

PROF. RAFAEL VARGAS MESQUITA

PROGRAMAÇÃO I – 1º SEMESTRE

CACHOEIRO DE ITAPEMIRIM  
2019

**Governo Federal**

**Curso Técnico em Informática Integrado ao Ensino Médio**

**Coordenação de Curso**

Eros Estevão de Moura

**Professor /Autor**

Rafael Vargas Mesquita

**DIREITOS RESERVADOS**

**Ifes – Instituto Federal do Espírito Santo**

**Diretor Geral**

Jadir José Pela

**Créditos de autoria da editoração**

**Capa:** Juliana Cristina da Silva

**Projeto gráfico:** Juliana Cristina e Nelson Torres

**Iconografia:** Nelson Torres

**Editoração eletrônica:** Rafael Vargas Mesquita

**Revisão de texto:** Zenas Vieira Romano

**COPYRIGHT – É proibida a reprodução, mesmo que parcial, por qualquer meio, sem autorização escrita dos autores e do detentor dos direitos autorais.**

*Olá, Aluno(a)!*

*É um prazer tê-lo conosco.*

*Desejamos a você sucesso e dedicação!*

*Equipe do Ifes*

## ICONOGRAFIA

Veja, abaixo, alguns símbolos utilizados neste material para guiá-lo em seus estudos.



Fala do professor.



Conceitos importantes. Fique atento!



Atividades que devem ser elaboradas por você, após a leitura dos textos.



Indicação de leituras complementares, referentes ao conteúdo estudado.



Destaque de algo importante, referente ao conteúdo apresentado. Atenção!



Reflexão/questionamento sobre algo importante, referente ao conteúdo apresentado.



Espaço reservado para as anotações que você julgar necessárias.

# Sumário

<b>CAPÍTULO 1 - INTRODUÇÃO .....</b>	<b>9</b>
1.1. SEQUÊNCIA LÓGICA.....	9
1.2. INSTRUÇÕES .....	9
1.3. ALGORITMO .....	10
1.4. LINGUAGEM DE COMPUTADOR .....	12
1.5. LINGUAGEM DE PROGRAMAÇÃO .....	13
1.6. PROGRAMA .....	13
<b>CAPÍTULO 2 - DESENVOLVENDO ALGORITMOS.....</b>	<b>15</b>
2.1. PSEUDO CÓDIGO .....	15
2.2. DIRETRIZES PARA CONSTRUÇÃO DE ALGORITMOS .....	15
2.3. PARTES .....	16
2.4. EXEMPLO DE ALGORITMO .....	17
<b>CAPÍTULO 3 - DIAGRAMA DE BLOCO .....</b>	<b>19</b>
3.1. O QUE É UM DIAGRAMA DE BLOCO? .....	19
<b>CAPÍTULO 4 - PORTUGOL - ITENS FUNDAMENTAIS .....</b>	<b>23</b>
4.1. CONSTANTES .....	23
4.1.1. Constantes Numéricas .....	24
4.1.2. Constantes Lógicas .....	24
4.1.3. Constantes Literais (Cadeia).....	24
4.2. VARIÁVEIS.....	25
4.2.1. Declaração de variáveis.....	26
4.3. EXPRESSÕES ARITMÉTICAS .....	27
4.3.1. Prioridade das Operações.....	27
4.4. EXPRESSÕES RELACIONAIS .....	28
4.5. EXPRESSÕES LÓGICAS .....	30
<b>CAPÍTULO 5 - PORTUGOL – ITENS COMPLEMENTARES .....</b>	<b>34</b>
5.1. COMENTÁRIOS .....	34
5.2. FUNÇÕES MATEMÁTICAS .....	35
5.3. COMANDOS DE ENTRADA E SAÍDA .....	35
5.4. ATRIBUIÇÃO .....	36
<b>CAPÍTULO 6 - COMANDOS DE DECISÃO .....</b>	<b>43</b>
6.1. SE .....	44
6.2. SE ... SENAO .....	45
6.3. ESCOLHA ... CASO .....	47
<b>CAPÍTULO 7 - COMANDOS DE REPETIÇÃO .....</b>	<b>52</b>
7.1. ENQUANTO .....	52
7.2. FAÇA ... ENQUANTO.....	54
7.3. PARA .....	55
<b>CAPÍTULO 8 - ESTRUTURA DE DADOS HOMOGÊNEAS - VETOR.....</b>	<b>57</b>
8.1. ESTRUTURAS DE DADOS .....	57
8.2. VARIÁVEIS COMPOSTAS HOMOGÊNEAS .....	61
8.3. VARIÁVEIS COMPOSTAS UNIDIMENSIONAIS – VETORES .....	63
8.3.1. Declaração de Vetores .....	63
<b>CAPÍTULO 9 - MODULARIZAÇÃO.....</b>	<b>66</b>
9.1. INTRODUÇÃO.....	66

9.2. PROCEDIMENTOS E FUNÇÕES .....67

9.2.1. Funções.....67

9.2.2. Passagem de Parâmetros por Valor e Passagem de Parâmetros por Referência .....68

9.2.3. Procedimentos .....69

9.2.4. Uso de Funções e Procedimentos .....69



Olá!

Meu nome é Rafael Vargas Mesquita, responsável pela disciplina de Programação I. Atuo como professor do Ifes há cinco anos. Sou graduado em Ciência da Computação (2001) e Mestre em Gestão da Qualidade de Software (2007), ambos pela UFLA, e Doutor pela UENF – Universidade Estadual Norte Fluminense. Minhas áreas de interesse são: Qualidade de Software, Melhoria de Processo de Software, Medição de Software, Análise e Projeto de Sistemas Orientados a Objetos.

Nesta disciplina, você entenderá alguns conceitos importantes relacionados a Algoritmos.

Os capítulos 1, 2 e 3 visam apresentar conceitos iniciais a respeito da disciplina. Nos capítulos 4 e 5, você começará a desenvolver algoritmos simples em uma linguagem alto nível. Os últimos capítulos, 6, 7, 8 e 9, são capítulos nos quais vamos elaborar algoritmos mais complexos, com a finalidade de nos aprofundarmos na lógica de programação.

O objetivo deste material é auxiliá-lo no estudo da disciplina de Algoritmos, por meio de dicas e sugestões que destacam os pontos mais importantes a serem estudados. Aqui você encontrará conceitos com os quais trabalharemos ao longo de todo o Curso.

É importante esclarecer que, além deste material, você poderá encontrar outros materiais, listas de exercícios, etc no Moodle.

Para acessar o FTP acesse: <https://ava.cefor.ifes.edu.br/>

Bons estudos, e sucesso!

Prof. Rafael Vargas Mesquita







## Capítulo 1 - INTRODUÇÃO



Prezado aluno,

Começaremos o primeiro capítulo com uma introdução sobre Algoritmos. Para entendermos essa matéria precisaremos, primeiramente, definir alguns conceitos fundamentais como: Linguagem de Computador, Linguagem de Programação, Programa, etc. Em geral, esta disciplina é acumulativa, ou seja, a compreensão dos conceitos estudados em um capítulo é a base para o entendimento dos capítulos posteriores.

Bom estudo!

A lógica de programação é necessária para pessoas que desejam trabalhar com desenvolvimento de sistemas e programas. Ela permite definir a sequência lógica para o desenvolvimento.

Então, o que é lógica?



**Lógica de Programação** é a técnica de encadear pensamentos para atingir determinado objetivo.

### 1.1. SEQUÊNCIA LÓGICA

Os pensamentos podem ser descritos como uma sequência de instruções que devem ser seguidas para se cumprir uma determinada tarefa.



**Sequência Lógica** são passos executados até atingir um objetivo ou solução de um problema.

### 1.2. INSTRUÇÕES

Na linguagem comum, entende-se por *instruções* um conjunto de regras ou normas definidas para a realização ou emprego de algo.



Em Informática, porém, instrução é a informação que indica a um computador uma ação elementar a ser executada.

Convém ressaltar que uma ordem isolada não permite realizar o processo completo. Para isso, é necessário um conjunto de instruções colocadas em ordem sequencial lógica.

Por exemplo, se quisermos fazer uma omelete de batatas, precisaremos colocar em prática uma série de instruções: descascar as batatas, bater os ovos, fritar as batatas etc.

É evidente que essas instruções têm que ser executadas em uma ordem adequada – não se pode descascar as batatas depois de fritá-las.

Dessa maneira, uma instrução tomada em separado não tem muito sentido; para obtermos o resultado, precisamos colocar em prática o conjunto de todas as instruções, na ordem correta.



**Instruções** são um conjunto de regras ou normas definidas para a realização ou emprego de algo. Em Informática, é o que indica a um computador uma ação elementar a ser executada.

### 1.3. ALGORITMO



Um **Algoritmo** é formalmente uma sequência finita de passos que levam à execução de uma tarefa.

Podemos pensar em algoritmo como uma receita, uma sequência de instruções para se atingir uma meta específica. Essas tarefas não podem ser redundantes nem subjetivas na sua definição, devem ser claras e precisas.

Como exemplos de algoritmos podemos citar os das operações básicas (adição, multiplicação, divisão e subtração) de números reais decimais.

Outros exemplos seriam os manuais de aparelhos eletrônicos, como um videocassete, que explicam passo-a-passo como, por exemplo, gravar um evento.

Até mesmo as coisas mais simples, podem ser descritas por sequências lógicas. Por exemplo:

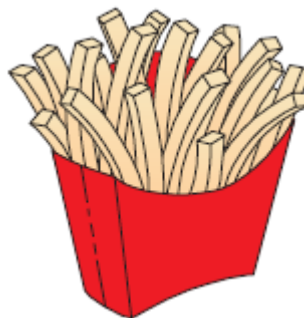
**“Chupar uma bala”.**

- Pegar a bala
- Retirar o papel.
- Chupar a bala.
- Jogar o papel no lixo.



**“Fritar Batatas”** – Suponhamos a existência de uma despensa onde se encontram as batatas e a existência de um cesto para se colocar tais batatas. Suponhamos ainda que o óleo, a panela, a faca, o prato e o sal estejam em um armário.

- Pegar cesto.
- Ir à dispensa.
- Colocar as batatas no cesto.
- Trazer o cesto para próximo do fogão.
- Pegar a panela e o óleo no armário.
- Colocar o avental.
- Pegar a faca no armário.
- Descascar as batatas que estão no cesto.
- Cortar as batatas em pedaços menores.
- Colocar o óleo na panela.
- Colocar a panela com óleo no fogão.
- Ligar o fogão.
- Aguardar até o óleo esquentar.
- Colocar as batatas no óleo.
- Aguardar fritar.
- Pegar prato no armário.
- Retirar batatas da panela.
- Colocar as batatas no prato.
- Pegar o sal no armário.
- Colocar sal na batata.
- Jogar cascas da batata no lixo.



Vamos supor que a colocação do avental seja condicionada à cor da roupa da pessoa que frita as batatas. Se a cor da roupa for clara, o avental é colocado, senão, o avental não é colocado. Vejamos como fica o algoritmo:

- Pegar cesto.
- Ir à dispensa.
- Colocar as batatas no cesto.
- Trazer o cesto para próximo do fogão.
- Pegar a panela e o óleo no armário.
- **Se** roupa clara, então Colocar o avental.



