INTRODUÇÃO A ALGORITMOS

AUTORES
Fábio Parreira
Sidnei Silveira
Cristiano Bertolini
Rosane Severo

LICENCIATURA EM COMPUTAÇÃO

INTRODUÇÃO A ALGORITMOS

AUTORES
Fábio Parreira
Sidnei Silveira
Cristiano Bertolini
Rosane Severo

1ª Edição
UAB/NTE/UFSM

UNIVERSIDADE FEDERAL DE SANTA MARIA

Santa Maria | RS
2017

Este caderno foi elaborado pelo Núcleo de Tecnologia Educacional da Universidade Federal de Santa Maria para os cursos da UAB.

NÚCLEO DE TECNOLOGIA EDUCACIONAL

DIRETOR DO NTE
Paulo Roberto Colusso

ELABORAÇÃO DO CONTEÚDO

Fábio Parreira, Sidnei Silveira
Cristiano Bertolini, Rosane Severo

REVISÃO LINGUÍSTICA
Camila Marchesan Cargnelutti
Maurício Sena

APOIO PEDAGÓGICO
Carmem Eloísa Berlote Brenner
Caroline da Silva dos Santos
Keila de Oliveira Urrutia
Siméia Tussi Jacques

EQUIPE DE DESIGN
Carlo Pozzobon de Moraes – Capa e Ilustração
Juliana Facco Segalla – Diagramação
Matheus Tanuri Pascotini

PROJETO GRÁFICO
Ana Letícia Oliveira do Amaral

I61 Introdução a algoritmos [recurso eletrônico] / Fábio Parreira ... [et al.] . – 1. ed. – Santa Maria, RS : UFSM, NTE, UAB, 2017.
1 e-book

Este caderno foi elaborado pelo Núcleo de Tecnologia Educacional da Universidade Federal de Santa Maria para os cursos da UAB
Acima do título: Licenciatura em computação
ISBN 978-85-8341-201-4

1. Computação 2. Algoritmos I. Parreira, Fábio II. Universidade Federal de Santa Maria. Núcleo de Tecnologia Educacional

CDU 004.421

Ficha catalográfica elaborada por Alenir Goularte - CRB-10/990
Biblioteca Central da UFSM

Ministério da **Educação**

APRESENTAÇÃO

Este livro foi produzido para servir como material didático digital básico para o desenvolvimento da disciplina de *Introdução a Algoritmos*, do Curso de Licenciatura em Computação. Para que você, futuro Licenciado em Computação, inicie o processo de aprendizado de programação de computadores, utilizaremos um ambiente de programação, o VisuAlg. A disciplina envolve o estudo de algoritmos e lógica de programação e o livro está organizado em 7 unidades, assim divididas:

– Unidade 1: Fundamentos de Programação – aborda os conceitos de algoritmos, constantes, variáveis, tipos de dados e formas de representação de algoritmos;
– Unidade 2: Operadores e Tipos de Dados – envolve o uso de expressões aritméticas e lógicas necessárias para a construção de algoritmos;
– Unidade 3: Algoritmos Sequenciais – compreende o estudo e aplicação dos comandos básicos de um algoritmo sequencial, envolvendo entrada de dados, processamento e saída;
– Unidade 4: Algoritmos com Decisão – envolve o estudo dos comandos que permitem o controle do fluxo de decisão em algoritmos;
– Unidade 5: Algoritmos com Repetição: estudo dos comandos que permitem o controle de fluxo de repetição em algoritmos;
– Unidade 6: Estruturas de Dados Básicas – aborda o uso de vetores e matrizes em algoritmos;
– Unidade 7: Funções e Procedimentos – aborda o uso de métodos/sub-rotinas em algoritmos.

Cabe lembrar que, em todas as unidades, são apresentadas séries de exercícios, elaborados a partir da experiência dos professores em sala de aula e de outros materiais referentes à lógica de programação. A metodologia empregada é a aprendizagem baseada em problemas *(Problem Based Learning)*. Por meio desta metodologia, os processos de ensino e de aprendizagem são transformados em questões (problemas), permitindo que os alunos aprendam a resolver problemas relacionados ao seu futuro profissional.

SAIBA MAIS: a metodologia PBL (Problem Based Learning) ou ABP (Aprendizagem Baseada em Problemas) enfatiza o aprendizado auto-dirigido, centrado no estudante. Nesta metodologia, os estudantes devem trabalhar com o objetivo de solucionar um problema real ou simulado a partir de um contexto.

A utilização de um ambiente de programação, como é o caso do VisuAlg, permite que você teste, na prática, os algoritmos estudados. A utilização deste ambiente está baseada no ciclo experimentar-refletir-generalizar-testar proposto por Cowan (2002). Você, nosso aluno, poderá construir um algoritmo e/ou programa, verificar os resultados apresentados por meio do VisuAlg, generalizar a solução, ou seja, pensar em um algoritmo que possa ser aplicado nos mais variados casos e validar esta generalização em outros problemas apresentados.

Desejamos bons estudos e um ótimo aprendizado nesta disciplina que é a base para os processos de ensino e de aprendizagem de programação de computadores.

ENTENDA OS ÍCONES

ATENÇÃO: faz uma chamada ao leitor sobre um assunto, abordado no texto, que merece destaque pela relevância.

INTERATIVIDADE: aponta recursos disponíveis na internet (sites, vídeos, jogos, artigos, objetos de aprendizagem) que auxiliam na compreensão do conteúdo da disciplina.

SAIBA MAIS: traz sugestões de conhecimentos relacionados ao tema abordado, facilitando a aprendizagem do aluno.

TERMO DO GLOSSÁRIO: indica definição mais detalhada de um termo, palavra ou expressão utilizada no texto.

SUMÁRIO

1

FUNDAMENTOS DA PROGRAMAÇÃO

INTRODUÇÃO

Esta unidade apresenta alguns conceitos importantes para que possamos iniciar o estudo da programação de computadores. Você, como futuro Licenciado em Computação, poderá atuar ensinando seus alunos a programar. Nesta unidade, vamos compreender como funciona a execução de um programa (*software*) no computador *(hardware*), além de entendermos os conceitos de algoritmo, pseudocódigo, código-fonte, regras de sintaxe e de semântica.

Ao final da unidade, você terá condições de resolver os exercícios propostos, escrevendo algoritmos em linguagem natural.

1.1 PROCESSAMENTO DE DADOS ELETRÔNICO

Um sistema computacional é dividido, basicamente, em duas partes: 1) ***hardware*** (parte física) e 2) ***software*** (parte lógica). O *hardware* abrange os componentes e o funcionamento interno do computador, enquanto que o *software* envolve o modo como se utiliza o computador. O *hardware* e o *software* são interdependentes, ou seja, de nada adianta um equipamento de última geração (hardware) sem que haja um bom programa (*software*) sendo executado nele, nem um *software* de última geração sendo executado em um computador ultrapassado (o que, na maioria das vezes, torna-se impossível).

O processamento de dados realizado pelo computador (processamento de dados eletrônico) possui muitas vantagens sobre o processamento manual, principalmente pela sua velocidade de execução. Para que um computador possa desempenhar tarefas (processar dados) faz-se necessário (FALKEMBACH; SILVEIRA, 2005) (SILVA; FALKEMBACH; SILVEIRA, 2010):

1. Descrever antecipadamente as operações que serão efetuadas, prevendo-se todos os casos possíveis para o problema que deve ser resolvido. Esta especificação é realizada por meio de um algoritmo. Um **algoritmo** é um conjunto de regras que permite realizar mecanicamente todas as operações particulares correspondentes a uma determinada tarefa. O algoritmo é uma decomposição de um problema do usuário em operações elementares que podem ser executadas pelo computador utilizando alguma notação que programadores possam entender.
2. Traduzir o algoritmo obtido para uma linguagem de programação (**programa fonte**). Uma linguagem de programação é composta por um conjunto bem definido de símbolos permitidos, regras de escrita e regras de comportamento.
3. Traduzir para o formato interno do computador essa sequência de operações, por meio de um programa especial chamado **compilador**. Um compilador é um programa que traduz instruções em linguagem de programação para o formato interno do computador.
4. Solicitar ao computador para que o programa seja executado, por meio de um comando do sistema operacional.

TERMO DO GLOSSÁRIO: compilador: programa que traduz um programa escrito em uma linguagem de programação de alto nível (mais próxima da linguagem natural) para um programa equivalente em código de máquina para que possa ser executado.

Uma segunda forma, mais completa, de apresentar os passos necessários para a resolução de problemas utilizando o computador é a seguinte (FALKEMBACH; SILVEIRA, 2005) (SILVA; FALKEMBACH; SILVEIRA, 2010):

1. Definição do Problema: consiste na descrição da situação a ser resolvida, por meio de um enunciado que deve ser claro e completo, fornecendo todas as informações necessárias para a sua resolução.
2. Análise do problema: consiste em obter, a partir da definição, informações, ou seja, subsídios para construir um modelo para a solução do problema e formalizar este modelo por meio de mecanismos, tais como os **algoritmos**. Quanto mais complexo for o problema mais se recomenda a utilização desta técnica.
3. Programação ou Codificação: consiste em transcrever, em uma **Linguagem de Programação**, as instruções referentes ao modelo da solução, criando o **Programa Fonte.**
4. Edição: consiste em transferir para o computador, mais especificamente para a memória RAM (*Random Access Memory*), as instruções do Programa Fonte.
5. Compilação: consiste na interpretação das instruções do **Programa Fonte** por meio da verificação da sintaxe do programa. O processo de compilação gera o **Programa Objeto**.
6. Ligação (*linkedição*): a partir do Programa Objeto, gera-se o **Programa Executável**.
7. Análise dos resultados (fase de testes): consiste em verificar se os resultados correspondem à expectativa, pois o fato do computador apresentar um resultado indica, inicialmente, que o programa não possui erros de sintaxe, mas não significa que não apresenta erros que envolvem a lógica de programação.
8. Documentação: consiste em descrever os procedimentos que foram utilizados na resolução do problema, para utilizá-los na solução de problemas similares.

TERMO DO GLOSSÁRIO: Programa Fonte ou Código-fonte: programa escrito (codificado) em uma linguagem de programação. Por exemplo, ao escrever um programa utilizando a Linguagem C, o programador tem acesso ao seu código-fonte. O usuário final pode ter acesso somente ao código executável, que lhe permite usar o programa, sem saber como o mesmo foi desenvolvido.

Programa Objeto: o programa objeto é gerado a partir do programa fonte, por meio de uma das etapas da compilação e ainda não representa um programa executável. Antes de gerar o programa executável é preciso passar por mais uma fase da compilação, realizada por um montador (ou linkeditor) que vai gerar o programa executável.

Uma **linguagem de programação** é um simbolismo que permite a comunicação entre o programador e o computador. A linguagem natural (como a Língua Portuguesa) não é útil para esta finalidade, pelo fato de que a sua sintaxe é muito complexa, assim como sua semântica. Por exemplo, quando escrevemos em Língua

Portuguesa a frase: "Apague a luz", esta frase está sintaticamente correta (escrevemos corretamente, seguindo as regras de Português) e semanticamente correta também, pois esta frase tem sentido. Mas, se escrevermos "Apague a porta"? Sintaticamente continua correto, mas semanticamente pode não fazer sentido. Como assim, apagar a porta? No entanto, se estivermos usando um *software* para construção de plantas arquitetônicas, pode ser que apagar a porta tenha sentido. O sentido (semântica), muitas vezes, depende do contexto.

Uma linguagem de programação é, portanto, mais restrita, e será definida por meio de um vocabulário autorizado, determinando sua sintaxe e sua semântica. Em uma linguagem de programação uma frase é chamada de instrução e corresponde à descrição de uma ou várias operações elementares do computador.

Atualmente, há mais de 2000 linguagens de programação diferentes, a maioria delas utilizadas apenas em situações muito particulares. Para facilitar o estudo, estas linguagens são divididas em classes diferentes, de acordo com suas características principais. Estas diferentes classes são chamadas de paradigmas de programação (SEBESTA, 2010).

TERMO DO GLOSSÁRIO: a sintaxe de uma linguagem de programação é definida por meio de um conjunto de regras através de uma gramática. A notação mais utilizada é a BNF (*Backus-Naur Form*).

Sintaxe: é um conjunto de regras que define a forma de uma linguagem, estabelecendo como são compostas suas estruturas básicas. Por exemplo, ao programar utilizando a linguagem de programação Java é preciso colocar um sinal de ponto-e-vírgula no final de cada instrução (comando). Esta é uma regra de sintaxe da Linguagem Java. As regras de sintaxe de cada linguagem de programação são criadas pelos programadores que desenvolvem tais linguagens e, se não forem seguidas, não permitem que o programa possa ser compilado e executado, ou seja, um programa que não seguir as regras de sintaxe não funciona!

Semântica: descreve o significado de construções sintáticas válidas. Podemos seguir as regras de sintaxe e escrevermos algo que não faz sentido. Por exemplo, ao calcular o salário de um funcionário, podemos criar uma fórmula para multiplicar dois valores: valor da hora e o nome do funcionário.

1.2
CONCEITOS FUNDAMENTAIS

Para entendermos o que é um algoritmo, vamos partir de um exemplo. Queremos escrever um programa que calcule a média do rendimento acadêmico de um aluno. Sabemos que, na UFSM, um aluno precisa atingir a média 7,0 (média aritmética simples) para ser aprovado (sem que seja necessário realizar o exame final), sendo que existem duas notas parciais, que iremos chamar de P1 e P2. Sendo assim, a fórmula para o cálculo da média é a seguinte:

$$\text{Média} = \frac{P1 + P2}{2}$$

A média é igual à nota atribuída à 1ª parte da avaliação (P1) mais a nota atribuída à 2ª parte da avaliação (P2), sendo que este resultado é dividido por 2. Se a média for maior ou igual a 7,0 o aluno está aprovado; caso contrário, ainda poderá realizar a avaliação final (exame final).

A intenção é construirmos um algoritmo que automatize o cálculo da média. O algoritmo é a série de passos que permite que o cálculo da média seja executado pelo computador. Basicamente, teríamos os seguintes passos:

1. obter as notas de P1 e P2;
2. aplicar a fórmula da média;
3. verificar a média calculada.

Estes passos estão descritos em **linguagem natural**. Cada um pode descrever os passos utilizando as palavras e a forma que achar melhor. Em programação isto não é possível, pois, como vimos anteriormente, uma **linguagem de programação** é composta por um conjunto bem definido de símbolos permitidos e regras de escrita (sintaxe) e regras de comportamento (semântica), ou seja, para escrevermos um programa que automatize o cálculo da média, precisaremos seguir estas regras, caso contrário o programa não funcionará, ou seja, não será executado.

SAIBA MAIS: Quem pode criar uma linguagem de programação? Qualquer programador pode criar sua própria linguagem, desde que tenha tempo e conhecimento suficientes para isso.

Uma das primeiras regras que precisamos entender é a utilização de informações **variáveis** e **constantes**. As informações variáveis precisam ficar armazenadas em posições da memória do computador ("gavetas") para que possam ser encontradas pelo programa. As variáveis são as informações que mudam no decorrer do tempo.

No caso do programa que calcula a média, as informações variáveis são as notas de P1 e P2 e o resultado (média calculada). A Figura 1 apresenta, de forma gráfica, como as variáveis P1, P2 e média poderiam estar dispostas na memória do computador. Na figura foram representadas algumas gavetas vazias (podem existir inúmeras gavetas vazias, sendo limitadas à capacidade da memória utilizada).

FIGURA 1 – Representação gráfica das variáveis na memória do computador.

FONTE: dos autores, adaptado por NTE, 2017

As informações constantes não variam durante a execução do programa. Por exemplo, no cálculo da média, independentemente da nota obtida pelo aluno, o resultado é sempre dividido por 2, pois estamos trabalhando com duas avaliações parciais para compor a média do aluno.

As informações variáveis, além de precisarem de um espaço de memória, precisam ter um tipo de dado associado, ou seja, o computador precisa saber qual é o tipo de informação que cada uma das variáveis pode manipular. No início do algoritmo, o programador precisa **declarar as variáveis** que serão utilizadas no seu programa. O processo de declaração de variáveis consiste em definir um nome para as variáveis e o tipo de dado associado.

O nome de uma variável é um identificador (também conhecido como ID – *identifier*) e não muda durante a execução de um programa. Para definir o nome de uma variável devem ser seguidas algumas regras:

– não deve começar por um número;
– não deve conter espaços em branco;
– não deve conter caracteres especiais (+ - * / % $ # @ !), exceto o sublinhado (_), que é utilizado para separar os nomes das variáveis;
– não deve conter nenhum caractere de acentuação;
– não pode ser uma palavra reservada da linguagem de programação (um comando – da linguagem, por exemplo, é uma palavra reservada).

SAIBA MAIS: Palavra Reservada: os comandos de uma linguagem de programação são palavras reservadas e não podem ser utilizados como nomes de variáveis. Por exemplo, utilizando-se a linguagem do VisuAlg, que será estudada neste livro, uma das palavras reservadas é o comando (ou instrução) escreva que permite que sejam impressas mensagens na tela. Sendo assim, a palavra reservada escreva não pode ser utilizada como um nome de variável pelo programador.

Seguem alguns exemplos de nomes de variáveis válidos:

– Endereco (sem o ç que é um caracter especial)
– Idade
– Nome
– Nome1
– Nome_do_Cliente
– P1
– Nota_P1

Seguem alguns exemplos de nomes de variáveis inválidos:

– Nome do Aluno (contém espaços em branco)
– 1P (inicia por um número)
– Endereço (contém o caracter cedilha)
– escreva (é uma palavra reservada da linguagem, neste caso da linguagem de programação utilizada no ambiente VisuAlg)

Além do nome, na declaração de variáveis precisamos definir o tipo de dado que cada variável irá manipular. O VisuAlg (ambiente que será utilizado nesta disciplina) permite os seguintes tipos de dados:

– *Inteiro*: qualquer informação do tipo numérica inteira (a idade de uma pessoa, por exemplo);

– *Real*: informação do tipo numérica que precisa ser armazenada com casas decimais (vírgulas), tais como o salário de uma pessoa (que precisa ser armazenado com os valores correspondentes aos centavos) e a média de um aluno;

– *Cadeia de Caracteres*: qualquer caracter (letras, números e caracteres especiais), tais como o nome de uma pessoa, o endereço onde mora, etc.;

– *Logico*: o tipo *logico* permite os valores *verdadeiro* e *falso*.

ATENÇÃO: A palavra reservada *logico* deve ser escrita sem acento.

Antes de iniciar a codificação de um algoritmo precisamos, por meio da análise do problema, identificar as variáveis que serão necessárias (espaços de memória que serão necessários para armazenar os dados manipulados pelo algoritmo). Estes dados podem ser divididos em **dados de entrada** e **dados de saída**.

Os **dados de entrada** são todas as informações que um programa precisa para chegar ao resultado. Estas informações são fornecidas por meio dos **periféricos de entrada.** O periférico de entrada mais comum é o teclado (o usuário digita as informações solicitadas pelo programa). Atualmente, com o uso de *smartphones* e o avanço dos dispositivos móveis, a tela do celular é muito utilizada como periférico de entrada, por meio da função *touch screen* ou tela de toque.

TERMO DO GLOSSÁRIO: *Touch Screen*: a tela sensível ao toque é um dispositivo eletrônico visual que pode detectar a localização de um toque dentro de uma área de exibição. Além de reconhecer o toque com o dedo ou a mão, uma tela sensível ao toque pode reconhecer objetos, tais como uma caneta (muito comum em tablets). Atualmente, a maioria dos smartphones possuem telas sensíveis ao toque.

Os **dados de saída** são os resultados obtidos por meio da execução de um programa.

No caso do algoritmo para calcular a média, os dados de entrada são as notas de P1 e P2 e o resultado é a média do aluno. Sendo assim, são necessários três espaços de memória para armazenar os dados. As três variáveis manipulam dados numéricos.

A declaração das variáveis para o algoritmo da média poderia ser a seguinte:

```
Var
   P1, P2, Media: Real
```

ATENÇÃO: A palavra reservada *Var* indica a seção de declarações de variáveis em um algoritmo, seguindo a sintaxe do VisuAlg. Cada ambiente e/ou linguagem de programação tem suas regras de sintaxe próprias. Sendo assim, a palavra var pode ou não ser utilizada em outros ambientes.

Lembre-se que os nomes das variáveis não podem conter acentos e caracteres especiais

Algumas observações importantes com relação às variáveis:

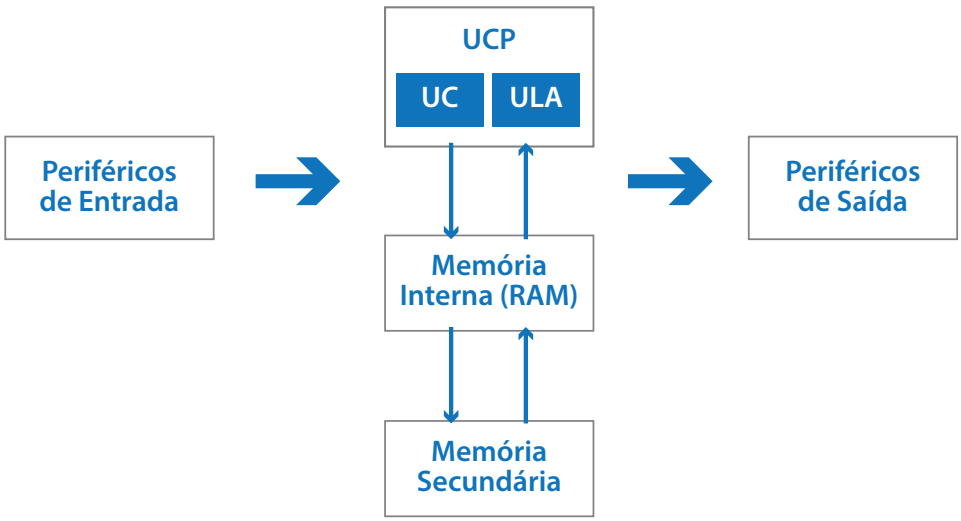
1) o uso do nome de uma variável, dentro de uma expressão (fórmula), não altera o seu valor. Para alterar o valor de uma variável na memória é preciso utilizar um comando da linguagem de programação que é o comando de atribuição (comando que indica que desejamos atribuir ou definir um valor para a variável, ou seja, que queremos modificar o valor que está armazenado na memória cujo espaço é representado pelo nome desta variável);

2) o uso do nome de uma variável, dentro de uma expressão (fórmula) significa que está sendo recuperado o seu conteúdo na memória;

3) uma variável recebe o valor que lhe está sendo atribuído, perdendo o valor anteriormente armazenado (se guardamos, anteriormente, o valor 7 na variável P1 e, por meio de um comando de atribuição, damos ordem para que seja guardado o valor 8, a variável P1 manterá apenas o último valor, o 8, ou seja, uma variável armazena apenas um valor de cada vez, não sendo possível resgatar um histórico dos valores armazenados. Caso precisemos armazenar mais de um valor utilizando um mesmo nome de variável, precisaremos utilizar uma estrutura de dados, tais como os vetores, que serão estudados mais adiante neste livro).

Esta declaração de variáveis informa ao computador que o programa precisa de 3 espaços de memória (como se fossem 3 gavetas), que serão reservados durante a execução do programa. Este processo é denominado de **alocação de memória**. Reservar os espaços (gavetas) significa que outros programas e outras variáveis não poderão usar aqueles mesmos espaços, para que não haja o problema de conflito entre os dados, ou seja, dados de diferentes programas compartilhando as mesmas gavetas. A Figura 2 apresenta um esquema básico de funcionamento do *hardware*.

FIGURA 2 – Esquema Básico de Funcionamento do Hardware.



FONTE: dos autores, adaptado por NTE, 2017

Conforme apresenta a Figura 2, baseada na arquitetura de *Von Neumann*, temos as 3 fases do processamento de dados (entrada, processamento e saída) representadas, respectivamente: 1) pelos periféricos de entrada; 2) Unidade Central de Processamento e Memória e 3) periféricos de saída.

SAIBA MAIS: Na arquitetura de computadores proposta por John von Neumann tanto os dados quanto os programas são armazenados na mesma memória e a CPU é separada da memória.

1) Os periféricos de entrada são todos os dispositivos que permitem que uma informação do meio externo seja enviada para o computador. Por exemplo, quando utilizamos um caixa eletrônico em um banco, podemos enviar informações de diferentes formas. Quando inserimos o cartão magnético, as informações são lidas por meio de uma leitora de cartões. Quando digitamos nossa senha, podemos usar o teclado ou o vídeo (*touch screen*);

2) O processamento é executado por meio de duas unidades principais, que são a UCP (Unidade Central de Processamento) e a Memória. A UCP tem dois componentes principais, que são a UC (Unidade de Controle) e a ULA (Unidade Lógico-Aritmética). A UC é responsável por acompanhar o fluxo de execução do algoritmo/programa, indicando qual é o próximo comando a ser executado. A ULA, por sua vez, é responsável por interpretar as expressões aritméticas e lógicas. A memória é responsável por armazenar o algoritmo (ou programa), bem como as variáveis utilizadas. Quando o programa é colocado em execução, os valores das variáveis são enviados à UCP para serem utilizados nas expressões e são devolvidos à memória quando os resultados são calculados. Por isso é que existem setas indicando a mão dupla entre a Memória e a UCP. Com relação à Memória, a figura ilustra as memórias interna e secundária. A memória interna (ou Memória RAM) armazena as instruções do programa que está sendo executado, bem como as variáveis que estão sendo utilizadas. Esta memória é volátil, ou seja, o programa, bem como as variáveis, só ficam armazenados enquanto o programa estiver sendo executado. O código-fonte do programa precisa ser armazenado (salvo) em uma memória secundária, para que possa ser executado outras vezes. Como exemplos de memórias secundárias temos o HD (*Hard Disk* ou Disco Rígido) e o *pendrive*.

TERMO DO GLOSSÁRIO: UCP (Unidade Central de Processamento) ou CPU (Central Process Unit): popularmente conhecida como o cérebro do computador; componente onde todas as operações são executadas.

A UC (Unidade de Controle) acessa, decodifica e executa as instruções sucessivas de um programa armazenado na memória.

A ULA (Unidade Lógico-Aritmética) executa e fornece os resultados das operações lógicas e aritméticas.

Memória Secundária: permite o armazenamento "permanente" de dados.

SAIBA MAIS: Memória RAM (*Random Access Memory*) ou Memória de Acesso Aleatório: em um computador baseado na arquitetura de von Neumann, a memória principal, onde ficam armazenados o programa e os dados manipulados pelo mesmo, é constituída por RAM. Esta memória é volátil, ou seja, seus dados são perdidos quando o computador é desligado.

3) O resultado do processamento é apresentado por meio de periféricos de saída. Os periféricos de saída mais comuns são o vídeo (ou tela) e a impressora. Por exemplo, após executar o algoritmo que calcula a média do aluno, a média é armazenada na memória. Se o programador (no caso, você, aluno) não incluir

no algoritmo/programa um comando para apresentar a média na tela, o usuário final (quem irá utilizar o seu programa) não visualizará o resultado.

Detalhando ainda mais a execução de um algoritmo, vimos que envolve 3 etapas:

1) entrada de dados: nesta etapa, os dados são transmitidos de um meio externo para o computador. Normalmente, esta etapa é realizada por meio de um **periférico de entrada**. O periférico de entrada mais comum é o teclado, mas existem outros, tais como: *scanner*, leitora de códigos de barras, *touch screen*;
2) processamento dos dados: esta etapa consiste na execução dos cálculos do programa, por meio da utilização dos dados armazenados na memória e da UCP (Unidade Central de Processamento ou CPU – *Central Process Unity*). A memória interna e a UCP se comunicam, permitindo que os dados sejam enviados à ULA (Unidade Lógico e Aritmética), processados e devolvidos para a memória, já com os resultados;
3) saída de dados: esta etapa permite que os resultados obtidos sejam apresentados aos usuários, através de periféricos de saída. Os **periféricos de saída** mais comuns são o vídeo e a impressora.

No exemplo do algoritmo para calcular a média, estas 3 etapas ficariam assim distribuídas:

1) por meio de um periférico de entrada, como o teclado, o usuário informaria as notas de P1 e P2; estas notas seriam armazenadas na memória interna, nos espaços reservados;
2) por meio da ULA, a média seria calculada de acordo com a fórmula adequada;
3) por meio de um periférico de saída, como o vídeo, o usuário receberia o resultado, ou seja, a sua média.

Neste ponto, cabe uma pergunta: Como é que o computador sabe que deve armazenar as notas de P1 e P2 na memória, calcular a média e depois mostrar o resultado? Estas ordens (comandos ou instruções) estão descritos no nosso algoritmo (programa). Este programa deve ser armazenado na memória do computador e, quando colocado em execução, aciona os dispositivos do *hardware*, necessários para o funcionamento do programa. Sendo assim, a memória, além de armazenar os dados (de entrada e de saída), também armazena o programa (a série de passos que deve ser seguida para que se chegue à solução do problema).

1.3 FORMAS DE REPRESENTAÇÃO DE ALGORITMOS

Esta subunidade apresenta algumas formas para representar algoritmos (passos para resolver um determinado problema).

1.3.1 Construção de um Modelo da Solução de um Problema

Um modelo é a representação das instruções que levam à solução de um problema. A partir da análise da definição de um problema, são obtidas as informações para a construção do modelo, que podem ser resumidas nas respostas das 3 perguntas abaixo:

1) Quais os dados do problema?
2) Qual o resultado esperado?
3) O que é necessário para se obter o resultado?

A partir destas respostas, se constrói o modelo da solução. Existem técnicas para sistematizar a construção de um modelo, visando facilitar o desenvolvimento do programa-fonte. As técnicas mais utilizadas são o fluxograma (ou diagrama de blocos) e os algoritmos.

1.3.2 Fluxograma

Um fluxograma é a representação gráfica da solução de um problema. É o modelo em linguagem simbólica ou gráfica. Um fluxograma é formado por figuras geométricas ligadas por setas. Os símbolos mais utilizados em um fluxograma são apresentados no Quadro 1.

QUADRO 1 – Símbolos mais utilizados em um fluxograma

FONTE: dos autores, adaptado por NTE, 2017

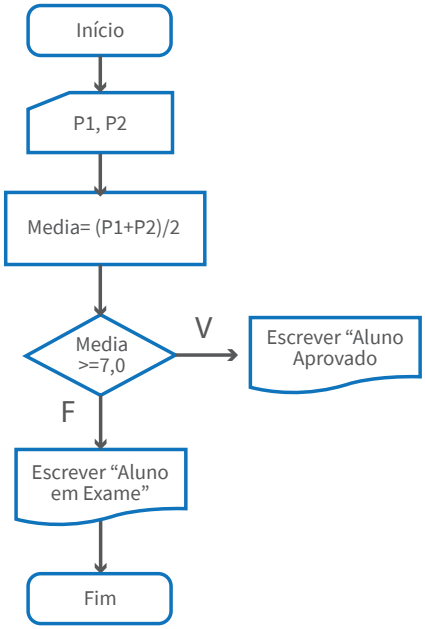
Utilizando nosso exemplo de programa para calcular a média de um aluno, podemos representar o modelo da solução da seguinte forma:

Problema: Dadas 2 notas (P1 e P2) de um aluno, calcular a média, sabendo-se que devemos utilizar a média aritmética simples. Se o aluno obtiver média maior ou igual a 7,0 está aprovado, caso contrário, deve realizar o exame final.

