



# ALGORITMOS

EDIÇÃO Nº 1 - 2007



PROF. CLÁUDIA WERLICH

---

**Apoio**



**Gestão e Execução**



**Conteúdo e Tecnologia**



## SUMÁRIO

<b>AULA 1 - NOÇÕES BÁSICAS SOBRE ALGORITMOS.....</b>	<b>7</b>
<b>AULA 2 - VARIÁVEIS, OPERADORES E CONSTANTES .....</b>	<b>14</b>
<b>AULA 3 - ALGORITMO SEQUÊNCIAL .....</b>	<b>20</b>
<b>AULA 4 - ESTRUTURA DE SELEÇÃO .....</b>	<b>29</b>
<b>AULA 5 - SELEÇÃO COMPOSTA .....</b>	<b>39</b>
<b>AULA 6 - SELEÇÃO MÚLTIPLA ESCOLHA .....</b>	<b>47</b>
<b>AULA 7 - ESTRUTURA DE REPETIÇÃO .....</b>	<b>53</b>
<b>AULA 8 - VETORES .....</b>	<b>67</b>

## Apresentação



Este livro-texto contém a disciplina de **Algoritmos**.

Este material disponibiliza aos alunos do EAD técnicas para a construção de algoritmos, pois criar algoritmos é essencial para o desenvolvimento e aperfeiçoamento da lógica do programador.

Para sua melhor compreensão, o livro está estruturado em 8 capítulos, com vários exemplos e exercícios.

Lembre-se de que a sua passagem por esta disciplina será também acompanhado pelo Sistema de Ensino **Tupy Virtual**, seja por correio postal, fax, telefone, e-mail ou Ambiente Virtual de Aprendizagem.

Sempre entre em contato conosco quando surgir alguma dúvida ou dificuldade.

Toda a equipe terá a maior alegria em atendê-lo, pois a sua aquisição de conhecimento nessa jornada é o nosso maior objetivo.

Acredite no seu sucesso e bons momentos de estudo!

Equipe Tupy Virtual.

## Carta do Professor



“O grande amor nasce, do grande conhecimento da coisa amada”. (Leonardo Da Vinci)

Caro aluno,

No decorrer dos capítulos, você aprenderá a criar algoritmos.

Nesta disciplina, as principais estruturas básicas utilizadas na programação de computadores serão demonstradas, visando sempre ao aperfeiçoamento de sua lógica de programação, que é essencial à sua qualificação profissional.

Iniciantes no mundo da programação devem resolver algoritmos. Alguns alunos conseguem resolver os problemas rapidamente. Outros, porém, podem ter mais dificuldades e se este for o seu caso, tenho uma sugestão: repita, repita e repita os exemplos e exercícios.

Todos os algoritmos apresentados neste livro possuem uma versão na linguagem de programação C++, para que você possa digitar e testar. Dessa forma, você conseguirá aprimorar a sua lógica, facilitando o aprendizado das próximas disciplinas.

Vamos agora para um novo desafio: a programação de computadores!

Professora Cláudia Werlich

## Cronograma de Estudo



Acompanhe no cronograma os conteúdos das aulas e atualize as possíveis datas de realização de aprendizagem e avaliações.

Semana	Carga horária	Aula	Data/ Avaliação
1	1	Noções Básicas de Algoritmos	_/_ a _/_
	2	Variáveis, Operadores e Constantes	_/_ a _/_
	2	Algoritmo Sequencial	_/_ a _/_
2	2	Algoritmo Seleção Simples	_/_ a _/_
	2	Algoritmo Seleção Composta	_/_ a _/_
	2	Algoritmo Seleção Múltipla Escolha	_/_ a _/_
3	2	Algoritmo de Repetição Para..Faça	_/_
	2	Algoritmo de Repetição Repita ... Até	_/_
	2	Algoritmo de Repetição Enquanto...Faça	_/_
	3	Algoritmo com Vetores	_/_

## Plano de Estudo



### Ementa

### Objetivos da Disciplina

#### **Geral**

Utilizar ambientes de programação com edição, compilação, testes e geração de códigos integrados, inclusive aqueles que acompanham programas aplicativos, a partir de especificações.

#### **Específicos**

Desenvolver programas e a lógica; definir algoritmos, estabelecendo natureza, origem e estrutura de dados; utilizar ambientes de programação com edição, compilação, testes e geração de códigos integrados, inclusive aqueles que acompanham programas aplicativos, a partir de especificações; atender à necessidade do usuário em sua estação de trabalho.

Carga Horária: 40 horas/aula.

## Aula 1

# NOÇÕES BÁSICAS SOBRE ALGORITMOS

### Objetivos da aula



Ao final desta aula, você deverá ser capaz de:

- Estabelecer a relação entre algoritmo e linguagem de programação, conhecendo os principais conceitos sobre o assunto;
- Conhecer a estrutura básica do algoritmo.

### Conteúdos da aula



Acompanhe os assuntos desta aula, se preferir, após o término, assinale o conteúdo já estudado.

- ☐ Definição de algoritmo;
- ☐ Conceitos básicos do ambiente de programação;
- ☐ Estrutura básica do algoritmo;
- ☐ Linguagem de programação;
- ☐ Exercícios propostos.



Prezado aluno, seja bem vindo a nossa primeira aula!

Iremos dar início a disciplina de algoritmos começando pelas noções básicas sobre algoritmos.

Boa aula!

## 1. NOÇÕES BÁSICAS SOBRE ALGORITMOS



Segundo Forbellone (2000, p.03), algoritmo pode ser definido como uma sequência de passos que visam atingir um objetivo bem definido. Quando elaboramos um algoritmo, devemos especificar ações claras e precisas que possam resultar na solução de um problema proposto.

Podemos criar algoritmos de qualquer situação. Uma receita de bolo é um exemplo muito claro de algoritmo. Primeiramente separamos os ingredientes e após devemos, passo-a-passo, juntar os ingredientes. No caso do bolo, a sequência de passos é fundamental. Imagine: se primeiro misturarmos o leite com o fermento. Após, colocamos o trigo, ovos e por último o açúcar. Qual seria o resultado dessa mistura? Um bolo fofinho com certeza não seria! O sucesso do bolo está justamente na sequência de passos corretos. Surge então uma palavra chave: **lógica**.

A lógica está na correta sequência de passos, que deve ser seguida para ter um objetivo específico. No exemplo do bolo, a lógica é inserir o fermento por último, senão o bolo não irá crescer.

Outro exemplo que pode ser analisado: fazer um ovo frito. Os passos a serem feitos poderiam ser:

- Aquecer a frigideira;
- Adicionar óleo para fritar o ovo;
- Quebrar o ovo;
- Colocar o ovo na frigideira;
- Esperar fritar;
- Virar o ovo para fritar do outro lado;
- Servir o ovo frito.

Os itens listados para fritar o ovo, já dão uma idéia de todo o processo básico de fritar um ovo. Porém, outra pessoa que fosse fazer a mesma atividade, poderia acrescentar mais opções. Como por exemplo: verificar se existe um ovo para ser frito, adicionar sal ou utilizar algum utensílio para virar o ovo. O grau de detalhamento pode ou não ser indispensável no desenvolvimento de um algoritmo, tudo depende da situação que deverá ser muito bem observada pelo programador.



A experiência do programador reflete diretamente em uma lógica mais apurada, no desenvolvimento de um algoritmo. **Para conseguir mais experiência e uma lógica mais refinada só há um caminho: fazer algoritmos!**

Um algoritmo, se digitado no computador, não irá funcionar. Cada pessoa pode elaborar o seu próprio algoritmo. Com a finalidade de padronizar, devemos utilizar uma linguagem denominada pseudocódigo, que nada mais é do que utilizarmos regras para a criação dos algoritmos. Regras essas, que não são rígidas, pois, conforme mencionado, o algoritmo servirá somente para o apoio no aprendizado e no aperfeiçoamento da lógica dos programadores.

Os algoritmos criados com o pseudocódigo devem ser independentes da linguagem de programação. Para isso, devemos converter o algoritmo para uma linguagem de programação. O objetivo de criarmos algoritmos é a base de conhecimento de outras linguagens de programação. Será a nossa referência para aprendermos outras linguagens de computadores.

O algoritmo é flexível e, dessa forma, estaremos utilizando algumas regras voltadas à linguagem de programação C++, sem perder as principais características de sua finalidade, que é a prática da programação.

## 1.1 CONCEITOS BÁSICOS

Alguns termos serão freqüentemente utilizados nesse livro-texto e nos outros que estarão por vir, para isso é fundamental que alguns conceitos estejam bem claros em nossas mentes, são eles:

**Linguagem de Programação:** comandos específicos utilizados para o ser humano programar o computador, para que seja executada alguma tarefa específica. As linguagens de programação podem ser divididas em duas categorias: Linguagens de Baixo Nível e Linguagens de Alto Nível.

**Linguagem de Baixo Nível:** são linguagens voltadas para a máquina, são escritas usando as instruções do microprocessador do computador.

**Linguagem de Alto Nível:** utilizam uma sintaxe, ou seja, comandos que são facilmente interpretados pelos humanos. Para que o computador as compreenda, precisamos utilizar: os compiladores ou interpretadores.

**Compiladores:** fazem a tradução de todas as instruções do programa-fonte, criando um programa executável.

**Interpretadores:** fazem a tradução de cada instrução do programa-fonte, executando-a dentro de um ambiente de programação.

**Programa-fonte:** é a passagem ou a tradução do algoritmo para uma linguagem que o computador compreenda. Existem diversas linguagens de programação como: **C, C++, Delphi, Cobol, Java**, etc... Cada linguagem possui características particulares, como símbolos e palavras específicos que devem ser rigorosamente respeitados.

**Compilação:** Após a passagem do algoritmo para o programa-fonte, devemos compilar o programa (usando os compiladores – no nosso caso estaremos utilizando a linguagem Turbo C++). O programa irá verificar se você cometeu erros ao transformar o algoritmo em um programa-fonte. Nessa fase, é muito comum a ocorrência de erros, por isso é essencial muita atenção na digitação. A maioria dos erros são comandos digitados errados.

**Programa objeto:** após a compilação, realizada pelo computador, é gerado um código que somente o computador entende: a linguagem de máquina. Nessa fase, o computador transforma todos os códigos digitados para uma linguagem binária (composta de 1 e 0), que é a linguagem que o computador entende.

**Programa executável:** após a realização do programa objeto, será criado um programa executável e este é o programa que você terá como resultado de seu algoritmo. Um programa executável não depende da linguagem de programação. Isso significa que o programa poderá funcionar independente de termos ou não o *software* que utilizamos para criar o programa fonte.

Há muitos outros termos que aparecerão no decorrer do livro e todos serão abordados no momento oportuno.

**Dica:** Os programas-fonte são fáceis de serem identificados. Sempre que você salvar o programa digitado, ele terá a extensão **.CPP**, assim que for feita a compilação, o programa passará a ter a extensão **.EXE**.

## 1.2 ESTRUTURA BÁSICA DO ALGORITMO

O que pretendemos neste livro-texto é criar algoritmos e fazer a conversão para uma linguagem de programação.

Para atingirmos tal objetivo, observe, na figura 1, o primeiro exemplo de um algoritmo – no **Algoritmo Soma\_de\_dois\_numeros**. A proposta é pedir dois números para que possam ser somados e, após, imprimir o resultado dessa operação.

```
Algoritmo Soma_de_dois_numeros;  
Variáveis numero1,numero2,soma: inteiro;  
Inicio  
  Escreva (" Informe um número: ");  
  Leia (numero1);  
  Escreva (" Informe outro número: ");  
  Leia (numero2);  
  soma = numero1 + numero2;  
  Escreva (" O resultado da soma dos números informados é :", soma);  
Fim.
```

Figura 1 - Algoritmo Soma\_de-dois\_numeros

No exemplo do Algoritmo Soma\_de\_dois\_numeros, podemos observar os seguintes itens que aparecem em todos os algoritmos:

Sempre iniciam com a palavra Algoritmo. O nome do algoritmo, que no exemplo é Soma\_de\_dois\_numeros, deve sempre começar com uma letra e nunca ter espaços entre as palavras e jamais deverá começar com número;

O nome do algoritmo deverá ter um significado de acordo com o objetivo do algoritmo, no caso do exemplo, como o objetivo é somar dois números, o nome mais apropriado é: Soma\_de\_dois\_numeros;

Ao final, as frases deverão sempre terminar com ponto e vírgula, com exceções que futuramente serão discutidas;

Todo algoritmo tem um início e um fim, que deve terminar com um ponto final;

O comando **Escreva ()**, irá imprimir na tela do computador tudo do que estiver entre as aspas.

O comando **Leia ()**, serve para que o computador receba um valor, que deverá ser inserido numa variável;

Todo programa possui variáveis que devem ser declaradas, no caso do exemplo, foram criadas três variáveis que armazenaram valores.

Uma dica importante: antes de começar a fazer um algoritmo, você deve analisar o que deverá ser feito. A leitura e a compreensão do enunciado é o ponto inicial da resolução do algoritmo. O ideal é a separar por partes. No caso da nossa proposta inicial, que é: pedir dois números para que possamos somá-los e depois, imprimir o resultado da operação. Separando por partes:

Primeiro, devemos pedir para que os dois números sejam informados;  
Segundo, devemos calcular a soma dos dois números e guardar o resultado;  
Terceiro, imprimir o resultado.

Separar e escrever por tópicos o que deve ser feito num programa ajudará na solução. O ideal é escrever num papel esses tópicos, mas com prática você irá fazer isso mentalmente com muita facilidade.

