

# **Fundamentos da Programação de Computadores**

**Algoritmos, Pascal e C/C++**

**Ana Fernanda Gomes Ascencio  
Edilene Aparecida Veneruchi de Campos  
Baixebr**



São Paulo  
Brasil Argentina Colômbia Costa Rica Chile Espanha  
Guatemala México Peru Porto Rico Venezuela

## APRESENTAÇÃO

### OBJETIVOS E RESUMO

Este livro tem como objetivos:

- ◆ apresentar técnicas para a elaboração de algoritmos;
- ◆ apresentar comandos para a implementação de algoritmos nas linguagens PASCAL e C/C++;
- ◆ apresentar a solução de problemas em algoritmos e em programas escritos em PASCAL e em C/C++;
- ◆ incentivar os leitores à programação por meio da proposição de várias situações-problemas ao final de cada capítulo.

Todos os capítulos apresentarão nas seções iniciais conceitos teóricos sobre a utilização do assunto em questão em algoritmos, em PASCAL e em C/C++.

A penúltima seção de cada capítulo apresenta uma série de problemas resolvidos em algoritmos. Os problemas resolvidos em PASCAL e, também, em C/C++ serão encontrados no CD.

A última seção de cada capítulo apresenta diversos problemas para serem resolvidos pelos leitores.

### RELEVÂNCIA, ATUALIDADE E PÚBLICO-ALVO

Durante os anos em que ensinamos algoritmos e fundamentos da programação, observamos a grande dificuldade que os alunos têm em assimilar esses novos conceitos e em adquirir habilidades que lhes permitam resolver problemas reais relacionados à programação.

Observamos, também, que quando os alunos conseguem fazer a análise aprofundada de alguns problemas já resolvidos, parte dessas dificuldades são superadas, o que lhes proporciona maior motivação para os estudos.

Esta obra será aproveitada pelos alunos iniciantes na programação de computadores, visto que as linguagens PASCAL e C/C++ são muito utilizadas no início da programação, por serem de fácil compreensão e ótimas para despertar o raciocínio lógico nos alunos.

Esta obra se diferencia das demais por apresentar grande quantidade de exercícios resolvidos e propostos após cada capítulo, com exceção do Capítulo 1, servindo como apoio para aulas de laboratório, que é uma prática muito comum nas universidades atualmente.

## CONVENÇÕES USADAS NESTE LIVRO

Para facilitar o aprendizado utilizamos fonte monoespaçada em todas as soluções em Algoritmo bem como em nomes de variáveis, campos etc.

Nas soluções dos exercícios em Algoritmo usamos este símbolo (↔) para indicar que a linha continua, isto é, a quebra que usamos existe somente para efeito de diagramção, uma vez que não caberia toda a frase em uma única linha. Ao digitar, o leitor deve dar continuidade sem usar o símbolo (↔) que não existe na solução.

Os seguintes ícones também foram usados:



Para indicar a localização do exercício em PASCAL no CD que acompanha o livro



Para indicar a localização do exercício em C++ no CD que acompanha o livro



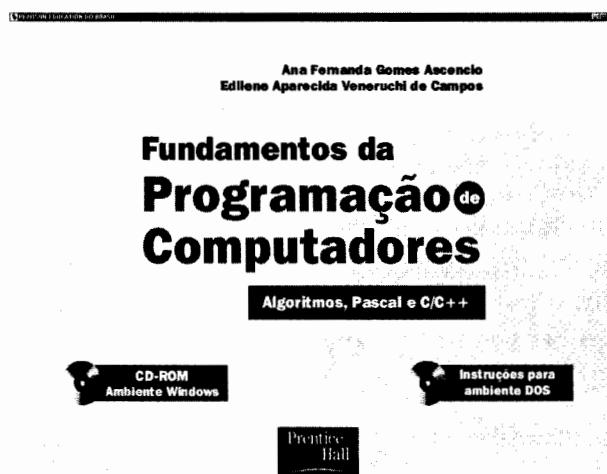
Para indicar a solução do exercício em Algoritmo.

## SOBRE O CD

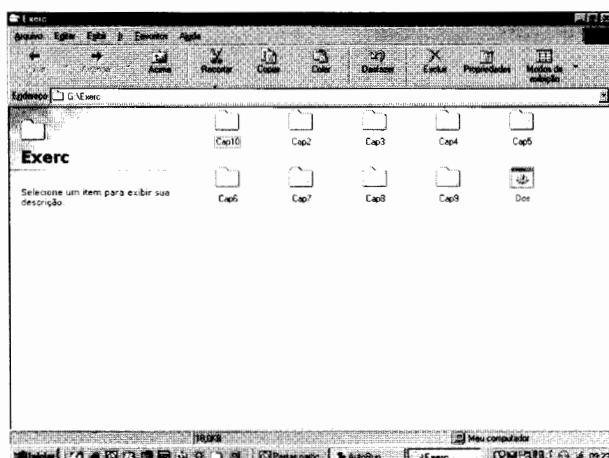
O CD que acompanha este livro possui os exercícios resolvidos de cada capítulo em Pascal e em C/C++, incluindo fontes e executáveis.

Para utilizar o CD insira o mesmo na unidade de CD-ROM do seu computador e aguarde sua execução, que deverá ocorrer de maneira automática. Caso isso não ocorra, dê um clique em **Iniciar → Executar** e preencha o espaço que aparece com a letra correspondente ao drive de CD-ROM de seu computador. Agora dê um clique duplo em **Autorun** e o CD será executado.

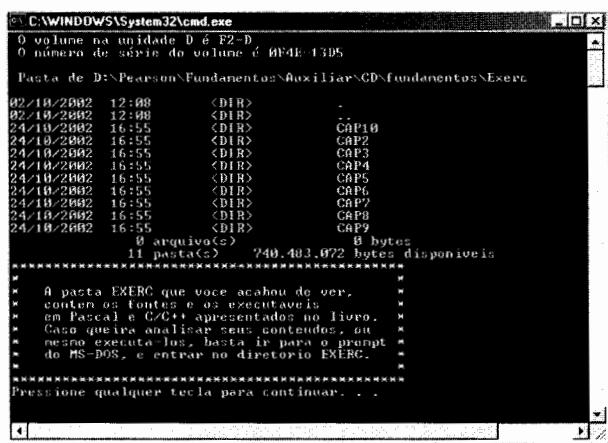
A primeira tela que aparece é a apresentada na figura a seguir.



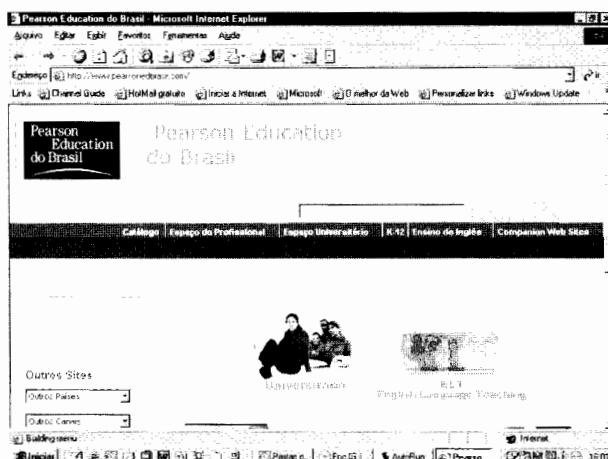
Ao clicar no ícone **CD-ROM Ambiente Windows** você verá uma tela com as pastas que o levarão aos exercícios correspondentes ao capítulo do seu interesse.



Ao clicar no ícone **Instruções para ambiente DOS** você verá a tela apresentada a seguir.



Para que você possa conhecer melhor a Pearson Education acrescentamos um ícone que o remeterá diretamente à nossa home page (figura abaixo). Para isso, dê um clique no logotípo da Prentice Hall. Em caso de dúvidas ou para fazer sugestões ou críticas, dê um clique em **Espaço do Profissional → Serviço ao Cliente** – nossa equipe de atendimento estará pronta para atendê-lo.



## AS AUTORAS

### **ANA FERNANDA GOMES ASCENCIO**

Professora universitária desde 1993 na área de informática. Bacharel em ciência da computação pela Pontifícia Universidade Católica de São Paulo, especialista em sistemas de informação pela Universidade Federal de São Carlos e mestre em ciência da computação pela Universidade Federal do Rio Grande do Sul.

Atualmente é professora da Universidade para o Desenvolvimento do Estado e da Região do Pantanal (Uniderp) e da Universidade Católica Dom Bosco (UCDB), ambas em Campo Grande, Mato Grosso do Sul, ministrando as disciplinas de Fundamentos da Informática e Linguagem de Programação.

### **EDILENE APARECIDA VENERUCHI DE CAMPOS**

Professora universitária desde 1997 na área de informática. Bacharel em ciência da computação pela Universidade Federal de Mato Grosso do Sul, especialista em métodos e técnicas de ensino pela Universidade para o Desenvolvimento do Estado e da Região do Pantanal e mestre em ciência da computação pela Universidade Federal do Rio Grande do Sul.

Atualmente é professora da Universidade para o Desenvolvimento do Estado e da Região do Pantanal (Uniderp), ministrando as disciplinas de Fundamentos da Informática e Estrutura de Dados. Coordenadora dos cursos de Ciência da Computação, Processamento de Dados e Engenharia da Computação, além de ser coordenadora de recursos tecnológicos do Núcleo de Educação a Distância (NEAD) da Uniderp.

# SUMÁRIO

## CAPÍTULO 1

### **CONCEITOS BÁSICOS 1**

1.1	Conceito de algoritmo .....	2
1.2	Método para a construção de algoritmos .....	4
1.3	Tipos de algoritmos .....	4
	1.3.1 Descrição narrativa .....	4
	1.3.2 Fluxograma .....	4
	1.3.3 Pseudocódigo ou portugol .....	4
1.4	Exemplos de algoritmos .....	5
1.5	Conceito de variável .....	8
1.6	Tipos de dados .....	10
	1.6.1 Numérico .....	10
	1.6.2 Lógico .....	10
	1.6.3 Literal ou caractere .....	10
1.7	Formação de identificadores .....	11
1.8	Exemplos de identificadores .....	11
1.9	Linguagem PASCAL .....	11
1.10	Linguagem C/C++ .....	12

## CAPÍTULO 2

### **ESTRUTURA SEQÜENCIAL 15**

2.1	Estrutura seqüencial em ALGORITMOS .....	15
	2.1.1 Declaração de variáveis em ALGORITMOS .....	15
	2.1.2 Comando de atribuição em ALGORITMOS .....	15
	2.1.3 Comando de entrada em ALGORITMOS .....	15
	2.1.4 Comando de saída em ALGORITMOS .....	16
2.2	Estrutura seqüencial em PASCAL .....	16
	2.2.1 Declaração de variáveis em PASCAL .....	16
	2.2.2 Comando de atribuição em PASCAL .....	17
	2.2.3 Comando de entrada em PASCAL .....	18
	2.2.4 Comando de saída em PASCAL .....	18
	2.2.5 Comentários em PASCAL .....	18
	2.2.6 Operadores e funções predefinidas em PASCAL .....	19
2.3	Estrutura seqüencial em C/C++ .....	20
	2.3.1 Declaração de variáveis em C/C++ .....	20
	2.3.2 Comando de atribuição em C/C++ .....	21
	2.3.3 Comando de entrada em C/C++ .....	22

2.3.4 Comando de saída em C/C++ .....	22
2.3.5 Comentários em C/C++ .....	23
2.3.6 Operadores e funções predefinidas em C/C++ .....	23
Exercícios resolvidos .....	25
Exercícios propostos .....	38

## CAPÍTULO 3

### **ESTRUTURA CONDICIONAL 41**

3.1 Estrutura condicional em ALGORITIMOS .....	41
3.1.1 Estrutura condicional simples .....	41
3.1.2 Estrutura condicional composta .....	41
3.2 Estrutura condicional em PASCAL .....	42
3.2.1 Estrutura condicional simples .....	42
3.2.2 Estrutura condicional composta .....	42
3.2.3 Estrutura CASE.....	42
3.2.4 Operadores lógicos .....	43
3.3 Estrutura condicional em C/C++ .....	43
3.3.1 Estrutura condicional simples .....	43
3.3.2 Estrutura condicional composta .....	44
3.3.3 Estrutura CASE.....	44
3.3.4 Operadores lógicos .....	44
Exercícios resolvidos .....	45
Exercícios propostos .....	72

## CAPÍTULO 4

### **ESTRUTURA DE REPETIÇÃO 79**

4.1 Estrutura de repetição em ALGORITMO .....	79
4.1.1 Estrutura de repetição para número definido de repetições (estrutura PARA) .....	79
4.1.2 Estrutura de repetição para número indefinido de repetições e teste no início (estrutura ENQUANTO) .....	79
4.1.3 Estrutura de repetição para número indefinido de repetições e teste no final (estrutura REPITA) .....	80
4.2 Estrutura de repetição em PASCAL .....	81
4.2.1 Estrutura de repetição FOR.....	81
4.2.2 Estrutura de repetição WHILE.....	81
4.2.3 Estrutura de repetição REPEAT .....	82
4.3 Estrutura de repetição em C/C++ .....	82
4.3.1 Estrutura de repetição FOR .....	82

4.3.2 Estrutura de repetição WHILE .....	83
4.3.3 Estrutura de repetição DO-WHILE .....	83
Exercícios resolvidos .....	83
Exercícios propostos .....	124

**CAPÍTULO 5****VETORES 131**

5.1 Vetor em ALGORITMOS .....	131
5.1.1 Definição de vetor .....	131
5.1.2 Declaração de vetor .....	131
5.1.3 Exemplo de vetor .....	131
5.1.4 Atribuindo valores ao vetor .....	131
5.1.5 Carregando um vetor .....	131
5.1.6 Mostrando os elementos do vetor .....	132
5.2 Vetor em PASCAL .....	132
5.2.1 Definição de vetor .....	132
5.2.2 Declaração de vetor .....	132
5.2.3 Exemplo de vetor .....	132
5.2.4 Atribuindo valores ao vetor .....	133
5.2.5 Carregando um vetor .....	133
5.2.6 Mostrando os elementos do vetor .....	133
5.3 Vetor em C/C++.....	133
5.3.1 Definição de vetor .....	133
5.3.2 Declaração de vetor .....	133
5.3.3 Exemplo de vetor .....	133
5.3.4 Atribuindo valores ao vetor .....	134
5.3.5 Carregando um vetor .....	134
5.3.6 Imprimindo um vetor .....	134
Exercícios resolvidos .....	134
Exercícios propostos .....	163

**CAPÍTULO 6****MATRIZ 167**

6.1 Matriz em ALGORITMOS .....	167
6.1.1 Definição de matriz .....	167
6.1.2 Declaração de matriz .....	167
6.1.3 Exemplo de matriz .....	167
6.1.4 Atribuindo valores à matriz .....	167
6.1.5 Carregando uma matriz .....	168

6.1.6 Mostrando os elementos de uma matriz .....	168
<b>6.2 Matriz em PASCAL .....</b>	<b>168</b>
6.2.1 Definição de matriz .....	168
6.2.2 Declaração de matriz .....	169
6.2.3 Exemplo de matriz .....	169
6.2.4 Atribuindo valores à matriz .....	169
6.2.5 Carregando uma matriz .....	169
6.2.6 Mostrando os elementos de uma matriz .....	170
<b>6.3 Matriz em C/C++ .....</b>	<b>170</b>
6.3.1 Definição de matriz .....	170
6.3.2 Declaração de matriz .....	170
6.3.3 Exemplo de matriz .....	170
6.3.4 Atribuindo valores à matriz .....	171
6.3.5 Carregando uma matriz .....	171
6.3.6 Mostrando os elementos de uma matriz .....	171
Exercícios resolvidos .....	172
Exercícios propostos.....	201

**CAPÍTULO 7****FUNÇÕES DE TRATAMENTO DE CARACTERES 205**

<b>7.1 Principais funções de tratamento de caracteres em PASCAL .....</b>	<b>205</b>
<b>7.2 Principais funções de tratamento de caracteres em C/C++ .....</b>	<b>205</b>
7.2.1 Manipulando cadeias de caracteres .....	206
7.2.2 Inicializando cadeias de caracteres .....	207
Exercícios resolvidos .....	207
Exercícios propostos .....	213

**CAPÍTULO 8****REGISTROS 215**

<b>8.1 Definição de registros .....</b>	<b>215</b>
<b>8.2 Declaração de registros em ALGORITMOS .....</b>	<b>215</b>
<b>8.3 Declaração de registros em PASCAL .....</b>	<b>216</b>
8.3.1 Acesso aos campos de um registro em PASCAL .....	216
<b>8.4 Declaração de registros em C/C++ .....</b>	<b>216</b>
8.4.1 Declaração de variáveis do tipo registros em C/C++ .....	217
8.4.2 Acesso a membros de estruturas .....	217
Exercícios resolvidos .....	218
Exercícios propostos .....	285

**CAPÍTULO 9****ARQUIVOS 291**

9.1 Definição de arquivos em ALGORIMTO .....	291
9.2 Declaração de arquivos em PASCAL .....	291
9.3 Comandos de arquivos em PASCAL .....	292
9.3.1 Comando ASSIGN .....	292
9.3.2 Comando REWRITE .....	292
9.3.3 Comando RESET .....	292
9.3.4 Comando CLOSE .....	292
9.3.5 Comando READ .....	293
9.3.6 Comando WRITE .....	293
9.3.7 Comando SEEK .....	293
9.3.8 Comando FILESIZE .....	293
9.3.9 Comando FILEPOS .....	293
9.3.10 Comando NOT EOF .....	293
9.4 Declaração de arquivos em C/C++ .....	294
9.5 Comandos de arquivos em C/C++ .....	294
9.5.1 Função fopen() .....	294
9.5.2 Função fclose() .....	295
9.5.3 Função ferror() .....	295
9.5.4 Função fputc() .....	296
9.5.5 Função fgetc() .....	296
9.5.6 Função fputs() .....	296
9.5.7 Função fgets() .....	296
9.5.8 Função fwrite() .....	296
9.5.9 Função fread() .....	297
9.5.10 Função fseek() .....	297
9.5.11 Função feof() .....	298
9.5.12 Função rewind() .....	298
9.5.13 Função remove() .....	298
9.5.14 Função fflush() .....	298
Exercícios resolvidos .....	298
Exercícios propostos .....	314

**CAPÍTULO 10****SUB-ROTINAS 317**

10.1 Sub-rotinas (programação modularizada) .....	317
10.2 Sub-rotinas em PASCAL .....	318
10.2.1 Procedures sem passagem de parâmetros .....	318
10.2.2 Procedures com passagem de parâmetros .....	319

10.2.3 – Function sem passagem de parâmetros .....	320
10.2.4 – Function com passagem de parâmetros .....	321
10.2.5 – Unit .....	322
10.3 Sub-rotinas em C/C++ (Funções) .....	324
10.3.1 Passagem de parâmetros e tipos de retorno .....	324
10.3.2 Passagem de parâmetros por valor .....	325
10.3.3 Passagem de parâmetros por referência .....	326
Exercícios resolvidos .....	328
Exercícios propostos .....	344

**BIBLIOGRAFIA      347****ÍNDICE REMISSIVO      349**

## CAPÍTULO

# 1

## CONCEITOS BÁSICOS

Desde o início da existência do homem ele tem procurado criar máquinas que o auxiliem em seus trabalhos, diminuindo esforços e economizando tempo. Dentre essas máquinas, o computador tem se mostrado uma das mais versáteis, rápidas e seguras.

O computador é capaz de auxiliar em qualquer coisa que lhe seja solicitada, é consciente, trabalhador e possui muita energia, mas não tem iniciativa, nenhuma independência, não é criativo nem inteligente, por isso precisa receber instruções nos mínimos detalhes.

A finalidade de um computador é receber, manipular e armazenar dados. Se visto somente como um gabinete composto de circuitos eletrônicos, cabos e fontes de alimentação, certamente ele não tem utilidade alguma. O computador só consegue armazenar dados em discos, imprimir relatórios, gerar gráficos, realizar cálculos, entre outras funções, por meio de programas. Portanto, sua finalidade principal é realizar a tarefa de *processamento de dados*, isto é, receber dados por um dispositivo de entrada (por exemplo, teclado, mouse, scanner, entre outros), realizar operações com esses dados e gerar uma resposta que será expressa em um dispositivo de saída (por exemplo, impressora, monitor de vídeo, entre outros) (ASCENCIO, 1999).

### PROCESSAMENTO DE DADOS

Informações iniciais + Operações sobre as informações → Resposta

Dispositivo de entrada

Memória(CPU)

Dispositivo de saída

**FIGURA 1.1:** Ilustração do processamento de dados.

Portanto, um computador tem duas partes diferentes que trabalham juntas: o hardware composto pelas partes físicas e o software composto pelos programas.

Quando queremos escrever (criar, desenvolver) um software para realizar um determinado tipo de processamento de dados, devemos escrever um programa ou vários programas interligados. No entanto, para que o computador comprehenda e execute esse programa, devemos escrevê-lo usando uma linguagem que tanto o computador quanto o desenvolvedor de software entendam. Essa linguagem é chamada de *linguagem de programação*.

As etapas para o desenvolvimento de um programa são:

- ◆ *análise* – nessa etapa estuda-se o enunciado do problema para definir os dados de entrada, o processamento e os dados de saída;
- ◆ *algoritmo* – onde ferramentas do tipo descrição narrativa, fluxograma ou português estruturado são utilizadas para descrever o problema com suas soluções;
- ◆ *codificação* – onde o algoritmo é transformado em códigos da linguagem de programação escolhida para se trabalhar.

Portanto, um programa é a codificação de um algoritmo em uma determinada linguagem de programação (ASCENCIO, 1999).

## 1.1 CONCEITO DE ALGORITMO

A seguir apresentamos alguns conceitos de algoritmos.

“Algoritmo é uma seqüência de passos que visa atingir um objetivo bem definido” (FORBELLONE, 1999).

“Algoritmo é a descrição de uma seqüência de passos que deve ser seguida para a realização de uma tarefa” (ASCENCIO, 1999).

“Algoritmo é uma seqüência finita de instruções ou operações cuja execução, em tempo finito, resolve um problema computacional, qualquer que seja sua instância” (SALVETTI, 1999).

“Algoritmo são regras formais para a obtenção de um resultado ou da solução de um problema, englobando fórmulas de expressões aritméticas” (MANZANO, 1997).

“Ação é um acontecimento que, a partir de um estado inicial, após um período de tempo finito, produz um estado final previsível e bem-definido. Portanto, um algoritmo é a descrição de um conjunto de comandos que, obedecidos, resultam numa sucessão finita de ações” (FARRER, 1999).

Analizando as definições anteriores podemos observar que executamos no dia-a-dia vários algoritmos, como se pode observar nos exemplos descritos a seguir.

### ALGORITMO 1 – SOMAR TRÊS NÚMEROS

**PASSO 1** – Receber os três números.

**PASSO 2** – Somar os três números.

**PASSO 3** – Mostrar o resultado obtido.

**ALGORITMO 2 – FAZER UM SANDUÍCHE**

- PASSO 1 – Pegar o pão.
- PASSO 2 – Cortar o pão ao meio.
- PASSO 3 – Pegar a maionese.
- PASSO 4 – Passar a maionese no pão.
- PASSO 5 – Pegar e cortar alface e tomate.
- PASSO 6 – Colocar alface e tomate no pão.
- PASSO 7 – Pegar o hambúrguer.
- PASSO 8 – Fritar o hambúrguer.
- PASSO 9 – Colocar o hambúrguer no pão.

**ALGORITMO 3 – TROCAR UMA LÂMPADA**

- PASSO 1 – Pegar uma lâmpada nova.
- PASSO 2 – Pegar uma escada.
- PASSO 3 – Posicionar a escada embaixo da lâmpada queimada.
- PASSO 4 – Subir na escada com a lâmpada nova na mão.
- PASSO 5 – Retirar a lâmpada queimada.
- PASSO 6 – Colocar a lâmpada nova.
- PASSO 7 – Descer da escada.
- PASSO 8 – Testar o interruptor.
- PASSO 9 – Guardar a escada.
- PASSO 10 – Jogar a lâmpada velha no lixo.

**ALGORITMO 4 – IR PARA A ESCOLA**

- PASSO 1 – Acordar cedo.
- PASSO 2 – Ir ao banheiro.
- PASSO 3 – Abrir o armário para escolher uma roupa.
- PASSO 4 – Se o tempo estiver quente, pegar uma camiseta e calça jeans; caso contrário, pegar um agasalho e calça jeans.
- PASSO 5 – Vestir a roupa escolhida.
- PASSO 6 – Tomar café.
- PASSO 7 – Pegar uma condução.
- PASSO 8 – Descer próximo à escola.

**ALGORITMO 5 – SACAR DINHEIRO NO BANCO 24 HORAS**

- PASSO 1 – Ir até um banco 24 horas.
- PASSO 2 – Colocar o cartão.
- PASSO 3 – Digitar a senha.
- PASSO 4 – Solicitar a quantia desejada.
- PASSO 5 – Se o saldo for maior ou igual à quantia desejada, sacar; caso contrário, mostrar mensagem de impossibilidade de saque.
- PASSO 6 – Retirar o cartão.
- PASSO 7 – Sair do banco 24 horas.

**OBSERVAÇÃO:**

Você pode estar pensando: “*Mas eu realizo essas atividades de maneira diferente!*” Esse pensamento está correto, pois às vezes um problema pode ser resolvido de maneiras diferentes, porém, gerando a mesma resposta, ou seja, podem existir vários algoritmos para resolver o mesmo problema.

## 1.2 MÉTODO PARA A CONSTRUÇÃO DE ALGORITMOS

Para a construção de qualquer tipo de algoritmo são necessários os passos descritos a seguir:

- a) ler atentamente o enunciado, destacando os pontos mais importantes;
- b) definir os dados de entrada, ou seja, quais dados serão fornecidos;
- c) definir o processamento, ou seja, quais cálculos serão efetuados e quais as restrições para esses cálculos. O processamento é responsável pela transformação dos dados de entrada em dados de saída;
- d) definir os dados de saída, ou seja, quais dados serão gerados depois do processamento;
- e) construir o algoritmo utilizando um dos tipos descritos na próxima seção;
- f) testar o algoritmo realizando simulações.

## 1.3 TIPOS DE ALGORITMOS

Os três tipos mais utilizados de algoritmos são: **descrição narrativa**, **fluxograma** e **pseudocódigo ou portugol**, que descrevemos a seguir.

### 1.3.1 DESCRIÇÃO NARRATIVA

A *descrição narrativa* consiste em analisar o enunciado do problema e escrever, utilizando uma linguagem natural (por exemplo, a língua portuguesa), os passos a serem seguidos para a resolução do problema.

**Vantagem:** não é necessário aprender nenhum conceito novo, pois uma língua natural, neste ponto, já é bem conhecida.

**Desvantagem:** a língua natural abre espaço para várias interpretações, o que posteriormente dificultará a transcrição desse algoritmo para programa.

### 1.3.2 FLUXOGRAMA

O **fluxograma** consiste em analisar o enunciado do problema e escrever, utilizando símbolos gráficos predefinidos (**TABELA 1.1**), os passos a serem seguidos para a resolução do problema.

**Vantagem:** o entendimento de elementos gráficos é mais fácil que o entendimento de textos.

**Desvantagem:** é necessário aprender a simbologia dos fluxogramas e, além disso, o algoritmo resultante não apresenta muitos detalhes, dificultando a sua transcrição para um programa.

### 1.3.3 PSEUDOCÓDIGO OU PORTUGOL

O **pseudocódigo** ou **portugol** consiste em analisar o enunciado do problema e escrever, por meio de regras predefinidas, os passos a serem seguidos para a resolução do problema.

**Vantagem:** a passagem do algoritmo para qualquer linguagem de programação é quase imediata, bastando conhecer as palavras reservadas da linguagem de programação que será utilizada.

**Desvantagem:** é necessário aprender as regras do pseudocódigo, que serão apresentadas nas próximas seções.

**TABELA 1.1:** Conjunto de símbolos utilizados no fluxograma.

	Símbolo utilizado para indicar o início e o fim do algoritmo.
	Permite indicar o sentido do fluxo de dados. Serve exclusivamente para conectar os símbolos ou blocos existentes.
	Símbolo utilizado para indicar cálculos e atribuições de valores.
	Símbolo utilizado para representar a entrada de dados.
	Símbolo utilizado para representar a saída de dados.
	Símbolo que indica que deve ser tomada uma decisão, indicando a possibilidade de desvios.

## 1.4 EXEMPLOS DE ALGORITMOS

Os exemplos a seguir mostram alguns algoritmos desenvolvidos com os três tipos citados anteriormente.

- a) Faça um algoritmo para mostrar o resultado da multiplicação de dois números.