- Pegar a faca no armário.
- Descascar as batatas que estão no cesto.
- Cortar as batatas em pedaços menores.
- Colocar o óleo na panela.
- Colocar a panela com óleo no fogão.
- Ligar o fogão.
- Aguardar até o óleo esquentar.
- Colocar as batatas no óleo.
- Aguardar fritar.
- Pegar prato no armário.
- Retirar batatas da panela.
- Colocar as batatas no prato.
- Pegar o sal no armário.
- Colocar sal na batata.
- Jogar cascas da batata no lixo.

Nós podemos também subdividir certas instruções. Por exemplo, a instrução:

- Descascar as batatas que estão no cesto.

Pode ser mais bem detalhada na forma:

- Enquanto houver batatas não descascadas no cesto faça
  - ✓ Descascar uma batata.

## 1.4. LINGUAGEM DE COMPUTADOR

O computador possui uma linguagem própria, formada por conjuntos específicos de 'zeros e uns' (Linguagem Binária ou de Máquina) para a qual todos os caracteres da linguagem humana são convertidos de modo que o computador possa entender os comandos que desejamos que ele execute.



A **Linguagem de Máquina** é a forma de se representar os dois estados (ligado/desligado) da corrente elétrica utilizados nos circuitos eletrônicos dos computadores. É a única linguagem inteligível pelo computador.



## 1.5. LINGUAGEM DE PROGRAMAÇÃO

Como a Linguagem de Máquina é de difícil compreensão e manipulação, foram desenvolvidas linguagens intermediárias entre a da máquina e a do homem; essas linguagens denominam-se Linguagens de Programação.



A **Linguagem de Programação** é um método padronizado para expressar instruções para um computador.



Dizemos que quanto mais próxima da linguagem humana for a linguagem de programação ela é de alto nível, caso contrário é considerado baixo nível.

Os programas são geralmente, escritos em linguagem de programação e convertidos para linguagem de máquina através de programas específicos. Tais programas se dividem em duas categorias:

- **Compiladores:** convertem o programa escrito em uma linguagem de programação em linguagem de máquina uma única vez, a partir daí, toda vez que o programa é executado ele é executado já no formato binário.
- **Interpretadores:** convertem o programa escrito em uma linguagem de programação em linguagem de máquina toda vez que ele for executado.

## 1.6. PROGRAMA



Os **Programas de Computadores** nada mais são do que algoritmos escritos numa linguagem de programação (Portugol Studio, Pascal, C, Cobol, Fortran, Visual Basic, Java, entre outras) e que são interpretados e executados por uma máquina, no caso um computador.

Notem que dada essa interpretação rigorosa, um programa é, por natureza, muito específico e rígido em relação aos algoritmos da vida real.



Notar que: Toda linguagem de programação define um conjunto de instruções que ela consegue converter para linguagem de máquina. Logo se tivermos uma linguagem que entenda a instrução:

- Descascar todas as batatas.

Ele irá executar tal instrução, entretanto normalmente as linguagens de programação não definem instruções tão específicas, daí é necessário transformar essa instrução em sub-instruções compreensíveis pela linguagem escolhida e passível de geração de linguagem de computador.



***Vamos resumir o conteúdo visto neste capítulo:***

Aprendemos conceitos importantes, como:

- **Instruções**, em informática, é o que indica a um computador uma ação elementar a executar.
- **Algoritmo** é uma sequência finita de passos que levam a execução de uma tarefa.
- **Linguagem de Máquina ou de Computador** é a única linguagem inteligível pelo computador.
- **Linguagem de Programação** é um método padronizado para expressar instruções para um computador.
- **Programa** são algoritmos escritos numa linguagem de programação.



## Atividades

1. Elabore um algoritmo para tomar banho.
2. Faça um algoritmo para trocar o pneu de um carro.
3. Faça um algoritmo para trocar uma lâmpada. (Descreva em detalhes)
4. Escreva um algoritmo para descrever como você faz para ir de sua casa até a escola.



## Capítulo 2 - DESENVOLVENDO ALGORITMOS



Prezado aluno,

Agora que você já tem uma visão geral de algoritmos, apresentaremos neste capítulo como desenvolver alguns algoritmos simples.

Os algoritmos desenvolvidos nesta fase da disciplina são bem simples, todavia não menos importantes, pois a partir do entendimento destes você terá uma maior facilidade na compreensão dos demais.

Bom estudo!

### 2.1. PSEUDO CÓDIGO

Os algoritmos são descritos em uma linguagem chamada pseudocódigo. Esse nome é uma alusão à posterior implementação em uma linguagem de programação, ou seja, quando formos programar em linguagem C, por exemplo, estaremos gerando código em C. Por isso os algoritmos são independentes das linguagens de programação. Ao contrário de uma linguagem de programação, não existe um formalismo rígido de como deve ser escrito o algoritmo.



O algoritmo deve ser fácil de interpretar e fácil de codificar. Ou seja, ele deve ser o intermediário entre a linguagem falada e a linguagem de programação.

### 2.2. DIRETRIZES PARA CONSTRUÇÃO DE ALGORITMOS

Para escrever um algoritmo, precisamos descrever a sequência de instruções, de maneira simples e objetiva. Para isso, utilizaremos algumas técnicas:

- Usar somente um verbo por frase.



- Imaginar que você está desenvolvendo um algoritmo para pessoas que não trabalham com informática.
- Usar frases curtas e simples.
- Ser objetivo.
- Procurar usar palavras que não tenham sentido dúbio.

## 2.3. PARTES

No Capítulo anterior, vimos que ALGORITMO é uma sequência lógica de instruções que podem ser executadas.

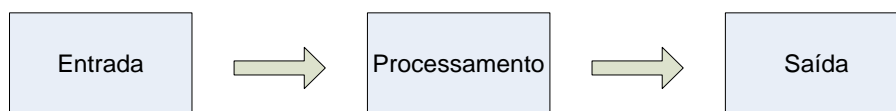
É importante ressaltar que qualquer tarefa que siga determinado padrão pode ser descrita por um algoritmo, como por exemplo:

***COMO FAZER ARROZ DOCE***

ou então

***CALCULAR O SALDO FINANCEIRO DE UM ESTOQUE***

Entretanto, ao montar um algoritmo, precisamos primeiro dividir o problema apresentado em três partes fundamentais (Figura 2-1).



**Figura 2-1: Partes Fundamentais.**

Onde temos:

**ENTRADA:** São os dados de entrada do algoritmo. Em algoritmos de alto nível, podem ser identificados por verbos como: Ler, Receber etc.

**PROCESSAMENTO:** São os procedimentos utilizados para chegar ao resultado final

**SAÍDA:** São os dados já processados. Em algoritmos de alto nível, podem ser identificados por verbos como: Retornar, Imprimir, Mostrar etc.

A analogia com o homem pode ser observada na Figura 2-2.



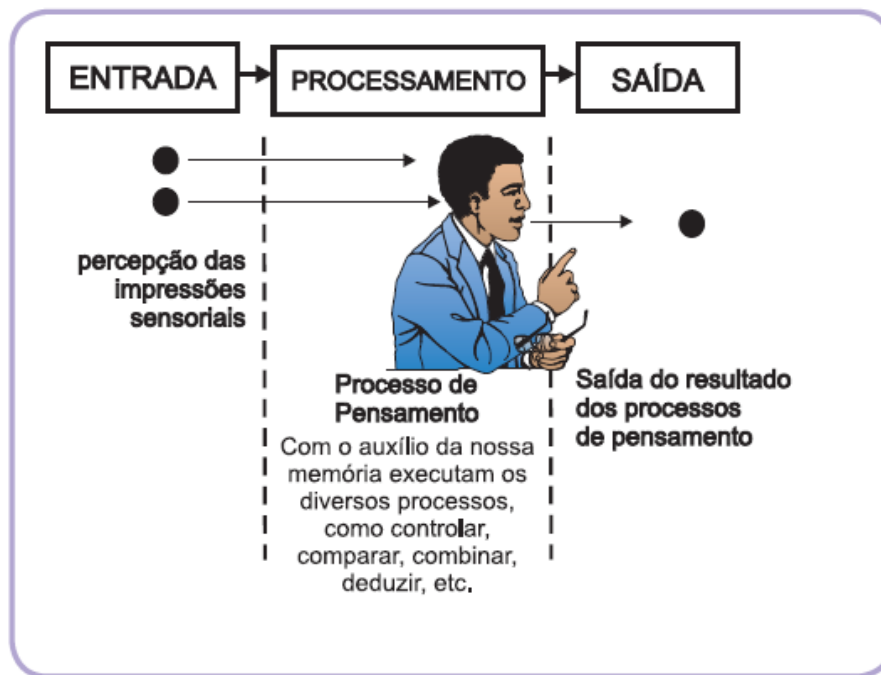


Figura 2-2: Analogia com o Homem.

## 2.4. EXEMPLO DE ALGORITMO

Imagine o seguinte problema:

Calcular a média final de um aluno da 3ª Série. O aluno realizará quatro provas: P1, P2, P3 e P4.

Onde:

$$\text{Média Final} = (P1 + P2 + P3 + P4) / 4$$

Para montar o algoritmo proposto, faremos três perguntas:

**a) Quais são os dados de entrada?**

R: Os dados de entrada são P1, P2, P3 e P4

**b) Qual será o processamento a ser utilizado?**

R: O procedimento será somar todos os dados de entrada e dividi-los por 4 (quatro)

**c) Quais serão os dados de saída?**

R: O dado de saída é a média final



## Algoritmo

- Receba a nota da prova1.
- Receba a nota de prova2.
- Receba a nota de prova3.
- Receba a nota da prova4.
- Some todas as notas e divida o resultado por 4.
- Mostre o resultado da divisão.



### ***Vamos resumir o conteúdo visto neste capítulo:***

Aprendemos que os algoritmos podem ser descritos utilizando-se uma linguagem denominada Pseudo-Código. Essa linguagem nos ajuda a entendermos os passos que um algoritmo possui.

Compreendemos algumas regras básicas que devem ser seguidas para a construção de algoritmos, como: utilizar apenas um verbo por frase, etc.

E, principalmente, passamos a entender que podemos dividir nosso algoritmo em três partes: Entrada, Processamento e Saída.



## Atividades

1. Identifique os dados de entrada, processamento e saída no algoritmo abaixo:
  - Receba código da peça.
  - Receba valor da peça.
  - Receba Quantidade de peças.
  - Calcule o valor total da peça (Quantidade \* Valor da peça).
  - Mostre o código da peça e seu valor total.
2. Faça um algoritmo para “Calcular o estoque médio de uma peça”, sendo que:  
$$\text{ESTOQUE M\u00c9DIO} = (\text{QUANTIDADE M\u00cdNIMA} + \text{QUANTIDADE M\u00c1XIMA}) / 2$$
3. Teste o algoritmo anterior com dados definidos por você.




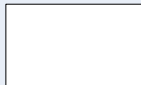

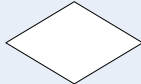

## Capítulo 3 - DIAGRAMA DE BLOCO

### 3.1. O QUE É UM DIAGRAMA DE BLOCO?

O Diagrama de Blocos é uma forma padronizada e eficaz para representar os passos lógicos de um determinado processamento.

Com o diagrama podemos definir uma sequência de símbolos, com significado bem definido. Portanto, sua principal função é a de facilitar a visualização dos passos de um processamento.