### 1.3 LINGUAGEM DE PROGRAMAÇÃO C++

Os algoritmos elaborados neste livro serão implementados, ou seja, convertidos para a linguagem de programação C++ (fala-se *C plus plus*). Utilizaremos para isso o software TC++ 3.0 da Borland, que está na biblioteca.

A linguagem de programação C++ originou-se da linguagem C, inventada por Dennis Ritchie. Essa linguagem, a C, ficou muito conhecida por se tratar de uma linguagem para programadores profissionais. Segundo Cordeiro (1998, p.10), por volta de 1985, Bjarne Stroustrup começou a desenvolver algumas extensões à linguagem C com o objetivo de prover suporte ao paradigma de orientação a objetos. Esse conjunto de extensões determinou o surgimento da linguagem C++, em 1986. O nome "C++" advém do fato de que esta linguagem é tida como uma extensão da linguagem C, onde o primeiro "+" representa as melhorias realizadas na linguagem C, e o segundo "+" representa o acréscimo de funcionalidades para suportar o paradigma de orientação a objetos.

**Exercícios propostos**

- 1) Defina algoritmo.
- 2) Qual a diferença entre um programa-fonte e um executável?
- 3) Defina linguagem de Programação.
- 4) Pesquise e descubra o nome de no mínimo cinco linguagens de programação para computadores.

## Aula 2

# VARIÁVEIS, OPERADORES E CONSTANTES



### Objetivos da aula

Ao final desta aula, você deverá ser capaz de:

- Conhecer a importância da declaração das variáveis nos algoritmos e programas, bem como a definição de um tipo para cada tipo de variável;
- Diferenciar os operadores: aritméticos, lógicos e relacionais;
- Estabelecer a importância da constante em algoritmos.



### Conteúdos da aula

Acompanhe os assuntos desta aula, se preferir, após o seu término, assinale o conteúdo já estudado.

- ☐ Definição de variáveis e de seus tipos básicos;
- ☐ Utilização de operadores em expressões;
- ☐ Exemplificação de constantes;
- ☐ Exercícios propostos.



Seja bem vindo a nossa segunda aula!

Iremos dar início ao assunto variáveis, operadores e constantes.

Boa aula!

# 2

Virtual  
Tupai

## 2. VARIÁVEIS, OPERADORES E CONSTANTES



O sucesso de um algoritmo ou programa depende da correta declaração de variáveis e constantes. A utilização de operadores é fundamental, se utilizados erroneamente, o programa pode apresentar o famoso: **erro de lógica**.

Erro de lógica significa: **o programa não tem erro nos comandos**, porém **apresenta um resultado inesperado ou errado**. A grande possibilidade do erro estará no uso errado dos operadores. Ressaltando assim a importância dessa aula.

### 2.1 VARIÁVEIS

Segundo ASCENCIO (1999, p.10), quando fazemos um programa, este recebe os dados que devem ser armazenados no computador para que possam ser utilizados no processamento e armazenado na memória do computador.

Um dado é classificado como variável quando tem a possibilidade de ser alterado em algum determinado momento do programa, conforme Forbellone (2000, p.17). **Toda variável deverá ser declarada antes de ser utilizada**. Na sua declaração, informaremos um **tipo** para ela. Um tipo significa “**informar**” **ao computador o que essa variável poderá receber e armazenar na memória do computador**. Muitos são os tipos de variáveis. Neste livro utilizaremos principalmente os tipos:

**Inteiros** - para valores inteiros (números positivos ou negativos - sem vírgulas);

**Caracter** - para valores que receberão somente um caracter (letra ou número de 0 a 9 ou um sinal);

**Real** - para valores decimais (números com vírgulas);

**Lógico** - para retornar os valores Falso ou Verdadeiro e, no caso de C++, retornar 1 ou 0.

Algumas regras na criação das variáveis:

Utilize nomes significativos para as variáveis, de preferência nomes curtos;

Nunca utilize nomes para variáveis começando com números, exemplo:

1numero, o certo é numero1 – sem espaços e nem acento. Não utilize caracteres especiais como: \* , @ ou {, etc...

Em C++, letras maiúsculas são diferentes de letras minúsculas, ou seja, **A** é diferente de **a**. Portanto, nos programas, evite utilizar nomes de variáveis com letras maiúsculas, isso só irá atrapalhar você.

## 2.2 TIPOS DE OPERADORES

Os operadores são utilizados nas expressões matemáticas, lógicas e relacionais. A maioria dos operadores soa igual nas linguagens de programação. Em C++, a linguagem que utilizaremos possui algumas diferenças. Observe com atenção as tabelas nos próximos tópicos. Você, com certeza, irá utilizar os operadores em todos os programas.

### 2.2.1 OPERADORES ARITMÉTICOS

Segundo Forbellone (2000, p.20), chamamos de operadores aritméticos o conjunto de símbolos que representa as operações básicas da matemática. Confira na Tabela 1, os operadores aritméticos.

**Tabela 1 – Operadores Aritméticos**

Algoritmo	C++	Função
+	+	Adição
-	-	Subtração
*	*	Multiplicação
/	/	Divisão
mod	%	Resto da divisão
div	/	Quociente da divisão



A utilização do operador **MOD** (em algoritmo) ou **%** (em C++) será da seguinte forma:

15 MOD 2, resulta em 1 – o MOD sempre irá retornar o resto da divisão que, no caso do exemplo: 15 dividido por 2, o resto desta divisão é 1 – ignorando assim o resultado da divisão que é 7,5.

O mesmo exemplo é aplicado ao operador % em C++. Nas linguagens de programação, deveremos utilizar o % somente para variáveis do tipo inteiro (isso será devidamente explicado no tópico sobre variáveis).

## 2.2.2 OPERADORES RELACIONAIS

Toda vez que precisamos comparar valores, utilizaremos os operadores relacionais, que estão na Tabela 2:

**Tabela 2 – Operadores Relacionais**

Algoritmo	C++	Função
>	>	Maior que
<	<	Menor que
>=	>=	Maior ou igual a
<=	<=	Menor ou igual a
<>	!=	Diferente de
=	==	Igual a

Os operadores relacionais serão utilizados praticamente em todos os programas. **Exemplicando**, observe o resultado: Verdadeiro ou Falso e poderemos ter as respostas se atribuímos valores para **X** e **Y**. O **X** receberá o valor de **2** e **Y** receberá o valor de **10**.

Então teremos: X=2 e Y=10.

Agora observe as comparações abaixo:

X > Y, resposta: falso  
X >= Y, resposta: falso  
X < Y, resposta: verdadeiro  
X <= Y, resposta: verdadeiro  
X = Y, resposta: falso  
X <> Y, resposta: verdadeiro

### 2.2.3 OPERADORES LÓGICOS

Segundo Mizrahi (1994, p.53), **operadores lógicos fazem comparações**. A diferença entre comparações lógicas e relacionais está na forma como os operadores avaliam seus operandos, esta avaliação resulta em verdadeiro ou falso, conforme poderá ser visto na Tabela 3.

**Tabela 3 – Operadores Lógicos**

Algoritmo	C++	Função
E	&&	Será verdadeiro o resultado somente se na comparação todos os valores forem verdadeiros
OR		Será verdadeiro o resultado se um dos valores na comparação for verdadeiro
Não	!	Será verdadeiro somente se a expressão for falsa

Da mesma forma que os operadores relacionais, os lógicos serão intensamente utilizados nos algoritmos de seleção.

## 2.3 CONSTANTES

A utilização de constantes nos programas é muito útil. Imagine uma situação onde você precisaria informar a todo o momento um valor específico como: 45,12453698. Este número imenso, a todo o momento deve ser digitado, facilitando a ocorrência de erro. Uma vez definida a variável como sendo uma constante, o seu valor não poderá ser modificado. Exemplos de variáveis:

Nome = “Claudia Werlich”

Pi= 3,14

**Dica:**

Quando houver uma expressão aritmética com parênteses. Sempre terão precedência os parênteses de dentro para fora.

Observe a expressão:

$(9 + ((10 * 2) + (14 - 10)))$  o resultado será igual a 33

**Exercícios propostos**

1. Analise as expressões aritméticas e dê o resultado:

a)  $X = (10 * ((100 - 90) - 10))$  – o valor de X é: \_\_\_\_\_

b)  $W = (14 + (13 * (13 - 8)))$  – o valor de W é \_\_\_\_\_

2. O valor de X é igual a 15 e o valor de W é igual a 23, agora analise expressões abaixo e informe se a expressão é falsa ou verdadeira.

1.  $(X < Y)$  a expressão é: \_\_\_\_\_

2.  $(Y > X)$  a expressão é: \_\_\_\_\_

3.  $(X = Y)$  a expressão é: \_\_\_\_\_

3. Indique o tipo da variável “Valor”, conforme o valor que a variável receber. Você deverá informar se a variável será do tipo: inteiro, real, caracter ou lógico.

a) Valor = “A” o tipo será: \_\_\_\_\_

b) Valor = 15,65 o tipo será: \_\_\_\_\_

c) Valor = 47 o tipo será: \_\_\_\_\_

d) Valor = 3 o tipo será: \_\_\_\_\_

**Aula 3****ALGORITMO SEQUENCIAL****Objetivo da aula**

Ao final desta aula, você deverá ser capaz de:

Criar algoritmos seqüenciais e convertê-los para a linguagem de programação C++.

**Conteúdos da aula**

Acompanhe os assuntos desta aula, se preferir, após o seu término, assinale o conteúdo já estudado.

- ☐ Exemplicando a estrutura do algoritmo;
- ☐ Conversão do algoritmo para a linguagem C++;
- ☐ Exercícios propostos.



Prezado aluno!

Chegamos na terceira aula. Agora você irá aprender como desenvolver algoritmos seqüenciais e converte-los para a linguagem C++.

Boa aula!

**3****Virtual Tutor**



### 3. ALGORITMO SEQUENCIAL

Relembrando: alguns princípios básicos sobre o algoritmo já foram passados na primeira aula.

Vamos observar na figura 2, o algoritmo 2 - **Soma\_de\_Dois\_Numeros**

```
Algoritmo Soma_de_dois_numeros;  
Variáveis numero1, numero2, soma: inteiro;  
Inicio  
  Escreva (" Informe um número: ");  
  Leia (numero1);  
  Escreva (" Informe outro número: ");  
  Leia (numero2);  
  soma = numero1 + numero2;  
  Escreva ("O resultado da soma dos números informados é :", soma);  
Fim.
```

Figura 2 - Algoritmo 2 - Soma\_de-dois\_numeros

Comentários fundamentais:

Primeiramente você deve fazer uma releitura do item 1.3, da pág. .

Os nomes das variáveis, se precisarem de números, devem ser usados **sem espaços**, como no caso: numero1 e numero2.

Observe que, na hora de somar os numero1 e numero2, utilizamos o sinal de **igual**, que **não representa igualdade. Representa a atribuição**, ou seja, a variável soma “recebe” o valor da soma de numero1 e numero2.

Definimos três variáveis: soma, numero1 e numero2. Todas são do tipo inteiro, pois somente queremos utilizar números sem a parte decimal, sem vírgulas.

No último comando Escreva (), observe que o que está dentro das aspas: “ O resultado da soma dos números informados é: “ – observe que agora vem a vírgula para separar o que será escrito na tela do computador e a variável. Se, porventura você esquecer e deixar as aspas em toda a frase, nenhum valor será impresso, aparecerá na tela a palavra soma. Se isso acontecer, você já sabe: errou algo importante: o lugar das aspas.

Num terceiro algoritmo estaremos utilizando tipos de variáveis diferentes, conforme demonstra a figura 3.

```
Algoritmo Variáveis_diferentes;  
Variáveis idade: inteiro;  
           nome: caracter;  
           nota1, nota2, media: real;  
  
Início  
  Escreva (" Informe o seu nome: ");  
  Leia (nome);  
  Escreva (" Informe a sua idade: ");  
  Leia (idade);  
  Escreva (" Informe a sua primeira nota: ");  
  Leia (nota1);  
  Escreva (" Informe a sua segunda nota: ");  
  Leia (nota2);  
  media = nota1 + nota2;  
  Escreva ( Nome," Você tem: " , idade, " anos e sua média é: " , media);  
Fim.
```

**Figura 3 – Algoritmo 3 - Variáveis\_diferentes**

Comentários importantes sobre o Algoritmo:

Variáveis do mesmo tipo podem ser declaradas juntas somente se forem separadas por vírgula, como no caso: nota1, nota2, média: real; o ponto e vírgula deve sempre estar no final.

Nunca utilize a mesma linha para declarar variáveis de tipos diferentes. Isso evitará futuros problemas. Mas, se forem do mesmo tipo não há problemas, desde que separados por vírgulas.

Observe o último comando Escreva (), dentro dos parênteses toda a vez que você deseja imprimir o conteúdo da variável, as aspas não podem ser usadas. E a variável deve estar entre vírgulas, um exemplo seria a variável idade, que está entre as vírgulas.

### 3.1 CONVERSÃO DO ALGORITMO PARA O C++

Transformar o algoritmo para uma linguagem de programação requer muita atenção de sua parte. Os detalhes são muitos. E, pior, qualquer engano ou esquecimento o programa simplesmente não irá funcionar.

Na biblioteca você irá encontrar o programa TC ++, junto haverá um roteiro para utilizar a ferramenta adequadamente. Alguns detalhes sempre devem ser muito observados:

Na hora que você for salvar o programa, salve-o com somente um nome e com a extensão **.CPP** que indicará que o programa é do C++. Por exemplo: programa1.cpp ou exemplo.cpp ou teste.cpp .

Quando for realizada a compilação, podem aparecer diversas mensagens. Se apareceu uma ou mais mensagens, isso ocorreu porque há erros. O programa somente irá funcionar se todos os erros forem corrigidos.