1) Dados: duas notas (P1 e P2)
2) Resultado esperado: média do aluno
3) O que é necessário: declaração de 3 variáveis: duas para as notas (P1 e P2) - dados de entrada e uma para o resultado (Média).

A Figura 3 apresenta o fluxograma do algoritmo para calcular a média.

FIGURA 3 – Fluxograma do Algoritmo para Calcular a Média.



FONTE: dos autores, adaptado por NTE, 2017

1.3.3 Algoritmos

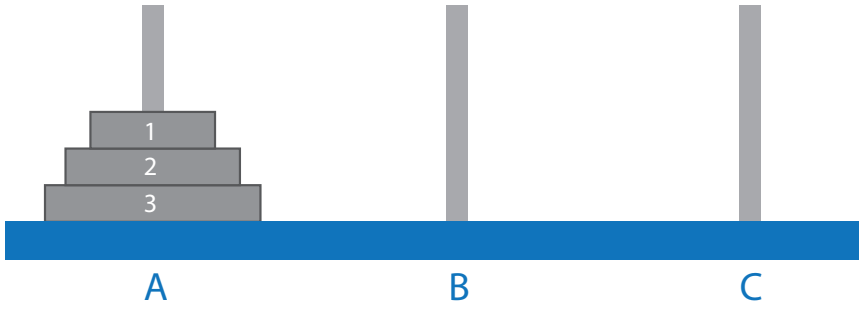
Um algoritmo é a representação descritiva da solução de um problema. É formado por um conjunto de instruções (comandos) que representam a solução de um problema. Podem ser escritos em linguagem natural (a linguagem que utilizamos para falar e escrever) ou em pseudocódigo.

Como exemplos de algoritmos descritos em linguagem natural, podemos citar o conjunto de instruções para instalação e utilização de um eletrodoméstico ou uma receita culinária. Estes algoritmos não precisam de regras para escrita, como as utilizadas nos algoritmos em pseudocódigo.

Por exemplo, se quisermos fazer um bolo precisaremos de uma lista de ingredientes (poderíamos considerar estes ingredientes como os dados de entrada), o modo de fazer (processamento) e teremos o bolo pronto (resultado, como se fossem nossos dados de saída).

Vamos escrever, como exemplo, um algoritmo, em linguagem natural, que nos permita mover 3 discos de uma Torre de Hanói (Discos 1, 2 e 3), que consiste em 3 hastes (A, B e C), uma das quais serve de suporte para 3 discos de tamanhos diferentes, sendo os menores sobre os maiores, como mostra a Figura 4. Devemos seguir algumas regras para os movimentos, que são: podemos mover um disco de cada vez para qualquer haste, contanto que nunca seja colocado um disco maior sobre um menor. O objetivo é transferir os 3 discos para uma outra haste. Alguns exemplos que você pode encontrar na literatura e/ou na Internet podem conter mais do que 3 discos.

FIGURA 4 – Torre de Hanói.



FONTE: NTE, 2017

Uma das soluções possíveis para resolver este problema é a seguinte (algoritmo escrito em linguagem natural):

1) Mover o disco 1 da haste A para a haste B
2) Mover o disco 2 da haste A para a haste C
3) Mover o disco 1 da haste B para a haste C
4) Mover o disco 3 da haste A para a haste B
5) Mover o disco 1 da haste C para a haste A
6) Mover o disco 2 da haste C para a haste B
7) Mover o disco 1 da haste A para a haste B

Você pode encontrar soluções diferentes para resolver o problema, desde que não infrinja as regras anteriormente estabelecidas. Como escrevemos o algoritmo (série de passos para resolvermos o problema) em linguagem natural, as regras de sintaxe são as da Língua Portuguesa. Você pode escrever *mover, mova, retire, desempilhe,* pois o que importa é que alguém que conheça a Língua Portuguesa consiga, usando o seu algoritmo, resolver o problema. Quando vamos realizar a programação de computadores de fato, precisaremos seguir as regras de sintaxe e de semântica estabelecidas pela linguagem de programação que iremos utilizar, ou seja, a nossa liberdade de criação na escrita do código-fonte (codificação) ficará limitada às regras da linguagem que optarmos por utilizar. Por exemplo, utilizando o VisuAlg precisaremos seguir as regras de sintaxe e semântica estabelecidas pelos desenvolvedores (programadores) deste ambiente, caso contrário nosso algoritmo (programa) não será executado.

Vamos ver um outro exemplo de algoritmo em linguagem natural, este mais voltado à solução de um problema computacional. Sabendo-se que 100 quilowatts de energia custam um sétimo do salário mínimo, escrever um algoritmo que leia o valor do salário mínimo e a quantidade de quilowatts gasta em uma residência. Com base nestes valores, devemos calcular e escrever:

– o valor em R$ de cada quilowatt
– o valor em R$ a ser pago referente à conta mensal
– o valor a ser pago aplicando-se um desconto de 10%

Uma solução possível para este algoritmo, em linguagem natural, é a seguinte:

1. Obter o valor do salário mínimo atual
2. Obter a quantidade de quilowatts gasta no mês atual na residência
3. Calcular o valor referente à 1 quilowatt, aplicando a fórmula valor unitário do quilowatt = salário mínimo dividido por 7 e este resultado dividido por 100
4. Calcular o valor da conta mensal, aplicando a fórmula conta mensal = valor unitário do quilowatt multiplicado pela quantidade de quilowatts gasta na residência
5. Calcular o valor da conta mensal com desconto de 10%, por meio da fórmula conta mensal com desconto = conta mensal – (conta mensal multiplicada por 10%)
6. Escrever o valor em R$ de cada quilowatt
7. Escrever o valor da conta mensal
8. Escrever o valor da conta mensal com desconto de 10%

Um algoritmo em pseudocódigo é a representação da solução de um problema por meio de instruções em pseudocódigo, ou seja, instruções voltadas para uma linguagem de programação.

A Figura 5 apresenta o algoritmo para calcular a média, em pseudocódigo, utilizando-se a sintaxe do VisuAlg.

FIGURA 5 – Algoritmo em Pseudocódigo

```
algoritmo "calculamedia"
// Função : Calcular a média de um aluno
// Autor :  Autores do Livro de Introdução a Algoritmos
// Data : 06/04/2017
// Seção de Declarações
var
  P1, P2, media: real
inicio
// Seção de Comandos
// Entrada de Dados
  escreva("Digite a nota de P1:")
  leia(P1)
  escreva("Digite a nota de P2:")
  leia(P2)
//Processamento
  media <- (P1+P2)/2
//Saída de Dados
  escreva("Média do aluno:",media)
fimalgoritmo
```

FONTE: dos autores, adaptado por NTE, 2017

Com base neste exemplo verificamos que existem duas seções básicas em um algoritmo: 1) a seção de declarações, onde são declaradas ou definidas as variáveis, por meio da palavra reservada *var*. Nesta seção devem ser declarados os dados de entrada e saída que serão utilizados no programa para que sejam reservados

espaços na memória para armazená-los e 2) a seção onde são descritos os passos necessários para a execução do algoritmo (comandos ou instruções). Esta seção é delimitada pelas palavras *inicio...fimalgoritmo*.

Além destas seções, o algoritmo apresentado como exemplo utiliza alguns comandos (ordens ou instruções). Estes comandos são as **palavras reservadas** da linguagem: *escreva, leia* e o comando de atribuição, representado pelo símbolo <-. O símbolo <- (sinal de menor mais o sinal de menos) indica que a variável que está do lado esquerdo do símbolo irá receber o valor ou o resultado da expressão que está do lado direito (veremos as expressões na próxima unidade).

As linhas que iniciam por duas barras (//) são opcionais, pois são comentários. Você, quando for criar seus algoritmos poderá decidir quantos e quais comentários irá inserir. Quando o código-fonte é composto por um número reduzido de linhas pode parecer que os comentários não possuem muita utilidade. Entretanto, à medida que o número de linhas de código aumenta, bem como a complexidade das linguagens de programação a serem utilizadas, os comentários podem ajudar a explicar o que e como o desenvolvedor resolveu o problema.

Cada um destes comandos tem uma função específica que será vista na unidade seguinte. Os comandos possuem regras de escrita (**regras de sintaxe**), que devem ser seguidas para que executem as ações determinadas. Se forem escritos na forma incorreta o programa não será executado. Por exemplo, a sintaxe do comando leia, que serve para a entrada de dados é leia(<*lista-de-variáveis*>).

A Figura 6 apresenta o algoritmo para calcular o valor dos quilowatts em pseudocódigo, considerando a linguagem do VisuAlg (algoritmo que escrevemos, anteriormente, em linguagem natural).

FIGURA 6 – Algoritmo em Pseudocódigo – Cálculo do Valor dos Quilowatts

```
algoritmo "calculamedia"
// Função : Calcular a média de um aluno
// Autor :  Autores do Livro de Introdução a Algoritmos
// Data : 06/04/2017
// Seção de Declarações
var
  P1, P2, media: real
inicio
// Seção de Comandos
// Entrada de Dados
  escreva("Digite a nota de P1:")
  leia(P1)
  escreva("Digite a nota de P2:")
  leia(P2)
//Processamento
  media <- (P1+P2)/2
//Saída de Dados
  escreva("Média do aluno:",media)
fimalgoritmo
```

FONTE: dos autores, adaptado por NTE, 2017

2

OPERADORES E TIPOS DE DADOS

INTRODUÇÃO

Para desenvolver algoritmos que se aproximem do código-fonte de uma linguagem de programação, não é possível que façamos a escrita dos mesmos em linguagem natural. Atualmente ainda não existe nenhum ambiente de programação de computadores que consiga interpretar a linguagem natural. Sendo assim, precisamos escrever os algoritmos utilizando um pseudocódigo (linguagem mais próxima das linguagens de programação). Existem vários ambientes de programação e cada um deles conta com suas regras de sintaxe e semântica próprias, de acordo com o pseudocódigo utilizado. Entre estas regras encontram-se as formas para descrever expressões aritméticas e lógicas.

As expressões aritméticas são muito utilizadas no processamento das informações, para que sejam realizados diversos cálculos, tais como: média de um aluno, salário líquido de um funcionário, valor do imposto de renda, entre outros. As expressões aritméticas permitem que nossos algoritmos realizem operações de soma, subtração, multiplicação, divisão e exponenciação, entre outras.

Além das expressões aritméticas, também existem as expressões lógicas, que já foram estudadas na disciplina de *Lógica Matemática* (BERTOLINI; CUNHA; FORTES, 2017). As expressões lógicas permitem que nossos algoritmos possam "tomar decisões". Por exemplo, vamos supor que uma pessoa que ganha até R$1.500,00 mensais não deve pagar imposto de renda e, para os demais valores (acima de R$1.500,00), a alíquota de imposto de renda é de 10%. Para fazermos um algoritmo para calcular o imposto de renda de uma pessoa, então, precisamos tomar uma decisão. Se o valor for menor ou igual a R$1.500,00 o valor do imposto é zero. Caso contrário (para valores maiores do que R$1.500,00), precisaremos utilizar uma expressão aritmética para calcularmos o valor referente aos 10% da alíquota. Esta expressão "Se o salário de uma pessoa é menor ou igual a R$1.500,00" é uma expressão lógica, pois o resultado pode ser V (Verdadeiro) ou F (Falso).

Além das expressões aritméticas e lógicas, a unidade aborda as expressões literais, que envolvem uma sequência de caracteres, por exemplo, como manipular, em um algoritmo, o nome de uma pessoa, o endereço, o nome do curso em que está matriculado, entre outros.

2.1
USO DE EXPRESSÕES EM ALGORITMOS

Uma expressão representa a manipulação, ou seja, o processamento de informações na resolução de um problema. As expressões podem ser **aritméticas**, **literais** ou **lógicas**.

2.1.1 Expressões Aritméticas

As expressões aritméticas são formadas pela combinação de constantes numéricas, variáveis numéricas e/ou funções matemáticas ligadas pelos operadores aritméticos. Os operadores aritméticos disponibilizados pelo VisuAlg são:

+	adição
-	subtração
*	multiplicação
/	divisão
^	potenciação
mod ou %	resto da divisão inteira
\	parte inteira de uma divisão

Para determinar a prioridade na execução das operações dentro de uma expressão, são utilizados parênteses. É executado o conteúdo do par de parênteses mais interno em primeiro lugar.

Toda expressão é executada da esquerda para a direita, obedecendo a seguinte ordem de prioridade:

1) parênteses mais internos;
2) funções matemáticas;
3) potenciação ou raiz quadrada;
4) multiplicação, divisão, resto, inteiro;
5) adição ou subtração.

2.1.2 Expressões Literais

As expressões literais são formadas por constantes e/ou variáveis alfanuméricas, ligadas por um operador literal. O operador literal mais comum é o de concatenação, que consiste em juntar duas ou mais cadeias de caracteres. No VisuAlg o operador de concatenação é o sinal de +.

Por exemplo, supondo que temos duas variáveis do tipo caractere (literal), conforme exemplo a seguir:

Cidade <- “Frederico Westphalen”
Estado <- “RS”

Os dois exemplos utilizam o operador de atribuição (<-). Segundo a semântica deste operador, estamos dizendo que a variável Cidade *recebe* o valor “Frederico Westphalen” e a variável Estado *recebe* o valor “RS”.

Se aplicarmos o operador de concatenação, teremos “Frederico Westphalen RS”:

Localidade <- Cidade + Estado

Podemos concatenar diretamente com os literais, sem utilizarmos variáveis:

Localidade <- “Frederico Westphalen” + “RS”

2.1.3 Expressões Lógicas

As expressões lógicas são formadas por uma ou mais relações. Uma relação é a comparação entre duas expressões, ligadas pelos operadores relacionais. Os operadores relacionais utilizados no VisuAlg são:

>	maior que
<	menor que
>=	maior ou igual a
<=	menor ou igual
=	igual
<>	diferente

O resultado de uma expressão lógica é sempre uma constante lógica (**verdadeiro** ou **falso**). As expressões lógicas podem ser **simples** ou **compostas**. As expressões lógicas simples são aquelas que possuem a seguinte forma geral: *Expressão-Operador relacional-Expressão* como mostram os exemplos no Quadro 2.

QUADRO 2 – Exemplos de Expressões Lógicas Simples

EXPRESSÃO LÓGICA	RESULTADO DA AVALIAÇÃO DA EXPRESSÃO
5>2	Verdadeiro
5+4<7	Falso
A>C	O resultado será verdadeiro se o valor da variável A for maior do que o valor da variável C; caso contrário o resultado será falso
A*B<10	O resultado será verdadeiro se o valor da variável A multiplicado pela variável B for maior que 10; caso contrário o resultado será falso
“Maria” <> “João”	Verdadeiro

FONTE: dos autores, adaptado por NTE, 2017

As *expressões lógicas* compostas são aquelas que agregam mais de uma expressão lógica na mesma linha de comando. Para isso, precisam de um operador lógico que compare o resultado destas expressões lógicas, como mostram os exemplos do Quadro 3.

QUADRO 3 – Exemplos de Expressões Lógicas Composta

EXPRESSÃO LÓGICA	RESULTADO DA AVALIAÇÃO DA EXPRESSÃO
(5 > 2) E (5+4 < 7)	Falso
("Maria" <> "João") OU (5 * 2 < 8)	Verdadeiro

FONTE: dos autores, adaptado por NTE, 2017

No VisuAlg existem os operadores lógicos **NAO, E, OU e XOU**. Os operadores lógicos obedecem as tabelas de resultados apresentados nos Quadros 4, 5, 6 e 7.

QUADRO 4 – Tabela de Resultados do Operador Lógico NAO

RESULTADO DA EXPRESSÃO A	NAO(A)
Verdadeiro	Falso
Falso	Verdadeiro

FONTE: DOS AUTORES, ADAPTADO POR NTE, 2017

QUADRO 5 – Tabela de Resultados do Operador Lógico E

RESULTADO DA EXPRESSÃO A	RESULTADO DA EXPRESSÃO B	A E B
Verdadeiro	Verdadeiro	Verdadeiro
Verdadeiro	Falso	Falso
Falso	Verdadeiro	Falso
Falso	Falso	Falso

FONTE: dos autores, adaptado por NTE, 2017

QUADRO 6 – Tabela de Resultados do Operador Lógico OU

RESULTADO DA EXPRESSÃO A	RESULTADO DA EXPRESSÃO B	A E B
Verdadeiro	Verdadeiro	Verdadeiro
Verdadeiro	Falso	Verdadeiro
Falso	Verdadeiro	Verdadeiro
Falso	Falso	Falso

FONTE: dos autores, adaptado por NTE, 2017

QUADRO 7 – Tabela de Resultados do Operador Lógico XOU

RESULTADO DA EXPRESSÃO A	RESULTADO DA EXPRESSÃO B	A E B
Verdadeiro	Verdadeiro	Falso
Verdadeiro	Falso	Verdadeiro
Falso	Verdadeiro	Verdadeiro
Falso	Falso	Falso

FONTE: dos autores, adaptado por NTE, 2017

Em uma expressão lógica é obedecida a seguinte ordem de prioridade:

1) parênteses mais internos;
2) operadores aritméticos;
3) operadores relacionais;
4) operadores lógicos (10 NAO, 20 E e 30 OU).

3

O AMBIENTE VisuAlg E A CRIAÇÃO DE ALGORITMOS SEQUENCIAIS

INTRODUÇÃO

O VisuAlg é um ambiente de aprendizado de programação criado pela empresa Apoio Informática. Os algoritmos construídos são baseados na sintaxe do "Portugol", um pseudocódigo muito utilizado em livros de introdução à programação de computadores. As regras de sintaxe e de semântica da linguagem do VisuAlg foram estabelecidas pelos desenvolvedores do ambiente. No site *apoioinformatica.inf.br/produtos/visualg* é possível realizar o *download* do VisuAlg.

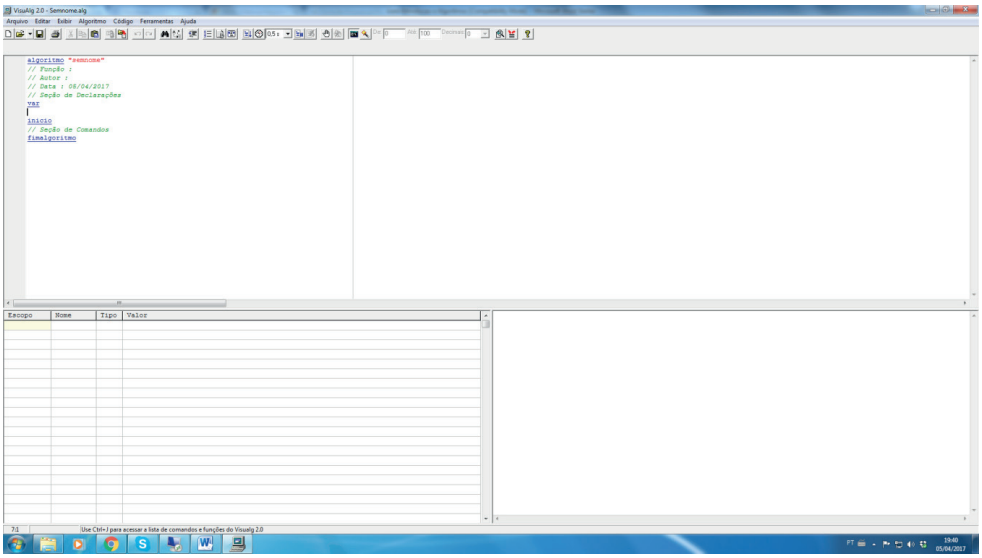
Nesta unidade vamos aprender a utilizar o ambiente VisuAlg, a partir de comandos que permitirão a construção de algoritmos sequenciais, seguindo as etapas de entrada, processamento e saída.

INTERATIVIDADE: Acesse o site da Apoio Informática apoioinformatica.inf.br/produtos/visualg e faça o *download* da ferramenta que iremos utilizar.

3.1
USO DE EXPRESSÕES EM ALGORITMOS

Esta seção apresenta os recursos disponíveis para execução dos algoritmos utilizando o VisuAlg. A Figura 9 mostra a tela principal do VisuAlg.

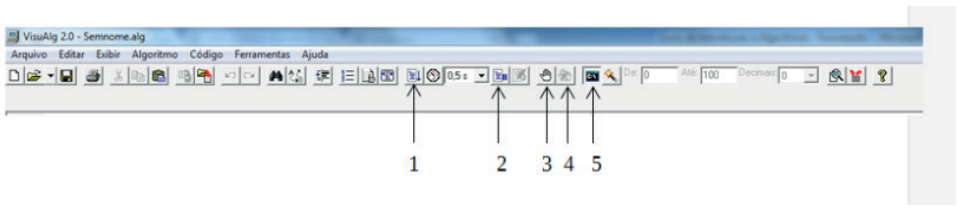
FIGURA 9 – Tela principal do VisuAlg



FONTE: dos autores

A Figura 10 apresenta a barra de ferramentas do VisuAlg. Vamos destacar alguns recursos importantes, que são: Executar (equivalente à tecla F9), Passo (F8), Liga/Desliga *breakpoint* (F5), Desmarcar todos os *breakpoints* (teclas CTRL + F5) e Executar o algoritmo como DOS. Estes recursos são representados, respectivamente, pelos números de 1 a 5 na Figura 10.

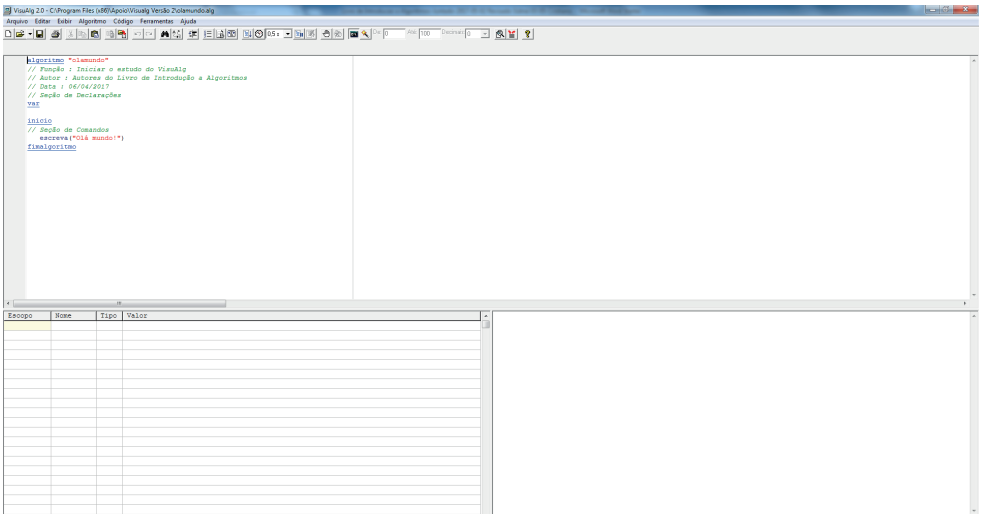
FIGURA 10 – Barra de Ferramentas do VisuAlg



FONTE: dos autores

Para testarmos o funcionamento do VisuAlg, vamos escrever nosso primeiro algoritmo. Quando iniciamos o estudo de uma nova linguagem de programação, geralmente fazemos um programa inicial com a mensagem "Olá mundo!", para dizermos que iniciamos os estudos. Sendo assim, vamos escrever, na seção de comandos, após a palavra reservada inicio, um comando que escreve a mensagem "Olá mundo!" na tela: escreva("*Olá mundo!*"), como mostra a Figura 11.

FIGURA 11 – Algoritmo "Olá Mundo!"

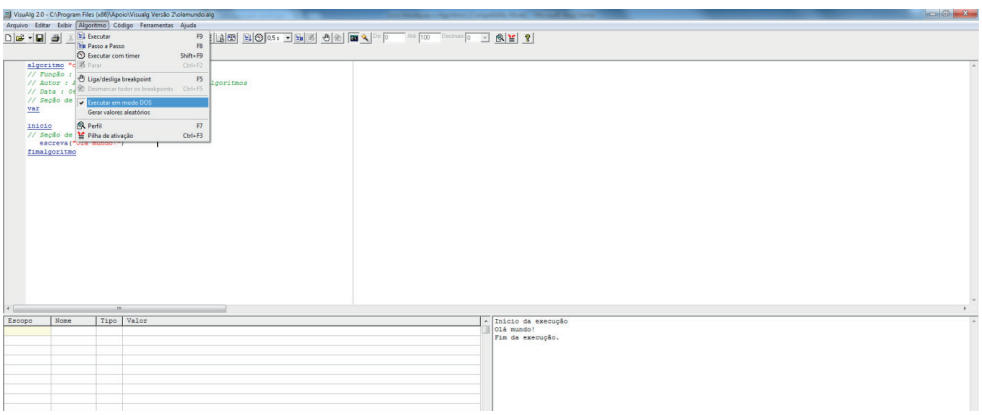


FONTE: dos autores

Após digitarmos o comando, vamos executar o nosso algoritmo, pressionando a tecla F9 (ou clicando no botão Executar apresentado na Figura 10). Veremos, então, a janela apresentada na Figura 13. Para visualizarmos esta tela é preciso configurar a execução do algoritmo para o modo DOS como mostra a Figura 12.

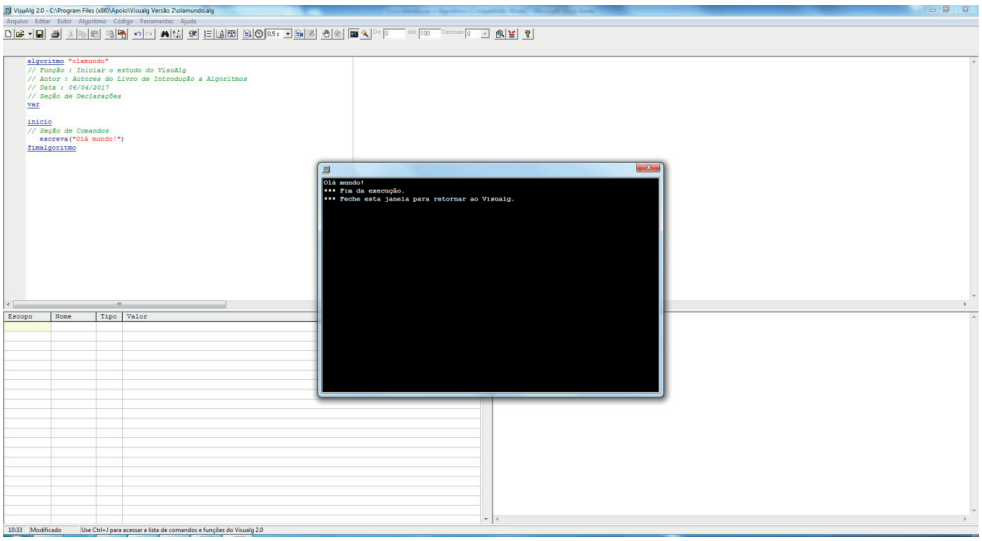
SAIBA MAIS: DOS (*Disk Operating System*) é um Sistema Operacional com interface baseada em caracter.

FIGURA 12 – Configuração do Modo de Execução



FONTE: dos autores

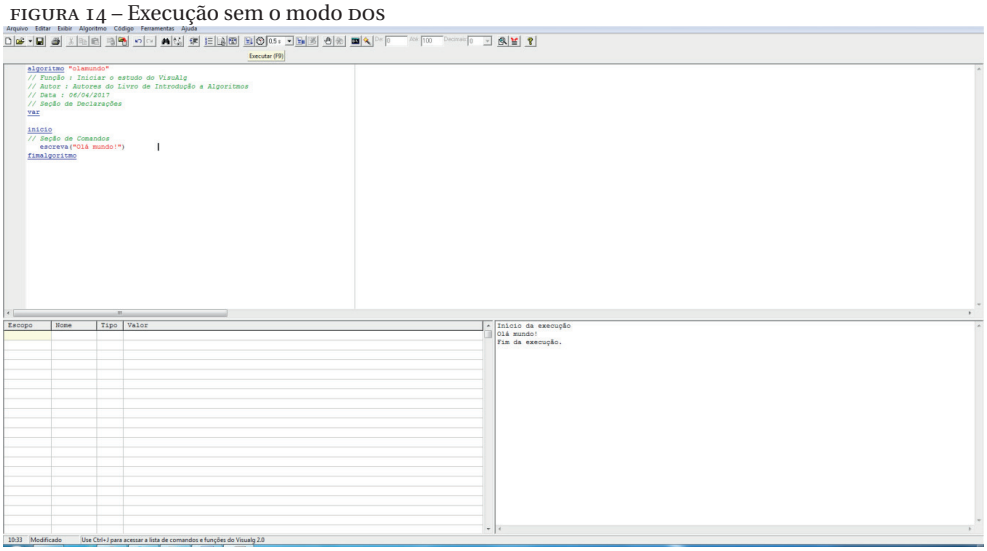
FIGURA 13 – Execução no modo DOS



FONTE: dos autores

A Figura 14 mostra a execução do mesmo algoritmo sem a janela do DOS. Neste caso os resultados da saída são apresentados na parte inferior direita da tela do VisuAlg.

FIGURA 14 – Execução sem o modo DOS



FONTE: dos autores

Vamos digitar o algoritmo para cálculo da média, visto na unidade anterior, para começarmos a nos acostumar com o uso do VisuAlg? O algoritmo é apresentado na Figura 15.

