular, seja para pesquisa, edição de textos, redes sociais, jogos ou tarefas específicas exigidas pela profissão. No entanto nem todos já se perguntaram como o computador, ou o celular, executa todas as tarefas que lhes são solicitadas.

Para essa conversa entre o homem e a máquina acontecer é necessária uma linguagem específica, a linguagem de programação. Existem diversas linguagens de programação, e todas elas utilizam uma lógica ao serem escritas. E é aí que entram os algoritmos.

Segundo David Berlinski, o algoritmo é a segunda grande ideia científica do Ocidente, pois possibilitou muitos avanços tecnológicos computacionais. O cálculo é considerada a primeira, resultando na física moderna. Ele diz,

> Um algoritmo é um procedimento eficaz, um modo de fazer uma coisa em um número finito de passos discretos. [...]
>
> No mundo de onde surge a matemática e para o qual o matemático, como nós, deve voltar, um algoritmo, por assim dizer, é um conjunto de regras, uma receita, uma prescrição para a ação, um guia, uma diretiva concatenada e controlada, uma intimidação, um código, um esforço feito para jogar um complexo xale verbal sobre o caos inarticulado da vida. ([2], p. 16)

O algoritmo é definido por Boratti e Oliveira [3] como sendo uma "sequência finita e lógica de instruções executáveis, especificadas em uma determinada linguagem, que mostram como resolver determinado problema".

Observemos que o conceito de algoritmo não é exclusivo da computação. A programação de computadores é apenas um de seus campos de aplicação. Um algoritmo é, em síntese, um método passo a passo para fazer uma determinada tarefa, e um programa de computador é essencialmente um algoritmo que dita ao computador os passos necessários para realizar uma tarefa, seja ela um problema de matemática, informática ou de qualquer outra área.

Um exemplo de algoritmo comumente citado por diversos autores é a receita de bolo. Uma receita nada mais é que uma descrição detalhada, e finita, das etapas que devem ser seguidas para se produzir um bolo. Assim como na execução da receita, o algoritmo é utilizado de forma intuitiva e automática diariamente em tarefas comuns.

Na matemática, ainda nas séries iniciais do Ensino Fundamental, quando aprendemos técnicas para a resolução das quatro operações básicas estamos executando algoritmos,

que com o tempo se tornam automáticos. Com o fácil acesso à informação devido à internet, hoje diversos algoritmos são difundidos pelas redes sociais. Dentre eles o método chinês para multiplicação, que pode ser acessado pelo site

`https://www.youtube.com/watch?v=Q9toSL44XJI`

ou ainda o método Hindu, disponível no site

`https://www.youtube.com/watch?v=ixhGFrtyk-o`

Gilberto Farias aponta em [8] que historiadores divergem quanto à origem da palavra algoritmo. A mais aceita segundo ele é que a palavra deriva do nome do matemático persa do século IX, Mohamed ben Musa Al-Khwarizmi, cujas obras foram traduzidas para o ocidente no século XII, sob o título "Algorithmi de número indorum", um livro de aritmética onde são descritos os cálculos na numeração decimal.

Para a Ciência da computação, o conceito de algoritmo foi formalizado em 1936 por Alan Turing como sendo uma sequência de operações que pode ser simulada por uma máquina de Turing. Segundo Clézio Fonseca Filho

> Em 1936, Turing consagrou-se como um dos maiores matemáticos do seu tempo, quando fez antever aos seus colegas que é possível executar operações computacionais sobre a teoria dos números por meio de uma máquina que tenha embutida as regras de um sistema formal. Turing definiu uma máquina teórica que se tornou um conceito chave dentro da Teoria da Computação. Ele enfatizou desde o início que tais mecanismos podiam ser construídos e sua descoberta acabou abrindo uma nova perspectiva para o esforço de formalizar a matemática, e, ao mesmo tempo, marcou fortemente a História da Computação. ([9], p. 75)

2.2 OS ALGORITMOS NA PROGRAMAÇÃO

Para que um programa desempenhe qualquer tarefa é necessário que ela esteja detalhada passo a passo obedecendo a um algoritmo compreensível pelo computador. Adriano Cruz e Jonas Knopman enumeram em [7] cinco características indispensáveis às aplicações dos algoritmos:

> Finitude: Um algoritmo deve sempre terminar após um número finito de passos.
> Definição: Cada passo de um algoritmo deve ser precisamente definido. As ações devem ser definidas rigorosamente e sem ambiguidades.
> Entradas: Um algoritmo deve ter zero ou mais entradas, isto é,

> informações que são lhe são fornecidas antes do algoritmo iniciar.
> Saídas: Um algoritmo deve ter uma ou mais saídas, isto é quantidades que tem uma relação específica com as entradas.
> Efetividade: Um algoritmo deve ser efetivo. Isto significa que todas as operações devem ser suficientemente básicas de modo que possam ser em princípio executadas com precisão em um tempo finito por um humano usando papel e lápis.

Existem diversas formas de representar um algoritmo, e elas se diferem pela quantidade de detalhes de implementação. Dentre as principais formas tem-se a descrição narrativa, o fluxograma convencional e o pseudocódigo (ou português estruturado).

Na descrição narrativa os algoritmos são expressos diretamente em linguagem natural. Sua vantagem é que não há necessidade de aprender novas técnicas e conceitos, no entanto pode ser imprecisa e gerar ambiguidade na interpretação.

Como exemplo de um algoritmo escrito nessa forma tem-se o passo a passo para fazer um bolo:

Receita de bolo

Misture os ingredientes

Unte a forma com manteiga

Despeje a mistura na forma

Leve a forma ao forno

Enquanto não corar

deixe a forma no forno

Retire do forno

Deixe esfriar

Se houver cobertura

então despeje sobre o bolo

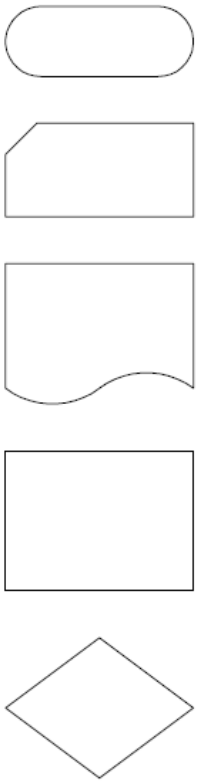
Note que o algoritmo acima, na forma narrativa, é impreciso e esconde detalhes. Ele poderia detalhar melhor quais ingredientes usar e em qual sequência eles devem ser misturados para não gerar resultados distintas na execução.

Já o fluxograma convencional é uma representação gráfica que faz uso de símbolos geométricos para representar as estruturas de um algoritmo. Cada instrução possui um símbolo específico, conforme a Figura 1, e o comando escrito dentro dele deve ser claro. Os símbolos são conectados por um caminho orientado que fornece a sequência de

execução, e esse caminho é único. Ou seja, ao representar um algoritmo usando fluxograma convencional, entre o símbolo inicial e o final há um único caminho orientado a ser seguido, representando a existência de uma única seqüência de execução das instruções. Assim, apesar do símbolo de decisão possuir mais de um possível caminho de saída (geralmente dois), apenas um deles é seguido.

Esta forma não é tão imprecisa quanto a descrição narrativa, no entanto não se preocupa com detalhes de implementação do programa tais como o tipo das variáveis usadas.

Figura 1 – Principais formas geométricas usadas em fluxogramas.



Início e final do fluxograma

Operação de entrada de dados

Operação de saída de dados

Operação de atribuição

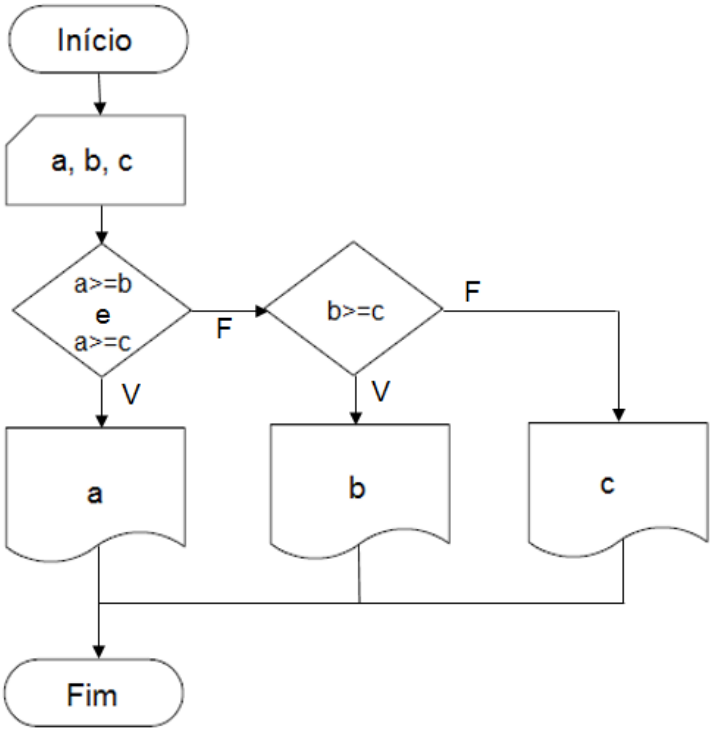
Decisão

Um exemplo de algoritmo que reconhece qual o maior dentre três números naturais, escrito na forma de fluxograma convencional esta representado na Figura 2.

O terceiro modo para se representar um algoritmo, o pseudocódigo (ou português estruturado) é o que mais se aproxima da definição de algoritmos como é conhecido. É uma maneira intermediária entre a descrição narrativa e uma linguagem de programação, e é escrito na linguagem do programador.

No desenvolvimento de um algoritmo, o uso do pseucodódigo permite que o foco seja a lógica em si, e não a sintaxe para representação em determinada linguagem de programação. Mas apesar disso, já apresenta conceitos como declaração de variáveis, linhas de códigos e palavras-chave, o que torna mais fácil a transição para uma linguagem.

Figura 2 – Exemplo de algoritmo na forma de fluxograma convencional.



A este respeito, Fabiano dos Santos diz

> O português estruturado é uma simplificação da nossa linguagem natural na qual usamos frases simples e estruturas que possuem um significado muito bem definido. Apesar disso, a aparência do português estruturado é muito semelhante a uma linguagem de programação tradicional. Possui um conjunto de palavras e regras que formam as sentenças válidas e compõem a sintaxe da linguagem. ([17], p. 41)

A maneira geral de representar um algoritmo na forma de pseudocódigo é:

Algoritmo <nome-do-algoritmo>
    Var <declaração-de-variáveis>
Início
    <corpo-do-algoritmo>
Fimalgoritmo

Algoritmo é uma palavra que indica o início da definição de um algoritmo em forma de pseudocódigo. Ela vem seguida do **<nome-do-algoritmo>** que é um nome simbólico dado ao algoritmo com a finalidade de distingui-los dos demais, informando sua funcionalidade.

A **Var <declaração-de-variáveis>** é onde são declaradas as variáveis[1] usadas no algoritmo. E, limitando o <corpo-do-algoritmo>, que é o conjunto de instruções a serem seguidas, estão **Início** e **Fimalgoritmo**.

Como exemplo de um algoritmo em pseudocódigo, temos o cálculo do quadrado de um número real qualquer:

Algoritmo "Quadrado de um número"
    Var numero, quadrado: **real**
Início
    escreva ("Digite um número real: ")
    leia (numero)
    quadrado ← numero ^ 2
    escreva ("O quadrado do número digitado é: "quadrado)
Fimalgoritmo

[1] A declaração de variáveis será detalhada na seção 3.1

3 VISUALG

Criado pelo programador/analista e professor universitário Cláudio Morgado de Souza, o VisuAlg permite criar, editar, interpretar e também executar os algoritmos. Segundo ele, o programa nasceu da necessidade de uma ferramenta para os alunos iniciantes em programação exercitarem seus conhecimentos num ambiente próximo da realidade de uma linguagem de programação [20].

Alguns fatores foram primordiais na sua escolha para o presente trabalho, dentre eles o fato de que o VisuAlg é um programa de livre uso e distribuição grátis, usado em várias escolas e universidades no Brasil. Ainda, sua linguagem é amplamente utilizada nos livros de introdução à programação e não é necessário instalar o programa, basta baixar e usar. Inclusive é possível executá-lo através de um pen-drive, fator de grande relevância pois, apesar de o laboratório de informática estar presente em muitas escolas, eles geralmente são formados por computadores mais antigos ou com pouca capacidade de armazenamento e baixo desempenho.

O VisuAlg pode ser executado sobre o sistema operacional Windows 95 ou posterior. Para a utilização em Linux, uma saída simples é usar o programa Wine, que simula uma plataforma Windows no seu computador e permite ao usuário instalar e executar vários programas para o Windows.