Praticamente, a maioria dos erros na compilação é devido a erros de digitação. Tenha muito cuidado durante a digitação, qualquer deslize provocará erros no programa.

Antes de compilar e testar o programa, tenha o hábito de salvá-lo. Ocorrem muitos erros durante o teste do programa que podem travar o TC ++ e, com isso, você poderá perder o programa que fez, pois não salvou. Acredite isso ocorre com muita frequência.

Um exemplo de programa em C++ pode ser observado no programa Exemplo.cpp, conforme mostra a figura 4.

```
// Exemplo.cpp
#include <iostream.h>
#include <conio.h>
void main( )
{
    int numero1, numero2, soma;
    cout<< " \n Informe o primeiro número:" ;
    cin>> numero1;
    cout<< " \n Informe o segundo número:" ;
```

```
cin>> numero2;  
soma = numero1 + numero2;  
cout<< " A soma dos dois números é: " << soma;  
getch( );  
}
```

Figura 4 - Programa Exemplo.cpp

Comentários sobre o primeiro programa em C++:

Todos os comandos da sintaxe devem ser digitados em letras minúsculas.

Após cada comando, deverá ser usado o ponto e vírgula. Ele indica que o comando terminou e o computador deve ler o próximo comando.

Utilizamos bibliotecas, que possuem diversas funções já definidas. Para chamarmos uma biblioteca devemos utilizar: **#include** <nome da biblioteca> .

Todos os programas que faremos terão as duas bibliotecas: **#include <iostream.h>** e **#include <conio.h>** . A primeira possui as funções básicas de leitura e escrita e a segunda possui recursos de tratamento de tela. Num programa, poderemos utilizar muitas bibliotecas.

O comando **cout <<** (lemos: c out) é equivalente ao comando Escreva () do algoritmo, ao invés de parênteses, precisamos utilizar o **<<** , assim como no algoritmo, as aspas devem ser utilizadas quando queremos escrever algo na tela do computador. Sempre que desejar imprimir uma variável, você deve fazer como no exemplo: **cout<< "A soma dos dois números é: " << soma;**

No comando **cout<<** entre as aspas utilizamos o **\n** isso significa: pular a linha. Se você não utilizar o **\n** tudo na tela irá ficar amontoadado. Podemos utilizar vários **\n**, pulando assim várias linhas, a única ressalva é que devem ser usadas no **cout** e entre aspas.

O comando **cin>>** (lemos: c in) é equivalente ao comando Leia() do algoritmo. Poderemos ler várias variáveis, veja: **cin>> n1 >> n2 >> n3;**

A função **main ()** marca o ponto inicial do programa. Só poderá haver uma função **main ()** no programa. O tipo **void**, indica que não haverá retorno de valor para a função **main ()**. Lembre-se: escreva sempre com letra minúscula. As chaves indicam o Início e o Fim.

A função **getch()** serve para o programa esperar algo a ser digitado. Se não houver uma parada, o programa é executado de forma tão rápida que você



nem irá ver o resultado. Usando o **getch()**, você fará com que o programa pare até você digitar algo.

Para a declaração das variáveis, no C++, o tipo sempre é colocado no início, sem os dois pontos, exemplo: **int** numero1.

Para comentar algum item, demos usar: **//** e tudo que for escrito após **//** será ignorado pelo compilador.

Na tabela 4, podemos encontrar uma comparação das variáveis e de comandos entre algoritmos e C++. Observe que há alguns tipos de variáveis que serão usados mais adiantes.

**Tabela 4 – Comparação de Comandos**

<b>Algoritmo</b>	<b>C++</b>	<b>Função</b>
Leia ( )	cin >>	Leitura de valores
Escreva ( )	cout <<	Impressão na tela
inteiro	int	Variáveis do tipo inteiro
real	float	Variáveis do tipo decimal
caracter	char	Variáveis do tipo caracteres
Início Fim	{ }	Inicia e termina um bloco

### 3.2 EXEMPLIFICANDO ALGORITMOS SEQUENCIAIS

A característica básica desses programas é que eles começam e terminam sem permitir desvio ou possibilidade do usuário poder escolher algum caminho.

É importante fazer uma análise prévia antes de começarmos a fazer o algoritmo. Nesta análise devemos ler com cuidado o que é pedido no enunciado. Devemos ter claramente os itens abaixo:

- ☞ **Definição do resultado:** O que realmente precisamos fazer no programa?
- ☞ **Os dados de saída:** Qual será o resultado do programa? Quais os dados de saída que serão impressos na tela?
- ☞ **Os dados de entrada:** quais são as informações que precisamos solicitar para chegar ao resultado de saída?

**Dica:**

Nos primeiros programas, você deve escrever essa análise antes do algoritmo. Fazendo isso, você rapidamente, conseguirá ler e entender o enunciado do programa que deverá ser feito.

**3.2.1 ANÁLISE – PASSOS PARA A RESOLUÇÃO DE UM ENUNCIADO****1º Passo: Ler o enunciado**

Crie um programa que peça um número inteiro e imprima na tela: o número informado, o dobro do número informado, o triplo do número informado.

**2º Passo: Análise do enunciado**

**Definição do resultado:** calcular o dobro e o triplo do número informado.

**Dados de saída:** devemos imprimir o dobro e o triplo; ambos do tipo inteiro.

**Dados de entrada:** devemos pedir um número, que no caso deve ser inteiro.

**3º Passo: Criação do Algoritmo**

**Algoritmo** Calculando\_o\_Dobro\_e\_Triplo;

**Variáveis** numero, dobro, triplo: **inteiro**;

**Inicio**

**Escreva** (" Programa que Calcula o Dobro e o Triplo de um Número ");

**Escreva** (" Informe um número: ");

**Leia** (numero);

dobro = numero \* 2;

triplo = numero \* 3;

**Escreva** (" O dobro de:", numero," é :", dobro, " e o Triplo é:", triplo);

**Fim.**

**Algoritmo Calculando\_o\_Dobro\_e\_Triplo**

#### 4º Passo: Criação do Programa em C++

```
// Programa dobro_triplo.cpp
#include <iostream.h>
#include <conio.h>
void main( )
{
    int numero, dobro, triplo;
    cout<< " \n que Calcula o Dobro e o Triplo de um Número" ;
    cout<< " \n Informe um número:" ;
    cin>> numero;
    dobro = numero * 2;
    triplo = numero * 3;
    cout<< "\n O dobro de:"<< numero<<" é :" << dobro<< " e o Triplo é:"<<triplo;
    getch( );
}
```

**Programa dobro\_triplo.cpp**

## Exercícios Propostos



1. Encontre os erros do algoritmo abaixo e reescreva o que está errado:

**Algoritmo** Procurando Erros;

**Variáveis** numero 1 e numero 2: **inteiro**;

soma: **caracter**;

**Inicio**

**Escreva** (" Digite o valor do Primeiro Número: ");

**Leia** (numero1);

**Escreva** ( Digite o valor do Segundo Número:);

**Leia** ( numero 2);

soma = numero1 + numero2;

**Escreva** (" O Número 1: " , numero1, " e o Número 2: " , numero2, " é : " , soma " );

**Fim.**

2. Crie um programa em algoritmo e em C++, que leia dois números inteiros. O programa deverá imprimir: a soma dos dois números, a multiplicação dos dois números. O programa deverá imprimir também o resultado do valor da soma, subtraído do valor da multiplicação que foram calculados antes.

**Aula 4****ESTRUTURA DE SELEÇÃO****Objetivos da aula**

Ao final desta aula, você deverá ser capaz de:

Criar algoritmos com estruturas de seleção simples;

Converter os algoritmos para a linguagem de programação C++.

**Conteúdos da aula**

Acompanhe os assuntos desta aula, se preferir, após o seu término, assinale o conteúdo já estudado.

- 🔗 Exemplos de programas com estrutura de seleção.
- 🔗 Exercícios propostos.



Olá aluno!

Você está evoluindo bem. Vamos agora entrar na estrutura de seleção. Preste muita atenção.

Boa aula!

**4**Virtual  
Tupai

#### 4. SELEÇÃO SIMPLES



Segundo Mizrahi (1994, p.87), uma das tarefas fundamentais de qualquer programa é decidir o que deve ser executado a seguir. Os comandos de seleção permitem determinar qual é a ação a ser tomada com base no resultado de uma expressão condicional. Podemos, dessa forma, selecionar alternativas, dependendo de critérios que foram estabelecidos no programa.

São três os tipos de seleção num programa:

Seleção simples

Seleção composta

Seleção de múltiplas escolhas.

É importante afirmar que podemos utilizar algum tipo de seleção ou todos num mesmo programa.

A seleção simples serve para uma comparação simples. Com o comando:

```
Se < comandos>  
    então < opções>;  
    senão < opções> ;
```

Observe:

```
Se (valor1 > valor2 )  
    então Escreva (" O valor1 é maior") ;  
    senão Escreva (" O valor1 não é maior");
```

A comparação é feita e, **Se** for verdadeira, o comando **então** fará o que for programado, no caso será: Escreva ("O valor1 é maior"). Se a comparação não for verdadeira, o comando **senão** fará o que for determinado, no caso do exemplo, imprimirá o comando: Escreva ("O valor1 não é maior").

Podemos utilizar o comando **Se** sem utilizarmos a opção **Senão**.

```
Se (valor1 > valor2 )  
    então Escreva ("O valor1 é maior");
```

Nesse caso, estaríamos somente analisando se o valor1 é maior que o valor2, caso contrário não interessa e não faremos nada a respeito.

**Dica:**

Muita atenção nos ponto e vírgulas. Você só poderá utilizá-los atrás dos comandos que estão logo após o **então** ou o **senão**.

#### 4.1 PRIMEIRO EXEMPLO – SELEÇÃO SIMPLES

No algoritmo Selecao\_Exemp1, temos o seguinte enunciado: Faça um programa que peça ao usuário dois números inteiros, você deverá somar os dois números informados, se o valor for menor que 100, diminua 10 e imprima. Senão, imprima o valor da soma dos números somente.

Passos para a resolução, analisando o enunciado:

**Definição do resultado:** calcular a soma dos 2 números informados. Analisar o resultado, para diminuir ou não antes de imprimir o resultado.

**Dados de saída:** imprimir o resultado da soma Se > 100 diminuir 10 Senão: somente imprimir o resultado da soma.

**Dados de entrada:** devemos pedir dois números inteiros.

O algoritmo:

```
Algoritmo Seleção_Exemp1;
Váriáveis numero1, numero2, soma, resultado: inteiro;
Início
  Escreva (" Programa que Verifica a Soma de 2 Números ");
  Escreva (" Informe um número: ");
  Leia (numero1);
  Escreva (" Informe outro número: ");
  Leia (numero2);
  soma= numero1 + numero2;
  SE (soma > 100) ENTÃO
    Início
      resultado = soma – 100;
      Escreva (" O resultado é :", resultado);
    Fim;
```

**SENÃO**

**Escreva** (" O resultado é :", soma);

**Fim.**

#### Algoritmo Seleção\_Exemp1

Comentários sobre o algoritmo Seleção\_Exemp1:

No comando **SE** (soma > 100) se a comparação for verdadeira, se o valor da soma for maior que 100 **ENTÃO** iremos subtrair 100 e após iremos imprimir o resultado. **Importante:** como são duas ações: somar e imprimir; devemos colocar um comando **Início** antes dos comandos e o comando **Fim** após os comandos.

No comando **SE** (soma > 100); se a comparação for falsa, o valor da soma será impresso. Como é um comando somente, no caso o **Escreva** (), não é necessário o **Início** e o **Fim**. Se preferir, poderá usar, isso não causará problemas.

Sempre devemos usar os comandos **Início** e **Fim** se houver mais de um comando no **Então** e ou no **Senão**.

A conversão do algoritmo Seleção\_Exemp1 para C++ :

```
// Programa Seleção_Exemp1.cpp
#include <iostream.h>
#include <conio.h>
void main( )
{
    int numero1, numero2, soma, resultado;
    cout<< " \n Programa que Verifica a Soma de 2 Números" ;
    cout<< " \n Informe um número:" ;
    cin>> numero1;
    cout<< " \n Informe outro número:" ;
    cin>> numero2;
    soma= numero1 + numero2;
    if (soma > 100) // Comentário: if é igual a Se
    {
        resultado = soma - 100;
```



```
        cout<<" O resultado é :"<< resultado;
    }
    else // Comentário : else é igual ao senão
        cout<<" O resultado é :"<< soma;
    getch( );
}
```

**Programa Seleção\_Exemp1.cpp**

Comentários sobre o programa Seleção\_Exemp1.cpp:

Observou a diferença do comando **SE** do algoritmo do **if** do C++? No C++ não temos o comando **ENTÃO**, simplesmente não precisamos colocar.

No C++, o comando fica assim: **if (<condição> ) else**

Toda vez que precisarmos inserir um comentário no programa, devemos utilizar as duas barras: **//** e tudo o que for digitado será ignorado.

A indentação do programa é muito importante. Indentação é a forma “arrumada” de arrumarmos o programa, com espaçamentos, respeitando os blocos de cada comando.

Blocos de comandos são comandos com várias opções. No programa acima, temos dois blocos. O primeiro bloco: é o próprio **main ( )** que inicia e termina, por isso precisa da abertura das chaves e de seu fechamento no final do programa – **{ }**. O segundo bloco: é o comando **if**.

## 4.2 SEGUNDO EXEMPLO – SELEÇÃO SIMPLES

No algoritmo Selecao\_Exemp2, temos o seguinte enunciado: Faça um programa que leia a idade de uma pessoa. Verifique se a pessoa pode ou não ter carteira de motorista. Escreva uma mensagem informativa ao usuário.