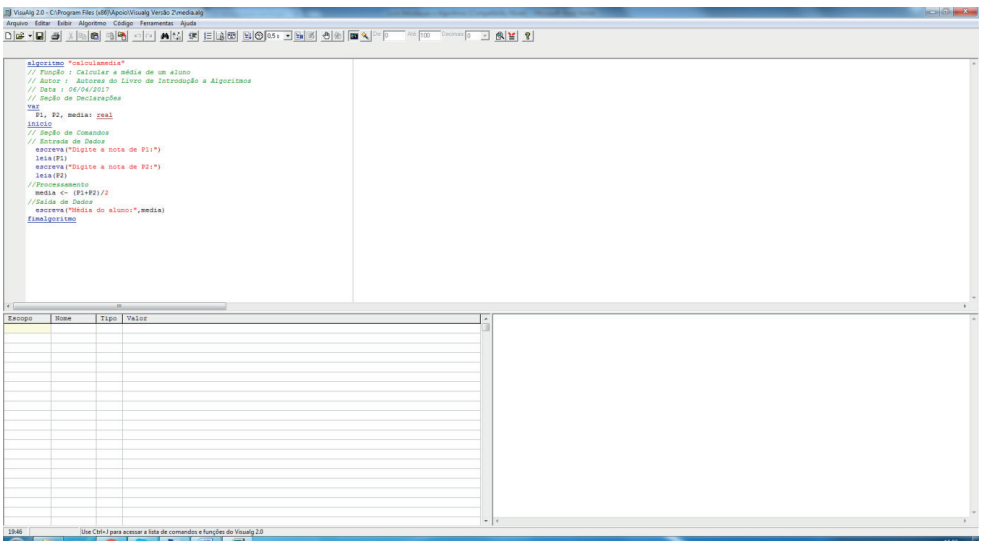
FIGURA 15 – Algoritmo para Calcular a Média

```
algoritmo "calculamedia"
// Função : Calcular a média de um aluno
// Autor :  Autores do Livro de Introdução a Algoritmos
// Data : 06/04/2017
// Seção de Declarações
var
  P1, P2, media: real
inicio
// Seção de Comandos
// Entrada de Dados
  escreva("Digite a nota de P1:")
  leia(P1)
  escreva("Digite a nota de P2:")
  leia(P2)
//Processamento
  media <- (P1+P2)/2
//Saída de Dados
  escreva("Média do aluno:",media)
fimalgoritmo
```

FONTE: dos autores, adaptado por NTE, 2017

A Figura 16 mostra o algoritmo do cálculo da média já digitado no VisuAlg. Os arquivos criados no VisuAlg são salvos com a extensão .alg

FIGURA 16 – Algoritmo para Cálculo da Média

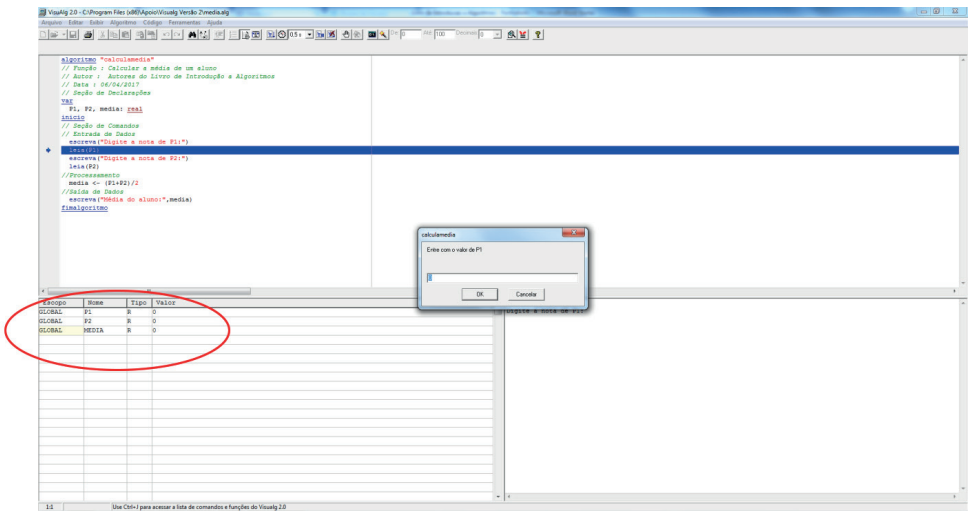


FONTE: dos autores

3.2
EXECUÇÃO PASSO-A-PASSO E TESTE DE MESA

Uma das formas de execução é a passo-a-passo, por meio da qual é possível acompanhar a execução de cada comando do algoritmo e dos valores atuais das variáveis na memória, como em um teste de mesa. A Figura 17 demonstra um algoritmo sendo executado passo-a-passo. Na execução passo-a-passo, deve-se pressionar a tecla F8 (ou utilizar o botão correspondente apresentado na Figura 10) para executar cada uma das linhas do algoritmo. A linha em execução fica destacada na cor azul. Além disso, é possível verificar os valores das variáveis na memória, no lado inferior esquerdo da tela (destacado pelo círculo em vermelho).

FIGURA 17 – Execução Passo-a-Passo



FONTE: dos autores

A execução passo-a-passo pode ser simulada pelo programador sem o uso de uma ferramenta como o VisuAlg. Esta simulação é conhecida como *teste de mesa.* Para um programador que está iniciando o estudo de algoritmos, pode ser difícil verificar se o algoritmo construído realiza realmente a tarefa para a qual foi construído. Um algoritmo só está correto se produz o resultado esperado para qualquer entrada possível realizada pelo usuário. Em um teste de mesa podemos verificar, instrução a instrução, o estado dos dados (variáveis), para confirmarmos se a lógica está correta (MEDINA; FERTIG, 2005). Por exemplo, para realizarmos o teste de mesa do algoritmo que calcula a média, devemos elencar as variáveis no papel, com um espaço para colocarmos os valores das mesmas a cada instrução, bem como um espaço para simularmos o periférico de saída (vídeo). Depois vamos seguir, linha a linha, as

instruções do algoritmo e vamos atualizando os valores das variáveis, simulando a execução. O quadro 8 apresenta uma forma de simular os valores das variáveis na memória e os resultados que são apresentados no vídeo (periférico de saída).

QUADRO 8 – Teste de Mesa do Algoritmo para calcular a média: Variáveis na Memória e Saída

NOME DA VARIÁVEL	VALOR DA VARIÁVEL NA MEMÓRIA (VALOR ARMAZENADO)	VÍDEO (SAÍDA)
P1		
P2		
Media		

FONTE: dos autores, adaptado por NTE, 2017

Para realizarmos o teste de mesa, vamos numerar as linhas do nosso algoritmo, como mostra a Figura 18.

FIGURA 18 – Algoritmo com Linhas Numeradas

```
1. algoritmo "calculamedia"
2. // Função : Calcular a média de um aluno
3. // Autor :  Autores do Livro de Introdução a Algoritmos
4. // Data : 06/04/2017
5. // Seção de Declarações
6. var
7.   P1, P2, media: real
8. inicio
9. // Seção de Comandos
10. // Entrada de Dados
11.   escreva("Digite a nota de P1:")
12.   leia(P1)
13.   escreva("Digite a nota de P2:")
14.   leia(P2)
15. //Processamento
16.   media <- (P1+P2)/2
17. //Saída de Dados
18.   escreva("Média do aluno:",media)
19. fimalgoritmo
```

FONTE: dos autores, adaptado por NTE, 2017

As linhas 2, 3, 4, 5, 9, 10, 15 e 17 (conforme Figura 19), que iniciam por duas barras (//) são comentários, ou seja, serão ignoradas na execução. Vamos iniciar nosso teste de mesa, então, na linha 1, onde temos a instrução que nomeia o algoritmo. As variáveis passarão a existir na memória (alocação de memória) somente quando executarmos a linha 7. Ao executarmos a linha 7 as variáveis serão alocadas na memória (serão reservados 3 espaços) e podemos inicializar seus valores (no caso, como ainda não fizemos nenhuma atribuição de valores, as variáveis, já que são numéricas, serão iniciadas com o valor zero, como mostra o Quadro 9).

QUADRO 9 – Teste de Mesa do Algoritmo para calcular a média: Passo 1

NOME DA VARIÁVEL	VALOR DA VARIÁVEL NA MEMÓRIA (VALOR ARMAZENADO)	VÍDEO (SAÍDA)
P1	0	
P2	0	
Media	0	

FONTE: dos autores, adaptado por NTE, 2017

Seguindo nosso teste de mesa, o próximo comando indica o início da seção de comandos (inicio) na linha 8 e, posteriormente, na linha 11, o comando escreva mostrará uma mensagem na tela, como mostra o Quadro 10.

QUADRO 10 – Teste de Mesa do Algoritmo para calcular a média: Passo 2

NOME DA VARIÁVEL	VALOR DA VARIÁVEL NA MEMÓRIA (VALOR ARMAZENADO)	VÍDEO (SAÍDA)
P1	0	Digite a nota de P1:
P2	0	
Media	0	

FONTE: dos autores, adaptado por NTE, 2017

O próximo comando a ser executado é o de entrada de dados (leia) na linha 12. Como estamos simulando a execução, precisamos informar um valor a ser testado que, após a execução do comando leia será armazenado na memória, na variável correspondente. A linha 12 indica que vamos ler o valor da variável P1. Vamos supor que a nota de P1 é 8,5. O Quadro 11 mostra como ficaria o nosso teste de mesa com este valor.

QUADRO 11 – Teste de Mesa do Algoritmo para calcular a média: Passo 3

NOME DA VARIÁVEL	VALOR DA VARIÁVEL NA MEMÓRIA (VALOR ARMAZENADO)	VÍDEO (SAÍDA)
P1	8,5	Digite a nota de P1: 8,5
P2	0	
Media	0	

FONTE: dos autores, adaptado por NTE, 2017

O próximo comando é o da linha 13, que escreve uma nova mensagem no vídeo (Quadro 12).

QUADRO 12 – Teste de Mesa do Algoritmo para calcular a média: Passo 4

NOME DA VARIÁVEL	VALOR DA VARIÁVEL NA MEMÓRIA (VALOR ARMAZENADO)	VÍDEO (SAÍDA)
P1	8,5	Digite a nota de P1: 8,5 Digite a nota de P2:
P2	0	
Media	0	

FONTE: dos autores, adaptado por NTE, 2017

O próximo comando a ser executado é, novamente, o de entrada de dados (leia) na linha 14, que indica que iremos ler o valor da variável P2. Vamos supor que a nota de P2 é 7,5. O Quadro 13 mostra como ficaria o nosso teste de mesa com este valor.

QUADRO 13 – Teste de Mesa do Algoritmo para calcular a média: Passo 5

NOME DA VARIÁVEL	VALOR DA VARIÁVEL NA MEMÓRIA (VALOR ARMAZENADO)	VÍDEO (SAÍDA)
P1	8,5	Digite a nota de P1: 8,5 Digite a nota de P2: 7,5
P2	7,5	
Media	0	

FONTE: dos autores, adaptado por NTE, 2017

O próximo comando a ser executado é a atribuição, na linha 16, que indica que a variável media irá receber o resultado da expressão aritmética (P1 + P2)/2. Lembre-se que, conforme vimos na unidade 1, os resultados das expressões aritméticas são calculados pela ULA e são devolvidos para a memória, sendo armazenados na variável que está indicada no lado esquerdo da expressão, neste caso a variável media. Como as notas de P1 e P2 são, respectivamente, 8,5 e 7,5, a média aritmética destas duas notas deve resultar em 8,0. Então, nosso teste de mesa deve ficar como mostra o Quadro 14.

QUADRO 14 – Teste de Mesa do Algoritmo para calcular a média: Passo 6

NOME DA VARIÁVEL	VALOR DA VARIÁVEL NA MEMÓRIA (VALOR ARMAZENADO)	VÍDEO (SAÍDA)
P1	8,5	Digite a nota de P1: 8,5 Digite a nota de P2: 7,5
P2	7,5	
Media	8,0	

FONTE: dos autores, adaptado por NTE, 2017

O próximo comando, na linha 18, indica que uma mensagem, bem como o valor armazenado na variável media será mostrado na tela. O resultado da execução deste comando é apresentado no Quadro 15.

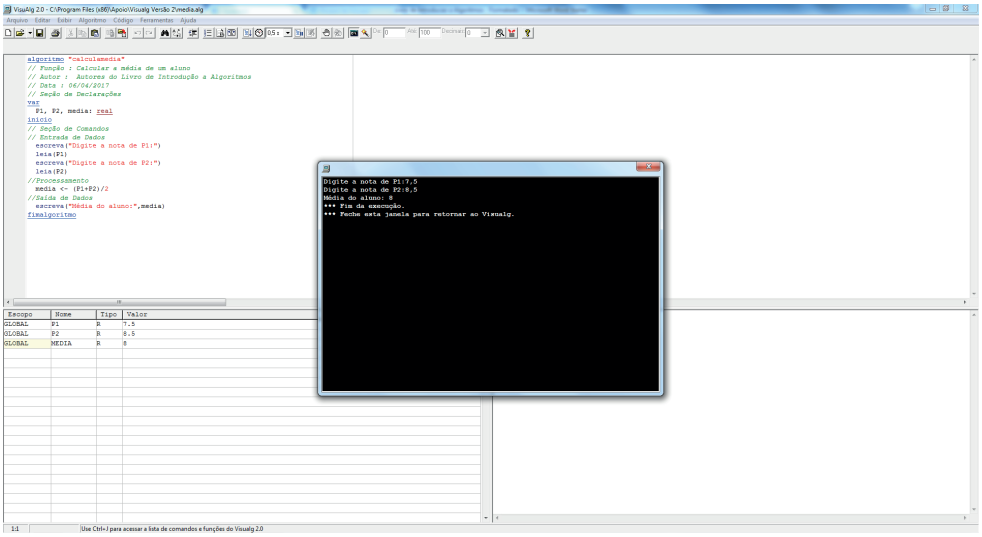
Quadro 15 – Teste de Mesa do Algoritmo para calcular a média: Passo 7

NOME DA VARIÁVEL	VALOR DA VARIÁVEL NA MEMÓRIA (VALOR ARMAZENADO)	VÍDEO (SAÍDA)
P1	8,5	Digite a nota de P1: 8,5 Digite a nota de P2: 7,5 Média do Aluno: 8,0
P2	7,5	
Media	8,0	

FONTE: dos autores, adaptado por NTE, 2017

O próximo comando (linha 19) encerra a execução do algoritmo. Para verificar se o nosso algoritmo está correto podemos executá-lo no VisuAlg utilizando as mesmas notas de P1 e P2 do nosso teste de mesa e verificarmos se a média é apresentada com o mesmo resultado. A Figura 19 mostra o resultado da execução do algoritmo que calcula a média (execução no modo DOS), com os mesmos valores do nosso teste de mesa.

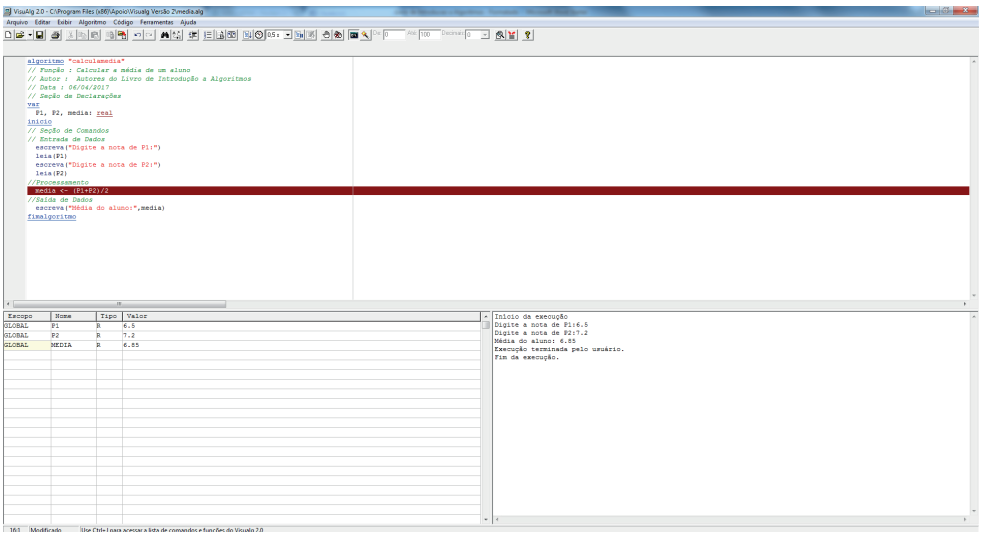
FIGURA 19 – Resultado da Execução do Algoritmo que Calcula a Média



FONTE: dos autores

Outra forma de executarmos um algoritmo passo-a-passo é a utilização de *breakpoints* (ou pontos de parada). Por exemplo, se quisermos verificar os valores das variáveis apenas durante a execução de uma expressão aritmética, podemos colocar um breakpoint na linha que contém o comando. A execução só será iniciada passo-a-passo a partir daquela linha. A Figura 20 mostra o algoritmo que calcula a média com a inserção de um *breakpoint* na linha que calcula a média do aluno. Note que esta linha fica marcada na cor vermelha. Ao executarmos este algoritmo (pressionando a tecla F9), apenas quando a execução chegar na linha marcada pelo *breakpoint* é que poderemos acompanhar passo-a-passo (com a tecla F8).

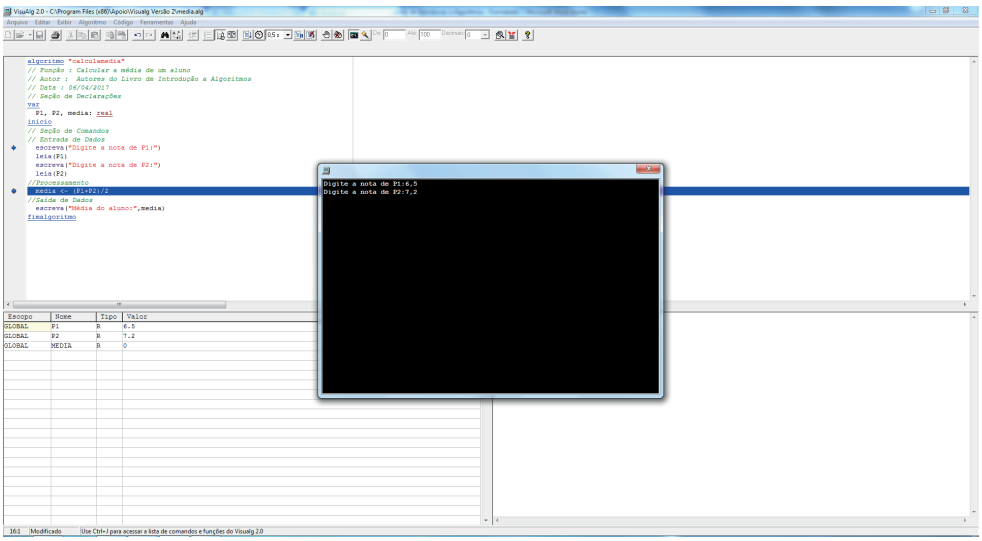
FIGURA 20 – Inserção de *Breakpoint*



FONTE: dos autores

Ao executar o algoritmo (estamos utilizando o modo DOS), antes de chegar à linha do *breakpoint* a execução segue normalmente. Quando o algoritmo chega na linha do *breakpoint* a execução é pausada e a linha fica destacada em azul (como mostra a Figura 21). Para dar sequência à execução, precisamos pressionar a tecla F8 (ou o botão correspondente, apresentado anteriormente na Figura 10). A execução passo-a-passo (com ou sem breakpoints) é conhecida como *debug*. *Debugar* um programa (ou algoritmo) significa executá-lo tentando encontrar suas falhas e problemas.

FIGURA 21– Execução com *Breakpoint*



FONTE: dos autores

3.3
ESTRUTURA BÁSICA DE UM ALGORITMO NO VisuAlg

A estrutura básica de um algoritmo no VisuAlg envolve duas seções principais: *seção de declarações* e *seção de comandos*. Quando iniciamos um novo algoritmo no VisuAlg, a estrutura básica já é apresentada, incluindo o cabeçalho (palavra reservada *algoritmo*), a seção de declarações (*var*) e a seção de comandos (delimitada por inicio e *fimalgoritmo*). Além disso, a data também é apresentada, com base na data configurada no Sistema Operacional. As linhas que apresentam duas barras no início (//) são comentários e podem ser omitidas, a seu critério. A Figura 22 apresenta a estrutura básica de um algoritmo no VisuAlg.

FIGURA 22 – Estrutura Básica de um Algoritmo no VisuAlg

```
algoritmo "semnome"
// Função :
// Autor :
// Data : 07/04/2017
// Seção de Declarações
var

inicio
// Seção de Comandos
fimalgoritmo
```

FONTE: dos autores, adaptado por NTE, 2017

3.3.1 Declaração de Variáveis

Variável é uma identificação dada a uma posição de memória, utilizada para armazenar dados. Todas as variáveis definidas pelo usuário (no caso, você, que é o programador neste caso) devem ser declaradas no bloco denominado Var.

O VisuAlg possui os seguintes tipos de dados que podem ser associados às variáveis: *inteiro, real, caracter* e *logico*.

A sintaxe para a declaração de variáveis é a seguinte:

Var
<lista-de-variáveis> : <tipo-de-dado>

A lista de variáveis (quando houver mais de uma variável do mesmo tipo) deve ser separada por vírgulas. Por exemplo, para declarar uma variável que deve armazenar o nome de uma pessoa, devemos usar o tipo *caracter*. Vamos usar, como nome da variável, *nome_pessoa*. Para armazenar a idade de uma pessoa devemos usar o tipo inteiro. Para armazenarmos, por exemplo, a média e as notas de um aluno, devemos usar o tipo *real*. Abaixo, vemos então, a declaração de variáveis para estes exemplos:

```
Var
  nome_pessoa: caracter
  idade: inteiro
 nota1, nota2, media: real
```

3.3.2 Comandos de Entrada e Saída

Os comandos de entrada e saída possibilitam a comunicação entre o usuário e o computador.

Comando Escreva

O comando Escreva apresenta uma informação por meio de um periférico de saída, no caso o vídeo. Este comando pode escrever mensagens e/ou conteúdos armazenados nas variáveis manipuladas pelo algoritmo. A sintaxe do comando *escreva* é:

escreva(<lista-de-expressões>)

A <*lista-de-expressões*> pode ser um mensagem entre aspas, uma variável, o resultado de uma expressão aritmética, entre outras possibilidades. As informações entre aspas são denominadas **mensagens**. Estas informações são simplesmente copiadas para o periférico de saída. As informações que estão fora das aspas representam variáveis e/ou expressões aritméticas.

No algoritmo apresentado como exemplo na unidade anterior, o comando:

escreva("Digite a nota de P1:")

apresenta no vídeo a mensagem: *Digite a nota de P1:*

A utilização do símbolo de 2 pontos (:) no final da mensagem é para dar a impressão de que o usuário (quem irá utilizar o programa que estamos desenvolvendo) deve entrar com alguma informação. Funciona como um *convite* para a entrada de dados.
Já o comando:

escreva("Média do Aluno:", media)

apresenta no vídeo a mensagem: *Média do Aluno*, seguida do conteúdo que estiver armazenado na memória, no espaço denominado *media*.

O comando *escreva* possui uma variação no VisuAlg, que é o comando *escreval*. Utilizando escreval, após a mensagem ser mostrada na tela, a próxima saída ou entrada será feita na linha seguinte. Este comando pode ser utilizado, então, para melhorar a saída de dados para o usuário.

Comando Leia

O comando *leia* permite que uma informação digitada pelo teclado seja capturada e armazenada na variável definida. A sintaxe do comando *leia* é:

leia(<lista-de-variáveis>)

No algoritmo da média, mostrado como exemplo na unidade anterior, o comando:

leia(P1)

permite que o usuário digite a nota P1 do aluno e armazena esta nota na posição de memória reservada para a variável P1.

3.3.3 Comando de Atribuição

O comando de atribuição, representado pelo sinal de menor seguido do sinal de menos (<-) determina que uma variável receba o valor de uma expressão. Receber (ou atribuir) significa armazenar o valor recebido na memória, na posição ("gaveta") correspondente a este valor. A sintaxe do comando de atribuição é:

<variável> <- *<expressão>*

Onde:

<variável> indica o nome da variável que vai receber o valor que lhe é atribuído pela *<expressão>*.

<- Símbolo de atribuição

<expressão> indica o valor que será atribuído à variável. Pode ser um valor absoluto ou o resultado de um cálculo (expressão aritmética).

No algoritmo da média, mostrado como exemplo na unidade anterior, o comando:
media <- (P1 + P2)/2

significa que a variável denominada *media* (a posição de memória reservada para esta variável) receberá o resultado da expressão aritmética que calcula o valor da média do aluno, a partir das notas armazenadas nas variáveis *P1* e *P2*.
Devemos lembrar que, quando atribuímos novos valores a uma variável, o valor anterior é perdido. Exemplo:

ATENÇÃO: Quando é realizada uma atribuição de valores a uma variável, o seu conteúdo anterior é perdido.

```
media <- 5.6    //A variável média recebeu o valor 5.6
media <- 0      //A variável média perde o conteúdo anterior e armazena o valor Zero
```

3.4
ALGORITMOS SEQUENCIAIS

A partir dos comandos estudados até o momento (*leia, escreva* e *atribuição*), é possível realizar a construção de **Algoritmos Sequenciais**. Um algoritmo sequencial é um conjunto de instruções executadas em sequência linear, de cima para baixo, da esquerda para a direita.

O exemplo da Figura 23 mostra um algoritmo sequencial: Escrever um algoritmo que leia o valor recebido por hora e o número de horas trabalhadas por um funcionário e calcule o salário bruto.

FIGURA 23 – Exemplo de Algoritmo para Calcular o Salário Bruto

```
algoritmo "Calculo_Salario_Bruto"
// Função : Calcular o valor do salário bruto de um funcionário
// Autor : Autores do Livro de Introdução a Algoritmos
// Data : 07/04/2017
// Seção de Declarações
var
   Valor_Hora, Numero_Horas, Salario_Bruto: real
inicio
// Seção de Comandos
 escreva(“Digite o valor recebido por hora em R$:”)
 leia(Valor_Hora)
 escreva(“Digite o número de horas trabalhadas no mês:”)
 leia(Numero_Horas)
Salario_Bruto <- Valor_Hora * Numero_Horas
Escreva(“Salário Bruto do Funcionário:”, Salario_Bruto)
fimalgoritmo
```

FONTE: dos autores, adaptado por NTE, 2017

Vejamos um segundo exemplo de algoritmo sequencial: Uma locadora aluga seus carros com uma taxa fixa por dia e uma taxa por quilômetro (km) rodado. Escrever um algoritmo que leia a taxa fixa por dia, a taxa por km rodado, o número de dias, o número de quilômetros rodados e calcule o valor total do aluguel considerando que a locadora está com uma promoção de 10% de desconto na taxa fixa por dia.

Dados de Entrada:
– taxa fixa por dia
– taxa por km rodado
– número de dias
– número de quilômetros rodados

Dados de Saída (Resultados):
– Valor total do aluguel do carro

Cada um destes dados precisa ser armazenado em uma variável de memória. Neste exemplo, todos os dados manipulados são numéricos. Os valores das taxas podem ser valores reais (permitem armazenar dados com vírgula) e os valores correspondentes aos números de dias e de quilômetros rodados são inteiros. No cálculo do valor do aluguel foi utilizada a variável Desconto, apenas para calcular o desconto de 10% separadamente, a título de exemplo. A Figura 24 apresenta uma solução possível, em pseudocódigo, para este problema.

FIGURA 24 – Algoritmo para Calcular o Valor do Aluguel do Carro

```
algoritmo "Aluguel_Carro"
// Função : Calcular a taxa de aluguel de um carro
// Autor : Autores do Livro de Introdução a Algoritmos
// Data : 07/04/2017
// Seção de Declarações
var
   Taxa_Diaria, Taxa_Km, Valor_Aluguel, Desconto: real
  Num_Dias, Num_Km: inteiro
inicio
// Seção de Comandos
 escreva("Digite o valor da taxa fixa por dia em R$:")
 leia(Taxa_Diaria)
 escreva("Digite o valor da taxa por quilômetro em R$:")
 leia(Taxa_Km)
escreva("Digite o número de dias da locação:")
leia(Num_Dias)
escreva("Digite o número de quilômetros rodados:")
leia(Num_Km)
        Valor_Aluguel <- (Num_Dias*Taxa_Diaria) + (Num_Km * Taxa_Km)
        Desconto <- (Num_Dias*Taxa_Diaria) * 0.1
        Valor_Aluguel <- Valor_Aluguel - Desconto
escreva("Valor da locação do veículo:",Valor_Aluguel)
fimalgoritmo
```

FONTE: dos autores, adaptado por NTE, 2017

3.5
ALGUMAS FUNÇÕES/COMANDOS DA LINGUAGEM DO VisuAlg

Esta subunidade apresenta alguns comandos que podem ser aplicados nos algoritmos construídos com o uso do VisuAlg.

3.5.1 Comando *Aleatorio*

O comando *aleatorio* permite que os dados de entrada sejam gerados de forma aleatória, facilitando os testes dos algoritmos no VisuAlg. Para utilizar este comando devemos ativar a geração de dados aleatórios, por meio do comando *aleatorio on* (sem acento, já que é uma regra de sintaxe). Opcionalmente podemos definir a faixa de valores permitidos. Por exemplo, no caso do algoritmo para calcular a média do aluno, o intervalo de notas permitido está entre 0 e 10. O algoritmo da Figura 25 apresenta um exemplo de utilização do comando aleatorio. Para facilitar o entendimento, as linhas do algoritmo foram numeradas. Para utilizar o comando *aleatorio* é preciso ativá-lo por meio da instrução *aleatorio on* antes do comando leia (como se vê na linha 11). A definição do intervalo de valores permitidos (linha 12) é opcional. Após as leituras, deve-se desativar a geração de números aleatórios (comando *aleatorio off* na linha 17).

FIGURA 25 – Algoritmo com o Comando *Aleatorio*

```
1. algoritmo "calculamedia"
2. // Função : Calcular a média de um aluno
3. // Autor :  Autores do Livro de Introdução a Algoritmos
4. // Data : 06/04/2017
5. // Seção de Declarações
6. var
7.  P1, P2, media: real
8. inicio
9. // Seção de Comandos
10. // Entrada de Dados
11.  aleatorio on
12.  aleatorio 0,10
13.  escreva("Digite a nota de P1:")
14.  leia(P1)
15.  escreva("Digite a nota de P2:")
16.  leia(P2)
17.  aleatorio off
18. //Processamento
19.  media <- (P1+P2)/2
20. //Saída de Dados
21.  escreva("Média do aluno:",media)
22. fimalgoritmo
```

FONTE: dos autores, adaptado por NTE, 2017

3.5.2 Comando *Limpatela*

O comando *limpatela* limpa a tela de saída, no modo de execução DOS, não afetando a janela que existe na parte inferior direita da janela do VisuAlg. No exemplo do algoritmo da Figura 26 incluímos o comando *limpatela* na linha 16, após terminarmos a entrada de dados. Isto significa que a tela de saída será limpa antes de serem apresentados os resultados (média do aluno).

FIGURA 26 – Algoritmo com o Comando *Limpatela*

```
1. algoritmo "calculamedia"
2. // Função : Calcular a média de um aluno
3. // Autor :  Autores do Livro de Introdução a Algoritmos
4. // Data : 06/04/2017
5. // Seção de Declarações
6. var
7.  P1, P2, media: real
8. inicio
9. // Seção de Comandos
10. // Entrada de Dados
11. escreva("Digite a nota de P1:")
12.  leia(P1)
13.  escreva("Digite a nota de P2:")
14.  leia(P2)
15.  //Processamento
16. limpatela
17.  media <- (P1+P2)/2
18. //Saída de Dados
19.  escreva("Média do aluno:",media)
20. fimalgoritmo
```

FONTE: dos autores, adaptado por NTE, 2017

3.5.3 Comando *Arquivo*

O comando *arquivo* permite que os dados de entrada sejam armazenados em um arquivo do tipo texto (TXT), permitindo que os testes sejam facilitados, especialmente quando você for construir algoritmos com repetição, que serão estudados em unidades posteriores deste livro. O algoritmo apresentado na Figura 27 ilustra a utilização do comando *arquivo*.