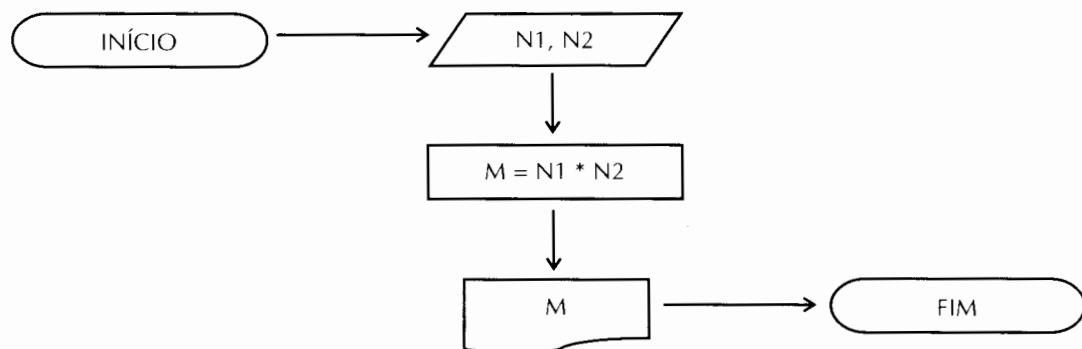
Algoritmo em descrição narrativa:

**PASSO 1** – Receber os dois números que serão multiplicados.

**PASSO 2** – Multiplicar os números.

**PASSO 3** – Mostrar o resultado obtido na multiplicação.

Algoritmo em fluxograma:



Algoritmo em pseudocódigo:

```

ALGORITMO
DECLARE N1, N2, M NUMÉRICO
ESCREVA "Digite dois números"
LEIA N1, N2
M ← N1 * N2
ESCREVA "Multiplicação = ", M
FIM_ALGORITMO.
  
```

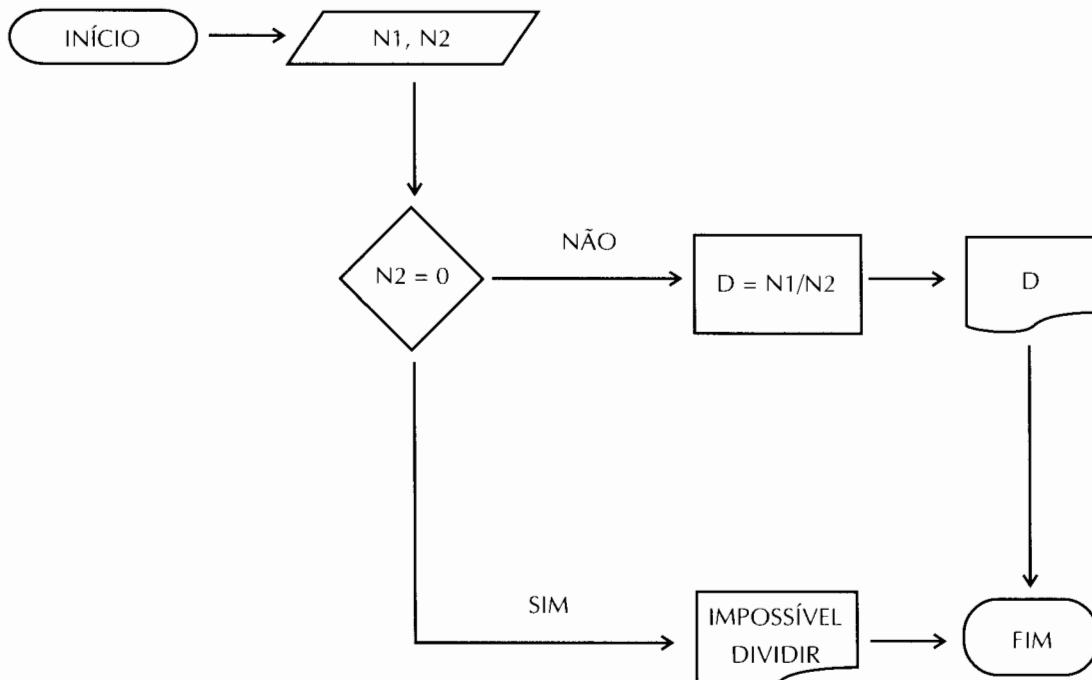
- b) Faça um algoritmo para mostrar o resultado da divisão de dois números.

Algoritmo em descrição narrativa:

**PASSO 1** – Receber os dois números que serão divididos.

**PASSO 2** – Se o segundo número for igual a zero, não poderá haver divisão, pois não existe divisão por zero; caso contrário, dividir os números e mostrar o resultado da divisão.

Algoritmo em fluxograma:



Algoritmo em pseudocódigo:

```

ALGORITMO
DECLARE N1, N2, D NUMÉRICO
ESCREVA "Digite dois números"
LEIA N1, N2
SE N2 = 0
ENTÃO ESCREVA "Impossível dividir"
SENÃO INÍCIO
    D ← N1/N2
    ESCREVA "Divisão = ", D
FIM
FIM_ALGORITMO.
    
```

- c) Faça um algoritmo para calcular a média aritmética entre duas notas de um aluno e para mostrar a situação desse aluno, que pode ser aprovado ou reprovado.

Algoritmo em descrição narrativa:

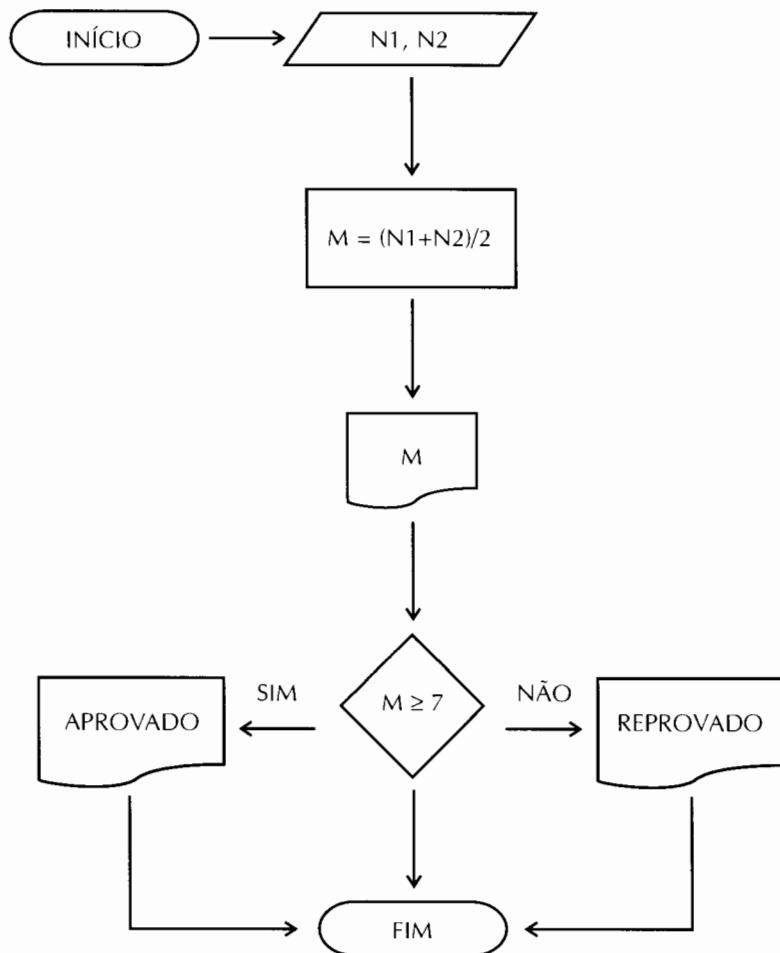
**PASSO 1** – Receber as duas notas.

**PASSO 2** – Calcular a média aritmética.

**PASSO 3** – Mostrar a média aritmética.

**PASSO 4** – Se a média aritmética for maior ou igual a sete, então a situação do aluno é *aprovado*; caso contrário, a situação é *reprovado*.

Algoritmo em fluxograma:



Algoritmo em pseudocódigo:

```

ALGORITMO
DECLARE N1, N2, M NUMÉRICO
ESCREVA "Digite as duas notas"
LEIA N1, N2
M ← (N1 + N2) / 2
ESCREVA "Média = ", M
SE M ≥ 7
ENTÃO ESCREVA "Aprovado"
SENÃO ESCREVA "Reprovado"
FIM_ALGORITMO.
  
```

- d) Faça um algoritmo para calcular o novo salário de um funcionário. Sabe-se que os funcionários que possuem salário atual até R\$ 500,00 terão aumento de 20%, os demais terão aumento de 10%.

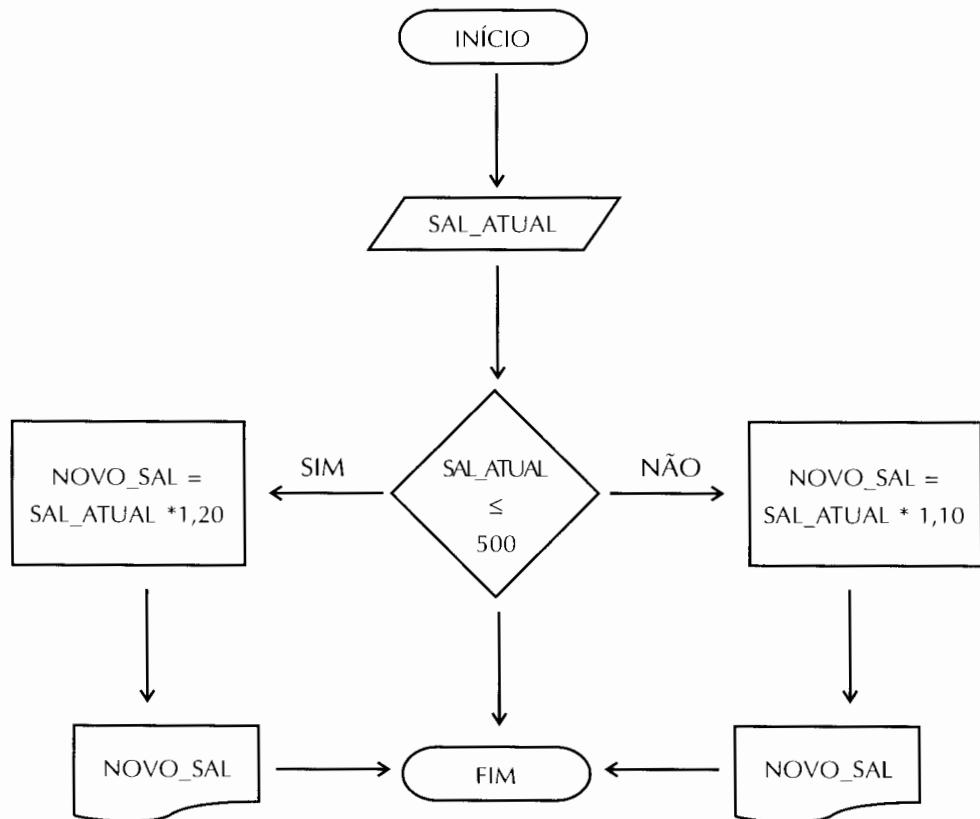
Algoritmo em descrição narrativa:

**PASSO 1** – Receber o salário atual do funcionário.

**PASSO 2** – Se o salário atual do funcionário for até R\$ 500,00, calcular o novo salário com percentual de aumento de 20%; caso contrário, calcular o novo salário com percentual de aumento de 10%.

**PASSO 3** – Mostrar o novo salário.

Algoritmo em fluxograma:



Algoritmo em pseudocódigo:

```

ALGORITMO
DECLARE SAL_ATUAL, NOVO_SAL NUMÉRICO
ESCREVA "Digite o salário atual do funcionário"
LEIA SAL_ATUAL
SE SAL_ATUAL ≤ 500
ENTÃO NOVO_SAL ← SAL_ATUAL * 1,20
SENÃO NOVO_SAL ← SAL_ATUAL * 1,10
ESCREVA "Novo salário = ", NOVO_SAL
FIM_ALGORITMO.
    
```

## 1.5 CONCEITO DE VARIÁVEL

Duas pessoas estão conversando e precisam realizar uma conta.

A primeira pessoa diz: “Vamos somar dois números”.

A primeira pessoa continua: “E o primeiro número é 5”.

A segunda pessoa guarda o primeiro número na cabeça, ou seja, na memória humana.

A primeira pessoa diz: “O segundo número é 3”.

A segunda pessoa também guarda o segundo número na cabeça, sem se esquecer do primeiro número, ou seja, cada número foi armazenado em posições diferentes da memória humana, sem sobreposição.

A primeira pessoa pergunta: “Qual o resultado da soma?”

A segunda pessoa resgata os valores armazenados na memória, realiza a conta e responde dizendo que o resultado é 8.

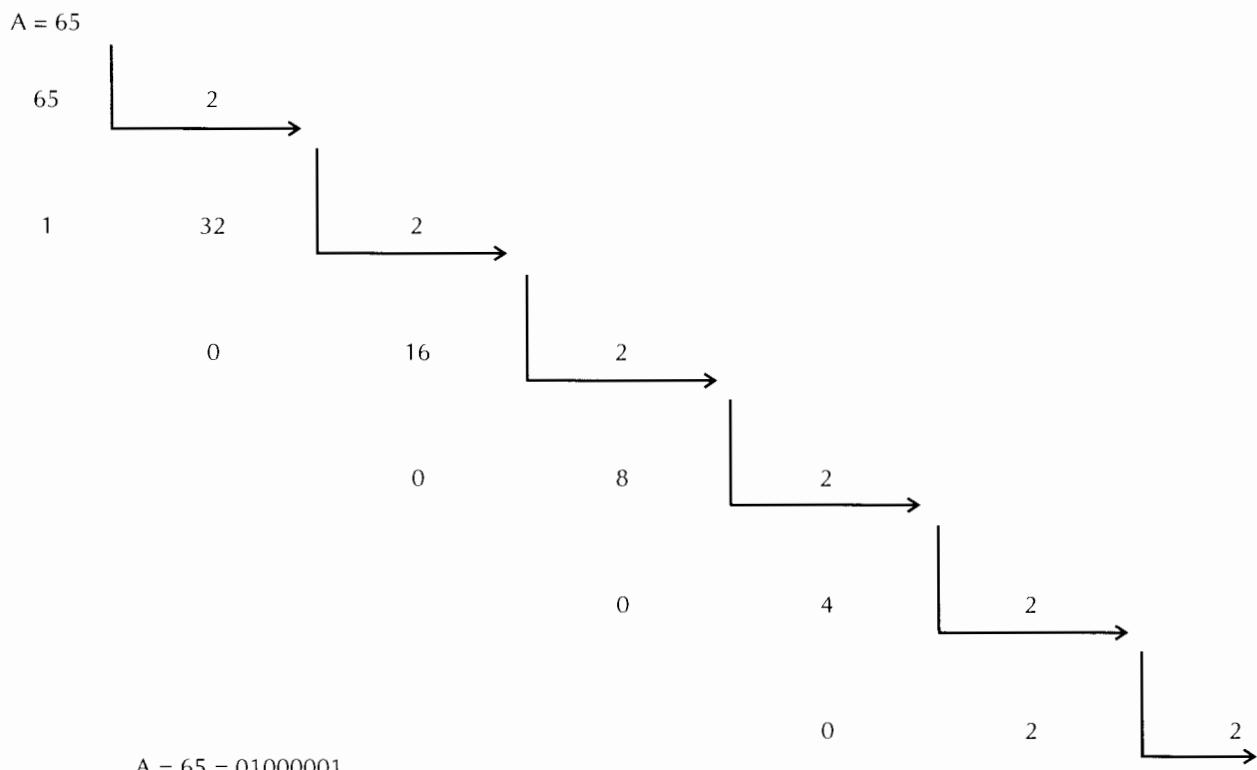
Um *algoritmo* e, posteriormente um *programa*, recebe dados. Tais dados precisam ser armazenados no computador para serem utilizados no processamento. Esse armazenamento é feito na memória.

Todos os computadores trabalham com sistema numérico binário. Nesse sistema numérico, os dados são transformados em 0 e 1 ('zeros' e 'uns') para então serem armazenados na memória. Cada dígito binário (0 ou 1) ocupa porções de memória chamadas bytes (8 bits), em que cada byte é identificado e acessado por meio de um endereço.

Todos os caracteres existentes possuem um caractere numérico correspondente na tabela ASCII e esse caractere numérico é transformado em binário pelo método da divisão para então ser armazenado na memória.

Dessa maneira, uma variável representa uma posição de memória, possuindo nome e tipo, cujo conteúdo pode variar ao longo do tempo, durante a execução de um programa. Embora uma variável possa assumir diferentes valores, ela só pode armazenar um valor a cada instante.

Exemplo de transformação em binário:



Todo computador possui uma tabela de alocação que contém o nome da variável, o tipo da variável (para saber quantos bytes ocupará) e o seu endereço inicial de armazenamento. Dessa maneira, quando queremos buscar algum dado na memória basta sabermos o nome da variável, que o computador, por meio da tabela de alocação, vai buscá-lo automaticamente.

## 1.6 TIPOS DE DADOS

Os tipos de dados mais utilizados são: *numérico, lógico e literal ou caractere* que descrevemos a seguir.

### 1.6.1 NUMÉRICO

Os dados numéricos dividem-se em dois grupos: *inteiros e reais*.

Os números inteiros podem ser positivos ou negativos e *não* possuem parte decimal. Esse tipo de dado, quando armazenado na memória do computador, ocupa 2 bytes, por isso temos  $2^8 \times 2^8 = 2^{16} = 65\,536$  possibilidades de representação dos números inteiros. A faixa de valores inteiros possíveis é de -32767, -32766, ..., 0, ..., 32767, 32768.

Exemplos de dados numéricos inteiros:

```
-23
 98
   0
1350
-357
 237
   -2
```

Os números reais podem ser positivos ou negativos e possuem parte decimal. Esse tipo de dado, quando armazenado na memória do computador, ocupa 4 bytes, por isso temos  $2^8 \times 2^8 \times 2^8 \times 2^8 = 2^{32}$  possibilidades de representação dos números reais. A faixa de valores reais possíveis é muito maior e possui de 6 a 11 dígitos significativos com sinal.

Exemplos de dados numéricos reais:

```
23,45
346,89
-34,88
  0,0
-247,0
```

#### OBSERVAÇÃO:

Os números reais seguem a notação da língua inglesa, ou seja, a parte decimal é separada da parte inteira por um . (ponto) e não por uma , (vírgula).

### 1.6.2 LÓGICO

São também chamados dados booleanos (por causa da álgebra de Boole) e podem assumir os valores *verdadeiro* ou *falso*. Esse tipo de dado, quando armazenado na memória do computador, ocupa 1 byte, pois possui apenas duas possibilidades de representação.

### 1.6.3 LITERAL OU CARACTERE

São dados formados por um único caractere ou por uma cadeia de caracteres. Esses caracteres podem ser as letras maiúsculas, as letras minúsculas, os números (não podem ser usados para cálculos) e os caracteres especiais (&, #, @, ?, +). Esse tipo de dado, quando armazenado na memória do computador, ocupa um byte para cada caractere.

Exemplos de dados literais:

```
'aluno'
'1234'
 '@ internet'
 '0.34'
 '1 + 2'
```

## 1.7 FORMAÇÃO DE IDENTIFICADORES

Os identificadores são os nomes das variáveis, dos programas, das constantes, das rotinas, das unidades etc.

As regras básicas para a formação dos identificadores são:

- ◆ os caracteres que você pode utilizar na formação dos identificadores são: os números, as letras maiúsculas, as letras minúsculas e o caractere sublinhado;
- ◆ o primeiro caractere deve ser sempre uma letra ou o caractere sublinhado;
- ◆ não são permitidos espaços em branco e caracteres especiais (@, \$, +, -, %, !);
- ◆ não podemos usar as palavras reservadas nos identificadores, ou seja, palavras que pertencem a uma linguagem de programação.

## 1.8 EXEMPLOS DE IDENTIFICADORES

Exemplos de identificadores válidos:

```
A  
a  
nota  
NOTA  
X5  
A32  
NOTA1  
MATRICULA  
nota_1  
dia  
IDADE
```

Exemplos de identificadores inválidos:

- 5b – por começar por número
- e 12 – por conter espaço em branco
- x-y – por conter o caractere especial –
- prova 2n – por conter espaço em branco
- nota(2) – por conter os caracteres especiais ()
- case – por ser palavra reservada
- SET – por ser palavra reservada

## 1.9 LINGUAGEM PASCAL

A linguagem PASCAL foi desenvolvida em 1968 por Niklaus Wirth, na Suíça, destinada principalmente à programação científica, mas sua grande evolução permitiu que nos dias de hoje ela seja utilizada para qualquer fim.

Por ser uma linguagem estruturada, isto é, uma linguagem que possui regras para a escrita de seus programas, é muito utilizada nas universidades por alunos que começam a aprender programação. A linguagem PASCAL possui um ambiente integrado de desenvolvimento chamado TURBO PASCAL com as seguintes características:

- ◆ possui um editor que permite ao desenvolvedor do programa digitar, salvar e modificar o código dos seus programas;
- ◆ possui um compilador, que converte os códigos dos seus programas em instruções de máquina e permite que você compile, ou seja, que verifique a existência de erros de sintaxe nos seus programas sem retornar ao sistema operacional.

- ◆ possui um depurador que lhe permite inspecionar um programa durante a sua execução, facilitando a localização de erros;
- ◆ possui um sistema de ajuda ativo que oferece diferentes níveis de informações;
- ◆ possui ainda o ambiente de execução propriamente dito, que lhe permite executar os programas sem sair do TURBO PASCAL (com arquivos de extensão PAS) ou, se preferir, que lhe permite gerar arquivos executáveis a serem executados fora do ambiente do TURBO PASCAL (com arquivos de extensão EXE).

Portanto, para fazer um programa utilizando a linguagem de programação PASCAL devemos:

- ◆ analisar o enunciado do problema, algoritmo, codificação utilizando o editor do ambiente de desenvolvimento TURBO PASCAL;
- ◆ compilar utilizando o compilador do ambiente de desenvolvimento TURBO PASCAL;
- ◆ executar.

O ambiente de desenvolvimento TURBO PASCAL trabalha com um sistema de menus e a sua tela principal possui:

- ◆ barra de menus, que é utilizada para acessar os comandos dos menus e as caixas de diálogos;
- ◆ área principal, onde são escritos os programas;
- ◆ linha de estado.

Cada um dos menus da barra principal possui as seguintes funções:

MENU	FUNÇÕES
FILE	Fornece um conjunto de opções para administrar os seus arquivos. As operações disponíveis incluem criar, abrir, salvar e imprimir arquivos, além de alterar diretórios, acessar o DOS e sair do TURBO PASCAL.
EDIT	Fornece todas as operações de edição necessárias, tais como cortar, colar e copiar, para criar e modificar arquivos.
SEARCH	Permite procurar um texto e efetuar operações de busca e substituição.
RUN	Fornece os comandos necessários para executar programas e iniciar o processo de depuração.
COMPILE	Fornece os comandos e as opções para compilar um programa.
DEBUG	Fornece os comandos e as opções para controlar o depurador.
OPTIONS	Fornece opções de configuração do compilador, do editor, do depurador e do ambiente.
WINDOW	Fornece os comandos necessários para acessar e controlar as várias janelas disponíveis no ambiente.
HELP	Permite acessar o sistema de ajuda.

## 1.10 LINGUAGEM C/C++

Segundo Schildt (1996), Dennis Ritchie inventou a linguagem C e foi o primeiro a implementá-la utilizando um computador DEC PDP-11, que utilizava o sistema operacional UNIX. C é resultante de um processo evolutivo de linguagens. O marco inicial foi uma

linguagem chamada BCPL, desenvolvida por Martin Richards, que teve fortes influências em uma linguagem chamada B, inventada por Ken Thompson. Na década de 1970, B levou ao desenvolvimento de C.

Durante alguns anos, o padrão da linguagem C foi aquele fornecido com a versão 5 do sistema operacional UNIX. Com a popularização dos microcomputadores, várias implementações de C foram criadas, gerando, assim, muitas discrepâncias. Para resolver tal situação o ANSI (*American National Standards Institute*), estabeleceu, em 1983, um comitê para definir um padrão que guiasse todas as implementações da linguagem C.

A linguagem C++ é uma extensão da linguagem C. As instruções que fazem parte da linguagem C representam um subconjunto de C++. Os incrementos encontrados na linguagem C++ foram feitos para dar suporte à programação orientada a objetos. A sintaxe da linguagem C++ é basicamente a mesma da linguagem C.

## ESTRUTURA SEQÜENCIAL

### 2.1 ESTRUTURA SEQÜENCIAL EM ALGORITMOS

```
ALGORITMO
    DECLARE
        bloco de comandos
    FIM_ALGORITMO
```

#### 2.1.1 DECLARAÇÃO DE VARIÁVEIS EM ALGORITMOS

As *variáveis* são declaradas após a palavra **DECLARE** e os tipos mais utilizados são: **NÚMERO** (para variáveis que receberão números), **LITERAL** (para variáveis que receberão caracteres) e **LÓGICO** (para variáveis que receberão apenas dois valores: verdadeiro ou falso).

Exemplo:

```
DECLARE
    X NÚMERO
    Y, Z LITERAL
    TESTE LÓGICO
```

#### 2.1.2 COMANDO DE ATRIBUIÇÃO EM ALGORITMOS

O *comando de atribuição* é utilizado para atribuir valores ou operações a variáveis, sendo representado pelo símbolo  $\leftarrow$ .

Exemplo:

```
x ← 4
x ← x + 2
y ← "aula"
teste ← falso
```

#### 2.1.3 COMANDO DE ENTRADA EM ALGORITMOS

O *comando de entrada* é utilizado para receber dados digitados pelo usuário. Os dados recebidos são armazenados em variáveis. Esse comando é representado pela palavra **LEIA**.

Exemplo:

```
LEIA X
```

Um valor digitado pelo usuário será armazenado na variável **X**.

```
LEIA Y
```

Um ou vários caracteres digitados pelo usuário serão armazenados na variável **Y**.

### 2.1.4 COMANDO DE SAÍDA EM ALGORITMOS

O comando de saída é utilizado para mostrar dados na tela ou na impressora. Esse comando é representado pela palavra ESCREVA e os dados podem ser conteúdos de variáveis ou mensagens.

Exemplo:

```
ESCREVA X
```

Mostra o valor armazenado na variável x.

```
ESCREVA "Conteúdo de Y = ",Y
```

Mostra a mensagem "Conteúdo de Y = " e em seguida o valor armazenado na variável Y.

## 2.2 ESTRUTURA SEQÜENCIAL EM PASCAL

```
PROGRAM nome;
USES nomes das unidades;
VAR nome das variáveis : tipo;
BEGIN
    bloco de comandos;
END
```

As unidades são bibliotecas utilizadas pela linguagem PASCAL para a correta execução do programa. A unidade CRT é obrigatória em todos os programas, pois faz a adequação do hardware com o seu programa.

### 2.2.1 DECLARAÇÃO DE VARIÁVEIS EM PASCAL

As variáveis são declaradas após a palavra VAR e os tipos mais utilizados são: INTEGER (para números inteiros), REAL (para números reais), CHAR (para um caractere), STRING (para vários caracteres) e BOOLEAN (para verdadeiro ou falso).

Exemplo:

```
VAR X: INTEGER;
    Y, Z:REAL;
    NOME: STRING;
    SEXO:CHAR;
    TESTE:BOOLEAN;
```

Os identificadores são os nomes das variáveis, dos programas, das constantes, das rotinas e unidades, entre outras.

As regras básicas para a formação dos identificadores são:

- ◆ podem ter qualquer tamanho. Entretanto, apenas os 63 primeiros caracteres são utilizados pelo compilador;
- ◆ os caracteres que você pode utilizar na formação dos identificadores são: os números, as letras maiúsculas, as letras minúsculas e o caractere sublinhado;
- ◆ o compilador não faz distinção entre letras maiúsculas e minúsculas, portanto o identificador NUM é exatamente igual ao identificador num;
- ◆ o primeiro caractere deve ser sempre uma letra ou o caractere sublinhado;
- ◆ não são permitidos espaços em branco e caracteres especiais (@, \$, +, -, %, !);
- ◆ não é permitido usar palavras reservadas.