Passos para a resolução, analisando o enunciado:

**Definição do resultado:** verificar a idade, se maior que 18 anos pode dirigir, senão é proibido.

**Dados de saída:** imprimir mensagens, se a idade > 18 poderá dirigir, caso contrário imprimir a mensagem negativa.

**Dados de entrada:** solicitar a idade da pessoa e deve ser do tipo inteiro.

O algoritmo:

**Algoritmo Seleção\_Exemp2;**

**Variáveis** idade: **inteiro**;

**Início**

**Escreva** (" Programa que Verifica Se Você Pode ou Não Dirigir");

**Escreva** (" Digite a sua idade: ");

**Leia** (idade);

**SE** (idade > = 18) **ENTÃO**

**Início**

**Escreva** (" Sua idade é :", idade);

**Escreva** (" Você já pode dirigir, mas primeiro precisar tirar a carteira");

**Fim**;

**SENÃO**

**Início**

**Escreva** (" Sua idade é :", idade);

**Escreva** (" Você não pode dirigir, precisa andar de ônibus, aguarde os 18");

**Fim**;

**Fim.**

**Algoritmo Seleção\_Exemp2**

**Atenção:** Verifique que na comparação do **SE** (idade > = 18), utilizamos o sinal de igual. Por quê? Simples: se você fez 18 anos, você já é considerado maior de idade. Se deixássemos apenas (idade > 18), somente quando você informasse 19 seria impresso a mensagem que poderia dirigir.

A conversão do algoritmo Seleção\_Exemp2 para C++

```
// Programa Seleção_Exemp2.cpp
#include <iostream.h>
#include <conio.h>
void main( )
{
    int idade;
    cout<< " \n Programa que Verifica Se Você Pode ou Não Dirigir" ;
    cout<< " \n Digite a sua idade:" ;
    cin>> idade;
    if (idade >= 18)
```

```

{
    cout<< "\n Sua idade é :"<< idade;
    cout<< "\n Você já pode dirigir, mas primeiro precisar tirar a carteira";
}
else
{
    cout<< "\n Sua idade é : " << idade;
    cout<< "\n Você não pode dirigir, precisa andar de ônibus, aguarde os 18";
}
getch( );
}

```

Programa Seleção\_Exemp2.cpp

### 4.3 TERCEIRO EXEMPLO – SELEÇÃO SIMPLES

No algoritmo Selecao\_Exemp3, temos o seguinte enunciado: Faça um programa que peça ao usuário que informe uma letra referente ao seu sexo. Digitar **F** para feminino ou **M** para masculino. Escreva uma mensagem informativa do sexo ao usuário.

Passos para a resolução, analisando o enunciado:

**Definição do resultado:** verificar a letra informada referente ao sexo.

**Dados de saída:** imprimir mensagens conforme o sexo.

**Dados de entrada:** solicitar a idade da pessoa e deve ser do tipo inteiro.

O algoritmo:

**Algoritmo** Seleção\_Exemp3;

**Variáveis** sexo: **character**;

**Início**

**Escreva** (" Programa que Imprime a Letra Referente ao Sexo");

**Escreva** (" Digite a letra F para Feminino ou M para Masculino: ");

**Leia** (sexo);

**SE** ( (sexo = 'F') ou (sexo = 'f') ) **ENTÃO**

**Início**

**Escreva** (" Sexo é FEMININO");

**Fim**;

**SE** ( (sexo = 'M') **OU** (sexo = 'm') ) **ENTÃO**

**Início**

**Escreva** (" Sexo é MASCULINO");

**Fim;**  
**Fim.**

### Algoritmo Seleção\_Exemp3

Comentários do algoritmo Seleção\_Exemp3:

Primeiro, observe que o tipo da variável é **character**, para que possamos ler uma letra.

Na comparação: sexo = 'M', o **M** está entre aspas simples (apóstrofo), precisamos utilizar aspa simples, senão precisaríamos declarar o **M** e o **F**. Esse recurso só vale para um único caracter.

Estamos utilizando uma comparação mais completa. Verifique que na comparação do **SE** ( (sexo = 'M') ou (sexo = 'm') ) utilizamos o operador **OU**. Conforme já mencionamos: letras maiúsculas são diferentes das letras minúsculas. Portanto, se o usuário digitar o **F** maiúsculo e o seu programa está testando o **f** minúsculo, não teremos o resultado correto sendo impresso.

A conversão do algoritmo Seleccion\_Exemp2 para C++

```
// Programa Seleção_Exemp3.cpp
#include <iostream.h>
#include <conio.h>
void main( )
{
    char sexo;
    cout<< " \n Programa que Imprime a Letra Referente ao Sexo" ;
    cout<< " \n Digite a letra F para Feminino ou M para Masculino:" ;
    cin>> idade;
    if ( (sexo == 'F') || (sexo == 'f') )
    {
        cout<< " \n Sexo é FEMININO";
    }
    if ( (sexo == 'M') || (sexo == 'm') )
    {
        cout<< " \n Sexo é MASCULINO";
    }
}
```

```
getch( );  
}
```

Programa Seleção\_Exemp3.cpp

Importante observar que:

Na expressão: **if** ( (sexo == 'F') || (sexo == 'f') ) nunca haverá ponto e vírgula no fim do parênteses. O comando **OU** está representado pelo **||** . É fundamental ter um parêntese maior entre as condições de teste.

O sinal de igualdade no algoritmo é o simples igual e no C++ é o duplo: **==**

É muito comum esquecer de abrir ou fechar uma chave. Preste muita atenção: toda vez que houver mais de um comando após o **if** ou após o **else** devemos colocar as chaves. Se você observar o exemplo, no comando **if**, só há um comando e mesmo assim usei a chaves. Isso pode? Claro que sim, aliás, tenho como hábito sempre colocar as chaves em meus **if**'s, por isso, a partir de agora, você verá em todos os meus programas o uso das chaves.

## Exercícios Propostos



1. Faça um programa em algortimo que deverá pedir a idade. Conforme a idade deverá ser impressa uma mensagem somente na tela, informando:

Se a pessoa for menor de 16 anos ➤ não pode votar ainda

Se a pessoa tiver entre 16 anos e 18 anos ➤ vota se quiser

Se a pessoa tiver mais de 18 anos ➤ por lei, a votar.

## Aula 5

# SELEÇÃO COMPOSTA



### Objetivos da aula

Ao final desta aula, você deverá ser capaz de:

Criar programas com seleção composta, convertendo-os para a linguagem de programação C++.



### Conteúdos da aula

Acompanhe os assuntos desta aula, se preferir, após o seu término, assinale o conteúdo já estudado.

- 🔗 Exemplo de programas com estruturas de seleção composta;
- 🔗 Exercícios propostos.



Caro aluno!

Muito bem. Para darmos continuidade, vamos agora entrar no assunto relacionado a seleção composta.

Boa aula!



## 5. SELEÇÃO COMPOSTA

A seleção composta são conjuntos de seleções. Podemos inserir condições dentro de condições. Isso é muito útil quando precisamos realizar vários testes. Observe a estrutura do comando:

```
SE <condição1>
  Então ➡ Se a condição do teste no Se for verdadeira
    Início
      Se <condição2> Então
        Início
          Comando 1;
          Comando 2;
        Fim;
      Fim;
    Senão
      Início
        Se <condição3>
          Então
            Início
              Comando 1;
              Comando 2;
            Fim;
          Fim;
      Fim;
```

### Dica:

Todo cuidado é pouco no comando de seleção composta. As múltiplas opções podem trazer o famoso “**erro de lógica**”, que nada mais é do que um erro na hora de construir as opções nas condições. Caso isso ocorra em um de seus programas, a forma mais fácil de achar o erro é verificar condição por condição. Analisando o valor de cada variável e verificando o teste para saber o resultado e assim procurar descobrir o erro.



## 5.1 PRIMEIRO EXEMPLO – SELEÇÃO COMPOSTA

Iremos complementar o algoritmo Seleccion\_Exemp3. Nele temos o seguinte enunciado: Faça um programa que peça ao usuário que informe uma letra referente ao seu sexo. Digitar **F** para feminino ou **M** para masculino. Escreva uma mensagem informativa do sexo ao usuário. Qualquer letra informada que não seja a letra F ou M deverá ser impressa a mensagem: Favor digitar certo da próxima vez.

Passos para a resolução, analisando o enunciado:

**Definição do resultado:** verificar a letra informada referente ao sexo.

**Dados de saída:** imprimir mensagens conforme o sexo e mensagem de erro, se digitado algo diferente de F ou M.

**Dados de entrada:** solicitar a idade da pessoa e deve ser do tipo inteiro.

O algoritmo:

```
Algoritmo Seleção_Composta_1;
Variáveis sexo: character;
Inicio
  Escreva (" Programa que Imprime a Letra Referente ao Sexo");
  Escreva (" Digite a letra F para Feminino ou M para Masculino: ");
  Leia (sexo);
  SE ( (sexo = 'F') ou (sexo = 'f') ) ENTÃO
    Inicio
      Escreva (" Sexo é FEMININO");
    Fim;
  SENÃO
    SE ( (sexo = 'M') OU (sexo = 'm') ) ENTÃO
      Inicio
        Escreva (" Sexo é MASCULINO");
      Fim;
    SENÃO
      Inicio
        Escreva (" Letra Errada -- Favor Digitar Certo da Próxima Vez ");
      Fim;
  Fim.
```

Algoritmo Seleção\_Composta\_1

Observe a seqüência dos testes, no algoritmo Seleção\_Composta1:

Primeiro **SE**: verificamos se a letra digitada é **F** ou **f**

Se o primeiro teste não for verdade, O **SENÃO** entra em ação e no segundo **SE**: verificamos se a letra digitada é **M** ou **m**

Se a letra informada também não for **M** ou **m**, isso significa que a pessoa digitou outra coisa, portanto o outro **SENÃO** entra em ação e imprimimos a mensagem, que a pessoa digitou algo diferente de **F** ou **M**

A conversão do algoritmo Seleção\_Composta1 para C++, ficará assim:

```
// Programa Seleção_Composta1.cpp
#include <iostream.h>
#include <conio.h>
void main( )
{
    char sexo;
    cout<< " \n Programa que Imprime a Letra Referente ao Sexo" ;
    cout<< " \n Digite a letra F para Feminino ou M para Masculino:" ;
    cin>> sexo;
    if ( (sexo == 'F') || (sexo == 'f') )
    {
        cout<< "\n Sexo é FEMININO";
    }
    else
    if ( (sexo == 'M') || (sexo == 'm') )
    {
        cout<< "\n Sexo é MASCULINO";
    }
    else
    {
        cout<< "\n Letra Errada -- Favor Digitar Certo da Próxima Vez";
    }

    getch( );
}
```

Programa Seleção\_Composta1.cpp

## 5.2 PRIMEIRO EXEMPLO – SELEÇÃO COMPOSTA

No exemplo de Forbellone (2000, p.41), temos a seguinte situação: Dados três valores A, B, C; verificar se eles podem ser os comprimentos dos lados de um triângulo. Se forem, verificar o tipo de triângulo: equilátero, isósceles ou escaleno. Caso os valores A, B e C não formarem um triângulo, uma mensagem deverá ser informada. Detalhes:

- a. É triângulo se:  $(A < B + C) \text{ e } (B < A + C) \text{ e } (C < A + B)$
- b. É equilátero se:  $(A = B) \text{ e } (B = C)$  ⇨ dois lados iguais
- c. É isósceles se:  $(A = B) \text{ ou } (A = C) \text{ ou } (B = C)$  ⇨ todos os lados iguais
- d. É escaleno se:  $(A <> B) \text{ e } (B <> C)$  ⇨ todos os lados diferentes

Passos para a resolução do enunciado:

**Definição do resultado:** verificar se os valores informados podem formar um triângulo. Determinar o tipo de triângulo:

**Dados de saída:** Uma mensagem informando se os valores de A, B e C não formam um triângulo; ou uma mensagem informando qual o tipo de triângulo: isósceles, escaleno ou equilátero.

**Dados de entrada:** pedir para informar três valores A, B e C.

O algoritmo:

```
Algoritmo Triangulo;
Váriáveis A, B, C : inteiro;
Início
  Escreva (" Programa que Imprime o Tipo de Triângulo ");
  Escreva (" Digite o Primeiro Lado do Triângulo ");
  Leia (A);
  Escreva (" Digite o Segundo Lado do Triângulo ");
  Leia (B);
  Escreva (" Digite o Terceiro Lado do Triângulo ");
  Leia (C);
  SE ( (A < B + C) e (B < A + C) e (C < A + B) )
    ENTÃO
```

```

SE ( ( A = B ) e ( B = C ) )
    ENTÃO
        Escreva ("Triângulo Equilátero");
    SENÃO
        SE ( ( A = B ) ou ( A = C ) ou ( B = C ) )
            ENTÃO
                Escreva ("Triângulo Isóscele");
            SENÃO
                Escreva ("Triângulo Isóscele");
    SENÃO
        Escreva ("Estes valores não formam um triângulo");
Fim.