O VisuAlg na versão 3.0.6.3 pode ser obtido pelo site

`https://sourceforge.net/projects/visualg30/files/latest/download`

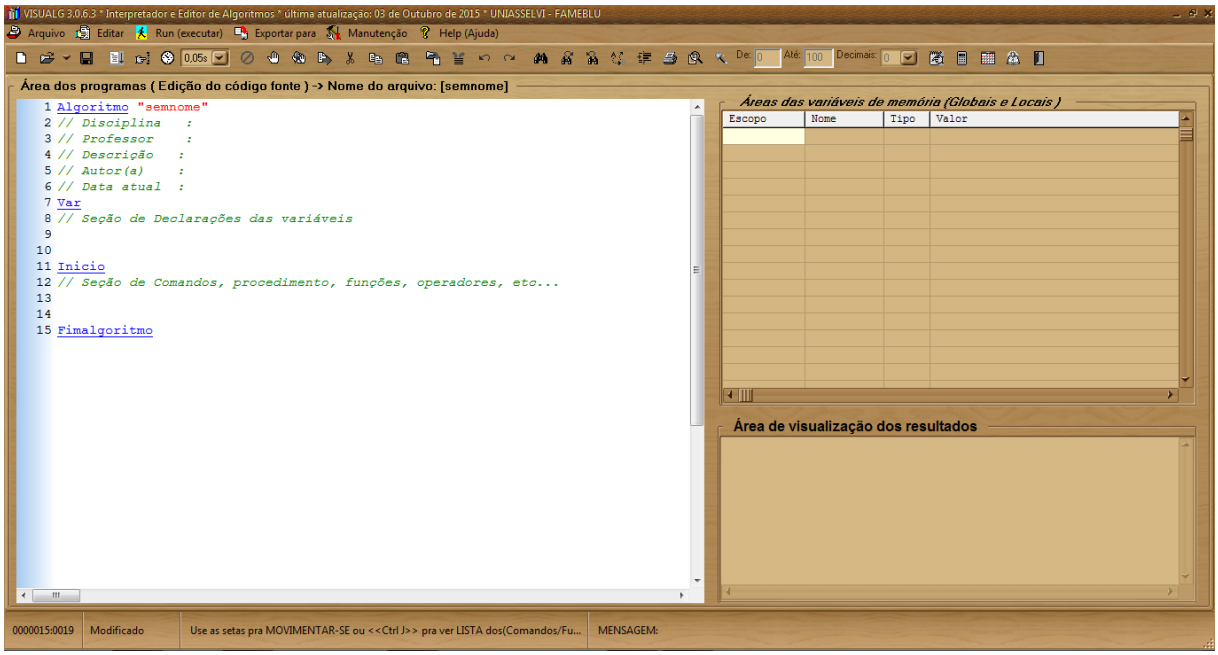
e o Wine versão 1.7.34 está disponível para download no site

`http://wine.soft32.com/old-version/2987277/1.7.34/`

Com sua interface simples e totalmente em português, o VisuAlg pode ser inserido logo na primeira aula facilitando, assim, o entendimento de como um programa de computador funciona e possibilitando um retorno imediato sobre a correção e exatidão do pseudocódigo digitado, além de instigar o aluno a experimentar e ver o resultado de suas alterações imediatamente.

A tela do VisuAlg consiste em barra de menus e barra de botões que, localizada na parte superior, contém os comandos mais utilizados; área de programas, que é o editor de textos em si e já apresenta uma estrutura de pseudocódigo, com a intenção de poupar o trabalho do usuário e mostrar o formato básico que deve ser seguido; área de variáveis, localizada do lado direito, na metade superior, traz detalhes sobre as variáveis declaradas; e a área de visualização de resultados, logo abaixo da área de variáveis, que mostra o

Figura 3 – Interface do VisuAlg



Fonte: Print screen da Interface do VisuAlg no Sistema Operacional Windows 7

resultado após a execução do código. Há ainda uma área ou tela, que simula o modo texto do MS-DOS e mostra a execução do algoritmo.

3.1 DECLARAÇÃO DE VARIÁVEIS NO VISUALG.

3.1.1 DADOS

Para que seja possível armazenar e manipular dados no computador é necessário representá-los internamente de alguma forma. A representação correta e adequada de um tipo de informação permite otimizar os recursos computacionais disponíveis, além de acelerar o processamento.

Dentro de um algoritmo podemos encontrar basicamente duas classes diferentes de dados, os dados *constantes* e os *variáveis*. Um dado é dito *constante* quando seu valor não se altera ao longo do tempo em que o algoritmo é executado, ou seja, permanece o mesmo desde o início até o final da execução. Já um dado que pode ter seu valor alterado durante a execução do programa é tido como uma *variável*.

Por exemplo, no cálculo do comprimento de uma circunferência, $C = 2\pi r$, o valor de π é sempre igual a 3,1415...., ou seja, uma constante. Já o raio pode assumir direfentes valores reais dependendo da circunferência, assim ele é uma variável.

O VisuAlg prevê quatro tipos de dados variáveis: inteiro, real, cadeia de caracteres e lógico (ou booleano).

- **Inteiro:** São os números pertencentes ao conjunto dos Números Inteiros.
- **Real:** São os números pertencentes ao conjunto dos Números Reais.
- **Cadeia de caracteres:** São os valores pertencentes ao conjunto de todos os caracteres numéricos (0, 1, ..., 9), alfabéticos (a/A, b/B, ..., z/Z) e especiais (!, @,...,*) . Esse conjunto também é conhecido como conjunto de caracteres alfanuméricos.
- **Lógico (booleano):** O tipo lógico é utilizado para representar informações que só podem assumir dois valores, o valor verdadeiro (V) ou o valor falso (F).

Ele permite também a declaração de variáveis estruturadas através da palavra-chave vetor, no entanto este tipo de variável não será abordado neste trabalho.

Há, ainda, três tipos de constantes:

- **Numéricas:** são valores numéricos, inteiros ou reais, escritos na forma usual. Ressaltando que o separador de decimais é o ponto e não a vírgula, independente da configuração regional do computador onde o VisuAlg está sendo executado. O VisuAlg também não suporta separadores de milhares.
- **Caracteres:** qualquer cadeia de caracteres delimitada por aspas duplas (").
- **Lógicos:** admite os valores verdadeiro ou falso.

3.1.2 DECLARANDO VARIÁVEIS

Declarar uma variável significa reservar uma região (endereço) da memória, com a finalidade de armazenar os dados de um programa por um determinado período de tempo. Essa região é identificada por meio de um endereço no formato hexadecimal, e o VisuAlg, assim como as demais linguagens de programação, permite nomear cada um deles para facilitar sua referência. Para isso são necessários dois elementos: o conteúdo deste identificador, que será o tipo de dado; e o seu nome, usado para acessar o dado e realizar operações com o mesmo.

Para identificar o tipo de conteúdo do dado, algumas palavras-chaves são usadas: `inteiro`, `real`, `caractere` e `logico`. Como nomes atribuídos às variáveis devem ser usadas palavras iniciadas por uma letra, e, a partir do segundo caractere, pode conter letras, números e underline até um limite de 30 caracteres, sendo que não há distinção de maiúsculas e minúsculas e as palavras não devem ser acentuadas. Não pode haver duas variáveis com o mesmo nome e não se pode usar palavras reservadas pelo VisuAlg.

A seção de declaração de variáveis começa com a palavra-chave `var`, e possui a seguinte sintaxe:

```
var
<lista de variáveis separadas por vírgulas>:  <tipo-de-dado>
```

Por exemplo, em um algoritmo que necessite das informações pessoais de um aluno, como nome, sexo, matrícula, idade e nota final, teremos:

```
Algoritmo "Informações pessoais"
Var
      nome, sexo:  caractere
      matricula, idade:  inteiro
      nota_final:  real
Inicio
      // Seção de Comandos, procedimento, funções, operadores, etc...
Fimalgoritmo
```

3.2 COMANDOS DE ENTRADA E SAÍDA DE DADOS.

Os comandos de entrada e saída são aqueles que definem como serão fornecidos valores para o sistema e como estes serão impressos para visualização por parte do usuário. Isto é, comandos de entrada são aqueles que permitem ao programador coletar dados a partir de dispositivos de entrada (geralmente o teclado) ou de armazenamento (tais como pendrive e disco rígido, e os comandos de saída são aqueles que permitem ao programador enviar dados para dispositivos de saída de dados (o monitor e a impressora, por exemplo) ou de armazenamento.

Como comandos de entrada e saída do VisuAlg, temos o **escreva**, **escreval** e o **leia**.

O comando **escreva** é usado para exibir algum tipo de informação na tela do computador. Sua sintaxe é **escreva (<lista de expressões>)**, dessa forma, se quisermos imprimir uma mensagem como por exemplo “O aluno está matriculado”, isto pode ser feito com a algoritmo

```
Algoritmo "Mensagem"
Var
      // Seção para declaração de variáveis
Inicio
      escreva ("O aluno está matriculado")
Fimalgoritmo
```

As aspas servem para delimitar uma sequência de caracteres, uma constante, mas não serão exibidas na execução do programa. Para exibir o valor de uma variável, deve-se colocar o seu identificador (seu nome) diretamente, sem aspas, separando-o da parte constante, se houver, por uma vírgula.

O comando **escreval** difere do **escreva** apenas pelo fato de saltar para a linha seguinte assim que a informação é exibida.

Já o **leia** é usado para receber informações de fora do algoritmo, ou seja, recebe e armazena informações digitadas pelo usuário. Sua sintaxe é **leia (<lista de variáveis>)**, em que **<lista de variáveis>** é o nome atribuído na declaração de variáveis.

```
Algoritmo "Nome"
Var
      nome:  caractere
Inicio
      escreva ("Digite o nome do aluno:  ")
      leia (nome)
Fimalgoritmo
```

3.3 ATRIBUIÇÃO.

Nem todos os dados de um programa são digitados pelo usuário. Por exemplo, ao calcular a área de um retângulo, as medidas da base e da altura podem ser fornecidas pelo usuário, mas a área, geralmente, é calculada atribuindo-lhe o resultado do produto da base pela altura desse retângulo.

A atribuição de valores às variáveis é feita com o operador **<-**. Sua sintaxe é **<identificador da variável> <- <expressão/valor cujo resultado é do mesmo tipo da variável>**. Ou seja, do seu lado esquerdo fica a variável à qual está sendo atribuído o valor, e à sua direita pode-se colocar qualquer expressão (constantes, variáveis, expressões), desde que seu resultado tenha tipo de dado igual ao da variável.

É importante ressaltar que uma varíavel pode armazenar apenas um único valor por vez, sendo que sempre que um novo valor é atribuído à variável, o valor anterior que estava armazenado na mesma é perdido.

Ainda, ao representar as operações matemáticas, o sinal ***** indica a multiplicação, **/** indica a divisão, e os sinais de **+** e **-** indicam a adição e subtração, respectivamente.

```
Algoritmo "Área do retângulo"
Var
      base, altura, area:  real
Inicio
      escreva ("Digite a base:  ")
      leia (base)
      escreva ("Digite a altura:  ")
      leia (altura)
      area<-base*altura
      escreva ("A área é:",area)
Fimalgoritmo
```

3.4 FUNÇÕES DO VISUALG

Toda linguagem de programação já vem com um grupo de funções pré-definidas que facilitam a programação. Estas funções realizam os cálculos aritméticos, trigonométricos e de manipulação e conversão de dados mais comuns. A este grupo de funções dá-se o nome de biblioteca. No presente trabalho não serão abordadas todas as funções disponíveis no VisuAlg, restringiremos às funções matemáticas que possam ser facilmente utilizadas por alunos do Ensino Médio bem como alunos do Ensino Fundamental.

- **Pi** - Retorna o valor 3.141592.
- **Quad(expressão)** - Retorna quadrado do valor representado por expressão.
- **RaizQ(expressão)** - Retorna a raiz quadrada do valor representado por expressão.
- **Abs(expressão)** - Retorna o valor absoluto de uma expressão do tipo inteiro ou real. Equivale a | expressão | na álgebra.
- **Cos(expressão**) - Retorna o co-seno do ângulo (em radianos) representado por expressão.
- **Sen(expressão**) - Retorna o seno do ângulo (em radianos) representado por expressão.
- **Tan(expressão**) - Retorna a tangente do ângulo (em radianos) representado por expressão.
- **Exp(base, expoente**) - Retorna o valor de base elevado a expoente, sendo ambos expressões do tipo real.
- **Int(expressão)** - Retorna a parte inteira do valor representado por expressão.