Existem diversos símbolos em um diagrama de bloco. Na Figura 3-1 apresentaremos os principais:

SÍMBOLO	FUNÇÃO
 <b>TERMINAL</b>	Indica o INÍCIO ou FIM de um processamento. <i>Exemplo: Início do Algoritmo.</i>
 <b>PROCESSAMENTO</b>	Processamento em geral. <i>Exemplo: Cálculo de dois números</i>
 <b>ENTRADA DE DADOS</b>	Operação de entrada e saída de dados. <i>Exemplo: Leitura e gravação de arquivos.</i>
 <b>DECISÃO</b>	Indica uma decisão a ser tomada. <i>Exemplo: Verificação de Sexo.</i>
 <b>EXIBIR</b>	Mostra informações ou resultados <i>Exemplo: Mostre o resultado do cálculo.</i>

**Figura 3-1: Símbolos de um Diagrama de Blocos.**

Dentro do símbolo sempre haverá algo escrito, pois somente os símbolos não nos dizem nada. Veja no exemplo a seguir:



Exemplos de Diagrama de Bloco (Figura 3-2):

“CHUPAR UMA BALA”      “CALCULAR MÉDIA DE 4 NOTAS”

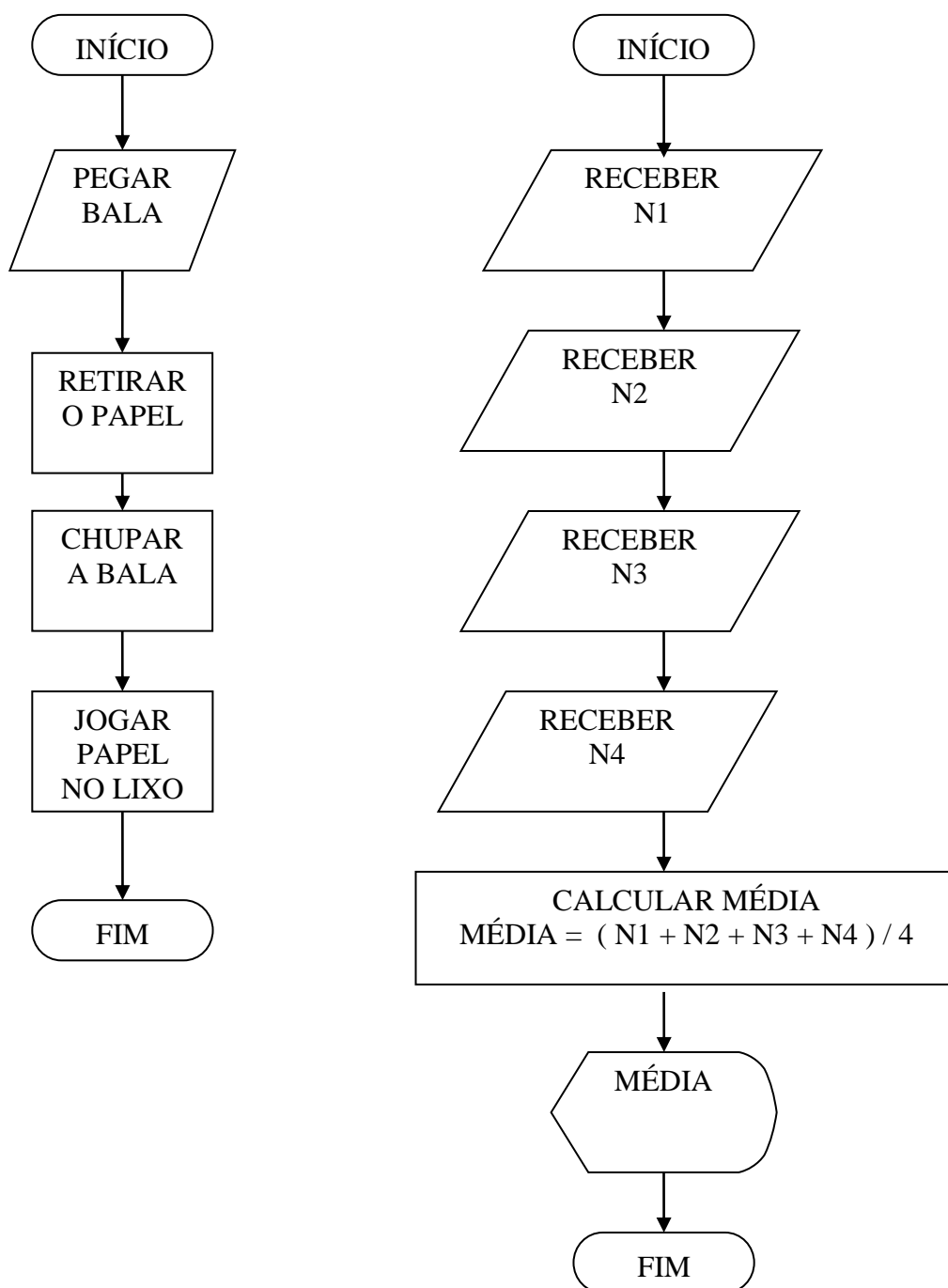


Figura 3-3: Exemplos de Diagramas de Blocos.



***Vamos resumir o conteúdo visto neste capítulo:***  
Aprendemos a construir algoritmos graficamente, utilizando símbolos padrões, os quais juntos constroem o que chamado de Diagrama de Blocos. Será importante este entendimento, pois alguns algoritmos mais complexos, de capítulos posteriores, serão melhor entendidos graficamente, ou seja, em Diagrama de Blocos.



## Atividades

1. Construa um diagrama de blocos que:
  - Leia a cotação do dólar.
  - Leia um valor em dólares.
  - Converta esse valor para Real.
  - Mostre o resultado.
2. Desenvolva um diagrama que:
  - Leia 4 (quatro) números.
  - Calcule o quadrado para cada um.
  - Somem todos os quadrados.
  - Mostre o resultado.
3. Faça testes numéricos para testar seu algoritmo.
4. Identifique nestes algoritmos o que é entrada, processamento e saída.





## Capítulo 4 - PORTUGOL - ITENS FUNDAMENTAIS



Prezado aluno,

Para compreendermos melhor como elaborar algoritmos que possam ser executados em um computador, precisamos escolher uma Linguagem de Programação. A partir desta escolha, poderemos então escrever nossos algoritmos na Linguagem de Programação escolhida, fazendo com que estes algoritmos possam ser considerados Programas de Computador.

Nesta disciplina, a Linguagem de Programação escolhida foi o **Portugol**, que é uma Linguagem bem próxima da nossa linguagem natural e, por isso, ficará bem fácil de aprendermos a Lógica de Programação em si.

A partir deste capítulo, os conceitos apresentados serão associados à Linguagem de Programação Portugol.

Bom estudo!

### 4.1. CONSTANTES



Uma **Constante** é um determinado valor fixo que não se modificará ao longo do tempo, durante a execução do programa.

Uma constante pode ser um número (como se conhece na Matemática), um valor lógico ou uma sequência de caracteres com algum significado para o problema em estudo. No contexto de Algoritmos escritos em Portugol uma constante pode ser classificada como sendo numérica (inteira ou real), lógica ou literal (cadeia).



### 4.1.1. Constantes Numéricas

A representação de uma constante numérica nos algoritmos é feita no sistema decimal, podendo ser um número com ou sem parte fracionária.

Exemplos:

- 33
- 44.21
- -4

É comum a representação de constantes com parte exponencial, isto é, um fator de 10 elevado a um expoente inteiro.

Exemplos:

- $44.17 \times 10^5$
- $22.35 \times 10^{-1}$

### 4.1.2. Constantes Lógicas

É um valor lógico, isto é, só pode ser falso ou verdadeiro.

### 4.1.3. Constantes Literais (Cadeia)

Uma constante deste tipo pode ser qualquer sequência de caracteres (literais, dígitos ou símbolos especiais) que formem um literal com algum significado para o problema em estudo.

Para que não ocorra confusão na identificação de constantes literais elas serão colocadas entre aspas.

Exemplos:

- “JOÃO DE DEUS”
- “MK3227”
- “23/02/2006”





## 4.2. VARIÁVEIS

Na Matemática, uma variável é a representação simbólica dos elementos de certo conjunto.



Na Informática, uma **Variável** é um espaço de memória em uma determinada posição, cujo conteúdo pode variar ao longo do tempo durante a execução de um programa, mas a cada instante existe um único valor.

Toda variável é definida por um nome ou identificador. Um identificador é uma sequência de caracteres alfanuméricos válida na nomeação de variáveis. Nos compiladores atuais, uma sequência validade de caracteres alfanuméricos é definida como:

- O primeiro caractere deve ser obrigatoriamente uma letra.
- Os caracteres seguintes devem ser letras ou dígitos, não sendo permitido o uso de caracteres especiais.



Não devemos utilizar acentos (‘, ^, ~). Isso porque as linguagens de programação classificam estes símbolos como caracteres especiais.

### Exemplos:

#### Identificadores válidos

- A
- NOTA
- X5
- A33C423
- FITTKM
- nome\_da\_cidade

#### Identificadores Não Válidos

- 5B
- (14)B5



- B(B)
- X – Y
- 5(KM32)



Considerando a linguagem Portugal é importante entender que: o caractere *underline* ('\_') pode ser utilizado em identificadores. Já o caractere *hífen* ('-') não deve ser utilizado.

#### 4.2.1. Declaração de variáveis

No Visualg, as variáveis só podem armazenar valores de um mesmo tipo, podendo ser classificadas em numéricas, lógicas e literais, assim como as constantes.

A declaração de variáveis deve ser feita da seguinte forma:

**<Nome do Tipo > <Lista de Identificadores >**

Onde:

**Lista de Identificadores:** é a lista de nomes escolhidos para as variáveis, que devem estar separados por vírgula.

**Nome do Tipo:** é um dos quatro tipos que indicam o tipo da variável: inteiro, real, logico ou cadeia.

**Exemplos:**

- idade: inteiro.
- peso, altura, nota: real.
- cidade, nome, sobrenome : cadeia.
- flag, teste, maior\_de\_18 : logico.



Observe que o nome dos tipos deve estar sem acentos. Exemplos: inteiro, real, cadeia e logico. No Portugal a palavra **cadeia** está relacionada ao armazenamento de uma cadeia de caracteres.



### 4.3. EXPRESSÕES ARITMÉTICAS

Expressão aritmética é aquela que utiliza operadores aritméticos e cujos operandos são constantes e/ou variáveis do tipo numérico.

São operadores aritméticos: Adição, Subtração, Multiplicação, Divisão, Potenciação e Radiciação.

A tabela abaixo mostra exemplos de utilização de operadores na matemática em comparação com a utilização no Portugol Studio. Para tal, considere A e B como sendo variáveis numéricas.

OPERADOR	MATEMÁTICA (EXEMPLO)	PORTUGOL (EXEMPLO)
Adição	$A + B$	$A + B$
Subtração	$A - B$	$A - B$
Multiplicação	$A \times B$	$A * B$
Divisão	$A / B$	$A / B$
Potenciação	$A^B$	Próximo capítulo...
Radiciação	$\sqrt[B]{A}$	Próximo capítulo...

#### Exemplos de Expressões Aritméticas em Portugol:

- $X + Y$
- $X - Y$
- $2 * \text{NOTA}$

#### 4.3.1. Prioridade das Operações

Assim como na Matemática os operadores aritméticos obedecem a seguinte ordem:

1º Potenciação e Radiciação.