FIGURA 27 – Algoritmo com o Comando Arquivo

```
algoritmo "lendo do arquivo"
arquivo "teste.txt"
var x,y: inteiro
inicio
para x de 1 ate 5 faca
  leia (y)
fimpara
fimalgoritmo
```

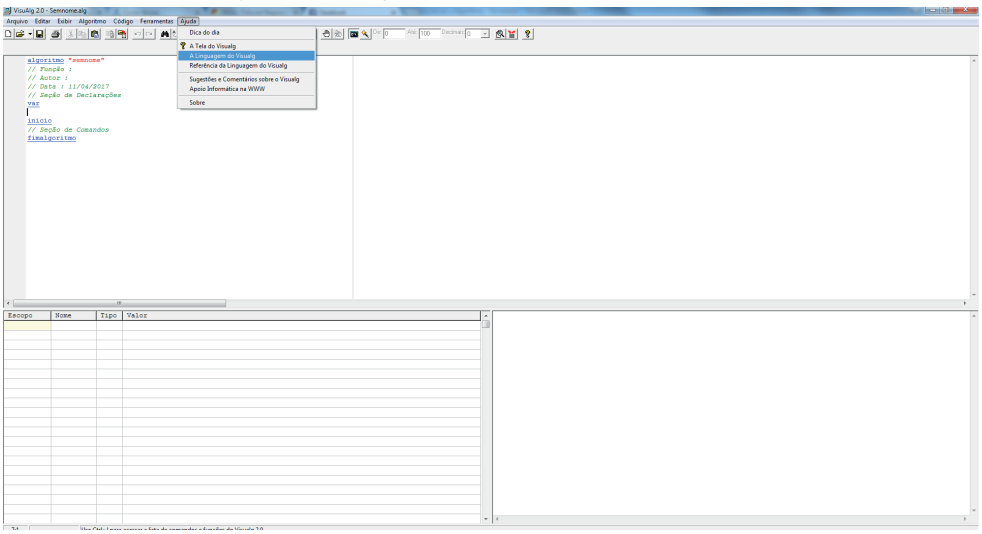
FONTE: Apoio Informática.

A sintaxe do comando é arquivo <*nome-de-arquivo*>, onde <*nome-de-arquivo*> é uma constante caractere (entre aspas duplas). Este comando funciona da seguinte forma (APOIO INFORMÁTICA, 2017):

1) Se **não existir** o arquivo com nome especificado, o VisuAlg fará uma leitura de dados por meio da digitação, armazenando os dados lidos neste arquivo, na ordem em que forem fornecidos;
2) Se o arquivo **existir**, o VisuAlg obterá os dados deste arquivo até chegar ao seu fim. Daí em diante, fará as leituras de dados por meio da digitação;
3) Somente um comando *arquivo* pode ser empregado em cada algoritmo, e ele deverá estar na seção de declarações;
4) Caso não seja fornecido um caminho, o VisuAlg irá procurar este arquivo na pasta de trabalho corrente (geralmente, é a pasta onde o programa VISUALG.EXE está armazenado). Este comando não prevê uma extensão padrão; portanto, a especificação do nome do arquivo deve ser completa, inclusive com sua extensão (por exemplo, .txt, .dat, etc.).

Além destes comandos, você pode consultar outros utilizando a Ajuda do VisuAlg, como mostra a Figura 28. Existem as opções "*A Linguagem do Visualg*" que apresenta as regras de sintaxe das instruções da linguagem e "*Referência da Linguagem do Visualg*", que apresenta uma lista de todos os comandos existentes.

FIGURA 28 – Opções de Ajuda do VisuAlg



FONTE: dos autores

4

ALGORITMOS COM DECISÃO

INTRODUÇÃO

Agora que você já aprendeu a trabalhar com o VisuAlg, nesta unidade continuaremos desenvolvendo soluções com esta ferramenta, principalmente as que façam uso dos comandos que permitem o controle de fluxo de decisão.

Nas unidades anteriores, foram apresentadas soluções que lidam com a estrutura sequencial, que corresponde ao fato das ações serem executadas em uma sequência linear, de cima para baixo e da esquerda para a direita, ou seja, na mesma ordem em que são escritas.

Nessa unidade apresentamos formas de criar algoritmos fazendo uso do controle de decisão, ou **estrutura condicional**, que constitui um recurso para determinar a ordem em que os comandos de um algoritmo devem ser executados e se serão executados ou não. Sendo assim, a estrutura de seleção permite a escolha do fluxo em um grupo de ações a ser executado quando determinadas condições, que são representadas **por expressões lógicas** ou **relacionais** são ou não satisfeitas.

4.1
SELEÇÃO SIMPLES

Um comando de seleção simples possui instruções que devem ser executadas apenas quando a condição foi satisfeita, isto é, quando a condição for verdadeira.

Sintaxe:

se <expressão condicional> entao
<bloco de instruções>
fimse

A expressão *condicional*, que pode ser *lógica* ou *relacional*, ainda pode ser subdividia em simples ou composta. Se o resultado da expressão for verdadeiro, são executados os comandos contidos no bloco de instruções, ou seja, antes do *fimse*. Caso contrário, os comandos, contidos no bloco de instrução, não serão executados e a execução passa para a linha seguinte ao *fimse*.

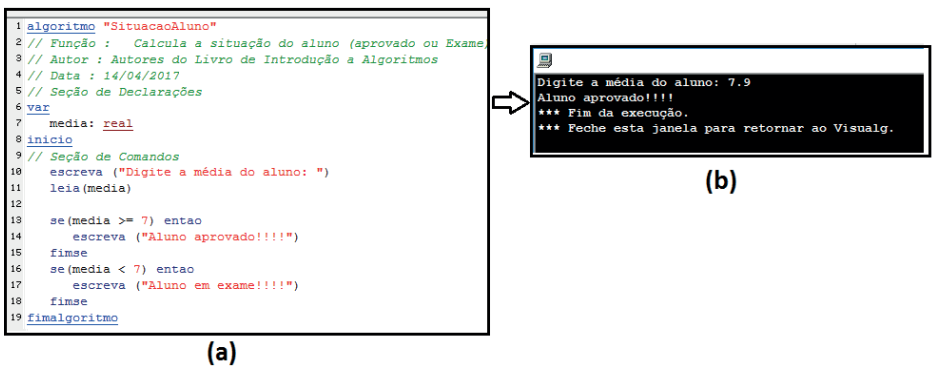
Exemplo: Escrever um trecho de algoritmo que verifique se um aluno foi aprovado por média (média >= 7.0):

```
se (media>=7) entao
   escreva("Aluno aprovado!")
fimse

se (media<7) entao
  escreva("Aluno em Exame!")
fimse
```

O algoritmo completo, no VisuAlg, para classificar a situação do aluno como sendo aprovado ou reprovado está representado, graficamente, na Figura 29.

FIGURA 29 – Seleção Simples



FONTE: dos autores

A Figura 29, letra a, apresenta o algoritmo. Inicialmente foi declarada, na **linha 7**, uma variável do tipo **real**, chamada **media**. Na sequência é apresentada uma informação (mensagem), que solicita ao usuário a digitação da média do aluno, na **linha 10**. Já na **linha 11** é realizada a leitura da informação digitada.

A primeira expressão condicional simples inicia na **linha 13** e verifica se a **media** é maior ou igual a 7 (>= 7). Caso seja verdadeira a expressão, então é executado o comando da **linha 14**, que escreve na tela, **Aluno aprovado!!!**. Após verificar a primeira condição, o algoritmo segue verificando a segunda expressão condicional simples, que inicia na linha 16 e verifica se a media é menor do que 7 (<7). Caso seja verdadeira essa expressão, então é executado o comando da linha 17, que escreve na tela, **Aluno em exame!!!**. Um dos possíveis resultados da execução do algoritmo é apresentado na Figura 29 letra b.

4.2
SELEÇÃO COMPOSTA

Um comando de seleção composta possui instruções que devem ser executadas quando a condição for verdadeira (*então*), bem como quando a condição for falsa (*senão*), ou seja, duas alternativas que dependem de uma mesma condição.

Sintaxe:

```
se <expressão condicional > entao
        <bloco de instruções 1>
              senao
        <bloco de instruções 2>
              fimse
```

É importante frisar que a expressão condicional, como no caso da seleção simples, também pode ser simples ou composta. Se o resultado da expressão for verdadeiro, são executados os comandos determinados pelo bloco de instruções 1 (antes do *senao*). Caso contrário, quando o resultado for falso, serão executados os comandos do bloco de instruções 2 (antes do *fimse*). Após a execução (tanto do resultado verdadeiro como do falso), serão executados os comandos após o *fimse*.

Vamos retornar ao nosso exemplo anterior, que foi utilizado para demonstrar a seleção simples. Este exemplo pode ser reescrito por meio de uma seleção composta:

```
se (media>=7) entao
        escreva("Aluno aprovado!")
    senao
      escreva("Aluno em Exame!")
    fimse
```

É importante observarmos que, neste exemplo, elimina-se um comando *se*. O algoritmo completo, no VisuAlg, usando seleção composta está representado, graficamente, na Figura 30.

FIGURA 30 – Seleção Composta

```
1  algoritmo "SituacaoAluno"
2  // Função : Calcula a situação do aluno (aprovado ou Exame)
3  // Autor : Autores do Livro de Introdução a Algoritmos
4  // Data : 14/04/2017
5  // Seção de Declarações
6  var
7    media: real
8  inicio
9  // Seção de Comandos
10   escreva ("Digite a média do aluno: ")
11   leia(media)
12
13   se(media >= 7) entao
14      escreva ("Aluno aprovado!!!!")
15   senao
16      escreva ("Aluno em exame!!!!")
17   fimse
18
19 fimalgoritmo
```

FONTE: dos autores

Na seleção composta, conforme apresentado na Figura 30, primeiramente verificamos se a **media** é maior ou igual a **7, na linha 13**. Caso seja verdadeira a expressão, então é executado o comando da **linha 14**, que escreve "**Aluno aprovado**!!!!". Se a expressão da **linha 13** for falsa, logo é executado o comando da **linha 16**, que escreve "**Aluno em exame**!!!!".

4.3
SELEÇÃO ENCADEADA

Passaremos a estudar a seleção encadeada. Nela é permitido o agrupamento de diversas seleções. Por exemplificar, vamos levar em consideração o cálculo simplificado do valor do Imposto de Renda a ser pago por um funcionário, a partir dos valores do Quadro 16.

QUADRO 16 – Valores para Cálculo do Imposto de Renda

SALÁRIO	ALÍQUOTA DO IMPOSTO DE RENDA (IR)
Até R$1.200,00	Isento de IR
Acima de R$1.200,00 Até R$2.500,00	15%
Acima de R$ 2.500,00	25%

FONTE: dos autores, adaptado por NTE, 2017

O trecho do algoritmo para verificar em que alíquota se enquadra um funcionário pode ser escrito por meio de seleções simples ou de uma seleção encadeada.

Exemplo 1: Com seleções simples

```
se (salario<=1200) entao
        escreva("Funcionário Isento do IR!")
    fimse
    se (salario>1200) e (salario<=2500) entao
        escreva("Imposto de Renda a Pagar:", Salario*0.15)
    fimse
    se (salario>2500) entao
        escreva("Imposto de Renda a Pagar:", Salario*0.25)
fimse
```

O problema na execução do algoritmo com seleção simples é que, por exemplo, quando um funcionário estiver isento, as condições das alíquotas de 15% e 25% serão testadas assim mesmo, pois estão na ordem de execução do algoritmo, ou seja, será desperdiçado o tempo de processamento.

Para otimizar o processamento vamos resolver o mesmo problema usando seleção encadeada, conforme apresentado na proxima página.

Exemplo 2: Com seleção encadeada

```
        se (salario<=1200) entao
            escreva("Funcionário Isento do IR!")
        senao
se (salario>1200) e (salario<=2500) entao
                    escreva("Imposto de Renda a Pagar:", Salario*0.15)
senao
    se (salario>2500) entao
            escreva("Imposto de Renda a Pagar:", Salario*0.25)
        fimse
    fimse
fimse
```

ATENÇÃO: Você deve ter notado que, ao escrevermos os comandos, algumas vezes, costumamos deixar um espaço (uma tabulação) no início da linha. Estes espaços são conhecidos como *indentação* de código e servem para deixar o código legível, além de apresentarem a hierarquia dos comandos. Por exemplo, a linha de comando escreva("*Funcionário Isento de IR*!") está dentro do comando *se*, por isso foi recuada.

Ao analisar o algoritmo no exemplo 2, podemos observar que, se um funcionário tem um salário líquido de R$800,00, por exemplo, o algoritmo mostra a mensagem "**Funcionário Isento do IR**" e a execução do algoritmo é desviada para o próximo comando após o último *fimse*. Isto ocorre porque as demais condições foram colocadas no *senao*, ou seja, serão executadas somente quando o resultado da primeira expressão for falsa. A implementação deste exemplo, no VisuAlg, é apresentada na Figura 31.

FIGURA 31 – Seleção Encadeada

```
1  algoritmo "CalculoIR"
2  // Função : Calcula o IR de un funcionário
3  // Autor : Autores do Livro de Introdução a Algoritmos
4  // Data : 15/04/2017
5  // Seção de Declarações
6  var
7    salario: real
8
9  inicio
10 // Seção de Comandos
11    escreva("Digite o salário do funcionário: ")
12    leia(salario)
13
14    Se (salario<=1200) entao
15        Escreva("Funcionário Isento do IR!")
16    Senao
17       Se (salario>1200) e (salario<=2500) entao
18           Escreva("Imposto de Renda a Pagar: ", salario*0.15)
19       senao
20          Se (Salario>2500) entao
21              Escreva("Imposto de Renda a Pagar:", salario*0.25)
22          fimse
23       fimse
24    fimse
25
26 fimalgoritmo
```

FONTE:dos autores

Vamos fazer uma análise detalhada no algoritmo da Figura 31 e pensarmos como ele pode ser melhorado.

SAIBA MAIS: Pense! O código na linha 21 (Salario > 2500) pode ser retirado do algoritmo sem que o mesmo tenha seu resultado modificado. Por quê?

4.4
SELEÇÃO DE MÚLTIPLA ESCOLHA

Por meio de uma estrutura de seleção do tipo múltipla escolha, é possível escrever algoritmos que têm sua execução similar a uma seleção encadeada; porém, com maior facilidade de entendimento do código. A sintaxe da estrutura de seleção de múltipla escolha do VisuAlg é descrita a seguir.

Sintaxe:

escolha <variável de verificação>
caso <Valor1>
"Instruções a serem executadas caso <Variável De Verificação> = <Valor1>"
caso <Valor2>
"Instruções a serem executadas caso <Variável De Verificação> = <Valor2>"
caso <Valor3>
"Instruções a serem executadas caso <Variável De Verificação> = <Valor3>"
Outrocaso
"Instruções a serem executadas caso <Variável De Verificação> diferente dos Valores anteriores".
fimescolha

Para demonstrar a aplicação de seleção de múltiplas escolhas, vamos considerar o exemplo de uma calculadora, contendo quatro operações, sendo elas soma, subtração, adição e multiplicação. Primeiramente, vamos desenvolver a solução usando seleção encadeada (comandos *se* e *senao*) diretamente no VisuAlg, conforme apresentado, graficamente, na Figura 32.

FIGURA 32 – Calculadora usando Seleção Encadeada

```
algoritmo "Calculadora"
// Função : Calculadora que faz as 4 operações básicas
// Autor : Autores do Livro de Introdução a Algoritmos
// Data : 15/04/2017
// Seção de Declarações
var
   numero1, numero2, resultado: real
   operacao: caractere
inicio
// Seção de Comandos
   escreva("Digite o número 1: ")
   leia(numero1)
   escreva("Digite o número 2: ")
   leia(numero2)
   escreva("Digite a operação: ")
   leia(operacao)

   se(operacao = "+") entao
         resultado <- numero1 + numero2
   senao
        se(operacao = "-") entao
             resultado <- numero1 - numero2
        senao
            se(operacao = "*") entao
               resultado <- numero1 * numero2
            senao
               se(operacao = "/") entao
                  resultado <- numero1 / numero2
               fimse
            fimse
        fimse
   fimse

   escreva("o resultado de ",numero1," ",operacao," ",numero2," =",resultado)
fimalgoritmo
```

FONTE: dos autores

Analisando o algoritmo da Figura 32, começaremos com a declaração de variáveis, **linha 7**, onde foram declaradas todas as variáveis do tipo *real*, cujas finalidades são armazenar os números digitados pelo usuário e o resultado calculado. Na **linha 8** foi declarada uma variável para armazenar a operação, a ser digitada, do tipo caractere.

Nas **linhas 11**, **13** e **15** são apresentadas as instruções de como o usuário deve proceder, utilizando os comandos ***escreva***. Na sequência, são realizadas as leituras da informações digitadas pelo usuário, por meio dos comandos ***leia***, nas **linhas 12**, **14** e **16**.

Agora vamos analisar a seleção encadeada, a começar, na **linha 18**, que faz o teste condicional para verificar se operação digitada é igual à soma. Se for verdadeira, o algoritmo calcula a soma dos números e atribui à variável resultado, na **linha 19**. Caso o teste condicional seja falso, a instrução *se*, na **linha 21** é executada fazendo o teste condicional pra verificar a operação digitada. Se for igual à subtração é realizada a operação da linha 22. Se o teste da linha 21 falhar, então é executada a linha 24, que faz a verificação da operação de multiplicação. Caso esta verificação seja verdadeira executa-se a linha 25 e caso seja falsa a **linha 27**. Se o teste condicional da linha 27 for verdadeiro são executados os comandos da **linha 28**.

Vejam que, devido ao uso da seleção encadeada, pode haver um pouco de dificuldade para compreender o algoritmo. Ao fazermos uso da estrutura de seleção do tipo múltipla escolha o algoritmo torna-se mais compreensivo. Veja a representação, gráfica, da solução do mesmo problema, na Figura 33.

FIGURA 33 – Calculadora usando a Estrutura de Seleção do Tipo Múltipla Escolha

```
1  algoritmo "Calculadora"
2  // Função : Calculadora contendo as 4 operações básicas
3  // Autor : Autores do Livro de Introdução a Algoritmos
4  // Data : 15/04/2017
5  // Seção de Declarações
6  var
7     numero1, numero2, resultado: real
8     operacao: caractere
9  inicio
10 // Seção de Comandos
11    escreva("Digite o número 1: ")
12    leia(numero1)
13    escreva("Digite o número 2: ")
14    leia(numero2)
15    escreva("Digite a operação: ")
16    leia(operacao)
17
18    escolha operacao
19       caso "+"
20          resultado <- numero1 + numero2
21       caso "-"
22          resultado <- numero1 - numero2
23       caso "*"
24          resultado <- numero1 * numero2
25       caso "/"
26          resultado <- numero1 / numero2
27       outrocaso
28           escreva("Operação inexistente!")
29    fimescolha
30
31    escreva("o resultado de ",numero1," ",operacao," ",numero2," =",resultado)
32 fimalgoritmo
```

FONTE: dos autores

Ao analisar o algoritmo apresentado na Figura 34, chegaremos à conclusão de que poucas instruções mudaram, acontecendo apenas a inserção dos comandos nas **linhas 18** a **29**. A primeira verificação ocorre na **linha 19**. Nesta linha inicia-se a verificação da operação digitada pelo usuário. Se ela for a soma, então executam-se os comandos na **linha 20**. Caso a operação digitada não seja a soma, o algoritmo passa a verificar a próxima operação, na **linha 21**. Na linha 21, caso a operação digitada pelo usuário seja a subtração, executam-se os comandos na **linha 22**. Caso a operação digitada não seja a subtração, o algoritmo passa a verificar a próxima operação, na **linha 23**. Na **linha 23**, caso a operação digitada pelo usuário seja a multiplicação, então executam-se os comandos na **linha 24**. Se a operação digitada não for a multiplicação, o algoritmo passa a verificar a próxima operação, na **linha 25**, que é a **divisão**. Se a operação digitada for a divisão executam-se os comandos na **linha 26**.

Caso a operação digitada pelo usuário não seja a soma, subtração, multiplicação ou divisão, o algoritmo executa a **linha 28**.

5

ALGORITMOS COM REPETIÇÃO

INTRODUÇÃO

Vamos imaginar que temos de calcular, para todos os alunos de uma escola ou de uma turma, a média das notas e classificar a situação de cada um como "Aprovado" ou "Em exame". Para isso, temos que aplicar a solução já estudada nas unidades anteriores, que faz uso do valor da média para classificar a situação do aluno.

O fato é que a escola pode possuir mais de um aluno, por exemplo 30, como poderemos calcular o valor da média de cada aluno? A primeira solução que poderíamos pensar é replicar o código 30 vezes, uma para cada aluno e ao final da execução do algoritmo teríamos a classificação para cada aluno. Está solução torna-se inviável quando a escola matricula novos alunos, pois teremos que replicar novamente o código, além disso, existe uma grande quantidade de códigos que fazem a mesma tarefa, ou seja, precisamos otimizar o nosso algoritmo. Outra solução seria executar o mesmo algoritmo 30 vezes.

Neste sentido, vamos pensar em aprimorar a solução. Como podemos iniciar o fluxo da execução do algoritmo exatamente no início dos cálculos e fazer com que esse fluxo seja repetido enquanto for necessário?

A resposta dessa pergunta, que busca melhorar o algoritmo, denomina-se estrutura de repetição, pois essa estrutura permite que um mesmo trecho do algoritmo seja executado mais de uma vez. Além disso, a repetição pode ser determinada (**repetição contada**) ou indeterminada (**baseada em uma condição**). São estas estruturas de repetição que iremos estudar nesta unidade.

5.1
REPETIÇÃO CONTADA

Neste tipo de repetição, a contada, a execução de um bloco de comandos é executada um número predefinido de vezes, com limites definidos. Voltando ao exemplo utilizado nas unidades anteriores, o cálculo da média de um aluno, vamos supor que desejamos calcular a média de uma turma de 10 alunos. Existem, basicamente, três possibilidades para que isto seja realizado:

1) executar o programa que calcula a média de um aluno, 10 vezes seguidas;
2) repetir o mesmo trecho de código (leitura, cálculo, impressão) 10 vezes;
3) utilizar uma estrutura de repetição.

Ao analisarmos as alternativas 1 e 2, apesar de estarem corretas, não são as mais adequadas. Veja o exemplo de como ficaria um trecho do algoritmo de cálculo da média utilizando a alternativa 2 (Figura 34).

FIGURA 34 – Exemplo de Algoritmo para Calcular a Média

```
algoritmo “Calcular_Media”
var
                    P1,P2, Media: real
inicio
                    // Trecho de código para o aluno 1
                    escreva("Digite a nota de P1:")
                    leia(P1)
                    escreva( "Digite a nota de P2:")
                    leia(P2)
                    Media<-(P1 + P2 )/2
                    escreva( "Média do Aluno:", Media)
                    se (Media>=7) entao
                      escreva( "O aluno está aprovado!")
                    senao
                      escreva( "O aluno está em Exame!")
                    fimse
          // Trecho de código para o aluno 2
                    escreva("Digite a nota de P1:")
                    leia(P1)
                    escreva( "Digite a nota de P2:")
                    leia(P2)
                    Media<-(P1 + P2 )/2
                    escreva( "Média do Aluno:", Media)
                    se (Media>=7) entao
                      escreva( "O aluno está aprovado!")
                    senao
                      escreva( "O aluno está em Exame!")
                    fimse

                    // e assim sucessivamente, para os demais alunos
fimalgoritmo
```

FONTE: dos autores, adaptado por NTE, 2017

FIGURA 35 – Algoritmo para Calcular a Média utilizando a Repetição Contada

```
algoritmo “Calcular_Media”
var
                    P1, P2, Media: real
                    Contador: inteiro
inicio
                    para Contador de 1 ate 10 faca
                escreva( "Digite a nota de P1:")
                        leia(P1)
                        escreva( "Digite a nota de P2:")
                        leia(P2)
                        Media<-(P1 + P2 )/2
                        escreva( "Média do Aluno:", Media)
                        se (Media>=7) entao
                          escreva( "O aluno está aprovado!")
                        senao
                            escreva( "O aluno está em Exame!")
                        fimse
                    fimpara
            fimalgoritmo
```

FONTE: dos autores, adaptado por NTE, 2017

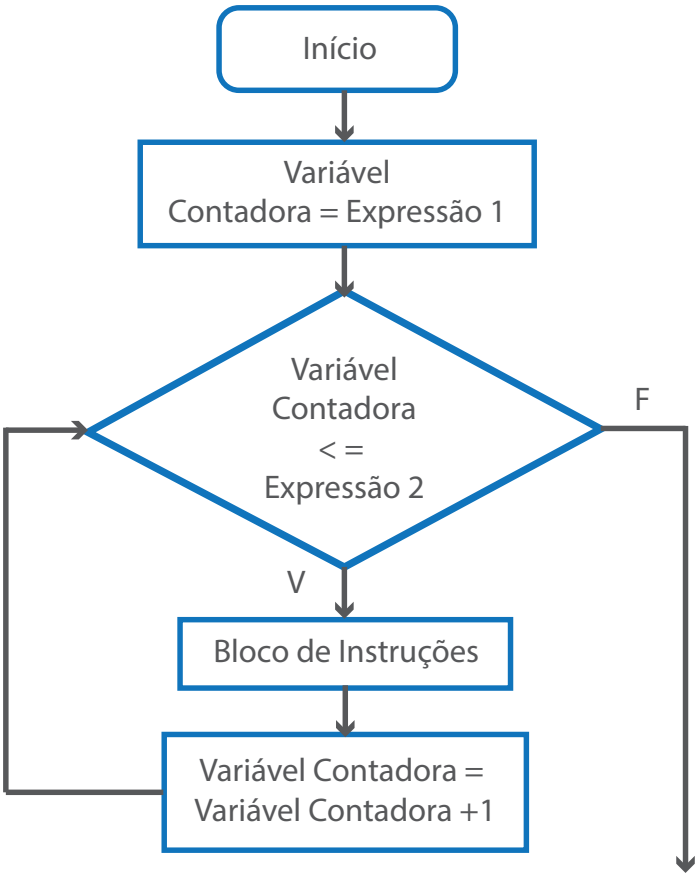
Observando o algoritmo apresentado na Figura 35, notamos que foi incluída outra variável numérica, a variável Contador, que serve para limitar a quantidade de vezes que o código irá ser repetido.

A seguir vamos descrever a sintaxe do comando *para... ate* , no VisuAlg:

para <variável contadora > de <valor-inicial> ate <valor final> [passo <valor de incremento>] faca
<sequência-de-comandos a serem executados repetidamente até a <variável contadora> atingir o valor final>
fimpara

Continuando a análise do comando *Para...ate,* o fluxograma de execução deste comando é apresentado na Figura 36.

FIGURA 36 – Fluxograma de execução do comando *para..ate*



FONTE: dos autores, adaptado por NTE, 2017

Analisando a Figura 36, inicialmente, a variável contadora recebe o valor representado pelo resultado da expressão 1. Enquanto o valor da variável contadora for menor ou igual ao valor estipulado pela expressão 2, o bloco de instruções será repetido. Por *default* (padrão), cada vez que o bloco de instruções é repetido, a variável contadora é incrementada em 1. Este valor pode ser modificado, utilizando-se a opção passo. O *passo* pode ser incremental ou decremental. Para exemplificar, vemos abaixo uma repetição (laço de repetição) executada de forma decrescente:

```
para Contador de 10 ate 1 passo -1 faca
    escreva( "Valor do Contador:", Contador )
fimpara
```

A Figura 37 apresenta o algoritmo para calcular a média de uma turma de 10 alunos utilizando o comando *para...ate* no VisuAlg.

FIGURA 37 – Estrutura de repetição *para...ate*

```
1  algoritmo "CalcularMedia"
2  // Função : Calcular a média de 10 alunos e classifica-los
3  // Autor : Autores do Livro de Introdução a Algoritmos
4  // Data : 16/04/2017
5  // Seção de Declarações
6  var
7     p1, p2, media: real
8     contador:inteiro
9  inicio
10 // Seção de Comandos
11    para contador de 1 ate 10 faca
12       escreva("Digite a nota da prova p1: ")
13       leia(p1)
14       escreva("Digite a nota da prova p2: ")
15       leia(p2)
16
17       media <-(p1 + p2 )/2
18       escreva("Média do Aluno",contador," é: ", media)
19
20       se(media>=7)entao
21         escreval(", ele está aprovado!")
22       senao
23          escreval(", ele está em Exame!")
24       fimse
25
26    fimpara
27 fimalgoritmo
```

FONTE: dos autores

A partir do exemplo de algoritmo, representado graficamente na Figura 37, podemos propor os seguintes questionamentos, para incrementar o funcionamento do mesmo:

– como calcular a média geral da turma?
– como contar o número de alunos aprovados e em exame?
– como mostrar a porcentagem de aprovação?
– como permitir que o usuário escolha o número de alunos da turma?

Alguns destes questionamentos serão abordados na subunidade 5.3.

5.2
REPETIÇÃO INDETERMINADA

Uma estrutura de repetição indeterminada repete um trecho do algoritmo de acordo com uma condição estabelecida, por exemplo, uma expressão lógica. A sintaxe da estrutura *enquanto...faca*, de acordo com o VisuAlg, está detalhada abaixo.

Sintaxe:

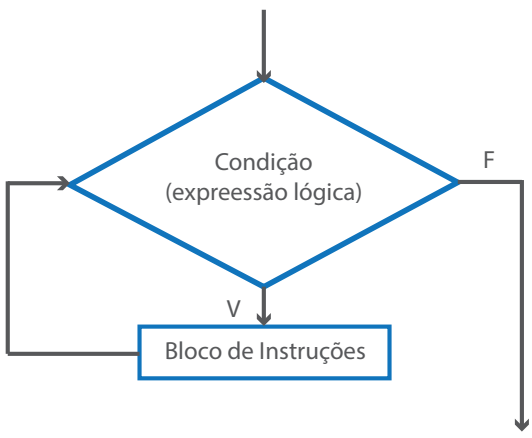
enquanto <expressão-lógica> faca
<bloco de instruções que serão repetidas>
fimenquanto

Esta estrutura também é conhecida como estrutura de repetição com teste no início, ou seja, o resultado da expressão lógica (condição) é testado no início e, se a condição for falsa, o trecho não é executado nenhuma vez. Um problema muito comum ocorre quando a repetição não termina (*loop* infinito). Isto ocorre quando a condição sempre é verdadeira.

ATENÇÃO: O problema do *loop* infinito deve ser controlado pelo programador, ou seja, deve existir uma linha de comando que faça com que a expressão lógica tenha seu valor alterado (conhecido como *condição de parada*), para que os comandos não sejam repetidos de forma indefinida.

O fluxograma da Figura 38 apresenta o esquema de execução do comando *enquanto... faca.*

FIGURA 38 – Fluxograma de Execução do Comando *enquanto...faca*



FONTE: dos autores, adaptado por NTE, 2017

Voltando novamente ao exemplo do cálculo da média de um aluno, vamos supor que desejamos calcular a média de uma turma de alunos, mas não sabemos o total de alunos da turma. Neste caso, a repetição é condicional, ou seja, o algoritmo precisará de uma condição de parada para ser concluído, caso contrário, o programa entrará em loop infinito. O programador pode definir a condição de parada, mas é importante que a mesma seja informada ao usuário. Por exemplo, no caso das médias, podemos estabelecer que a repetição será concluída quando o usuário digitar uma nota negativa para P1. O algoritmo ficaria codificado, então, como mostra o exemplo da Figura 39.