**PALAVRAS RESERVADAS:** são nomes utilizados pelo compilador para representar comandos, operadores e nomes de seções de programas. As palavras reservadas da linguagem PASCAL são:

AND	END	NIL	SHR
ASM	FILE	NOT	STRING
ARRAY	FOR	OBJECT	THEN
BEGIN	FUNCTION	OF	TO
CASE	GOTO	OR	TYPE
CONST	IF	PACKED	UNIT
CONSTRUCTOR	IMPLEMENTATION	PROCEDURE	UNTIL
DESTRUCTOR	IN	PROGRAM	USES
DIV	INLINE	RECORD	VAR
DO	INTERFACE	REPEAT	WHILE
DOWNTO	LABEL	SET	WITH
ELSE	MOD	SHL	XOR

Os tipos de dados mais utilizados na linguagem PASCAL estão descritos na tabela a seguir:

TIPO	REPRESENTAÇÃO	FAIXA	TAMANHO
shortint	numérico inteiro	-128 a 127	1 byte
integer	numérico inteiro	-32.768 a 32.767	2 bytes
longint	numérico inteiro	-2.147.483.648 a 2.147.483.647	4 bytes
byte	numérico inteiro	0 a 255	1 byte
word	numérico inteiro	0 a 65.535	2 bytes
real	numérico real	$2,9 \times 10^{-39}$ a $1,7 \times 10^{38}$ (11 a 12 dígitos com sinal)	6 bytes
single	numérico real	$1,5 \times 10^{-45}$ a $3,4 \times 10^{38}$ (7 a 8 dígitos com sinal)	4 bytes
double	numérico real	$5,0 \times 10^{-324}$ a $1,7 \times 10^{308}$ (15 a 16 dígitos com sinal)	8 bytes
extended	numérico real	$3,4 \times 10^{-4932}$ a $1,1 \times 10^{4932}$ (19 a 20 dígitos com sinal)	10 bytes
comp	numérico real	$-9,2 \times 10^{18}$ a $9,2 \times 10^{18}$ (19 a 20 dígitos, inteiro)	8 bytes
boolean	lógico	true ou false	1 byte
char	1 caractere	qualquer caractere	1 byte
string	cadeia de caracteres	qualquer conjunto de caracteres	tantos bytes quantos forem os caracteres

## 2.2.2 COMANDO DE ATRIBUIÇÃO EM PASCAL

O comando de atribuição é utilizado para atribuir valores ou operações a variáveis, sendo representado por := (o sinal de dois pontos e o sinal de igual).

Exemplo:

```
x := 4;
x := x + 2;
y := 2.5;
nome:='AULA';
sexo := 'F';
teste := false;
```

Em PASCAL os caracteres literais são representados entre apóstrofos simples e os números reais utilizam o ponto como separador decimal.

Em PASCAL cada comando é finalizado com o sinal de ponto-e-vírgula.

### **2.2.3 COMANDO DE ENTRADA EM PASCAL**

O comando de entrada é utilizado para receber dados digitados pelo usuário. Os dados recebidos são armazenados em variáveis. Esse comando é representado pela palavra READLN. Sua sintaxe está representada a seguir:

Sintaxe:

```
READLN (nome_da_variável);
READLN (nome_da_variável1, nome_da_variável2);
```

Exemplo:

```
READLN (X);
```

Um valor digitado pelo usuário será armazenado na variável X.

```
READLN (NOME);
```

Um ou vários caracteres digitados pelo usuário serão armazenados na variável NOME.

### **2.2.4 COMANDO DE SAÍDA EM PASCAL**

O comando de saída é utilizado para mostrar dados na tela ou na impressora. Esse comando é representado pelas palavras WRITE ou WRITELN e os dados podem ser conteúdos de variáveis ou mensagens.

Sintaxe:

```
WRITE (nome_da_variável);
WRITELN (nome_da_variável);
WRITE ('mensagem');
WRITELN ('mensagem');
WRITE ('mensagem', nome_da_variável);
WRITELN ('mensagem', nome_da_variável);
```

Exemplo:

```
WRITELN (X);
WRITE (X);
```

Mostra o valor armazenado na variável X.

```
WRITELN ('Conteúdo de Y = ', Y);
WRITE ('Conteúdo de Y = ', Y);
```

Mostra a mensagem "Conteúdo de Y = " e em seguida o valor armazenado na variável Y.

A diferença entre esses comandos é que o comando WRITELN mostra o seu conteúdo e passa o cursor para a linha de baixo, enquanto o comando WRITE mantém o cursor na mesma linha, após mostrar a mensagem.

### **2.2.5 COMENTÁRIOS EM PASCAL**

Os comentários não são interpretados pelo compilador, servem apenas para esclarecer o programador, são excelentes instrumentos de documentação e devem sempre estar entre {.....} ou entre (\*.....\*).

## 2.2.6 OPERADORES E FUNÇÕES PREDEFINIDAS EM PASCAL

OPERADOR	FUNÇÃO	OPERADORES	OPERANDOS	RESULTADO
+	Somar	+	Z ou R	Z ou R
-	Subtrair	-	Z ou R	Z ou R
*	Multiplicar	*	Z ou R	Z ou R
/	Dividir	/	Z ou R	R
div	Quociente inteiro	div	Z	Z
mod	Resto da divisão	mod	Z	Z

Os operadores DIV e MOD só podem ser aplicados com operandos inteiros.

O operador / sempre conduz a um resultado real.

Com os operadores +, -, \* e /, se pelo menos um dos operandos for real, então o resultado será real.

OPERADOR	FUNÇÃO
=	Igual
< >	Diferente
< =	Menor igual
> =	Maior igual
<	Menor
>	Maior

FUNÇÃO	ARGUMENTO	RESULTADO	PASCAL
x	Z ou R	Z ou R	ABS (X)
e <sup>x</sup>	Z ou R	R	EXP (X)
sen x	Z ou R	R	SIN (X)
cos x	Z ou R	R	COS (X)
arctg x	Z ou R	R	ARCTAN (X)
ln x	Z ou R	R	LN (X)
parte inteira de x	R	Z	TRUNC (X)
arredondar x	R	Z	ROUND (X)
parte fracionária de x	R	Z	FRAC (X)
raiz quadrada de x	Z ou R	R	SQRT (X)
x <sup>2</sup>	Z ou R	Z ou R	SQR (X)
π	-	R	PI
incrementar	Z	Z	INC (X, VALOR)
decrementar	Z	Z	DEC (X, VALOR)

**OBSERVAÇÃO:** Por não existir o operador de potenciação, temos:

$$A^B = \text{EXP}(B * \text{LN}(A))$$

Exemplo:

$$3^4 = \text{exp}(4 * \text{ln}(3))$$

$$5^{10} = \text{exp}(10 * \text{ln}(5))$$

**OBSERVAÇÃO:**

As funções SIN, COS e ARCTG esperam receber argumentos no formato de radianos; para receber argumentos em graus siga o próximo exemplo. Na linguagem PASCAL não existe uma função para tangente, assim utilize seno/cosseno.

Exemplo com variável para o valor de  $\pi$ :

```
VALORPI := 3.1415;
READLN(X); { X EM GRAUS }
Y := SIN ((VALORPI * X) / 180);
```

Exemplo utilizando a função PI:

```
READLN(X); { X EM GRAUS }
Y := SIN ((PI * X) / 180);
```

As prioridades entre os operadores são:

- 1º - ( )
- 2º - funções
- 3º - \*, /, DIV, MOD
- 4º - +, -

## 2.3 ESTRUTURA SEQÜENCIAL EM C/C++

```
#include <nome da biblioteca>
void main()
{
    bloco de comandos;
}
```

Bibliotecas são arquivos contendo várias funções que podem ser incorporadas aos programas escritos em C/C++. A diretiva `#include` faz com que o texto contido dentro da biblioteca especificada seja inserido no programa.

As bibliotecas `iostream.h` e `conio.h` permitem a utilização de diversos comandos de entrada e saída.

É importante salientar que a linguagem C/C++ é sensível a letras maiúsculas e minúsculas, ou seja, considera que letras maiúsculas são diferentes de minúsculas (por exemplo, `a` é diferente de `A`). Sendo assim, todos os comandos devem, obrigatoriamente, ser escritos com letras minúsculas.

### 2.3.1 DECLARAÇÃO DE VARIÁVEIS EM C/C++

As *variáveis* são declaradas após a especificação de seus tipos. Os tipos de dados mais utilizados são: `int` (para números inteiros), `float` (para números reais) e `char` (para um caractere). A linguagem C/C++ não possui tipo de dados `boolean` (que pode assumir valores verdadeiro ou falso), pois considera qualquer valor diferente de 0 (zero) como sendo verdadeiro. A linguagem C/C++ não possui um tipo especial para armazenar cadeias de caracteres (strings). Deve-se, quando necessário, utilizar um vetor contendo vários elementos do tipo `char`. Os vetores serão abordados no Capítulo 5.

Exemplo:

```
float X;
```

Declara uma variável chamada `X`, onde pode ser armazenado um número real.

```
float Y, Z;
```

Declara duas variáveis chamadas `Y` e `Z`, onde podem ser armazenados dois números reais.

```
char SEXO;
```

Declara uma variável chamada SEXO, onde pode ser armazenado um caractere.

```
char NOME[40];
```

Declara uma variável chamada NOME, onde podem ser armazenados 40 caracteres.

A linguagem C/C++ possui cinco tipos básicos que podem ser utilizados na declaração das variáveis. São eles: int, float, double, void e char. A partir desses tipos básicos, podem ser definidos outros, conforme apresentado na tabela a seguir.

TIPO	FAIXA DE VALORES	TAMANHO (APROXIMADO)
char	-127 a 127	8 bits
unsigned char	0 a 255	8 bits
signed char	-127 a 127	8 bits
int	-32.767 a 32.767	16 bits
unsigned int	0 a 65.535	16 bits
signed int	-32.767 a 32.767	16 bits
short int	-32.767 a 32.767	16 bits
unsigned short int	0 a 65.535	16 bits
signed short int	-32.767 a 32.767	16 bits
long int	-2.147.483.647 a 2.147.483.647	32 bits
unsigned long int	0 a 4.294.967.295	32 bits
signed long int	-2.147.483.647 a 2.147.483.647	32 bits
float	3.4 E - 38 a 3.4E + 38	32 bits
double	1.7 E - 308 a 1.7 E + 308	64 bits
long double	3.4 E - 4.932 a 1.1 E + 4.932	80 bits

É importante ressaltar que, de acordo com o processador ou compilador C/C++ que estiver sendo utilizado, o tamanho e a faixa de valores podem variar. A faixa de valores apresentada está de acordo com o padrão ANSI e é considerada como faixa mínima.

### 2.3.2 COMANDO DE ATRIBUIÇÃO EM C/C++

O comando de atribuição é utilizado para atribuir valores ou operações a variáveis, sendo representado por = (sinal de igualdade).

Exemplo:

```
x = 4;
x = x + 2;
y = 2.5;
sexo = 'F';
```

Em C/C++ os caracteres são representados entre apóstrofos ('). As cadeias de caracteres devem ser representadas entre aspas (").

Caso seja necessário armazenar uma cadeia de caracteres dentro de uma variável, deve-se utilizar uma função para manipulação de caracteres, conforme apresentado a seguir:

```
strcpy(nome, "João");
```

Para que seja possível a utilização da função strcpy deve-se inserir ao programa, por meio da diretiva include, a biblioteca string.h. As funções de manipulação de strings serão abordadas no Capítulo 7.

Em C/C++ cada comando é finalizado com o sinal de ponto-e-vírgula.

### 2.3.3 COMANDO DE ENTRADA EM C/C++

O comando *de entrada* é utilizado para receber dados digitados pelo usuário. Os dados recebidos são armazenados em variáveis. Os comandos de entrada mais utilizados na linguagem C/C++ são `cin`, `gets` e `scanf`.

Exemplo:

```
cin >> x;
```

Um valor digitado pelo usuário será armazenado na variável `x`.

```
gets(NOME);
```

Um ou vários caracteres digitados pelo usuário serão armazenados na variável `NOME`.

```
scanf(&x);
```

Um valor digitado pelo usuário será armazenado na variável `x`.

O comando `gets` deve ser utilizado quando se deseja digitar uma cadeia contendo espaços em branco, por exemplo, um nome completo como João da Silva. O comando `cin` consegue armazenar os caracteres até que seja encontrado o primeiro espaço em branco e os caracteres posteriores serão desprezados (sendo assim, seria armazenado apenas João). O comando `gets` e o comando `scanf` armazenam toda a cadeia até que seja pressionada a tecla ENTER.

### 2.3.4 COMANDO DE SAÍDA EM C/C++

O comando *de saída* é utilizado para mostrar dados na tela ou na impressora. Os comandos de saída mais utilizados na linguagem C/C++ são `cout` e `printf`.

Exemplo:

```
cout << x;
```

Mostra o valor armazenado na variável `x`.

```
cout << "Conteúdo de X = " << x;
```

Mostra a mensagem "Conteúdo de X = " e em seguida o valor armazenado na variável `x`.

```
printf("%d", Y);
```

Mostra o número inteiro armazenado na variável `Y`.

```
printf("Conteúdo de Y = %d", Y);
```

Mostra a mensagem "Conteúdo de Y = " e em seguida o número inteiro armazenado na variável `Y`.

```
printf("%f", X);
```

Mostra o número real armazenado na variável `X`.

```
printf("%5.2f", X);
```

Mostra o número real armazenado na variável `X`, utilizando cinco casas para a parte inteira e duas casas decimais.

```
printf("Conteúdo de X = %5.2f", X);
```

Mostra a mensagem "Conteúdo de X =" e em seguida o número real armazenado na variável X, utilizando cinco casas para a parte inteira e duas casas decimais.

### 2.3.5 COMENTÁRIOS EM C/C++

Comentários são textos que podem ser inseridos em programas com o objectivo de documentá-lo. Os comentários não são analisados pelo compilador.

Os comentários podem ocupar uma ou várias linhas, devendo ser inseridos nos programas utilizando os símbolos /\* ..... \*/ ou //.

Exemplo:

```
/*
linhas de comentário
linhas de comentário
*/
```

A região de comentários é aberta com os símbolos /\* e é encerrada com os símbolos \*/.

```
// comentário
```

A região de comentários é aberta pelos símbolos // e é encerrada automaticamente ao final da linha.

### 2.3.6 OPERADORES E FUNÇÕES PREDEFINIDAS EM C/C++

A linguagem C/C++ possui operadores e funções predefinidas destinadas a cálculos matemáticos e à manipulação de caracteres. Alguns são apresentados a seguir.

#### OPERADOR DE ATRIBUIÇÃO

OPERADOR	EXEMPLO	COMENTÁRIO
=	x = y	O conteúdo da variável y é atribuído à variável x. (A uma variável pode ser atribuído o conteúdo de outra variável; um valor constante ou, ainda, o resultado de uma função).
.		

#### OPERADORES MATEMÁTICOS

OPERADOR	EXEMPLO	COMENTÁRIO
+	x + y	Soma o conteúdo de x e de y.
-	x - y	Subtrai o conteúdo de y do conteúdo de x.
*	x * y	Multiplica o conteúdo de x pelo conteúdo de y.
/	x / y	Obtém o quociente da divisão de x por y.
%	x % y	Obtém o resto da divisão de x por y.
++	x++	Aumenta o conteúdo de x em uma unidade.
--	x--	Diminui o conteúdo de x em uma unidade.

O operador % só pode ser utilizado com operandos do tipo inteiro.

**OPERADORES MATEMÁTICOS DE ATRIBUIÇÃO**

OPERADOR	EXEMPLO	COMENTÁRIO
<code>+=</code>	<code>x += y</code>	Equivale a $x = x + y$
<code>-=</code>	<code>x -= y</code>	Equivale a $x = x - y$
<code>*=</code>	<code>x *= y</code>	Equivale a $x = x * y$
<code>/=</code>	<code>x /= y</code>	Equivale a $x = x / y$
<code>%=</code>	<code>x %= y</code>	Equivale a $x = x \% y$

Os operadores matemáticos de atribuição são utilizados para representar de maneira sintética uma operação aritmética e, posteriormente, uma operação de atribuição. Por exemplo, na tabela anterior o operador `+=` está sendo usado para realizar a operação  $x+y$  e, posteriormente, atribuir o resultado obtido à variável `x`.

**OPERADORES RELACIONAIS**

OPERADOR	EXEMPLO	COMENTÁRIO
<code>==</code>	<code>x == y</code>	O conteúdo de <code>x</code> é igual ao conteúdo de <code>y</code> .
<code>!=</code>	<code>x != y</code>	O conteúdo de <code>x</code> é diferente do conteúdo <code>y</code> .
<code>&lt;=</code>	<code>x &lt;= y</code>	O conteúdo de <code>x</code> é menor ou igual ao conteúdo de <code>y</code> .
<code>&gt;=</code>	<code>x &gt;= y</code>	O conteúdo de <code>x</code> é maior ou igual ao conteúdo de <code>y</code> .
<code>&lt;</code>	<code>x &lt; y</code>	O conteúdo de <code>x</code> é menor que o conteúdo de <code>y</code> .
<code>&gt;</code>	<code>x &gt; y</code>	O conteúdo de <code>x</code> é maior que o conteúdo de <code>y</code> .

**FUNÇÕES MATEMÁTICAS**

FUNÇÃO	EXEMPLO	COMENTÁRIO
<code>ceil</code>	<code>ceil(x)</code>	Arredonda um número real para cima, por exemplo, <code>ceil(3.2)</code> é 4.
<code>cos</code>	<code>cos(x)</code>	Calcula o cosseno de <code>x</code> ( <code>x</code> deve estar representado em radianos).
<code>exp</code>	<code>exp(x)</code>	Obtém o logaritmo natural e elevado à potência <code>x</code> .
<code>fabs</code>	<code>fabs(x)</code>	Obtém o valor absoluto de <code>x</code> .
<code>floor</code>	<code>floor(x)</code>	Arredonda um número real para baixo, por exemplo, <code>floor(3.2)</code> é 3.
<code>log</code>	<code>log(x)</code>	Obtém o logaritmo natural de <code>x</code> .
<code>log10</code>	<code>log10(x)</code>	Obtém o logaritmo de base 10 de <code>x</code> .
<code>modf</code>	<code>modf(x, y)</code>	Decompõe um determinado número real em duas partes: <code>x</code> recebe a parte fracionária e <code>y</code> recebe a parte inteira do número.
<code>pow</code>	<code>pow(x, y)</code>	Calcula a potência de <code>x</code> elevado a <code>y</code> .
<code>sin</code>	<code>sen(x)</code>	Calcula o seno de <code>x</code> ( <code>x</code> deve estar representado em radianos).
<code>sqrt</code>	<code>sqrt(x)</code>	Calcula a raiz quadrada de <code>x</code> .
<code>tan</code>	<code>tan(x)</code>	Calcula a tangente de <code>x</code> ( <code>x</code> deve estar representado em radianos).

A linguagem C/C++ possui muitas outras funções matemáticas. Todas elas podem ser observadas detalhadamente na documentação da biblioteca `math.h`.

**PALAVRAS RESERVADAS DE C/C++:** são nomes utilizados pelo compilador para representar comandos de controle do programa, operadores e diretivas.

asm	else	_loadds	signed
auto	enum	long	sizeof
break	_es	_near	_ss
case	_export	near	static
catch	estern	new	struct
_cdecl	_far	operator	switch
cdecl	far	_pascal	template
char	float	pascal	this
class	for	private	typedef
const	friend	protected	union
continue	goto	public	unsigned
_cs	huge	register	virtual
default	if	return	void
do	inline	_saveregs	volatile
double	int	_seg	while
_ds	interrupt	short	

## EXERCÍCIOS RESOLVIDOS

1. Faça um programa que receba quatro números inteiros, calcule e mostre a soma desses números.

### ALGORITMO

```

ALGORITMO
DECLARE n1, n2, n3, n4, soma NUMÉRICO
LEIA n1, n2, n3, n4
soma ← n1 + n2 + n3 + n4
ESCREVA soma
FIM_ALGORITMO.

```

### RESOLUÇÃO PASCAL

#### 1<sup>a</sup> SOLUÇÃO:

\EXERC\CAP2\PASCAL\EX1\_A.PAS e \EXERC\CAP2\PASCAL\EX1\_A.EXE

#### 2<sup>a</sup> SOLUÇÃO:

\EXERC\CAP2\PASCAL\EX1\_B.PAS e \EXERC\CAP2\PASCAL\EX1\_B.EXE

### RESOLUÇÃO C/C++

#### 1<sup>a</sup> SOLUÇÃO:

\EXERC\CAP2\C++\EX1\_A.CPP e \EXERC\CAP2\C++\EX1\_A.EXE

#### 2<sup>a</sup> SOLUÇÃO:

\EXERC\CAP2\C++\EX1\_B.CPP e \EXERC\CAP2\C++\EX1\_B.EXE

 2. Faça um programa que receba três notas, calcule e mostre a média aritmética entre elas.

 1<sup>a</sup> SOLUÇÃO:

```
ALGORITMO
  DECLARE nota1, nota2, nota3, media NUMÉRICO
  LEIA nota1, nota2, nota3
  media ← (nota1 + nota2 + nota3) / 3
  ESCREVA media
FIM_ALGORITMO.
```

2<sup>a</sup> SOLUÇÃO:

```
ALGORITMO
  DECLARE nota1, nota2, nota3, soma, media NUMÉRICO
  LEIA nota1, nota2, nota3
  soma ← nota1 + nota2 + nota3
  media ← soma / 3
  ESCREVA media
FIM_ALGORITMO.
```

 1<sup>a</sup> SOLUÇÃO:

\EXERC\CAP2\PASCAL\EX2\_A.PAS e \EXERC\CAP2\PASCAL\EX2\_A.EXE

2<sup>a</sup> SOLUÇÃO:

\EXERC\CAP2\PASCAL\EX2\_B.PAS e \EXERC\CAP2\PASCAL\EX2\_B.EXE

Quando estamos trabalhando com tipos de dados reais, precisamos fazer a formatação desses números, pois se isso não for feito, esses números serão apresentados com formatação científica.

Exemplo de números com formatação científica:

1.5000000000E+03 = 15000  
7.0000000000E+00 = 7

Exemplo de formatação:

X:6:2 a variável X será mostrada com seis casas, sendo que dessas seis casas, duas casas para a parte decimal, uma casa para o ponto e as quatro casas restantes para a parte inteira.

Y:8:3 a variável Y será mostrada com oito casas, sendo que dessas oito casas, três casas para a parte decimal, uma casa para o ponto e as quatro casas restantes para a parte inteira.

*Variável: número total de casas: número de casas decimais*

O primeiro parâmetro da formatação corresponde ao número total de casas ocupadas pela variável, o segundo parâmetro corresponde ao total de casas ocupadas pela parte decimal. O ponto, que é o separador entre a parte inteira e decimal, também ocupa uma casa.

3<sup>a</sup> SOLUÇÃO:

\EXERC\CAP2\PASCAL\EX2\_C.PAS e \EXERC\CAP2\PASCAL\EX2\_C.EXE

**1<sup>a</sup> SOLUÇÃO:**

```
\EXERC\CAP2\C++\EX2_A.CPP e \EXERC\CAP2\C++\EX2_A.EXE
```

Quando estamos trabalhando com tipos de dados reais, precisamos fazer a formatação desses números para definir o número de casas decimais que devem ser mostradas.

Deve-se utilizar a função `setprecision`, conforme apresentada a seguir:

```
cout << setprecision(4);
```

No exemplo anterior, a formatação permitirá que sejam mostradas até quatro casas decimais. Para a utilização da função `setprecision`, deve-se incluir ao programa a biblioteca `iomanip.h` (`#include <iomanip.h>`).

Outra maneira de formatar a saída é substituir o comando `cout` pelo comando `printf`, como apresentado a seguir:

```
printf("Conteúdo de variável X é: %6.3f", X);
```

A formatação é especificada imediatamente antes da letra que define o tipo da variável que será mostrada (no exemplo acima, `%f` especifica que será mostrado um número real e `6.3` significa que serão utilizadas seis casas para mostrar o número e, dessas, uma será utilizada para mostrar o ponto e três outras para mostrar a parte fracionária do número).

**2<sup>a</sup> SOLUÇÃO:**

```
\EXERC\CAP2\C++\EX2_B.CPP e \EXERC\CAP2\C++\EX2_B.EXE
```

**3<sup>a</sup> SOLUÇÃO:**

```
\EXERC\CAP2\C++\EX2_C.CPP e \EXERC\CAP2\C++\EX2_C.EXE
```

3. Faça um programa que receba três notas e seus respectivos pesos, calcule e mostre a média ponderada dessas notas.

**1<sup>a</sup> SOLUÇÃO:**

## ALGORITMO

```
DECLARE nota1, nota2, nota3, peso1, peso2, peso3, media NUMÉRICO
LEIA nota1, nota2, nota3, peso1, peso2, peso3
media ← (nota1 * peso1 + nota2 * peso2 + nota3 * peso3) / (peso1 +
➥ peso2 + peso3)
ESCREVA media
FIM_ALGORITMO.
```

**2<sup>a</sup> SOLUÇÃO:**

## ALGORITMO

```
DECLARE nota1, nota2, nota3, peso1, peso2, peso3 NUMÉRICO
som1, som2, som3, total, media NUMÉRICO
LEIA nota1, nota2, nota3, peso1, peso2, peso3
som1 ← nota1 * peso1
som2 ← nota2 * peso2
som3 ← nota3 * peso3
total ← peso1 + peso2 + peso3
media ← (som1 + som2 + som3) / total
ESCREVA media
FIM_ALGORITMO.
```

**1<sup>A</sup> SOLUÇÃO:**

\EXERC\CAP2\PASCAL\EX3\_A.PAS e \EXERC\CAP2\PASCAL\EX3\_A.EXE

**2<sup>A</sup> SOLUÇÃO:**

\EXERC\CAP2\PASCAL\EX3\_B.PAS e \EXERC\CAP2\PASCAL\EX3\_B.EXE

**3<sup>A</sup> SOLUÇÃO:**

\EXERC\CAP2\PASCAL\EX3\_C.PAS e \EXERC\CAP2\PASCAL\EX3\_C.EXE

**1<sup>A</sup> SOLUÇÃO:**

\EXERC\CAP2\C++\EX3\_A.CPP e \EXERC\CAP2\C++\EX3\_A.EXE

**2<sup>A</sup> SOLUÇÃO:**

\EXERC\CAP2\C++\EX3\_B.CPP e \EXERC\CAP2\C++\EX3\_B.EXE

**3<sup>A</sup> SOLUÇÃO:**

\EXERC\CAP2\C++\EX3\_C.CPP e \EXERC\CAP2\C++\EX3\_C.EXE

- 4.** Faça um programa que receba o salário de um funcionário, calcule e mostre o novo salário, sabendo-se que este sofreu um aumento de 25%.

**1<sup>A</sup> SOLUÇÃO:**

```
ALGORITMO
DECLARE sal, novosal NUMÉRICO
LEIA sal
novosal ← sal + sal * 25/100
ESCREVA novosal
FIM_ALGORITMO.
```

**2<sup>A</sup> SOLUÇÃO:**

```
ALGORITMO
DECLARE sal, aumento, novosal NUMÉRICO
LEIA sal
aumento ← sal * 25/100
novosal ← sal + aumento
ESCREVA novosal
FIM_ALGORITMO.
```

**1<sup>A</sup> SOLUÇÃO:**

\EXERC\CAP2\PASCAL\EX4\_A.PAS e \EXERC\CAP2\PASCAL\EX4\_A.EXE

**2<sup>A</sup> SOLUÇÃO:**

\EXERC\CAP2\PASCAL\EX4\_B.PAS e \EXERC\CAP2\PASCAL\EX4\_B.EXE

**1<sup>A</sup> SOLUÇÃO:**

\EXERC\CAP2\C++\EX4\_A.CPP e \EXERC\CAP2\C++\EX4\_A.EXE

**2<sup>a</sup> SOLUÇÃO:**

\EXERC\CAP2\C++\EX4\_B.CPP e \EXERC\CAP2\C++\EX4\_B.EXE

-  5. Faça um programa que receba o salário de um funcionário e o percentual de aumento, calcule e mostre o valor do aumento e o novo salário.

**ALGORITMO****SOLUÇÃO:**

```

ALGORITMO
DECLARE sal, perc, aumento, novosal NUMÉRICO
LEIA sal, perc
aumento ← sal * perc/100
ESCREVA aumento
novosal ← sal + aumento
ESCREVA novosal
FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP2\PASCAL\EX5.PAS e \EXERC\CAP2\PASCAL\EX5.EXE

**SOLUÇÃO:**

\EXERC\CAP2\C++\EX5.CPP e \EXERC\CAP2\C++\EX5.EXE

-  6. Faça um programa que receba o salário-base de um funcionário, calcule e mostre o salário a receber, sabendo-se que esse funcionário tem gratificação de 5% sobre o salário-base e paga imposto de 7% sobre o salário-base.

**ALGORITMO****SOLUÇÃO:**

```

ALGORITMO
DECLARE sal, salreceive, grat, imp NUMÉRICO
LEIA sal
grat ← sal * 5/100
imp ← sal * 7/100
salreceive ← sal + grat - imp
ESCREVA salreceive
FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP2\PASCAL\EX6.PAS e \EXERC\CAP2\PASCAL\EX6.EXE

**SOLUÇÃO:**

\EXERC\CAP2\C++\EX6.CPP e \EXERC\CAP2\C++\EX6.EXE

-  7. Faça um programa que receba o salário-base de um funcionário, calcule e mostre o seu salário a receber, sabendo-se que esse funcionário tem gratificação de R\$ 50,00 e paga imposto de 10% sobre o salário-base.