- **x mod y** - Retorna o resto da divisão de x por y.

Para efeito de declaração de variáveis, deve-se notar que enquanto as funções **Int(expressão)** e **x mod y** sempre retornam valores inteiros, as demais funções podem ter como saída números reais.

3.5 ESTRUTURA CONDICIONAL.

De forma geral, a execução de um algoritmo começa na primeira linha e vai avançando sequencialmente executando o código linha após linha até chegar ao final. Entretanto, algumas vezes surge a necessidade de colocar instruções dentro de um programa que só serão executadas caso alguma condição específica aconteça. Para esta finalidade as linguagens de programação possuem estruturas de condição. As estruturas condicionais do VisuAlg são **se-entao** e **se-entao-senao**.

Para compor as estruturas de condição geralmente são usados operadores relacionais **=** (igual), **>** (maior), **<** (menor), **>=** (maior ou igual), **<=** (menor ou igual) e **< >** (diferente).

3.5.1 Comando se-entao

É a estrutura condicional mais simples. Sua sintaxe é:

```
se <expressão lógica> entao
      <sequência de comandos>
fimse
```

em que `<expressão lógica>` é uma expressão que deverá ser avaliada como verdadeira (V) ou falsa (F), e caso o resultado dessa expressão for verdadeiro, será executado o bloco de comandos, `<sequência de comandos>`, que está dentro da estrutura. Caso seja falso, a execução do programa ignora todo o bloco de comandos e continua na linha seguinte à estutura de condição, após o **fimse**.

Por exemplo, é comum o comércio oferecer desconto aos clientes nas compras acima de um determinado valor pré-estabelecido. Assim, se a compra for maior que esse valor então o cliente ganha desconto. A condição nesta frase é "a compra é maior que um determinado valor (digamos R$100,00)". Ela é uma expressão lógica, pois deve ser respondida com "Sim"ou "Não". Se a condição for verdadeira (sim), a ação a ser executada é "ganha desconto (digamos 20%)", caso contrário toda a estrutura é ignorada.

Uma maneira de escrever um algoritmo com essa finalidade seria:

```
Algoritmo "Desconto"
```

```
Var
      preco, pagar:  real
Inicio
      escreva ("Digite o valor total:  ")
      leia(preco)
      pagar<-preco
      se preco>=100 entao
            pagar<-preco*0.8
      fimse
      escreva ("O valor total a pagar é:",pagar)
Fimalgoritmo
```

3.5.2 Comando se-entao-senao

Suponha, no exemplo anterior, que além do desconto de 20% dado para compras à partir de R$100,00, seja oferecido um desconto de 5% para compras abaixo de R$100,00. Verificam-se aqui duas condições para desconto, e um possível algoritmo para essa situação é:

```
Algoritmo "Desconto"
Var
      preco, pagar:  real
Inicio
      escreva ("Digite o preço:  ")
      leia(preco)
      se preco>=100 entao
            pagar<-preco*0.8
      fimse
      se preco<100 entao
            pagar<-preco*0.9
      fimse
      escreva ("O valor a pagar é:",pagar)
Fimalgoritmo
```

Observa-se que sempre que a condição do primeiro comando for verdadeira, a segunda condição será falsa e vice-versa. Para expressar essa situação mais facilmente, e também de forma mais eficaz, as linguagens de programação permitem associar um conjunto de instruções a ser executado se a condição do comando resultar em falso, evitando assim que uma segunda estrutura condicional se faça necessária.

No VisuAlg, o comando utilizado nessa situação é o **se-entao-senao**. Nessa estrutura a expressão lógica é analisada e, caso seja verdadeira, a primeira sequência de comandos é executada. Mas se for falsa, o VisuAlg salta para o bloco de comandos disposto no **senao**.

Sua sintaxe é:

```
se <expressão lógica> entao
      <seqüência de comandos caso a expressão lógica seja verdadeira>
senao
      <seqüência de comandos caso a expressão lógica seja falsa>
fimse
```

Logo, um possível algoritmo que calcula o desconto é:

```
Algoritmo "Desconto"
Var
      preco, pagar:  real
Inicio
      escreva ("Digite o preço:  ")
      leia(preco)
      se preco>=100 entao
            pagar<-preco*0.8
      senao
            pagar<-preco*0.9
      fimse
      escreva ("O valor a pagar é:",pagar)
Fimalgoritmo
```

3.5.3 Expressões Lógicas Compostas.

É possível compor expressões lógicas utilizando os operadores lógicos **e** (conjunção), **ou** (disjunção) ou **nao** (negação).

- Uma expressão com o operador **e** é verdadeira, se todas as condições forem verdadeiras.
- Uma expressão com o operador **ou** é verdadeira, se pelo menos uma das condições for verdadeira.

- Uma expressão com o operador **nao**, inverte o valor da expressão ou condição, se verdadeira inverte para falsa e vice-versa.

Por exemplo, considere a situação de um determinado aluno em uma disciplina. Para ser aprovado, é necessário que sua média seja no mínimo 6,0 e, ao mesmo tempo, que sua frequência seja de no mínimo 75%, isto é, uma conjunção lógica representada pelo operador **e**, caso uma das condições seja falsa ele será reprovado. Em algoritmo:

```
Algoritmo "Situação do aluno"
Var
      nome:  caractere
      media, frequencia:  real
Inicio
      escreva ("Digite o nome do aluno:  ")
      leia(nome)
      escreva ("Digite a média do aluno (0 a 10):  ")
      leia(media)
      escreva ("Digite frequência, em porcentagem (0 a 100):  ")
      leia(frequencia)
      se (media>=6) e (frequencia>=75) entao
            escreva ("O(a) aluno(a)",nome," foi aprovado(a)")
      senao
            escreva ("O(a) aluno(a)",nome," foi reprovado")
      fimse
Fimalgoritmo
```

3.5.4 Estruturas de Condição Encadeadas.

Dentro de uma estrutura **se-entao-senao** é perfeitamente possível utilizar mais de uma linha de comando, ou até mesmo outras estruturas **se-entao-senao**. Existem situações em que os caminhos para a tomada de uma decisão acabam formando uma espécie de árvore com diversas ramificações, onde cada caminho é um conjunto de ações. Nesses casos podemos recorrer à utilização de várias dessas estruturas embutidas umas dentro das outras, comumente chamadas de ninhos.

Por exemplo, se o aluno reprovado na situação exposta em 3.5.3 obteve uma média acima de 3,0, independente de sua frequência, ele terá direito à fazer avaliação para recuperação. Assim, caso a expressão lógica `(media>=6) e (frequencia>=75)` seja falsa, haverá ainda duas situações, "Recuperação", caso sua média seja maior ou igual à 3,0, `media>=3`, e "Reprovado" caso sua média não seja. Assim, temos:

```
Algoritmo "Situação do aluno"
Var
      nome:  caractere
      media, frequencia:  real
Inicio
      escreva ("Digite o nome do aluno:  ")
      leia(nome)
      escreva ("Digite a média do aluno (0 a 10):  ")
      leia(media)
      escreva ("Digite frequência, em porcentagem (0 a 100):  ")
      leia(frequencia)
      se (media>=6) e (frequencia>=75) entao
            escreva ("O(a) aluno(a)",nome," foi aprovado(a)")
      senao
            se media>=3 entao
                  escreva("O(a) aluno(a)",nome," está de recuperação")
            senao
                  escreva ("O(a) aluno(a)",nome," foi reprovado")
            fimse
      fimse
Fimalgoritmo
```

3.6 ESTRUTURAS DE REPETIÇÃO

As estruturas de repetição são comandos que repetem um conjunto de procedimentos, sob determinadas condições, quantas vezes for necessário, até que determinado objetivo seja atingido, quando a repetição se encerra.

Elas são úteis, por exemplo, para repetir uma série de operações semelhantes que são executadas para todos os elementos de uma lista ou de uma tabela de dados, ou simplesmente para repetir um mesmo processamento até que uma certa condição seja satisfeita.

Todas as estruturas de repetição têm em comum o fato de haver uma condição de controle, expressa através de uma expressão lógica, que é testada em cada ciclo para determinar se a repetição prossegue ou não.

As estruturas de repetição do VisuAlg são três: **enquanto-faca**, **repita-ate** e **para-ate-faca**.

3.6.1 Comando enquanto-faca

Esta estrutura repete uma sequência de comandos enquanto uma determinada condição, especificada através de uma expressão lógica, for satisfeita. Sua sintaxe é

```
enquanto <expressão lógica> faca
      <sequência de comandos>
fimenquanto
```

De modo geral, antes de entrar na estrutura de repetição, uma `<expressão-lógica>` é avaliada e, caso o resultado da mesma seja verdadeiro, os comandos que estão dentro da estrutura, `<sequência de comandos>`, serão executados. Após a execução dos comandos, a `<expressão lógica>` é novamente avaliada. Caso o resultado seja verdadeiro, os comandos que estão dentro da estrutura são novamente executados. Esse processo de avaliar a `<expressão-lógica>`, executar a `<sequência de comandos>` e retornar para avaliar a `<expressão-lógica>` se repete até que se tenha um resultado falso para a `<expressão-lógica>`. Quando isso acontece, o algoritmo sai da estrutura de repetição e segue para a próxima linha.

Nessa estrutura, o mecanismo que altera o valor da `<expressão lógica>` que controla o laço está embutido dentro do bloco `<sequência de comandos>` ou depende de alguma variável externa que será fornecida na execução.

Essa estrutura é usada principalmente quando não se sabe com antecedência a quantidade de repetições que precisam ser realizadas.

Observa-se que em nenhum dos exemplos citados neste capículo houve a preocupação em verificar a compatibilidade dos dados digitados pelo usuário com o tipo de variável a que está sendo atribuído o valor. Quando em 3.5.3 é solicitado ao usuário que digite a média, entre 0 e 10, e a frequência, entre 0 e 100, não há, também, nenhuma estrutura que se certifique que o número fornecido está realmente dentro dos intervalos determinado. Um exemplo de uso para o comando **enquanto-faca** é essa verificação. Para 3.5.3, o comando é inserido após a leitura do dado digitado para verificar se o dado se encontra dentro do intervalo. Enquanto não atender às condições de intervalo um novo valor será solicitado.

```
Algoritmo "Situação do aluno"
Var
      nome:  caractere
      media, frequencia:  real
Inicio
      escreva ("Digite o nome do aluno:  ")
```

```
      leia(nome)
      escreva ("Digite a média do aluno (0 a 10):  ")
      leia(media)
      enquanto (media<0) ou (media>10) faca
            escreva ("A média deve ser um número entre 0 e 10:  ")
            leia(media)
      fimenquanto
      escreva ("Digite frequência, em porcentagem (0 a 100):  ")
      leia(frequencia)
      enquanto (frequencia<0) ou (frequencia>100) faca
            escreva ("A frequencia do aluno deve ser um número entre 0 e
            100:  ")
            leia(frequencia)
      fimenquanto
      se (media>=6) e (frequencia>=75) entao
            escreva ("O(a) aluno(a)",nome," foi aprovado(a)")
      senao
            se media>=3 entao
                  escreva("O(a) aluno(a)",nome," está de recuperação")
            senao
                  escreva ("O(a) aluno(a)",nome," foi reprovado")
            fimse
      fimse
Fimalgoritmo
```

3.6.2 Comando repita-ate

Semelhante ao comando **enquanto-faca**, com a diferença que este, primeiro testa a condição, para depois realizar o bloco de comando, ao passo que **repita-ate** que primeiro executa o bloco para depois realizar o teste. Assim, utilizando o **repita-ate** o bloco <sequência de comandos> será sempre executado pelo menos uma vez, mesmo que a <expressão lógica> seja falsa. Sua sintaxe é

```
repita
      <seqüência de comandos>
ate <expressão lógica>
```

Podemos suprimir algumas linhas de comando no exemplo da seção 3.6.1 usando o comando **repita-ate** da seguinte maneira:

```
Algoritmo "Situação do aluno"
Var
      nome:  caractere
      media, frequencia:  real
Inicio
      escreva ("Digite o nome do aluno:  ")
      leia(nome)
      repita
            escreva ("Digite a média do aluno (entre 0 e 10):  ")
            leia(media)
      ate (media>=0) e (media<=10)
      repita
            escreva ("Digite a frequência do aluno (entre 0 e 100):  ")
            leia(frequencia)
      ate (frequencia>=0) e (frequencia<=100)
      se (media>=6) e (frequencia>=75) entao
            escreva ("O(a) aluno(a)",nome," foi aprovado(a)")
      senao
            se media>=3 entao
                  escreva("O(a) aluno(a)",nome," está de recuperação")
            senao
                  escreva ("O(a) aluno(a)",nome," foi reprovado")
            fimse
      fimse
Fimalgoritmo
```

3.6.3 Comando para-ate-faca

A estrutura **para-ate-faça** tem embutido um mecanismo de controle para determinar de antemão quando o laço deverá ser terminado. Sua sintaxe é:

```
para <variável> de <valor-inicial> ate <valor-limite> passo <incremento>
faca
      <sequência de comandos>
fimpara
```

Sendo que `<valor-inicial >`, `<valor-limite >` e `<incremento>` são avaliados uma única vez antes da execução da primeira repetição, e não se alteram durante a

execução do laço, mesmo que variáveis eventualmente presentes nessas expressões tenham seus valores alterados. Ainda, **passo** <incremento> especifica o valor adicionado à variável e, caso seja 1, não precisa estar indicado.