FIGURA 39 – Algoritmo para Calcular a Média usando *enquanto...faca*

```
algoritmo “Calcular_Media”
var
            P1, P2, Media: real
  Contador: inteiro
inicio
   escreva( “Digite a nota de P1:”)
  leia(P1)
  enquanto (P1 >= 0) faca
            escreva( "Digite a nota de P2:")
                        leia(P2)
                        Media <- (P1 + P2)/2
                        escreva( "Média do Aluno:", media)
                        se (Media>=7) entao
                           escreva( "O aluno está aprovado!")
                        senao
                           escreva( "O aluno está em exame!")
                        fimse
          escreva( “Digite a nota de P1:”)
           leia(P1)
           fimenquanto
          fimalgoritmo
```

FONTE: dos autores, adaptado por NTE, 2017

A implementação do algoritmo da Figura 39 no VisuAlg está representada na Figura 40.

FIGURA 40 – Estrutura de Repetição *enquanto...faca*

```
1  algoritmo "CalcularMedia"
2  // Função : Calcular a média de 10 alunos e classifica-los
3  // Autor : Autores do Livro de Introdução a Algoritmos
4  // Data : 16/04/2017
5  // Seção de Declarações
6  var
7     p1, p2, media: real
8  inicio
9  // Seção de Comandos
10    escreva("Digite a nota da prova p1: ")
11    leia(p1)
12    Enquanto (p1 >= 0) faca
13       escreva("Digite a nota da prova p2: ")
14       leia(p2)
15
16       media <- (p1 + p2 )/2
17       escreva("Média do Aluno é: ", media)
18
19       se(media>=7)entao
20          escreval(", ele está aprovado!")
21       senao
22          escreval(", ele está em Exame!")
23       fimse
24       escreval("Para encerrar digite uma nota negativa para p1: ")
25       escreva("Digite a nota da prova p1: ")
26       leia(p1)
27    fimenquanto
28 fimalgoritmo
```

FONTE: dos autores

Vamos analisar o algoritmo da Figura 40, e refletir sobre alguns pontos, dentre eles:

– O que acontecerá se na primeira leitura o usuário digitar, na linha 11, um valor negativo?
– Por que é preciso repetir a leitura da nota de P1, na linha 26, antes do fimenquanto?

O comando de repetição indeterminada *enquanto...faca* também pode ser utilizado para realizar a validação da entrada de dados. Por exemplo, quando o usuário digitar a nota de P1 (ou de P2), não devem ser permitidos valores menores que zero nem valores maiores que 10. Para evitar que estes dados sejam fornecidos erroneamente, criamos uma validação na entrada de dados, também conhecida como consistência na entrada de dados. Observe o trecho de algoritmo abaixo:

ATENÇÃO: Neste exemplo utilizamos o comando escreval para que, após ser mostrada a mensagem, haja uma quebra de linha

```
escreva( “Digite a nota de P1:”)
leia(P1)
    enquanto (P1 < 0) ou (P1 > 10) faca
        escreval( “Nota inválida! Digite novamente!”)
        escreval( “Digite a nota de P1:”)
leia(P1)
    fimenquanto
```

A seguir, na Figura 41, é apresentada, graficamente, a solução para o problema de validação das notas das provas no VisuAlg. Lembrem que a estrutura *enquanto... faca* repete um bloco de instruções enquanto a condição estabelecida for verdadeira.

FIGURA 41 – Estrutura de repetição enquanto...faca para validação de informações

```
1  algoritmo "CalcularMedia"
2  // Função : Calcular a média e valida as notas do aluno(a)
3  // Autor : Autores do Livro de Introdução a Algoritmos
4  // Data : 16/04/2017
5  // Seção de Declarações
6  var
7     p1, p2, media: real
8  inicio
9  // Seção de Comandos
10    escreva("Digite a nota da prova p1: ")
11    leia(p1)
12    enquanto (p1 < 0) ou (p1>10) faca
13       escreval("Nota da prova p1 INVALIDA!!!")
14       escreva("Digite NOVAMENTE a nota da prova p1: ")
15       leia(p1)
16    fimenquanto
17
18    escreva("Digite a nota da prova p2: ")
19    leia(p2)
20    enquanto (p2 < 0) ou (p2>10) faca
21       escreval("Nota da prova p2 INVALIDA!!!")
22       escreva("Digite NOVAMENTE a nota da prova p2: ")
23       leia(p2)
24    fimenquanto
25
26    media <- (p1 + p2 )/2
27    escreva("Média do Aluno(a) é: ", media)
28
29    se(media>=7)entao
30       escreval(", ele(a) está aprovado!")
31    senao
32       escreval(", ele(a) está em Exame!")
33    fimse
34 fimalgoritmo
```

FONTE: dos autores

As estruturas de repetição, iniciadas nas **linhas 12** e **20**, são indeterminadas, pois não sabemos quantas vezes o mesmo erro poderá ser cometido, a não ser que tenhamos um limite de erro, como no caso dos bancos, onde existe um número limite de tentativas erradas, principalmente na entrada da senha dos clientes. Embora tenhamos inserido a validação das notas de *P1* e *P2* o algoritmo lê informações apenas de um único aluno (a). Como podemos alterá-lo de tal forma que o algoritmo possa ler informações de vários alunos?

5.3
VARIÁVEIS ESPECIAIS: ACUMULADORES E CONTADORES

Nos algoritmos com repetição (contada e/ou indeterminada) é comum a utilização de variáveis com funções especiais: contadores e acumuladores.

5.3.1 Acumuladores

Os acumuladores são variáveis que acumulam os valores de outras variáveis. Eles têm seus valores alterados em intervalos variáveis. Esta alteração pode ser realizada por meio de uma soma, subtração ou até por meio de uma multiplicação. A sintaxe de um acumulador é a seguinte:

Acumulador <- Acumulador + <Expressão>
Acumulador <- Acumulador - <Expressão>
*Acumulador <- Acumulador * <Expressão>*

Exemplos:

Soma <- Soma+A
Total_Valores <- Total_Valores - Valor_Unitario
*Produto <- Produto * A*

Por exemplo, no cálculo da média para uma turma com 10 alunos, poderíamos calcular a média geral da turma. Esta média geral é a soma de todas as médias (acumulador), dividida pelo número de alunos da turma. O algoritmo incluindo o cálculo da média geral é apresentado na Figura 42.

FIGURA 42 – Algoritmo para Calcular a Média com Cálculo da Média Geral

```
algoritmo “Calcular_Media”
var
  P1, P2, Media, somaMedias: real
 Contador: inteiro
Inicio
            para Contador de 1 ate 10 faca
                  escreva( "Digite a nota de P1:")
                  leia(P1)
                  escreva( "Digite a nota de P2:")
                  leia(P2)
                  Media <- (P1 + P2 )/2
                  Escreva( "Média do Aluno:", Media)
                  somaMedias  <-  somaMedias + Media
                  se (media>=7) entao
                    escreva( "O aluno está aprovado!")
                  senao
                    escreva( "O aluno está em Exame!")
                  fimse
             fimpara
             escreva(“Média Geral da Turma:”, somaMedias/10)
          fimalgoritmo
```

FONTE: dos autores, adaptado por NTE, 2017

Ao analisarmos o exemplo acima, foi incluída uma variável acumuladora *somaMedias*, que armazena a média total dos alunos da turma. No final do algoritmo, após o término da repetição (para que a média geral não seja mostrada 10 vezes), é impressa a média geral da turma.

5.3.2 Contadores

Os contadores são variáveis que têm seus valores alterados em intervalos constantes. Esta alteração pode ser para mais ou para menos. A sintaxe de um contador é a seguinte:

Contador <- *Contador* + <*Valor Constante*>
Contador <- *Contador* - <*Valor Constante*>

Exemplos:

Contador <- *Contador+1*
Conta <- *Conta+2*
C <- *C-1*

Por exemplo, no cálculo da média para uma turma com 10 alunos, poderíamos mostrar o número de alunos aprovados e o número de alunos que ficaram em exame. Para calcularmos estes resultados, precisamos contar quantos alunos ficaram em cada uma das situações.

O algoritmo incluindo a contagem de alunos aprovados e em exame é apresentado na Figura 43.

FIGURA 43 – Algoritmo para Calcular a Média com contagem de Alunos Aprovados e em Exame

```
algoritmo “Calcular_Media”
var
  P1, P2, Media, somaMedias: real
 Contador, Aprovados, emExame: inteiro
inicio
                    para Contador de 1 ate 10 faca
                          escreva( "Digite a nota de P1:")
                          leia(P1)
                          escreva( "Digite a nota de P2:")
                          leia(P2)
                          Media <-  (P1 + P2 )/2
                          escreva( "Média do Aluno:", Media)
                          somaMedias <- somaMedias + Media
                          se (Media>=7) entao
                             escreva("O aluno está aprovado!")
                             Aprovados <- Aprovados + 1
                          senao
                             escreva("O aluno está em exame!")
                             emExame <-  emExame + 1
                          fimse
                    fimpara
                    escreval(Média Geral da Turma:”, somaMedias/10)
                    escreval( “Número de alunos aprovados:”, Aprovados)
                    escreval(“Número de alunos em exame:”, emExame)
fimalgoritmo
```

FONTE: dos autores

No exemplo apresentado na Figura 43, foram incluídas as variáveis contadoras Aprovados e em Exame, que armazenam o total de alunos em cada situação. No final do algoritmo, após o término da repetição (para que estes resultados não sejam impressos 10 vezes), são mostrados os valores finais dos contadores.

Podemos implementar este último exemplo apenas com um dos contadores (*aprovados ou emExame*). Pense de que forma podemos implementá-lo?

6

ESTRUTURAS DE DADOS BÁSICAS

INTRODUÇÃO

Na unidade anterior, a Unidade 5 – Algoritmos Com Repetição, implementamos um algoritmo que solicita as informações de 10 alunos e calcula a média das notas das provas. Esse algoritmo não armazena todas as informações, apenas executa as leituras e os cálculos. Neste sentido, toda vez que as variáveis recebem novas informações, os dados armazenados anteriormente são perdidos, ou seja, as notas das provas e as médias não são armazenadas, de forma que possamos verificar, posteriormente, quais as notas atribuídas para cada um dos alunos.

Agora vamos imaginar que, além da média, temos que calcular a quantidade de alunos que obtiveram nota acima e abaixo da média. Vejam que para fazer esses cálculos dependemos do valor da média, valores estes que não estão mais armazenadas no algoritmo.

Para resolver tal problema temos duas soluções. Na primeira solução, temos que redigitar as informações de todos os alunos, o que gera, além do retrabalho, possíveis erros de digitação. A segunda solução seria a utilização de uma variável capaz de armazenar vários valores simultaneamente, de tal forma que possamos acessar cada um deles independente uns dos outros, e tais valores não sejam apagados enquanto o algoritmo estiver em execução.

Sendo assim, nesta unidade vamos aprender a implementar a segunda solução, por meio de estruturas de dados básicas, ou também chamadas de variáveis compostas. Uma variável composta é um conjunto de variáveis simples, por exemplo, do tipo Inteiro, identificadas pela concatenação de índices entre colchetes, ao identificador da variável composta.

As ***variáveis compostas*** se dividem em **homogêneas**, sendo os vetores (uma dimensão) e matrizes (mais de uma dimensão), e **heterogêneas**, que são os registros. Neste livro, vamos trabalhar com vetores e matrizes, deixando os registros para uma próxima oportunidade.

6.1 VARIÁVEIS COMPOSTAS HOMOGÊNEAS

Vamos definir variáveis compostas homogêneas como sendo estruturas de dados que se caracterizam por apresentarem um conjunto de variáveis do mesmo tipo básico, que nada mais é do que uma forma de organizar os dados armazenados na memória.

Estas estruturas de dados são abstratas, ou seja, não existem fisicamente na memória. Na memória do computador os dados estão armazenados todos da mesma forma. Os algoritmos, por meio das estruturas de dados, organizam e visualizam estes dados de diferentes formas, visando facilitar a programação e/ou o entendimento do usuário. Além disso, elas são referenciáveis pelo mesmo nome e individualizadas entre si por meio de sua posição dentro desse conjunto, utilizando-se índices. Ainda podemos distingui-las quanto à dimensionalidade, podendo ser unidimensionais ou multidimensionais.

Quando forem unidimensionais, as variáveis compostas possuem um único índice e são chamadas de vetores (ou matrizes de uma dimensão). Quando possuírem dois índices (linha e coluna, tais como tabelas ou planilhas) elas são denominadas de matrizes. Existem ainda alguns casos, que na prática são pouco utilizados, em que se têm três ou mais índices.

Neste panorama, os vetores precisam de um único índice para localizar os elementos que foram armazenados. Já as matrizes exigem dois índices, um para posicionar a linha e outro a coluna, para localizar os elementos. Vejam que a dimensão destas variáveis recebe como nome o número de índices necessários à localização de um componente dentro da variável indexada. A seguir serão detalhados os vetores, que são unidimensionais, e as matrizes, que são multidimensionais.

6.2 VETORES

Vamos iniciar os nossos estudos sobre vetores. Por definição, eles são estruturas de dados, ou seja, são formas de organizar os dados em um algoritmo, tornando as informações referenciáveis como um todo. Também são homogêneos, porque, a princípio, armazenam o mesmo tipo de dado.

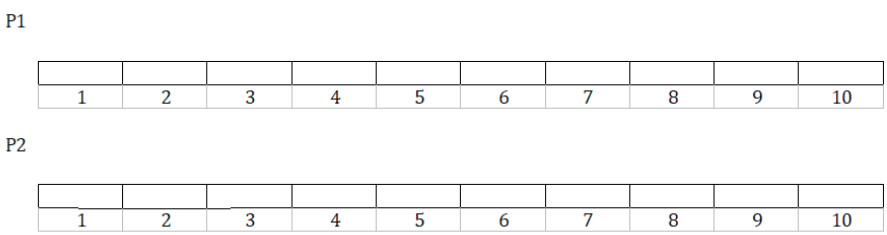
Quando declararmos um vetor, estamos reservando na memória principal do computador uma série de espaços para uso da variável, daquele tipo. O nome do vetor aponta para a base dos espaços e o seu início indica a posição relativa do primeiro elemento referenciado, sendo acessado por meio do índice.

Seguindo o exemplo utilizado nos capítulos anteriores, o cálculo da média, vamos supor que tenhamos uma turma de 10 alunos. Queremos imprimir, no final do algoritmo, um relatório geral, contendo todas as notas de P1 e P2, dos 10 alunos. Se utilizarmos variáveis simples, a única maneira é armazenarmos as 20 notas (10 de P1 e 10 de P2) em variáveis diferentes, o que torna a manipulação dos dados do programa um tanto quanto complicada. Por exemplo, se a turma aumentar a quantidade de alunos para 50, teremos que ter 100 variáveis. Neste caso, a solução mais viável é a utilização de variáveis compostas, como os vetores. No exemplo a seguir são declarados dois conjuntos do tipo vetor, cada um armazenando 10 informações do tipo *Real*:

```
var
        P1: vetor [1..10] de real
        P2: vetor [1..10] de real
```

Estes vetores são representados graficamente na Figura 44.

FIGURA 44 – Representação Gráfica dos Vetores P1 e P2



FONTE: dos autores

Internamente, na memória do computador, estes dois vetores ocuparão o mesmo espaço necessário para 20 variáveis simples. A vantagem na utilização dos vetores é a redução na complexidade e tamanho do código do programa, visto que as rotinas de manipulação dos dados como, por exemplo, a leitura, atribuição e impressão

poderão ser realizadas para todo o conjunto de dados, por meio de comandos de repetição contada. Utiliza-se a repetição contada, já que sabemos a dimensão dos vetores antecipadamente.
Por exemplo, vamos escrever um algoritmo que permita a leitura das notas de P1 de toda a turma, utilizando o exemplo de vetores, como mostra a Figura 45.

FIGURA 45 – Exemplo de Utilização de Vetores

```
algoritmo "Leitura_das_Notas"
var
                P1: vetor [1..10] de real
                Contador: Inteiro
        inicio
                para Contador de 1 ate 10 faça
                        escreva( "Digite a nota de P1:")
                        leia(P1[contador])
                fimpara
        fimalgoritmo
```

FONTE: dos autores, adaptado por NTE, 2017

Uma modificação que poderia auxiliar na entrada de dados é apresentar o número do aluno ao usuário (ou seja, o valor da variável *Contador*, que é o índice utilizado para armazenar as notas em cada uma das posições do vetor). A Figura 46 apresenta o algoritmo contendo a indicação do número do aluno ao apresentar a mensagem para a entrada de dados.

FIGURA 46 – Exemplo de Utilização de Vetores com Identificação do Índice na Entrada de Dados

```
algoritmo "Leitura_das_Notas"
var
                P1: vetor [1..10] de real
                Contador: Inteiro
        inicio
                para Contador de 1 ate 10 faça
                        escreva( "Digite a nota de P1 do aluno:", Contador)
                        leia(P1[contador])
                fimpara
        fimalgoritmo
```

FONTE: dos autores, adaptado por NTE, 2017

No próximo exemplo (Figura 47), vamos construir um algoritmo para ler as notas de P1 e P2 (em seus respectivos vetores) e calcular a média de cada um dos alunos da turma, que será armazenada em um terceiro vetor (*Media*). Além disso, vamos imprimir a média geral da turma.

FIGURA 47 – Exemplo de Algoritmo utilizando 3 vetores

```
algoritmo " Leitura_das_Notas”
var
                P1: vetor [1..10] de real
                P2: vetor [1..10] de real
                Media: vetor [1..10] de real
                MediaGeral: real
                Contador: inteiro
        inicio
                //Leitura das notas de P1
                para Contador de 1 ate 10 faca
                        escreva(“Digite a nota de P1:”)
                        leia(P1[contador])
                fimpara

                //Leitura das notas de P2
                para Contador de 1 ate 10 faca
                        escreva( “Digite a nota de P2:”)
                        leia(P2[contador])
                fimpara

                //Cálculo das Médias
                para Contador de 1 ate 10 faca
                        Media[Contador] <- (P1[Contador]+P2[Contador])/2
                        MediaGeral <- MediaGeral+Media[contador]
                fimpara

                //Impressão dos Resultados
                para Contador de 1 ate 10 faca
                        escreva(“Média do aluno:”,Contador,”=”,Media[Contador])
                fimpara
                escreva(“Média Geral da Turma:”,MediaGeral/10)
        fimalgoritmo
```

FONTE: dos autores, adaptado por NTE, 2017

ATENÇÃO: Como armazenamos todas as 10 notas de P1 e todas as 10 notas de P2, de forma individualizada, podemos calcular a média geral após termos encerrado o laço de repetição contada *(para)*. Até o encerramento da execução do algoritmo, teremos estes valores armazenados na memória

A leitura dos dados de P1 e P2, bem como o cálculo e a impressão das médias, poderia ter sido realizada dentro de um único laço de repetição contada (*para*). Neste exemplo, utilizamos 4 laços para apenas para melhorar a compreensão do código.

Agora vejamos como fica algoritmo apresentado na Figura 47 codificado no VisuAlg (Figura 48).

FIGURA 48 – Exemplo de Algoritmo utilizando 3 vetores codificado no VisuAlg

```
1  algoritmo "CalculoMedia"
2  // Função : Cular a média de 10 alunos e classifica-los, usando vetores
3  // Autor :Autores do Livro de Introdução a Algoritmos
4  // Data : 21/04/2017
5  // Seção de Declarações
6  var
7     p1: vetor [1..10] de real
8     p2: vetor [1..10] de real
9     media: vetor [1..10] de real
10    contador:inteiro
11    mediaGeral: real
12 inicio
13 // Seção de Comandos
14   //Leitura das notas de p1
15    para contador de 1 ate 10 faca
16       Escreva("Digite a nota de p1, para o",contador,"° aluno: ")
17       leia(p1[contador])
18    fimpara
19
20   //Leitura das notas de p2
21    para contador de 1 ate 10 faca
22       Escreva("Digite a nota de p2, para o",contador,"° aluno: ")
23       leia(p2[contador])
24    fimpara
25
26
27    //Cálculo das Médias
28    para contador de 1 ate 10 faca
29       media[contador]:=(p1[contador]+p2[contador])/2
30       mediaGeral:=mediaGeral+media[contador]
31    fimpara
32
33    //Impressão dos Resultados
34    para contador de 1 ate 10 faca
35       escreval("Média do ",contador,"° aluno = ",media[contador])
36    fimpara
37
38    escreval("Média Geral da Turma = ",mediaGeral/10)
39 fimalgoritmo
```

FONTE: dos autores

Analisando a Figura 48, vemos que as declarações das variáveis do tipo vetor, ocorrem nas **linhas 7, 8 e 9**. É importante observar que a declaração dos vetores obedece a sintaxe: <*nome das variáveis*>: vetor "*["índice inicial...índice final"]" de* <*tipo-de-dado*>, no nosso exemplo temos: P1: *vetor [1..10] de real.*

Na **linha 15**, se inicia o código para leitura das notas das provas p1, dos 10 alunos, por meio da estrutura de repetição *para*. A **linha 16** apresenta a mensagem para o usuário "Digite a nota de P1, para o 1° aluno:". Já a **linha 17** faz a leitura e atribuição da nota para o vetor *P1[contador]*. A leitura das notas de P2 é semelhante à leitura das notas de P1, sendo apresentada na **linha 20** até a 24.

O cálculo da média de cada aluno se inicia na **linha 28**, por meio da estrutura de repetição *para.*. Na linha 29 ocorre o cálculo propriamente dito, que é *(P1[contador]+P2[contador])/2* e a atribuição desse resultado ao vetor *Media[contador]*. Na sequência, na **linha 30**, realiza-se a soma de todas as médias por meio da variável *MediaGeral*, que é um acumulador.

A impressão da média de cada aluno é realizada nas **linhas 34 até 36**, e por fim, imprime o resultado da média geral da turma, na **linha 38**.

Vamos pensar em como podemos melhorar o algoritmo apresentado na Figura 48? Uma possível solução é diminuir a utilização de estruturas de repetição. Veja como fica a solução com apenas duas estruturas de repetição (Figura 49).

FIGURA 49 – Cálculo da média usando vetores de forma otimizada.

```
algoritmo "CalculoMedia"
// Função : Cular a média de 10 alunos e classifica-los, usando vetores
// Autor :Autores do Livro de Introdução a Algoritmos
// Data : 21/04/2017
// Seção de Declarações
var
   p1, p2, media: vetor [1..10] de real
   contador:inteiro
   mediaGeral: real
inicio
// Seção de Comandos
   para contador de 1 ate 10 faca
      //Leitura das notas de p1
      Escreva("Digite a nota de p1, para o",contador,"° aluno: ")
      leia(p1[contador])
      //Leitura das notas de p2
      Escreva("Digite a nota de p2, para o",contador,"° aluno: ")
      leia(p2[contador])
      //Cálculo das Médias
      media[contador]:=(p1[contador]+p2[contador])/2
      mediaGeral:=mediaGeral+media[contador]
   fimpara

   //Impressão dos Resultados
   para contador de 1 ate 10 faca
      escreval("Média do ",contador,"° aluno = ",media[contador])

   fimpara
   escreval("Média Geral da Turma = ",mediaGeral/10)
fimalgoritmo
```

FONTE: dos autores

6.2.1 Ordenação de Vetores

Quando trabalhamos com vetores, podemos aplicar métodos que permitem a ordenação dos dados (de forma crescente ou decrescente). Um dos métodos mais conhecidos para a classificação (ou ordenação) é o método da bolha, também conhecido como *bubble sort*. Este método é bastante popular, devido a sua facilidade de implementação.

O método da bolha consiste em varrer o vetor, comparando os elementos 2 a 2 (vizinhos entre si). Neste sentido aplicamos o critério de comparação maior (>) quando queremos que o vetor seja ordenado de forma crescente e menor (<) quando queremos a ordenação seja decrescente. Caso os valores estejam fora de ordem, são trocados de posição. Esta comparação é realizada até o final do vetor. Ao concluir a primeira etapa de classificação, que corresponde à primeira "varredura" no vetor, o último elemento já estará colocado no seu devido lugar. Sendo assim, na próxima varredura, este elemento poderá ser descartado. A segunda "varredura" será, então, apenas até o penúltimo elemento. Este processo é repetido até que seja realizado um número de varreduras igual ao número de elementos a serem ordenados menos 1.

A Figura 50 mostra o algoritmo para ordenação crescente de um vetor com 10 elementos.

FIGURA 50 – Exemplo de Algoritmo para Ordenação de um Vetor

```
algoritmo “Ordenacao_Metodo_Bolha”
var
                    Vet: vetor [1..10] de inteiro
          Contador, Ordenar, Auxiliar: inteiro
Inicio
           //Entrada de dados no vetor
para Contador de 1 ate 10 faca
                    escreva(“Digite um número:”)
                    leia(Vet[Contador])
fimpara
//Ordenação do Vetor
                    Ordenar <- 10
enquanto (Ordenar > 1) faca
                    para Contador de 1 ate Ordenar-1 faca
                              se (Vet[Contador]>Vet[Contador+1]) entao
                                Auxiliar <- Vet[Contador]
                                Vet[Contador] <- Vet[Contador+1]
                                Vet[Contador+1] <- Auxiliar
                              Fimse
                    Fimpara
                    Ordenar <- Ordenar-1
fimenquanto
           //Impressão do Vetor Ordenado
escreval(“Vetor em ordem crescente:”)
para Contador de 1 ate 10 faca
                    escreval(Vet[Contador])
fimpara
fimalgoritmo
```

FONTE: dos autores, adaptado por NTE, 2017

Vejamos, agora, o mesmo algoritmo codificado noVisuAlg (Figura 51).

FIGURA 51 – Ordenação de vetor usando o método Bolha.

```
1  algoritmo "OrdenacaoMetodoBolha"
2  // Função : Ordenação crescente de um vetor com 10 elementos
3  // Autor :Autores do Livro de Introdução a Algoritmos
4  // Data : 21/04/2017
5  // Seção de Declarações
6  var
7     vet: vetor [1..10] de inteiro
8     ordenar, auxiliar, contador:inteiro
9
10 inicio
11 // Seção de Comandos
12    //Entrada de dados no vetor
13    para contador de 1 ate 10 faca
14       escreva("Digite um número: ")
15       leia(vet[contador])
16    fimpara
17
18    //Ordenação do vet
19    ordenar <- 10
20    enquanto (ordenar > 1) faca
21       para contador de 1 ate ordenar-1 faca
22          Se (vet[contador]>vet[contador+1]) entao
23              auxiliar <- vet[contador]
24              vet[contador] <- vet[contador+1]
25              vet[contador+1] <- auxiliar
26          fimse
27       fimpara
28       ordenar:=ordenar-1
29     fimenquanto
30
31    //Impressão do vet Ordenado
32    Escreva("Vetor em ordem crescente: ")
33    para contador de 1 ate 10 faca
34      Escreva(vet[contador])
35    fimpara
36 fimalgoritmo
```

```
Digite um número: 3
Digite um número: 6
Digite um número: 9
Digite um número: 8
Digite um número: 5
Digite um número: 2
Digite um número: 1
Digite um número: 4
Digite um número: 7
Digite um número: 0
Vetor em ordem crescente:  0 1 2 3 4 5 6 7 8 9
*** Fim da execução.
*** Feche esta janela para retornar ao Visualg.
```

FONTE: dos autores

6.2.2 Pesquisa em Vetores

Além da ordenação de vetores, estudada anteriormente, outra funcionalidade que pode ser bastante utilizada é a pesquisa. Os métodos de pesquisa servem para verificar se um determinado valor faz ou não parte de um conjunto. Por exemplo, para verificar se um aluno está cadastrado em uma turma, temos que fazer uma pesquisa. O método mais comum de pesquisa em vetores é a pesquisa sequencial.

A pesquisa sequencial consiste em varrer todo o vetor à procura do valor fornecido e verificar a sua existência. Ao final da varredura, o usuário deve ser informado se o valor foi encontrado ou não. Para realizar tal pesquisa, é necessária a inserção de uma variável de controle, também conhecida como flag ou bandeira. Esta variável permitirá identificar se a pesquisa encontrou resultados ou não.

Vamos analisar o algoritmo apresentado na Figura 52. Ele demonstra uma pesquisa sequencial em um vetor que tem cadastrado os códigos de 10 produtos de uma loja.

FIGURA 52 – Algoritmo de Pesquisa Sequencial em Vetores

```
algoritmo "Pesquisa_Sequencial"
var
                    Produtos: vetor [1..10] de inteiro
        Contador, Codigo, Achou: inteiro
Inicio
          //Entrada de Dados no Vetor
para Contador de 1 ate 10 faca
              escreva("Digite o código do produto:")
              leia(Produtos[Contador])
fimpara
//Pesquisa Sequencial
          escreva( "Digite um código a ser pesquisado:")
          leia(Codigo)
          Achou <- 0 //Inicialização da Flag
          para Contador de 1 ate 10 faca
              se (Produtos[Contador]=Codigo) entao
                    Achou <- 1
                    escreva("Código está cadastrado!")
              fimse
         fimpara
          se (Achou=0) entao
            escreva( "Código do produto não existe!")
          fimse
          fimalgoritmo
```

FONTE: dos autores, adaptado por NTE, 2017

A variável Achou é utilizada como uma flag. No início da pesquisa, esta flag recebe o valor zero. Se a pesquisa for concluída sem sucesso, ou seja, se for realizada a varredura em todo o vetor e o código do produto não for encontrado, esta flag manterá o valor zero, caso contrário assumirá o valor 1. Sendo assim, sabemos que, se a flag manteve o valor zero, é porque o código não foi encontrado.

Este mesmo algoritmo, codificado no VisuAlg, é apresentado na Figura 53.

FIGURA 53 – Método de Pesquisa Sequencial

```
algoritmo "PesquisaSequencial"
// Função : Pesquisa sequencialem um vetor com 10 elementos
// Autor :Autores do Livro de Introdução a Algoritmos
// Data : 21/04/2017
// Seção de Declarações
var
   produtos: vetor [1..10] de inteiro
   contador, codigo, achou:inteiro
inicio
// Seção de Comandos
   //Entrada de dados no vetor
   para contador de 1 ate 10 faca
      escreva("Digite o código do produto: ")
      leia(produtos[contador])
   fimpara

   //Pesquisa Sequencial
   escreva("Digite um código a ser pesquisado: ")
   leia(codigo)
   Achou<-0

   para contador de 1 ate 10 faca
         Se (produtos[contador]=codigo) entao
            achou:=1
            escreva("Código está cadastrado!")
         fimse
   fimpara

   Se (achou=0) entao
       escreva("Código do produto não existe!")
   fimse
fimalgoritmo
```

```
Digite o código do produto: 11
Digite o código do produto: 88
Digite o código do produto: 55
Digite o código do produto: 22
Digite o código do produto: 44
Digite o código do produto: 77
Digite o código do produto: 32
Digite o código do produto: 99
Digite o código do produto: 65
Digite o código do produto: 32
Digite um código a ser pesquisado: 32
Código está cadastrado!Código está cadastrado!
*** Fim da execução.
*** Feche esta janela para retornar ao Visualg.
```

FONTE: dos autores

6.3 MATRIZES

Já estudamos vetores e vimos que eles precisam de apenas uma variável indexadora para que seja possível manipular os valores armazenados nos mesmos. Já as matrizes precisam de duas variáveis indexadoras, um para as linhas e outra para as colunas. Estas matrizes, que usam duas variáveis indexadoras, possuem duas dimensões e são chamadas bidimensionais. Também existem matrizes com mais dimensões, por exemplo, três, sendo chamadas tridimensionais, e assim por diante.