2º Multiplicação e Divisão.

3º Adição e Subtração.

Para mudar a precedência podemos, como na Matemática, utilizar parênteses.



Colchetes e Chaves não devem ser utilizados nas expressões aritméticas, pois são usados nos algoritmos com outras finalidades.

#### 4.4. EXPRESSÕES RELACIONAIS

Expressões Relacionais são aquelas que utilizam operadores relacionais, cujos operandos em uma relação são do mesmo tipo, e cujo resultado é do tipo lógico.

São operadores relacionais (Tabela 4-1):

DESCRIÇÃO	SÍMBOLO
IGUAL A	=
DIFERENTE DE	!=
MAIOR QUE	>
MENOR QUE	<
MAIOR OU IGUAL A	>=
MENOR OU IGUAL A	<=

**Tabela 4-1: Operadores Relacionais.**

##### **Exemplo:**

Suponhamos duas variáveis do tipo numérico:

A = 50

B = 30

Ao fazermos comparações entre elas, teríamos a seguinte solução (Tabela 4-2):



EXPRESSÃO	RESULTADO
$A == B$	FALSO
$A != B$	VERDADEIRO
$A > B$	VERDADEIRO
$A < B$	FALSO
$A >= B$	VERDADEIRO
$A <= B$	FALSO

Tabela 4-2: Solução para  $A = 50$  e  $B = 30$ .

Fazendo uma pequena referência aos diagramas de blocos, o símbolo que indica os operadores relacionais é (Figura 4-1):

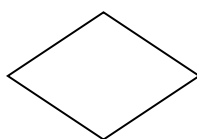


Figura 4-4: Símbolo do operador Relacional.

Vejamos, então, alguns exemplos de uso. Na Figura 4-2, temos a comparação de duas variáveis, na Figura 4-3 temos a comparação de uma variável numérica com uma constante numérica e na Figura 4-4 temos a comparação de uma variável (literal) com uma constante literal.

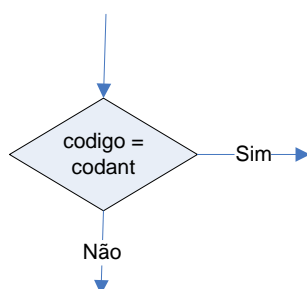
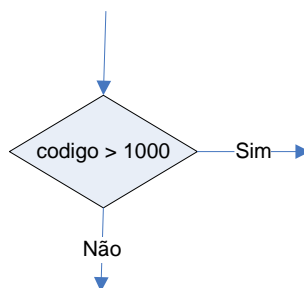
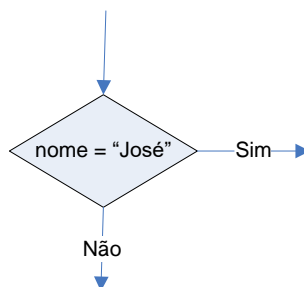


Figura 4-5: Comparando duas variáveis.



**Figura 4-6: Comparando uma variável (numérica) com uma constante numérica.**



**Figura 4-7: Comparando uma variável (literal) com uma constante literal.**

## 4.5. EXPRESSÕES LÓGICAS

Expressões lógicas são aquelas que utilizam operadores lógicos e cujos operandos são expressões relacionais, pois seus resultados são do tipo lógico, constantes e/ou variáveis do tipo lógico. Toda expressão lógica tem como resultado um valor lógico TRUE ou FALSE.

Os operadores lógicos são (Tabela 4-3):

e
ou
nao

**Tabela 4-3: Operadores Lógicos.**

e: Uma expressão lógica “e” é verdadeira (true) se todos os seus operandos possuírem o valor TRUE.

ou: Uma expressão lógica “ou” é verdadeira (true) se algum de seus operandos possuir o valor TRUE.

nao: Uma expressão lógica “nao” é verdadeira (true) se seu operando é FALSE.



A Tabela 4-4 ilustra o comportamento dos operadores lógicos:

1º VALOR	OPERADOR	2º VALOR	RESULTADO
TRUE	e	TRUE	TRUE
TRUE	e	FALSE	FALSE
FALSE	e	TRUE	FALSE
FALSE	e	FALSE	FALSE
TRUE	ou	TRUE	TRUE
TRUE	ou	FALSE	TRUE
FALSE	ou	TRUE	TRUE
FALSE	ou	FALSE	FALSE
TRUE	nao		FALSE
FALSE	nao		TRUE

**Tabela 4-4: Comportamento dos Operadores Lógicos.**

**Exemplo:**

Suponha que temos 3 variáveis  $A = 50$ ,  $B = 80$  e  $C = 10$

Os resultados das expressões seriam conforme a Tabela 4-5:

EXPRESSÕES			RESULTADO
$A == B$	e	$B > C$	FALSE
$A != B$	ou	$B < C$	TRUE
$A > B$	nao		TRUE
$A < B$	e	$B > C$	TRUE
$A >= B$	ou	$B == C$	FALSE
$A <= B$	nao		FALSE

**Tabela 4-5: Resultado das expressões de acordo com o exemplo dado.**



***Vamos resumir o conteúdo visto neste capítulo:***

Neste capítulo definimos:

**Constante:** é um determinado valor fixo que não se modificará ao longo do tempo, durante a execução do programa.

**Variável:** é um espaço de memória em uma determinada posição, cujo conteúdo pode variar ao longo do tempo durante a execução de um programa, mas a cada instante existe um único valor.

Vimos também que podemos realizar expressões com estas constantes e variáveis. Os tipos de expressões são: Aritméticas, Relacionais e Lógicas.



## Atividades

1. Identifique o tipo das constantes utilizando a seguinte legenda:

1 – Constante Numérica

2 – Constante Lógica

3 – Constante Literal

- |  |  |
|--|--|
| a. <input type="checkbox"/> 33                   | h. <input type="checkbox"/> falso              |
| b. <input type="checkbox"/> "33"                 | i. <input type="checkbox"/> verdadeiro         |
| c. <input type="checkbox"/> "Constante Lógica"   | j. <input type="checkbox"/> "falso"            |
| d. <input type="checkbox"/> "Constante Literal"  | k. <input type="checkbox"/> "123KO"            |
| e. <input type="checkbox"/> "Constante Numérica" | l. <input type="checkbox"/> "123verdadeiro123" |
| f. <input type="checkbox"/> 5878987.23           | m. <input type="checkbox"/> 9999999            |
| g. <input type="checkbox"/> "verdadeiro"         | n. <input type="checkbox"/> "fim"              |

2. Assinale X nos identificadores válidos:

- |                                   |   |
|-----------------------------------|---|
| a. <input type="checkbox"/> VALOR | h. <input type="checkbox"/> AHHHH           |
| b. <input type="checkbox"/> 33    | i. <input type="checkbox"/> AHHHH!!!        |
| c. <input type="checkbox"/> "33"  | j. <input type="checkbox"/> SALARIO-LÍQUIDO |
| d. <input type="checkbox"/> X2    | k. <input type="checkbox"/> NOTA*DO*ALUNO   |
| e. <input type="checkbox"/> 2X    | l. <input type="checkbox"/> m{23}           |
| f. <input type="checkbox"/> "x2"  | m. <input type="checkbox"/> sacada256       |
| g. <input type="checkbox"/> "2X"  | n. <input type="checkbox"/> 256sacada       |
|                                   | o. <input type="checkbox"/> NOME DA EMPRESA |





3. Supondo que temos as variáveis SALARIO, IR e SALLIQ e que os valores estão representados na tabela abaixo. Aplicando as expressões na tabela qual é o resultado lógico obtido:

SALARIO	IR	SALLIQ	EXPRESSÃO	V ou F
100,00	0,00	100	(SALLIQ >= 100,00)	
200,00	10,00	190,00	(SALLIQ < 190,00)	
300,00	15,00	285,00	SALLIQ = SALARIO - IR	

4. Sabendo que  $A = 3$ ,  $B = 7$  e  $C = 4$ , informe se as expressões abaixo são V (verdadeiro) ou F (falso).

- a.  $(A + B) > C$
- b.  $B \geq (A + 2)$
- c.  $C == (B - A)$
- d.  $(B - A) \leq C$
- e.  $(B + A) \leq C$
- f.  $(C + A) > B$

5. Repita o exercício anterior com  $A = 1$ ,  $B = 2$  e  $C = 16$ .

- a.  $(A + B) > C$
- b.  $B \geq (A + 2)$
- c.  $C = (B - A)$
- d.  $(B - A) \leq C$
- e.  $(B + A) \leq C$
- f.  $(C + A) > B$



## Capítulo 5 - PORTUGOL – ITENS COMPLEMENTARES



Caro aluno,

Até o presente momento, abordamos conceitos fundamentais com relação a Linguagem de Programação Portugal.

Vamos continuar aprofundando nossos conhecimentos sobre essa Linguagem de Programação neste capítulo.

Leia com atenção todas as novas informações aqui contidas, pois começaremos a programar de verdade.

Depois de entendermos os conceitos vamos, de fato, utilizar a Ferramenta de Programação Portugal Studio.

Bom estudo!

### 5.1. COMENTÁRIOS

Quando implementamos um algoritmo, é fundamental a preocupação com sua clareza (facilidade de entendimento). Um comentário é uma observação que se coloca no código, de forma a possibilitar seu entendimento.

No Portugal, os comentários são representados por um texto após '//’.

No seguinte exemplo a declaração das variáveis foi comentada de forma a explicar para que serve cada variável:

**inteiro** MATRICULA // número da matricula do aluno

**real** NOTA // nota do aluno

**inteiro** CODIGO // código do curso



## 5.2. FUNÇÕES MATEMÁTICAS

Além dos operadores aritméticos normalmente utilizados, podemos usar funções disponibilizadas pela linguagem Portugol, que comumente estão presentes em muitas linguagens (Quadro 6):

NOME	RESULTADO
Matematica.potencia(X, Y)	Potência de X elevado a Y
Matematica.raiz(X, Y)	Raiz no índice Y de X
Matematica.logaritmo (X, Y)	Logaritmo na base Y de X
Matematica.valor_absoluto(X)	Valor Absoluto de X
Matematica.arredondar(X, Y)	Arredonda X em Y casas decimais
X % Y	Resto da divisão de X por Y
Matematica.seno( X )	Função <i>seno</i> aplicada ao argumento X (em radianos)
Matematica.cosseno( X )	Função <i>cosseno</i> aplicada ao argumento X (em radianos)
Matematica.tangente( X )	Função <i>tangente</i> aplicada ao argumento X (em radianos)

Tabela 5-1: Funções do Portugol Studio.

### Exemplos de Funções em Portugol Studio:

Sendo A, B, X, Y variáveis do tipo numérico, quais os resultados fornecidos por cada uma das seguintes funções, onde A = 10, B = 3 e X = 2.5 e Y = 1.2.

- $A \text{ div } B = 10 \text{ div } 3 = 3$
- $A \text{ mod } B = 10 \text{ mod } 3 = 1$
- $\text{abs} ( B - A ) = \text{abs} ( 3 - 10 ) = \text{abs} ( -7 ) = 7$
- $\text{int} ( 4 * Y ) = \text{int} ( 4 * 1.2 ) = \text{int} ( 4.8 ) = 4$

## 5.3. COMANDOS DE ENTRADA E SAÍDA

Precisamos de alguns comandos para entrada/saída de dados nos algoritmos. Para tanto, utilizaremos os seguintes comandos:

### Entrada:

**leia** (<Nome da Variável>)

Onde:

leia: é uma palavra-chave.