```

#### Algoritmo Triangulo

A conversão do algoritmo Seleção\_Composta1 para C++, ficará assim:

```

// Programa Triangulo.cpp
#include <iostream.h>
#include <conio.h>
void main( )
{
    int A, B, C;
    cout<< " \n Programa que Imprime o Tipo de Triângulo" ;
    cout<< " \n Digite o Primeiro Lado do Triângulo:" ;
    cin>> A;
    cout<< " \n Digite o Segundo Lado do Triângulo:" ;
    cin>> B;
    cout<< " \n Digite o Terceiro Lado do Triângulo:" ;
    cin>> C;
    if ( ( A < B + C ) && ( B < A + C ) && ( C < A + B ) )
    {
        if ( ( A == B ) && ( B == C ) )
            cout <<"\n Triângulo Equilátero";
        else
            if ( ( A ==B ) || ( A ==C ) || ( B == C ) )
                cout <<"\n Triângulo Isóscele";
            else
                cout <<"\n Triângulo Isóscele";
    }
}

```

```
else
    cout << "\n Estes valores não formam um triângulo";
getch();
}
```

Programa Triangulo.cpp

Comentários:

Na comparação: `if ( ( A < B + C ) && ( B < A + C ) && ( C < A + B ) )` utilizamos o operador `&&` que é o **E** no algoritmo. Como usamos dois conjuntos de `&&`, significa que, somente **SE** todas as condições forem verdadeiras, como resultado, ao final formam um triângulo.

Na comparação: `if ( ( A == B ) || ( A == C ) || ( B == C ) )` Se qualquer uma das condições for verdadeira, teremos um triângulo equilátero, que tem dois lados iguais.

Na comparação: `if ( ( A == B ) && ( B == C ) )`, um erro que pode passar despercebido é trocar os dois sinais de igual por um sinal. Isso não irá provocar um erro. O computador “pensará” que a variável A receberá o valor de B. Por isso é fundamental muita atenção na digitação correta da igualdade.

## Exercícios Propostos



1. Faça um algoritmo que peça dois valores: uma letra referente ao sexo e número inteiro referente à idade. Imprima uma mensagem avisando se a pessoa poderá ou não se aposentar. Para aposentadoria, os seguintes critérios deverão ser analisados:

⇒ Aposentadoria para homens somente depois dos 65 anos.

⇒ Aposentadoria para mulheres somente depois dos 60 anos.

## Aula 6

# SELEÇÃO MÚLTIPLA ESCOLHA

### Objetivos da aula



Ao final desta aula, você deverá ser capaz de:

- Conhecer e aplicar a seleção de múltipla escolha em programas;
- Estabelecer a necessidade desse comando na construção de programas.

### Conteúdos da aula



Acompanhe os assuntos desta aula, se preferir, após o seu término, assinale o conteúdo já estudado.

- 📌 Exemplicação de programas de múltipla escolha;
- 📌 Exercícios propostos.



Prezado aluno!

Chegamos na etapa de estudarmos a seleção múltipla escolha.

Boa aula!



## 6. SELEÇÃO MÚLTIPLA ESCOLHA

Quando precisamos testar a mesma variável com uma série de valores, podemos utilizar a Seleção de Múltipla Escolha: **Escolha**. A variável do teste deve ser sempre do tipo **inteiro** ou **caracter**. O comando **Escolha** é muito utilizado quando oferecemos várias opções ao usuário, deixando que escolha um valor dentre vários. A vantagem principal desse comando é que ele evita uma série de testes com o comando **SE**. A desvantagem é que os testes somente irão funcionar para variáveis inteiras ou do tipo caracter. A sintaxe do comando é:

**Escolha** < condição>

**Início**

**caso 1:** comandos;

**caso 2:** comandos;

**senão** comandos

**Fim;**

### 6.1 Primeiro Exemplo de Seleção de Múltipla Escolha

Observe o algoritmo Exemplo\_Escolha, demonstrando o uso do comando:

**Algoritmo** Exemplo\_Escolha;

**Variáveis** sexo: **caracter**;

**Início**

**Escreva** (" Programa que Imprime a Letra Referente ao Sexo");

**Escreva** (" Digite a letra F para Feminino ou M para Masculino: ");

**Leia** (sexo);

**Escolha** (sexo)

**Início**

**Caso 'f','F': Escreva** (" Sexo é FEMININO");

**Caso 'm','M': Escreva** (" Sexo é MASCULINO");

**Senão Escreva** (" Letra Errada -- Favor Digitar Certo da Próxima Vez ");

**Fim;**

**Fim.**

**Algoritmo Exemplo\_Escolha**

Comentários:



Observe que este programa é o mesmo da aula anterior, porém usamos o comando **Escolha**. A diferença é visível pela diminuição do tamanho.

O comando **Escolha** avalia o conteúdo da variável **sexo**, caso seja F, imprimirá a mensagem: “Sexo é FEMININO”. **Se** a resposta não for F e for M, a mensagem a ser impressa será: “Sexo é MASCULINO”. E, se este teste também não for F ou não for M, a mensagem do **Senão** será impressa.

Podemos, separando por vírgulas as opções como no **Caso ‘f’,’F’**:

Atenção aos ponto e vírgulas: após o comando **Escolha** não há ponto e vírgula. Em cada opção dos Comandos de cada Caso, sempre teremos o ponto e vírgula.

Podemos colocar mais comandos após o Caso: basta colocar o comando início e fim – fique atento ao próximo algoritmo.

Conversão para a linguagem C++:

```
// Programa Exemplo_Escolha.cpp
#include <iostream.h>
#include <conio.h>
void main( )
{
    char sexo;
    cout<< “ \n Programa que Imprime a Letra Referente ao Sexo” ;
    cout<< “ \n Digite a letra F para Feminino ou M para Masculino:” ;
    cin>> idade;
    switch (sexo)
    {
        case ‘f’,’F’: {cout << “\n Sexo é FEMININO”; break;}
        case ‘m’,’M’: {cout << “\n Sexo é MASCULINO”; break;}
        else cout << “\n Letra Errada -- Favor Digitar Certo da Próxima Vez”;
    }
    getch( );
}
```

Programa Exemplo\_Escolha.cpp

Considerações:

No C++ o comando **Escolha** é o **switch ()**. Após a condição, é necessário o uso das chaves (que representam o início e o fim do comando)

**Dica:**

O algoritmo que verifica o tipo do triângulo: Algoritmo Triângulo, não pode utilizar o comando **Escolha**. Como no algoritmo Triângulo, há uma série de condições para verificar os lados do triângulo. O comando **Escolha** é para um teste simples, com várias possibilidades de resultado.

## 6.2 Segundo Exemplo de Seleção de Múltipla Escolha

No próximo exemplo, veremos um programa que solicita ao usuário dois números inteiros. Em seguida o usuário deve escolher um caracter entre esses: + ou – ou \*, pois cada sinal irá representar o cálculo que deverá ser feito. Imprima o valor resultante.

**Algoritmo** Mini\_Calculadora;

**Variáveis** sinal: **caracter**;

num1, num2 : **inteiro**;

**Início**

**Escreva** (“ Programa Mini Calculadora”);

**Escreva** (“ Informe o Primeiro Número: “);

**Leia** (num1);

**Escreva** (“ Informe o Segundo Número: “);

**Leia** (num2);

**Escreva** (“ Informe um dos sinais: + - \* “);

**Leia** (sinal);

**Escolha** (sinal)

**Início**

**Caso ‘+’:** **Escreva** (“ A soma dos números é : “, num1 + num2);

**Caso ‘-’:** **Escreva** (“ A subtração dos números é : “, num1- num2);

**Caso ‘\*’:** **Escreva** (“ O produto dos números é : “, num1 \* num2);

**Senão Escreva** (“ Sinal Errado -- Favor Digitar Certo da Próxima Vez ”);

**Fim**;

**Fim.**

**Algoritmo Mini\_Calculadora**

Comentário:

No programa não foi criado uma variável para guardar o resultado das operações de cálculo. Como simplesmente iremos somar os dois números, podemos calcular diretamente no comando **Escreva( )** desde que

não estejam entre as aspas. O computador imprime tudo que há entre as aspas. Por essa razão todo cuidado é pouco!

Convertendo para C++, teremos o seguinte programa:

```
// Programa Mini_Calculadora.cpp
#include <iostream.h>
#include <conio.h>
void main( )
{
    char sinal;
    int num1,num2;
    cout<< " \n Programa Mini Calculadora" ;
    cout<< " \n Informe o Primeiro Número:" ;
    cin>> num1;
    cout<< " \n Informe o Segundo Número:" ;
    cin>> num2;
    cout<< " \n Informe um dos sinais: + - * " ;
    cin>> sinal;
    switch (sinal)
    {
        case '+': { cout<< "\n " A soma dos números é : " << num1 + num2; break; }
        case '-': { cout<< "\n " A subtração dos números é : " << num1- num2; break; }
        case '*': { cout<< "\n " O produto dos números é : " << num1 * num2; break; }
        else { cout<< "\n Letra Errada -- Favor Digitar Certo da Próxima Vez"; break; }
    }
    getch( );
}
```

Programa Mini\_Calculadora.cpp

Comentários:

1. A variável sinal é do tipo **char** e é analisada no comando **switch ()** – escolha no algoritmo – conforme o valor da variável, o comando case será executado.
2. Em cada linha do comando **switch()**, usamos o case, <opção> e precisamos utilizar o comando **break**, para forçar a saída no comando **switch ()** se não utilizarmos o **break** todas as opções do case serão testadas, mesmo se algum comando case, já tenha sido usado antes.

**Exercício Proposto**

1. Faça um algoritmo que peça um número inteiro. Cada número representa um dia da semana. Imprima o dia da semana por extenso, utilizando o comando Caso.

## Aula 7

**ESTRUTURAS DE REPETIÇÃO****Objetivos da aula**

Ao final desta aula, você deverá ser capaz de:

- Demonstrar o uso das vantagens da estrutura de repetição;
- Utilizar as três estruturas de repetição.

**Conteúdos da aula**

Acompanhe os assuntos desta aula, se preferir, após o seu término, assinale o conteúdo já estudado.

- 🔗 Comando de repetição com teste no início;
- 🔗 Comando de repetição com teste no fim;
- 🔗 Comando de repetição com variável de controle;
- 🔗 Exercícios propostos.



Olá Aluno!

Caso você já tenha compreendido o assunto até aqui, está evoluindo bem. Agora iremos entrar em estruturas de repetição.

Boa aula!

7

Virtual  
Tupai

## 7. ESTRUTURAS DE REPETIÇÃO



Todos os programas apresentados até agora não permitem que sejam executados mais de uma vez. Se quisermos repetir algum programa precisamos sair do programa e executá-lo novamente. Isso é inviável. Precisamos permitir repetições. Outra necessidade: imagine se precisássemos calcular a média das notas de 500 alunos sabendo-se que cada aluno tem 4 notas. Se fôssemos criar todas essas variáveis perderíamos um tempo enorme só na declaração dessas variáveis.

Segundo Forbellone (2000, p.49), uma estrutura de repetição é uma estrutura com controle de fluxo de execução, que permite repetir diversas vezes um mesmo trecho do programa. Do mesmo jeito que na estrutura de decisão, a estrutura de repetição depende do teste de uma condição.

São três os tipos de estruturas de repetição:

Repetição com teste no início: **Enquanto ... Faça**

Repetição com teste no fim: **Repita ... Até**

Repetição com variável de controle **Para ... Faça**

Na tabela 5, Estruturas de Repetição, observe o funcionamento das estruturas de repetição:

**Tabela 5: Estruturas de Repetição**

Algoritmo	C++	Como funciona?
<b>Enquanto</b> < condição> <b>Faça</b> <b>Início</b> comandos; <b>Fim</b> ;	<b>while</b> (x>b) { comandos; }	O comando <b>Enquanto</b> primeiro a condição é analisada, se for verdadeira os comandos serão executados.
<b>Repita</b> <b>Início</b> comandos; <b>Fim</b> ; <b>Até</b> <condição>;	<b>do</b> { comandos; } <b>while</b> ( x>b);	O comando <b>Repita</b> executa os comandos e faz o teste na condição. Fica repetindo até a condição ser verdadeira para poder terminar.
<b>Para</b> variável = valor_inicial <b>Até</b> valor_final <b>Faça</b> <b>Início</b> comandos; <b>Fim</b> ;	<b>for</b> (cont=1; cont<=100; cont++) { comandos; }	O comando <b>Para</b> incrementa, a variável de controle a partir do <i>valor_inicial</i> , até que, esta atinja o <i>valor_final</i> . Aumentando o valor da variável de controle até que o valor final seja alcançado. Executando os comandos a cada passagem.

**Dica:**

Coloque sempre dentro do bloco de repetição os comandos: **Início** e **Fim**. Mesmo se for somente um comando. Isso ajudará a prevenir uma série de problemas, pois dificilmente teremos somente um comando nessas estruturas de repetições. E não esqueça da indentação, para facilitar a visualização do comando.

**7.1 COMANDO DE REPETIÇÃO: Para....Faça**

A estrutura de repetição **Para Faça**, deve ser usada quando sabemos o **número exato** de repetições. Na estrutura de repetição **Para Faça** devemos usar uma variável de controle. Essa variável deve ser sempre do tipo inteiro ou caracter. Observe o algoritmo abaixo, que tem como objetivo imprimir na tela do computador, todos os números de 1 até 100.

```
Algoritmo Repetir_com_Para;  
Variáveis cont: inteiro;  
Início  
  Escreva (" Programa que Imprime Todos os Números de 1 até 100 ");  
  Para cont= 1 Até 100 Faça  
    Início  
      Escreva ( " ", cont);  
    Fim;  
Fim.
```

Algoritmo Repetir\_com\_Para

Comentários:

No algoritmo Repetir\_com\_Para, não precisamos pedir nenhum valor. O programa irá imprimir todos os números de 1 até 100.

O funcionamento da variável **cont** dentro do **Para Faça**: primeiro ela, a variável, recebe o valor um e este valor é impresso na tela com o comando **Escreva**. Após o comando, internamente, é realizada a soma de mais um ao valor da variável cont (agora ela ficará com o valor igual a dois) e imprimirá na tela o dois, o comando ficará repetindo essa operação até o valor do cont chegar a

cem, que será impresso e o comando irá parar. Indo para o próximo comando do algoritmo que, nesse algoritmo, é fim, mas poderia ser outro qualquer.