Por exemplo, para um algoritmo que escreva os dez primeiros múltiplos não negativos de um número fornecido pelo usuário:

```
algoritmo "Múltiplos não negativos de um número"
var
      contar, numero:  inteiro
inicio
      escreva("Digite um número:  ")
      leia(numero)
      para contar de 0 ate 10 faca
            escreva (contar*abs(numero))
      fimpara
fimalgoritmo
```

3.6.4 Contadores e Acumuladores

Acumulador, ou *somador*, é o nome dado à uma variável que atua acumulando os valores a cada vez que a <sequência de comandos> é executada. Por exemplo, um algoritmo que soma os valores das compras em um caixa de supermercado atua acumulando numa variável o total de todas as compras. Essa implementação é feita fazendo com que essa variável receba o seu próprio valor, que inicialmente é zero, adicionado o valor de cada novo produto.

Contador é o nome dado à uma variável que acumula seu próprio valor, acrescentando uma quantidade fixa, geralmente 1, à cada execução da <sequência de comandos>. Por exemplo, o algoritmo mencionado acima, que soma os valores das compras do caixa de supermercado, pode também calcular o número total de itens comprados pelo cliente. Essa implementação é feita fazendo com que essa variável receba seu próprio valor, que inicialmente deve ser zero, adicionado à 1 a cada item que passa pelo caixa.

Temos, como exemplo, o algoritmo:

```
Algoritmo "Compras no Supermercado"
Var
      valor, total:real
      quant, cont:  inteiro
Inicio
```

```
      total<-0
      cont<-0
      repita
            escreva("(Para finalizar, digite 0)")
            escreva("Valor unitário da mercadoria:  ")
            leia(valor)
            se valor<>0 entao
                  escreva("Quantidade de itens:  ")
                  leia(quant)
                  total<-total+(valor*quant)
                  cont<-cont+quant
            fimse
      ate valor=0
      escreva(cont," itens, totalizando R$",total)
Fimalgoritmo
```

4 PROPOSTA DE ATIVIDADES

É comum, no ensino de algoritmo, iniciar alunos apenas de forma teórica, sem a implementação do código em um programa de validação, ou seja, ele escreve e executa o algoritmo apenas no papel. Apesar de importante, é um grande obstáculo (quase intransponível para alguns) no aprendizado das técnicas de elaboração de algoritmos, como afirma Cláudio Souza. Segundo ele,

> Um ponto positivo da abordagem tradicional é a possibilidade de se usar apenas lápis e papel para se codificar um algoritmo, sem necessidade de computadores, interpretadores ou compiladores. Entretanto, para a maioria dos alunos esta estratégia se revela muito abstrata, pois não conseguem associar, por exemplo, a execução dos comandos escreva("Digite um valor:") e leia(valor) à exibição de um prompt na tela do computador e ao cursor esperando a entrada de dados, nem perceber como o fluxo de processamento ocorre na execução de comandos de decisão e iteração. A isto se soma o conhecido fato de que o aluno ainda enfrenta o obstáculo de entender o problema, criar por si uma solução ou compreender aquela apresentada por seus colegas ou pelo professor, isto é, a lógica de programação em si. ([19], p. 2)

A sequência de atividades, sugeridas abaixo, envolvem a construção de algoritmos, em situações que contemplam os conteúdos de matemática básica do ensino fundamental e sua implementação em um programa de validação. A construção e execução se darão diretamente no editor de texto do VisuAlg para que o aluno tenha a experiência mais próxima daquilo que um programador encontra em seu trabalho.

Para construir cada algoritmo, o aluno deve conseguir organizar seu pensamento matemático buscando estratégias para resolução de situações problemas, estruturá-lo, inicialmente com auxílio do professor, no formato exigido pelo VisuAlg, além de identificar os possíveis erros de construção buscando possíveis soluções.

As atividades são propostas de forma a trabalhar alguns dos comando básicos, cumulativamente, na seguinte ordem:

- **Declaração de Variáveis:** Declarar uma variável é uma etapa que se fará presente em todos os algoritmos a serem criados pelos alunos. Eles devem conseguir identificar corretamente que tipos de dados serão necessários e nomeá-los apropriadamente. As atividades 1 e 2 propostas são apenas para familiarizarização com esses tipos de dados e também com a estrutura de formatação do código.
- **Comandos de entrada, saída e atribuição:** Conhecendo a formatação e a declaração de variáveis, as atividades 3, 4 e 5 são propostas simples para que os

alunos comecem a escrever o algoritmo em si (enviar e receber informações do usuário, efetuar e registrar cálculos simples e repassar ao usuário alguma informação como saída). Além resgatar conteúdos básicos da matemática, essas atividades têm a finalidade de fazer com que o aluno compreenda as etapas fundamentais de um algoritmo (Entrada - Processamento - Saída).

- **Funções do VisuAlg:** Na elaboração das atividades 6, 7 e 8 serão abordadas algumas das funções pré-definidas pelo VisuAlg. Até esse momento o aluno não tem como se precaver quanto a possíveis erros que possam vir a ser cometidos por usuários, como inserir um inteiro negativo onde caberia um não negativo por exemplo.

- **Estrutura condicional:** As atividades 9 à 13 têm como elemento principal a tomada de decisão. Os alunos devem identificar sobre que condição os comandos devem ser executados e então definir qual a estrutura condicional apropriada. Entram, a partir daí, etapas antes ignoradas, como a verificação da validade de um número inserido pelo usuário.

- **Estrutura de repetição:** O aluno deve identificar que algumas operações têm a necessidade de serem repetidas um certo número de vezes ou até que alguma condição esteja satisfeita. Para isso, as atividades 14 à 18 trazem algumas propostas de situações em que essa repetição se fará necessária. Também são inseridas aqui a noção de variáveis contadoras e acumuladoras.

A cada nova atividade é interessante uma discussão entre os alunos e professor a respeito da proposta matemática e o estudo da sintaxe do comando, juntamente com um exemplo para facilitar a compreensão.

É muito importante, em todas as atividade, que o aluno faça uso correto da precedência dos operadores matemáticos, para evitar possíveis erros e também parênteses em excesso: primeiro são resolvidas as funções do VisuAlg, seguidas da exponenciação (ˆ), da multiplicação (*) e divisão (/) na ordem em que aparecerem, e finalizando com a adição (+) e subtração (-) também na ordem em que aparecerem, respeitando-se a prioridade das operações no interior dos parênteses.

As sugestões de resolução para cada uma das atividades abaixo está disponível para download, em formato próprio do Visual, no link:

`https://drive.google.com/drive/folders/0B34qug9LVg0wdDRkLUN5SnZKcEU?usp=sharing`

4.1 ATIVIDADE 1

Situação: No dia a dia, quando um motorista põe seu veículo em um estacionamento rotativo pago e se dirige à uma portaria para regularizar sua situação, algumas informações são registradas, como a placa do veículo, tipo de veículo (carro, motocicleta, etc.), tempo de permanência, dentre outros. Ao sair, o valor a ser pago é calculado baseado nesse tempo de permanência no estacionamento.

Proposta: A proposta para essa atividade é identificar que informações o algoritmo irá receber e qual será seu tipo. A ideia é discutir com os alunos que informações costumam ser coletadas e de que tipo elas são. Se forem numéricas verificar se basta ser do tipo **`inteiro`** ou se há a necessidade de ser **`real`**.

Sugestão: Como o VisuAlg não possui um tipo de dado específico para armazenamento de hora, nesta atividade podemos limitar ao tempo de permanência como sendo um número inteiro positivo fornecido pelo usuário do sistema, assim, um dado do tipo **`inteiro`**. Podemos declarar também variáveis do tipo **`caractere`** para o tipo de veículo (carro, motocicleta, etc.) e a placa do veículo, já que a mesma consta de três letras e quatro digitos, separados pelo caracter especial **`-`** , e também uma variável para armazenar o valor a ser pago de acordo com a permanência, esta deve ser do tipo **`real`**, já que o valor pode ser um decimal, por exemplo R$10,55.

Desta forma, podemos declarar as variáveis da seguinte maneira:

```
Algoritmo "Dados para estacionamento de veículo"
Var
      permanencia:  inteiro
      tipo, placa:  caractere
      total_pagar:  real
Inicio
      // Seção de Comandos, procedimento, funções, operadores, etc...
Fimalgoritmo
```

4.2 ATIVIDADE 2

Situação: Quando um consumidor adquire uma determinada mercadoria e decide parcelar o pagamento, uma taxa de juro incide sobre o valor do produto.

Juro corresponde à quantia, paga ou recebida, proporcional a certa quantia em dinheiro emprestada ou investida por determinado período de tempo. Juro também é

cobrado ao se pagar faturas ou prestações em atraso, sendo, neste caso, um meio de recompensar quem deveria receber e não recebeu na data prevista. Para estudar situações envolvendo juro, utiliza-se alguns termos:

Capital: é a quantia emprestada ou investida.

Juro: valor do acréscimo pago ou do rendimento recebido.

Taxa de juro: porcentagem do valor emprestado ou investido em certo período de tempo.

Tempo: período em que certa quantia é emprestada ou investida.

Montante: soma do capital com o juro.

Proposta: O objetivo dessa atividade é identificar que dados serão necessários para a criação de um algoritmo que calcule o juro no parcelamento de uma mercadoria. Os alunos devem identificar, novamente, a necessidade de declarar um dado como **inteiro** ou **real**.

Sugestão: Podemos criar uma variável do tipo **inteiro** para armazenar o número de parcelas (que deve ser na verdade um inteiro não negativo), variáveis do tipo **real** para o valor inicial da mercadoria, taxa de juro (que pode ser por exemplo de 0,67% a.m.), valor de cada parcela e o valor total a se pagar pelo produto. Pode também haver uma variável do tipo **caractere** para armazenar a identificação (nome ou código) do produto. Lembrando que para declarar variáveis é importante usar nomes que facilitem a identificação, assim cada aluno usa o nome que lhe é mais apropriado. Para ficar de acordo com os termos usados em matemática, uma ideia é nomear parcelas como tempo, valor inicial como capital, valor final como montante e a taxa de juro apenas de taxa.

Assim, uma sugestão é declarar da seguinte maneira:

```
Algoritmo "Compra parcelada"
Var
      tempo:  inteiro
      produto:  caractere
      capital, montante, valor_parcela, taxa:  real
Inicio
      // Seção de Comandos, procedimento, funções, operadores, etc...
Fimalgoritmo
```

4.3 ATIVIDADE 3

Situação: A média aritmética, ou simplesmente média, de um conjunto de valores numéricos é calculada somando-se todos esses valores e dividindo o resultado pelo número de elementos somados. Por exemplo, a média entre os quatro números naturais $5, 7, 8$ e 12 é calculada como

$$Média = \frac{5+7+8+12}{4} = \frac{32}{4} = 8$$

Proposta: O objetivo dessa tarefa é construir um algoritmo que calcule a média entre quatro números reais digitados pelo usuário. Além da declaração de variáveis, os alunos agora devem identificar as três etapas (Entrada - Processamento - Saída) do algoritmo: solicitar ao usuário os quatro números e registrá-los, de posse desses valores efetuar o cálculo da média e por último fornecer ao usuário o resultado encontrado. É importante observar que, mesmo se o aluno limitar os valores digitados pelo usuário como números inteiros, há a possibilidade do resultado da média ser um número real.

Sugestão: Há, nesta atividade, a necessidade de quatro variáveis para armazenamento dos valores a serem fornecidos pelo usuário, que podem ser inteiros ou reais, dependendo do que se deseja do usuário. Além de uma variável para armazenar o resultado da média a ser calculada.