**ALGORITMO****SOLUÇÃO:**

```

ALGORITMO
DECLARE sal, salreceive, imp NUMÉRICO
LEIA sal

```

```

imp ← sal * 10/100
salreceber ← sal + 50 - imp
ESCREVA salreceber
FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP2\PASCAL\EX7.PAS e \EXERC\CAP2\PASCAL\EX7.EXE

**SOLUÇÃO:**

\EXERC\CAP2\C++\EX7.CPP e \EXERC\CAP2\C++\EX7.EXE

- 8.** Faça um programa que receba o valor de um depósito e o valor da taxa de juros, calcule e mostre o valor do rendimento e o valor total depois do rendimento.

**SOLUÇÃO:**

```

ALGORITMO
DECLARE dep, taxa, rend, total NUMÉRICO
LEIA dep, taxa
rend ← dep * taxa/100
total ← dep + rend
ESCREVA rend
ESCREVA total
FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP2\PASCAL\EX8.PAS e \EXERC\CAP2\PASCAL\EX8.EXE

**SOLUÇÃO:**

\EXERC\CAP2\C++\EX8.CPP e \EXERC\CAP2\C++\EX8.EXE

- 9.** Faça um programa que calcule e mostre a área de um triângulo.

Sabe-se que: Área = (base \* altura)/2

**SOLUÇÃO:**

```

ALGORITMO
DECLARE base, altura, area NUMÉRICO
LEIA base, altura
area ← (base * altura)/2
ESCREVA area
FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP2\PASCAL\EX9.PAS e \EXERC\CAP2\PASCAL\EX9.EXE

**SOLUÇÃO:**

\EXERC\CAP2\C++\EX9.CPP e \EXERC\CAP2\C++\EX9.EXE

 10. Faça um programa que calcule e mostre a área de um círculo.

Sabe-se que: Área =  $\pi R^2$



#### SOLUÇÃO:

```
ALGORITMO
DECLARE area, raio NUMÉRICO
LEIA raio
area ← 3.1415 * raio2
ESCREVA area
FIM_ALGORITMO.
```



#### 1<sup>a</sup> SOLUÇÃO:

\EXERC\CAP2\PASCAL\EX10\_A.PAS e \EXERC\CAP2\PASCAL\EX10\_A.EXE

#### 2<sup>a</sup> SOLUÇÃO:

\EXERC\CAP2\PASCAL\EX10\_B.PAS e \EXERC\CAP2\PASCAL\EX10\_B.EXE

Esse programa usou algumas funções predefinidas da linguagem PASCAL que estão descritas na Seção 2.2.6.



#### 1<sup>a</sup> SOLUÇÃO:

\EXERC\CAP2\C++\EX10\_A.CPP e \EXERC\CAP2\C++\EX10\_A.EXE

#### 2<sup>a</sup> SOLUÇÃO:

\EXERC\CAP2\C++\EX10\_B.CPP e \EXERC\CAP2\C++\EX10\_B.EXE

Esse programa usou algumas funções predefinidas da linguagem C/C++ que estão descritas na Seção 2.3.6.

 11. Faça um programa que receba um número positivo e maior que zero, calcule e mostre:

- o número digitado ao quadrado;
- o número digitado ao cubo;
- a raiz quadrada do número digitado;
- a raiz cúbica do número digitado.



#### SOLUÇÃO:

```
ALGORITMO
DECLARE num, quad, cubo, r2, r3 NUMÉRICO
LEIA num
quad ← num2
cubo ← num3
r2 ← √num
r3 ← ∛num
ESCREVA quad, cubo, r2, r3
FIM_ALGORITMO.
```



#### SOLUÇÃO:

\EXERC\CAP2\PASCAL\EX11.PAS e \EXERC\CAP2\PASCAL\EX11.EXE

Esse programa usou algumas funções predefinidas da linguagem PASCAL que estão descritas na Seção 2.2.6.

**SOLUÇÃO:**

\EXERC\CAP2\C++\EX11.CPP e \EXERC\CAP2\C++\EX11.EXE

Esse programa usou algumas funções predefinidas da linguagem C/C++ que estão descritas na Seção 2.3.6.

- 12.** Faça um programa que receba dois números maiores que zero, calcule e mostre um elevado ao outro.

**ALGORITMO****SOLUÇÃO:**

```
ALGORITMO
DECLARE num1, num2, r1, r2 NUMÉRICO
LEIA num1, num2
r1 ← num1 num2
r2 ← num2 num1
ESCREVA r1, r2
FIM_ALGORITMO.
```

**SOLUÇÃO:**

\EXERC\CAP2\PASCAL\EX12.PAS e \EXERC\CAP2\PASCAL\EX12.EXE

Este programa usou algumas funções predefinidas da linguagem PASCAL que estão descritas na Seção 2.2.6.

**SOLUÇÃO:**

\EXERC\CAP2\C++\EX12.CPP e \EXERC\CAP2\C++\EX12.EXE

Esse programa usou algumas funções predefinidas da linguagem C/C++ que estão descritas na Seção 2.3.6.

- 13.** Sabe-se que:

1 pé = 12 polegadas

1 jarda = 3 pés

1 milha = 1.760 jardas

Faça um programa que receba uma medida em pés, faça as conversões a seguir e mostre os resultados.

- polegadas;
- jardas;
- milhas.

**SOLUÇÃO:**

```
ALGORITMO
DECLARE pes, polegadas, jardas, milhas NUMÉRICO
LEIA pes
polegadas ← pes * 12
jardas ← pes / 3
milhas ← jardas / 1760
ESCREVA polegadas, jardas, milhas
FIM_ALGORITMO.
```

**SOLUÇÃO:**

\EXERC\CAP2\PASCAL\EX13.PAS e \EXERC\CAP2\PASCAL\EX13.EXE

**SOLUÇÃO:**

\EXERC\CAP2\C++\EX13.CPP e \EXERC\CAP2\C++\EX13.EXE

- 14.** Faça um programa que receba o ano de nascimento de uma pessoa e o ano atual, calcule e mostre:

- a idade dessa pessoa;
- quantos anos essa pessoa terá em 2005.

**SOLUÇÃO:**

```
ALGORITMO
  DECLARE ano_atual, ano_nascimento, idade_atual, idade_2005
  ↪ NUMÉRICO
  LEIA ano_atual
  LEIA ano_nascimento
  idade_atual ← ano_atual - ano_nascimento
  idade_2005 ← 2005 - ano_nascimento
  ESCREVA idade_atual
  ESCREVA idade_2005
FIM_ALGORITMO.
```

**SOLUÇÃO:**

\EXERC\CAP2\PASCAL\EX14.PAS e \EXERC\CAP2\PASCAL\EX14.EXE

**SOLUÇÃO:**

\EXERC\CAP2\C++\EX14.CPP e \EXERC\CAP2\C++\EX14.EXE

- 15.** O custo ao consumidor de um carro novo é a soma do preço de fábrica com o percentual de lucro do distribuidor e dos impostos aplicados ao preço de fábrica. Faça um programa que receba o preço de fábrica de um veículo, o percentual de lucro do distribuidor e o percentual de impostos. Calcule e mostre:

- o valor correspondente ao lucro do distribuidor;
- o valor correspondente aos impostos;
- o preço final do veículo.

**SOLUÇÃO:**

```
ALGORITMO
  DECLARE p_fab, perc_d, perc_i, vlr_d, vlr_i, p_final NUMÉRICO
  LEIA p_fab
  LEIA perc_d
  LEIA perc_i
  vlr_d ← p_fab * perc_d / 100
  vlr_i ← p_fab * perc_i / 100
  p_final ← p_fab + vlr_d + vlr_i
  ESCREVA vlr_d
  ESCREVA vlr_i
  ESCREVA p_final
FIM_ALGORITMO.
```

**SOLUÇÃO:**

\EXERC\CAP2\PASCAL\EX15.PAS e \EXERC\CAP2\PASCAL\EX15.EXE

**SOLUÇÃO:**

\EXERC\CAP2\C++\EX15.CPP e \EXERC\CAP2\C++\EX15.EXE

**16.** Faça um programa que receba o número de horas trabalhadas e o valor do salário mínimo. Calcule e mostre o salário a receber seguindo as regras abaixo:

- a hora trabalhada vale a metade do salário mínimo;
- o salário bruto equivale ao número de horas trabalhadas multiplicado pelo valor da hora trabalhada;
- o imposto equivale a 3% do salário bruto;
- o salário a receber equivale ao salário bruto menos o imposto.

**SOLUÇÃO:****ALGORITMO**

```

DECLARE horas_t, vlr_sal_min, vlr_hora_t NUMÉRICO
  vlr_sal_bru, imp, vlr_sal_liq
LEIA horas_t
LEIA vlr_sal_min
  vlr_hora_t ← vlr_sal / 2
  vlr_sal_bru ← vlr_hora_t * horas_t
  imp ← vlr_sal_bru * 3 / 100
  vlr_sal_liq ← vlr_sal_bru - imp
ESCREVA vlr_sal_liq
FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP2\PASCAL\EX16.PAS e \EXERC\CAP2\PASCAL\EX16.EXE

**SOLUÇÃO:**

\EXERC\CAP2\C++\EX16.CPP e \EXERC\CAP2\C++\EX16.EXE

**17.** Um trabalhador recebeu seu salário e o depositou em sua conta corrente bancária. Esse trabalhador emitiu dois cheques e agora deseja saber seu saldo atual. Sabe-se que cada operação bancária de retirada paga CPMF de 0,38% e o saldo inicial da conta está zerado.

**SOLUÇÃO:****ALGORITMO**

```

DECLARE salario, cheque1, cheque2, cpmf1, cpmf2, saldo NUMÉRICO
LEIA salario
LEIA cheque1
LEIA cheque2
  cpmf1 ← cheque1 * 0.38 / 100
  cpmf2 ← cheque2 * 0.38 / 100
  saldo ← salario - cheque1 - cheque2 - cpmf1 - cpmf2
ESCREVA saldo
FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP2\PASCAL\EX17.PAS e \EXERC\CAP2\PASCAL\EX17.EXE

**SOLUÇÃO:**

\EXERC\CAP2\C++\EX17.CPP e \EXERC\CAP2\C++\EX17.EXE

- 18.** Pedro comprou um saco de ração com peso em quilos. Pedro possui dois gatos para os quais fornece a quantidade de ração em gramas. Faça um programa que receba o peso do saco de ração e a quantidade de ração fornecida para cada gato. Calcule e mostre quanto restará de ração no saco após cinco dias.

**SOLUÇÃO:**

```
ALGORITMO
DECLARE peso_saco, racao_gato1, racao_gato2, total_final NUMÉRICO
LEIA peso_saco
LEIA racao_gato1
LEIA racao_gato2
racao_gato1 ← racao_gato1 / 1000
racao_gato2 ← racao_gato2 / 1000
total_final ← peso_saco - 5 * (racao_gato1 + racao_gato2)
ESCREVA total_final
FIM_ALGORITMO.
```

**SOLUÇÃO:**

\EXERC\CAP2\PASCAL\EX18.PAS e \EXERC\CAP2\PASCAL\EX18.EXE

**SOLUÇÃO:**

\EXERC\CAP2\C++\EX18.CPP e \EXERC\CAP2\C++\EX18.EXE

- 19.** Cada degrau de uma escada tem X de altura. Faça um programa que receba essa altura e a altura que o usuário deseja alcançar subindo a escada. Calcule e mostre quantos degraus o usuário deverá subir para atingir seu objetivo, sem se preocupar com a altura do usuário.

**SOLUÇÃO:**

```
ALGORITMO
DECLARE a_degrau, a_usuario, qtd_degraus NUMÉRICO
LEIA a_degrau
LEIA a_usuario
qtd_degraus ← a_usuario / a_degrau
ESCREVA qtd_degraus
FIM_ALGORITMO.
```

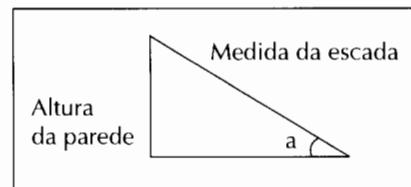
**SOLUÇÃO:**

\EXERC\CAP2\PASCAL\EX19.PAS e \EXERC\CAP2\PASCAL\EX19.EXE

**SOLUÇÃO:**

\EXERC\CAP2\C++\EX19.CPP e \EXERC\CAP2\C++\EX19.EXE

-  **20.** Faça um programa que receba a medida do ângulo formado por uma escada apoiada no chão e encostada na parede e a altura da parede onde está a ponta da escada. Calcule e mostre a medida desta escada.

**ALGORITMO****SOLUÇÃO:**

## ALGORITMO

```
DECLARE ang, alt, escada, radiano NUMÉRICO
LEIA ang
LEIA alt
radiano ← ang * 3.14 / 180
escada ← alt / seno(radiano)
ESCREVA escada
FIM_ALGORITMO.
```

**SOLUÇÃO:**

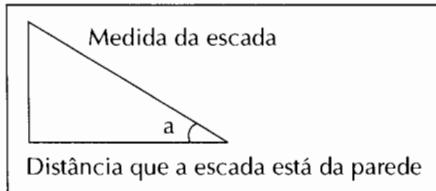
```
\EXERC\CAP2\PASCAL\EX20.PAS e \EXERC\CAP2\PASCAL\EX20.EXE
```

**SOLUÇÃO:**

```
\EXERC\CAP2\C++\EX20.CPP e \EXERC\CAP2\C++\EX20.EXE
```

-  **21.** Faça um programa para calcular e mostrar a que distância deve estar uma escada da parede. O usuário deve fornecer o tamanho da escada e a altura em que deseja pregar o quadro.

Lembre-se de que o tamanho da escada deve ser maior que a altura que se deseja alcançar.



X – Altura em que deseja pregar o quadro  
Y – Distância em que deverá ficar a escada  
Z – Tamanho da escada

**ALGORITMO****SOLUÇÃO:**

## ALGORITMO

```
DECLARE X, Y, Z NUMÉRICO
LEIA Z
LEIA X
Y ← Z2 - X2
Y ← ∛Y
ESCREVA Y
FIM_ALGORITMO.
```

**SOLUÇÃO:**

```
\EXERC\CAP2\PASCAL\EX21.PAS e \EXERC\CAP2\PASCAL\EX21.EXE
```

**SOLUÇÃO:**

```
\EXERC\CAP2\C++\EX21.CPP e \EXERC\CAP2\C++\EX21.EXE
```

 **22.** Sabe-se que o quilowatt de energia custa um quinto do salário mínimo. Faça um programa que receba o valor do salário mínimo e a quantidade de quilowatts consumida por uma residência. Calcule e mostre:

- o valor, em reais, de cada quilowatt;
- o valor, em reais, a ser pago por essa residência;
- o valor, em reais, a ser pago com desconto de 15%.

**ALGORITMO****SOLUÇÃO:**

```
ALGORITMO
DECLARE vlr_sal, qtd_kw, vlr_kw, vlr_reais, desc, vlr_desc
  ↪ NUMÉRICO
LEIA vlr_sal
LEIA qtd_kw
vlr_kw ← vlr_sal / 5
vlr_reais ← vlr_kw * qtd_kw
desc ← vlr_reais * 15 / 100
vlr_desc ← vlr_reais - desc
ESCREVA vlr_kw
ESCREVA vlr_reais
ESCREVA vlr_desc
FIM_ALGORITMO.
```

**SOLUÇÃO:**

\EXERC\CAP2\PASCAL\EX22.PAS e \EXERC\CAP2\PASCAL\EX22.EXE

**SOLUÇÃO:**

\EXERC\CAP2\C++\EX22.CPP e \EXERC\CAP2\C++\EX22.EXE

 **23.** Faça um programa que receba um número real, calcule e mostre:

- a parte inteira desse número;
- a parte fracionária desse número;
- o arredondamento desse número.

**ALGORITMO****SOLUÇÃO:**

```
ALGORITMO
DECLARE num, i, f, a NUMÉRICO
LEIA num
i ← trunc(num)
f ← num - i
a ← arredonda (num)
ESCREVA i
ESCREVA f
ESCREVA a
FIM_ALGORITMO.
```

**SOLUÇÃO (ARREDONDANDO O NÚMERO COMO NA MATEMÁTICA):**

\EXERC\CAP2\PASCAL\EX23.PAS e \EXERC\CAP2\PASCAL\EX23.EXE

**SOLUÇÃO (ARREDONDANDO O NÚMERO PARA CIMA):**

\EXERC\CAP2\C++\EX23\_A.CPP e \EXERC\CAP2\C++\EX23\_A.EXE

**SOLUÇÃO (ARREDONDANDO O NÚMERO PARA BAIXO):**

\EXERC\CAP2\C++\EX23\_B.CPP e \EXERC\CAP2\C++\EX23\_B.EXE

-  **24.** Faça um programa que receba uma hora formada por hora e minutos (um número real), calcule e mostre a hora digitada apenas em minutos. Lembre-se de que:

- ◆ para quatro e meia deve-se digitar 4.30;
- ◆ os minutos vão de 0 a 60.

**SOLUÇÃO:**

```
ALGORITMO
DECLARE hora, h, m, conversao NUMÉRICO
LEIA hora
h ← trunc(hora)
m ← hora - h
conversao ← (h * 60) + (m * 100)
ESCREVA conversao
FIM_ALGORITMO.
```

**SOLUÇÃO:**

\EXERC\CAP2\PASCAL\EX24.PAS e \EXERC\CAP2\PASCAL\EX24.EXE

**SOLUÇÃO:**

\EXERC\CAP2\C++\EX24.CPP e \EXERC\CAP2\C++\EX24.EXE

-  **25.** Faça um programa que receba o custo de um espetáculo teatral e o preço do convite desse espetáculo. Esse programa deve calcular e mostrar a quantidade de convites que devem ser vendidos para que pelo menos o custo do espetáculo seja alcançado.

**SOLUÇÃO:**

```
ALGORITMO
DECLARE custo, convite, qtd NUMÉRICO
LEIA custo
LEIA convite
qtd ← custo / convite
ESCREVA qtd
FIM_ALGORITMO.
```

**SOLUÇÃO:**

\EXERC\CAP2\PASCAL\EX25.PAS e \EXERC\CAP2\PASCAL\EX25.EXE

**SOLUÇÃO:**

\EXERC\CAP2\C++\EX25.CPP e \EXERC\CAP2\C++\EX25.EXE

## EXERCÍCIOS PROPOSTOS

1. Faça um programa que receba dois números, calcule e mostre a subtração do primeiro número pelo segundo.

2. Faça um programa que receba três números, calcule e mostre a multiplicação desses números.
3. Faça um programa que receba dois números, calcule e mostre a divisão do primeiro número pelo segundo. Sabe-se que o segundo número não pode ser zero, portanto não é necessário se preocupar com validações.
4. Faça um programa que receba duas notas, calcule e mostre a média ponderada dessas notas, considerando peso 2 para a primeira nota e peso 3 para a segunda nota.
5. Faça um programa que receba o preço de um produto, calcule e mostre o novo preço, sabendo-se que este sofreu um desconto de 10%.
6. Um funcionário recebe um salário fixo mais 4% de comissão sobre as vendas. Faça um programa que receba o salário fixo de um funcionário e o valor de suas vendas, calcule e mostre a comissão e o salário final do funcionário.
7. Faça um programa que receba o peso de uma pessoa, calcule e mostre:
  - a) o novo peso se a pessoa engordar 15% sobre o peso digitado;
  - b) o novo peso se a pessoa emagrecer 20% sobre o peso digitado.
8. Faça um programa que receba o peso de uma pessoa em quilos, calcule e mostre esse peso em gramas.
9. Faça um programa que calcule e mostre a área de um trapézio.  
Sabe-se que:  $A = ((\text{base maior} + \text{base menor}) * \text{altura})/2$
10. Faça um programa que calcule e mostre a área de um quadrado.  
Sabe-se que:  $A = \text{lado} * \text{lado}$
11. Faça um programa que calcule e mostre a área de um losango.  
Sabe-se que:  $A = (\text{diagonal maior} * \text{diagonal menor})/2$
12. Faça um programa que receba o valor do salário mínimo e o valor do salário de um funcionário, calcule e mostre a quantidade de salários mínimos que ganha esse funcionário.
13. Faça um programa que calcule e mostre a tabuada de um número digitado pelo usuário.
14. Faça um programa que receba o ano de nascimento de uma pessoa e o ano atual, calcule e mostre:
  - a) a idade dessa pessoa em anos;
  - b) a idade dessa pessoa em meses;
  - c) a idade dessa pessoa em dias;
  - d) a idade dessa pessoa em semanas.
15. João recebeu seu salário e precisa pagar duas contas que estão atrasadas. Como as contas estão atrasadas, João terá de pagar multa de 2% sobre cada conta. Faça um programa que calcule e mostre quanto restará do salário do João.
16. Faça um programa que receba o valor dos catetos de um triângulo, calcule e mostre o valor da hipotenusa.

**17.** Faça um programa que receba o raio, calcule e mostre:

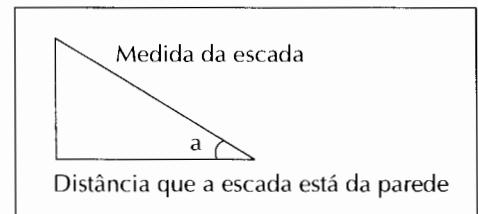
- o comprimento de uma esfera, sabe-se que  $C = 2\pi R$ ;
- a área de uma esfera, sabe-se que  $A = \pi R^2$ ;
- o volume de uma esfera, sabe-se que  $V = \frac{4}{3}\pi R^3$ .

**18.** Faça um programa que receba uma temperatura em Celsius, calcule e mostre essa temperatura em Fahrenheit.

Sabe-se que  $F = 180(C + 32)/100$ .

**19.** Sabe-se que para iluminar de maneira correta os cômodos de uma casa, para cada  $m^2$ , deve-se usar 18 W de potência. Faça um programa que receba as duas dimensões de um cômodo (em metros), calcule e mostre a sua área (em  $m^2$ ) e a potência de iluminação que deverá ser utilizada.

**20.** Faça um programa que receba a medida do ângulo formado por uma escada apoiada no chão e a distância que a escada está da parede. Calcule e mostre a medida da escada para que se possa alcançar a ponta da escada.



**21.** Faça um programa que receba o número de horas trabalhadas, o valor do salário mínimo e o número de horas extras trabalhadas. Calcule e mostre o salário a receber seguindo as regras a seguir:

- a hora trabalhada vale  $\frac{1}{8}$  do salário mínimo;
- a hora extra vale  $\frac{1}{4}$  do salário mínimo;
- o salário bruto equivale ao número de horas trabalhadas multiplicado pelo valor da hora trabalhada;
- a quantia a receber pelas horas extras equivale ao número de horas extras trabalhadas multiplicado pelo valor da hora extra;
- o salário a receber equivale ao salário bruto mais a quantia a receber pelas horas extras.

**22.** Faça um programa que receba o número de lados de um polígono convexo, calcule e mostre o número de diagonais desse polígono, onde  $N$  é o número de lados do polígono. Sabe-se que  $ND = N(N - 3)/2$ .

**23.** Faça um programa que receba a medida de dois ângulos de um triângulo, calcule e mostre a medida do terceiro ângulo. Sabe-se que a soma dos ângulos de um triângulo é 180.

**24.** Faça um programa que receba a quantidade de dinheiro em reais que uma pessoa que vai viajar possui. Essa pessoa vai passar por vários países e precisa converter seu dinheiro em dólares, marco alemão e libra esterlina. Sabe-se que a cotação do dólar é de R\$ 1,80, do marco alemão é de R\$ 2,00 e da libra esterlina é de R\$ 1,57. O programa deve fazer as conversões e mostrá-las.

**25.** Faça um programa que receba uma hora (uma variável para hora e outra para minutos), calcule e mostre:

- a hora digitada convertida em minutos;
- o total dos minutos, ou seja, os minutos digitados mais a conversão anterior;
- o total dos minutos convertidos em segundos.

## CAPÍTULO

# 3

# ESTRUTURA CONDICIONAL

## 3.1 ESTRUTURA CONDICIONAL EM ALGORITMOS

### 3.1.1 ESTRUTURA CONDICIONAL SIMPLES

```
SE condição  
ENTÃO comando
```

O **comando** só será executado se a **condição** for verdadeira. Uma condição é uma comparação que possui dois valores possíveis, verdadeiro ou falso.

```
SE condição  
ENTÃO INÍCIO  
    comando1  
    comando2  
    comando3  
FIM
```

Os **comandos 1, 2 e 3** só serão executados se a **condição** for verdadeira. As palavras **INÍCIO** e **FIM** serão necessárias apenas quando dois ou mais comandos forem executados.

### 3.1.2 ESTRUTURA CONDICIONAL COMPOSTA

```
SE condição  
ENTÃO comando1  
SENÃO comando2
```

Se a **condição** for verdadeira, será executado o **comando1**; caso contrário, se a **condição** for falsa, será executado o **comando2**.

```
SE condição  
ENTÃO INÍCIO  
    comando1  
    comando2  
    FIM  
SENÃO INÍCIO  
    comando3  
    comando4  
    FIM
```

Se a **condição** for verdadeira, o **comando1** e o **comando2** serão executados; caso contrário, se a **condição** for falsa, o **comando3** e o **comando4** serão executados.

## 3.2 ESTRUTURA CONDICIONAL EM PASCAL

### 3.2.1 ESTRUTURA CONDICIONAL SIMPLES

```
IF condição
THEN comando;
```

O **comando** só será executado se a **condição** for verdadeira. Uma condição é uma comparação que possui dois valores possíveis: verdadeiro ou falso.

```
IF condição
THEN BEGIN
    comando1;
    comando2;
    comando3;
END;
```

Os **comandos 1, 2 e 3** só serão executados se a **condição** for verdadeira.

### 3.2.2 ESTRUTURA CONDICIONAL COMPOSTA

```
IF condição
THEN comando1
ELSE comando2;
```

Se a **condição** for verdadeira, será executado o **comando1**; caso contrário, se a **condição** for falsa, será executado o **comando2**.

```
IF condição
THEN BEGIN
    comando1;
    comando2;
END
ELSE BEGIN
    comando3;
    comando4;
END;
```

Se a **condição** for verdadeira, o **comando1** e o **comando2** serão executados; caso contrário, se a **condição** for falsa, o **comando3** e o **comando4** serão executados.

**OBSEVAÇÃO:** Antes do comando ELSE não existe ponto-e-vírgula.

### 3.2.3 ESTRUTURA CASE

Em alguns programas existem comandos mutuamente exclusivos, isto é, se um comando for executado, os demais não serão. Quando este for o caso, um comando seletivo é o mais indicado e esse comando seletivo em PASCAL tem a seguinte sintaxe:

```
CASE seletor OF
    lista de alvos1: comando1;
    lista de alvos2: comando2;
    alvo3: comando3;
    alvo4: BEGIN
        comando4;
        comando5;
    END;
END;
```

Se o **seletor** atingir a lista de alvos1, o **comando1** será executado; se o **seletor** atingir a lista de alvos2, o **comando2** será executado; se o **seletor** atingir o alvo3, o **comando3** será executado ou se o **seletor** atingir o alvo4 então o **comando4** e o **comando5** serão executados. Se nenhum alvo for atingido, nada será executado.

```
CASE seletor OF
    lista de alvos1: BEGIN
        comando1;
        comando2;
    END;
    lista de alvos2: comando3;
ELSE comando4;
END;
```

Se o **seletor** atingir a lista de alvos1, o **comando1** e o **comando2** serão executados; se o **seletor** atingir a lista de alvos2, o **comando3** será executado. Se nenhum alvo for atingido, será executado o **comando4**.

**OBSERVAÇÃO:** A restrição da estrutura CASE é que o **seletor** só pode ser uma variável do tipo CHAR, INTEGER ou BOOLEAN.

### 3.2.4 OPERADORES LÓGICOS

Os operadores lógicos são: AND, OR e NOT que significam *e*, *ou*, *não* e são usados para conjunção, disjunção e negação respectivamente.

**TABELA E**

V e V = V
V e F = F
F e V = F
F e F = F

**TABELA OU**

V ou V = V
V ou F = V
F ou V = V
F ou F = F

**TABELA NÃO**

Não V = F
Não F = V

**OBSERVAÇÃO:** Quando existe mais de uma condição, essas devem estar entre parênteses.

Exemplo:

```
IF (X > 5) AND (X < 10)
THEN WRITELN('Número entre 5 e 10');
```

## 3.3 ESTRUTURA CONDICIONAL EM C/C++

### 3.3.1 ESTRUTURA CONDICIONAL SIMPLES

```
if condição
    comando;
```

O **comando** só será executado se a **condição** for verdadeira. Uma **condição** é uma comparação que possui dois valores possíveis: verdadeiro ou falso.

```
if condição
{ comando1;
  comando2;
  comando3;
}
```

Em C/C++, torna-se obrigatória a utilização de chaves quando houver mais de um comando a ser executado. Os comandos entre as chaves {} só serão executados se a **condição** for verdadeira.

**OBSERVAÇÃO:** Todas as condições devem estar entre parênteses.

### 3.3.2 - ESTRUTURA CONDICIONAL COMPOSTA

```
if condição
    comando1;
else comando2;
```

Se a **condição** for verdadeira, será executado o **comando1**; caso contrário, se a **condição** for falsa, será executado o **comando2**.