( ): Os parênteses delimitam o identificador da variável.



<Nome da Variável>: é o nome da variável na qual será armazenado o valor proveniente do meio de entrada.



É importante ressaltar que as linguagens de programação normalmente utilizadas obrigam a leitura variável a variável.

### Saída:

**escreva** (<Lista de variáveis>)

Onde:

escreva: é uma palavra chave.

( ): Os parênteses delimitam os identificadores das variáveis

<Lista de variáveis>: são os nomes das variáveis, separados por vírgula, das quais serão obtidos os valores a serem escritos.



Você ainda pode utilizar o caractere '\n' para saltar linha na descrição do comando escreva.  
Por exemplo:  
escreva("Alô mundo! \n")

### Exemplos de Comandos de Entrada e Saída:

- leia (A)
- leia (NOME)
- escreva (A)
- escreva (NOME, SOBRENOME, TECEIRONOME)

## 5.4. ATRIBUIÇÃO

O comando responsável pela colocação de um valor em uma variável é o *comando de atribuição*. Tal comando pode ser apresentado da seguinte forma:

**Identificador = Expressão**



Onde:

Identificador: é o nome da variável à qual está sendo atribuído o valor.

= : é o símbolo de atribuição

Expressão: pode ser uma expressão aritmética, expressão lógica ou expressão literal, de cuja avaliação é obtido o valor a ser atribuído à variável.

### Exemplo de Atribuição:

- `NOTA = 5`
- `MEDIA = NOTA + 3`



Vamos resumir o conteúdo visto neste capítulo: Neste capítulo, aprendemos sobre alguns comandos da linguagem Portugol muito importantes:

Descrição	No Portugol
Comentários	//
Comandos de Entrada	leia
Comandos de Saída	escreva
Atribuição	=



### Atividades

1. Suponha as seguintes variáveis:

real SOMA, NUMERO, CONT  
cadeia NOME, COR, DIA, MES, ANO  
logico TESTE, CODIGO, TUDO

Determine que atribuições são válidas e escreva nas que não são o motivo de não serem:

- a. `NOME = 5`
- b. `SOMA = NUMERO + 2 * CONT`
- c. `TESTE = CODIGO ou CONT*2 != SOMA`
- d. `TUDO = SOMA`
- e. `COR = "Preto" – CONT*(1/2)`



- f. `CONT = CONT + 1`
- g. `NUMERO = “*abC*”`
- h. `DIA = “SEGUNDA FEIRA”`
- i. `MES = “AGOSTO”`
- j. `ANO = 2006`
- k. `SOMA + 2 = CONT*2 – NUMERO*(1/2)`
- l. `CONT = NOME >= CODIGO`

2. Determine os valores obtidos por cada uma das expressões aritméticas a seguir:



Para fazer este exercício é muito importante lembrar que:

$$\text{Matematica.raiz}(2,2) = \sqrt[2]{2}$$

$$\text{Matematica.raiz}(2,3) = \sqrt[3]{2}$$

Suponha as seguintes instruções antes das expressões aritméticas:

inteiro P, Q, R

real S

$$P = 2$$

$$Q = 3$$

$$R = 12$$

$$S = 4.5$$

- a. `100 * (Q % P) + R`
- b. `P * (R%5) - Q/2`
- c. `Matematica.valor_absoluto ( S - R ) + 100 * (Q % P) + R - P * (R % 5) - Q/2`
- d. `Matematica.raiz ( Matematica.potencia(P,2) , P ) + Matematica.arredondar ( S, 1 )`
- e. `R % (P + 1) - Q * R`
- f. `1 + ( (Matematica.potencia(P,3) + 2 * R ) - Matematica.arredondar (S+1, 0))`
- g. `Matematica.valor_absoluto (Matematica.arredondar (Matematica.valor_absoluto ( S - R ), 0 ) )`
- h. `Matematica.arredondar (Matematica.valor_absoluto ( S - R + 0.4 ) , 0)`
- i. `1 + (Q % (P)) * (Matematica.arredondar ((2*P*Q-S), 1))`
- j. `Matematica.valor_absoluto (Matematica.arredondar (Matematica.valor_absoluto ( R - S ), 1 ) )`
- k. `Matematica.arredondar (Matematica.valor_absoluto ( R - S ) , 1)`
- l. `P + Matematica.valor_absoluto ( 2.9 + Matematica.arredondar ( 0.3 + S, 1 ) * 2 )`



3. Determine o que será escrito pelo comando ESCREVA:

A) Supondo que o comando *leia* lerá informações a partir do teclado e que as informações digitadas serão as seguintes:

INFORMATICA, 35

**programa**

```
{
    funcao inicio()
    {
        cadeia N, P
        inteiro X, A
        X = 10
        leia (N)
        leia (A)
        X = X + A
        P = N
        escreva (P, X)
        X = X + A
        escreva (X)
        A = X
        escreva (N, X, A)
    }
}
```

Utilize as tabelas abaixo para responder o exercício 3.A:

Variáveis			
N	P	X	A
Comando <b>escreva</b>			
1			
2			
3			

B) Supondo que o comando *leia* lerá informações a partir do teclado e que as informações digitadas serão as seguintes:

JOAO, SILVA, SILVA, 35, 47, 26, VERDADEIRO, FALSO, VERDADEIRO



```
programa
{

    funcao inicio()
    {
        cadeia NOME, SOBRENOME, TERCEIRONOME
        inteiro NUM1, NUM2, NUM3
        logico FLAG1, FLAG2, FLAG3
        leia(NOME)
        leia(SOBRENOME)
        leia(TERCEIRONOME)
        leia(NUM1)
        leia(NUM2)
        leia(NUM3)
        leia(FLAG1)
        leia(FLAG2)
        leia(FLAG3)
        FLAG1 = ( SOBRENOME == TERCEIRONOME )
        FLAG2 = ( NUM2 <= NUM1 + NUM3 )
        FLAG3 = FLAG1 e (FLAG2 ou FLAG3)
        NUM1 = NUM2 + NUM1 - (NUM3 % 3)
        NUM2 = NUM1 + (NUM3 / 3)
        NUM3 = 10
        TERCEIRONOME = NOME
        NOME = SOBRENOME
        FLAG3 = ( NOME == TERCEIRONOME )
        escreva(NOME, SOBRENOME, TERCEIRONOME)
        escreva(NUM1, NUM2, NUM3)
        escreva(FLAG1, FLAG2, FLAG3)
    }
}
```

*Utilize as tabelas abaixo para responder o exercício 3.B:*



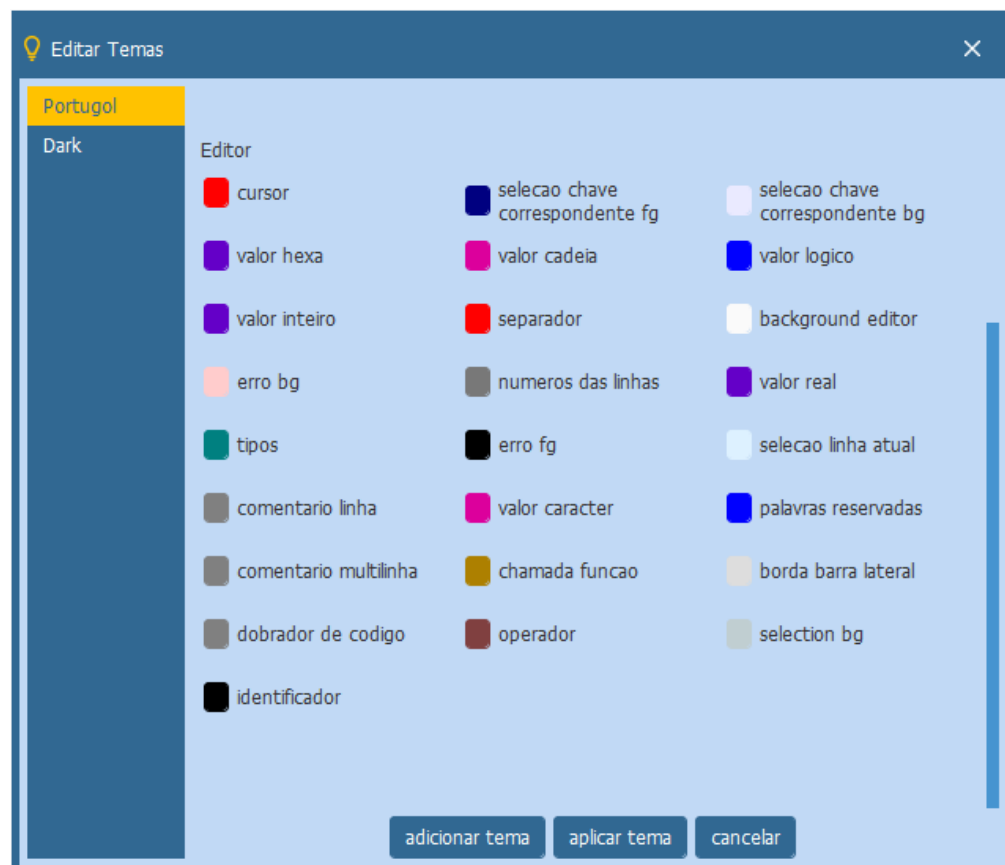


Variáveis								
N O M E	S O B R E N O M E	T E R C E I R O N O M E	N U M 1	N U M 2	N U M 3	F L A G 1	F L A G 2	F L A G 3

Comando <b>escreva</b>	
1	
2	
3	



Para facilitar o entendimento dos algoritmos descritos na apostila, daqui para frente, iremos colorir os códigos de acordo com o padrão de cores da ferramenta Portugol Studio. Cada cor tem um significado específico:





## Capítulo 6 - COMANDOS DE DECISÃO



Prezado programador (agora já posso te chamar assim),

Fizemos nossos primeiros programas utilizando a Linguagem de Programação Portugol Studio.

Neste capítulo, serão abordados os importantíssimos conceitos de Comandos de Decisão.

A partir de agora, você vai entender que nem todas as linhas de código de nossos programas precisam ser executadas pelo computador. Ou seja, vai aprender que por meio de Comandos de Decisão você pode criar diferentes possibilidades de execução de um determinado programa.

Bom estudo!

Os *comandos de decisão* ou *de desvio* fazem parte das técnicas de programação que conduzem a estruturas de programas que não são totalmente sequenciais. Com as instruções de SALTO ou DESVIO pode-se fazer com que o programa proceda de uma ou outra maneira, de acordo com as decisões lógicas tomadas em função dos dados ou resultados anteriores. As principais estruturas de decisão são:

- se
- se senao
- escolha caso



## 6.1. SE

A estrutura de decisão “se” normalmente vem acompanhada de um comando, ou seja, se determinada condição for satisfeita pelo comando “se” então execute determinado comando.

Imagine um algoritmo em que determinado aluno somente estará aprovado se sua média for maior ou igual a 5.0, veja no exemplo de algoritmo como ficaria.

```
se (media >= 5.0) {  
    escreva("Aluno Aprovado")  
}
```



O comando de decisão é caracterizado por:

```
se (expressão lógica){  
    comandos  
}
```

Onde:

*Expressão lógica* tem como resultado um valor lógico **verdadeiro** ou **falso**.