Observe o comando **Escreva( )** dentro do Para: **Escreva(“ “,cont)** - colocamos entre aspas dois espaços em branco, isso impedirá que os números fiquem grudados ao aparecer na tela. Sem os espaços, a resposta na tela seria assim: 12345678910....até chegar ao 100. Com os espaços, ficará assim: 1 2 3 4 5 6 7 8 9 10 11 12 ...até 100

Convertendo para C++, teremos o seguinte programa:

```
// Programa Repetir_For.cpp
#include <iostream.h>
#include <conio.h>
void main( )
{
    int cont;
    cout<< “\n Programa que Imprime Todos os Números de 1 até 100 ” ;
    for (cont = 1; cont<=100;cont++)
    {
        cout<< “ “ << cont;
    }
    getch( );
}
```

**Programa Repetir\_For.cpp**

Comentário:

No C++ o comando **Para Faça** sofre algumas alterações. A primeira é que somente escrevemos o **for**, a segunda é que a condição tem três partes, aonde: **cont=1** serve para inicializar a variável cont; **cont<=100** é a condição de teste que determina o limite do for; **cont++** (isso é o mesmo que cont= cont+1) significa que cont incrementa mais um, ou seja, que o cont recebe mais um a cada vez que o comando for é executado.

Outro exemplo do uso do comando de Repetição Para .. Faça. Enunciado: Calcule a soma de todos os números pares de 2 até 1000. Um número par é encon-



trado quando o resto de sua divisão for igual a zero. Portanto utilize o operador **MOD**.

Passos para a resolução do enunciado:

**Definição do resultado:** verificar se o número é par, se for par deveremos acumular o valor da soma.

**Dados de saída:** o resultado da soma de todos os pares.

**Dados de entrada:** nada há para ser solicitado ao usuário.

Algoritmo:

```
Algoritmo Soma_dos Pares;
Variáveis cont,result: inteiro;
Início
    result=0;
    Escreva (" Programa que Imprime a Soma de Todos os Números Pares de 2 até 1000 ");
    Para cont = 1 Até 1000 Faça
        Início
            Se (cont MOD 2 = 0) Então
                Início
                    result= result + cont;
                Fim;
            Fim;
        Escreva ( "A Soma de Todos os Números Pares é: ", result);
    Fim.
```

**Algoritmo Soma\_dos Pares**

Convertendo para C++:

```
// Soma_Pares.cpp
#include <iostream.h>
#include <conio.h>
void main( )
{
    int cont, result;
    cout<< " \n Programa que Imprime a Soma de Todos os Números Pares de 2 até 1000 ";
    for (cont = 1; cont<=100;cont++)
    {
        if (cont % 2==0)
        {
            result= result+cont;
        }
    }
}
```

```
    }  
}  
    cout << " \n A Soma de Todos os Números Pares é: " << result;  
}  
getch( );  
}
```

Programa Soma\_Pares.cpp

Comentários:

No comando **for** ( cont=1; cont <= 100 ; cont ++), utilizamos a variável cont ++, isso significa que a variável cont recebe cont + 1 ou seja, cont = cont +1. O comando funciona da seguinte forma: primeiro o cont recebe o valor um; por segundo o comando faz a comparação e enquanto o cont **for** menor até 100 o comando for será repetido. Por último, o cont ++ é executado aonde o contador é incrementado com mais um.

## 7.2 COMANDO DE REPETIÇÃO: Enquanto..Faça

A estrutura de repetição **Enquanto** é utilizada para repetir um bloco de comandos por várias vezes, sem sabermos ao certo a quantidade exata de vezes. Para isso criamos uma condição de teste, que é testada já no início do bloco. O uso do comando **Enquanto** é ideal, quando precisamos sair do comando de uma forma repentina ou assim que atingir algum objetivo. A forma geral do comando é:

```
Enquanto ( condição ) Faça  
    Início  
        Comando1;  
        Comando2;  
    Fim;
```

No comando **Enquanto** primeiro a condição é avaliada e enquanto a condição for verdadeira, o comando **Enquanto** é executado. No momento em que a condição deixar de ser verdadeira, o comando pára. Observe o algoritmo Media\_Idades que tem como objetivo: ler várias idades, sem limite, assim que for digitada a idade igual a zero o programa deverá encerrar a leitura e apresentar a média de todas as idades informadas e a quantidade de idades informadas.

Passos para a resolução, analisando o enunciado:

**Definição do resultado:** calcular a média das idades de várias pessoas – não sabemos o limite - precisamos acumular o valor de cada idade e contar quantas pessoas participaram da pesquisa.

**Dados de saída:** imprimir o valor da média das idades informadas e a quantidade de idades informadas.

**Dados de entrada:** solicitar a idade da pessoa e deve ser do tipo inteiro. Informar que para saber o resultado, deve ser digitada a idade igual a zero.

O algoritmo:

```
Algoritmo Media_Idades;
Váriáveis idade, acumulador, contador: inteiro;
          media: real;
Início
  media = 0;
  acumulador = 0;
  contador = 0;
  Escreva ( " Programa que Imprime a Média das Idades de Várias Pessoas");
  Escreva ( " Informe Uma Idade – Digite [0] Para Parar");
  Leia ( idade);
  Enquanto (idade <> 0 ) Faça
    Início
      contador = contador +1;
      acumulador = acumulador + idade;
      Escreva ( " Informe Uma Idade – Digite [0] Para Parar");
      Leia (idade);
    Fim;
  media = acumulador / contador;
  Escreva ( " A Quantidade de Idades Informadas é: " , contador);
  Escreva ( " A Média de Idades Informadas é: " , media);
Fim.
```

#### Algoritmo Media\_Idades

Comentários sobre o algoritmo:

A diferença entre as variáveis: contador e acumulador. **Contador** é uma variável que servirá para contar a quantidade de vezes que foram informadas as idades. A cada passagem pelo comando, o contador receberá mais um. **Acumulador** é a variável que irá armazenar a soma das idades informadas. Um detalhe importante: variáveis que recebem valores durante o programa devem ser inicializadas com um valor. No caso do exemplo, elas foram inicializadas com o valor zero.

Convertendo o algoritmo para C++:

```
// Media_Idades.cpp
#include <iostream.h>
#include <conio.h>
void main( )
{
    int idade, acumulador=0, contador=0;
    float media=0;
    cout << "\n Programa que Imprime a Média das Idades de Várias Pessoas" ;
    cout << "\n Informe Uma Idade – Digite [0] Para Parar";
    cin>> idade;
    while (idade<> 0 ) do
    {
        contador = contador +1;
        acumulador = acumulador + idade;
        cout << "\n Informe Uma Idade – Digite [0] Para Parar";
        cin>> idade;
    }
    media = acumulador / contador;
    cout << "\n A Quantidade de Idades Informadas é: " << contador;
    cout << "\n A Média de Idades Informadas é: " << media;
    getch( );
}
```

**Programa Media\_Idades.cpp**

### 7.3 COMANDO DE REPETIÇÃO: Repita ... Até

O comando **Repita** realiza o teste da condição no final do comando. Isso quer dizer que ao menos uma vez o comando será executado. Esse comando é o oposto do comando **Enquanto**, visto que, o **Enquanto** primeiro testa para entrar no bloco e o **Repita** primeiro executa para depois testar a condição. A estrutura do comando é:

```
Repita
Início
    Comando1;
    Comando 2;
    Comando 3;
Fim;
Até <condição>;
```

No algoritmo anterior: Media\_Idades, não seria conveniente utilizar o comando **Repita**, visto que o teste é no fim do comando. Imagine se alguém logo no primeiro teste forneça um valor de idade igual a zero. As variáveis: acumulador e contador irão acumular e contar respectivamente. Tornando o resultado errado, pois estariam permitindo a entrada de da idade iguala zero nos cálculos.

Aproveitando o algoritmo Mini\_Calculadora, feito na Aula 6, o uso do comando **Repita** será demonstrado. Esse programa irá pedir dois números e um sinal ( + ou – ou \*). Conforme o sinal, ele irá mostrar o resultado da operação e pronto. Se desejar repetir com novos números para teste, não será possível. A não ser, é claro, que você feche o programa e o compile novamente.

Para evitar esse transtorno, podemos facilmente utilizar o comando **Repita**, perguntando ao usuário se ele deseja continuar a testar o programa, informando novos valores.

Para isso, basta criar uma nova variável, que irá armazenar a resposta (sim ou não).

Observe o algoritmo Mini\_Calculadora2:

**Algoritmo** Mini\_Calculadora2;

**Variáveis** sinal, resposta: **caracter**;

num1, num2 : **inteiro**;

**Início**

**Repita**

**Início**

**Escreva** (" Programa Mini Calculadora");

**Escreva** (" Informe o Primeiro Número: ");

**Leia** (num1);

**Escreva** (" Informe o Segundo Número: ");

**Leia** (num2);

**Escreva** (" Informe um dos sinais: + - \* ");

**Leia** (sinal);

**Escolha** (sinal)

**Início**

**Caso '+'**: **Escreva** (" A soma dos números é : ", num1 + num2);

**Caso '-'**: **Escreva** (" A subtração dos números é : ", num1- num2);

**Caso '\*'**: **Escreva** (" O produto dos números é : ", num1 \* num2);

**Senão** **Escreva** (" Sinal Errado -- Favor Digitar Certo da Próxima Vez ");

**Fim**;

**Escreva** (" Deseja Repetir o Programa? Digite S – para sim ou N para não ");

**Leia** (resposta);

**Fim**;

**Até** ( ( resposta = 'n' ) **OU** (resposta = 'N' ) );

**Fim**.

**Algoritmo Mini\_Calculadora2**

Comentário:

Após o programa imprimir o resultado do cálculo, uma nova mensagem irá surgir, perguntando: “Deseja Repetir o Programa? Digite S – para sim ou N para não”. O comando **Leia ( )** irá ler a resposta do usuário. Se for **S** o programa irá repetir, mas se for **N** o programa irá encerrar. Resumindo: o programa só termina quando for digitada a letra **n** ou **N**.

Convertendo para C++, teremos o seguinte programa:

```
// Programa Mini_Calculadora.cpp
#include <iostream.h>
#include <conio.h>
void main( )
{
    char sinal, resposta;
    int num1,num2;
    do
    {
        cout<< " \n Programa Mini Calculadora" ;
        cout<< " \n Informe o Primeiro Número:" ;
        cin>> num1;
        cout<< " \n Informe o Segundo Número:" ;
        cin>> num2;
        cout<< " \n Informe um dos sinais: + - * " ;
        cin>> sinal;
        switch (sinal)
        {
            case '+': { cout<< "\n A soma dos números é : " << num1 + num2; break; }
            case '-': { cout<< "\n A subtração dos números é : " << num1- num2; break; }
            case '*': { cout<< "\n O produto dos números é : " << num1 * num2; break; }
            else { cout<< "\n Letra Errada -- Favor Digitar Certo da Próxima Vez"; break; }
        }
        cout<< " \n Deseja Repetir o Programa? Digite S – para sim ou N para não " ;
        cin>> resposta;
    } while ( ( resposta == 'n' ) || (resposta =='N' ) );
}
```

**Programa Mini\_Calculadora.cpp**

Comentário:

Nesse programa, o comando **getch()** não é necessário. Assim que o programa receber a letra n ou N ele irá terminar e não precisa ficar esperando por nada.

## 7.4 EXEMPLO COM AS TRÊS ESTRUTURAS DE REPETIÇÃO

Agora, um programa mais elaborado, com opções diferentes. O programa consiste em imprimir a tabuada de um número inteiro, informado pelo usuário. O usuário poderá escolher que a tabuada seja calculada e impressa com o comando Para ou com o comando Enquanto ou com o comando Repita.

Passos para a resolução do enunciado:

**Definição do resultado:** calcular a tabuada de um número inteiro, com o comando de repetição escolhido:

**Dados de saída:** a tabuada de um número inteiro

**Dados de entrada:** pedir um número inteiro e a opção de escolha do comando de repetição.

O algoritmo:

```
Algoritmo Tabuada;
Variáveis opcao: caracter;
          numero, cont: inteiro;

Início
  Escreva (" Programa Tabuada de um Número Desejado");
  Escreva (" Informe um Número para a Tabuada: ");
  Leia (numero);
  Escreva (" Escolha uma das Estruturas: ");
  Escreva (" [ P ] – Digite P para Imprimir a Tabuada com o comando Para ");
  Escreva (" [ R ] – Digite R para Imprimir a Tabuada com o comando Repita ");
  Escreva (" [ E ] – Digite E para Imprimir a Tabuada com o comando Enquanto ");
  Leia (opcao);
  Escolha (opcao)
  Início
    Caso 'P', 'p': Início
      Para cont =1 Até 10 Faça
```

```

        Início
            Escreva ( numero , " x " , cont , " = " , numero * cont);
        Fim;
    Fim;
Caso 'R','r': Início
    cont =1;
    Repita
    Início
        Escreva ( numero , " x " , cont , " = " , numero * cont);
        cont = cont + 1;
    Fim;
    Até (cont =10);
Fim;
Caso 'E','e': Início
    cont =1;
    Enquanto (cont <=10 ) Faça
    Início
        Escreva ( numero , " x " , cont , " = " , numero * cont);
        cont = cont + 1;
    Fim;
Fim;
Else
    Escreva (" Escolha Errada....");
Fim;
Fim.