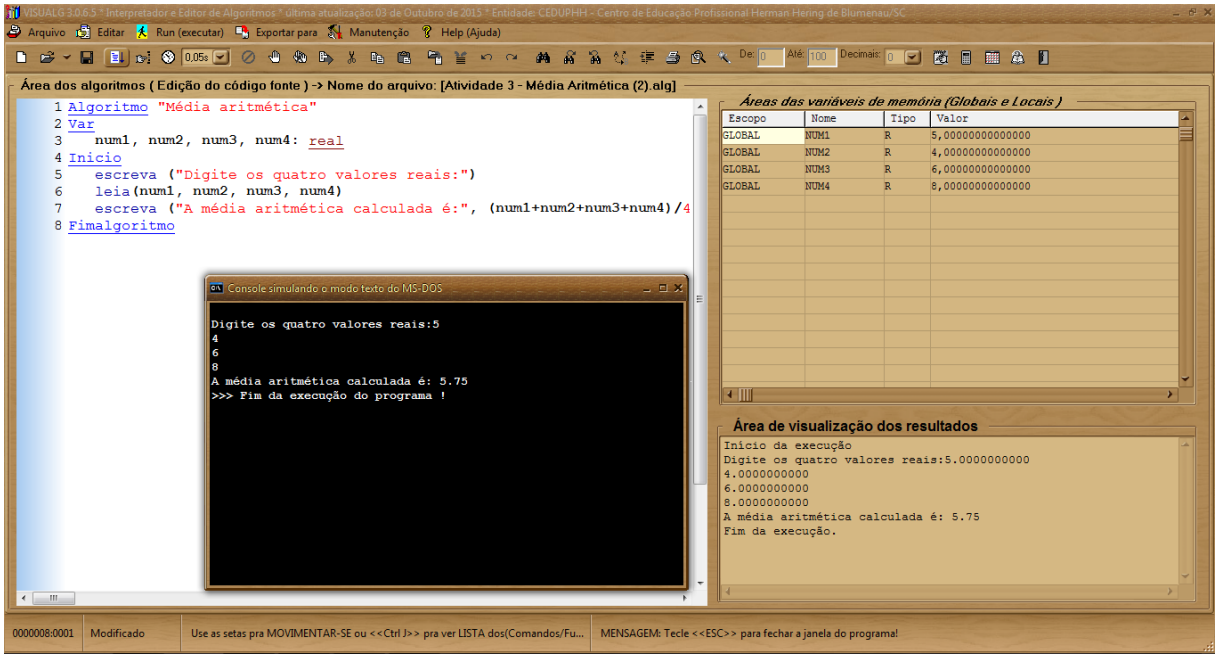
Um sugestão de algoritmo é:

```
Algoritmo "Média aritmética"
Var
      num1, num2, num3, num4, media:  real
Inicio
      escreva ("Digite o primeiro número real:")
      leia(num1)
      escreva ("Digite o segundo número real:")
      leia(num2)
      escreva ("Digite o terceiro número real:")
      leia(num3)
      escreva ("Digite o quarto número real:")
      leia(num4)
      media<-(num1+num2+num3+num4)/4
      escreva ("A média aritmética é:  ", media)
Fimalgoritmo
```

Esta escrita pode ser reduzida observando-se que a sintaxe do comando **leia** permite a leitura de mais de uma variável no mesmo comando. Ainda, a variável real media não se faz obrigatória, pois cálculos podem ser executados dentro do comando **escreva**. Assim uma sugestão de algoritmo com forma mais simples é:

```
Algoritmo "Média aritmética"
Var
      num1, num2, num3, num4:  real
Inicio
      escreva ("Digite os quatro valores reais:")
      leia(num1, num2, num3, num4)
      escreva ("A média aritmética calculada é:  ", (num1+num2+num3+
      num4)/4)
Fimalgoritmo
```

Figura 4 – Implementação do Algoritmo "Média aritmética" no VisuAlg

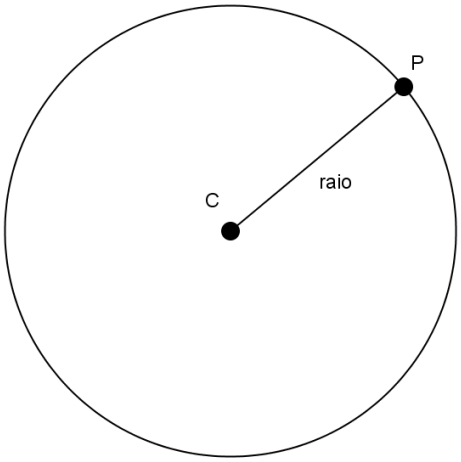


Fonte: Print screen da implementação do Algoritmo "Média aritmética" no VisuAlg no Sistema Operacional Windows 7

4.4 ATIVIDADE 4

Situação: Circunferência é a figura geométrica plana formada por todos os pontos que estão a uma mesma distância de um ponto fixo, chamado centro.

Figura 5 – Circunferência



Seu comprimento é dado por $C = 2\pi r$, enquanto sua área é $A = \pi r^2$, em que $\pi \approx 3,14$ e r é a distância de um ponto fixo ao centro, chamada de raio da circunferência.

Assim, uma circunferência de raio igual a $5cm$ possui comprimento

$$C = 2 \times 3,14 \times 5 = 31,4cm$$

e área

$$A = 3,14 \times 5^2 = 3,14 \times 25 = 78,5cm^2$$

Proposta: O objetivo desta atividade é construir um algoritmo que determine o comprimento e a área de uma circunferência sendo o raio e π números reais fornecidos pelo usuário. Os alunos devem identificar quais variáveis serão necessárias e as etapas (Entrada - Processamento - Saída) do algoritmo: solicitar e receber do usuário o raio e o valor de π, efetuar os cálculos e informar os resultados encontrados.

Sugestão: Nesta atividade, é natural tentar nomear a variável π por seu nome `pi`, o que acarretará em erro, pois se trata de uma palavra reservada no VisuAlg, logo não pode ser usada como identificação.

Além disso, para o cálculo da área, $A = \pi r^2$, como o aluno desconhece ainda uma função que calcule a potência diretamente, ele pode fazer uso da definição e escrever

$r^2 = r \times r$. Desta forma, a área pode ser calculada como $A = \pi \times r \times r$, sem a necessidade de parênteses já que na multiplicação, a ordem dos fatores não altera o produto.

Possíveis erros de digitação por parte do usuário, como digitação de números negativos, ainda serão ignorados nessa etapa.

Logo, uma sugestão de algoritmo é:

```
Algoritmo "Comprimento e Área de uma circunferência"
Var
      raio, comprimento, area, valor_pi:  real
Inicio
      escreva("Digite o raio da circunferência:")
      leia(raio)
      escreva("Digite um valor aproximado para π:")
      leia(valor_pi)
      comprimento<-2*valor_pi*raio
      area<-valor_pi*raio*raio
      escreva("O comprimento da circunferência é:  ",comprimento,"e a área
é:  ",area)
Fimalgoritmo
```

4.5 ATIVIDADE 5

Situação: Equação do segundo grau na incógnita x é toda equação que pode ser escrita na forma $ax^2 + bx + c = 0$, em que a, b e c são números reais e $a \neq 0$. Chama-se de equação do segundo grau incompleta, quando $b = 0$ ou $c = 0$.

Um dos métodos para determinar a solução de uma equação do segundo grau incompleta quando $c = 0$, ou seja, do tipo $ax^2 + bx = 0$ é colocar o fator comum, x, em evidência. Assim,

$$x.(ax + b) = 0$$

e segue que

$$x = 0$$

ou

$$ax + b = 0 \Rightarrow x = -\frac{b}{a}$$

De modo geral, as equações do tipo $ax^2 + bx = 0$, com a e b não nulos, possuem duas raízes reais e diferentes, $x = 0$ e $x = -\frac{b}{a}$.

Proposta: O objetivo desta atividade é construir um algoritmo que calcule as raízes reais de uma equação do segundo grau da forma $ax^2 + bx = 0$. Seguindo a linha das atividades executadas anteriormente, os alunos devem identificar quais variáveis serão necessárias e seu tipo. Devem também identificar as etapas, solicitar e registrar os valores dos coeficientes a e b, efetuar os cálculos necessários e por fim, fornecer o resultado encontrado.

Sugestão: Para os coeficientes, duas variáveis reais, uma para o valor de a e outra para b. Uma terceira variável real pode ser criada para armazenar o quociente $x = -\frac{b}{a}$, mas este resultado pode também ser obtido dentro do comando **escreva**.

A proposta é que o usuário insira, para os coeficientes, valores reais não nulos. No entanto, como não é possível restringir essa entrada ainda, é interessante que os alunos verifiquem o funcionamento do algoritmo para valores nulos e discutam o resultado, já que é muito comum eles apresentarem dificuldades quanto ao uso do número zero na operação de divisão.

Uma sugestão de algoritmo é:

```
Algoritmo "Raízes de uma equação do segundo grau incompleta"
Var
      a, b, raiz:  real
Inicio
      escreva("Digite os valores dos coeficientes a e b (não nulos), res-
      pectivamente:  ")
      leia(a,b)
      raiz<- -b/a
      escreva("As raízes calculadas são x=0 e x=",raiz)
Fimalgoritmo
```

Vale salientar que, no VisuAlg, todo o texto do comando **escreva** deve ser digitado sem quebra de linha.

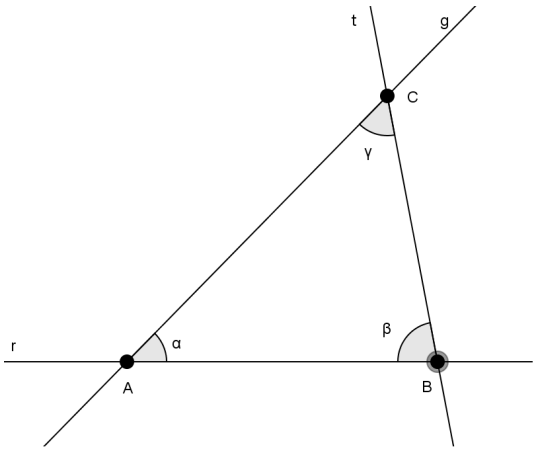
4.6 ATIVIDADE 6

Situação: Triângulo é a figura geométrica formada por três retas que se encontram duas a duas e não passam pelo mesmo ponto, formando três lados e três ângulos.

A fórmula tradicional para cálculo da área do triângulo, ensinada e muito utilizada no ensino fundamental é

$$Área = \frac{base \times altura}{2}$$

Figura 6 – Triângulo ABC



Entretanto, outras fórmulas foram desenvolvidas para realizar este cálculo. Uma delas é a fórmula de Herão, que calcula a área do triângulo em função da medida dos seus três lados. O nome faz referência ao matemático grego Herão de Alexandria, que viveu em um período que varia entre 150 a.C. e 250 d.C.

Segundo Herão, dado um triângulo qualquer cujas medidas dos lados são a, b e c, sua área A é dada por

$$A = \sqrt{p(p-a)(p-b)(p-c)}$$

em que p é o semiperímetro do triângulo, isto é,

$$p = \frac{a+b+c}{2}$$

Assim, para determinar a área de um triângulo de lados $a = 3cm$, $b = 4cm$ e $c = 5cm$, calculamos inicialmente o semiperímetro

$$p = \frac{a+b+c}{2} = \frac{3+4+5}{2} = \frac{12}{2} = 6cm$$

e em seguida a área

$$A = \sqrt{p(p-a)(p-b)(p-c)} = \sqrt{6(6-3)(6-4)(6-5)} = \sqrt{36} = 6cm^2$$

Proposta: O objetivo desta atividade é elaborar um algoritmo que calcule a área de um triângulo cujos lados serão fornecidos pelo usuário. O aluno deve identificar, além das variáveis e seu tipo, as etapas do algoritmo: solicitar e registrar os lados digitados pelo usuário, efetuar os cálculos necessários e fornecer o resultado encontrado.

Sugestão: A existência ou não de um triângulo com as medidas que serão informadas pelo usuário será analisada em exercício posterior. No entanto, uma breve discussão a respeito é válida, inclusive com o uso do software Geogebra, se houver a possibilidade.

Considerando que o triângulo exista, podem ser declaradas três variáveis reais para armazenar suas medidas, uma para o semiperímetro e uma para a área. O aluno deve perceber que primeiro será necessário determinar o semiperímetro para, em seguida, calcular a área, onde introduziremos o uso das funções pré-definidas no VisuAlg, neste caso a função **RaizQ(expressão)**.