```
if condição
{
    comando1;
    comando2;
}
else {
    comando3;
    comando4;
}
```

Se a **condição** for verdadeira, o **comando1** e o **comando2** serão executados; caso contrário, se a **condição** for falsa, o **comando3** e o **comando4** serão executados.

**OBSERVAÇÃO:** Todas as condições devem estar entre parênteses.

### 3.3.3 ESTRUTURA CASE

Em alguns programas existem comandos mutuamente exclusivos, isto é, se um comando for executado, os demais não o serão. Quando esse for o caso, um comando seletivo é o mais indicado e esse comando seletivo em C/C++ tem a seguinte sintaxe:

```
switch (variável)
{
    case valor1: lista de comandos;
        break;
    case valor2: lista de comandos;
        break;
    ...
    default: lista de comandos;
}
```

O comando **switch** (**variável**) avalia o valor de uma variável para decidir qual **case** será executado.

Cada **case** está associado a um possível valor da variável.

O comando **break** deve ser utilizado para impedir que sejam executados os comandos definidos nos **cases** subsequentes.

Quando o valor da variável não coincidir com aqueles especificados nos **cases**, será executado então o **default**.

### 3.3.4 OPERADORES LÓGICOS

Os operadores lógicos são: **&&**, **||** e **!** significam *e*, *ou*, *não* e são usados para conjunção, disjunção e negação, respectivamente.

**TABELA E**

V e V = V
V e F = F
F e V = F
F e F = F

**TABELA OU**

V ou V = V
V ou F = V
F ou V = V
F ou F = F

**TABELA NÃO**

Não V = F
Não F = V

**OBSERVAÇÃO:** Quando existe mais de uma condição, essas devem estar entre parênteses assim como todas as condições.

Exemplo:

```
if ((X > 5) && (X < 10))
cout << "Número entre 5 e 10";
```

## EXERCÍCIOS RESOLVIDOS

1. A nota final de um estudante é calculada a partir de três notas atribuídas respectivamente a um trabalho de laboratório, a uma avaliação semestral e a um exame final. A média das três notas mencionadas anteriormente obedece aos pesos a seguir:

NOTA	PESO
Trabalho de laboratório	2
Avaliação semestral	3
Exame final	5

Faça um programa que receba as três notas, calcule e mostre a média ponderada e o conceito que segue a tabela abaixo:

MÉDIA PONDERADA	CONCEITO
8,0	A
7,0	B
6,0	C
5,0	D
0,0	E

### ALGORITMO

### SOLUÇÃO:

```

ALGORITMO
DECLARE nota_trab, aval_sem, exame, media NUMÉRICO
ESCREVA "Digite a nota do trabalho em laboratório: "
LEIA nota_trab
ESCREVA "Digite a nota da avaliação semestral: "
LEIA aval_sem
ESCREVA "Digite a nota do exame final: "
LEIA exame
media ← (nota_trab * 2 + aval_sem * 3 + exame * 5) / 10
ESCREVA "Média ponderada: ", media
SE (media >= 8) E (media <= 10)
    ENTÃO ESCREVA "Obteve conceito A"
SE (media >= 7) E (media < 8)
    ENTÃO ESCREVA "Obteve conceito B"
SE (media >= 6) E (media < 7)
    ENTÃO ESCREVA "Obteve conceito C"
SE (media >= 5) E (media < 6)
    ENTÃO ESCREVA "Obteve conceito D"
SE (media >= 0) E (media < 5)
    ENTÃO ESCREVA "Obteve conceito E"
FIM_ALGORITMO.
```


**1<sup>a</sup> SOLUÇÃO - UTILIZANDO ESTRUTURA CONDICIONAL SIMPLES:**

\EXERC\CAP3\PASCAL\EX1\_A.PAS e \EXERC\CAP3\PASCAL\EX1\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:**

\EXERC\CAP3\PASCAL\EX1\_B.PAS e \EXERC\CAP3\PASCAL\EX1\_B.EXE


**1<sup>a</sup> SOLUÇÃO - UTILIZANDO ESTRUTURA CONDICIONAL SIMPLES:**

\EXERC\CAP3\C++\EX1\_A.CPP e \EXERC\CAP3\C++\EX1\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:**

\EXERC\CAP3\C++\EX1\_B.CPP e \EXERC\CAP3\C++\EX1\_B.EXE

- 2.** Faça um programa que receba três notas de um aluno, calcule e mostre a média aritmética e a mensagem que segue a tabela abaixo. Para alunos de exame, calcule e mostre a nota que deverá ser tirada no exame para aprovação, considerando que a média no exame é 6,0.

<b>MÉDIA ARITMÉTICA</b>			<b>MENSAGEM</b>
0,0	● —○	3,0	Reprovado
3,0	● —○	7,0	Exame
7,0	● —●	10,0	Aprovado


**SOLUÇÃO:**

```

ALGORITMO
DECLARE nota1, nota2, nota3, media, nota_exame NUMÉRICO
ESCREVA "Digite a primeira nota: "
LEIA nota1
ESCREVA "Digite a segunda nota: "
LEIA nota2
ESCREVA "Digite a terceira nota: "
LEIA nota3
media ← (nota1 + nota2 + nota3) / 3
ESCREVA "Média aritmética: ", media
SE (media >= 0) E (media < 3)
    ENTÃO ESCREVA "Reprovado"
SE (media >= 3) E (media < 7)
    ENTÃO INÍCIO
        ESCREVA "Exame"
        nota_exame ← 12 - media;
        ESCREVA "Deve tirar nota ", nota_exame, " para ser aprovado"
    FIM
SE (media >= 7) E (media < 10)
    ENTÃO ESCREVA "Aprovado"
FIM_ALGORITMO.

```


**1<sup>a</sup> SOLUÇÃO - UTILIZANDO ESTRUTURA CONDICIONAL SIMPLES:**

\EXERC\CAP3\PASCAL\EX2\_A.PAS e \EXERC\CAP3\PASCAL\EX2\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:**

\EXERC\CAP3\PASCAL\EX2\_B.PAS e \EXERC\CAP3\PASCAL\EX2\_B.EXE

**1<sup>a</sup> SOLUÇÃO - UTILIZANDO ESTRUTURA CONDICIONAL SIMPLES:**

\EXERC\CAP3\C++\EX2\_A.CPP e \EXERC\CAP3\C++\EX2\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:**

\EXERC\CAP3\C++\EX2\_B.CPP e \EXERC\CAP3\C++\EX2\_B.EXE

- 3.** Faça um programa que receba dois números e mostre o maior.

**SOLUÇÃO:****ALGORITMO**

```

DECLARE num1, num2 NUMÉRICO
ESCREVA "Digite o primeiro número: "
LEIA num1
ESCREVA "Digite o segundo número: "
LEIA num2
SE num1 > num2
  ENTÃO ESCREVA "O maior número é: ", num1
SE num2 > num1
  ENTÃO ESCREVA "O maior número é: ", num2
SE num1 = num2
  ENTÃO ESCREVA "Os números são iguais "
FIM_ALGORITMO.

```

**1<sup>a</sup> SOLUÇÃO - UTILIZANDO ESTRUTURA CONDICIONAL SIMPLES:**

\EXERC\CAP3\PASCAL\EX3\_A.PAS e \EXERC\CAP3\PASCAL\EX3\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:**

\EXERC\CAP3\PASCAL\EX3\_B.PAS e \EXERC\CAP3\PASCAL\EX3\_B.EXE

**1<sup>a</sup> SOLUÇÃO - UTILIZANDO ESTRUTURA CONDICIONAL SIMPLES:**

\EXERC\CAP3\C++\EX3\_A.CPP e \EXERC\CAP3\C++\EX3\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:**

\EXERC\CAP3\C++\EX3\_B.CPP e \EXERC\CAP3\C++\EX3\_B.EXE

- 4.** Faça um programa que receba três números e mostre-os em ordem crescente.

**SOLUÇÃO:****ALGORITMO**

```

DECLARE num1, num2, num3 NUMÉRICO
ESCREVA "Digite o primeiro número: "
LEIA num1
ESCREVA "Digite o segundo número: "
LEIA num2
ESCREVA "Digite o terceiro número: "
LEIA num3

```

```

SE (num1 < num2) E (num1 < num3)
ENTÃO SE (num2 < num3)
    ENTÃO ESCREVA "A ordem crescente é: ", num1, "-", num2, "-"
    ↪ ", num3
    SENÃO ESCREVA "A ordem crescente é: ", num1, "-", num3, "-"
    ↪ ", num2
SE (num2 < num1) E (num2 < num3)
ENTÃO SE (num1 < num3)
    ENTÃO ESCREVA "A ordem crescente é: ", num2, "-", num1, "-"
    ↪ ", num3
    SENÃO ESCREVA "A ordem crescente é: ", num2, "-", num3, "-"
    ↪ ", num1
SE (num3 < num1) E (num3 < num2)
ENTÃO SE (num1 < num2)
    ENTÃO ESCREVA "A ordem crescente é: ", num3, "-", num1, "-"
    ↪ ", num2
    SENÃO ESCREVA "A ordem crescente é: ", num3, "-", num2, "-"
    ↪ ", num1
FIM_ALGORITMO.

```



#### 1<sup>a</sup> SOLUÇÃO:

\EXERC\CAP3\PASCAL\EX4\_A.PAS e \EXERC\CAP3\PASCAL\EX4\_A.EXE

#### 2<sup>a</sup> SOLUÇÃO:

\EXERC\CAP3\PASCAL\EX4\_B.PAS e \EXERC\CAP3\PASCAL\EX4\_B.EXE



#### 1<sup>a</sup> SOLUÇÃO:

\EXERC\CAP3\C++\EX4\_A.CPP e \EXERC\CAP3\C++\EX4\_A.EXE

#### 2<sup>a</sup> SOLUÇÃO:

\EXERC\CAP3\C++\EX4\_B.CPP e \EXERC\CAP3\C++\EX4\_B.EXE

5. Faça um programa que receba três números obrigatoriamente em ordem crescente e um quarto número que não siga esta regra. Mostre, em seguida, os quatro números em ordem decrescente.



#### SOLUÇÃO:

```

ALGORITMO
DECLARE num1, num2, num3, num4 NUMÉRICO
ESCREVA "Digite três números em ordem crescente: "
LEIA num1
LEIA num2
LEIA num3
ESCREVA "Digite um número (fora de ordem): "
LEIA num4
SE (num4 > num3)
    ENTÃO ESCREVA "A ordem decrescente é: ", num4, "-", num3, "-",
    ↪ num2, "-", num1
SE (num4 > num2) E (num4 < num3)
    ENTÃO ESCREVA "A ordem decrescente é: ", num3, "-", num4, "-",
    ↪ num2, "-", num1
SE (num4 > num1) E (num4 < num2)
    ENTÃO ESCREVA "A ordem decrescente é: ", num3, "-", num2, "-",
    ↪ num4, "-", num1

```

```

SE (num4 < num1)
ENTÃO ESCREVA "A ordem decrescente é: ", num3, "-", num2, "-",
→ num1, "-", num4
FIM_ALGORITMO.

```

**1<sup>a</sup> SOLUÇÃO:**

\EXERC\CAP3\PASCAL\EX5\_A.PAS e \EXERC\CAP3\PASCAL\EX5\_A.EXE

**2<sup>a</sup> SOLUÇÃO:**

\EXERC\CAP3\PASCAL\EX5\_B.PAS e \EXERC\CAP3\PASCAL\EX5\_B.EXE

**1<sup>a</sup> SOLUÇÃO:**

\EXERC\CAP3\C++\EX5\_A.CPP e \EXERC\CAP3\C++\EX1\_A.EXE

**2<sup>a</sup> SOLUÇÃO:**

\EXERC\CAP3\C++\EX5\_B.CPP e \EXERC\CAP3\C++\EX5\_B.EXE

6. Faça um programa que receba um número inteiro e verifique se esse número é par ou ímpar.

**SOLUÇÃO:**

```

ALGORITMO
DECLARE num, r NUMÉRICO
ESCREVA "Digite um número: "
LEIA num
r ← RESTO(num/2)
SE r = 0
ENTÃO ESCREVA "O número é par"
SENÃO ESCREVA "O número é ímpar"
FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP3\PASCAL\EX6.PAS e \EXERC\CAP3\PASCAL\EX6.EXE

**SOLUÇÃO:**

\EXERC\CAP3\C++\EX6.CPP e \EXERC\CAP3\C++\EX6.EXE

7. Faça um programa que receba quatro valores, I, A, B e C. I é um valor inteiro e positivo e A, B e C são valores reais. Escreva os números A, B e C obedecendo à tabela a seguir.

**OBSERVAÇÃO:**

Supor que o valor digitado para I seja sempre um valor válido, ou seja, 1, 2 ou 3.

VALOR DE I	FORMA A ESCREVER
1	A, B e C em ordem crescente.
2	A, B e C em ordem decrescente.
3	O maior fica entre os outros dois números.

**SOLUÇÃO:**

```

ALGORITMO
DECLARE A, B, C, I NUMÉRICO
ESCREVA "Digite um valor para A: "
LEIA A
ESCREVA "Digite um valor para B: "
LEIA B
ESCREVA "Digite um valor para C: "
LEIA C
ESCREVA "Digite um valor para I (1, 2 ou 3): "
LEIA I
SE I=1
    ENTÃO INÍCIO
        SE (A<B) E (A<C)
            ENTÃO SE (B<A)
                ENTÃO ESCREVA "A ordem crescente dos números é: ",
                ↪ A, "-", B, "-", C
                SENÃO ESCREVA "A ordem crescente dos números é: ",
                ↪ A, "-", C, "-", B
            SE (B<A) E (B<C)
                ENTÃO SE (A<C)
                    ENTÃO ESCREVA "A ordem crescente dos números é:
                    ↪ ", B, "-", A, "-", C
                    SENÃO ESCREVA "A ordem crescente dos números é:
                    ↪ ", B, "-", C, "-", A
            SE (C<A) E (C<B)
                ENTÃO SE (A<B)
                    ENTÃO ESCREVA "A ordem crescente dos números é:
                    ↪ ", C, "-", A, "-", B
                    SENÃO ESCREVA "A ordem crescente dos números é:
                    ↪ ", C, "-", B, "-", A
            FIM
        SE (I=2)
            ENTÃO INÍCIO
                SE (A>B) E (A>C)
                    ENTÃO SE (B>A)
                        ENTÃO ESCREVA "A ordem decrescente dos números é:
                        ↪ ", A, "-", B, "-", C
                        SENÃO ESCREVA "A ordem decrescente dos números é:
                        ↪ ", A, "-", C, "-", B
                SE (B>A) E (B>C)
                    ENTÃO SE (A>C)
                        ENTÃO ESCREVA "A ordem decrescente dos números é:
                        ↪ ", B, "-", A, "-", C
                        SENÃO ESCREVA "A ordem decrescente dos números é:
                        ↪ ", B, "-", C, "-", A
                SE (C>A) E (C>B)
                    ENTÃO SE (A>B)
                        ENTÃO ESCREVA "A ordem decrescente dos números é:
                        ↪ ", C, "-", A, "-", B
                        SENÃO ESCREVA "A ordem decrescente dos números é:
                        ↪ ", C, "-", B, "-", A
            FIM
        SE (I=3)
            ENTÃO INÍCIO
                SE (A>B) E (A>C)
                    ENTÃO ESCREVA "A ordem desejada é: ", B, "-", A, "-", C
                SE (B>A) E (B>C)
                    ENTÃO ESCREVA "A ordem desejada é: ", A, "-", B, "-", C
                SE (C>A) E (C>B)
                    ENTÃO ESCREVA "A ordem desejada é: ", A, "-", C, "-", B
            FIM
    FIM_ALGORITMO.

```

**1<sup>a</sup> SOLUÇÃO:**

\EXERC\CAP3\PASCAL\EX7\_A.PAS e \EXERC\CAP3\PASCAL\EX7\_A.EXE

**2<sup>a</sup> SOLUÇÃO:**

\EXERC\CAP3\PASCAL\EX7\_B.PAS e \EXERC\CAP3\PASCAL\EX7\_B.EXE

**3<sup>a</sup> SOLUÇÃO:**

\EXERC\CAP3\PASCAL\EX7\_C.PAS e \EXERC\CAP3\PASCAL\EX7\_C.EXE

**1<sup>a</sup> SOLUÇÃO:**

\EXERC\CAP3\C++\EX7\_A.CPP e \EXERC\CAP3\C++\EX7\_A.EXE

**2<sup>a</sup> SOLUÇÃO:**

\EXERC\CAP3\C++\EX7\_B.CPP e \EXERC\CAP3\C++\EX7\_B.EXE

**3<sup>a</sup> SOLUÇÃO:**

\EXERC\CAP3\C++\EX7\_C.CPP e \EXERC\CAP3\C++\EX7\_C.EXE

- 8.** Faça um programa que mostre o menu de opções a seguir, receba a opção do usuário e os dados necessários para executar cada operação.

Menu de opções:

1. Somar dois números
  2. Raiz quadrada de um número
- Digite a opção desejada

**SOLUÇÃO:**

```

ALGORITMO
DECLARE num1,num2,soma, raiz, op NUMÉRICO
ESCREVA " MENU"
ESCREVA "1- Somar dois números"
ESCREVA "2- Raiz quadrada de um número"
ESCREVA "Digite sua opção: "
LEIA op
SE op = 1
ENTÃO INÍCIO
    ESCREVA "Digite um valor para o primeiro número: "
    LEIA num1
    ESCREVA "Digite um valor para o segundo número: "
    LEIA num2
    soma ← num1 + num2
    ESCREVA "A soma de ",num1," e ",num2," é ",soma
FIM
SE op = 2
ENTÃO INÍCIO
    ESCREVA "Digite um valor: "
    LEIA num1
    raiz ← √num1
    ESCREVA "A raiz quadrada de ",num1," é ",raiz
FIM

```

```

SE (op ≠ 1) E (op ≠ 2)
ENTÃO ESCREVA Opção inválida !
FIM_ALGORITMO.

```



**1<sup>a</sup> SOLUÇÃO:**

\EXERC\CAP3\PASCAL\EX8\_A.PAS e \EXERC\CAP3\PASCAL\EX8\_A.EXE

**2<sup>a</sup> SOLUÇÃO:**

\EXERC\CAP3\PASCAL\EX8\_B.PAS e \EXERC\CAP3\PASCAL\EX8\_B.EXE

**3<sup>a</sup> SOLUÇÃO:**

\EXERC\CAP3\PASCAL\EX8\_C.PAS e \EXERC\CAP3\PASCAL\EX8\_C.EXE



**1<sup>a</sup> SOLUÇÃO:**

\EXERC\CAP3\C++\EX8\_A.CPP e \EXERC\CAP3\C++\EX8\_A.EXE

**2<sup>a</sup> SOLUÇÃO:**

\EXERC\CAP3\C++\EX8\_B.CPP e \EXERC\CAP3\C++\EX8\_B.EXE

**3<sup>a</sup> SOLUÇÃO:**

\EXERC\CAP3\C++\EX8\_C.CPP e \EXERC\CAP3\C++\EX8\_C.EXE

- 9.** Faça um programa que mostre a data e a hora do sistema nos seguintes formatos:  
dia/mês/ano – mês por extenso e hora:minuto.



**SOLUÇÃO:**

```

ALGORITMO
DECLARE T, D, dia, mes, ano, hora, min NUMÉRICO
D ← OBTENHA_DATA;
dia ← OBTENHA_DIA(D)
mes ← OBTENHA_MÊS(D)
ano ← OBTENHA_ANO(D)
ESCREVA "Data Atual: " , dia, "/", mes, "/", ano, " - "
SE (mes = 1)
    ENTÃO ESCREVA "janeiro"
SE (mes = 2)
    ENTÃO ESCREVA "fevereiro"
SE (mes = 3)
    ENTÃO ESCREVA "março"
SE (mes = 4)
    ENTÃO ESCREVA "abril"
SE (mes = 5)
    ENTÃO ESCREVA "maio"
SE (mes = 6)
    ENTÃO ESCREVA "junho"
SE (mes = 7)
    ENTÃO ESCREVA "julho"
SE (mes = 8)
    ENTÃO ESCREVA "agosto"
SE (mes = 9)
    ENTÃO ESCREVA "setembro"

```

```

SE (mes = 10)
ENTÃO ESCREVA "outubro"
SE (mes = 11)
ENTÃO ESCREVA "novembro"
SE (mes = 12)
ENTÃO ESCREVA "dezembro"
ESCREVA(ano)
T ← OBTERNA_HORARIO;
hora ← OBTERNA_HORA(T)
min ← OBTERNA_MINUTO(T)
ESCREVA "Hora Atual: "
ESCREVA hora, ":" , min
FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP3\PASCAL\EX9.PAS e \EXERC\CAP3\PASCAL\EX9.EXE

**SOLUÇÃO:**

\EXERC\CAP3\C++\EX9.CPP e \EXERC\CAP3\C++\EX9.EXE

10. Faça um programa que determine a data cronologicamente maior de duas datas fornecidas pelo usuário. Cada data deve ser fornecida por três valores inteiros, onde o primeiro representa o dia, o segundo o mês e o terceiro o ano.

**SOLUÇÃO:**

```

ALGORITMO
DECLARE d1,m1,a1,d2,m2,a2 NUMÉRICO
ESCREVA "Digite a primeira data"
ESCREVA " dia (dd): "
LEIA d1
ESCREVA " mês (mm): "
LEIA m1
ESCREVA " ano (aaaa): "
LEIA a1
ESCREVA "Digite a segunda data"
ESCREVA " dia (dd): "
LEIA d2
ESCREVA " mês (mm): "
LEIA m2
ESCREVA " ano (aaaa): "
LEIA a2
SE (a1>a2)
ENTÃO ESCREVA "A maior data é: ",d1,"-",m1,"-",a1
SENÃO SE (a2>a1)
    ENTÃO ESCREVA "A maior data é: ",d2,"-",m2,"-",a2
SENÃO SE (m1>m2)
    ENTÃO ESCREVA "A maior data é: ",d1,"-",m1,"-",a1
    SENÃO SE (m2>m1)
        ENTÃO ESCREVA "A maior data é: ",d2,"-
        ↪ ",m2,"-",a2
        SENÃO SE (d1>d2)
            ENTÃO ESCREVA "A maior data é:
            ↪ ",d1,"-",m1,"-",a1
            SENÃO SE (d2>d1)
                ENTÃO ESCREVA "A maior data
                ↪ é: ",d2,"-",m2,"-",a2

```

SENÃO ESCREVA "As datas são  
iguais!"

FIM\_ALGORITMO.



#### SOLUÇÃO:

\EXERC\CAP3\PASCAL\EX10.PAS e \EXERC\CAP3\PASCAL\EX10.EXE



#### SOLUÇÃO:

\EXERC\CAP3\C++\EX10.CPP e \EXERC\CAP3\C++\EX10.EXE

11. Faça um programa que receba a hora de início de um jogo e a hora final do jogo (cada hora é composta por duas variáveis inteiros: hora e minuto). Calcule e mostre a duração do jogo (horas e minutos) sabendo-se que o tempo máximo de duração do jogo é de 24 horas e que o jogo pode iniciar em um dia e terminar no dia seguinte.



#### SOLUÇÃO:

```

ALGORITMO
DECLARE hora_i, min_i, hora_f, min_f, hora_d, min_d NUMÉRICO
ESCREVA "Digite o horário inicial"
ESCREVA "hora: "
LEIA hora_i
ESCREVA "minuto: "
LEIA min_i
ESCREVA "Digite o horário final "
ESCREVA "hora: "
LEIA hora_f
ESCREVA "minuto: "
LEIA min_f
SE (min_i > min_f)
    ENTÃO INÍCIO
        min_f ← min_f + 60
        hora_f ← hora_f - 1
    FIM
SE (hora_i > hora_f)
    ENTÃO hora_f ← hora_f + 24
min_d ← min_f - min_i;
hora_d ← hora_f - hora_i;
ESCREVA "O jogo durou ",hora_d," hora(s) e ",min_d," minuto(s)"
FIM_ALGORITMO.

```



#### SOLUÇÃO:

\EXERC\CAP3\PASCAL\EX11.PAS e \EXERC\CAP3\PASCAL\EX11.EXE



#### SOLUÇÃO:

\EXERC\CAP3\C++\EX11.CPP e \EXERC\CAP3\C++\EX11.EXE

12. Faça um programa que receba o código correspondente ao cargo de um funcionário e seu salário atual e mostre o cargo, o valor do aumento e seu novo salário. Os cargos estão na tabela a seguir.

CÓDIGO	CARGO	PERCENTUAL
1	Escriturário	50%
2	Secretário	35%
3	Caixa	20%
4	Gerente	10%
5	Diretor	Não tem aumento

**ALGORITMO****SOLUÇÃO:**

```

ALGORITMO
DECLARE salario, aumento, novo_sal, cargo NUMÉRICO
ESCREVA "Digite o cargo do funcionário (1,2,3,4 ou 5)"
LEIA cargo
ESCREVA "Digite o valor do salário: "
LEIA salario
SE (cargo = 1)
    ENTÃO INÍCIO
        ESCREVA "O cargo é Escriturário"
        aumento ← salario * 50 / 100
        ESCREVA "O valor do aumento é: ", aumento
        novo_sal ← salario + aumento
        ESCREVA "O novo salário é: ", novo_sal
    FIM
SENÃO SE (cargo = 2)
    ENTÃO INÍCIO
        ESCREVA "O cargo é Secretário"
        aumento ← salario * 35 / 100
        ESCREVA "O valor do aumento é: ", aumento
        novo_sal ← salario + aumento
        ESCREVA "O novo salário é: ", novo_sal
    FIM
SENÃO SE (cargo = 3)
    ENTÃO INÍCIO
        ESCREVA "O cargo é Caixa"
        aumento ← salario * 20 / 100
        ESCREVA "O valor do aumento é: ", aumento
        novo_sal ← salario + aumento
        ESCREVA "O novo salário é: ", novo_sal
    FIM
SENÃO SE (cargo = 4)
    ENTÃO INÍCIO
        ESCREVA "O cargo é Gerente"
        aumento ← salario * 10 / 100
        ESCREVA "O valor do aumento é: ",
        ↪ aumento
        novo_sal ← salario + aumento
        ESCREVA "O novo salário é: ",
        ↪ novo_sal
    FIM
SENÃO SE (cargo = 5)
    ENTÃO INÍCIO
        ESCREVA "O cargo é Diretor"
        aumento ← salario * 0 / 100
        ESCREVA "O valor do aumento
        ↪ é: ", aumento
        novo_sal ← salario +
        ↪ aumento
        ESCREVA "O novo salário é:
        ↪ ", novo_sal
    FIM

```

SENÃO ESCREVA "Cargo Inexistente  
! "

FIM\_ALGORITMO.



**1<sup>A</sup> SOLUÇÃO:**

\EXERC\CAP3\PASCAL\EX12\_A.PAS e \EXERC\CAP3\PASCAL\EX12\_A.EXE

**2<sup>A</sup> SOLUÇÃO:**

\EXERC\CAP3\PASCAL\EX12\_B.PAS e \EXERC\CAP3\PASCAL\EX12\_B.EXE



**1<sup>A</sup> SOLUÇÃO:**

\EXERC\CAP3\C++\EX12\_A.CPP e \EXERC\CAP3\C++\EX12\_A.EXE

**2<sup>A</sup> SOLUÇÃO:**

\EXERC\CAP3\C++\EX12\_B.CPP e \EXERC\CAP3\C++\EX12\_B.EXE

- 13.** Faça um programa que apresente o menu de opções a seguir, permita ao usuário escolher a opção desejada, receba os dados necessários para executar a operação e mostre o resultado. Verifique a possibilidade de opção inválida e não se preocupe com restrições como salário negativo.

Menu de opções:

1. Imposto
  2. Novo salário
  3. Classificação
- Digite a opção desejada.

**Na opção 1:** receber o salário de um funcionário, calcular e mostrar o valor do imposto usando as regras a seguir:

SALÁRIOS	PERCENTUAL DO IMPOSTO
Menor que R\$ 500,00	5%
De R\$ 500,00 a R\$ 850,00	10%
Acima de R\$ 850,00	15%

**Na opção 2:** receber o salário de um funcionário, calcular e mostrar o valor do novo salário, usando as regras a seguir:

SALÁRIO	AUMENTO
Maiores que R\$ 1.500,00	R\$ 25,00
De R\$ 750,00 (inclusive) a R\$ 1.500,00 (inclusive)	R\$ 50,00
De R\$ 450,00 (inclusive) a R\$ 750,00	R\$ 75,00
Menores que R\$ 450,00	R\$ 100,00

**Na opção 3:** receber o salário de um funcionário e mostrar sua classificação usando a tabela a seguir:

SALÁRIO	CLASSIFICAÇÃO
Até R\$ 700,00 (inclusive)	Mal remunerado
Maiores que R\$ 700,00	Bem remunerado

**ALGORITMO****1ª SOLUÇÃO - UTILIZANDO ESTRUTURA CONDICIONAL SIMPLES:**