*Comandos* são linhas de código no algoritmo.

Em diagrama de blocos, ficaria conforme Figura 6-1:

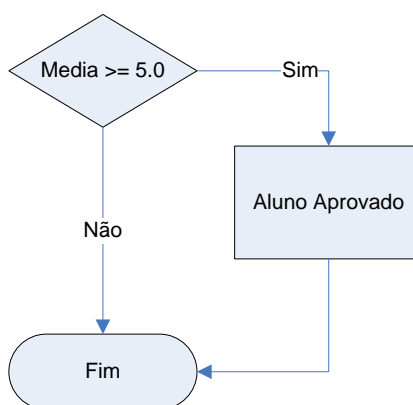


Figura 6-1: Diagrama de Blocos – se.



Vejamos outro exemplo com um algoritmo completo:

O algoritmo a seguir escreve uma mensagem caso o terceiro número lido seja maior que a soma dos outros dois números lidos.

```
programa
{
    funcao inicio()
    {
        real NUM1, NUM2, NUM3
        leia(NUM1)
        leia(NUM2)
        leia(NUM3)
        se (NUM1 + NUM2 < NUM3){
            escreva ("O NUM3 é maior NUM1 + NUM2")
        }
    }
}
```

## 6.2. SE ... SENAO

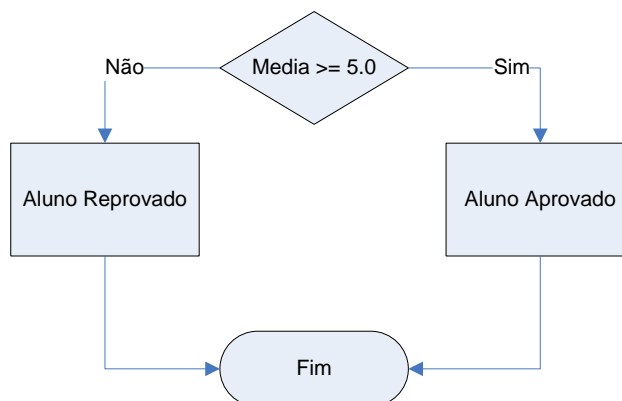
A estrutura de decisão “se/senao”, funciona exatamente como a estrutura “se”, com apenas uma diferença, em “se” somente podemos executar comandos caso a condição seja verdadeira, diferente de “se/senao” pois sempre um comando será executado independente da condição, ou seja, caso a condição seja “verdadeira”, seu comando será executado; caso contrário, o comando da condição “falsa” será executado.

Em algoritmo o exemplo da seção anterior ficaria assim:

```
se (media >= 5.0){
    escreva("Aluno aprovado")
}senao{
    escreva("Aluno reprovado")
}
```

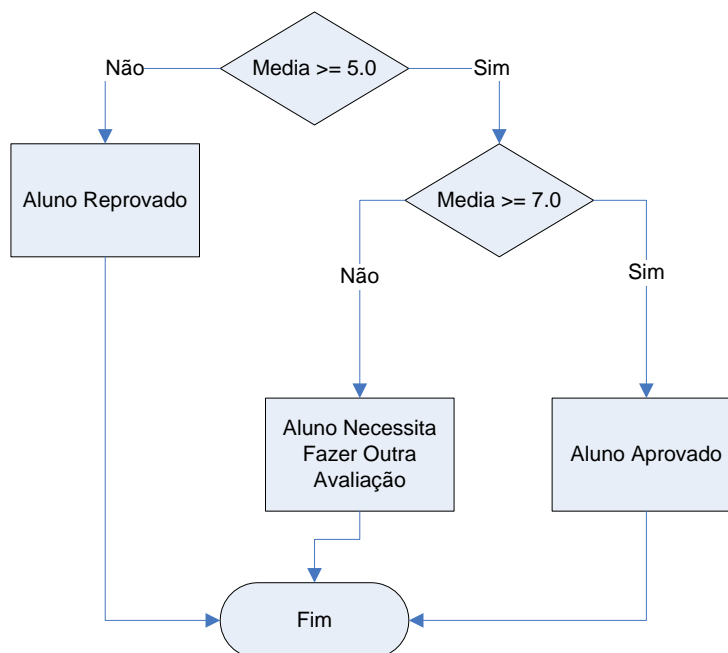


Em diagrama seria conforme Figura 6-2 .



**Figura 6-2: Diagrama de Blocos – se senao.**

No exemplo acima, está sendo executada uma condição que, se for verdadeira, executa o comando “APROVADO”, caso contrário executa o segundo comando “REPROVADO”. Podemos também testar outras condições dentro de uma mesma condição. Veja a Figura 6-3.



**Figura 6-3: Diagrama de Blocos – se senao se.**

Neste caso, poderíamos utilizar o seguinte trecho de algoritmo para representar o diagrama de blocos acima:

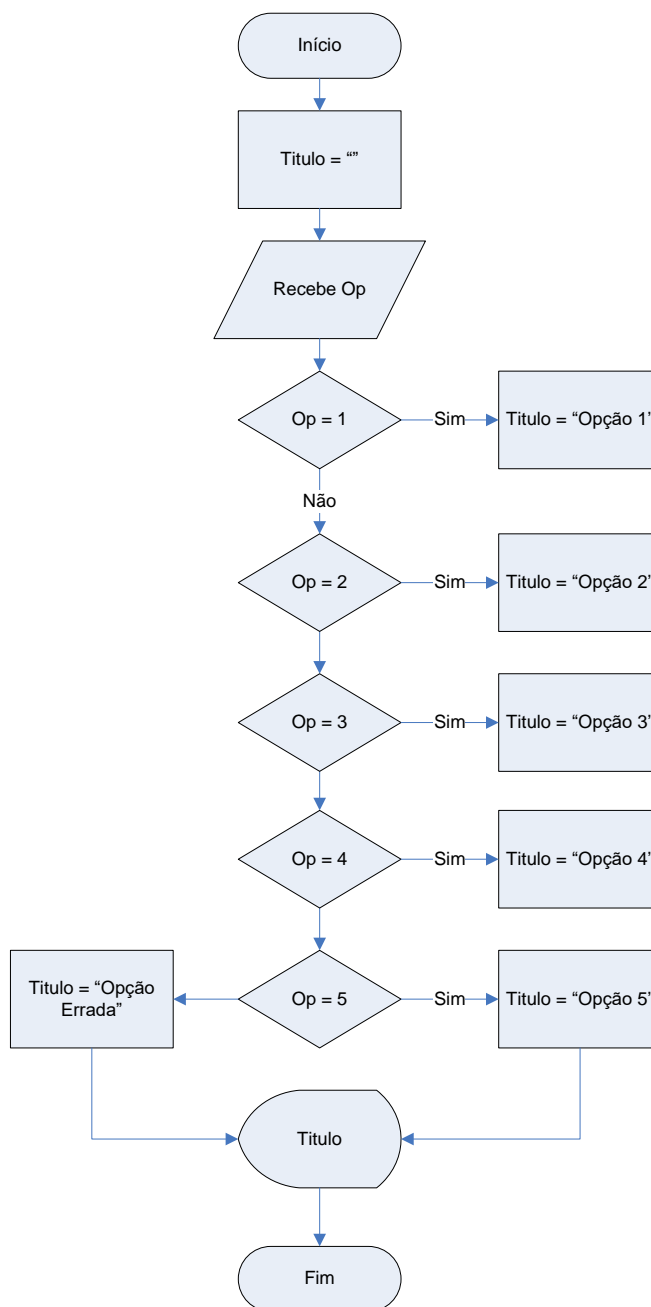


```
se (media >= 5.0){  
    se (media >= 7.0){  
        escreva("Aluno aprovado")  
    }senao{  
        escreva("Aluno em recuperação")  
    }  
}senao{  
    escreva("Aluno reprovado")  
}
```

### 6.3. ESCOLHA ... CASO

A estrutura de decisão “escolha ... caso” é utilizada para testar na condição uma única expressão que produz um resultado, ou, então, o valor de uma variável em que está armazenado um determinado conteúdo. Compara-se, então, o resultado obtido no teste com os valores fornecidos em cada cláusula “Caso”.

No exemplo do diagrama de blocos da Figura 6-4, uma variável “Op” é recebida e seu conteúdo é testado. Caso uma das condições seja satisfeita, é atribuída para a Variável Título a constante literal “Opção X”, caso contrário é atribuída a constante literal “Opção Errada”.



**Figura 6-4: Diagrama de Blocos – escolha caso.**

Neste caso, poderíamos utilizar o seguinte trecho de algoritmo para representar o diagrama de blocos anteriormente mostrado:





```
cadeia titulo
inteiro op
leia (op)
escolha (op){
    caso 1:
        titulo = "Opção 1"
        pare
    caso 2:
        titulo = "Opção 2"
        pare
    caso 3:
        titulo = "Opção 3"
        pare
    caso 4:
        titulo = "Opção 4"
        pare
    caso 5:
        titulo = "Opção 5"
        pare
    caso contrario:
        titulo = "Opção Incorreta"
}
escreva(titulo)
```



***Vamos resumir o conteúdo visto neste capítulo:***

Neste capítulo, aprendemos sobre comandos que nos ajudarão (e muito!) a programar:

**Comandos de Decisão:**

- se
- se senao
- escolha caso



## Atividades

1. Crie um algoritmo que receba pelo teclado o nome de um Funcionário e seu salário bruto. Se o salário for acima de R\$ 1.000,00, calcule 11% de desconto de INSS; se não, calcule 9%. Ao final, exiba o nome do funcionário, o salário bruto e o salário com o desconto de INSS.
2. Crie um algoritmo que receba pelo teclado o nome de um aluno e três notas. Ao final, deverá ser exibido o nome do aluno, sua média e o resultado (se for acima ou igual a 6, o aluno estará “aprovado”; se não for, estará “reprovado”).
3. Usando o algoritmo do item 2, altere o resultado para:
  - Média  $\leq 3$ , “reprovado”, Média  $< 6$ , “recuperação” e Média  $\geq 6$ , “aprovado”.
4. Crie um algoritmo em que, dada a tabela a seguir, calcule o valor de desconto a ser concedido para um determinado cliente, de acordo com o valor da compra. O algoritmo deverá receber pelo teclado o nome do cliente e o valor total da compra.

Valor a compra	% de desconto
Até R\$ 1.000,00	5
Entre R\$ 1.000,00 a R\$ 5.000,00	10
Acima de R\$ 5.000,00	15

5. Crie um algoritmo que calcule o valor a ser pago de acordo com a quantidade de m<sup>2</sup> de um tecido no valor de R\$ 25,00. Se a quantidade de metros for acima de 10 m<sup>2</sup>, o valor do m<sup>2</sup> passa a ser R\$ 23,00. Ao final, o algoritmo deverá exibir a quantidade de m<sup>2</sup>, o valor pago pelo m<sup>2</sup> e o valor a ser pago.
6. Usando a linguagem Visualg, desenvolva um programa que leia dois números inteiros e execute a operação que o usuário deseja realizar. O usuário poderá escolher entre as operações: somar, subtrair, multiplicar ou dividir. Ao final, o programa deverá exibir o resultado da operação. Utilize o comando **escolha-caso**.