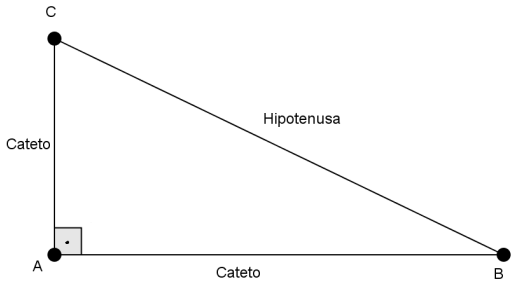
Uma sugestão de algoritmo é:

```
Algoritmo "Área de um triângulo"
Var
      a, b, c, p, area:  real
Inicio
      escreval("Digite as medidas dos lados, a, b e c, do triângulo:  ")
      leia(a,b,c)
      p<-(a+b+c)/2
      area<-RaizQ(p*(p-a)*(p-b)*(p-c))
      escreva("A área do triângulo é de ",area,"unidades de área")
Fimalgoritmo
```

4.7 ATIVIDADE 7

Situação: Pitágoras, importante matemático e filósofo grego, provavelmente viveu no século VI a.C. na ilha de Samos, no mar Egeu, onde teria nascido, e mais tarde em Crotona, na Magna Grécia, onde foi fundada uma escola filosófica com características de seita em que se estudava música, matemática, filosofia e astronomia. Um importante teorema atribuído a ele, o conhecido Teorema de Pitágoras, diz que em qualquer triângulo retângulo, a soma dos quadrados dos catetos é igual ao quadrado da hipotenusa. Em que, por definição, a hipotenusa é o lado oposto ao ângulo reto, e os catetos são os dois lados que o formam.

Figura 7 – Triângulo Retângulo



Assim, se um triângulo retângulo possui catetos $b = 5cm$, $c = 12cm$ e hipotenusa

a, então

$$a^2 = b^2 + c^2 = 5^2 + 12^2 = 169 \Rightarrow a = \sqrt{169} \Rightarrow a = 13cm$$

Proposta: O objetivo desta atividade é construir um algoritmo que calcule a hipotenusa de um triângulo retângulo em que os catetos são fornecidos pelo usuário. O aluno deve identificar, as variáveis e seu tipo e também as etapas do algoritmo: solicitar e registrar os catetos digitados pelo usuário, efetuar os cálculos necessários e fornecer o resultado encontrado.

Sugestão: Não é possível, ainda, verificar se os valores fornecidos pelo usuário são válidos. Deve-se perceber que os catetos não necessariamente são números inteiros, podendo ser números reais, assim como a hipotenusa a ser calculada. E, para esta, podem ser utilizadas a funções **Quad (expressão)** e **RaizQ(expressão)**.

Uma sugestão de algoritmo é:

```
Algoritmo "Cálculo da Hipotenusa"
Var
      cateto1, cateto2, hipotenusa:  real
Inicio
      escreval("Digite as medidas dos catetos do triângulo retângulo:  ")
      leia(cateto1, cateto2)
      hipotenusa<-RaizQ(Quad(cateto1)+Quad(cateto2))
      escreva("A hipotenusa do triângulo retângulo mede :", hipotenusa,"u-
      nidades de comprimento")
Fimalgoritmo
```

4.8 ATIVIDADE 8

Situação: Um dos métodos para se determinar as raízes de uma equação do segundo grau do tipo $ax^2 + bx + c = 0$, com a, b e c reais quaisquer e $a \neq 0$ é a fórmula resolutiva conhecida como fórmula de Báskara. O nome é uma homenagem a um dos mais importantes matemáticos do século XII, o indiano Bháskara (c.1114-1185), que estudou a resolução das equações do 2º grau.

Na fórmula resolutiva, as soluções são dadas por

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

em que a expressão $b^2 - 4ac$ é chamada de discriminante e geralmente é substituída pela letra grega Δ (delta).

Desta forma, dada a equação $x^2 + 5x + 6 = 0$, temos $a = 1$, $b = 5$ e $c = 6$ e, então,

$$\Delta = b^2 - 4ac = 5^2 - 4 \times 1 \times 6 = 25 - 24 = 1$$

assim,

$$x = \frac{-5 \pm \sqrt{1}}{2 \times 1} = \frac{-5 \pm \sqrt{1}}{2} = \frac{-5 \pm 1}{2}$$

e segue que

$$x_1 = \frac{-5+1}{2} = -2 \quad \text{e} \quad x_2 = \frac{-5-1}{2} = -3$$

Proposta: O objetivo dessa atividade é construir um algoritmo que calcule as raízes de uma equação do 2º grau, cujos coeficientes serão fornecidos pelo usuário. O aluno deve identificar quais variáveis serão necessárias e quais serão as etapas: solicitar e registrar os coeficientes da equação, efetuar os cálculos necessários e fornecer o resultado encontrado.

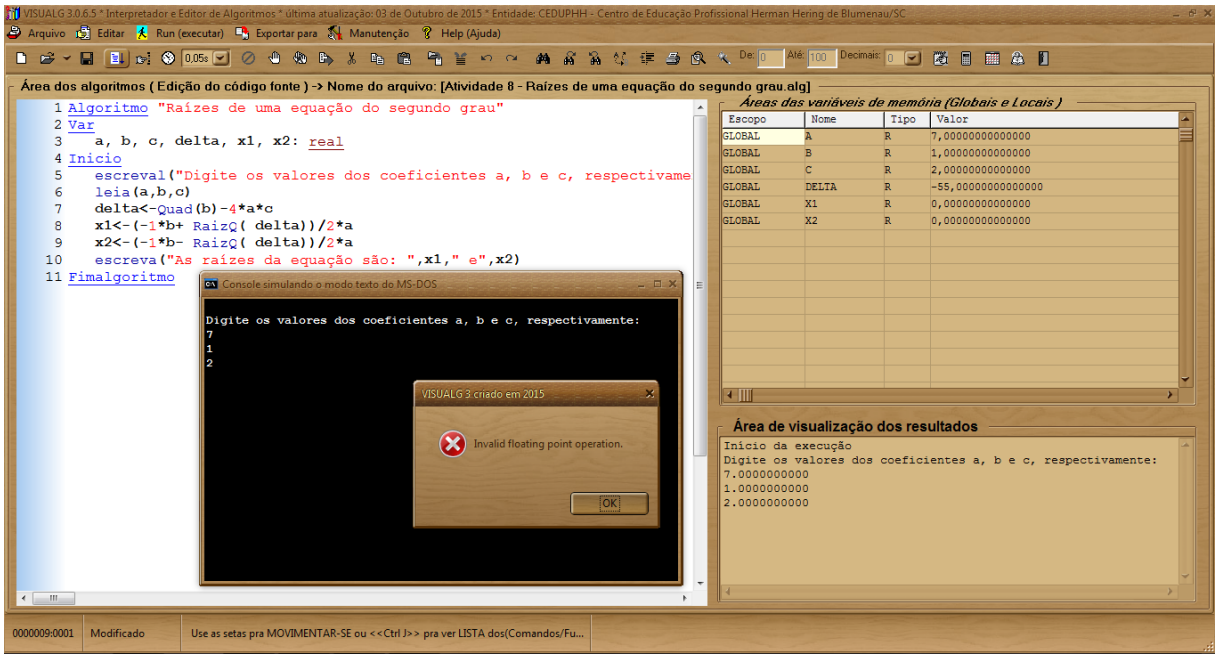
Sugestão: É importante observar que, para esta atividade, o valor de Δ pode ser obtido separado em uma linha de comando, com uma variável declarada especificamente para tal, ou diretamente no cálculo das duas raízes, já que ainda não temos recursos para verificar os possíveis casos (Δ positivo, negativo ou igual a zero). Ainda, caso Δ assuma valores negativos, quando o o VisuAlg for executado, será informado o erro "Invalid floating point operation", como mostra a Figura 8 ,que se deve ao fato das raízes `x1` e `x2` não serem dados do tipo `real`, logo não podem ser armazenadas.

É importante que o aluno verifique essas situações e compreenda a necessidade de estruturas que evitem esse tipo de erro.

Assim, uma sugestão de algoritmo é:

```
Algoritmo "Raízes de uma equação do segundo grau"
Var
      a, b, c, delta, x1, x2:  real
Inicio
      escreval("Digite os valores dos coeficientes a, b e c, respectiva-
      mente:  ")
      leia(a,b,c)
      delta<-Quad(b)-4*a*c
      x1<-(-1*b+ RaizQ( delta))/2*a
      x2<-(-1*b- RaizQ( delta))/2*a
      escreva("As raízes da equação são:  ",x1,"e ",x2)
Fimalgoritmo
```

Figura 8 – Implementação do Algoritmo "Raízes de uma equação do segundo grau" no VisuAlg



Fonte: Print screen da implementação do Algoritmo "Raízes de uma equação do segundo grau" no VisuAlg no Sistema Operacional Windows 7

4.9 ATIVIDADE 9

Situação: No algoritmo sugerido anteriormente sobre resolução de uma equação do segundo grau, para valores de coeficientes a, b e c que resultam em Δ negativo o VisuAlg informa a ocorrência do erro "Invalid floating point operation", que se deve ao fato de que as raízes `x1` e `x2` não são dados do tipo `real`, declarados no algoritmo.

Proposta: De posse das estruturas condicionais, nessa atividade o aluno deve, como no exercício anterior, elaborar um algoritmo que determine as raízes de uma equação do segundo grau. No entanto, agora, deve garantir, fazendo uso da estrutura condicional, que as raízes sejam calculadas apenas para casos em que Δ seja não negativo, ou seja, $\Delta \geq 0$.

Sugestão: A partir de agora a etapa de processamento começa a ficar mais elaborada, podendo ser feita de diversas maneiras. É interessante, a partir deste ponto, que os alunos sejam colaboradores uns dos outros, discutindo a respeito dos algoritmos elaborados por todos. Ele deve ser capaz de identificar erros e possíveis exageros para conseguir elaborar um algoritmo eficaz.

Um fator importante também é o posicionamento do comando **`escreva`** na etapa de saída do resultado para o usuário.

Uma sugestão de algoritmo é:

```
Algoritmo "Raízes de uma equação do segundo grau"
Var
      a, b, c, delta, x1, x2:real
Inicio
      escreva("Digite os coeficientes reais a, b e c, respectivamente:  ")
      leia(a,b,c)
      delta<-Quad(b)-4*a*c
      se delta>=0 entao
            x1<-(-1*b+ RaizQ( delta))/2*a
            x2<-(-1*b- RaizQ( delta))/2*a
            escreva("As raízes da equação são:  ",x1,"e ",x2)
      senao
            escreva("A equação não possui raiz real.")
      fimse
Fimalgoritmo
```

4.10 ATIVIDADE 10

Situação: Determinar a raiz quarta de um número real, não negativo, x, consiste em calcular um número não negativo que, elevado à quarta potência, gera o valor x. Por exemplo, a raiz quarta do número 16. Este resultado pode ser obtido manualmente através da fatoração em primos $16 = 2^4$.

Assim , $\sqrt[4]{16} = \sqrt[4]{2^4} = 2$, pela propriedade de potência $\sqrt[q]{n^q} = n$, $n \geq 0$.

Ou em uma calculadora, usando duas vezes seguidas o operador $\sqrt{}$, que nada mais é do que fazer uso da propriedade de potência ${}^{p.q}\sqrt{n} = \sqrt[p]{\sqrt[q]{n}}$.

Proposta: O objetivo desta atividade é criar um algoritmo que calcule a raiz quarta de um número real, não negativo, fornecido pelo usuário.

Sugestão: Novamente temos um resultado que só pode ser obtido para valores positivos e será necessário o uso de uma estrutura condicional para garantir que só aconteça sob essa condição. Ainda, o VisuAlg possui uma função pré-definida para cálculo de raiz quadrada, mas não para raiz quarta, logo será necessário recorrer à propriedade de potência ${}^{p.q}\sqrt{n} = \sqrt[p]{\sqrt[q]{n}}$.

Uma sugestão de algoritmo é:

```
Algoritmo "Raiz quarta de um número real"
Var
      numero, raiz:  real
Inicio
      escreva("Digite um número:  ")
      leia (numero)
      se numero<0 entao
            escreva ("A raiz quarta de ",numero,"não é um número real.")
      senao
            raiz<-RaizQ(RaizQ(numero))
            escreva ("A raiz quarta de ",numero,"é ", raiz)
      fimse
Fimalgoritmo
```

4.11 ATIVIDADE 11

Situação: Segundo a concepção pitagórica, par é o número que pode ser dividido em duas partes iguais, sem que uma unidade fique no meio, e ímpar é aquele que não pode ser dividido em duas partes iguais, porque sempre há uma unidade no meio. Atualmente definimos par como o número que ao ser dividido por dois têm resto zero e número ímpar aquele que ao ser dividido por dois tem resto diferente de zero.

Proposta: O objetivo desta atividade é construir um algoritmo que, dado um número inteiro qualquer, reconheça se ele é par ou ímpar.