Com o objetivo de facilitar os estudos, vamos limitar a matriz somente com duas dimensões, logo a definição das matrizes tem a seguinte sintaxe no VisuAlg:

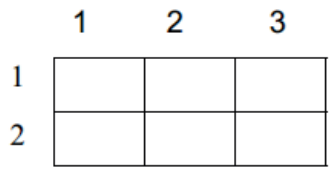
<nome da variável>: vetor "["índice inicial...índice final", "índice inicial...índice final"]" de <tipo-de-dado>

Para criar a estrutura de dados, de acordo com a sintaxe acima, no algoritmo, antes é necessário declarar uma matriz, que é composta de um nome e dimensões. No exemplo a seguir, vamos criar uma matriz bidimensional, chamada Notas, contendo 2 linhas e 3 colunas (ou seja, poderá armazenar 6 valores):

```
var
        Notas: vetor [1..2,1..3] de real
```

A matriz Notas é representada graficamente como mostra a Figura 54.

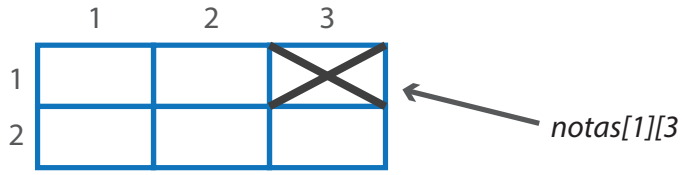
FIGURA 54 – Representação gráfica da Matriz *Notas*



FONTE: dos autores

Tomando como base a Figura 63, que é uma representação gráfica da matriz *Notas*, cujas dimensões são 2 linhas e 3 colunas, vamos referenciar um elemento desta matriz. Para isso são necessários 2 índices, pois a matriz *Notas* é bidimensional (duas dimensões), o primeiro índice indica a linha e o segundo, a coluna. Por exemplo, vamos referenciar Notas[1,3] – vejam que o elemento está posicionado na linha número 1 e na coluna número 3, como mostra a Figura 55.

FIGURA 55 – Localização do elemento [1,3] na matriz Notas



FONTE: dos autores, adaptado por NTE, 2017

Internamente, na memória do computador, a matriz ocupa um espaço de 6 variáveis simples. Tomando novamente o exemplo para armazenar as notas das provas *P1* e *P2*, vamos escrever um algoritmo que permita a leitura das notas da turma, com 10 alunos, utilizando uma matriz bidimensional, contendo 2 linhas (uma linha para as notas de P1 e uma linha para as notas de P2) e 10 colunas (uma coluna para cada nota), como mostra o exemplo da Figura 56.

FIGURA 56 – Exemplo de Algoritmo com Matriz Bidimensional

```
algoritmo "Leitura_das_Notas"
var
        Notas: vetor [1..2,1..10] de real
        contLinha, contColuna: inteiro
    inicio
        para contColuna de 1 ate 10 faca //contador para colunas
            para contLinha de 1 ate 2 faca //contador para linhas
                escreva("Digite a nota de P", contLinha ,":")
                leia(Notas[contLinha, contColuna])
        fimpara
    fimalgoritmo
```

FONTE: dos autores, adaptado por NTE, 2017

No algoritmo da Figura 56, a matriz *Notas[2][10]* é percorrida, para armazenar as notas, por coluna, ou seja, fixa o índice da coluna e varia o índice da linha.

Dando sequência ao exemplo, vamos construir um algoritmo para ler as notas de P1 e P2 e calcular a média de cada um dos alunos da turma, e ao final, armazenar estas informações numa matriz. Além disso, vamos imprimir a média geral da turma, como mostra o exemplo da Figura 57.

FIGURA 57 – Exemplo de Algoritmo com Matriz Bidimensional

```
algoritmo “Leitura_das_Notas”
var
        Notas: vetor [1..3,1..10] de real
        contLinha, contColuna: inteiro
        SomaP1eP2, mediaGeral: real
    inicio
        //Leitura das notas de P1 e P2
        para contColuna de 1 ate 10 faca //contador para colunas
            somaP1eP2 <- 0
            para contLinha de 1 ate 2 faca //contador para linhas
                escreva(“Digite a nota de p”, contLinha ,”:”)
                leia(Notas[contLinha, contColuna])
                somaP1eP2 <-  somaP1eP2 + Notas[contLinha, contColuna]
            fimpara
            //Após o comando para contLinha terá armazenado o número 3 (linha da média)
            Notas[contLinha, contColuna] <-  somaP1eP2/2
            mediaGeral <- mediaGeral+ Notas[contLinha, contColuna]
        fimpara

        //Impressão dos Resultados
        para contColuna de 1 ate 10 faca //contador para colunas
            para contLinha de 1 ate 2 faca //contador para linhas
                escreva(“A nota P”,contLinha, “é: “,Notas[contLinha, contColuna])
            fimpara
            escreva(“A média do aluno”, contColuna, “é: “,Notas[contLinha, contColuna])
        fimpara
        escreva(“Média Geral da Turma:”,mediaGeral/10)
    fimalgoritmo
```

FONTE: dos autores, adaptado por NTE, 2017

Agora vejamos como fica algoritmo descrito acima, codificado no VisuAlg (Figura 58).

FIGURA 58 – Exemplo de Algoritmo com Matriz Bidimensional no VisuAlg

```
1  algoritmo "MediaUsandoMatriz"
2  // Função : Calcular a média das notas p1 e p2 usando matriz
3  // Autor :Autores do Livro de Introdução a Algoritmos
4  // Data : 21/04/2017
5  // Seção de Declarações
6  var
7     notas: vetor [1..3,1..10] de real
8     contLinha, contColuna: inteiro
9     somaP1eP2,mediaGeral: real
10 inicio
11 // Seção de Comandos
12       //Entrada de dados para a matriz das notas de p1 e p2
13       para contColuna de 1 ate 3 faca //contador para colunas
14          somaP1eP2<-0
15          para contLinha de 1 ate 2 faca //contador para linhas
16             escreva("Digite a nota de p", contLinha ," do aluno",contColuna,": ")
17             leia(notas[contLinha,contColuna])
18             somaP1eP2:= somaP1eP2+ notas[contLinha,contColuna]
19          fimpara
20          notas[contLinha,contColuna]:= somaP1eP2/2
21          mediaGeral:=mediaGeral+ notas[contLinha,contColuna]
22       fimpara
23
24       //Impressão dos Resultados
25       para contColuna de 1 ate 3 faca //contador para colunas
26          para contLinha de 1 ate 2 faca //contador para linhas
27             escreval("A nota p",contLinha, " é: ",notas[contLinha,contColuna])
28          fimpara
29          escreval("A media do aluno", contColuna, " é:",notas[contLinha,contColuna])
30       fimpara
31       escreval("Média Geral da Turma:",mediaGeral/3)
32 fimalgoritmo
```

```
Digite a nota de p 1 do aluno 1: 2
Digite a nota de p 2 do aluno 1: 2
Digite a nota de p 1 do aluno 2: 3
Digite a nota de p 2 do aluno 2: 3
Digite a nota de p 1 do aluno 3: 8
Digite a nota de p 2 do aluno 3: 8
A nota p 1 é:  2
A nota p 2 é:  2
A media do aluno 1 é: 2
A nota p 1 é:  3
A nota p 2 é:  3
A media do aluno 2 é: 3
A nota p 1 é:  8
A nota p 2 é:  8
A media do aluno 3 é: 8
Média Geral da Turma: 4.33333333333333

*** Fim da execução.
*** Feche esta janela para retornar ao Visualg.
```

FONTE: dos autores

6.3.1 Operações entre matrizes

As operações entre matrizes possuem regras previamente definidas na matemática. Desta forma, quando vamos criar um algoritmo para realizar tais operações, também temos que obedecer às regras já definidas. A seguir veremos como implementar algumas operações entre matrizes, dentre elas a adição, subtração e multiplicação.

Vamos começar com a adição. Como regra, esta operação só pode ser feita por matrizes de mesma ordem, ou seja, que possuem o mesmo número de linhas e colunas. Assim, sejam as matrizes A e B, de tal forma que $A = (a_{ij})_{m \times n}$ e $B = (b_{ij})_{m \times n}$, podemos diz que a soma de A + B é dada pela matriz $C = (c_{ij})_{m \times n}$, sendo $c_{ij} = a_{ij} + b_{ij}$ para todo i compreendido no intervalo $1 \leq i \leq m$ e para todo j compreendido no intervalo $1 \leq j \leq n$.

Agora, vamos escrever o algoritmo que permita a leitura das matrizes A e B, e faça a soma das mesmas, como mostra a Figura 59.

FIGURA 59 – Exemplo de Algoritmo com Soma de Matrizes

```
algoritmo "soma_de_matrizes"
var
                    matA: vetor [1..2,1..2] de inteiro
                    matB: vetor [1..2,1..2] de inteiro
                    matC: vetor [1..2,1..2] de inteiro
                    i, j, k: inteiro
          inicio
//Entrada de dados matA
escreva "Carregando a matriz A:"
Para i de 1 ate 2 faca //linhas de A
      para j de 1até 2 faça //colunas de A
           escreva( "matA [", i, ",", j, "]=")
           leia(matA [i,j])
      fimpara
fimpara

//Entrada de dados matB
escreva("Carregando a matriz matB:")
para i de 1 ate 2 faca //linhas de matB
      para j de 1 ate 2 faca //colunas de matB
           escreva("matB [", i, ",", j, "]=")
           leia(matB [i,j])
      fimpara
fimpara

//Cálculo da matriz soma matC
para i de 1 ate 2 faca //linhas de matA= linhas de matB = linhas de matC
      para j de 1 ate 2 faca // colunas de matA= colunas de matB = colunas de matC
         matC [i,j] <- matA[i,j] + matB[i,j]
      fimpara
fimpara

//Saída
Escreva("Matriz Soma matC:" )
para i  de 1 ate 2 faca // linhas de C
      escreva("linha = ", i)
      para j de 1 ate  2 faca // colunas de matC
         escreva(matC [i,j])
      fimpara
fimpara
fimalgoritmo
```

FONTE: dos autores, adaptado por NTE, 2017

Agora vejamos como fica algoritmo descrito anteriormente, codificado em VisuAlg (Figura 60).

FIGURA 60 – Soma de Matrizes

```
algoritmo "SomaDeMatrizes"
// Função : Somar a MatA pela MatB
// Autor :Autores do Livro de Introdução a Algoritmos
// Data : 22/04/2017
// Seção de Declarações
var
   matA: vetor[1..2,1..2] de real
   matB: vetor[1..2,1..2] de real
   matC: vetor[1..2,1..2] de real
   i, j, k: inteiro
inicio
      // Seção de Comandos
      //Entrada de dados para a matA
      escreval("Carregando a matriz A:")
      Para i de 1 ate 2 faca //linhas de A
         Para j de 1 ate 2 faca //colunas de A
            escreva("matA [",i,",",j,"]=")
            leia(matA[i,j])
         FimPara
      FimPara

      //Entrada de dados para a matB
      escreval("Carregando a matriz B:")
      Para i de 1 ate 2 faca //linhas de B
         Para j de 1 ate 2 faca //colunas de B
            escreva("matB[",i,",",j,"]=")
            leia(matB[i,j])
         FimPara
      FimPara

      //Cálculo matC = MatA + MatB
      Para i de 1 ate 2 faca //linhas
         Para j de 1 ate 2 faca //colunas
               matC[i,j] <- matA[i,j] + matB[i,j]
         FimPara
      FimPara

      //Impressão dos Resultados
      escreval("Matriz produto matC:")
      para i  de 1 ate 2 faca // linhas de matC
         escreval("")
         escreva("linha =",i,"=>")
         para j de 1 ate  2 faca // colunas de matC
             escreva(matC[i,j])
         fimpara
      fimpara
fimalgoritmo
```

```
Carregando a matriz A:
matA [ 1, 1]=2
matA [ 1, 2]=0
matA [ 2, 1]=-1
matA [ 2, 2]=5
Carregando a matriz B:
matB[ 1, 1]=1
matB[ 1, 2]=4
matB[ 2, 1]=-2
matB[ 2, 2]=0
Matriz produto matC:

linha = 1=> 3 4
linha = 2=> -3 5
*** Fim da execução.
*** Feche esta janela para retornar ao Visualg.
```

FONTE: dos autores

A subtração entre duas matrizes A e B, ambas de mesma ordem, é obtida a partir da soma da matriz A com a oposta de B, ou seja, A – B = A + (-B). O algoritmo apresentado na Figura 61 realiza subtração A – B e obtém a matriz C = A – B.

FIGURA 61 – Exemplo de Algoritmo para Subtração de Matrizes

```
algoritmo "subtracao_de_matrizes
var
        matA: vetor [1..2,1..2] de inteiro
                matB: vetor [1..2,1..2] de inteiro
                matC: vetor [1..2,1..2] de inteiro
                i, j, k: inteiro
        inicio
//Entrada de dados matA é igual ao algoritmo da soma

//Entrada de dados matB é igual ao algoritmo da soma

//Cálculo da matriz subtração matC
para i de 1 ate 2 faca //linhas de matA= linhas de matB = linhas de matC
    para j de 1 ate 2 faça // colunas de matA= colunas de matB = colunas de matC
       matC [i,j] <- matA[i,j] - matB[i,j]
    fimpara
fimpara

//Saída é igual ao algoritmo da soma

fimalgoritmo
```

FONTE: dos autores, adaptado por NTE, 2017

A multiplicação de duas matrizes, sendo uma matriz A_{mxn} e uma matriz B_{nxp}, define-se produto da matriz A pela matriz B, como sendo a matriz C_{mxp}, tal que cada elemento de C (cij) satisfaz:

$$C_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \ldots + a_{in}b_{nj}$$

Ao analisarmos a definição acima, concluímos que cada elemento de C é calculado multiplicando-se ordenadamente os elementos da linha *i* da matriz *A*, pelos elementos correspondentes da coluna ***j*** da matriz *B* e, a seguir, somando-se os produtos obtidos, como mostra o exemplo da Figura 62.

FIGURA 62 – Multiplicação da matriz A pela matriz B

$$A=\begin{bmatrix} 2 & 3 & 1 \\ -1 & 4 & 2 \end{bmatrix}_{2x3} \qquad B=\begin{bmatrix} 3 & 4 \\ 1 & 5 \\ 2 & -6 \end{bmatrix}_{3x2}$$

$$AxB=\begin{bmatrix} 2 \times 3 + 3 \times 1 + 1 \times 2 & 2 \times 4 + 3 \times 5 + 1 \times (-6) \\ (-1) \times 3 + 4 \times 1 + 2 \times 2 & (-1) \times 4 + 4 \times 5 + 2 \times (-6) \end{bmatrix}$$

$$AxB=\begin{bmatrix} 11 & 17 \\ 5 & 4 \end{bmatrix}_{2x2}$$

FONTE: NTE, 2017

Vamos analisar, então, o algoritmo que faz a multiplicação da matriz A pela matriz B (Figura 63).

FIGURA 63 – Algoritmo de Multiplicação de Matrizes

```
algoritmo “Multiplicacao_de_matrizes”
var
                    matA: vetor [1..2,1..3] de inteiro
                    matB: vetor [1..3,1..2] de inteiro
                    matC: vetor [1..2,1..2] de inteiro
                    i, j, k: inteiro
          inicio
//Entrada de dados matA
Escreva(“Carregando a matriz A:”)
para i de 1 ate 2 faca //linhas de A
   para j de 1ate 3 faca //colunas de A
      escreva(“matA [”, i, “,”, j, “]=”)
      leia(matA [i,j])
   fimpara
fimpara

//Entrada de dados matB
escreva( “Carregando a matriz matB:”)
para i de 1 ate 3 faca //linhas de matB
   para j de 1 ate 2 faca //colunas de matB
      escreva(“matB [”, i, “,”, j, “]=”)
      leia(matB [i,j])
   fimpara
fimpara

//Cálculo da matriz produto matC
para i de 1 ate 2 faca //linhas de matC = linhas de matA
   para j de 1 ate 2 faca //colunas de matC = colunas de matB
      matC [i,j] <- 0
      para k de 1 ate 3 faca // colunas de matA = linhas de matB
         matC [i,j] <- matC [i,j] + matA[i, k] * matB[k,j]
      fimpara
   fimpara
fimpara

//Saída
escreva(“Matriz produto matC:” )
para i  de 1 ate 2 faca // linhas de C
   escreva(“linha = ”, i)
   para j de 1 ate  2 faca // colunas de matC
     escreva(matC [i,j])
   fimpara
fimpara
          fimalgoritmo
```

FONTE: dos autores, adaptado por NTE, 2017

Agora vejamos como fica o algoritmo apresentado na Figura 63, codificado no VisuAlg (Figura 64).

FIGURA 64– Algortimo para Multiplicação de Matrizes

```
algoritmo "MultiplicacaoDeMatrizes"
// Função : Multiplicar a MatA pela MatB
// Autor :Autores do Livro de Introdução a Algoritmos
// Data : 22/04/2017
// Seção de Declarações
var
   matA: vetor[1..2,1..3] de real
   matB: vetor[1..3,1..2] de real
   matC: vetor[1..2,1..2] de real
   i, j, k: inteiro
inicio
      // Seção de Comandos
      //Entrada de dados para a matA
      escreval("Carregando a matriz A:")
      Para i de 1 ate 2 faca //linhas de A
         Para j de 1 ate 3 faca //colunas de A
            escreva("matA [",i,",",j,"]=")
            leia(matA[i,j])
         FimPara
      FimPara

      //Entrada de dados para a matB
      escreval("Carregando a matriz B:")
      Para i de 1 ate 3 faca //linhas de B
         Para j de 1 ate 2 faca //colunas de B
            escreva("matB[",i,",",j,"]=")
            leia(matB[i,j])
         FimPara
      FimPara

      //Cálculo da matriz produto C
      Para i de 1 ate 2 faca //linhas de C = linhas de A
         Para j de 1 ate 2 faca //colunas de C = colunas de B
            matC[i,j] <- 0
            Para k de 1 ate 3 faca // colunas de A = linhas de B
                matC[i,j] <- matC[i,j] + matA[i,k] * matB[k,j];
            FimPara
         FimPara
      FimPara

      //Impressão dos Resultados
      escreval("Matriz produto C:")
      para i  de 1 ate 2 faca // linhas de C
         escreval("")
         escreva("linha =",i,"=>")
         Para j de 1 ate  2 faca // colunas de C
            escreva(matC[i,j])
         fimpara
      fimpara
fimalgoritmo
```

```
Carregando a matriz A:
matA [ 1, 1]=2
matA [ 1, 2]=3
matA [ 1, 3]=1
matA [ 2, 1]=-1
matA [ 2, 2]=4
matA [ 2, 3]=2
Carregando a matriz B:
matB[ 1, 1]=3
matB[ 1, 2]=4
matB[ 2, 1]=1
matB[ 2, 2]=5
matB[ 3, 1]=2
matB[ 3, 2]=-6
Matriz produto C:

linha = 1=> 11 17
linha = 2=> 5 4
*** Fim da execução.
*** Feche esta janela para retornar ao Visualg.
```

FONTE: dos autores

7

FUNÇÕES E PROCEDIMENTOS

INTRODUÇÃO

Até agora construímos algoritmos em que todo o fluxo de execução estava inserido no início e fim do corpo do algoritmo, ou seja, não era subdividido. Embora não seja obrigatória a divisão (modularização) dos algoritmos essa prática é recomendável, uma vez que traz vários benefícios.

Quando pensamos em construir algoritmos modularizados, utilizando funções ou procedimentos, temos como meta principal a redução da complexidade, ou seja, a decomposição de um problema complexo em subproblemas. Logo, quando decompomos um problema estamos obrigatoriamente dividindo a complexidade, o que leva a uma simplificação da solução.

Além da redução da complexidade, a utilização de funções e/ou procedimentos nos proporciona a possibilidade de resolver pequenos problemas por vez, o que facilita a compreensão e a percepção do problema em sua totalidade.

A principal pergunta que chegamos é como iremos fazer a modularização de um algoritmo? Para modularizar vamos utilizar a técnica de refinamentos sucessivos, ou top-Down. Após a aplicação da técnica de refinamento sucessivo, podemos iniciar a modularização do algoritmo utilizando funções ou procedimentos. Basicamente, a diferença entre eles é que o procedimento não retorna resultado e a função pode retornar. A seguir, iremos estudar como modularizar um algoritmo usando funções e procedimentos, de acordo com a sintaxe do VisuAlg.

SAIBA MAIS: Pesquise mais sobre a técnica de modularização *top down*.

7.1
MODULARIZAÇÃO DO PROBLEMA

Para modularizarmos um algoritmo, vamos aplicar a técnica de refinamentos sucessivos em um problema que trabalhamos anteriormente, cuja finalidade é ler as notas das provas P1 e P2, calcular a média de cada um dos 10 alunos da turma e, ao final, armazenar estas informações em uma matriz. Além disso, temos que imprimir a situação de cada aluno que pode ser "aprovado" ou "reprovado", e a média geral da turma. O algoritmo que resolve esse problema, sem modularização, está apresentado na Figura 65.

FIGURA 65 – Algoritmo sem Modularização

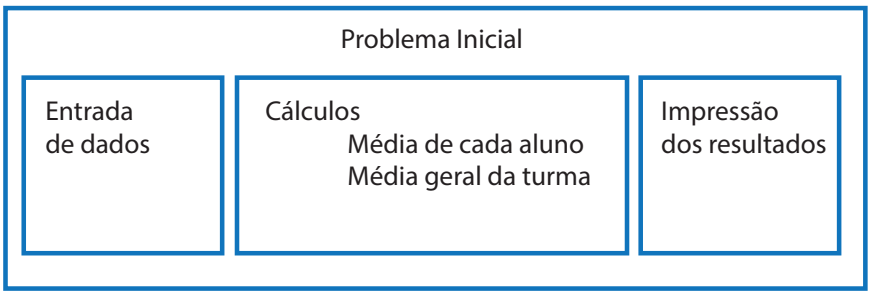
```
algoritmo "Calculo_Situacao_Aluno"
var
                Notas: vetor [1..3,1..10] de real
                contLinha, contColuna: inteiro
                somaP1eP2, mediaGeral: real
        inicio
                //Leitura das notas de P1 ePp2
                para contColuna de 1 ate 10 faca //contador para colunas
                        somaP1eP2 <- 0
                        para contLinha de 1 ate 2 faca //contador para linhas
                                escreva("Digite a nota de p", contLinha ,":")
                                leia(Notas[contLinha, contColuna])
                                somaP1eP2 <-  somaP1eP2+ notas[contLinha, contColuna]
                        fimpara
                        Notas[contLinha, contColuna] <-  somaP1eP2/2
                        mediaGeral <- mediaGeral+ Notas[contLinha, contColuna]
                fimpara

                //Impressão dos Resultados
                para contColuna de 1 ate 10 faca //contador para colunas
                        para contLinha de 1 ate 2 faca //contador para linhas
                                escreva("A nota P",contLinha, "é: ",notas[contLinha, contColuna])
                        fimpara
                        escreva("A média do aluno", contColuna, "é: ",Notas[contLinha, contColuna])
                        se (Notas[contLinha, contColuna] >= 7.0) entao
                                escreva("o aluno está aprovado")
                        senao
                                escreva("O aluno está reprovado")
                        fimse
                fimpara
                escreva("Média Geral da Turma:",mediaGeral/10)
        fimalgoritmo
```

FONTE: dos autores, adaptado por NTE, 2017

O nosso problema, de maneira geral, é classificar a situação do aluno em aprovado ou reprovado. Para isso, a princípio, vamos subdividir o problema em 3 partes: entrada de dados, cálculos e impressão dos resultados. A Figura 66 mostra, graficamente, o resultado da modularização.

FIGURA 66 – Modularização da Solução



FONTE: dos autores, adaptado por NTE, 2017

Agora vamos transformar cada um destes módulos em algoritmo. A começar com a entrada de dados. Este módulo (do tipo procedimento) é composto, basicamente, com o trecho que é responsável pela leitura das notas de *P1* e *P2*. Reparem que retiramos os cálculos das médias, que fazem parte do próximo módulo, como mostra a Figura 67. Os exemplos desta subunidade utilizam os módulos do tipo procedimento (são módulos que não retornam valor à rotina chamadora ao término de sua execução).

FIGURA 67 – Procedimento para Entrada de Dados

```
procedimento EntradaDeDados()
inicio
    contColuna<-0
    contLinha<-0
   escreva("Quantos alunos serão cadastrados: ")
   leia(numMaxAlunos)
          para contColuna de 1 ate numMaxAlunos faca
                para contLinha de 1 ate 2 faca
                      escreva("Digite a nota de p", contLinha ," do aluno",contColuna,": ")
                      leia(notas[contLinha][contColuna])
                fimpara
          fimpara
fimprocedimento
```

FONTE: dos autores, adaptado por NTE, 2017

O procedimento *CalculoDasMedias* apresenta uma particularidade, que é a declaração da variável *somaP1eP2* dentro do corpo do módulo. Isso significa que esta variável pertence ao módulo e é acessível somente dentro dele. A média de cada aluno é armazenada na matriz notas, já a média geral da turma é armazenada na variável *mediaGeral*, como mostra o trecho de algoritmo da Figura 68.

ATENÇÃO: Esta declaração de variáveis dentro do módulo significa que a mesma tem escopo local (ou seja, só pode ser utilizada dentro do módulo). As variáveis declaradas na seção *var*, antes do início do algoritmo, são globais (escopo global) e podem ser utilizadas em qualquer parte do código.

FIGURA 68 – Procedimento para Cálculo das Médias

```
procedimento CalculoDasMedias()
        var
          somaP1eP2: real
inicio
   contColuna<-0
   contLinha<-0
         para contColuna de 1 ate numMaxAlunos faca
                  somaP1eP2<-0
                  para contLinha de 1 ate 2 faca
                           somaP1eP2 <- somaP1eP2+ notas[contLinha,contColuna]
                  fimpara
                Notas[contLinha,contColuna] <-  somaP1eP2/2
                  mediaGeral <- mediaGeral+ notas[contLinha,contColuna]
         fimpara
fimprocedimento
```

FONTE: dos autores, adaptado por NTE, 2017

Para finalizar a modularização, temos o procedimento de impressão dos resultados, conforme apresentado na Figura 69.

FIGURA 69 – Procedimento para Impressão dos Resultados

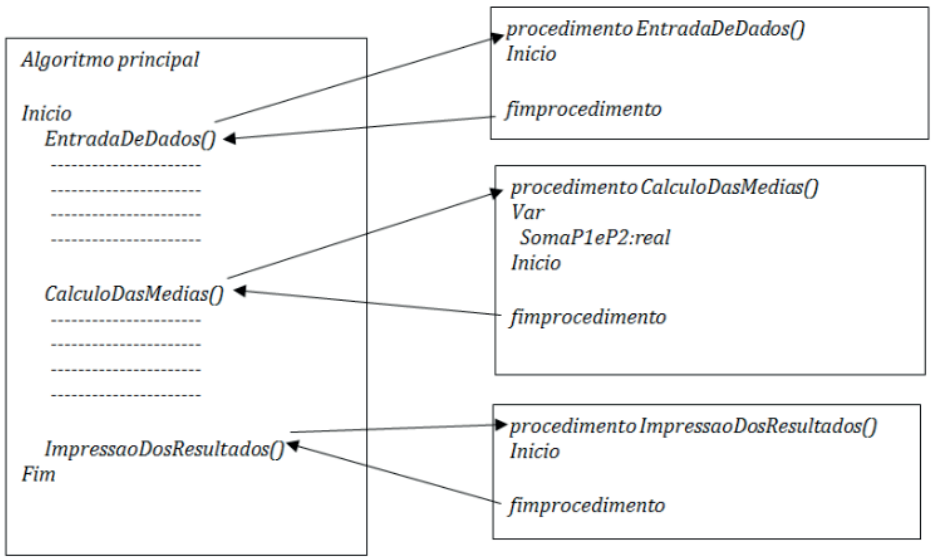
```
procedimento ImpressaoDosResultados()
inicio
   contColuna <- 0
   contLinha <- 0
                  para contColuna de 1 ate numMaxAlunos faca
                           para contLinha de 1 ate 2 faca
                                    escreval("A nota p",contLinha, " é: ",Notas[contLinha,contColuna])
                           fimpara
                           escreval("A média do aluno", contColuna, " é:",Notas[contLinha,contColuna])
                  fimpara
                  escreval("Média Geral da Turma:",mediaGeral/numMaxAlunos)
fimprocedimento
```

FONTE: dos autores, adaptado por NTE, 2017

7.2
FLUXO DAS INFORMAÇÕES

A seguir, vamos entender como se comporta o fluxo do algoritmo durante a execução dos módulos, dentro do algoritmo principal e o relacionamento entre ambas as partes. A Figura 70 representa, graficamente, o fluxo de informações.

FIGURA 70 – Fluxo entre os módulos (procedimentos)



FONTE: dos autores

Analisando a Figura 70, tendo em vista que as setas representam os fluxos e os sentidos, percebemos que quando o fluxo chega a um identificador do procedimento (ou nome do procedimento), o fluxo de execução do algoritmo é desviado para o respectivo procedimento, pois foi encontrado nesse instante uma chamada para o referido procedimento. Assim que terminar a execução, o fluxo retorna ao algoritmo de origem. Essa sequência de passos ocorre em todos os procedimentos.

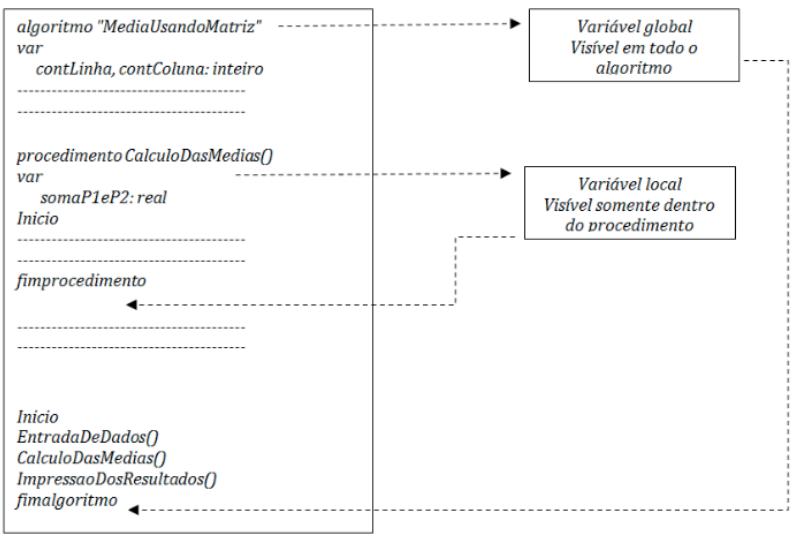
7.3
ESCOPO DAS VARIÁVEIS

Quando falamos em escopo das variáveis, estamos preocupados com a abrangência das variáveis no algoritmo, desde o programa principal até os módulos (procedimentos e/ou funções) que fazem parte da resolução do problema.