```

#### Algoritmo Tabuada

Convertendo para C++:

```

//Tabuada.cpp
#include <iostream.h>
#include <conio.h>
void main( )
{
    char opcao;
    int numero, cont;
    cout<< " \n Programa Tabuada de um Número Desejado";
    cout<< " \n Informe um Número para a Tabuada: ";
    cin>>numero;
    cout<< " \n Escolha uma das Estruturas: ";
    cout<< " \n [ P ] – Digite P para Imprimir a Tabuada com o comando Para ";

```



```

cout<< "\n [ R ] – Digite R para Imprimir a Tabuada com o comando Repita ";
cout<< "\n [ E ] – Digite E para Imprimir a Tabuada com o comando Enquanto ";
cin>>opcao;
switch (opcao)
{
    case 'P','p': { for ( cont = 1; com<=10; cont = cont +1 )
        { cout<< "\n" << numero << " x " << cont << " = " << numero * cont;
        }
        break;
    }
    case 'R','r': { cont=1;
        do { cout<< "\n" << numero << " x " << cont << " = " << numero * cont;
            cont= cont +1;
        } while ( cont< = 10) ;
        break;
    }
    case 'E','e': { cont =1;
        while (cont<=10 ) do
        { cout<< "\n" << numero << " x " << cont << " = " << numero * cont;
            cont= cont +1;
        }
        break;
    }

    else { cout<< "\n Letra Errada -- Favor Digitar Certo da Próxima Vez"; break; }
}
getch( );
}

```

#### Programa Tabuada.cpp

Comentários:

O programa somente parece complexo. Conforme o que for digitado na variável **opcao** (observe que não há acentos e nem o sinal da cedilha) será escolhida uma opção com o comando **switch ()** .

Muito cuidado com as chaves, uma a mais ou a menos o programa não irá funcionar.

## Exercícios Propostos



1. Altere o algoritmo Tabuada, para que, além do que ele já faz, possibilite ao usuário escolher se deseja ou não repetir o programa. Utilize a estrutura Repita ..Até.
2. Crie um programa em algoritmo utilizando a estrutura Para...Faça imprima todos os números ímpares de 3 até 303.
3. Crie um programa em algoritmo que peça a altura de várias pessoas. Quando for digitada a altura -1 o programa deverá encerrar e imprimir a média das alturas.

## Aula 8

# VETORES



### Objetivos da aula

Ao final desta aula, você deverá ser capaz de:

Introdução a vetores



Declaração e exemplificação de Vetores

Exercícios Propostos



### Conteúdos da aula

Acompanhe os assuntos desta aula, se preferir, após o seu término, assinale o conteúdo já estudado.

-  Exemplificação de programas de utilizando vetores.
-  Exercícios propostos



Caro aluno!

Chegamos a etapa final do nosso estudo.

Preste muita atenção no assunto sobre vetores, pois também será muito usado nos demais materiais.

# 8

Virtual  
Tupã



## 8. VETORES

Chamamos de vetores os conjuntos de variáveis agrupadas do mesmo tipo.

Vamos a um exemplo prático da utilização de vetores. Imagine se você precisasse armazenar as notas de 50 alunos. Na forma como vínhamos declarando as variáveis, teríamos que declarar 50 variáveis, ficando assim: nota1, nota2, nota3, nota4,.....até nota50. Na forma de vetores, a declaração fica da seguinte maneira: nota[50] : real. Isso é equivalente a criar 50 variáveis do tipo real.

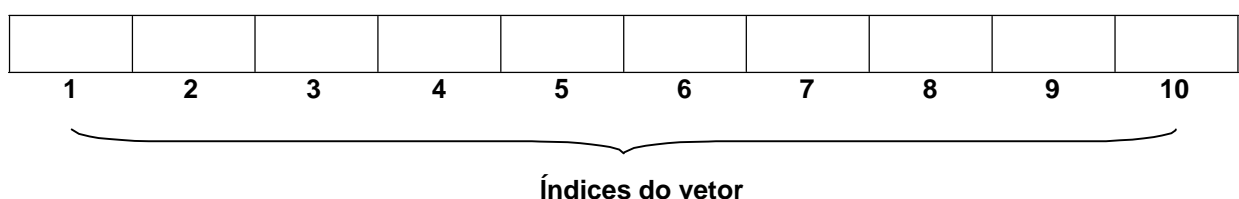
Agora vamos criar uma variável do tipo vetor com 10 posições, mas cada elemento do vetor deverá ser do tipo inteiro.

A declaração da variável ficaria assim:

Variáveis número [10]: inteiro; - criamos assim um vetor com dez elementos inteiros.

Para acessar uma variável, podemos nos referir diretamente a ela: numero [1], numero[2], assim, se quisermos atribuir um valor de 3000 ao elemento 8, do vetor número, deveremos nos referir à variável da seguinte forma: numero[8] = 3000. No caso do numero [8], estamos nos referindo à variável de posição 8. Observe a figura 5. Os quadrados representam cada elemento da variável numero[10]. O número que está dentro dos colchetes representa o índice do vetor ou a posição dele.

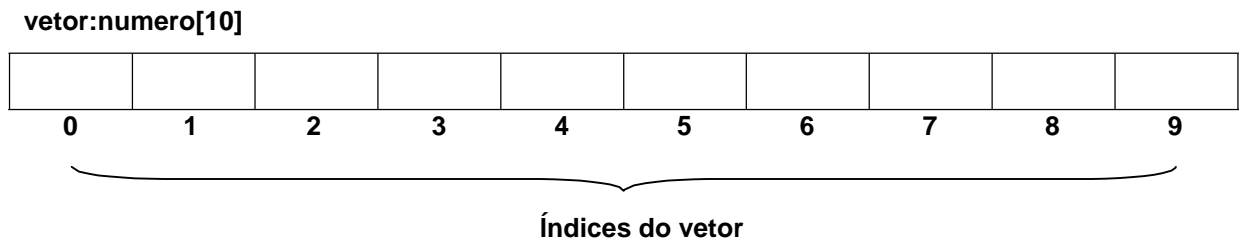
**vetor:numero[10]**



**Figura 5 - Vetor 1**

Trabalhar com vetor, significa trabalhar com estruturas de repetições. O motivo é simples: trabalharemos com muitas variáveis e deixaremos o trabalho mais pesado para a estrutura de repetição. A melhor estrutura de repetição para trabalhar com vetores é o **Para...Faça**, pelo simples motivo de sempre termos o número exato dos elementos do vetor.

Agora temos um detalhe muito importante: no algoritmo e em várias linguagens, o índice do vetor começa sempre do 1. No C++, o índice começa do 0. Veja a figura 6. Continuamos com dez elementos, porém o valor do índice começa do zero.



**Figura 6 - Vetor 2**

Um programa que demonstra o uso de vetores: Ler e guardar num vetor as notas de 20 alunos. Após, verifique quantos alunos tiveram a nota menor que 7. Calcule a média das notas dos alunos..

Passos para a resolução, analisando o enunciado:

**Definição do resultado:** verificar quantos alunos, entre 20, tiveram notas menores que 7. Calcular a média das notas

**Dados de saída:** imprimir a quantidade de alunos que tiveram a nota menor que 7 e a média geral das notas.

**Dados de entrada:** solicitar as notas de 20 alunos.

Algoritmo:

**Algoritmo** Vetor\_de\_Notas;

**Variáveis** cont, quant: **inteiro**;

notas[20]: **real**;

media, acum: **real**;

**Início**

quant = 0; media=0, cont=0;

**Escreva** (" Programa que Imprime a Média das Notas de 20 alunos ");

**Escreva** (" Imprime a Quantidade de Alunos que Tiveram Notas Menores de 7.0 ");

**Para** cont = 1 **Até** 20 **Faça**

**Início**

**Escreva** (" Informe a nota do aluno nº: ", cont);

```

Leia ( notas [ cont ] );
acum = acum + notas [ cont ];
Se ( notas [cont] < 7) Então
    quant =quant + 1;
Fim;
media= acum / 20;
Escreva ( “ A média das notas dos alunos é : “, media);
Escreva ( “ A quantidade de alunos que tiveram notas menos que 7,0 são: “, quant
Fim.

```

#### Algoritmo Vetor\_de\_Notas

Comentários:

Observe a declaração do vetor: notas [20] : real – isso cria uma variável com vinte posições para guardar informações.

A leitura da variável: Leia (notas[ cont ]), dentro dos colchetes precisamos de uma variável para controlar o índice do vetor. Quando o valor do cont for igual a 1 estaremos nos referindo à nota 1, quando o valor do cont for igual a 2, estaremos nos referindo à nota 2 e assim sucessivamente.

Convertendo o programa para C++:

```

// Vetor_Notas.cpp
#include <iostream.h>
#include <conio.h>
#include <iomanip.h>
void main( )
{
    int cont=0, quant=0;
    float notas[20];
    float acum=0, media=0;
    cout<< “ \n Programa Programa que Imprime a Média das Notas de 20 alunos” ;
    cout<< “ \n Imprime a Quantidade de Alunos que Tiveram Notas Menores de 7.0”;
    cout << setprecision(2);
    for (cont = 0; cont<20; cont ++ )
    {
        cout<< “ \n Informe a nota do aluno nº: “, cont+1
        cin >> notas [ cont ];
        acum = acum + notas [ cont ];
        if ( notas [cont] < 7)

```

```
        quant = quant + 1;
    }
    media = acum / 20;
    cout << " \n A média das notas dos alunos é : " << media;
    cout << " \n A quantidade de alunos que tiveram notas menos que 7,0 são: " << quant;
    getch( );
}
```

**Programa Vetor\_Notas.cpp**

#### Comentários:

Em C++, o tipo de variável real fica como **float**. Lembrando que o tipo da variável é inserido antes da variável. Observe que estamos declarando a variável e ao mesmo tempo já estamos atribuindo valores, inicializando as variáveis.

Utilizamos mais uma biblioteca: **#include<iomanip.h>** para podermos utilizar a função **setprecision(2)**, que tem como objetivo a impressão de 2 casas decimais após a vírgula.

O comando **for** (Para..Faça) é essencial ao programa. É ele que faz o trabalho pesado do programa. É nele que são informadas as 20 notas, acumulando os valores e testando cada nota para verificar se a nota é menor de 7.



### Exercícios Propostos

1. Crie um programa que leia 15 números inteiros. Armazene-os num vetor. Some todos os números e imprima a média dos números e a quantidade de números no vetor que são menores do que a média dos números informados.
2. Crie um programa que leia um vetor de 30 caracteres. Verifique quantos desses caracteres que foram informados no vetor, são vogais.



## SOLUÇÕES DOS EXERCÍCIOS



### AULA 1

#### 1. Defina algoritmo.

Algoritmos são seqüências de passos que têm como objetivo atingir um determinado objetivo.

#### 2. Qual a diferença entre um programa fonte e um executável?

Programa Fonte: é a passagem, “tradução”, do algoritmo para uma determinada linguagem de programação. O programa fonte precisa ser compilado para funcionar.

Programa Executável: é o programa pronto, o programa fonte é compilado e após é gerado um arquivo **.exe**, que funciona sem precisar abri-lo num software de programação.

#### 3. Defina linguagem de Programação.

São comandos específicos utilizados para programar o computador. Possuem uma linguagem e sintaxe própria.

#### 4. Pesquise e descubra o nome de no mínimo cinco linguagens de programação para computadores.

Java, C, Pascal, Delphi, Visual Basic, C++, COBOL, Progress, PHP, .NET, C#

### AULA 2

#### 1. Analise as expressões aritméticas e dê o resultado:

a)  $X = ( 10 * ( (100 - 90) - 10) )$  – o valor de X é: **700**

b)  $W = ( 14 + ( 13 * (13 - 8) ) )$  – o valor de W é: **79**

2. O valor de X é igual a 15 e o valor de W é igual a 23, agora analise as expressões abaixo e informe se a expressão é falsa ou verdadeira.

1. (  $X < Y$  ) a expressão é: **verdadeira**
2. (  $Y > X$  ) a expressão é: **verdadeira**
3. (  $X = Y$  ) a expressão é: **falsa**

3. Indique o tipo que a variável “Valor”, conforme o valor que a variável receber. Você deverá informar se a variável será do tipo: inteiro, real, caracter ou lógico.

- e) Valor = “ A ”, o tipo será: **caracter**
- f) Valor = 15,65 o tipo será: **real**
- g) Valor = 47 o tipo será: **inteiro**
- h) Valor = 3 o tipo será: **inteiro**

### AULA 3

1. Encontre os erros do algoritmo abaixo e reescreva o que está errado:

**Algoritmo** Procurando Erros;

**Variáveis** numero 1 e numero 2: **inteiro**;

soma: **caracter**;

**Inicio**

**Escreva** (“ Digite o valor do Primeiro Número: “);

**Leia** (numero1);

**Escreva** ( Digite o valor do Segundo Número:);

**Leia** ( numero 2);

soma = numero1 + numero2;

**Escreva** (“ O Número 1: “ , numero1, “ e o Número 2: “ , numero2, “ é : , soma “ );

**Fim.**

#### Algoritmo Procurando Erros

**Erros:**

1. **Algoritmo** Procurando Erros;

Não pode haver espaço entre Procurando Erros

Forma correta: **Algoritmo** Procurando\_Erros;

**2. Variáveis** numero 1 e numero 2: **inteiro**;

Não pode haver espaço entre o nome da variável e o número, muito menos utilizar a letra **e**

Forma correta: **Variáveis** numero1, numero2: **inteiro**;

**3. Escreva** ( Digite o valor do Segundo Número:);

Não há aspas antes do texto.

Forma correta: **Escreva** (“ Digite o valor do Segundo Número:”);

**4. Escreva** (“ O Número 1: “ , numero1, “ e o Número 2: “ , numero2, “ é : , soma “ );

As aspas estão colocadas no lugar errado.