```

ALGORITMO
DECLARE op,sal, imp, aum, novo_sal NUMÉRICO
LEIA op
SE op = 1
ENTÃO INÍCIO
    LEIA sal
    SE sal < 500
        ENTÃO imp ← sal * 5%
    SE (sal >= 500) E (sal <= 850)
        ENTÃO imp ← sal * 10%
    SE (sal > 850)
        ENTÃO imp ← sal * 15%
    ESCREVA imp
    FIM
SE op = 2
ENTÃO INÍCIO
    LEIA sal
    SE sal > 1500
        ENTÃO aum ← 25
    SE (sal >= 750) E (sal <= 1500)
        ENTÃO aum ← 50
    SE (sal >= 450) E (sal < 750)
        ENTÃO aum ← 75
    SE sal < 450
        ENTÃO aum ← 100
    novo_sal ← sal + aum
    ESCREVA novo_sal
    FIM
SE op = 3
ENTÃO INÍCIO
    LEIA sal
    SE sal <= 700
        ENTÃO ESCREVA "Mal Remunerado"
    SE sal > 700
        ENTÃO ESCREVA "Bem Remunerado"
    FIM
SE (op < 1) OU (op > 3)
    ENTÃO ESCREVA "Opção Inválida"
FIM_ALGORITMO.

```

**2ª SOLUÇÃO - UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:**

```

ALGORITMO
DECLARE op, sal, imp, aum, novo_sal NUMÉRICO
LEIA op
SE op = 1
ENTÃO INÍCIO
    LEIA sal
    SE sal < 500
        ENTÃO imp ← sal * 5%
    SENÃO SE sal <= 850

```

```

ENTÃO imp ← sal * 10%
SENÃO imp ← sal * 15%
ESCREVA imp
FIM
SENÃO SE op = 2
ENTÃO INÍCIO
LEIA sal
SE sal > 1500
ENTÃO aum ← 25
SENÃO SE (sal >= 750)
ENTÃO aum ← 50
SENÃO SE (sal >= 450)
ENTÃO aum ← 75
SENÃO aum ← 100
novo_sal ← sal + aum
ESCREVA novo_sal
FIM
SENÃO SE op = 3
ENTÃO INÍCIO
LEIA sal
SE sal <= 700
ENTÃO ESCREVA "Mal Remunerado"
SENÃO ESCREVA "Bem Remunerado"
FIM
SENÃO ESCREVA "Opção Inválida"
FIM_ALGORITMO.

```



#### 1<sup>a</sup> SOLUÇÃO - UTILIZANDO ESTRUTURA CONDICIONAL SIMPLES:

\EXERC\CAP3\PASCAL\EX13\_A.PAS e \EXERC\CAP3\PASCAL\EX13\_A.EXE

#### 2<sup>a</sup> SOLUÇÃO - UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:

\EXERC\CAP3\PASCAL\EX13\_B.PAS e \EXERC\CAP3\PASCAL\EX13\_B.EXE

#### 3<sup>a</sup> SOLUÇÃO - UTILIZANDO ESTRUTURA SELETORA:

\EXERC\CAP3\PASCAL\EX13\_C.PAS e \EXERC\CAP3\PASCAL\EX13\_C.EXE



#### 1<sup>a</sup> SOLUÇÃO - UTILIZANDO ESTRUTURA CONDICIONAL SIMPLES:

\EXERC\CAP3\C++\EX13\_A.CPP e \EXERC\CAP3\C++\EX13\_A.EXE

#### 2<sup>a</sup> SOLUÇÃO - UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:

\EXERC\CAP3\C++\EX13\_B.CPP e \EXERC\CAP3\C++\EX13\_B.EXE

#### 3<sup>a</sup> SOLUÇÃO - UTILIZANDO ESTRUTURA SELETORA:

\EXERC\CAP3\C++\EX13\_C.CPP e \EXERC\CAP3\C++\EX13\_C.EXE

- 14.** Faça um programa que receba o salário de um funcionário, calcule e mostre o novo salário desse funcionário, acrescido de bonificação e de auxílio-escola.

SALÁRIO	BONIFICAÇÃO	SALÁRIO	AUXÍLIO-ESCOLA
Até R\$ 500,00	5% do salário	Até R\$ 600,00	R\$ 150,00
Entre R\$ 500,01 e R\$ 1.200,00	12% do salário	Mais que R\$ 600,00	R\$ 100,00
Acima de R\$ 1.200,00	Sem bonificação		

**ALGORITMO****1<sup>a</sup> SOLUÇÃO - UTILIZANDO ESTRUTURA CONDICIONAL SIMPLES:**

```

ALGORITMO
DECLARE sal, novo_sal, boni, aux NUMÉRICO
LEIA sal
SE sal <= 500
    ENTÃO boni ← sal * 5%
SE (sal > 500) E (sal <= 1200)
    ENTÃO boni ← sal * 12%
SE (sal > 1200)
    ENTÃO boni ← 0
SE sal <= 600
    ENTÃO aux ← 150
SE sal > 600
    ENTÃO aux ← 100
novo_sal ← sal + boni + aux
ESCREVA novo_sal
FIM_ALGORITMO.
```

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:**

```

ALGORITMO
DECLARE sal, novo_sal, boni, aux NUMÉRICO
LEIA sal
SE sal <= 500
    ENTÃO boni ← sal * 5%
SENÃO SE sal <= 1200
    ENTÃO boni ← sal * 12%
    SENÃO boni ← 0
SE sal <= 600
    ENTÃO aux ← 150
    SENÃO aux ← 100
novo_sal ← sal + boni + aux
ESCREVA novo_sal
FIM_ALGORITMO.
```

**RESOLUÇÃO  
PASCAL****1<sup>a</sup> SOLUÇÃO - UTILIZANDO ESTRUTURA CONDICIONAL SIMPLES:**

\EXERC\CAP3\PASCAL\EX14\_A.PAS e \EXERC\CAP3\PASCAL\EX14\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:**

\EXERC\CAP3\PASCAL\EX14\_B.PAS e \EXERC\CAP3\PASCAL\EX14\_B.EXE

**RESOLUÇÃO  
C/C++****1<sup>a</sup> SOLUÇÃO - UTILIZANDO ESTRUTURA CONDICIONAL SIMPLES:**

\EXERC\CAP3\C++\EX14\_A.CPP e \EXERC\CAP3\C++\EX14\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:**

\EXERC\CAP3\C++\EX14\_B.CPP e \EXERC\CAP3\C++\EX14\_B.EXE

**15.** Faça um programa que receba o valor do salário mínimo, o número de horas trabalhadas, o número de dependentes do funcionário e a quantidade de horas extras trabalhadas. Calcule e mostre o salário a receber do funcionário de acordo com as regras a seguir:

- ◆ o valor da hora trabalhada é igual a  $\frac{1}{5}$  do salário mínimo;
- ◆ o salário do mês é igual ao número de horas trabalhadas multiplicado pelo valor da hora trabalhada;

- ◆ para cada dependente acrescentar R\$ 32,00;
- ◆ para cada hora extra trabalhada calcular o valor da hora trabalhada acrescida de 50%;
- ◆ o salário bruto é igual ao salário do mês mais o valor dos dependentes mais o valor das horas extras;
- ◆ calcular o valor do imposto de renda retido na fonte de acordo com a tabela a seguir.

<b>IRRF</b>	<b>SALÁRIO BRUTO</b>
Isento	Inferior a R\$ 200,00
10%	De R\$ 200,00 até R\$ 500,00
20%	Superior a R\$ 500,00

- ◆ o salário líquido é igual ao salário bruto menos IRRF;
- ◆ a gratificação de acordo com a tabela a seguir.

<b>SALÁRIO LÍQUIDO</b>	<b>GRATIFICAÇÃO</b>
Até R\$ 350,00	R\$ 100,00
Superior a R\$ 350,00	R\$ 50,00

- ◆ o salário a receber do funcionário é igual ao salário líquido mais a gratificação.

## ALGORITMO

### SOLUÇÃO:

```

ALGORITMO
DECLARE sal_min, nht, ndep, nhet NUMÉRICO
           sal_receber, vh, smes, vdep, vhe, imp NUMÉRICO
           sbruto, sliq, grat NUMÉRICO
LEIA sal_min, nht, ndep, nhet
vh ← 1/5 * sal_min
smes ← nht * vh
vdep ← 32 * ndep
vhe ← nhet * (vh + (vh * 50/100))
sbruto ← smes + vdep + vhe
SE sbruto < 200
  ENTÃO imp ← 0
SE (sbruto >= 200) E (sbruto <= 500)
  ENTÃO imp ← sbruto * 10%
SE sbruto > 500
  ENTÃO imp ← sbruto * 20%
sliq ← sbruto - imp
SE sliq <= 350
  ENTÃO grat ← 100
SE sliq > 350
  ENTÃO grat ← 50
sal_receber ← sliq + grat
ESCREVA sal_receber
FIM_ALGORITMO.

```



### 1ª SOLUÇÃO - UTILIZANDO ESTRUTURA CONDICIONAL SIMPLES:

\EXERC\CAP3\PASCAL\EX15\_A.PAS e \EXERC\CAP3\PASCAL\EX15\_A.EXE

### 2ª SOLUÇÃO - UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:

\EXERC\CAP3\PASCAL\EX15\_B.PAS e \EXERC\CAP3\PASCAL\EX15\_B.EXE

**1<sup>a</sup> SOLUÇÃO - UTILIZANDO ESTRUTURA CONDICIONAL SIMPLES:**

\EXERC\CAP3\C++\EX15\_A.CPP e \EXERC\CAP3\C++\EX15\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:**

\EXERC\CAP3\C++\EX15\_B.CPP e \EXERC\CAP3\C++\EX15\_B.EXE

- 16.** Um supermercado deseja reajustar os preços de seus produtos usando o seguinte critério: o produto poderá ter seu preço aumentado ou diminuído. Para alterar o preço o produto deve preencher pelo menos um dos requisitos a seguir:

REQUISITOS	REAJUSTES		
VENDA MÉDIA MENSAL	PREÇO ATUAL	% DE AUMENTO	% DE DIMINUIÇÃO
< 500	<R\$ 30,00	10	-
= 500 e < 1.200	= R\$ 30,00 e < R\$ 80,00	15	-
= 1.200	= R\$ 80,00	-	20

Faça um programa que receba o preço atual e a venda mensal média do produto, calcule e mostre o novo preço.

**SOLUÇÃO:**

```

ALGORITMO
DECLARE pre, venda, novo_pre NUMÉRICO
LEIA pre, venda
SE (venda<500) OU (pre<30)
ENTÃO novo_pre ← pre + 10% * pre
SENÃO SE ((venda>=500) E (venda<1200)) OU ((pre>=30) E (pre<80))
    ENTÃO novo_pre ← pre + 15% * pre
    SENÃO SE (venda>1200) OU (pre>=80)
        ENTÃO novo_pre ← pre - 20% * pre
ESCREVA novo_pre
FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP3\PASCAL\EX16.PAS e \EXERC\CAP3\PASCAL\EX16.EXE

**SOLUÇÃO:**

\EXERC\CAP3\C++\EX16.CPP e \EXERC\CAP3\C++\EX16.EXE

- 17.** Faça um programa para resolver equações do 2º grau.

$$ax^2 + bx + c = 0$$

A variável  $a$  deve ser diferente de zero.

$\Delta < 0 \rightarrow$  não existe raiz real

$\Delta = 0 \rightarrow$  existe uma raiz real

$$x = -b / (2 * a)$$

$\Delta > 0 \rightarrow$  existem duas raízes reais

$$x1 = (-b + \sqrt[3]{\Delta}) / (2 * a)$$

$$x2 = (-b - \sqrt[3]{\Delta}) / (2 * a)$$

**ALGORITMO****SOLUÇÃO:**

```

ALGORITMO
    DECLARE a, b, c, delta, x1, x2 NUMÉRICO
    LEIA a, b, c
    SE a = 0
        ENTÃO ESCREVA "Estes valores não formam uma equação de segundo grau"
        SENÃO INÍCIO
            delta ← (b * b) - (4 * a * c)
            SE delta < 0
                ENTÃO ESCREVA "Não existe raiz real"
            SE delta = 0
                ENTÃO INÍCIO
                    ESCREVA "Existe uma raiz real"
                    x1 ← -b / (2 * a)
                    ESCREVA x1
                    FIM
            SE delta > 0
                ENTÃO INÍCIO
                    ESCREVA "Existem duas raízes reais"
                    x1 ← (-b + \sqrt[3]{\Delta}) / (2 * a)
                    x2 ← (-b - \sqrt[3]{\Delta}) / (2 * a)
                    ESCREVA x1, x2
                    FIM
            FIM
        FIM_ALGORITMO.
    
```

**SOLUÇÃO:**

\EXERC\CAP3\PASCAL\EX17.PAS e \EXERC\CAP3\PASCAL\EX17.EXE

**SOLUÇÃO:**

\EXERC\CAP3\C++\EX17.CPP e \EXERC\CAP3\C++\EX17.EXE

**18.** Dados três valores X, Y e Z, verificar se eles podem ser os comprimentos dos lados de um triângulo e, se forem, verificar se é um triângulo eqüilátero, isósceles ou escaleno. Se eles não formarem um triângulo escrever uma mensagem. Considerar que:

- ◆ o comprimento de cada lado de um triângulo é menor que a soma dos outros dois lados;
- ◆ chama-se triângulo eqüilátero o triângulo que tem três lados iguais;
- ◆ chama-se triângulo isósceles o triângulo que tem o comprimento de dois lados iguais;
- ◆ chama-se triângulo escaleno o triângulo que tem os três lados diferentes.

**ALGORITMO****SOLUÇÃO:**

```

ALGORITMO
    DECLARE x, y, z NUMÉRICO
    LEIA x, y, z
    SE (x < y + z) E (y < x + z) E (z < x + y)
    ENTÃO INÍCIO
        SE (x = y) E (y = z)
    
```

```

ENTÃO ESCREVA "Triângulo Eqüilátero"
SENÃO SE (x = y) OU (x = z) OU (y = z)
    ENTÃO ESCREVA "Triângulo Isósceles"
    SENÃO SE (x ≠ y) E (x ≠ z) E (y ≠ z)
        ENTÃO ESCREVA "Triângulo Escaleno"
    FIM
SENÃO ESCREVA "Essas medidas não formam um triângulo"
FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP3\PASCAL\EX18.PAS e \EXERC\CAP3\PASCAL\EX18.EXE

**SOLUÇÃO:**

\EXERC\CAP3\C++\EX18.CPP e \EXERC\CAP3\C++\EX18.EXE

- 19.** Faça um programa que receba a altura e o peso de uma pessoa. De acordo com a tabela a seguir verifique e mostre qual a classificação dessa pessoa.

ALTURA	PESO		
	ATÉ 60	ENTRE 60 E 90 (INCLUSIVE)	ACIMA DE 90
Menores que 1,20	A	D	G
De 1,20 a 1,70	B	E	H
Maiores que 1,70	C	F	I

**SOLUÇÃO:**

```

ALGORITMO
DECLARE altura, peso NUMÉRICO
LEIA altura, peso
SE altura < 1.20
    ENTÃO INÍCIO
        SE peso < = 60
            ENTÃO ESCREVA "A"
        SE (peso > 60) E (peso <= 90)
            ENTÃO ESCREVA "D"
        SE peso > 90
            ENTÃO ESCREVA "G"
        FIM
    SE (altura >= 1.20) E (altura <= 1.70)
        ENTÃO INÍCIO
            SE peso < = 60
                ENTÃO ESCREVA "B"
            SE (peso > 60) E (peso <= 90)
                ENTÃO ESCREVA "E"
            SE peso > 90
                ENTÃO ESCREVA "H"
            FIM
        SE altura > 1.70
            ENTÃO INÍCIO
                SE peso < = 60
                    ENTÃO ESCREVA "C"
                SE (peso > 60) E (peso <= 90)
                    ENTÃO ESCREVA "F"
                SE peso > 90
                    ENTÃO ESCREVA "I"
                FIM
            FIM_ALGORITMO.

```


**1<sup>A</sup> SOLUÇÃO - UTILIZANDO ESTRUTURA CONDICIONAL SIMPLES:**

\EXERC\CAP3\PASCAL\EX19\_A.PAS e \EXERC\CAP3\PASCAL\EX19\_A.EXE

**2<sup>A</sup> SOLUÇÃO - UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:**

\EXERC\CAP3\PASCAL\EX19\_B.PAS e \EXERC\CAP3\PASCAL\EX19\_B.EXE


**1<sup>A</sup> SOLUÇÃO - UTILIZANDO ESTRUTURA CONDICIONAL SIMPLES:**

\EXERC\CAP3\C++\EX19\_A.CPP e \EXERC\CAP3\C++\EX19\_A.EXE

**2<sup>A</sup> SOLUÇÃO - UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:**

\EXERC\CAP3\C++\EX19\_B.CPP e \EXERC\CAP3\C++\EX19\_B.EXE

**20.** Faça um programa que receba:

- ◆ o código de um produto comprado, supondo que a digitação do código do produto seja sempre válida, ou seja, um número inteiro entre 1 e 10;
- ◆ o peso do produto em quilos;
- ◆ o código do país de origem, supondo que a digitação do código do país seja sempre válida, ou seja, um número inteiro entre 1 e 3.

Tabelas:

CÓDIGO DO PAÍS DE ORIGEM	IMPOSTO	CÓDIGO DO PRODUTO	PREÇO POR GRAMA
1	0%	1 a 4	10
2	15%	5 a 7	25
3	25%	8 a 10	35

Calcule e mostre:

- ◆ o peso do produto convertido em gramas;
- ◆ o preço total do produto comprado;
- ◆ o valor do imposto, sabendo-se que o imposto é cobrado sobre o preço total do produto comprado e que depende do país de origem;
- ◆ o valor total, preço total do produto mais imposto.


**SOLUÇÃO:**

```

ALGORITMO
DECLARE cod_prod, peso_quilos NUMÉRICO
cod_pais, peso_gramas, pre_total NUMÉRICO
imposto, valor_total, pre_grama NUMÉRICO
LEIA cod_prod, peso_quilos, cod_pais
peso_gramas ← peso_quilos * 1000
ESCREVA peso_gramas
SE (cod_prod >= 1) E (cod_prod <= 4)
    ENTÃO pre_grama ← 10
SE (cod_prod >= 5) E (cod_prod <= 7)
    ENTÃO pre_grama ← 25
SE (cod_prod >= 8) E (cod_prod <= 10)
    ENTÃO pre_grama ← 35

```

```

pre_total ← peso_gramas * pre_grama
ESCREVA pre_total
SE cod_pais = 1
    ENTÃO imposto ← 0
SE cod_pais = 2
    ENTÃO imposto ← pre_total * 15%
SE cod_pais = 3
    ENTÃO imposto ← pre_total * 25%
ESCREVA imposto
valor_total ← pre_total + imposto
ESCREVA valor_total
FIM_ALGORITMO.

```

**1<sup>a</sup> SOLUÇÃO - UTILIZANDO ESTRUTURA CONDICIONAL SIMPLES:**

\EXERC\CAP3\PASCAL\EX20\_A.PAS e \EXERC\CAP3\PASCAL\EX20\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:**

\EXERC\CAP3\PASCAL\EX20\_B.PAS e \EXERC\CAP3\PASCAL\EX20\_B.EXE

**3<sup>a</sup> SOLUÇÃO - UTILIZANDO ESTRUTURA CONDICIONAL SELETORA:**

\EXERC\CAP3\PASCAL\EX20\_C.PAS e \EXERC\CAP3\PASCAL\EX20\_C.EXE

**1<sup>a</sup> SOLUÇÃO - UTILIZANDO ESTRUTURA CONDICIONAL SIMPLES:**

\EXERC\CAP3\C++\EX20\_A.CPP e \EXERC\CAP3\C++\EX20\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:**

\EXERC\CAP3\C++\EX20\_B.CPP e \EXERC\CAP3\C++\EX20\_B.EXE

**3<sup>a</sup> SOLUÇÃO - UTILIZANDO ESTRUTURA CONDICIONAL SELETORA:**

\EXERC\CAP3\C++\EX20\_C.CPP e \EXERC\CAP3\C++\EX20\_C.EXE

**21.** Faça um programa que receba:

- ◆ o código do estado de origem da carga de um caminhão, supondo que a digitação do código do estado seja sempre válida, ou seja, um número inteiro entre 1 e 5;
- ◆ o peso da carga do caminhão em toneladas;
- ◆ o código da carga, supondo que a digitação do código da carga seja sempre válida, ou seja, um número inteiro entre 10 e 40.

Tabelas:

<b>CÓDIGO DO ESTADO</b>	<b>IMPOSTO</b>
1	35%
2	25%
3	15%
4	5%
5	Isento

<b>CÓDIGO DA CARGA</b>	<b>PREÇO POR QUILO</b>
10 a 20	100
21 a 30	250
31 a 40	340

Calcule e mostre:

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

### ALGORITMO

#### SOLUÇÃO:

```

ALGORITMO
DECLARE cod_est, cod_carga, peso_toneladas, peso_quilos NUMÉRICO
      pre_carga, imposto, valor_total NUMÉRICO
LEIA cod_est, peso_toneladas, cod_carga
peso_quilos ← peso_toneladas * 1000
ESCREVA peso_quilos
SE (cod_carga >= 10 ) E (cod_carga <= 20)
    ENTÃO pre_carga ← 100 * peso_quilos
SE (cod_carga >= 21) E (cod_carga <= 30)
    ENTÃO pre_carga ← 250 * peso_quilos
SE (cod_carga >= 31) E (cod_carga <= 40)
    ENTÃO pre_carga ← 340 * peso_quilos
ESCREVA pre_carga
SE cod_est = 1
    ENTÃO imposto ← 35% * pre_carga
SE cod_est = 2
    ENTÃO imposto ← 25% * pre_carga
SE cod_est = 3
    ENTÃO imposto ← 15% * pre_carga
SE cod_est = 4
    ENTÃO imposto ← 5% * pre_carga
SE cod_est = 5
    ENTÃO imposto ← 0
ESCREVA imposto
valor_total ← pre_carga + imposto
ESCREVA valor_total
FIM_ALGORITMO.

```

### RESOLUÇÃO PASCAL

#### SOLUÇÃO:

\EXERC\CAP3\PASCAL\EX21.PAS e \EXERC\CAP3\PASCAL\EX21.EXE

### RESOLUÇÃO C/C++

#### SOLUÇÃO:

\EXERC\CAP3\C++\EX21.CPP e \EXERC\CAP3\C++\EX21.EXE

22. Faça um programa que receba o código, o salário-base e o tempo de serviço de um funcionário. Calcule e mostre:

- ◆ o imposto que está na tabela a seguir.

SALÁRIO-BASE	% SOBRE O SALÁRIO-BASE
< R\$ 200,00	isento
Entre R\$ 200,00 (inclusive) e R\$ 450,00 (inclusive)	3%
Entre R\$ 451,01 e R\$ 700,00	8%
=> R\$ 700,00	12%

- ◆ A gratificação que está na tabela a seguir.

<b>SALÁRIO-BASE</b>	<b>TEMPO DE SERVIÇO</b>	<b>GRATIFICAÇÃO</b>
Superior a R\$ 500,00	Até 3 anos	20
	Mais de 3 anos	30
Até R\$ 500,00	Até 3 anos	23
	Entre 3 e 6 anos	35
	De 6 anos para cima	33

- ◆ O salário líquido, ou seja, salário-base menos imposto mais gratificação;
- ◆ A categoria que está na tabela a seguir.

<b>SALÁRIO LÍQUIDO</b>	<b>CLASSIFICAÇÃO</b>
Até R\$ 350,00	A
Entre R\$ 350,01 e R\$ 600,00	B
De R\$ 600,01 para cima	C

**ALGORITMO****SOLUÇÃO:**

```

ALGORITMO
DECLARE codigo, sal_base, tempo, imposto, grat, sal_liq NUMÉRICO
LEIA codigo, sal_base, tempo
SE sal_base < 200
ENTÃO imposto ← 0
SENÃO SE sal_base <= 450
    ENTÃO imposto ← 3% * sal_base
    SENÃO SE sal_base < 700
        ENTÃO imposto ← 8% * sal_base
        SENÃO imposto ← 12% * sal_base
ESCREVA imposto
SE sal_base > 500
ENTÃO INÍCIO
    SE tempo <= 3
        ENTÃO grat ← 20
        SENÃO grat ← 30
    FIM
SENÃO INÍCIO
    SE tempo <= 3
        ENTÃO grat ← 23
        SENÃO SE tempo < 6
            ENTÃO grat ← 35
            SENÃO grat ← 33
        FIM
    ESCREVA grat
    sal_liq ← sal_base - imposto + grat
    ESCREVA sal_liq
    SE sal_liq <= 350
        ENTÃO ESCREVA "Classificação A"
        SENÃO SE sal_liq < 600
            ENTÃO ESCREVA "Classificação B"
            SENÃO ESCREVA "Classificação C"
    FIM_ALGORITMO.

```

**SOLUÇÃO:**

```
\EXERC\CAP3\PASCAL\EX22.PAS e \EXERC\CAP3\PASCAL\EX22.EXE
```

**SOLUÇÃO:**

```
\EXERC\CAP3\C++\EX22.CPP e \EXERC\CAP3\C++\EX22.EXE
```

- 23.** Faça um programa que receba o valor do salário mínimo, o turno de trabalho (M – Matutino, V – Vespertino ou N – Noturno), a categoria (O – Operário, G – Gerente) e o número de horas trabalhadas no mês de um funcionário. Supondo a digitação apenas de dados válidos e, quando houver digitação de letras, utilize letras maiúsculas. Calcule e mostre:

- ◆ o coeficiente do salário, de acordo com a tabela a seguir.

TURNO DE TRABALHO	VALOR DO COEFICIENTE
M – Matutino	10% do salário mínimo
V – Vespertino	15% do salário mínimo
N – Noturno	12% do salário mínimo

- ◆ o valor do salário bruto, ou seja, o número de horas trabalhadas multiplicado pelo valor do coeficiente do salário.
- ◆ o imposto, de acordo com a tabela a seguir.

CATEGORIA	SALÁRIO BRUTO	IMPOSTO SOBRE O SALÁRIO BRUTO
O – Operário	> = R\$ 300,00	5%
	< R\$ 300,00	3%
G – Gerente	> = R\$ 400,00	6%
	< R\$ 400,00	4%

- ◆ a gratificação, de acordo com as regras a seguir.

Se o funcionário preencher **todos** os requisitos abaixo, sua gratificação será de R\$ 50,00; caso contrário será de R\$ 30,00. Os requisitos são:

**Turno:** Noturno

**Número de horas trabalhadas:** Superior a 80 horas

- ◆ o auxílio-alimentação, de acordo com as regras a seguir.

Se o funcionário preencher **algum** dos requisitos abaixo, seu auxílio-alimentação será de um terço do seu salário bruto; caso contrário será de metade do seu salário bruto. Os requisitos são:

**Categoria:** Operário

**Coeficiente do salário:** < = 25

- ◆ o salário líquido, ou seja, salário bruto menos imposto mais gratificação mais auxílio-alimentação.

- ◆ a classificação, de acordo com a tabela a seguir.

SALÁRIO LÍQUIDO	MENSAGEM
Menor que R\$ 350,00	Mal remunerado
Entre R\$ 350,00 e R\$ 600,00	Normal
Maior que R\$ 600,00	Bem remunerado

**ALGORITMO****SOLUÇÃO:**

```

ALGORITMO
DECLARE sal_min, nht, coeficiente, sal_bruto, imposto, grat NUMÉRICO
auxilio, sal_liq NUMÉRICO
turno, categoria LITERAL
LEIA sal_min, turno, categoria, nht
SE turno = "M"
    ENTÃO coeficiente ← 10% * sal_min
SE turno = "V"
    ENTÃO coeficiente ← 15% * sal_min
SE turno = "N"
    ENTÃO coeficiente ← 12% * sal_min
ESCREVA coeficiente
sal_bruto ← nht * coeficiente
ESCREVA sal_bruto
SE categoria = "O"
    ENTÃO INÍCIO
        SE sal_bruto >= 300
            ENTÃO imposto ← 5% * sal_bruto
            SENÃO imposto ← 3% * sal_bruto
        FIM
    SENÃO INÍCIO
        SE sal_bruto >= 400
            ENTÃO imposto ← 6% * sal_bruto
            SENÃO imposto ← 4% * sal_bruto
        FIM
    ESCRIFA imposto
SE (turno = "N") E (nht > 80)
    ENTÃO grat ← 50
    SENÃO grat ← 30
ESCREVA grat
SE (categoria = "O") OU (coeficiente <= 25)
    ENTÃO auxilio ← 1/3 * sal_bruto
    SENÃO auxilio ← 1/2 * sal_bruto
ESCREVA auxilio
sal_liq ← sal_bruto - imposto + grat + auxilio
ESCREVA sal_liq
SE sal_liq < 350
    ENTÃO ESCRIFA "Mal Remunerado"
SE (sal_liq >= 350) E (sal_liq <= 600)
    ENTÃO ESCRIFA "Normal"
SE sal_liq > 600
    ENTÃO ESCRIFA "Bem Remunerado"
FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP3\PASCAL\EX23.PAS e \EXERC\CAP3\PASCAL\EX23.EXE

**SOLUÇÃO:**

\EXERC\CAP3\C++\EX23.CPP e \EXERC\CAP3\C++\EX23.EXE

**24.** Faça um programa que receba o preço, o tipo (A – Alimentação, L – Limpeza e V – Vestuário) e a refrigeração (S – Produto que necessita de refrigeração e N – Produto que não necessita de refrigeração) de um produto. Suponha que haverá apenas a digitação de dados válidos e, quando houver digitação de letras, utilize letras maiúsculas. Calcule e mostre:

- ◆ o valor adicional, de acordo com a tabela a seguir.