Sugestão: Espera-se nessa atividade que o aluno seja capaz de identificar a necessidade da estrutura condicional. Usando a definição de paridade ele deve perceber que, se um número ao ser dividido por dois deixa resto zero então ele é dito par, senão ele é dito ímpar.

Uma sugestão é utilizar a função `Int(expressão)`, que fornece como resultado apenas a parte inteira da expressão. Com ela, podemos determinar se a divisão é ou não exata da seguinte forma:

```
Algoritmo "Paridade de um número inteiro"
Var
      num:inteiro
      divisao:  real
```

```
Inicio
      escreva("Digite um número:  ")
      leia(num)
      divisao<-num/2
      se divisao-int(divisao)=0 entao
            escreva("O número digitado é par.")
      senao
            escreva ("O número digitado é ímpar.")
      fimse
Fimalgoritmo
```

Ou ainda, usando a função `x mod y` que tem como saída o resto da divisão de x por y, sem precisar de um cálculo auxiliar. Podemos escrever o algoritmo:

```
Algoritmo "Paridade de um número inteiro"
Var
      num:inteiro
Inicio
      escreva("Digite um número:  ")
      leia(num)
      se (num mod 2)=0 entao
            escreva("O número digitado é par.")
      senao
            escreva ("O número digitado é ímpar.")
      fimse
Fimalgoritmo
```

4.12 ATIVIDADE 12

Situação: Em período de liquidação, o comércio oferece diversas promoções, dentre elas é comum encontrar a promoção na qual, comprando três peças, o cliente não paga a peça de menor valor. Por exemplo, se um cliente compra três camisas nos valores R$45,60, R$54,00 e R$36,99, esta última, como possui menor valor não será cobrada. Assim, o cliente pagará pela compra um total de R$99,60.

Proposta: O objetivo desta atividade é que o aluno construa um algoritmo que receba os valores das três peças adquiridas pelo cliente e retorne o valor final a pagar, analisando qual valor será descartado por ser o menor dos três.

Sugestão: Para verificar qual das peças possui o menor valor numérico, podemos usar a estrutura condicional encadeada de diversas formas. Na tentativa de diminuir o número de estruturas, podemos utilizar o conector lógico **e** e verificar se um determinado número é maior que os outros dois em uma mesma estrutura. Além disso, para calcular o total que o cliente deve pagar, podemos somar todos os valores e na sequência subtrair o menor valor encontrado da seguinte forma:

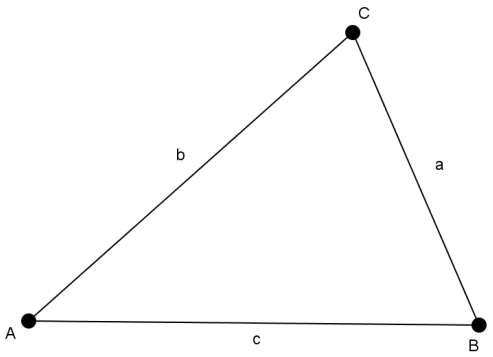
```
Algoritmo "Promoção da peça de menor valor"
Var
      valor1, valor2, valor3, menor, total :real
Inicio
      escreva("Informe o preço das três peças:  ")
      leia(valor1, valor2, valor3)
      se (valor1<valor2) e (valor1<valor3) entao
            menor<-valor1
      senao
            se valor2<valor3 entao
                  menor<-valor2
            senao
                  menor<-valor3
            fimse
      fimse
      total<-valor1+valor2+valor3-menor
      escreva("O total a pagar é:  ",total)
Fimalgoritmo
```

4.13 ATIVIDADE 13

Situação: No algoritmo proposto na atividade da seção 4.6, para cálculo da área de um triângulo dadas as medidas dos três lados, a, b e c, ignoramos o fato de que nem todos os valores fornecidos pelo usuário podem, de fato, representar um triângulo. Para nos certificarmos da possibilidade de existência de um triângulo, faremos uso da condição de existência de triângulos.

Para representar um triângulo é necessário que a medida de um lado qualquer seja menor que a soma das medidas dos outros dois lados. Isto é, dado o triângulo ABC de lados a, b e c, temos que as três condições abaixo devem ser satisfeitas:

Figura 9 – Triângulo ABC



$$a < b + c$$

$$b < a + c$$

$$c < a + b$$

É importante ressaltar que só é realmente necessário verificar a desigualdade para o maior lado do triângulo, ou seja, basta que a medida do maior lado seja menor que a soma dos outros dois. Logo, seria fundamental apenas uma estrutura que identificasse o maior valor digitado para em seguida compará-lo. Estrutura essa que se assemelharia à construída na atividade da seção anterior, 4.12. Daí a opção por não resolver esta atividade por este caminho.

Proposta: A atividade proposta é construir um algoritmo que, dadas as medidas dos lados do triângulo, verifique a existência desse triângulo.

Sugestão: Como três condições devem ser confirmadas simultaneamente, além do uso da estrutura condicional, será necessário o conector lógico **`e`** para que para que isso ocorra.

Ainda, também podemos verificar se todos os valores digitados pelo usuário são reais não negativos e não nulos. Se pelo menos um deles for negativo ou nulo também não é possível construir um triângulo. Para essa etapa podemos usar o conector lógico **`ou`**.

Nesta atividade o aluno precisa planejar bem a etapa de processamento, que é bem mais complexa que as anteriores e envolve o uso de dois conectores lógicos.

Um sugestão de algoritmo é:

```
Algoritmo "Condição de existência de um triângulo"
Var
```

```
      a, b, c :real
Inicio
      escreva("Digite as medidas dos lados do triângulo:  ")
      leia(a,b,c)
      se (a<=0) ou (b<=0) ou (c<=0) entao
            escreva("Não é possível construir um triângulo com essas medi-
            das, pois pelo menos um de seus lados é negativo ou nulo.")
      senao
            se (a<b+c) e (b<a+c) e (c<a+b) entao
                  escreva("É possível construir um triângulo com as medi-
informadas.")
            senao
                  escreva("Segundo a condição de existência de um triân-
             gulo, não é possível construí-lo com as medidas informadas.")
            fimse
      fimse
Fimalgoritmo
```

4.14 ATIVIDADE 14

Situação: A poupança, ou "caderneta de poupança" como também é chamada, é regulada pelo Banco Central, com uma remuneração de 0,5% ao mês, aplicada sobre os valores atualizados, ou seja, a um regime de juro composto, mais a variação do TR (Taxa Referencial), índice criado pelo governo para complementação do rendimento das poupanças. Ele é calculado a partir da taxa SELIC (Sistema Especial de Liquidação e de Custódia), que indica a taxa de juros no Brasil.

Um regime de juro é dito simples quando o taxa de juro incide apenas sobre o valor do capital. Sobre o juro gerado a cada período não incidirão novos juros. Já o juro composto é calculado sobre o capital apenas no primeiro período. Nos demais períodos é calculado sobre o montante obtido no período anterior.

Por exemplo, se uma pessoa faz um depóstito de R$3.000,00 na poupança, esse é seu capital. Ao final do primeiro mês o juro será calculado sobre o capital, assim ela terá:

$$Juros = 3.000,00 \times \frac{0,5}{100} = 15,00 \quad \text{e} \quad Montante_1 = 3.000,00 + 15,00 = 3.015,00$$

A partir do segundo mês, o juro passa a ser calculado sobre o montante do mês anterior, assim, no final do segundo mês ela terá:

$$Juros = 3.015,00 \times \frac{0,5}{100} = 15,075 \quad \text{e} \quad Montante_2 = 3.015,00 + 15,075 = 3.030,075$$

Proposta: O objetivo desta atividade é criar um algoritmo que calcule o montante acumulado na aplicação de um capital, na conta poupança, por um determinado período de tempo. Neste algoritmo, o usuário deve fornecer o capital aplicado, a taxa de rendimento da poupança e o tempo da aplicação.

Sugestão: Para esta atividade o objetivo é não mencionar nenhuma fórmula direta para cálculo de juros compostos. Espera-se que o aluno perceba que um mesmo cálculo será utilizado um determinado número de vezes e que por isso será necessária uma estrutura de repetição.

Além disso, deve ser observado que o montante é sempre a soma do montante anterior com o juro calculado para o mês em questão, sendo assim uma variável acumuladora.

Um algoritmo para esta atividade pode usar qualquer uma das estruturas de repetição do VisuAlg, sendo que para os comandos **enquanto-faca** e **repita-ate** será necessário o auxílio de uma variável contadora, e para o comando **para-ate-faça** não é obrigatório.

Usando o comando **para-ate-faça**, uma sugestão é:

```
Algoritmo "Juro de poupança"
Var
      juro, capital, taxa:  real
      temp:  inteiro
Inicio
      escreva ("Digite o capital investido:  ")
      leia(capital)
      escreva( "Digite o rendimento (taxa):  ")
      leia(taxa)
      escreva ("Digite o tempo total do investimento:  ")
      leia(temp)
      se temp>=0 entao
            para temp de temp ate 1 passo -1 faca
                  juro<-capital*taxa/100
                  capital<-capital+juro
            fimpara
            escreva("O montante acumulado é de ",capital, "reais")
      senao
            escreva("Não é possível calcular o Montante para tempo negati-
            vo.")
      fimse
```

```
Fimalgoritmo
```

Já com o comando **enquanto-faca** um sugestão de algoritmo é:

```
Algoritmo "Juro de poupança"
Var
      juro, capital, taxa:  real
      temp:  inteiro
Inicio
      escreva ("Digite o capital investido:  ")
      leia (capital)
      escreva ("Digite o rendimento (taxa):  ")
      leia (taxa)
      escreva ("Digite o tempo total do investimento:  ")
      leia (temp)
      se temp>=0 entao
            enquanto temp>0 faca
                  juro<-capital*taxa/100
                  capital<-capital+juro
                  temp<-temp-1
            fimenquanto
            escreva ("O montante acumulado é de ",capital, "reais")
      senao
            escreva("Não é possível calcular o Montante para tempo negati-
            vo.")
      fimse
Fimalgoritmo
```

4.15 ATIVIDADE 15

Situação: Já foi visto na atividade da seção 4.6 que média aritmética, ou simplesmente média, de um conjunto de valores numéricos é calculada somando-se todos esses valores e dividindo o resultado pelo número de elementos somados, que é igual ao número de elementos do conjunto.

Proposta: O objetivo deste exercício é construir um algoritmo que calcule a média aritmética dentre um conjunto de valores digitados pelo usuário. Neste algoritmo, o usuário pode inserir quantos valores ele quiser.

Sugestão: Como o usuário pode inserir quantos valores quiser, e não sabemos quantos serão, não é possível criar uma variável para cada um. Uma possível saída é declarar uma única variável que acumule a soma dos valores conforme forem digitados. Será necessária ainda uma variável contadora para registrar essa quantidade.

Para que o usuário informe que todos os valores já foram digitados, e então o algoritmo calcular a média, pode-se criar algum tipo de código, que ao ser digitado pára a repetição.

Para cálculo de média aritmética não há nenhuma restrição quanto ao número fornecido pelo usuário, logo nenhum teste com esse objetivo é obrigatório.

Uma sugestão de algoritmo é:

```
Algoritmo "Média Aritmética"
Var
      numero, soma:  real
      cont, parar:  inteiro
Inicio
      cont<-0
      soma<-0
      repita
            escreva("Digite um número:  ")
            leia(numero)
            cont<-cont+1
            soma<-soma+numero
            escreval("Digite:")
            escreval("1 - Se deseja inserir outro número.")
            escreval("0 - Se deseja calcular a média aritmética")
            leia(parar)
      ate parar=0
      escreva("A média aritmética é:  ",soma/cont)
Fimalgoritmo
```

4.16 ATIVIDADE 16

Situação: Define-se fatorial de um número natural n>1, representado por $n!$, o produto de todos os inteiros positivos menores ou iguais a n. Ou seja $n! = n.(n-1).(n-2).....3.2.1$. Assim, o fatorial de 5, é escrito como $5! = 5 \times 4 \times 3 \times 2 \times 1 \times = 120$. Define-se

ainda que $1! = 1$ e $0! = 1$.

Proposta: O objetivo dessa atividade é construir um algoritmo que calcule o fatorial de um número natural digitado pelo usuário. Embora normalmente não seja objeto de estudo no ensino fundamental, é um conceito relativamente simples que exige do aluno apenas o conhecimento matemático sobre a operação de multiplicação e de antecessor.

Sugestão: Já que o algoritmo deve operar para qualquer valor fornecido pelo usuário, não é possível prever o número de fatores que serão necessários. Mas sabe-se que o resultado é o produto do valor informado com todos os seus antecessores até o número 1, e isso pode ser obtido usando uma estrutura de repetição.