### Explicando:

Exibiremos, para o usuário, as operações disponíveis e solicitaremos os dois números e a opção, assim:



Digite 1 para somar.

Digite 2 para subtrair.

Digite 3 para multiplicar.

Digite 4 para dividir.

Primeiro número: (leia o número 1)

Segundo número: (leia o número 2)

Digite a opção: (leia a opção)

Assumindo que o usuário digitou os números 20 e 40 e a opção 1, o resultado exibido será 60.

**Obs.:** Se o usuário digitar um valor inválido para opção deve ser exibido uma mensagem: "Opção Inválida!"



## Capítulo 7 - COMANDOS DE REPETIÇÃO



Prezado programador,

Neste capítulo, serão abordados os importantíssimos conceitos Comandos de Repetição.

Por meio de Comandos de Repetição, compreenderá que certo trecho do código de um programa pode ser executado repetidamente um determinado número de vezes.

Bom estudo!

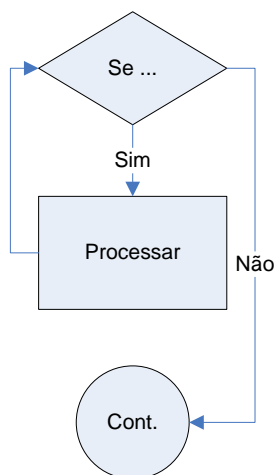
A estrutura de repetição permite que uma sequência de comandos seja executada repetidamente até que uma determinada condição de interrupção seja satisfeita.

Trabalharemos com modelos de comandos de repetição:

- enquanto
- repita
- para

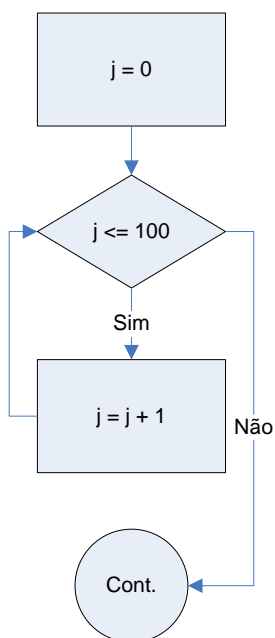
### 7.1. ENQUANTO

Neste caso, o bloco de operações será executado enquanto a condição “x” for verdadeira. O teste da condição será sempre realizado antes de qualquer operação. Enquanto a condição for verdadeira o processo se repete. Podemos utilizar essa estrutura para trabalharmos com contadores. Em diagrama de bloco a estrutura é a seguinte (Figura 6-5):



**Figura 6-5: Diagrama de Blocos – enquanto.**

A Figura 6-6 mostra um exemplo em Diagrama de Blocos. Em seguida, temos o algoritmo.



**Figura 6-6 Diagrama de Blocos – Exemplo: enquanto.**



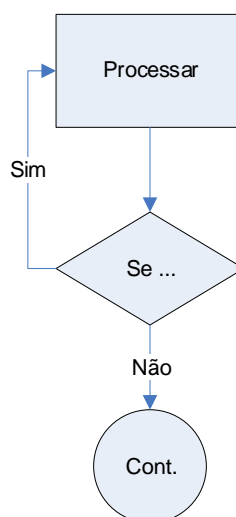
```
programa
{
    funcao inicio()
    {
        inteiro j = 0

        enquanto (j <= 100)
        {
            j = j + 1
        }
    }
}
```

## 7.2. FACA ... ENQUANTO

Neste caso, executa-se primeiro o bloco de operações e somente depois é realizado o teste de condição. Se a condição for verdadeira, o fluxo do programa continua normalmente. Caso contrário, são processados novamente os comandos antes do teste da condição.

Em diagrama de Bloco (Figura 6-7).



**Figura 6-7 Diagrama de Blocos – faça ... enquanto.**

A Figura 6-8 mostra um exemplo em Diagrama de Blocos. Em seguida, temos o algoritmo.

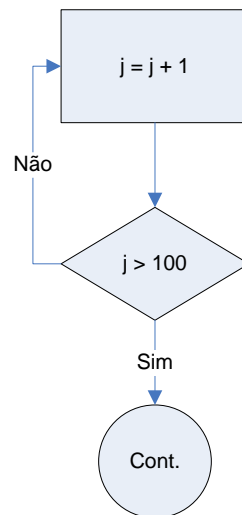


Figura 6-8 Diagrama de Blocos – Exemplo: faça...enquanto.

```
programa
{
    funcao inicio()
    {
        inteiro j = 0

        faca
        {
            j = j + 1
        } enquanto (j <= 100)
    }
}
```

### 7.3. PARA

O comando “para” possui a característica de ser executado uma quantidade fixa de vezes de acordo com um incremento fixo. No caso do comando na forma padrão, esse incremento é 1 (um). Em seguida temos o algoritmo:



```
programa
{
    funcao inicio()
    {
        inteiro j

        para (j = 0; j <= 100; j++)
        {

        }
    }
}
```



Caso você queira utilizar o comando “para” com um incremento diferente de 1 (um), por exemplo dois (dois), você deve fazer da seguinte forma:  
**para (j=0; j <= 100; j=j+2)**



***Vamos resumir o conteúdo visto neste capítulo:***  
Neste capítulo, aprendemos sobre comandos que nos ajudarão (e muito!) a programar:  
**Comandos de Repetição:**

- enquanto
- faça...enquanto
- para



## Atividades

1. Elabore um algoritmo que gere e escreve os números ímpares lidos entre 100 e 200.
2. Construa um algoritmo que leia 500 valores inteiros e positivos e:
  - Encontre o maior valor
  - Encontre o menor valor
  - Calcule a média dos números lidos





## Capítulo 8 - ESTRUTURA DE DADOS HOMOGÊNEAS - VETOR



Olá Aluno,

Este é o capítulo em que vamos aprender sobre estruturas de dados; mais precisamente sobre Vetores.

Em muitos casos, os Vetores são alternativas interessantes para se resolver alguns problemas. Além deles, existem muitas outras estruturas de armazenamentos de dados, como por exemplo: matrizes, listas, filas, etc.

Todavia, como o foco de nossa disciplina tem cunho introdutório, vamos trabalhar apenas com Vetores.

É de suma importância entendermos bem a programação com Vetores, pois isso facilitará muito o nosso aprendizado quanto às outras possibilidades de armazenamento de dados.

Bom estudo!

### 8.1. ESTRUTURAS DE DADOS

Já sabemos que variáveis correspondem a posições de memória, às quais o programador tem acesso, através de algum algoritmo, visando atingir resultados desejados.

Uma variável passa a existir a partir de sua declaração, quando, então, lhe é associado um nome ou identificador, que é a identificação da posição de memória que a variável representa.

Analisaremos no exemplo a seguir a necessidade de se utilizarem estruturas de dados.

Exemplo:

Suponhamos que seja requisitado um algoritmo para calcular a média das notas de 5 alunos. Nesse contexto, o algoritmo ficaria assim:



```
programa
{
    funcao inicio()
    {
        inteiro I
        real NOTA, SOMA, MEDIA
        SOMA = 0
        para (I=1; I<=5; I++){
            escreva("Digite a nota:")
            leia(NOTA)
            SOMA = SOMA + NOTA
        }
        MEDIA = SOMA / 5.0
        escreva ("A média é:", MEDIA)
    }
}
```

Suponhamos, agora, que se deseja saber quantos alunos tiveram nota acima da média. É fácil ver que, para saber o número de alunos com nota acima da média é necessário saber as notas e a média de cada um deles. Daí, existem duas abordagens possíveis:

- Abordagem 1: Fazer dois algoritmos: um que lê as notas e calcula a média e outro que leia a média e novamente todas as notas para que as comparações pudessem ser feitas.
- Abordagem 2: Fazer um único algoritmo e armazenar cada uma das notas numa variável; calcular a média dessas notas e finalmente calcular a quantidade de alunos com nota superior a média.

**Abordagem 1 (Abordagem de dois algoritmos):**

O primeiro algoritmo já estaria implementado: o algoritmo do cálculo da média. O segundo algoritmo ficaria da seguinte forma:

```
programa //Abordagem 1
{
    funcao inicio()
    {
        inteiro I, QUANTIDADE = 0
        real NOTA, MEDIA

        escreva("Digite a média:")
        leia(MEDIA)
        para (I=1; I<=5; I++){
            escreva("Digite a nota:")
            leia(NOTA)
            se (NOTA > MEDIA) {
                QUANTIDADE = QUANTIDADE + 1
            }
        }
        escreva ("Quantidade: ", QUANTIDADE)
    }
}
```



## Abordagem 2 (Abordagem de um algoritmo):

Este algoritmo realiza o armazenamento das notas em variáveis:

```
programa //Abordagem 2
{
    funcao inicio()
    {
        real MEDIA
        real NOTA1, NOTA2, NOTA3, NOTA4, NOTAS5
        inteiro QUANTIDADE = 0
        escreva("Digite a nota 1:")
        leia(NOTA1)
        escreva("Digite a nota 2:")
        leia(NOTA2)
        escreva("Digite a nota 3:")
        leia(NOTA3)
        escreva("Digite a nota 4:")
        leia(NOTA4)
        escreva("Digite a nota 5:")
        leia(NOTAS5)
        MEDIA = (NOTA1+NOTA2+NOTA3+NOTA4+NOTAS5)/5
        escreva("Média: ", MEDIA)
        se (NOTA1 > MEDIA){
            QUANTIDADE = QUANTIDADE + 1
        }
        se (NOTA2 > MEDIA){
            QUANTIDADE = QUANTIDADE + 1
        }
        se (NOTA3 > MEDIA){
            QUANTIDADE = QUANTIDADE + 1
        }
        se (NOTA4 > MEDIA){
            QUANTIDADE = QUANTIDADE + 1
        }
        se (NOTAS5 > MEDIA){
            QUANTIDADE = QUANTIDADE + 1
        }
        escreva("Quantidade: ", QUANTIDADE)
    }
}
```



Note que nenhuma das soluções é completamente adequada, pois:

- Abordagem 1: implica na releitura das notas dos alunos. Supondo que essas notas fossem informadas via teclado, isso seria bastante desagradável para quem utiliza esse algoritmo.
- Abordagem 2: não traz problemas para o usuário, mas implica em termos uma quantidade possivelmente grande de variáveis que representam o mesmo tipo de informação. Imaginemos que não fossem 5 alunos e sim 5000; a elaboração desse algoritmo seria impraticável.

É nesse contexto que as estruturas de dados são apresentadas, especificamente as *variáveis compostas homogêneas* (vetores).

## 8.2. VARIÁVEIS COMPOSTAS HOMOGÊNEAS



**Variáveis compostas homogêneas** correspondem a posições de memória, identificadas por um mesmo identificador, individualizadas por índices e cujo conteúdo é do mesmo tipo.

No exemplo do cálculo da quantidade de alunos com nota acima da média, podemos representar o conjunto das 5 notas dos alunos por uma variável do tipo composta homogênea. Chamemos essa variável de **NOTA**. Tal variável é homogênea porque todos os seus elementos são do tipo real, e é composta porque é um conjunto de espaços de memória e não apenas um. A referência ao *n*-ésimo elemento do conjunto será indicado pela notação **NOTA[n]**, onde “n” é um número inteiro ou uma variável numérica contendo um valor inteiro.