REFRIGERAÇÃO	TIPO	PREÇO	VALOR ADICIONAL
N	A	< R\$ 15,00	R\$ 2,00
		≥ R\$ 15,00	R\$ 5,00
	L	< R\$ 10,00	R\$ 1,50
S	L	≥ R\$ 10,00	R\$ 2,50
		< R\$ 30,00	R\$ 3,00
	V	≥ R\$ 30,00	R\$ 2,50
S	A		R\$ 8,00
	L		R\$ 0,00
	V		R\$ 0,00

- ◆ o valor do imposto, de acordo com a regra a seguir.

PREÇO	PERCENTUAL SOBRE O PREÇO
< R\$ 25,00	5%
≥ R\$ 25,00	8%

- ◆ o preço de custo, ou seja, preço mais imposto.
- ◆ o desconto, de acordo com a regra a seguir.

O produto que **não** preencher nenhum dos requisitos abaixo terá desconto de 3%, caso contrário 0 (zero).

Os requisitos são:

**Tipo:** A

**Refrigeração:** S

- ◆ o novo preço, ou seja, preço mais adicional menos desconto.
- ◆ a classificação, de acordo com a regra a seguir.

NOVO PREÇO	CLASSIFICAÇÃO
≤ R\$ 50,00	Barato
Entre R\$ 50,00 e R\$ 100,00	Normal
≥ R\$ 100,00	Caro

## ALGORITMO

### SOLUÇÃO:

ALGORITMO

```
DECLARE pre, valor_adic, imposto NUMÉRICO
      pre_custo, desconto, novo_pre NUMÉRICO
      tipo, refrig LITERAL
```

```

LEIA pre, tipo, refrig
SE refrig = "N"
    ENTÃO INÍCIO
        SE tipo = "A"
            ENTÃO INÍCIO
                SE pre < 15
                    ENTÃO valor_adic ← 2
                    SENÃO valor_adic ← 5
                FIM
        SE tipo = "L"
            ENTÃO INÍCIO
                SE pre < 10
                    ENTÃO valor_adic ← 1,50
                    SENÃO valor_adic ← 2,50
                FIM
        SE tipo = "V"
            ENTÃO INÍCIO
                SE pre < 30
                    ENTÃO valor_adic ← 3
                    SENÃO valor_adic ← 2,5
                FIM
            FIM
        SENÃO INÍCIO
            SE tipo = "A"
                ENTÃO valor_adic ← 8
            SE tipo = "L"
                ENTÃO valor_adic ← 0
            SE tipo = "V"
                ENTÃO valor_adic ← 0
            FIM
        ESCREVA valor_adic
    SE pre < 25
        ENTÃO imposto ← 5% * pre
        SENÃO imposto ← 8% * pre
    ESCREVA imposto
    pre_custo ← pre + imposto
    ESCREVA pre_custo
    SE (tipo ≠ "A") E (refrig ≠ "S")
        ENTÃO desconto ← 3% * pre_custo
        SENÃO desconto ← 0
    ESCREVA desconto
    novo_pre ← pre + valor_adic - desconto
    ESCREVA novo_pre
    SE novo_pre <= 50
        ENTÃO ESCREVA "Barato"
        SENÃO SE novo_pre < 100
            ENTÃO ESCREVA "Normal"
            SENÃO ESCREVA "Caro"
    FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP3\PASCAL\EX24.PAS e \EXERC\CAP3\PASCAL\EX24.EXE

**SOLUÇÃO:**

\EXERC\CAP3\C++\EX24.CPP e \EXERC\CAP3\C++\EX24.EXE

- 25.** Faça um programa que receba a medida de um ângulo em graus. Calcule e mostre o quadrante em que se localiza esse ângulo. Considere os quadrantes da trigonometria e para ângulos maiores que  $360^\circ$  ou menores que  $-360^\circ$ , reduzi-los, mostrando também o número de voltas e o sentido da volta (horário ou anti-horário).

**SOLUÇÃO:**

```

ALGORITMO
DECLARE angulo, voltas NUMÉRICO
LEIA angulo
SE (angulo > 360) OU (angulo < -360)
    ENTÃO INÍCIO
        voltas ← angulo / 360
        angulo ← RESTO(angulo / 360)
    FIM
    SENÃO voltas ← 0
    SE (angulo = 0) OU (angulo = 90) OU (angulo = 180)
        OU (angulo = 270) OU (angulo = 360)
        OU (angulo = -90) OU (angulo = -180)
        OU (angulo = -270) OU (angulo = -360)
        ENTÃO ESCREVA "Está em cima de algum dos eixos"
    SE ((angulo > 0) E (angulo < 90)) OU ((angulo < -270) E (angulo > -360))
        ENTÃO ESCREVA "1º Quadrante"
    SE ((angulo > 90) E (angulo < 180)) OU ((angulo < -180) E (angulo > -270))
        ENTÃO ESCREVA "2º Quadrante"
    SE ((angulo > 180) E (angulo < 270)) OU ((angulo < -90) E (angulo > -180))
        ENTÃO ESCREVA "3º Quadrante"
    SE ((angulo > 270) E (angulo < 360)) OU ((angulo < 0) E (angulo > -90))
        ENTÃO ESCREVA "4º Quadrante"
    ESCREVA voltas, " volta(s) no sentido "
    SE (angulo < 0)
        ENTÃO ESCREVA "horário"
        SENÃO ESCREVA "anti-horário"
    FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP3\PASCAL\EX25.PAS e \EXERC\CAP3\PASCAL\EX25.EXE

**SOLUÇÃO:**

\EXERC\CAP3\C++\EX25.CPP e \EXERC\CAP3\C++\EX25.EXE

## EXERCÍCIOS PROPOSTOS

1. Faça um programa que receba quatro notas de um aluno, calcule e mostre a média aritmética das notas e a mensagem de aprovado ou reprovado, considerando para aprovação média 7.
2. Faça um programa que receba duas notas, calcule e mostre a média aritmética e a mensagem que está na tabela a seguir:

<b>MÉDIA ARITMÉTICA</b>	<b>MENSAGEM</b>
0,0    ●—○  4,0	Reprovado
4,0    ●—○  7,0	Exame
7,0    ●—●  10,0	Aprovado

3. Faça um programa que receba dois números e mostre o menor.
4. Faça um programa que receba três números e mostre o maior.
5. Faça um programa que receba dois números e execute as operações listadas a seguir de acordo com a escolha do usuário.

ESCOLHA DO USUÁRIO	OPERAÇÃO
1	Média entre os números digitados
2	Diferença do maior pelo menor
3	Produto entre os números digitados
4	Divisão do primeiro pelo segundo

Se a opção digitada for inválida, mostrar uma mensagem de erro e terminar a execução do programa. Lembre-se de que na operação 4 o segundo número deve ser diferente de zero.

6. Faça um programa que receba dois números e execute uma das operações listadas a seguir de acordo com a escolha do usuário. Se for digitada uma opção inválida mostrar mensagem de erro e terminar a execução do programa. As opções são:

1. Média entre os dois números.
2. Diferença do maior pelo menor.
3. O produto entre os dois números.
7. Uma empresa decide dar um aumento de 30% aos funcionários com salários inferiores a R\$ 500,00. Faça um programa que receba o salário do funcionário e mostre o valor do salário reajustado ou uma mensagem, caso o funcionário não tenha direito ao aumento.
8. Faça um programa para calcular e mostrar o salário reajustado de um funcionário. Sabe-se que o percentual de aumento é o mesmo da tabela a seguir.

SALÁRIO	PERCENTUAL DE AUMENTO
Até R\$ 300,00	35%
Acima de R\$ 300,00	15%

9. Um banco concederá um crédito especial aos seus clientes de acordo com o saldo médio no último ano. Faça um programa que receba o saldo médio de um cliente e calcule o valor do crédito, de acordo com a tabela a seguir. Mostre o saldo médio e o valor do crédito.

SALDO MÉDIO	PERCENTUAL
Acima de R\$ 400,00	30% do saldo médio
R\$ 400,00  R\$ 300,00	25% do saldo médio
R\$ 300,00  R\$ 200,00	20% do saldo médio
Até R\$ 200,00	10% do saldo médio

10. O custo ao consumidor de um carro novo é a soma do custo de fábrica com a porcentagem do distribuidor e com os impostos, ambos aplicados ao custo de fábrica. Sabe-se que as porcentagens são as mesmas que estão na tabela a seguir. Faça um programa que receba o custo de fábrica de um carro e mostre o custo ao consumidor.

CUSTO DE FÁBRICA	% DO DISTRIBUIDOR	% DOS IMPOSTOS
Até R\$ 12.000,00	5	isento
Entre R\$ 12.000,00 e R\$ 25.000,00	10	15
Acima de R\$ 25.000,00	15	20

**11.** Faça um programa que receba o salário de um funcionário e, usando a tabela a seguir, calcule e mostre o valor do aumento e o novo salário.

SALÁRIO	PERCENTUAL DE AUMENTO
Até R\$ 300,00	15
R\$ 300,00 ○—● R\$ 600,00	10
R\$ 600,00 ○—● R\$ 900,00	5
Acima de R\$ 900,00	0

**12.** Faça um programa que receba o salário de um funcionário e, usando a tabela a seguir, calcule e mostre o valor a receber. Sabe-se que este é composto pelo salário do funcionário acrescido de gratificação e descontado o imposto de 7% sobre o salário sem gratificação.

TABELA DAS GRATIFICAÇÕES	
SALÁRIO	GRATIFICAÇÃO
Até R\$ 350,00	R\$ 100,00
R\$ 350,00 ○—● R\$ 600,00	R\$ 75,00
R\$ 600,00 ○—● R\$ 900,00	R\$ 50,00
Acima de R\$ 900,00	R\$ 35,00

**13.** Faça um programa que receba o preço de um produto, calcule e mostre, de acordo com as tabelas a seguir, o novo preço e a classificação.

TABELA 1 - PERCENTUAL DE AUMENTO	
PREÇO	%
Até R\$ 50,00	5
Entre R\$ 50,00 e R\$ 100,00	10
Acima de R\$ 100,00	15

TABELA 2 - CLASSIFICAÇÕES	
NOVO PREÇO	CLASSIFICAÇÃO
Até R\$ 80,00	Barato
Entre R\$ 80,00 e R\$ 120,00 (inclusive)	Normal
Entre R\$ 120,00 e R\$ 200,00 (inclusive)	Caro
Maior que R\$ 200,00	Muito caro

**14.** Faça um programa que receba o salário de um funcionário e, usando a tabela a seguir, calcule e mostre o novo salário.

FAIXA SALARIAL	% DE AUMENTO
Até R\$ 300,00	50%
R\$ 300,00 <input type="radio"/> R\$ 500,00	40%
R\$ 500,00 <input type="radio"/> R\$ 700,00	30%
R\$ 700,00 <input type="radio"/> R\$ 800,00	20%
R\$ 800,00 <input type="radio"/> R\$ 1.000,00	10%
Acima de R\$ 1.000,00	5%

**15.** Uma agência bancária possui dois tipos de investimentos, conforme o quadro a seguir. Faça um programa que receba o tipo de investimento e o valor do investimento e que calcule e mostre o valor corrigido de acordo com o tipo de investimento.

TIPO	DESCRIÇÃO	RENDIMENTO MENSAL
1	Poupança	3%
2	Fundos de renda fixa	4%

**16.** Uma empresa decide aplicar descontos nos seus preços usando a tabela a seguir. Faça um programa que receba o preço atual de um produto e seu código e que calcule e mostre o preço atual, o valor do desconto e o novo preço.

PREÇO ATUAL	% DE DESCONTO
Até R\$ 30,00	Sem desconto
Entre R\$ 30,00 e R\$ 100,00	10
Acima de R\$ 100,00	15

**17.** Faça um programa que verifique a validade de uma senha fornecida pelo usuário. A senha é 4531. O programa deve mostrar uma mensagem de permissão de acesso ou não.

**18.** Faça um programa que receba a idade de uma pessoa e mostre a mensagem de maioriade ou não.

**19.** Faça um programa que receba a altura e o sexo de uma pessoa e que calcule e mostre o seu peso ideal, utilizando as seguintes fórmulas:

- ◆ para homens:  $(72.7 * h) - 58$ ;
- ◆ para mulheres:  $(62.1 * h) - 44.7$

**20.** Faça um programa que receba a idade de um nadador e mostre a sua categoria usando as regras a seguir.

CATEGORIA	IDADE
Infantil	5 a 7
Juvenil	8 a 10
Adolescente	11 a 15
Adulto	16 a 30
Sênior	Acima de 30

**21.** Faça um programa que receba o preço de um produto e o seu código de origem e mostre a sua procedência. A procedência obedece à tabela a seguir.

CÓDIGO DE ORIGEM	PROCEDÊNCIA
1	Sul
2	Norte
3	Leste
4	Oeste
5 ou 6	Nordeste
7 ou 8 ou 9	Sudeste
10 a 20	Centro-oeste
21 a 30	Nordeste

**22.** Faça um programa que receba a idade e o peso de uma pessoa. De acordo com a tabela a seguir, verifique e mostre em qual grupo de risco essa pessoa se encaixa.

IDADE	PESO		
	ATÉ 60	ENTRE 60 E 90 (INCLUSIVE)	ACIMA DE 90
Menores de 20	9	8	7
De 20 a 50	6	5	4
Maiores de 50	3	2	1

**23.** Faça um programa que receba:

- ◆ o código do produto comprado;
- ◆ a quantidade comprada de um produto.

Calcule e mostre:

- ◆ o preço unitário do produto comprado seguindo a Tabela I;
- ◆ o preço total da nota;
- ◆ o valor do desconto, seguindo a Tabela II e aplicado sobre o preço total da nota;
- ◆ o preço final da nota depois do desconto.

TABELA I	
CÓDIGO	PREÇO
1 a 10	R\$ 10,00
11 a 20	R\$ 15,00
21 a 30	R\$ 20,00
31 a 40	R\$ 30,00

TABELA II	
PREÇO TOTAL DA NOTA	% DE DESCONTO
Até R\$ 250,00	5%
Entre R\$ 250,00 e R\$ 500,00	10%
De R\$ 500,00 para cima	15%

**24.** Faça um programa que receba o preço, a categoria (1 – limpeza, 2 – alimentação ou 3 – vestuário) e a situação (R – produtos que necessitam de refrigeração e N – produtos que não necessitam de refrigeração). Calcule e mostre:

- ◆ o valor do aumento, usando as regras a seguir sobre o preço.

PREÇO	CATEGORIA	PERCENTUAL DE AUMENTO
	1	5%
$\leq R\$ 25,00$	2	8%
	3	10%
	1	12%
$> R\$ 25,00$	2	15%
	3	18%

- ♦ o valor do imposto, usando as seguintes regras.

O produto que preencher **pelo menos** um dos seguintes requisitos pagará imposto equivalente a 5% do preço, caso contrário pagará 8% do preço. Os requisitos são:

**Categoria:** 2

**Situação:** R

- ♦ o novo preço, ou seja, o preço mais aumento menos imposto.
- ♦ a classificação, de acordo com as regras a seguir.

NOVO PREÇO	CLASSIFICAÇÃO
$\leq R\$ 50,00$	Barato
Entre $R\$ 50,00$ e $R\$ 120,00$	Normal
$\geq R\$ 120,00$	Caro

**25.** Uma empresa decidiu dar uma gratificação de natal aos seus funcionários, baseada no número de horas extras e no número de horas que o funcionário faltou ao trabalho. O valor do prêmio é obtido pela consulta na tabela a seguir, em que:

$$H = (\text{número de horas extras}) - 2/3 * ((\text{número de horas-falta}))$$

H	GRATIFICAÇÃO
$> 2.400$	R\$ 500,00
1.800 ●—○ 2.400	R\$ 400,00
1.200 ●—● 1.800	R\$ 300,00
600 ●—○ 1.200	R\$ 200,00
$< 600$	R\$ 100,00

## CAPÍTULO

# 4

# ESTRUTURA DE REPETIÇÃO

## 4.1 ESTRUTURA DE REPETIÇÃO EM ALGORITMO

### 4.1.1 ESTRUTURA DE REPETIÇÃO PARA NÚMERO DEFINIDO DE REPETIÇÕES (ESTRUTURA PARA)

Essa estrutura de repetição é utilizada quando se sabe o número de vezes em que um trecho do algoritmo deve ser repetido.

```
PARA I ← valor inicial ATÉ valor final FAÇA  
    comando1
```

O **comando1** será executado utilizando a variável **I** como controle, cujo conteúdo vai variar do **valor inicial** até o **valor final**, de 1 em 1, incrementando automaticamente.

```
PARA J ← valor inicial ATÉ valor final FAÇA  
    INÍCIO  
        comando1  
        comando2  
    FIM
```

Os **comando1** e **comando2** serão executados utilizando a variável **J** como controle, cujo conteúdo vai variar do **valor inicial** até o **valor final**, de 1 em 1, incrementando automaticamente.

Exemplo:

```
PARA I ← 1 ATÉ 10 FAÇA  
    comando1
```

O **comando1** será executado dez vezes.

### 4.1.2 ESTRUTURA DE REPETIÇÃO PARA NÚMERO INDEFINIDO DE REPETIÇÕES E TESTE NO INÍCIO (ESTRUTURA ENQUANTO)

Essa estrutura de repetição é utilizada quando *não* se sabe o número de vezes em que um trecho do algoritmo deve ser repetido, embora também possa ser utilizada quando se sabe esse número.

Existem situações em que o teste condicional da estrutura de repetição, que fica no início, resulta em um valor falso logo na primeira comparação. Nesses casos, os comandos de dentro da estrutura de repetição não serão executados.

```
ENQUANTO condição FAÇA
    comando1
```

Enquanto a **condição** for verdadeira, o **comando1** será executado.

```
ENQUANTO condição FAÇA
    INÍCIO
        comando1
        comando2
        comando3
    FIM
```

Enquanto a **condição** for verdadeira, os **comando1**, **comando2** e **comando3** serão executados.

Exemplo:

```
X ← 1
Y ← 5
ENQUANTO X < Y FAÇA
    INÍCIO
        X ← X + 2
        Y ← Y + 1
    FIM
```

Simulação:

X	Y	
1	5	Valores iniciais
3	6	
5	7	
7	8	Valores obtidos dentro da estrutura de repetição
9	9	

Portanto, no trecho do algoritmo anterior, os comandos que se localizam dentro da estrutura de repetição serão repetidos quatro vezes.

#### 4.1.3 ESTRUTURA DE REPETIÇÃO PARA NÚMERO INDEFINIDO DE REPETIÇÕES E TESTE NO FINAL (ESTRUTURA REPITA)

Essa estrutura de repetição é utilizada quando *não* se sabe o número de vezes em que um trecho do algoritmo deve ser repetido, embora também possa ser utilizada quando se sabe esse número.

A diferença entre a estrutura **ENQUANTO** e a estrutura **REPITA** é que na estrutura **REPITA** os comandos serão repetidos pelo menos uma vez, já que a condição se encontra no final.

```
REPITA
    comandos
    ATÉ condição
```

Reita os **comandos** até a **condição** se tornar verdadeira.

Exemplo:

```
X ← 1
Y ← 5
REPITA
    X ← X + 2
    Y ← Y + 1
    ATÉ X >= Y
```

Simulação:

X	Y	
1	5	Valores iniciais
3	6	
5	7	
7	8	Valores obtidos dentro da estrutura de repetição
9	9	

Portanto, no trecho do algoritmo anterior, os comandos que se localizam dentro da estrutura de repetição serão repetidos quatro vezes.

## 4.2 ESTRUTURA DE REPETIÇÃO EM PASCAL

### 4.2.1 ESTRUTURA DE REPETIÇÃO FOR

Essa estrutura de repetição é utilizada quando se sabe o número de vezes em que um trecho do programa deve ser repetido.

```
FOR I := valor inicial TO valor final DO
  comando;
```

O **comando** será executado utilizando a variável **I** como controle, cujo conteúdo vai variar do **valor inicial** até o **valor final**, de 1 em 1, incrementando automaticamente.

```
FOR J := valor inicial TO valor final DO
BEGIN
  comando1;
  comando2;
END;
```

Os **comando1** e **comando2** serão executados utilizando a variável **J** como controle, cujo conteúdo vai variar do **valor inicial** até o **valor final**, de 1 em 1, incrementando automaticamente.

```
FOR K := valor inicial DOWNTO valor final DO
  comando;
```

O **comando** será executado utilizando a variável **K** como controle, cujo conteúdo vai variar do **valor inicial** até o **valor final**, de 1 em 1, decrementando automaticamente.

```
FOR H := valor inicial DOWNTO valor final DO
BEGIN
  comando1;
  comando2;
  comando3;
END;
```

Os **comando1**, **comando2** e **comando3** serão executados utilizando a variável **H** como controle, cujo conteúdo vai variar do **valor inicial** até o **valor final**, de 1 em 1, decrementando automaticamente.

### 4.2.2 ESTRUTURA DE REPETIÇÃO WHILE

```
WHILE condição DO
  comando;
```

Enquanto a **condição** for verdadeira, o **comando** será executado.

```
WHILE condição DO
BEGIN
    comando1;
    comando2;
END;
```

Enquanto a **condição** for verdadeira, os **comando1** e **comando2** serão executados.

#### 4.2.3 ESTRUTURA DE REPETIÇÃO REPEAT

```
REPEAT
comandos;
UNTIL condição;
```

Os **comandos** serão repetidos até que a **condição** se torne verdadeira.

### 4.3 ESTRUTURA DE REPETIÇÃO EM C/C++

#### 4.3.1 ESTRUTURA DE REPETIÇÃO FOR

Essa estrutura de repetição é utilizada quando se sabe o número de vezes em que um trecho do programa deve ser repetido.

O formato geral do comando **for** é composto por três partes:

```
for (i=valor inicial; condição; incremento ou decrecemento de i)
    comando;
```

A primeira parte atribui um valor inicial à variável **i**, que tem como função controlar o número necessário de repetições.

A segunda parte corresponde a uma expressão relacional que, quando assumir valor falso, determinará o fim da repetição.

A terceira parte é responsável por alterar o valor da variável **i** (incremento ou decrecemento) com o objetivo de, em algum momento, fazer com que a condição assuma valor falso.

Caso seja necessária a repetição de apenas um comando, o compilador entenderá que a estrutura de repetição terminará quando for encontrado o primeiro ; (ponto-e-vírgula).

Exemplo:

```
for (a=1;a<=20;a++)
    cout << "\nO valor de a é: " << a;
```

No exemplo anterior, a variável **a** é inicializada com o valor 1 e vai sendo incrementada em uma unidade e, a cada incremento, o comando **cout** é executado. Esse processo se repete até o valor da variável **a** se tornar maior que 20 (quando a condição **a<=20** assumir valor falso).

Caso seja necessária a repetição de mais de um comando, o compilador entenderá que a estrutura de repetição começará quando for encontrado o símbolo { e terminará quando for encontrado o símbolo }.

Exemplo:

```
for (a=15;a>=1;a=a-2)
{ cout << "Digite um número: ";
    cin >> x;
}
```

No exemplo anterior, a variável **a** é inicializada com o valor 15 e vai sendo decrementada em duas unidades e, a cada decremento, o bloco de comando que está entre as chaves { ... } é executado. Esse processo se repete até o valor da variável **a** se tornar menor que 1 (quando a condição **a>=1** assumir valor falso).

#### 4.3.2 ESTRUTURA DE REPETIÇÃO WHILE

Trata-se de uma estrutura de repetição que pode ser utilizada quando o número de repetições necessárias não é fixo. Os comandos serão repetidos até a condição assumir o valor falso.

Nesse tipo de estrutura o teste condicional ocorre no início, isso significa que existe a possibilidade da repetição não ser executada, quando a condição assumir valor falso logo na primeira verificação.

```
while (condição)
    comando;
```

Enquanto a **condição** for verdadeira o **comando** será executado.

```
while (condição)
    { comando1;
        comando2;
        comando3;
        ...
    }
```

Enquanto a **condição** for verdadeira, os **comandos** que estão dentro das chaves serão executados (**comando1**, **comando2**, **comando3**, ...).

#### 4.3.3 ESTRUTURA DE REPETIÇÃO DO-WHILE

Trata-se de uma estrutura de repetição que pode ser utilizada quando o número de repetições necessárias não é fixo. Os comandos serão repetidos até a condição assumir o valor falso.

Nesse tipo de estrutura o teste condicional ocorre no fim. Isso significa que a repetição será executada no mínimo uma vez, quando todo o bloco for executado uma vez e, ao final, a condição assumir valor falso.

```
do
    {
        comandos;
    }
    while (condição);
```

Os **comandos** serão repetidos até que a **condição** assuma valor falso.

### EXERCÍCIOS RESOLVIDOS

1. Um funcionário de uma empresa recebe aumento salarial anualmente. Sabe-se que:
  - a) esse funcionário foi contratado em 1995, com salário inicial de R\$ 1.000,00;
  - b) em 1996 recebeu aumento de 1,5% sobre seu salário inicial;
  - c) a partir de 1997 (inclusive), os aumentos salariais sempre corresponderam ao dobro do percentual do ano anterior.

Faça um programa que determine o salário atual desse funcionário.

**SOLUÇÃO:**

```

ALGORITMO
DECLARE i, ano_atual, salario, novo_salario, percentual NUMÉRICO
LEIA ano_atual
salario ← 1000
percentual ← 1,5%
novo_salario ← salario + percentual * salario
PARA i ← 1997 ATÉ ano_atual FAÇA
    INÍCIO
        percentual ← 2 * percentual
        novo_salario ← novo_salario + percentual * novo_salario
    FIM
ESCREVA novo_salario
FIM_ALGORITMO.

```

**1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP4\PASCAL\EX1\_A.PAS e \EXERC\CAP4\PASCAL\EX1\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP4\PASCAL\EX1\_B.PAS e \EXERC\CAP4\PASCAL\EX1\_B.EXE

**1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP4\C++\EX1\_A.CPP e \EXERC\CAP4\C++\EX1\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP4\C++\EX1\_B.CPP e \EXERC\CAP4\C++\EX1\_B.EXE

2. Faça um programa que leia um valor N inteiro e positivo, calcule e mostre o valor de E, conforme a fórmula a seguir:

$$E = 1 + 1/1! + 1/2! + 1/3! + \dots + 1/N!$$

**SOLUÇÃO:**

```

ALGORITMO
DECLARE n, e, i, j, fat NUMÉRICO
LEIA n
e ← 1
PARA i ← 1 ATÉ n FAÇA
    INÍCIO
        fat ← 1
        PARA j ← 1 ATÉ i FAÇA
            INÍCIO
                fat ← fat * j
            FIM
        e ← e + 1/fat
    FIM
ESCREVA e
FIM_ALGORITMO.

```

**1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP4\PASCAL\EX2\_A.PAS e \EXERC\CAP4\PASCAL\EX2\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA REPEAT:**

\EXERC\CAP4\PASCAL\EX2\_B.PAS e \EXERC\CAP4\PASCAL\EX2\_B.EXE

**1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP4\C++\EX2\_A.CPP e \EXERC\CAP4\C++\EX2\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA DO-WHILE:**

\EXERC\CAP4\C++\EX2\_B.CPP e \EXERC\CAP4\C++\EX2\_B.EXE

- 3.** Faça um programa que leia um número N e que indique quantos valores inteiros e positivos devem ser lidos a seguir. Para cada número lido, mostre uma tabela contendo o valor lido e o fatorial desse valor.

**SOLUÇÃO:**

```

ALGORITMO
DECLARE n, num, i, j, fat NUMÉRICO
LEIA n
PARA i ← 1 ATÉ n FAÇA
    INÍCIO
    LEIA num
    fat ← 1
    PARA j ← 1 ATÉ num FAÇA
        INÍCIO
        fat ← fat * j
        FIM
    ESCREVA fat
    FIM
FIM_ALGORITMO.

```

**1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP4\PASCAL\EX3\_A.PAS e \EXERC\CAP4\PASCAL\EX3\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP4\PASCAL\EX3\_B.PAS e \EXERC\CAP4\PASCAL\EX3\_B.EXE

**1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP4\C++\EX3\_A.CPP e \EXERC\CAP4\C++\EX3\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP4\C++\EX3\_B.CPP e \EXERC\CAP4\C++\EX3\_B.EXE

- 4.** Faça um programa que leia cinco pares de valores (a,b), todos inteiros e positivos, um de cada vez. Mostre os números inteiros pares de a até b (inclusive).