O aluno deve tomar os devidos cuidados com possíveis valores negativos e támbém com o 1! e o 0!.

Uma sugestão de algoritmo para esta atividade é:

```
Algoritmo "Fatorial"
Var
      num, fatorial:  inteiro
Inicio
      escreva ("Digite um número natural:  ")
      leia (num)
      fatorial<-num
      escreva ("O fatorial de ", num, "é igual à:  ")
      se (num=0) ou (num=1) entao
            escreva ("1")
      senao
            para num de num-1 ate 1 passo (-1) faca
                  fatorial<-fatorial*num
            fimpara
            escreva (fatorial)
      fimse
Fimalgoritmo
```

É importante lembrar que erros de digitação por parte do usuário podem ocorrer. Por exemplo, se o usuário digita um inteiro negativo, digamos -8, o algoritmo retorna que `O fatorial de -8 é igual à:  -8`, um erro. Logo é importante elaborar uma estrutura que verifique esses possíveis erros do usuário.

Uma sugestão é:

```
Algoritmo "Fatorial"
Var
      num, fatorial:  inteiro
Inicio
      escreva ("Digite um número natural:  ")
      leia (num)
      se num>=0 entao
            fatorial<-num
            escreva ("O fatorial de ", num, "é igual à:  ")
            se (num=0) ou (num=1) entao
                  escreva ("1")
            senao
                  para num de num-1 ate 1 passo (-1) faca
                        fatorial<-fatorial*num
                  fimpara
                  escreva (fatorial)
            fimse
      senao
            escreva ("Não é possível calcular o fatorial do número digita-
            do pois este não é um número natural.")
      fimse
Fimalgoritmo
```

4.17 ATIVIDADE 17

Situação: Diz-se que um número inteiro a divide b quando o resto da divisão de b por a for igual à zero. Chamamos de *D(b)* o conjunto formado por todos os divisores de b.

Por exemplo, os divisores positivos do número 20 são os números positivos que ao dividirem 20 deixam resto 0. São eles:

$$D(20) = \{1, 2, 4, 5, 10, 20\}$$

Ainda, todos os elementos do conjunto dos números naturais são divisores de zero.

Proposta: O objetivo deste exercício é criar um algoritmo que enumere todos os divisores naturais de um número inteiro fornecido pelo usuário.

Sugestão: Como o conceito de divisor envolve o valor do resto da divisão, podemos fazer uso da função `x mod y` para determiná-lo. Ainda, o aluno deve perceber que todos os naturais de 1 até o valor digitado devem ser testados, logo será necessário usar uma estrutura de repetição.

No algoritmo sugerido abaixo o teste foi iniciado diretamente no número 2 ja que 1 é divisor de todos os números inteiros. Outro detalhe importante foi a utilização da função **Abs(expressão)** já que a proposta é determinar os divisores positivos, mas o valor fornecido pelo usuário pode ser um inteiro negativo.

```
Algoritmo "Divisores"
Var
      numero, auxiliar:  inteiro
Inicio
      escreva("Digite um número inteiro:  ")
      leia (numero)
      se numero = 0 entao
            escreva("D(0) = { 1 , 2, 3, 4 ,5 , 6, 7, 8 , 9, 10, 11, ...}")
      senao
            auxiliar<-2
            escreva("Os divisores positivos de ",numero,"são:  1")
            enquanto auxiliar<=abs(numero) faca
                  se (numero mod auxiliar)=0 entao
                        escreva(", ",auxiliar)
                  fimse
                  auxiliar<-auxiliar + 1
            fimenquanto
      fimse
Fimalgoritmo
```

4.18 ATIVIDADE 18

Situação: Número primo é todo número natural que possui exatamente dois divisores positivos distintos, o número 1 e ele mesmo. Assim, são primos os números 2, 3, 5, 7, 11, 13, 17, 19, ...

Proposta: O objetivo desta atividade é, utilizando como base a atividade anterior sobre divisores, desenvolver um algoritmo que identifique se um número digitado pelo usuário é, ou não, um número primo.

Sugestão: Já sabemos escrever quais são os divisores de um número inteiro. Usando a atividade anterior, podemos remodelá-la adicionando uma variável contadora para determinar quantos são esses divisores. De posse desse número, uma alternativa é o uso de uma estrutura condicional para determinar se essa quantidade é igual a dois, confome definição de número primo.

Uma sugestão de algoritmo é:

```
Algoritmo "Número primo"
Var
      numero, auxiliar, contador:  inteiro
Inicio
      escreva("Digite um número inteiro:  ")
      leia (numero)
      se numero < 2 entao
            escreva(numero, "não é um número primo.")
      senao
            auxiliar<-2
            contador<-1
            enquanto auxiliar<=abs(numero) faca
                  se (numero mod auxiliar)=0 entao
                        contador<-contador+1
                  fimse
                  auxiliar<-auxiliar + 1
            fimenquanto
            se contador>2 entao
                  escreva(numero,"não é um número primo.")
            senao
                  escreva(numero,"é um número primo.")
            fimse
      fimse
Fimalgoritmo
```

5 CONSIDERAÇÕES FINAIS

É notório que o estudante de hoje é um grande consumidor de tecnologia, e ela se faz presente de forma maciça na escola, muitas vezes de forma indesejada pelos docentes. Nesse sentido, este trabalho busca a inserção desses alunos de maneira mais consciente no mundo tecnológico, promovendo mais que apenas a utilização desse aparato, de modo que ele seja útil ao aprendizado.

Além de fortificar conceitos básicos da matemática do Ensino Fundamental, as atividades propostas neste trabalho têm por finalidade que os alunos ingressem no mundo da computação de forma a compreender a estrutura básica na construção de algoritmos de programação.

Ainda que planejadas visando estudantes de 9º ano do Ensino Fundamental, as atividades podem ser propostas para alunos de qualquer série do Ensino Fundamental ou Ensino Médio, contanto que se atente à possível necessidade de alterações para que se respeite o conhecimento matemático do aluno em cada série.

Vale ressaltar ainda que este trabalho aqui apresentado trata-se de uma proposta, a qual ainda não foi efetivamente aplicada. Nesse sentido, é preciso, a partir do momento em que se iniciem as atividades, observar se os discentes apresentarão dificuldades a fim de que o professor conduza, de forma adequada, as adaptações que se fizerem necessárias para se atingir os objetivos desejados. O docente, com sua experiência e embasamento teórico, há de constatar se existirá a necessidade inclusive da utilização de atividades mais simples que possam complementar o trabalho.

Pelo fato de ser professora efetiva em uma escola em que leciono no Ensino Fundamental e no Ensino Médio, está sendo delineado um projeto para a aplicação desta proposta com o intuito de ensinar programação a longo prazo. Será possível iniciar o trabalho com os alunos do Ensino Fundamental e estendê-lo até o Ensino Médio. Assim, será possível verificar a efetividade da proposta em diversas etapas da vida escolar do indivíduo, promovendo um ensino mais pertinente ao interesse dos alunos, o qual tem total ligação com o uso das tecnologias.

REFERÊNCIAS

[1] BARROS, Laíssa. **As vantagens da tecnologia no ensino das crianças**. 2013. Disponível em: <http://www.b9.com.br/38376/tech/as-vantagens-da-tecnologia-no-ensino-das-criancas/>. Acesso em: 28 jan. 2017.

[2] BERLINSKI, David. **O advento do algoritmo: A ideia que governa o mundo**. São Paulo: Globo, 2002. 420 p.

[3] BORATTI, Isaias C. e OLIVEIRA, A. B. **Introdução a Programação – Algoritmos**. Visual Books, 3 Ed. 2007

[4] CARVALHO, Rafael. **Para especialista, aprender programação é importante para qualquer profissional**. 2015. Disponível em: <https://www.napratica.org.br/aprender-programacao/>. Acesso em: 20 fev. 2017.

[5] CHAVANTE, Eduardo Rodrigues. **Convergências**: Matemática, 6º à 9º ano. São Paulo: Sm, 2015.

[6] CODE CLUB BRASIL. Disponível em: <http://www.codeclubbrasil.org.br/>. Acesso em: 20 fev. 2017

[7] CRUZ, Adriano J. O.; KNOPMAN, Jonas. **O que são algoritmos?** Projeto de Desenvolvimento de Algoritmos: Núcleo de Computação Eletrônica. Rio de Janeiro, 2001. Disponível em: <http://equipe.nce.ufrj.br/adriano/algoritmos/apostila/algoritmos.htm>. Acesso em: 28 mai. 2017.

[8] FARIAS, Gilberto. **Introdução à Computação**. Disponível em: <http://producao.virtual.ufpb.br/books/camyle/introducao-a-computacao-livro/livro/livro.chunked/index.html>. Acesso em: 24 mai. 2017.

[9] FONSECA FILHO, Cléuzio. **História da Computação: O caminho do pensamento e da tecnologia**. Porto Alegre: Edipucrs, 2007. 205 p.

[10] GARCIA, Rogério Eduardo; CORREIA, Ronaldo Celso Messias; SHIMABUKURO, Milton Hirokazu. **Ensino de Lógica de Programação e Estruturas de Dados para Alunos do Ensino Médio**. In Anais do XXVIII Congresso da SBC-WEI. Belém do Pará: 2008.

[11] GERALDES, Wendell Bento. **PROGRAMAR É BOM PARA AS CRIANÇAS? UMA VISÃO CRÍTICA SOBRE O ENSINO DE PROGRAMAÇÃO NAS ESCOLAS**. Texto Livre: Linguagem e Tecnologia, Belo Horizonte, v. 7, p.105-117, jul. 2014. Disponível em: <http://periodicos.letras.ufmg.br/index.php/textolivre/article/view/6143/5963>. Acesso em: 20 maio 2017.

[12] GOULART, Frederico. **Ensino de programação e linguagens digitais ganha espaço e adeptos no Brasil**. 2016. Disponível em: <http://cbn.globoradio.globo.com/editorias/tecnologia/2016/02/27/ENSINO-DE-PROGRAMACAO-E-LINGUAGENS-DIGITAIS-GANHA-ESPACO-E-ADEPTOS-NO-BRASIL.htm>. Acesso em: 27 jan. 2017.

[13] MORAES, Maria Candida. **INFORMÁTICA EDUCATIVA NO BRASIL: UMA HISTÓRIA VIVIDA, ALGUMAS LIÇÕES APRENDIDAS**. Revista Brasileira de Informática na Educação, 1997. 35 p.

[14] NASCIMENTO, João Kerginaldo Firmino do. **Informática aplicada à educaçã**o. Brasília, 2007. 84 p.

[15] OBAMA, B. **Don't Just Play on Your Phone, Program It**. The White House Blog. 2013. Disponível em: <https://obamawhitehouse.archives.gov/blog/2013/12/09/don-t-just-play-your-phone-program-it>. Acesso em: 19 maio 2017.

[16] OLIVEIRA, Ramon de. **Informática educativa: dos planos e discursos à sala de aula**. 13.ed. Campinas: Papirus, 2007.

[17] SANTOS, Fabiano dos. **Lógica de programação**. Rio de Janeiro: Seses, 2015. 188 p.

[18] SILVA, Rodrigo Luis de Souza da; OLIVEIRA, Alessandreia Marta de. **ALGORITMOS EM C**. Juiz de Fora: Clube de Autores, 2014. 118 p.

[19] SOUZA, Cláudio Morgado de. **VisuAlg - Ferramenta de Apoio ao Ensino de Programação**. Revista Eletrônica TECCEN / Universidade Severino Sombra, Vassouras, v. 2, p.1-9, 2009. Quadrimestral. Disponível em <http://www.uss.br/pages/revistas/revistateccen/V2N22009/ArtigoVisuAlgSOUZA.pdf>. Acesso em: 03 jan. 2017.

[20] TONET, Bruno. **Software VisuAlg 2.0**, NAPRO / Universidade de Caxias do Sul, 2013. 14 p. Disponível em: <http://www.netsoft.inf.br/aulas/4_ECI_Introducao_a_Informatica/Apostila_4.pdf>. Acesso em: 20 dez. 2016.

[21] VALENTE, José Armando; ALMEIDA, Fernando José de. **Visão analítica da informática na educação no Brasil: a questão da formação do professor**. 199-?. Disponível em: <http://www.professores.uff.br/hjbortol/car/library/valente.html>. Acesso em: 26 jan. 2017.

[22] VALENTE, José Armando. **O uso inteligente do computador na educação**. 199-?. Disponível em <http://www.pucrs.br/famat/viali/tic_literatura/artigos/computador/USOINTELIGENTE.pdf>. Acesso em: 26 jan. 2017.