Basicamente existem duas maneiras de se declarar uma variável, no início do algoritmo principal ou no início dos módulos. Quando declaramos no início do algoritmo principal ela pode ser acessível em todas as partes, até mesmo dentro dos módulos, desta forma elas são chamadas de variáveis globais.

Quando a variável é declarada dentro do módulo, ela será acessível somente dentro daquele módulo em que foi declarada. O algoritmo principal e os demais módulos não terão acesso a essa variável, por conta desta característica ela é denominada de variável local. A Figura 71 apresenta, de forma gráfica, o escopo das variáveis.

FIGURA 71 - Escopo das variáveis



FONTE: dos autores

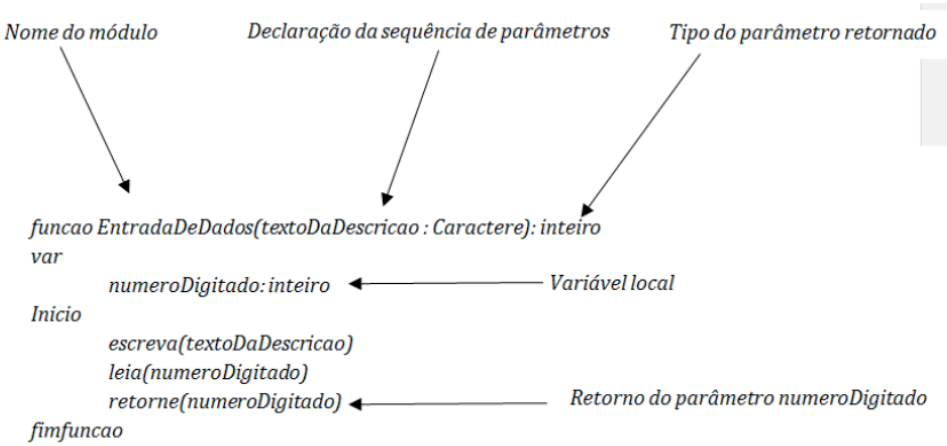
Analisando a Figura 71, temos as variáveis *contLinha* e *contColuna* definidas no início do algoritmo, por isso são definidas como **variáveis globais** e podem ser acessadas por qualquer módulo. Por outro lado, temos a variável *somaP1eP2*, que foi definida dentro do módulo *CalculoDasMedias()*. Sendo assim, ela é acessível somente dentro deste módulo (**variável local**).

Embora não haja, mas caso houvesse uma variável local com o mesmo nome da variável global, iria prevalecer a declaração local, quando o módulo for acionado, e a global quando o fluxo estiver no algoritmo principal.

7.4 PASSAGEM E RETORNO DE PARÂMETROS

Até o momento, quando criamos os módulos, não havia a preocupação de passar e nem retornar informações. Desta forma, sempre foram utilizadas as variáveis globais. Pensando em otimizar ainda mais os nossos algoritmos, vamos aplicar a passagem e retorno de parâmetros em alguns módulos. Um único módulo pode passar e retornar parâmetros, como mostra o exemplo da Figura 72. Os módulos que retornam valores são denominados de funções (*funcao* na sintaxe do VisuAlg).

FIGURA 72 – Sintaxe em um módulo para passar e receber parâmetros.



FONTE: dos autores

Analisando a sintaxe descrita na Figura 72, percebemos que o módulo *EntradaDeDados* recebe como parâmetro a variável *textoDaDescricao*, que contém uma frase com instruções ao usuário – por isso, é do tipo *caractere* e retorna um tipo *inteiro*.

Para exemplificar a passagem e o retorno de parâmetros, vamos criar um algoritmo cuja finalidade é solicitar dois números ao usuário, somá-los e, ao final, apresentar o resultado da soma, como mostra o exemplo da Figura 73.

FIGURA 73 – Exemplo de Algoritmo com Passagem e Retorno de Parâmetros

```
algoritmo “Soma_Dois_Numeros”
var
        numeroA, numeroB: inteiro

funcao EntradaDeDados(textoDaDescricao: Caractere): inteiro
var
   numeroDigitado: inteiro
inicio
   escreva(texto)
   leia(numeroDigitado)
   retorne(numeroDigitado)
fimfuncao

procedimento SomaNumeros(numero1, numero2: inteiro)
var
   resultado: inteiro
inicio
   resultado <- numero1 + numero2
  escreva("A soma é= ",resultado)
fimprocedimento

inicio //início do algoritmo principal
numeroA  <-  EntradaDeDados("Digite o número 1: ")
numeroB <-  EntradaDeDados("Digite o número 2: ")

escreval ("====Resultado======")
SomaNumeros(numeroA, numeroB)
fimalgoritmo
```

FONTE: dos autores, adaptado por NTE, 2017

Ao analisar o corpo do algoritmo apresentado na Figura 73, vemos que são realizadas duas chamadas para o módulo *EntradaDeDados*. Vamos no ater à primeira chamada. Esse módulo recebe um parâmetro que é do tipo caractere, em que foi inserido o texto “*Digite o número 1:* ". O retorno deste módulo é atribuído à variável numérica *numeroA*. Logo abaixo, temos a chamada para outro módulo, denominado de *SomaNumeros*. Neste módulo, não há retorno de parâmetros, somente passagem, por isso são passados a ele os parâmetros *numeroA e numeroB*.

A seguir, vamos classificar os módulos em procedimentos ou funções. A diferença básica é que a função pode retornar valores ao final da sua execução (resultados) e os procedimentos não retornam resultados.

7.5 PROCEDIMENTOS

Levando em conta a definição de um procedimento, que pode ser considerado um módulo que não retorna nenhum tipo de parâmetro (ou resultado), o algoritmo do cálculo da média se encaixa perfeitamente. Sendo assim, ele é apresentado codificado no VisuAlg, graficamente, na Figura 74.

FIGURA 74 – Cálculo da média usando procedimentos no VisuAlg

```
algoritmo "MediaUsandoMatriz"
// Função : Calcular a média das notas p1 e p2 usando matriz e procedimentos
// Autor :Autores do Livro de Introdução a Algoritmos
// Data : 01/05/2017
// Seção de Declarações
var
   notas: vetor [1..3,1..10] de real
   contLinha, contColuna: inteiro
   mediaGeral: real
   numMaxAlunos: inteiro

procedimento EntradaDeDados()
// Seção de declarações de variaveis internas
inicio
// Seção de Comandos
      contColuna<-0
      contLinha<-0
   //Entrada de dados para a matriz das notas de p1 e p2
   escreva("Quantos alunos serao cadastrados: ")
   leia(numMaxAlunos)
   para contColuna de 1 ate numMaxAlunos faca //contador para colunas
      para contLinha de 1 ate 2 faca //contador para linhas
         escreva("Digite a nota de p", contLinha ," do aluno",contColuna,": ")
         leia(notas[contLinha,contColuna])
      fimpara
   fimpara
fimprocedimento


procedimento CalculoDasMedias()
// Seção de declarações de variaveis internas
var
   somaP1eP2: real
inicio
      //Impressão dos Resultados
      contColuna<-0
      contLinha<-0
      para contColuna de 1 ate numMaxAlunos faca //contador para colunas
         somaP1eP2<-0
         para contLinha de 1 ate 2 faca //contador para linhas

              somaP1eP2:= somaP1eP2+ notas[contLinha,contColuna]

         fimpara
         notas[contLinha,contColuna]:= somaP1eP2/2
         mediaGeral:=mediaGeral+ notas[contLinha,contColuna]
      fimpara
fimprocedimento


procedimento ImpressaoDosResultados()
// Seção de declarações de variaveis internas
inicio
      //Impressão dos Resultados
      contColuna<-0
      contLinha<-0
      para contColuna de 1 ate numMaxAlunos faca //contador para colunas
         para contLinha de 1 ate 2 faca //contador para linhas
            escreval("A nota p",contLinha, " é: ",notas[contLinha,contColuna])
         fimpara
         escreval("A media do aluno", contColuna, " é:",notas[contLinha,contColuna])
      fimpara
      escreval("Média Geral da Turma:",mediaGeral/numMaxAlunos)
fimprocedimento


inicio
// Seção de Comandos
      EntradaDeDados()
      CalculoDasMedias()
      ImpressaoDosResultados()
fimalgoritmo
```

```
Quantos alunos serao cadastrados: 3
Digite a nota de p 1 do aluno 1: 5
Digite a nota de p 2 do aluno 1: 5
Digite a nota de p 1 do aluno 2: 9
Digite a nota de p 2 do aluno 2: 9
Digite a nota de p 1 do aluno 3: 10
Digite a nota de p 2 do aluno 3: 10
A nota p 1 é:  5
A nota p 2 é:  5
A media do aluno 1 é: 5
A nota p 1 é:  9
A nota p 2 é:  9
A media do aluno 2 é: 9
A nota p 1 é:  10
A nota p 2 é:  10
A media do aluno 3 é: 10
Média Geral da Turma: 8

*** Fim da execução.
*** Feche esta janela para retornar ao Visualg.
```

FONTE: dos autores

7.6 FUNÇÕES

Para exemplicar o uso de funções, vamos utilizar o algoritmo, exemplificado anteriormente, que solicita dois números ao usuário, faz soma e, ao final, apresenta o resultado. Além de usar funções, que por definição têm a possibilidade de retornar parâmetros, também vamos utilizar um procedimento, para mostrar a diferença entre ambos. O algoritmo em VisuAlg está representado, graficamente, na Figura 75.

FIGURA 75 – Soma usando funções, procedimentos e parâmetros no VisuAlg

```
1  algoritmo "PassagemParametro"
2  // Função : Efetuar a passagem de parâmetros
3  // Autor :Autores do Livro de Introdução a Algoritmos
4  // Data : 01/05/2017
5  // Seção de Declarações
6  var
7     numeroA, numeroB: inteiro
8
9  funcao EntradaDeDados(textoDaDescricao : caracter): inteiro
10 // Seção de declarações de variaveis internas
11 var
12    numeroDigitado: inteiro
13 inicio
14    // Seção de Comandos
15    escreva(textoDaDescricao )
16    leia(numeroDigitado)
17    retorne(numeroDigitado)
18 fimfuncao
19
20
21 procedimento SomaNumeros(numero1, numero2:inteiro)
22 // Seção de declarações de variaveis internas
23 var
24    resultado: inteiro
25 inicio
26       //Impressão dos Resultados
27       resultado<-numero1 + numero2
28       escreval("A soma é=",resultado)
29 fimprocedimento
30
31
32 inicio
33       // Seção de Comandos
34       numeroA <- EntradaDeDados("Digite o número 1: ")
35       numeroB <- EntradaDeDados("Digite o número 2: ")
36
37       escreval("====Resultado======")
38       SomaNumeros(numeroA, numeroB)
39 fimalgoritmo
```

```
Digite o número 1: 2
Digite o número 2: 5
====Resultado======
A soma é= 7

*** Fim da execução.
*** Feche esta janela para retornar ao Visualg.
```

FONTE: dos autores

CONSIDERAÇÕES FINAIS

Este livro apresentou diferentes conceitos importantes para que você aprenda a programar um computador. Iniciamos conceituando o que é um algoritmo, utilizando linguagem natural e, posteriormente, focamos nossos esforços no uso de uma linguagem conhecida como pseudocódigo, por meio do ambiente VisuAlg. Além do VisuAlg, existem outros ambientes, tais como o AMBAP e o *Scratch* que podem ser utilizados para aprender programação (ambientes que você pode utilizar, também, como futuro professor de Informática, quando estiver ensinando seus alunos a programar).

O aprendizado de algoritmos é muito importante, pois esta disciplina (*Introdução a Algoritmos*) é a base para as demais disciplinas da área de programação de computadores do Curso de Licenciatura em Computação. Por meio dos inúmeros exercícios propostos, você pode exercitar e aprender mais como programar, desenvolvendo, também, seu raciocínio lógico.

INTERATIVIDADE: http://sites.google.com/site/ldsicufal/softwares/projeto-ambap

REFERÊNCIAS

APOIO INFORMÁTICA. **VisuAlg**. Disponível em: <http://www.apoioinformatica.inf.br/produtos/visualg>. Acesso em: 12 abr. 2017.

ASCENCIO, A. F. G.; CAMPOS, E. A. V. **Fundamentos da Programação de Computadores**: algoritmos, Pascal e C/C++. São Paulo: Prentice-Hall, 2002.

BERG, A. C.; FIGUEIRÓ, J. P. **Lógica de Programação**. Canoas: ULBRA, 1998.

BERTOLINI, C.; CUNHA, G. B.; FORTES, P. R. **Lógica Matemática**. Santa Maria: UFSM/NTE, 2017.

COWAN, J. **Como ser um Professor Universitário Inovador**: reflexão na ação. Porto Alegre: Artmed, 2002.

EVARISTO, J.; CRESPO, S. **Aprendendo a Programar programando numa linguagem algorítmica executável (ILA)**. Rio de Janeiro: Book Express, 2003.

FALKEMBACH, G. M.; SILVEIRA, S. R. **Algoritmos e Programação I**. ULBRA: Canoas, 2005. Caderno Universitário.

FORBELLONE, A. L. V.; EBERSPÄCHER, H. F. **Lógica de Programação**: a construção de algoritmos e estruturas de dados. São Paulo: Makron Books, 2000.

LOPES, A.; GARCIA, G. **Introdução à Programação.** Rio de Janeiro: Campus, 2002.

MEDINA, M.; FERTIG, C. **Algoritmos e Programação**: teoria e prática. São Paulo: Novatec, 2005.

SEBESTA, R. W. **Conceitos de Linguagens de Programação**. Tradução de Eduardo Kessler Piveta. 9. ed. Porto Alegre: Bookman, 2010.

SILVA, I. C. S.; FALKEMBACH, G. M.; SILVEIRA, S. R. **Algoritmos e Programação em Linguagem C**. Porto Alegre: UniRitter, 2010.

ATIVIDADES DE REFLEXÃO OU FIXAÇÃO

Unidade 1:

Utilizando linguagem natural, escreva os algoritmos (série de passos necessários) para resolver os problemas a seguir:

1) Um homem precisa atravessar um rio com um barco que possui capacidade apenas para carregar ele mesmo e mais uma de suas três cargas, que são: um lobo, um bode e um maço de alfafa. Escreva um algoritmo, em linguagem natural, que permita que a travessia seja realizada com segurança, ou seja, para que o homem preserve as suas cargas (adaptado de FORBELLONE; EBERSPACHER, 2000, p. 13).

2) Quatro rãs estão dispostas em gaiolas, sendo que a primeira gaiola está vazia, como mostra a Figura 7 (adaptado de SILVA; FALKEMBACH; SILVEIRA, 2010):

FIGURA 7 – Disposição Inicial das Rãs

	Rã 1	Rã 2	Rã 3	Rã 4
1	2	3	4	5

FONTE: Adaptado de Silva; Falkembach; Silveira (2010).

Escrever um algoritmo que permita demonstrar todos os passos necessários para que as rãs fiquem na posição demonstrada na Figura 8, sabendo-se que as rãs podem pular de uma gaiola para outra, para a esquerda ou para a direita, se houver gaiola livre e/ou podem pular sobre apenas uma rã para ficarem em uma gaiola livre, para a esquerda ou para a direita:

FIGURA 8 – Disposição Final das Rãs

	Rã 4	Rã 3	Rã 2	Rã 1
1	2	3	4	5

FONTE: Adaptado de Silva; Falkembach; Silveira (2010).

3) Três jesuítas e três canibais precisam atravessar um rio. Para tanto, dispõem de um barco com capacidade para duas pessoas. Por medidas de segurança, não se deve permitir que em nenhuma das margens a quantidade de jesuítas seja inferior à de canibais. Escrever um algoritmo que mostre os passos necessários para que a travessia seja realizada com segurança (adaptado de FORBELLONE; EBERSPACHER, 2000, p. 13).

4) Esboce o fluxograma relativo ao seguinte algoritmo (adaptado de SILVA; FALKEMBACH; SILVEIRA, 2010):

– Leia os valores das variáveis A, B e C
– Calcule a média aritmética desses 3 valores
– Escreva a média calculada

5) Esboce o fluxograma relativo ao seguinte algoritmo (adaptado de SILVA; FALKEMBACH; SILVEIRA, 2010):

– Leia 3 valores, nas variáveis X, Y e Z
– Some os 3 valores e armazene o resultado na variável Soma
– Multiplique o valor da variável Soma por 2 e armazene este resultado na variável Result
– Divida o valor da variável Result por 4 e armazene o resultado na mesma variável Result
– Escreva os valores das variáveis Soma e Result

6) Escreva um algoritmo que, dadas as medidas de um trapézio, calcule e escreva a sua área. Lembre-se que a área de um trapézio é calculada pela fórmula: ((base maior + base menor) x altura) /2 (adaptado de SILVA; FALKEMBACH; SILVEIRA, 2010).

7) Dadas as notas de um trabalho de grupo, de um projeto e de uma prova de um aluno, escrever o algoritmo para calcular a sua média final, sabendo que a mesma é igual à média do trabalho de grupo com o projeto, com peso 3 e a prova com peso 7 (média ponderada). A fórmula para o cálculo desta média ponderada é: (((trabalho + projeto) / 2) x 3 + prova x 7)/10. Ao final do algoritmo, escreva a média calculada. Na multiplicação, como o algoritmo será escrito em linguagem natural, podemos utilizar o símbolo x para indicar a multiplicação. Entretanto, em pseudocódigo ou em uma linguagem de programação, a multiplicação é geralmente representada pelo símbolo * - asterisco (adaptado de SILVA; FALKEMBACH; SILVEIRA, 2010).

8) Escrever um algoritmo, em linguagem natural, que calcule o valor de venda de um carro, sabendo-se que esse custo inclui o custo de compra, 20% de lucro, 45% de impostos e mais 8% destinados à verba de publicidade. Inicialmente, deve-se ler o custo de fábrica do carro e, aplicando-se as alíquotas definidas, calcular e escrever o valor de venda do carro (adaptado de SILVA; FALKEMBACH; SILVEIRA, 2010).

9) Escrever um algoritmo, em linguagem natural, que calcule o número de litros de combustível gastos em uma viagem, sabendo-se que o carro faz 12Km com um litro. Deverão ser lidos o tempo gasto na viagem e a velocidade média. Devem ser aplicadas as fórmulas: Distância = tempo x velocidade e Litros Gastos = Distância / 12 (adaptado de LOPES; GARCIA, 2002).

10) Pedro comprou um saco de ração com peso em quilos. Pedro possui dois gatos para os quais fornece a quantidade de ração em gramas. Escrever um algoritmo, em linguagem natural, que leia o peso do saco de ração e a quantidade de ração fornecida para cada gato e, ao final, calcule e escreva quanto restará no saco de ração quando se passarem 5 dias (adaptado de ASCENCIO; CAMPOS, 2002)

Unidade 2:

1) Supondo que A, B e C são variáveis inteiras com valores iguais a 5, 10 e –8, respectivamente, e uma variável real D com valor igual a 1.5, quais os resultados das expressões aritméticas a seguir?

a) 2 * mod(A,3) – C
b) ((20\3)\3) + (2*8)/2
c) mod(30,4) * 3^2) * -1
d) –C^2 + (D*10)/A
e) A^B/A + C * D

2) Determine os resultados obtidos na avaliação das expressões lógicas seguintes, sabendo que A=2, B=7, C=3.5 e que a variável lógica L=falso:

a) B=A*C e (L ou NAO(L))
b) B>A ou B=A^A
c) L e Inteiro(B/A)>=C ou NAO(A<=C)
d) B/A=C ou B/A <> C
e) L ou B^A <=C*10+A*B

3) Considerando X= 2, Y = 4, Z = 3, LIVRO = "ficção" e MATERIAL = "lápis" e LOG = Verdadeiro, determinar o valor de cada expressão abaixo:

a) **NAO** (LIVRO = "técnico" **E** MATERIAL = "caderno")
b) X*Y < Z+Y **OU** LIVRO = "ficção" **E** MATERIAL <> "lápis"
c) X = 1 **E** LOG
d) Y < Z **OU** LOG **E** LIVRO = "romance"
e) X = 2 * Z – Y **E** MATERIAL **<>** "lápis"

4) Dados os valores de X = 9, Y = 3, Z = 2, W = 5 inteiros, determinar o resultado das seguintes expressões:

a) (X + Y) / Z + W
b) X + Y / Z + W
c) (X + Y) / (Z + W)
d) X + Y / (Z + W)

5) Indique o resultado (V para Verdadeiro e F para Falso) para cada uma das expressões lógicas:

a) (2<5 E 3=2+1)
b) (5<>2*4 E 35>=25+10)
c) (((2<5 E 3=2+1) OU 5=4) E “a” <> “b”)
d) ((42>23 OU 5>=2*1) OU (5<>2*4 E 35>=25+10)))

6) Escreva expressões aritméticas, utilizando as regras de sintaxe da linguagem do VisuAlg, para calcular os seguintes valores:

a) Somar os valores das variáveis A, B e C e armazenar o resultado na variável Soma
b) Calcular o valor reajustado de um produto, sendo que deve ser aplicado um reajuste de 15%
c) Calcular o valor promocional de um produto, sendo que deve ser aplicado um desconto de 5%
d) Calcular o valor do Imposto de Renda a ser pago, com base no Salário Bruto, sendo que a alíquota de imposto deve ser de 25%
e) Calcular o valor do Salário Bruto de um funcionário, que deve ser o valor que ele recebe por hora multiplicado pelo número de horas trabalhadas no mês. Considerar no cálculo, também, o valor recebido pelas horas extras (a remuneração das horas extras deve ser acrescida em 5%)

7) Resolva as expressões abaixo, destacando o resultado final das variáveis que recebem os resultados das atribuições (SOUZA et al., 2005):

a) A <- (18/3/2-1)*5-4-(2+3+5)/2
b) B <- 26/6/2 – mod(127\7,5)
c) C <- mod(7,4) – 8/(3+1)

8) Considerando as seguintes atribuições: R <- 2, S <- 5, T <- -1, X <- 3, Y <- 1 e Z <- 0, resolva as expressões lógicas abaixo indicando o seu resultado (V para verdadeiro e F para falso) (SOUZA et al., 2005):

a) (R>=5) OU (T>Z) E (X-Y + R> 3 * Z)
b) (X=2) OU (Y=1) E ((Z=0) OU (R>3)) E (S<10)
c) (R <> S) OU NÃO(R<X) E (4327 * X * S * Z = 0)

Unidade 3

Nestes exercícios, você deve escrever os algoritmos utilizando as regras de sintaxe do VisuAlg, a partir dos comandos apresentados na Unidade 3.

1) A partir do algoritmo em linguagem natural, escrevê-lo em pseudocódigo:

1.1) a) Ler os valores das variáveis A, B e C
b) Calcular a média aritmética desses valores
c) Imprimir “Média=” e o valor da média calculado
d) Encerrar o algoritmo

1.2) a) Ler 3 valores (utilizando variáveis quaisquer)
b) Somar os 3 valores e armazenar o resultado na variável Soma
c) Multiplicar o valor da variável Soma por 2 e armazenar na variável Resultado
d) Dividir o valor da variável Resultado por 4 e armazenar na mesma variável
e) Imprimir o conteúdo das variáveis Soma e Resultado

1.3) a) Ler a medida do lado de um quadrado
b) Calcular seu perímetro e armazená-lo na variável P
c) Calcular sua área e armazená-la na variável A
d) Calcular sua diagonal e armazená-la na variável D
e) Imprimir a mensagem "Dados do quadrado" e, nas linhas subsequentes, cada valor calculado (P, A e D) com sua respectiva mensagem.

2) Dadas as medidas de um trapézio, escrever um algoritmo que calcule e imprima sua área. Lembrar que a área de um trapézio é calculada através da fórmula: ((base maior + base menor) * altura)/ 2

3) Dadas as notas de um trabalho de grupo, de um projeto e de uma prova de um aluno, fazer um algoritmo para calcular sua média final sabendo que a mesma é igual a média do trabalho de grupo com o projeto, com peso 3 e a prova com peso 7. Imprimir a média final.

4) Escrever um algoritmo que calcule o valor de venda de um tipo de carro popular sabendo-se que esse custo inclui o custo de compra, os 20% de lucro, o percentual de impostos (45%) mais 8% relacionados à publicidade de lançamento. Imprimir o custo final.

5) Escrever um algoritmo que lê o código de identificação de um funcionário, seu número de horas trabalhadas, o valor que recebe por hora e o número de filhos com idade menor do que 14 anos (para cálculo do salário família). Para fins de cálculo, considere que o salário família é de R$15,00 para cada filho com idade menor do que 14 anos. Calcular o salário deste funcionário e escrevê-lo.

6) Escrever um algoritmo que lê um valor qualquer e calcula seu dobro. Escrever este resultado

7) Escrever um algoritmo que lê 3 valores A, B e C e calcula e escreve a média ponderada com pesos 5 para o valor A e para B e peso 2,5 para o valor de C.

8) Escrever um algoritmo que lê 4 valores (A, B, C, D) e calcule a média aritmética, harmônica e geométrica. Escrever as médias no final.

9) Escrever um algoritmo que leia dois valores (A e B) e troque os valores, de forma que a variável A fique com o valor da variável B e vice-versa.

10) Escrever um algoritmo que lê um valor em reais e calcula qual o menor número possível de notas de 100, 50, 10, 5 e 1 em que o valor lido pode ser decomposto. Escrever o valor lido e a relação de notas necessárias.

11) (Adaptado de LOPES; GARCIA, 2002). Escrever um algoritmo que calcule o número de litros de combustível gastos em uma viagem, sabendo-se que o carro faz 12Km com um litro. Deverão ser lidos o tempo gasto na viagem e a velocidade média. Aplicar as seguintes fórmulas:

– Distância = tempo * velocidade
– Litros gastos = Distância / 12

No final, escrever os valores da velocidade média, tempo gasto na viagem, distância percorrida e a quantidade de litros de combustível utilizada.

12) Escrever um algoritmo que leia a idade de uma pessoa expressa em dias e mostre-a expressa em anos, meses e dias (BERG; FIGUEIRÓ, 1998, p. 51).

13) Escrever um algoritmo que leia o tempo necessário para fabricação de um produto expresso em segundos e mostre-o expresso em horas, minutos e segundos (adaptado de BERG; FIGUEIRÓ, 1998).

14) O custo final ao consumidor de um carro novo é a soma do custo de fábrica com a percentagem do distribuidor e dos impostos (aplicados ao custo de fábrica). Supondo que a percentagem do distribuidor seja de 28% e os impostos de 45%, escreva um algoritmo que leia o custo de fábrica de um carro e escreva o custo final ao consumidor (BERG; FIGUEIRÓ, 1998, p. 52).

15) Uma loja vende seus produtos em 3 parcelas: uma entrada mais duas prestações. Escrever um algoritmo que leia o valor da compra e calcule o valor da entrada e das prestações, sabendo-se que as prestações devem ser sempre valores inteiros (adaptado de EVARISTO; CRESPO, 2003). Exemplos:

– Compra 1: R$270,00 Entrada + 2 Prestações de R$90,00
– Compra 2: R$325,50 Entrada R$ 109,50 + 2 Prestações de R$108,00

16) (Adaptado de ASCENCIO; CAMPOS, 2002) Pedro comprou um saco de ração com peso em quilos. Pedro possui dois gatos para os quais fornece a quantidade de ração em gramas. Escrever um algoritmo que leia o peso do saco de ração e a quantidade de ração fornecida para cada gato. Calcular e escrever quanto restará de ração no saco quando se passarem cinco dias.

17) Escreva um algoritmo que leia um número inteiro e escreva seu antecessor e seu sucessor (LOPES; GARCIA, 2002)

18) Escreva um algoritmo que leia dois números (dividendo e divisor). Escreva, ao final: o dividendo, o divisor, o resultado da divisão dividendo/divisor, a parte inteira da divisão e o resto (mod) da divisão (adaptado de LOPES; GARCIA, 2002)

19) Uma pessoa resolveu fazer uma aplicação em uma poupança programada. Para calcular seu rendimento, ela deverá fornecer o valor constante da aplicação mensal, a taxa e o número de meses. Para realizar o cálculo deve ser aplicada a fórmula Valor acumulado = Valor da aplicação mensal * (((1+ taxa)número de meses -1)/i) (adaptado de LOPES; GARCIA, 2002)

Unidade 4:

1) Realize o teste de mesa dos trechos de algoritmos abaixo. O algoritmo deve informar, corretamente, quando o valor estiver no intervalo de [10 a 99]. Os colchetes significam que o intervalo é fechado, ou seja, devem ser inclusos os números 10 e 99. Qual(is) algoritmo(s) efetua(m) esta operação corretamente?

a)
```
se (numero>9) e (numero<100) entao
    escreva( "O número está no intervalo de 10 a 99")
senao
   escreva( "O número não está no intervalo de 10 a 99")
fimse
```

b)
```
se (numero<10) ou (numero>=100) entao
     escreva( "O número não está no intervalo de 10 a 99")
senao
    escreva( "O número está no intervalo de 10 a 99")
fimse
```

c)
```
se NAO(numero<10) ou NAO(numero>99) entao
    escreva( "O número está no intervalo de 10 a 99")
senao
   escreva( "O número não está no intervalo de 10 a 99")
fimse
```

2) Qual(is) das condições abaixo NÃO permite concluir que o valor da variável A é maior do que as variáveis B e C?

```
a) se (A>B) e (A>C) entao
            escreva( "O valor da variável A é maior que o das variáveis B e C")
 fimse
```

b) se (A>B) e (B<C) entao
escreva("O valor da variável A é maior que o das variáveis B e C")
fimse

c) se (B<A) e (C<B) entao
escreva("O valor da variável A é maior que o das variáveis B e C")
fimse

d) se (A>B) ou (A>C) entao
escreva("O valor da variável A é maior que o das variáveis B e C")
fimse

3) Escrever um algoritmo que lê um caracter e testa se é vogal. Se for vogal escrever o caracter lido e a mensagem correspondente, caso contrário escrever "não é vogal".

4) Escrever um algoritmo que lê 4 valores: I, A, B e C. Se I for igual a 1, então calcular a média aritmética de A, B e C e escrever esta média; se I for igual a 2, somar os 3 valores atribuindo este valor a uma variável qualquer e escrevendo esta soma; se I for igual a 3, fazer um teste para saber se B é par, se é par escrever a mensagem e o valor, caso contrário escrever que é ímpar.

5) Dado um número inteiro, fazer o algoritmo para imprimir o dobro desse número se ele for positivo, se ele for negativo imprimir o seu quadrado.

6) Sabendo-se que, se a coordenada X de um ponto for igual a zero e a coordenada Y for diferente de zero, o ponto está sobre o eixo Y, caso contrário está sobre o eixo X. Escrever um algoritmo que imprima uma mensagem indicando onde se encontra o ponto no plano, a partir das coordenadas desse ponto no plano representadas, por X e Y.