Forma correta: **Escreva** (“ O Número 1: “ , numero1, “ e o Número 2: “ , numero2, “ é : “ , soma );

2. Crie um programa em algoritmo e em C++, que leia dois números inteiros. O programa deverá imprimir: a soma dos dois números, a multiplicação dos dois números. O programa deverá imprimir também o resultado do valor da soma, subtraído do valor da multiplicação que foram calculados antes.

**PRIMEIRA SOLUÇÃO:** criando várias variáveis.

```
Algoritmo Numeros_Inteiros;  
Variáveis num1,num2 , soma, mult, result: inteiro;  
Inicio  
  Escreva (" Programa Exercício ");  
  Escreva (" Informe dois números: ");  
  Leia (num1);  
  Leia (num2);  
  soma= num1+ num2;  
  mult = num1 * num2;  
  result = soma – mult;  
  Escreva (" O Resultado é : " , result);  
Fim.
```

Algoritmo Numeros\_Inteiros;

```
// Programa Numeros_Inteiros.cpp  
#include <iostream.h>  
#include <conio.h>  
void main( )  
{  
  int num1,num2 , soma, mult, result;  
  cout<< " \n Programa Exercício" ;  
  cout<< " \n Informe dois números:" ;  
  cin>> num1;  
  cin>> num2;  
  soma= num1+ num2;  
  mult = num1 * num2;  
  result = soma – mult;  
  cout<< "\n O O Resultado é : " << result;  
  getch( );  
}
```

Programa Numeros\_Inteiros.cpp

**SEGUNDA SOLUÇÃO:** criando várias variáveis.

**Algoritmo** Numeros\_Inteiros;

**Variáveis** num1,num2 : **inteiro**;

**Inicio**

**Escreva** (“ Programa Exercício “);

**Escreva** (“ Informe dois números: “);

**Leia** (num1);

**Leia** (num2);

**Escreva** (“ O Resultado é : “ , ( ( num1+ num2) - ( num1\* num2) );

**Fim.**

**Algoritmo Numeros\_Inteiros**

**4º Passo: Criação do Programa em C++**

```
// Programa Numeros_Inteiros.cpp
```

```
#include <iostream.h>
```

```
#include <conio.h>
```

```
void main( )
```

```
{
```

```
    int num1,num2;
```

```
    cout<< “ \n Programa Exercício” ;
```

```
    cout<< “ \n Informe dois números:” ;
```

```
    cin>> num1;
```

```
    cin>> num2;
```

```
    cout<< ”\n O O Resultado é : “ << ( ( num1+ num2) - ( num1* num2) );
```

```
    getch( );
```

```
}
```

**Programa Numeros\_Inteiros.cpp**

**AULA 4**

1. Faça um programa em algortimo que deverá pedir a idade. Conforme a idade deverá ser impressa uma mensagem somente na tela, informando:

Se a pessoa for menor de 16 anos ➡ não pode votar ainda

Se a pessoa tiver entre 16 anos e 18 anos ➡ vota se quiser

Se a pessoa tiver mais de 18 anos ➡ por lei, a votar.

**Algoritmo** Votar;

**Variáveis** idade: inteiro;

**Inicio**

**Escreva** (“ Programa que Verifica Se Você Pode ou Não Votar”);

**Escreva** (“ Digite a sua idade: “);

**Leia** (idade);

**SE** (idade< 16) **ENTÃO**

**Inicio**

**Escreva** (“ Você não pode votar!”);

**Fim;**

**SENÃO**

**SE** ( (idade >= 16) **E** ( idade <18) ) **ENTÃO**

**Inicio**

**Escreva** (“ Você vota se quiser!”);

**Fim;**

**SE** (idade > = 18) **ENTÃO**

**Inicio**

**Escreva** (“ Você deve votar!”);

**Fim;**

**Fim.**

**Algoritmo Votar;**

## AULA 5

1. Faça um algoritmo que peça dois valores: uma letra referente ao sexo e número inteiro referente à idade. Imprima uma mensagem avisando se a pessoa poderá ou não se aposentar. Para aposentadoria, os seguintes critérios deverão ser analisados:

⇒ Aposentadoria para homens somente depois dos 65 anos.

⇒ Aposentadoria para mulheres somente depois dos 60 anos.

**Algoritmo** Aposentadoria;

**Variáveis** sexo: **character**;

idade: **inteiro**;

**Inicio**

**Escreva** (“ Programa que Imprime se Você Pode ou Não se Aposentar”);

**Escreva** (“ Digite a letra F para Feminino ou M para Masculino: “);

**Leia** (sexo);

**Escreva** (“ Digite a sua Idade: “);

**Leia** (idade);

**SE** ( ( (sexo = ‘F’) OU (sexo = ‘f’) ) E ( idade >= 60) ) **ENTÃO**

**Inicio**

**Escreva** (“ Sexo é FEMININO e pode se aposentar”);

**Fim**;

**SENÃO**

**SE** ( ( (sexo = ‘M’) OU (sexo = ‘m’) ) E ( idade > = 65) ) **ENTÃO**

**Inicio**

**Escreva** (“ Sexo é MASCULINO e pode se aposentar”);

**Fim**;

**Fim.**

**Algoritmo Aposentadoria;**

**AULA 6**

1. Faça um algoritmo que peça um número inteiro. Cada número representa um dia da semana. Imprima o dia da semana por extenso, utilizando o comando Caso.

**Algoritmo** Dia\_Semana;

**Variáveis** dia: inteiro;

**Início**

**Escreva** (“ Programa que Imprime o Dia da Semana por Extenso”);

**Leia** (dia);

**Escolha** (dia)

**Inicio**

**Caso 1: Escreva** (“Domingo”);

**Caso 2: Escreva** (“Segunda”);

**Caso 3: Escreva** (“Terça”);

**Caso 4: Escreva** (“Quarta”);

**Caso 5: Escreva** (“Quinta”);

**Caso 6: Escreva** (“Sexta”);

**Caso 7: Escreva** (“Sábado”);

**Senão Escreva** (“ Número não é dia da Semana ”);

**Fim;**

**Fim.**

**Algoritmo** Dia\_Semana;



## AULA 7

1. Altere o algoritmo Tabuada, para que além do que ele já faz possibilite ao usuário escolher se deseja ou não repetir o programa. Utilize a estrutura Repita ..Até.

**Algoritmo** Tabuada;

**Variáveis** opcao: **caracter**;

numero, cont: **inteiro**;

resp: **caracter**;

**Início**

**REPITA**

**INÍCIO**

**Escreva** (“ Programa Tabuada de um Número Desejado”);

**Escreva** (“ Informe um Número para a Tabuada: “);

**Leia** (numero);

**Escreva** (“ Escolha uma das Estruturas: “);

**Escreva** (“ [ P ] – Digite P para Imprimir a Tabuada com o comando Para “);

**Escreva** (“ [ R ] – Digite R para Imprimir a Tabuada com o comando Repita “);

**Escreva** (“ [ E ] – Digite E para Imprimir a Tabuada com o comando Enquanto “);

**Leia** (opcao);

**Escolha** (opcao)

**Início**

**Caso ‘P’,’p’: Início**

**Para** cont =1 **Até** 10 **Faça**

**Início**

**Escreva** ( numero ,” x “, cont , “ = “ , numero \* cont);

**Fim;**

**Fim;**

**Caso ‘R’,’r’: Início**

cont =1;

**Repita**

**Início**

**Escreva** ( numero ,” x “, cont , “ = “ , numero \* cont);

cont = cont + 1;

**Fim;**

```

        Até (cont =10);
    Fim;
Caso 'E','e': Início
    cont =1;
    Enquanto (cont <=10 ) Faça
        Início
            Escreva ( numero ,” x “, cont , “ = “ , numero * cont);
            cont = cont + 1;
        Fim;
    Fim;
Else
    Escreva (“ Escolha Errada....”);
Fim;
Escreva (“ Deseja Repetir o Programa? Digite S – para sim ou N para não “);
Leia (resp);
Fim;
Até ( ( resp = 'n' ) OU (resp = 'N' ) );
Fim.

```

### **Algoritmo Tabuada;**

2. Crie um programa em algoritmo utilizando a estrutura Para...Faça imprima todos os números ímpares de 3 até 303.

```

Algoritmo Números_impares;
Variáveis cont : inteiro;
Início
    result=0;
    Escreva (“ Programa que Imprime Todos os Números Ímpares de 3 até 303 “);
    Para cont = 3 Até 303 Faça
        Início
            Se (cont MOD 2 <> 0) Então
                Início
                    Escreva ( “ “, cont);
                Fim;
        Fim;
Fim; Fim.

```

### **Algoritmo Números\_impares**

3. Crie um programa em algoritmo, peça a altura de várias pessoas. Quando for digitada a altura -1 o programa deverá encerrar e imprimir a média das alturas.

```
Algoritmo Media_Alturas;  
Variáveis altura, acumulador, contador: inteiro;  
           media: real;  
Início  
  media = 0;  
  acumulador = 0;  
  contador = 0;  
  Escreva ( " Programa que Imprimi a Média das Alturas de Várias Pessoas");  
  Escreva ( " Informe Uma Altura – Digite [-1] Para Parar");  
  Leia (altura);  
  Enquanto (altura<>-1 ) Faça  
    Início  
      contador = contador +1;  
      acumulador = acumulador + altura;  
      Escreva ( " Informe Uma Altura – Digite [-1] Para Parar");  
      Leia (altura);  
    Fim;  
  media = acumulador / contador;  
  Escreva ( " A Quantidade de Alturas Informadas é: " , contador);  
  Escreva ( " A Média das Alturas Informadas é: " , media);  
Fim.
```

#### **Algoritmo Media\_Alturas**

## AULA 8

1. Crie um programa que leia 15 números inteiros. Armazene-os num vetor. Some todos os números e imprima a média dos números e a quantidade de números no vetor que são menores do que a média dos números informados.

**Algoritmo** Vetor\_de\_Numeros;

**Variáveis** cont, quant: **inteiro**;  
                  numeros[15]: **real**;  
                  media, acum: **real**;

**Início**

quant = 0; media=0, cont=0;

**Escreva** ( “ Programa que Imprime a Média dos 15 Números Informados ”);

**Escreva** ( “ Informe 15 números”);

**Para** cont = 1 **Até** 15 **Faça**

**Início**

**Leia** ( numeros [ cont ] );

acum = acum + numeros [ cont ];

**Fim**;

media= acum / 15;

**Para** cont = 1 **Até** 15 **Faça**

**Início**

**Se** ( numeros [cont] <media) **Então**

quant =quant + 1;

**Fim**;

**Escreva** ( “ A média dos números informados é: “, media);

**Escreva** ( “ A quantidade de números informado e que são menores do que a média: “, quant);

**Fim.**

2. Crie um programa que leia um vetor de 30 caracteres. Verifique quantos desses caracteres são vogais.

**PRIMEIRA VERSÃO: USANDO ESCOLHA****Algoritmo** Vetor\_de\_Caracteres;**Variáveis** cont, quant: **inteiro**;letras[30]: **caracter**;**Início**

quant = 0; cont=0;

**Escreva** (“ Programa que Imprime a quantidade de Vogais que há no vetor “);**Para** cont = 1 **Até** 30 **Faça****Início****Escreva** (“ Informe um Caracter);**Leia** ( letras [ cont ] );**Escolha** (letras [ cont ] )**Início****Caso** ‘a’,’A’, ‘e’,’E’,’i’,’I’,’o’,’O’,’u’,’U’: quant =quant + 1;**Fim**;**Fim**;**Escreva** ( “ A quantidade de vogais no vetor é de: “, quant);**Fim.****SEGUNDA VERSÃO: USANDO SE****Algoritmo** Vetor\_de\_Caracteres;**Variáveis** cont, quant: **inteiro**;letras[30]: **caracter**;**Início**

quant = 0; cont=0;

**Escreva** (“ Programa que Imprime a quantidade de Vogais que há no vetor “);**Para** cont = 1 **Até** 30 **Faça****Início****Escreva** (“ Informe um Caracter);**Leia** ( letras [ cont ] );**SE** ( (letras [ cont ] ) = ‘ A ‘ **OU** (letras [ cont ] ) = ‘ a ‘ **OU** (letras [ cont ] ) = ‘ E ‘ **OU**(letras [ cont ] ) = ‘ e ‘ **OU** (letras [ cont ] ) = ‘ I ‘ **OU** (letras [ cont ] ) = ‘ i ‘ **OU**

```
(letras [ cont ] ) = 'O ' OU (letras [ cont ] ) = ' o' OU (letras [ cont ] ) = 'U ' OU  
(letras [ cont ] ) = ' u' ) ) ENTÃO  
    quant =quant + 1;  
Fim;  
Escreva ( " A quantidade de vogais no vetor é de: ", quant);  
Fim.
```

**Algoritmo Vetor\_de\_Caracteres**

## REFERÊNCIAS

FORBELLONE, André Luiz Villar, EBERSPASCHER, Henri Frederico. **Lógica de Programação: a construção de algoritmos e estrutura de dados**. 2.ed. São Paulo: Makron Books, 2000.

SALVETTI, Dirceu Douglas, BARBOSA, Lisbete Madsen. **Algoritmos**. São Paulo: Makron Books, 1998.

ASCENCIO. Ana Fernanda Gomes. **Lógica de Programação com Pascal**. São Paulo: Makron Books, 1999.

MIZRAHI. Victorine Viviane. **Treinamento em Linguagem C++**: curso completo módulo 1. São Paulo: McGraw-Hill, 1990.

[COR98] CORDEIRO, José. **Transição para C++ - para programadores de C**. Escola Superior de Tecnologia, Instituto Politécnico de Setúbal, 1998.