### Exemplo 1:

Suponhamos que a variável **NOTA** contivesse os seguintes valores:

10	20	30	40	50
----	----	----	----	----



NOTA[3] estaria referenciando o terceiro elemento do conjunto cujo conteúdo é 30.

### Exemplo 2:

Voltemos ao exemplo do cálculo da quantidade de alunos com nota acima da média; mas suponhamos que sejam 250 alunos. O algoritmo então ficaria da seguinte forma (não serão apresentadas as declarações de variáveis):

```
programa
{
    funcao inicio()
    {
        //Foram omitidas as declarações

        //Inicializando SOMA e QUANTIDADE
        SOMA = 0
        QUANTIDADE = 0
        //Aplicando o comando de repetição
        para (I=1; I<=250; I++){
            escreva("Digite a nota:")
            leia ( NOTA[I] )
            //Acumulando a NOTA lida
            SOMA = SOMA + NOTA[I]
        }
        //Calculando a média
        MEDIA = SOMA / 250
        para (I=1; I<=250; I++){
            se (NOTA[I] > MEDIA){
                //Acumulando a Quantidade
                QUANTIDADE = QUANTIDADE + 1
            }
        }
        escreva ("Quantidade: ", QUANTIDADE)
    }
}
```



### 8.3. VARIÁVEIS COMPOSTAS UNIDIMENSIONAIS – VETORES



**Vetor** é um conjunto de espaços de memória referenciados por um mesmo nome e que necessitam de apenas um índice para que seus elementos sejam endereçados.

Graficamente um vetor pode ser representado da seguinte forma:



#### Exemplos de Vetores:

Estados do Sudeste:

ES	MG	RJ	SP
----	----	----	----

Alguns Municípios do Sul do Espírito Santo:

Cachoeiro de Itapemirim	Jerônimo Monteiro	Alegre	Castelo	Muqui
-------------------------	-------------------	--------	---------	-------

#### 8.3.1. Declaração de Vetores

t identificador [ls]

Onde:

- identificador: nome associado a variável que se deseja declarar.
- ls: é o limite superior do intervalo de variação dos índices.
- t: tipo dos componentes do vetor (inteiro, real, cadeia, logico).



As regras de declaração de identificadores das variáveis compostas (vetores, por exemplo) são as mesmas das variáveis simples.



### Exemplo 1:

**real** NOTA[ 250 ]

### Exemplo 2:

Será mostrado novamente o exemplo do cálculo da quantidade de alunos com nota acima da média, também supondo 250 alunos.

```
programa
{
    funcao inicio()
    {
        inteiro I, QUANTIDADE
        real SOMA, MEDIA, NOTA[250]

        //Inicializando SOMA e QUANTIDADE
        SOMA = 0
        QUANTIDADE = 0
        // Aplicando o comando de repetição
        para (I=1; I<=250; I++){
            escreva("Digite a nota:")
            leia ( NOTA[I] )
            //Acumulando a NOTA lida
            SOMA = SOMA + NOTA[I]
        }
        //Calculando a média
        MEDIA = SOMA / 250
        para (I=1; I<=250; I++){
            se (NOTA[I] > MEDIA){
                //Acumulando a Quantidade
                QUANTIDADE = QUANTIDADE + 1
            }
        }
        escreva ("Quantidade: ", QUANTIDADE)
    }
}
```





## Atividades

1. Faça um algoritmo que leia 200 números e verifique quantos deles são iguais a 30. Se existirem, escrever as posições onde eles estão armazenados.
2. Faça um algoritmo que leia  $n$  números ( $n < 200$ ) e verifique quantos deles são iguais a um número  $k$ . Se existirem, escrever as posições onde eles estão armazenados.
3. Implemente um algoritmo que leia 200 números. Esse algoritmo deve escrever quantos deles são maiores que um número  $k_i$  e menores que um número  $k_j$ . ( $k_i$  e  $k_j$  também lidos).
4. O Instituto de Ciências Exatas da UFMG deseja saber se existem alunos cursando, simultaneamente, as disciplinas “Programação de Computadores” e “Cálculo Numérico”. Existem disponíveis na unidade de entrada os números de matrícula dos alunos de “Programação de Computadores” (no máximo 150 alunos) e de “Cálculo Numérico” (no máximo 220 alunos). Cada conjunto dos números de matrícula dos alunos de uma disciplina tem a matrícula fictícia 9999 no final.

Formular um algoritmo que imprima o número de matrícula dos alunos que estão cursando ambas as disciplinas simultaneamente.

Trata-se, então, da verificação da ocorrência de um elemento de um conjunto em outro conjunto. Assim, após a leitura dos dados, poderiam estar montadas as seguintes variáveis compostas unidimensionais  $PC$  e  $CN$  contendo, respectivamente, os números de matrícula dos alunos que estão cursando: “Programação de Computadores” e “Cálculo Numérico”.



## Capítulo 9 - MODULARIZAÇÃO



Prezado Aluno,

À medida que estamos avançando nos conceitos de Lógica de Programação, somos preparados para resolução de problemas cada vez mais complexos.

No geral, problemas complexos exigem algoritmos complexos, mas sempre é possível dividir um problema grande em problemas menores. Dessa forma, cada parte menor tem um algoritmo mais simples.

Neste capítulo, aprenderemos a técnica de programação *Dividir para Conquistar*, isto é, partindo da divisão de um problema complexo em vários subprogramas, ao conquistar o objetivo em cada um dos subprogramas, o programador estará caminhando rumo à Solução Total do problema original.

Através desse método, desaparece o problema complicado e aparecem vários subprogramas descomplicados, tornando assim a solução mais simples, direta e objetiva.

Sucesso a todos! E vamos aos estudos!

### 9.1. INTRODUÇÃO

No fim da década de 60, um conjunto de problemas no desenvolvimento de programas levou os países desenvolvidos à chamada “crise do software”. Isso ocorreu devido a uma ausência de metodologia no desenvolvimento dos softwares. Para suprir essa carência, foi criada a programação estruturada, a fim de organizar os métodos e diminuir os custos de desenvolvimento de softwares.

Uma das principais armas da programação estruturada no desenvolvimento de softwares é a modularização, pois permite um maior reaproveitamento de funcionalidades, diminuindo o tempo e os custos de desenvolvimento.



A **modularização** é a técnica de subdividir um algoritmo em módulos menores que podem ser reaproveitados em outros algoritmos.

## 9.2. PROCEDIMENTOS E FUNÇÕES

Os procedimentos e funções possuem três objetivos básicos:

- Reaproveitamento de código: quando certa sequência de comandos é usada em vários locais.
- Dividir o algoritmo em partes logicamente coerentes: gerar subalgoritmos.
- Aumentar a legibilidade do algoritmo.

### 9.2.1. Funções



Uma **função** é um trecho de código que pode ser utilizado em vários algoritmos, e que retorna um valor.

A função é fundamental para a manutenibilidade dos algoritmos uma vez que, dessa forma, se for necessário fazer uma manutenção em um procedimento, essa manutenção automaticamente será aplicada a todos os algoritmos que a utilizam.

#### Declaração de Funções

Uma função deve ser declarada da seguinte forma:

```
funcao <tipo do retorno> <nome da função>(  
<declaração de parâmetros> ){  
    < declaração de variáveis locais >  
    < implementação da funcao >  
}
```

#### Exemplo:

```
funcao real achar_menor(real num1, real num2){  
    se (num1 > num2){  
        retorne num2  
    } senao{  
        retorne num1  
    }  
}
```



## 9.2.2. Passagem de Parâmetros por Valor e Passagem de Parâmetros por Referência



**Passagem por Valor:** é a declaração de parâmetros de forma que eles não terão seus valores alterados após a execução do procedimento ou da função.



**Passagem por Referência:** é a declaração de parâmetros de forma que eles poderão ter seus valores alterados após a execução do procedimento ou função. Um parâmetro por referência é precedido pelo símbolo &.

### Passagem por Valor

```
funcao real achar_menor(real num1, real num2){  
    se (num1 > num2){  
        retorne num2  
    } senao{  
        retorne num1  
    }  
}
```

### Passagem por Referência

```
funcao inteiro achar_menor(real num1, real  
num2, real &menor){  
    se (num1 > num2){  
        menor = num2  
        retorne 1  
    } senao{  
        se (num1 < num2){  
            menor = num1  
            retorne 1  
        } senao{  
            //Os números são iguais  
            menor = 0  
            retorne -1  
        }  
    }  
}
```



### 9.2.3. Procedimentos



Um **procedimento** é uma função que não retorna valores.

#### Declaração de Procedimentos

Um procedimento deve ser declarado na seguinte forma:

```
funcao <nome do procedimento>( <parâmetros> ){  
    < declaração de variáveis locais >  
    < implementação do procedimento >  
}
```

#### Exemplo:

```
funcao achar_menor(real num1, real num2, real  
&menor) {  
    se (num1 > num2) {  
        menor = num2  
    } senao {  
        menor = num1  
    }  
}
```

### 9.2.4. Uso de Funções e Procedimentos

Para se utilizar uma função ou procedimento, basta fazer da seguinte forma:



**Exemplo:** O seguinte algoritmo determina o menor de 2 números

```
programa
{
    funcao inicio()
    {
        real numero1, numero2, menor
        escreva("Digite o número 1: ")
        leia(numero1)
        escreva("Digite o número 2: ")
        leia(numero2)
        menor = achar_menor(numero1, numero2)
        escreva("O menor número é: " + menor)
    }

    funcao real achar_menor(real num1, real num2){
        se (num1 > num2){
            retorne num2
        }senao{
            retorne num1
        }
    }
}
```



É importante ressaltar que variáveis do tipo vetor também podem ser passadas por parâmetro para funções ou procedimentos. Para tal, basta que você digite apenas o nome do vetor no algoritmo principal.



## Atividades

1. Escreva uma função que calcule a distância entre dois pontos, sabendo que a distância entre dois pontos é dada pela expressão:

$$\text{DISTANCIA} = ((X2 - X1)^2 + (Y2 - Y1)^2)^{(1/2)}$$

Onde: O primeiro ponto é dado por ( X1, Y1 ) e o segundo ponto é dado por ( X2, Y2 )



2. Sabendo-se que um triângulo é formado por três pontos:

A = ( X1, Y1 )

B = ( X2, Y2 )

C = ( X3, Y3 )

E que o perímetro de um triângulo é a soma das distâncias dos lados do triângulo, podendo ser escrito da seguinte forma:

Perímetro = DISTANCIA AB + DISTANCIA AC + DISTANCIA BC

Faça um algoritmo que calcule o perímetro de um triângulo.

3. Faça uma função em que, dados dois valores (X, Y), retorne  $X^Y$ ; entretanto não é permitido usar a função de potência.

*Sugestão: Use o operador de multiplicação.*

4. Faça uma função em que, dados dois valores ( X, Y ), retorne  $X * Y$ ; entretanto não é permitido usar o operador de multiplicação. ( \* ).

*Sugestão: Use o operador de soma.*

5. Refaça a função do exercício 3) utilizando a função de multiplicação do exercício 4).

6. Faça uma função para o operador de raiz  $X^{(1/Y)}$

7. Implemente um algoritmo de calculadora que faça as operações de:

- Adição
- Subtração
- Multiplicação ( usando a função do exercício 4 )
- Divisão
- Potenciação ( usando a função do exercício 5 )
- Radiciação ( usando a função do exercício 6 )