7) (Adaptado de ASCENCIO; CAMPOS, 2002) Um supermercado deseja reajustar os preços de seus produtos, para mais ou para menos, de acordo com os critérios demonstrados na tabela abaixo. Escrever um algoritmo que leia o preço atual e a venda mensal média do produto e calcule e escreva o seu preço reajustado.

Venda Média Mensal	Preço Atual	% de Aumento	% de Diminuição
< 500	< R$30,00	10	-
>=500 e < 1200	>=R$30,00 e <R$ 80,00	15	-
>= 1200	<= R$80,00	-	20

8) Dadas as notas de um trabalho de grupo, de um projeto e da prova, de um aluno, fazer o algoritmo para calcular sua média final sabendo que essa é igual à média do trabalho de grupo com o projeto, com peso 3 e a prova com peso 7. Se a média for maior ou igual a 7, imprimir "Aprovado por média". Caso contrário, calcular o que é preciso no exame para ser aprovado e imprimir "Preciso de" e o valor necessário para ser aprovado.

9) Uma empresa de eletrodomésticos gratifica seus vendedores com uma comissão proporcional às vendas efetuadas no mês. A comissão é de 2,5% se o total de vendas for de até R$500.00 e de 4% se as vendas forem maiores que esse valor. Fazer o algoritmo para ler o código do vendedor, seu salário básico, o total de vendas no mês e calcular o salário a ser recebido. Imprimir o código, o salário básico e o salário a ser recebido.

10) Elaborar um algoritmo que, dada a idade de um nadador, classifique-o em uma das seguintes categorias (BERG; FIGUEIRÓ, 1998, p. 57):

Infantil A: 5-7 anos
Infantil B: 8-10 anos
Juvenil A: 11-13 anos
Juvenil B: 14-17 anos
Adulto: Maiores de 18 anos

11) Escrever um algoritmo que leia a idade de um eleitor e verifique se ele pode ou não votar. Se ele puder votar, informe se o seu voto é facultativo. Sabe-se que a partir dos 16 até os 18 anos o voto é facultativo, assim como para os maiores de 70 anos.

12) Escrever um algoritmo que lê três valores e encontre o maior dos valores lidos. Escrever o maior valor ao final.

13) Tendo como dados de entrada a altura e o sexo de uma pessoa ("M" para Masculino e "F" para feminino), construa um algoritmo que calcule e escreva o seu peso ideal, de acordo com as fórmulas: (adaptado de BERG; FIGUEIRÓ, 1998):

– Homens: (72.7 * Altura) – 58
– Mulheres: (62.1 * Altura) – 44.7

14) Escrever um algoritmo que leia o destino de um passageiro (conforme tabela abaixo), se a viagem inclui retorno (ida e volta) e calcule o preço da passagem a ser adquirida (Adaptado de LOPES; GARCIA, 2002):

Destino	Ida	Ida e Volta
Região Norte	R$500,00	R$900,00
Região Nordeste	R$350,00	R$650,00
Região Centro-Oeste	R$350,00	R$600,00
Região Sul	R$300,00	R$550,00

15) Escreva um algoritmo que leia o código de um determinado produto e mostre a sua classificação, utilizando a seguinte tabela como referência (FORBELLONE; EBERSPÄCHER, 2000, p. 47):

Código	Classificação
1	Alimento não-perecível
2,3 ou 4	Alimento perecível
5 ou 6	Vestuário
7	Higiene pessoal
8 até 15	Limpeza e utensílios domésticos

16) Um banco concederá um crédito especial a seus clientes, variável de acordo com o saldo médio no último ano. Escrever um algoritmo que leia o saldo médio de um cliente e calcule o valor do crédito utilizando a tabela abaixo (adaptado de BERG; FIGUEIRÓ, 1998):

Saldo Médio	Percentual de Crédito
Até R$200,00	Nenhum crédito
De R$201,00 a R$400,00	20% do valor do saldo médio
De R$401,00 a R$600,00	30% do valor do saldo médio
Acima de R$600,00	40% do valor do saldo médio

17) Escreva um algoritmo que calcule o que deve ser pago por um produto, considerando o preço normal de etiqueta e a escolha da condição de pagamento. Utilize os códigos da tabela a seguir para ler qual a condição de pagamento escolhida e efetuar o cálculo adequado (FORBELLONE, EBERSPÄCHER, 2000, p. 48):

Código	Condição de Pagamento
1	À vista em dinheiro ou cheque, recebe 10% de desconto
2	À vista no cartão de crédito, recebe 5% de desconto
3	Em 2 vezes, preço normal de etiqueta sem juros
4	Em 3 vezes, preço normal de etiqueta mais juros de 10%

18) (Adaptado de LOPES; GARCIA, 2002). A prefeitura de uma determinada cidade abriu uma linha de crédito para os seus funcionários. O valor máximo da prestação não poderá ultrapassar 30% do salário bruto. Escrever um algoritmo que leia o valor do salário bruto e o valor da prestação e informe se o empréstimo pode ou não ser concedido.

19) (Adaptado de LOPES; GARCIA, 2002). Um comerciante comprou um determinado produto e quer vendê-lo com um lucro de 45% se o valor da compra for menor do que R$20,00; caso contrário, o lucro será de 30%. Escrever um algoritmo que leia o valor de compra de um produto e escreva o valor de venda.

20) (Adaptado de LOPES; GARCIA, 2002). Sabendo-se que somente os municípios com mais de 20.000 eleitores têm segundo turno para prefeito, caso o primeiro colocado não tenha mais do que 50% dos votos, escrever um algoritmo que leia a quantidade de eleitores de um município, o número de votos do candidato mais votado e informe se haverá ou não segundo turno.

21) (Adaptado de ASCENCIO; CAMPOS, 2002). Escrever um algoritmo que leia os seguintes dados:

– código do estado de origem da carga de um caminhão (conforme tabela 1);
– o peso da carga do caminhão em toneladas;
– o código da carga (conforme tabela 2).

Tabela 1

Código do Estado	Imposto
1	35%
2	25%
3	15%
4	5%
5	Isento

Tabela 2

Código da Carga	Preço por Quilo
10 a 20	100
21 a 30	250
31 a 40	340

Com base nestas informações o algoritmo deve calcular e escrever:

– o peso da carga do caminhão convertido em quilos;
– o preço da carga do caminhão;
– o valor do imposto, sabendo-se que o imposto é cobrado sobre o preço da carga do caminhão e que depende do estado de origem;
– o valor total transportado pelo caminhão (carga mais imposto).

22) (Adaptado de ASCENCIO; CAMPOS, 2002) Uma empresa decidiu dar uma gratificação de Natal aos seus funcionários, baseada no número de horas extras e no número de horas que o funcionário faltou ao trabalho. O valor da gratificação é obtido a partir do coeficiente encontrado por meio da fórmula: número de horas extras – (2/3 * número de horas de faltas). Este coeficiente é aplicado segundo a tabela abaixo:

Coeficiente	Gratificação
> 24	R$500,00
[18...24)	R$400,00
[12...18)	R$300,00
[6...12)	R$200,00
< 6	R$100,00

Escrever um algoritmo que leia o número de horas extras e o número de horas de faltas de um funcionário e calcule e escreva o valor da sua gratificação.

Unidade 5

1) Escrever um algoritmo que lê 10 valores, um de cada vez, e conta quantos deles estão em cada um dos intervalos: [0..25],(25..50],(50..75],(75..100]. No final imprima estes resultados com a mensagem adequada. (O colchete significa intervalo fechado – em que os valores estão inclusos e o parênteses, intervalo aberto).

2) Escrever um algoritmo que leia 20 valores numéricos, um de cada vez, e calcule o produto de todos os valores positivos e menores do que um valor K qualquer (que também deve ser lido). Imprimir o produto calculado no final do algoritmo.

3) Dados 30 valores numéricos quaisquer, escrever um algoritmo para determinar o percentual de valores positivos, o número de valores negativos e o número de zeros. Imprimir os valores calculados.

4) Escrever um algoritmo que leia 25 valores inteiros, positivos, determine o maior valor, o menor valor e calcule a média dos números lidos. Imprimir os resultados.

5) Fazer um algoritmo para ler o número de dias do mês vigente, as respectivas temperaturas médias diárias desse mês e imprimir o dia de maior temperatura ocorrida no mês.

6) Dados os valores de M e N, com N>M fazer um algoritmo para determinar a soma de todos os valores entre M e N.

7) Um banco concederá um crédito especial aos seus N clientes preferenciais a partir do saldo médio no último ano. Fazer um algoritmo que leia o nome do cliente, seu saldo médio em unidades e calcule o valor do crédito de acordo com o seguinte critério: se o saldo médio for menor que 200 unidades o percentual será de 20% do valor do saldo médio, se o saldo médio for de 200 a 400 unidades o percentual será de 30% do valor do saldo médio e se o saldo médio for maior que 400 unidades o percentual será de 40% do valor do saldo médio. Imprimir o nome do cliente, seu saldo médio e o valor do crédito.

8) Escrever um algoritmo que imprima todos os valores ímpares contidos entre 1 e 100.

9) Escrever um algoritmo que leia 2 valores quaisquer e escreva a soma dos valores compreendidos entre os 2 números lidos, a média aritmética dos valores compreendidos entre os números lidos e a quantidade de valores compreendidos entre os dois números lidos.

10) Escrever um algoritmo que gera os números de 1000 à 1999 e imprima aqueles que divididos por 11 têm resto igual à 5 (BERG; FIGUEIRÓ, 1998, p. 86).

11) Escrever um algoritmo que lê 10 valores para N, todos inteiros e positivos, e para cada N lido, escrever a tabuada de N*1 até N*N (Adaptado de BERG; FIGUEIRÓ, 1998).

12) Escrever um algoritmo que lê e imprime uma série de 15 números quaisquer. Contar quantos deles são menores que 50. No final imprimir a mensagem "menores que 50" e a quantidade de valores menores.

13) Escrever um algoritmo que lê cinco pares de valores A, B, todos inteiros e positivos, um par de cada vez e A < B. Escrever os inteiros pares de A até B, incluindo o A e o B se forem pares.

14) Escrever um algoritmo que lê 10 números quaisquer e conte quantos números são negativos e quantos positivos, escrevendo a soma no final.

15) Escrever um algoritmo que escreve os números pares entre 100 e 200, bem como a soma desses números.

16) Uma empresa deseja aumentar seus preços em 20%. Fazer um algoritmo que leia o código e o preço de custo de um grupo de 20 produtos e calcule o preço novo. Mostrar o código e o preço novo de cada produto. Se o preço do produto for maior do que R$1.000,00, aplicar um aumento de apenas 5%.

17) Escrever um algoritmo que calcula e escreve a média aritmética dos números pares entre 13 e 73 (Adaptado de BERG; FIGUEIRÓ, 1998).

18) Escrever um algoritmo que calcula e escreve o produto dos números ímpares entre 5 e 40.

19) Escrever um algoritmo que lê 10 valores, um de cada vez, e conta quantos deles estão no intervalo de [10...20] e quantos não estão, escrevendo essas informações.

20) Escrever um algoritmo que leia 15 números e calcule o produto dos números lidos.

21) Escrever um algoritmo que lê 10 números quaisquer e conte quantos números são negativos e quantos são positivos, escrevendo estas informações no final.

22) Escrever um algoritmo que leia um número e escreva na tela todos os seus divisores.

23) Escrever um algoritmo que lê 10 números, conte quantos são positivos, quantos são negativos e acumule os valores positivos e negativos separadamente. Imprimir os contadores e acumuladores no final.

24) Sendo H=1+1/2+1/3+1/4+...+1/N, escreva um algoritmo que gere o número H. O número N é fornecido pelo usuário (Adaptado de BERG; FIGUEIRÓ, 1998).

25) A conversão de graus Fahrenheit para centígrados é obtida pela seguinte fórmula: C=5/9(F-32). Escreva um algoritmo que calcule e escreva uma tabela de graus centígrados em função de graus Fahrenheit que variem de 50 a 150 de 1 em 1.

26) Escrever um algoritmo para encontrar todas as soluções inteiras para a seguinte equação: 3x + 2y + 7z = 5. Os valores encontrados para x, y e z devem variar entre 0 e 100.

27) Uma rainha requisitou os serviços de um monge e informou que pagaria qualquer preço. O monge, necessitando de alimentos, solicitou que o pagamento fosse efetuado em grãos de trigo, dispostos num tabuleiro de xadrez, de tal forma que o primeiro quadro contivesse apenas 1 grão e, cada quadro subsequente, o dobro de grãos do quadro anterior. Escrever um algoritmo que calcule o número de grãos que o monge esperava receber (adaptado de FORBELLONE; EBERSPÄCHER, 2000).

28) Escrever um algoritmo que imprima todas as possibilidades de que, no lançamento de dois dados, a soma dos valores de cada dado seja igual a 7 (adaptado de FORBELLONE; EBERSPÄCHER, 2000).

29) (Adaptado de LOPES; GARCIA, 2002). Em um campeonato de vôlei inscreveram-se 30 países. Sabendo-se que na lista oficial de cada país consta, além de outros dados, peso e idade de 12 jogadores, escrever um algoritmo que calcule e escreva:

– o peso médio de cada um dos times;
– a idade média de cada um dos times;
– o peso médio de todos os participantes (geral);
– a idade média de todos os participantes (geral).

30) Escrever um algoritmo que realize uma pesquisa com 30 alunos de um curso superior de Informática. Os alunos responderam as seguintes informações:

– idade;
– número de semestres já cursados;
– tipo de curso (1 – Bacharelado, 2 – Tecnológico);
– conceito atribuído ao curso (1 – Ótimo, 2 – Muito Bom, 3 – Bom e 4 – Regular).

Ao final da pesquisa, escrever:

– a média de idade dos pesquisados;
– o número de alunos que já cursou no mínimo 4 semestres;
– o número de alunos de Bacharelado e Tecnológico, separadamente;
– o número de alunos que atribuiu os conceitos Ótimo, Muito Bom, Bom e Regular, separadamente.

Obs.: Na entrada do tipo de curso (1 ou 2) e do conceito (1, 2, 3 ou 4), deve ser realizada a validação dos dados (Dica: utilize o comando *enquanto...faca*).

31) (Adaptado de ASCENCIO; CAMPOS, 2002). Uma companhia de teatro quer iniciar uma turnê de espetáculos. A direção calcula que, a R$5,00 o ingresso, serão vendidos 120 ingressos e que as despesas serão de R$200,00. Diminuindo-se R$0,50 no preço dos ingressos espera-se que as vendas aumentem em 26 ingressos. Escrever um algoritmo que mostre uma tabela de valores de lucros esperados em função do preço do ingresso, variando este preço de R$5,00 a R$1,00, de R$0,50 em R$0,50. Escrever, também, o lucro máximo esperado, o preço do ingresso e a quantidade de ingressos vendidos para a obtenção desse lucro.

Algoritmos com Repetição Indeterminada

32) Realizar o teste de mesa do algoritmo abaixo. Ao final do teste de mesa, escrever o valor final das variáveis A, B e C:

```
algoritmo "Teste_de_Mesa"
var
  A, B, C, Contador: inteiro
inicio
        A <- 1
        B <-2
        C <- 3
        para Contador de 1 ate 3 faca
                enquanto (A <> 2) faca
                        B <- B+2
                        C <- C+B+A
                        A <- A+1
                fimenquanto
                A <- 1
        fimpara
        se (B>5) entao
          B <- 0
        senao
          C <- 0
        fimse
fimalgoritmo
```

33) Escrever um algoritmo que lê um número não determinado de valores para M, todos inteiros e positivos, um de cada vez. Verificar se o valor M é par ou ímpar, escrevendo a mensagem correspondente. Contar o número de valores ímpares lidos e calcular a soma dos valores pares. No final, escrever o contador e a soma calculados.

34) (Adaptado de LOPES; GARCIA, 2002) Uma empresa de fornecimento de energia elétrica faz a leitura mensal dos medidores de consumo. Para cada consumidor são lidos os seguintes dados:

– código de cliente do consumidor;
– quantidade de Kwh consumidos durante o mês;
– tipo (código) do consumidor, de acordo com a tabela abaixo:

Código do Tipo de Consumidor	Valor por Kwh
1 – Residencial	R$0,30
2 – Comercial	R$0,50
3 - Industrial	R$0,70

Para cada consumidor deve ser escrito o custo da sua conta mensal de energia.

Os dados devem ser lidos até que seja digitado o código de cliente igual a zero. No final escrever:

– o total de consumo para cada um dos tipos de consumidor, separadamente;
– a média de consumo dos tipos 1 e 2;
– o total (em reais) que a empresa irá arrecadar no mês em questão.

35) Escrever um algoritmo que lê um número não determinado de valores positivos e, no final, imprima a média aritmética desses valores.

36) Escrever um algoritmo que lê um número não determinado de valores e que conta e acumula esses valores; quando o valor lido for menor que zero, escrever o acumulador e o contador.

37) Para cada aluno de um determinado curso ler o número de sua matrícula e sua nota. Imprimir o número do aluno e seu conceito, de acordo com a tabela abaixo:

[0..5): fraco
[5..7): regular
[7..9): bom
[9..10]: muito bom

O algoritmo deve terminar quando for informado um número de matrícula negativo para o aluno.

38) Escrever um algoritmo que calcule a soma de dois números enquanto esta soma for positiva.

39) Escrever um algoritmo que leia números fornecidos pelo usuário e calcule a raiz quadrada do número lido. Isto deve ser repetido até que o usuário entre com um número negativo.

40) (Adaptado de LOPES; GARCIA, 2002). Uma pousada estipulou sua diária em R$30,00, acrescida de uma taxa de serviços diários que pode ser: 1) R$15,00 se o número de dias da estadia for menor que 10; 2) R$8,00 se o número de dias de estadia for maior ou igual a 10. Escrever um algoritmo que leia o nome do cliente e o número de dias da estadia e escreva o valor total da conta a ser paga para cada cliente. O algoritmo deve ser repetido até que o número de dias de estadia seja igual a zero.

41) Escrever um algoritmo que calcule a soma dos números fornecidos pelo usuário. O algoritmo deve encerrar quando o usuário entrar com o valor zero.

42) Em uma eleição presidencial, existem quatro candidatos. Os votos são informados através de código. Os dados utilizados para a escrutinagem obedecem à seguinte codificação (FORBELLONE; EBERSPÄCHER, 2000, p. 62):

1,2,3,4 = voto para os respectivos candidatos
5 = voto nulo
6 = voto em branco

Elabore um algoritmo que calcule e escreva:

– total de votos para cada candidato;
– total de votos nulos;
– total de votos em branco;
– percentual dos votos em branco e nulos sobre o total.

Como finalizador do conjunto de votos, tem-se o valor 0 (zero).

43) Em um prédio há três elevadores denominados A, B e C. Para otimizar o sistema de controle dos elevadores, foi realizado um levantamento no qual cada usuário respondia (FORBELLONE; EBERSPÄCHER, 2000, p. 65):

– o elevador que utilizava com mais frequência;
– o período que utilizava o elevador: M – matutino, V – vespertino, N – noturno;

Construa um algoritmo que calcule e imprima:

– qual é o elevador mais frequentado e em que período se concentra o maior fluxo;
– qual o período mais usado de todos e a que elevador pertence;
– qual a diferença percentual entre o mais usado dos horários e o menos usado;
– qual a percentagem sobre o total de serviços prestados do elevador de média utilização.

44) Anacleto tem 1,50 metros de altura e cresce 2 centímetros por ano, enquanto Felisberto tem 1,10 metros e cresce 3 centímetros por ano. Construa um algoritmo que calcule e imprima quantos anos serão necessários para que Felisberto seja maior que Anacleto.

45) Foi feita uma pesquisa entre os habitantes de uma região. Foram coletados os dados de idade, sexo (M/F) e salário. Fazer um algoritmo que informe: a) a média de salário do grupo; b) maior e menor idade do grupo; c) quantidade de mulheres com salário até R$100,00. Encerrar a entrada de dados quando for digitada uma idade negativa (BERG; FIGUEIRÓ, 1998, p. 85).

Unidade 6

1) Escrever um algoritmo que lê um vetor qualquer de 10 posições e imprime todos os valores iguais à 10, e suas respectivas posições neste vetor.

2) Escrever um algoritmo que lê um vetor V, de 15 posições, e conte quantos valores de V são negativos.

3) Escrever um algoritmo que lê um vetor X de 20 posições. Substitua a seguir todos os valores nulos (iguais a zero) de X por 1. No final, imprima o vetor X modificado.

4) Escrever um algoritmo que lê um vetor C de 50 posições. Encontrar o maior e o menor elemento do vetor C e imprimi-los, juntamente com suas posições no vetor.

5) Escrever um algoritmo que lê um vetor N de 20 posições. Trocar o 10 elemento com o último, o 20 com o penúltimo, o 3º com o antepenúltimo e assim sucessivamente, até trocar o 100 elemento com o 110. Escrever o vetor N modificado. (Observação: as trocas devem ser realizadas no próprio vetor, não utilizar vetores auxiliares)

6) Escrever um algoritmo que leia 10 nomes de clientes e 10 saldos bancários e imprima os nomes dos clientes que tiverem saldo inferior a R$100,00. São necessários dois vetores.

7) Escrever um algoritmo que leia um vetor S(25) e compacte este vetor, retirando dele todos os valores repetidos. Imprimir o vetor antes de ser compactado e após a compactação.

8) Escrever um algoritmo que leia nome e nota de um aluno para uma turma de 20 alunos. Ao final, imprima a média das notas da turma, o nome do aluno que obteve a nota mais alta e o nome do aluno que obteve a nota mais baixa. São necessários dois vetores para realizar este exercício.

9) Escrever um algoritmo que leia 100 números inteiros, distribua os números lidos em dois vetores, sendo um para números pares e outro para números ímpares. No final do algoritmo, escreva os vetores.

10) (Adaptado de LOPES; GARCIA, 2002) Escrever um algoritmo que leia dois vetores de 25 posições cada um. A seguir, criar um terceiro vetor, intercalando os dados dos dois vetores. Este terceiro vetor deve ser impresso no final. Exemplo:

Vetor 1

7	6	1	8	9	3	0	4	2	8		
1	2	3	4	5	6	7	8	9	10	.....	25

Vetor 2

3	4	5	2	8	9	1	2	5	6		
1	2	3	4	5	6	7	8	9	10	.....	25

Vetor 3

7	3	6	4	1	5	8	2	9	8		
1	2	3	4	5	6	7	8	9	10	.....	50

11) Escrever um algoritmo que leia 2 vetores X(10) e Y(10) e os escreva. Criar, a seguir:

– um vetor contendo a união de X com Y (todos os elementos de X e os elementos de Y que não estejam em X);
– um vetor contendo a diferença entre X e Y (todos os elementos de X que não existam em Y);
– um vetor contendo o produto entre X e Y (multiplicação de cada elemento de X com o elemento de mesma posição em Y);
– um vetor contendo a intersecção entre X e Y (valores que aparecem nos dois vetores).

12) Escrever um algoritmo que leia um vetor com 20 valores. Após a leitura calcular e escrever a soma de todos os valores do vetor.

13) Escrever um algoritmo que permita o gerenciamento de reservas numa casa de espetáculos. Existem 30 mesas, cada uma com 5 lugares disponíveis. O algoritmo deve permitir que o usuário escolha a mesa desejada (de 1 a 30) e indique a quantidade de lugares necessários. O algoritmo deve informar se a reserva foi ou não realizada. O algoritmo deve ser encerrado quando o usuário digitar o número da mesa igual a zero (Adaptado de LOPES; GARCIA, 2002).

14) Em um concurso público foram aprovados 50 candidatos. Escrever um algoritmo que armazene os nomes dos candidatos aprovados em um vetor e, num segundo vetor, a nota obtida no concurso. Ordenar os vetores de acordo com a nota obtida (em ordem decrescente). Escrever os dois vetores no final, apresentando os nomes dos candidatos na ordem de chamada para ocuparem as vagas disponibilizadas no concurso.

15) Escrever um algoritmo que realize a manutenção de um controle de estoque de uma loja que possui 10 tipos de produtos em estoque. O controle de estoque utilizará três vetores: 1 vetor para armazenar os códigos dos produtos, 1 vetor para armazenar a descrição (nome) dos produtos e 1 vetor para armazenar a quantidade atual de cada produto. O algoritmo deve apresentar ao usuário as seguintes opções:

1 – Entrar com os dados do estoque: permitir a leitura do código, descrição e quantidade atual dos produtos;
2 – Entrada de produtos no estoque: solicitar o código do produto e verificar se o mesmo está cadastrado. Em caso afirmativo, solicitar a quantidade de entrada e adicioná-la ao estoque;
3 – Saída de produtos do estoque: solicitar o código do produto e verificar se o mesmo está cadastrado. Em caso afirmativo, solicitar a quantidade de saída e verificar se existe estoque suficiente. Existindo estoque, realizar a subtração da quantidade de saída, caso contrário, informar ao usuário;
4 – Relatório: mostrar o código, descrição e quantidade atual de cada produto armazenado no estoque.

16) Escreva um algoritmo que leia duas matrizes reais de dimensão 3 x 5, calcule e imprima a soma das matrizes.

17) Escreva um algoritmo que leia duas matrizes reais de dimensão 3 x 3, calcule e imprima a subtração das matrizes.

18) Escreva um algoritmo que leia um conjunto de 3 valores inteiros do usuário de tamanho variado e armazene cada um dos conjuntos em uma linha da matriz. Deverá ser impresso ao final:
a) A dimensão da matriz que será o tamanho do maior conjunto de valores x 3.
b) A matriz, para os conjuntos menores do que o maior conjunto, o restante deverá ser preenchido por zeros.

19) Escreva um algoritmo que leia uma matriz quadrada de dimensão 4 e encontre e imprima:
a) A matriz lida
b) O menor valor
c) O maior valor

20) Escreva um algoritmo que leia duas matrizes quadradas de dimensões 2 e faça a multiplicação delas. Imprima as matrizes lidas e a resultante.

21) Escreva um algoritmo que leia uma matriz de dimensões 4 x 4, calcule e imprima as somas:
a) De cada linha
b) De cada coluna
c) Da diagonal principal
d) De todos os elementos da matriz
e) Os quatro maiores elementos
f) Os quatro menores elementos
g) Os números primos

22) Escreva um algoritmo que corresponda ao jogo da velha. O Algorimo deverá ler apenas valores o e X, um de cada vez, onde cada um dos valores corresponde a um dos jogadores. A matriz deverá ter a dimensão de 3 x 3. Cada jogador deverá entrar com a posição a ser marcada (linha e coluna). O primeiro jogador que completar uma linha ou uma coluna ou uma diagonal vence. Use a criatividade para imprimir os valores e tornar o jogo divertido.

Unidade 7

1) Escreva um algoritmo que contenha uma função para cada um das operações aritméticas: soma, subtração, multiplicação e divisão. O algoritmo deverá aceitar apenas dois valores e a operação desejada.

2) Escreva um algoritmo que contenha uma função para calcular o IMC (Índice de Massa Corporal) de uma pessoa. Você deverá pesquisar como calcular o IMC e como o resultado deverá ser apresentado ao usuário.

3) Escreva um algoritmo que converta um número decimal para um número binário, ou seja, o algoritmo deverá ter uma função onde recebe um valor em decimal e retorna o valor convertido em binário.

4) Reescreva o algoritmo do Jogo da Velha desenvolvido nos exercícios anteriores. Você deverá criar várias funções para cada um dos procedimentos e regras do jogo.

5) Escreva uma função que receba um valor decimal e retorne um valor inteiro que corresponde ao número arrendondado. Para isso, utilize a seguinte regra: caso o número após a vírgula for maior que 5 arredonde para mais, caso contrário arredonde para menos.

6) Escreva um algoritmo que leia dois conjuntos de valores: Nomes e Matrículas de alunos. Cada Nome é associado a um número inteiro entre os valores 1 e 100 de Matrícula. Imprima a lista de Nomes e Matrículas ordenadas:
a) Por Nome (escreva uma função que receba o conjunto de matrículas e nomes e retorne-os de forma ordenada)
b) Por Número de Matrícula (escreva uma função que receba o conjunto de matrículas e nomes e retorne-os de forma ordenada).

APRESENTAÇÃO DOS AUTORES

Os autores deste livro são docentes do Departamento de Tecnologia da Informação da UFSM (Universidade Federal de Santa Maria)/Campus Frederico Westphalen.

Fábio Parreira: Possui graduação em Ciência da Computação pelo UNITRI (Centro Universitário do Triângulo), Especialista em Produção de Material Didático para EaD pela UFAM (Universidade Federal do Amazonas), Mestrado em Processamento Digital de Imagens pela UFU (Universidade Federal de Uberlândia) e Doutorado em Inteligência Artificial e Informática de Sinais Biomédicos pela UFU (Universidade Federal de Uberlândia). Atualmente, é Professor Associado do Departamento de Tecnologia da Informação no Campus de Frederico Westphalen - RS da UFSM (Universidade Federal de Santa Maria). Suas áreas de interesse envolvem, principalmente: Educação a Distância, Inteligência Artificial e Jogos Educacionais Digitais.

Sidnei Renato Silveira: Bacharel em Informática pela ULBRA (Universidade Luterana do Brasil), Especialista em Administração e Planejamento para Docentes pela ULBRA, Especialista em Gestão Educacional pelo SENAC, Mestre e Doutor em Ciência da Computação pela UFRGS (Universidade Federal do Rio Grande do Sul). Atualmente, é Professor Adjunto do Departamento de Tecnologia da Informação no campus de Frederico Westphalen - RS da UFSM (Universidade Federal de Santa Maria). Suas áreas de interesse envolvem, principalmente: Educação a Distância, Inteligência Artificial, Jogos Educacionais Digitais, Formação Docente e Educação em Informática.

Cristiano Bertolini: Bacharel em Ciência da Computação pela UPF (Universidade de Passo Fundo), Mestre em Ciência da Computação pela PUCRS (Pontifícia Universidade Católica do Rio Grande do Sul), Doutor em Ciência da Computação pela UFPE (Universidade Federal de Pernambuco). Pós-Doutorado pela United Nations University (Macau) na área de Métodos Formais e Informática Aplicada à Saúde. Suas áreas de interesse envolvem, principalmente, Métodos Formais, Teste de Software e Engenharia de Software. Atualmente, é Professor Adjunto do Departamento de Tecnologia da Informação da UFSM (Universidade Federal de Santa Maria)/campus Frederico Westphalen – RS.

Rosane Beatriz Oliveira Severo: Bacharel em Administração com ênfase em Análise de Sistemas pela PUCRS (Pontifícia Universidade Católica do Rio Grande do Sul), Especialista em Análise de Sistemas pela PUCRS, Mestre em Ciência da Computação pela UFRGS (Universidade Federal do Rio Grande do Sul) e Doutora em Engenharia Elétrica pela USP (Universidade de São Paulo). Atualmente, é Professora Adjunta do Departamento de Tecnologia da Informação no campus de Frederico Westphalen – RS da UFSM (Universidade Federal de Santa Maria). Suas áreas de interesse envolvem Engenharia Elétrica, Redes de Computadores, Sistemas Operacionais e Sistemas Distribuídos.