**SOLUÇÃO:**

```

ALGORITMO
DECLARE cont, a, b, i NUMÉRICO
PARA CONT ← 1 ATÉ 5 FAÇA

```

```

INÍCIO
LEIA a, b
PARA i ← a ATÉ b FAÇA
    INÍCIO
        SE RESTO(i/2) = 0
        ENTÃO ESCREVA i
    FIM
FIM
FIM_ALGORITMO.

```



**1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP4\PASCAL\EX4\_A.PAS e \EXERC\CAP4\PASCAL\EX4\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA REPEAT:**

\EXERC\CAP4\PASCAL\EX4\_B.PAS e \EXERC\CAP4\PASCAL\EX4\_B.EXE



**1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP4\C++\EX4\_A.CPP e \EXERC\CAP4\C++\EX4\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA DO-WHILE:**

\EXERC\CAP4\C++\EX4\_B.CPP e \EXERC\CAP4\C++\EX4\_B.EXE

5. Faça um programa que leia dez conjuntos de dois valores, o primeiro representando o número do aluno e o segundo representando a sua altura em centímetros. Encontre o aluno mais alto e o mais baixo. Mostre o número do aluno mais alto e o número do mais baixo, junto com suas alturas.



**SOLUÇÃO:**

```

ALGORITMO
DECLARE cont, num, alt, maior, num_maior, menor, num_menor NUMÉRICO
PARA cont ← 1 ATÉ 10 FAÇA
    INÍCIO
        LEIA num, alt
        SE cont = 1
        ENTÃO INÍCIO
            maior ← alt
            num_maior ← num
            menor ← alt
            num_menor ← num
        FIM
        SENÃO INÍCIO
            SE alt > maior
            ENTÃO INÍCIO
                maior ← alt
                num_maior ← num
            FIM
            SE alt < menor
            ENTÃO INÍCIO
                menor ← alt
                num_menor ← num
            FIM
        FIM
    FIM
ESCREVA maior, num_maior

```

ESCREVA menor, num\_menor  
FIM\_ALGORITMO.



**1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP4\PASCAL\EX5\_A.PAS e \EXERC\CAP4\PASCAL\EX5\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP4\PASCAL\EX5\_B.PAS e \EXERC\CAP4\PASCAL\EX5\_B.EXE



**1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP4\C++\EX5\_A.CPP e \EXERC\CAP4\C++\EX5\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP4\C++\EX5\_B.CPP e \EXERC\CAP4\C++\EX5\_B.EXE

6. Foi feita uma estatística em cinco cidades brasileiras para coletar dados sobre acidentes de trânsito. Foram obtidos os seguintes dados:

- a) código da cidade;
- b) número de veículos de passeio (em 1999);
- c) número de acidentes de trânsito com vítimas (em 1999).

Deseja-se saber:

- a) qual o maior e o menor índice de acidentes de trânsito e a que cidades pertencem;
- b) qual a média de veículos nas cinco cidades juntas;
- c) qual a média de acidentes de trânsito nas cidades com menos de 2.000 veículos de passeio.



**SOLUÇÃO:**

```

ALGORITMO
DECLARE cont, cod, num_vei, num_acid NUMÉRICO
        maior, cid_maior, menor, cid_menor NUMÉRICO
        media_vei, soma_vei, media_acid NUMÉRICO
        soma_acid, cont_acid NUMÉRICO
soma_vei ← 0
soma_acid ← 0
cont_acid ← 0
PARA cont ← 1 ATÉ 5 FAÇA
    INÍCIO
        LEIA cod, num_vei, num_acid
        SE cont = 1
        ENTÃO INÍCIO
            maior ← num_acid
            cid_maior ← cod
            menor ← num_acid
            cid_menor ← cod
            FIM
        SENÃO INÍCIO
            SE num_acid > maior
            ENTÃO INÍCIO
                maior ← num_acid
                cid_maior ← cod
            FIM
        FIM
    FIM

```

```

SE num_acid < menor
ENTÃO INÍCIO
    menor ← num_acid
    cid_menor ← cod
FIM
FIM
soma_vei ← soma_vei + num_vei
SE num_vei < 2000
    ENTÃO INÍCIO
        soma_acid ← soma_acid + num_acid
        cont_acid ← cont_acid + 1
    FIM
FIM
ESCREVA maior, cid_maior
ESCREVA menor, cid_menor
media_vei ← soma_vei/5
ESCREVA media_vei
SE cont_acid = 0
    ENTÃO ESCRIVA "Não foi digitada nenhuma cidade com menos de 2000
    ↪ veículos"
    SENÃO INÍCIO
        media_acid ← soma_acid/cont_acid
        ESCRIVA media_acid
    FIM
FIM_ALGORITMO.

```



#### 1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP4\PASCAL\EX6\_A.PAS e \EXERC\CAP4\PASCAL\EX6\_A.EXE

#### 2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA REPEAT:

\EXERC\CAP4\PASCAL\EX6\_B.PAS e \EXERC\CAP4\PASCAL\EX6\_B.EXE



#### 1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP4\C++\EX6\_A.CPP e \EXERC\CAP4\C++\EX6\_A.EXE

#### 2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP4\C++\EX6\_B.CPP e \EXERC\CAP4\C++\EX6\_B.EXE

7. Faça um programa que leia o número de termos e um valor positivo para X, calcule e mostre o valor da série a seguir:

$$S = -\frac{X^2}{1!} + \frac{X^3}{2!} - \frac{X^4}{3!} + \frac{X^5}{4!} - \frac{X^6}{3!} + \frac{X^7}{2!} - \frac{X^8}{1!} + \frac{X^9}{2!} - \frac{X^{10}}{3!} + \frac{X^{11}}{4!} - \dots$$



#### SOLUÇÃO:

```

ALGORITMO
DECLARE fim, i, j, x, expoente, num_termos NUMÉRICO
den, denominador, fat, s NUMÉRICO
LEIA num_termos, x
s ← 0
denominador ← 1
PARA i ← 1 TO num_termos FAÇA
INÍCIO

```

```

fim ← denominador
fat ← 1
PARA    j ← 1 ATÉ fim FAÇA
    INÍCIO
        fat ← fat * j
    FIM
expoente ← i + 1
SE RESTO (expoente/ 2) = 0
ENTÃO    s ← s - xexpoente/fat
SENÃO    s ← s + xexpoente /fat
SE denominador = 4
ENTÃO den ← -1
SE denominador = 1
ENTÃO den ← 1
SE den = 1
ENTÃO denominador ← denominador + 1
SENÃO denominador ← denominador - 1
FIM
ESCREVA s
FIM_ALGORITMO.

```


**1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP4\PASCAL\EX7\_A.PAS e \EXERC\CAP4\PASCAL\EX7\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP4\PASCAL\EX7\_B.PAS e \EXERC\CAP4\PASCAL\EX7\_B.EXE


**1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP4\C++\EX7\_A.CPP e \EXERC\CAP4\C++\EX7\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP4\C++\EX7\_B.CPP e \EXERC\CAP4\C++\EX7\_B.EXE

- 8.** Uma empresa possui dez funcionários com as seguintes características: código, número de horas trabalhadas no mês, turno de trabalho (M – Matutino, V – Vespertino ou N – Noturno), categoria (O – Operário ou G – Gerente), valor da hora trabalhada. Sabendo-se que essa empresa deseja informatizar sua folha de pagamento, faça um programa que:

- leia as informações dos funcionários, exceto o valor da hora trabalhada, não permitindo que sejam informados turnos nem categorias inexistentes. Trabalhar sempre com a digitação de letras maiúsculas;
- calcule o valor da hora trabalhada, conforme a tabela a seguir.

CATEGORIA	TURNO	VALOR DA HORA TRABALHADA
G	N	18% do salário mínimo
G	M ou V	15% do salário mínimo
O	N	13% do salário mínimo
O	M ou V	10% do salário mínimo

Adote o valor de R\$ 150,00 para o salário mínimo.

- calcule o salário inicial dos funcionários com base no valor da hora trabalhada e no número de horas trabalhadas;

- d) calcule o valor do auxílio-alimentação recebido por funcionário de acordo com o seu salário inicial, conforme a tabela a seguir.

SALÁRIO INICIAL	AUXÍLIO-ALIMENTAÇÃO
Até R\$ 300,00	20% do salário inicial
Entre R\$ 300,00 e R\$ 600,00	15% do salário inicial
Acima de R\$ 600,00	5% do salário inicial

- e) mostre o código, número de horas trabalhadas, valor da hora trabalhada, salário inicial, auxílio-alimentação e o salário final (salário inicial + auxílio-alimentação).

### ALGORITMO

#### SOLUÇÃO:

```

ALGORITMO
DECLARE cont, codigo, nht, valor NUMÉRICO
sal_min, sal_inicial, aux, sal_final NUMÉRICO
turno, categoria LITERAL
sal_min ← 150
PARA cont ← 1 ATÉ 10 FAÇA
INÍCIO
    LEIA codigo, nht, turno, categoria
    ENQUANTO (turno ≠ "M") E (turno ≠ "V") E (turno ≠ "N") FAÇA
        INÍCIO
            LEIA turno
            FIM
        ENQUANTO (categoria ≠ "G") E (categoria ≠ "O") FAÇA
            INÍCIO
                LEIA categoria
                FIM
            SE categoria = "G"
            ENTÃO INÍCIO
                SE turno = "N"
                    ENTÃO valor ← 18% * sal_min
                    SENÃO valor ← 15% * sal_min
                FIM
            SENÃO INÍCIO
                SE turno = "N"
                    ENTÃO valor ← 13% * sal_min
                    SENÃO valor ← 10% * sal_min
                FIM
            sal_inicial ← nht * valor
            SE sal_inicial <= 300
                ENTÃO aux ← 20% * sal_inicial
                SENÃO SE sal_inicial < 600
                    ENTÃO aux ← 15% * sal_inicial
                    SENÃO aux ← 5% * sal_inicial
                sal_final ← sal_inicial + aux
                ESCREVA codigo, nht, valor, sal_inicial, aux, sal_final
            FIM
        FIM_ALGORITMO.
    
```



#### 1ª SOLUÇÃO - UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP4\PASCAL\EX8\_A.PAS e \EXERC\CAP4\PASCAL\EX8\_A.EXE

#### 2ª SOLUÇÃO - UTILIZANDO A ESTRUTURA REPEAT:

\EXERC\CAP4\PASCAL\EX8\_B.PAS e \EXERC\CAP4\PASCAL\EX8\_B.EXE

**1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP4\C++\EX8\_A.CPP e \EXERC\CAP4\C++\EX8\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA DO-WHILE:**

\EXERC\CAP4\C++\EX8\_B.CPP e \EXERC\CAP4\C++\EX8\_B.EXE

- 9.** Uma empresa contratou 15 funcionários temporários. De acordo com o valor das vendas mensais, os funcionários adquirem pontos que determinarão seus salários ao final de cada mês. Sabe-se que esses funcionários trabalharão nos meses de novembro a janeiro do ano subseqüente.

Faça um programa que:

- leia as pontuações nos três meses de cada funcionário;
- calcule e mostre a pontuação geral de cada funcionário nos três meses;
- calcule e mostre a média das pontuações de cada funcionário nos três meses;
- determine e mostre a maior pontuação atingida entre todos os funcionários nos três meses.

**SOLUÇÃO:**

```

ALGORITMO
DECLARE cont_func, cont_mes, pont NUMÉRICO
pont_total, maior_pont, media_pont NUMÉRICO
maior_pont ← 0
PARA cont_func ← 1 ATÉ 15 FAÇA
    INÍCIO
        pont_total ← 0
        PARA cont_mes ← 1 ATÉ 3 FAÇA
            INÍCIO
                LEIA pont
                pont_total ← pont_total + pont
                SE pont > maior_pont
                    ENTÃO maior_pont ← pont
            FIM
            ESCREVA pont_total
            media_pont ← pont_total/3
            ESCREVA media_pont
            FIM
        ESCREVA maior_pont
    FIM_ALGORITMO.

```

**1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP4\PASCAL\EX9\_A.PAS e \EXERC\CAP4\PASCAL\EX9\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP4\PASCAL\EX9\_B.PAS e \EXERC\CAP4\PASCAL\EX9\_B.EXE

**1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP4\C++\EX9\_A.CPP e \EXERC\CAP4\C++\EX9\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP4\C++\EX9\_B.CPP e \EXERC\CAP4\C++\EX9\_B.EXE

- 10.** Faça um programa que monte os oito primeiros termos da seqüência de Fibonacci.

0-1-1-2-3-5-8-13-21-34-55- ...

**SOLUÇÃO:**

```

ALGORITMO
    DECLARE cont, num1, num2, res NUMÉRICO
    num1 ← 0
    num2 ← 1
    ESCREVA num1
    ESCREVA num2
    PARA cont ← 3 ATÉ 8 FAÇA
        INÍCIO
        res ← num1 + num2
        ESCREVA res
        num1 ← num2
        num2 ← res
        FIM
    FIM_ALGORITMO.

```

**1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP4\PASCAL\EX10\_A.PAS e \EXERC\CAP4\PASCAL\EX10\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA REPEAT:**

\EXERC\CAP4\PASCAL\EX10\_B.PAS e \EXERC\CAP4\PASCAL\EX10\_B.EXE

**1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP4\C++\EX10\_A.CPP e \EXERC\CAP4\C++\EX10\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA DO-WHILE:**

\EXERC\CAP4\C++\EX10\_B.CPP e \EXERC\CAP4\C++\EX10\_B.EXE

- 11.** Faça um programa que leia o número de termos, determine e mostre os valores de acordo com a série abaixo:

Série = 2, 7, 3, 4, 21, 12, 8, 63, 48, 16, 189, 192, 32, 567, 768, 64, ...

**SOLUÇÃO:**

```

ALGORITMO
    DECLARE i, num_termos, num1, num2, num3 NUMÉRICO
    LEIA num_termos
    num1 ← 2
    num2 ← 7
    num3 ← 3
    ESCREVA num1
    ESCREVA num2
    ESCREVA num3
    i ← 4
    enquanto i ≠ num_termos FAÇA

```

```

INÍCIO
num1 ← num1 * 2
ESCREVA num1
i ← i + 1
SE i ≠ num_termos
ENTÃO INÍCIO
    num2 ← num2 * 7
    ESCRIVA num2
    i ← i + 1
    SE i ≠ num_termos
    ENTÃO INÍCIO
        num3 ← num3 * 4
        ESCRIVA num3
        i ← i + 1
    FIM
FIM
FIM
FIM_ALGORITMO.

```



#### **1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP4\PASCAL\EX11\_A.PAS e \EXERC\CAP4\PASCAL\EX11\_A.EXE

#### **2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA REPEAT:**

\EXERC\CAP4\PASCAL\EX11\_B.PAS e \EXERC\CAP4\PASCAL\EX11\_B.EXE



#### **1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP4\C++\EX11\_A.CPP e \EXERC\CAP4\C++\EX11\_A.EXE

#### **2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA DO-WHILE:**

\EXERC\CAP4\C++\EX11\_B.CPP e \EXERC\CAP4\C++\EX11\_B.EXE

- 12.** Faça um programa que receba o valor de X e o número de termos desejados. Calcule e mostre o valor da seguinte seqüência:

$$S = 1 + X^2/3! - X^3/4! + X^4/5! - X^5/6! + X^6/7! - \dots$$



#### **SOLUÇÃO:**

```

ALGORITMO
DECLARE i, j, x, num, s, fat NUMÉRICO
LEIA x, num
s ← 1
PARA i ← 2 ATÉ num FAÇA
INÍCIO
    fat ← 1
    PARA j ← 1 ATÉ i+1 FAÇA
        INÍCIO
            fat ← fat * j
        FIM
    SE RESTO (i/2) = 0
    ENTÃO s ← s + (xi)/fat
    SENÃO s ← s - (xi)/fat
    FIM
    ESCRIVA s
FIM-ALGORTIMO.

```

**1<sup>A</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP4\PASCAL\EX12\_A.PAS e \EXERC\CAP4\PASCAL\EX12\_A.EXE

**2<sup>A</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA REPEAT:**

\EXERC\CAP4\PASCAL\EX12\_B.PAS e \EXERC\CAP4\PASCAL\EX12\_B.EXE

**1<sup>A</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP4\C++\EX12\_A.CPP e \EXERC\CAP4\C++\EX12\_A.EXE

**2<sup>A</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA DO-WHILE:**

\EXERC\CAP4\C++\EX12\_B.CPP e \EXERC\CAP4\C++\EX12\_B.EXE

**13.** Faça um programa que receba duas notas de seis alunos, calcule e mostre:

- ◆ a média aritmética das duas notas de cada aluno;
- ◆ a mensagem que está na tabela a seguir:

<b>MÉDIA ARITMÉTICA</b>	<b>MENSAGEM</b>
Até 3,0	Reprovado
Entre 3,0 e 7,0	Exame
Acima de 7,0	Aprovado

- ◆ o total de alunos aprovados;
- ◆ o total de alunos de exame;
- ◆ o total de alunos reprovados;
- ◆ a média da classe.

**SOLUÇÃO:**

```

ALGORITMO
DECLARE cont, n1, n2, media, ta, te, tr NUMÉRICO
media_classe, total_classe NUMÉRICO
total_classe ← 0
PARA cont ← 1 ATÉ 6 FAÇA
    INÍCIO
    LEIA n1, n2
    media ← (n1 + n2) /2
    ESCREVA media
    SE media <= 3
    ENTÃO INÍCIO
        tr ← tr + 1
        ESCREVA "Reprovado"
        FIM
    SE (media > 3) E (media < 7)
    ENTÃO INÍCIO
        te ← te + 1
        ESCREVA "Exame"
        FIM
    SE (media >= 7)
    ENTÃO INÍCIO
        ta ← ta + 1
        ESCREVA "Aprovado"
        FIM

```

```

        total_classe ← total_classe + media
        FIM
        ESCREVA tr
        ESCREVA te
        ESCREVA ta
        media_classe ← total_classe/6
        ESCREVA media_classe
        FIM_ALGORITMO.
    
```

**1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP4\PASCAL\EX13\_A.PAS e \EXERC\CAP4\PASCAL\EX13\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP4\PASCAL\EX13\_B.PAS e \EXERC\CAP4\PASCAL\EX13\_B.EXE

**1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP4\C++\EX13\_A.CPP e \EXERC\CAP4\C++\EX13\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP4\C++\EX13\_B.CPP e \EXERC\CAP4\C++\EX13\_B.EXE

- 14.** Faça um programa que calcule a soma dos primeiros 50 números pares. Esse programa não recebe valor do teclado. Os primeiros números pares são: 2, 4, 6, ....

**SOLUÇÃO:**

```

ALGORITMO
    DECLARE soma, num, qtde NUMÉRICO
    soma ← 0
    num ← 2
    PARA qtde ← 1 ATÉ 50 FAÇA
        INÍCIO
        soma ← soma + num
        num ← num + 2
        FIM
    ESCREVA soma
    FIM_ALGORITMO.

```

**1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP4\PASCAL\EX14\_A.PAS e \EXERC\CAP4\PASCAL\EX14\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA REPEAT:**

\EXERC\CAP4\PASCAL\EX14\_B.PAS e \EXERC\CAP4\PASCAL\EX14\_B.EXE

**1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP4\C++\EX14\_A.CPP e \EXERC\CAP4\C++\EX14\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA DO-WHILE:**

\EXERC\CAP4\C++\EX14\_B.CPP e \EXERC\CAP4\C++\EX14\_B.EXE

**15.** Em um campeonato de futebol existem cinco times e cada time possui onze jogadores. Faça um programa que receba a idade, o peso e a altura de cada um dos jogadores, calcule e mostre:

- ◆ a quantidade de jogadores com idade inferior a 18 anos;
- ◆ a média das idades dos jogadores de cada time;
- ◆ a média das alturas de todos os jogadores do campeonato;
- ◆ a percentagem de jogadores com mais de 80 quilos entre todos os jogadores do campeonato.

### ALGORITMO

#### SOLUÇÃO:

```

ALGORITMO
DECLARE cont_time, cont_jog, idade NUMÉRICO
peso, alt, qtde, media_idade NUMÉRICO
media_altura, porc, total80 NUMÉRCIO
qtde ← 0
total80 ← 0
PARA cont_time ← 1 ATÉ 5 FAÇA
    INÍCIO
        media_idade ← 0
        PARA cont_jog ← 1 ATÉ 11 FAÇA
            INÍCIO
                leia idade, peso, alt
                SE idade < 18
                    ENTÃO qtde ← qtde + 1
                media_idade ← media_idade + idade
                media_altura ← media_altura + alt
                SE peso > 80
                    ENTÃO tot80 ← tot80 + 1
                FIM
            media_idade ← media_idade/11
            ESCREVA media_idade
            FIM
        ESCREVA qtde
        media_altura ← media_altura/55
        ESCREVA media_altura
        porc ← tot80 * 100 / 55
        ESCREVA porc
    FIM_ALGORITMO.

```



#### 1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP4\PASCAL\EX15\_A.PAS e \EXERC\CAP4\PASCAL\EX15\_A.EXE

#### 2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP4\PASCAL\EX15\_B.PAS e \EXERC\CAP4\PASCAL\EX15\_B.EXE



#### 1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP4\C++\EX15\_A.CPP e \EXERC\CAP4\C++\EX15\_A.EXE

#### 2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP4\C++\EX15\_B.CPP e \EXERC\CAP4\C++\EX15\_B.EXE

**16.** Faça um programa que receba dois números. Calcule e mostre:

- ◆ a soma dos números pares desse intervalo de números, incluindo os números digitados;
- ◆ a multiplicação dos números ímpares desse intervalo de números, incluindo os números digitados.



**SOLUÇÃO:**

```

ALGORITMO
DECLARE num1, num2 , soma, mult, i NUMÉRICO
LEIA num1, num2
soma ← 0
mult ← 1
SE num1 = num2
ENTÃO INÍCIO
    SE RESTO (num1/2) = 0
        ENTÃO soma ← soma + num1
        SENÃO mult ← mult * num1
    FIM
SE num1 < num2
ENTÃO INÍCIO
    PARA i ← num1 ATÉ num2 FAÇA
    INÍCIO
        SE RESTO (i/2) = 0
            ENTÃO soma ← soma + i
            SENÃO mult ← mult * i
        FIM
    FIM
SE num1 > num2
ENTÃO INÍCIO
    PARA i ← num2 ATÉ num1 FAÇA
    INÍCIO
        SE RESTO (i/2) = 0
            ENTÃO soma ← soma + i
            SENÃO mult ← mult * i
        FIM
    FIM
ESCREVA soma
ESCREVA mult
FIM_ALGORITMO.

```



**1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP4\PASCAL\EX16\_A.PAS e \EXERC\CAP4\PASCAL\EX16\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP4\PASCAL\EX16\_B.PAS e \EXERC\CAP4\PASCAL\EX16\_B.EXE



**1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP4\C++\EX16\_A.CPP e \EXERC\CAP4\C++\EX16\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP4\C++\EX16\_B.CPP e \EXERC\CAP4\C++\EX16\_B.EXE

**17.** Faça um programa que receba dois números. Calcule e mostre:

- caso os números formem um intervalo crescente, a média dos números do intervalo, incluindo os números digitados;
- caso os números formem um intervalo decrescente, a quantidade de números pares, incluindo os números digitados;
- se os números forem iguais, mostrar uma mensagem.

### ALGORITMO

#### SOLUÇÃO:

```

ALGORITMO
DECLARE num1, num2 , soma, media, qtde, i, qtde_pares NUMÉRICO
LEIA num1, num2
soma ← 0
qtde ← 0
qtde_pares ← 0
SE num1 = num2
ENTÃO ESCREVA "Números iguais"
SE num1 < num2
ENTÃO INÍCIO
    PARA i ← num1 ATÉ num2 FAÇA
    INÍCIO
        soma ← soma + i
        qtde ← qtde + 1
    FIM
    media ← soma/qtde
    ESCREVA media
    FIM
SE num1 > num2
ENTÃO INÍCIO
    PARA i ← num2 ATÉ num1 FAÇA
    INÍCIO
        SE RESTO (i/2) = 0
            ENTÃO qtde_pares ← qtde_pares + 1
        FIM
        ESCREVA qtde_pares
        FIM
    FIM_ALGORITMO.

```



#### 1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP4\PASCAL\EX17\_A.PAS e \EXERC\CAP4\PASCAL\EX17\_A.EXE

#### 2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA REPEAT:

\EXERC\CAP4\PASCAL\EX17\_B.PAS e \EXERC\CAP4\PASCAL\EX17\_B.EXE



#### 1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP4\C++\EX17\_A.CPP e \EXERC\CAP4\C++\EX17\_A.EXE

#### 2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP4\C++\EX17\_B.CPP e \EXERC\CAP4\C++\EX17\_B.EXE

**18.** Faça um programa que receba um número inteiro maior que 1 e verifique se o número fornecido é primo ou não. Mostrar mensagem de número primo ou de número não primo.

**OBSERVAÇÃO:** Um número é primo quando é divisível *apenas* pelo número um e por ele mesmo.

**ALGORITMO****SOLUÇÃO:**

```

ALGORITMO
    DECLARE i, num, qtde NUMÉRICO
    LEIA num
    qtde ← 0
    PARA i ← 1 ATÉ num FAÇA
        INÍCIO
        SE RESTO(num/i) = 0
        ENTÃO qtde ← qtde + 1
        FIM
    SE qtde > 2
    ENTÃO ESCREVA "Número não primo"
    SENÃO ESCREVA "Número primo"
    FIM_ALGORITMO.

```

**RESOLUÇÃO  
PASCAL**
**1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP4\PASCAL\EX18\_A.PAS e \EXERC\CAP4\PASCAL\EX18\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA REPEAT:**

\EXERC\CAP4\PASCAL\EX18\_B.PAS e \EXERC\CAP4\PASCAL\EX18\_B.EXE

**RESOLUÇÃO  
C/C++**
**1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP4\C++\EX18\_A.CPP e \EXERC\CAP4\C++\EX18\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA DO-WHILE:**

\EXERC\CAP4\C++\EX18\_B.CPP e \EXERC\CAP4\C++\EX18\_B.EXE

**19.** Tem-se um conjunto de dados contendo a altura e o sexo (M ou F) de 15 pessoas. Faça um programa que calcule e mostre:

- a maior e a menor altura do grupo;
- a média de altura das mulheres;
- o número de homens;
- o sexo da pessoa mais alta.

**ALGORITMO****SOLUÇÃO:**

```

ALGORITMO
    DECLARE altura, cont, maior, menor, media NUMÉRICO
    soma, mediaf, somaf, cf, cm NUMÉRICO
        sexo, maior_sexo LITERAL
    soma ← 0
    somaf ← 0
    cf ← 0
    cm ← 0
    PARA cont ← 1 ATÉ 15 FAÇA
        INÍCIO
        LEIA altura, sexo
        SE cont = 1
        ENTÃO INÍCIO
            maior ← altura
            maior_sexo ← sexo
            menor ← altura
        FIM

```

```

SENÃO INÍCIO
    SE altura > maior
    ENTÃO INÍCIO
        maior ← altura
        maior_sexo ← sexo
    FIM
    SE altura < menor
    ENTÃO menor ← altura
    FIM
    SE sexo = "F"
    ENTÃO INÍCIO
        somaf ← somaf + altura
        cf ← cf + 1
    FIM
    SENÃO cm ← cm + 1
    FIM
ESCREVA maior, menor
SE cf = 0
ENTÃO mediaf ← 0
SENÃO mediaf ← somaf/cf
ESCREVA mediaf
ESCREVA cm
ESCREVA maior_sexo
FIM_ALGORITMO.

```



#### 1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP4\PASCAL\EX19\_A.PAS e \EXERC\CAP4\PASCAL\EX19\_A.EXE

#### 2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP4\PASCAL\EX19\_B.PAS e \EXERC\CAP4\PASCAL\EX19\_B.EXE



#### 1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP4\C++\EX19\_A.CPP e \EXERC\CAP4\C++\EX19\_A.EXE

#### 2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP4\C++\EX19\_B.CPP e \EXERC\CAP4\C++\EX19\_B.EXE

- 20.** A conversão de graus Fahrenheit para Celsius é obtida por  $c = 5/9*(f - 32)$ . Faça um programa que calcule e escreva uma tabela de graus Celsius e graus Fahrenheit, cujos graus variem de 50 a 65 de 1 em 1.



#### SOLUÇÃO:

```

ALGORITMO
DECLARE cels, faren NUMÉRICO
PARA faren ← 50 ATÉ 65 FAÇA
    INÍCIO
        ESCREVA faren
        cels ← 5/9 * (faren - 32)
        ESCREVA cels
    FIM
FIM_ALGORITMO.

```



#### 1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP4\PASCAL\EX20\_A.PAS e \EXERC\CAP4\PASCAL\EX20\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA REPEAT:**

\EXERC\CAP4\PASCAL\EX20\_B.PAS e \EXERC\CAP4\PASCAL\EX20\_B.EXE

**1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP4\C++\EX20\_A.CPP e \EXERC\CAP4\C++\EX20\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA DO-WHILE:**

\EXERC\CAP4\C++\EX20\_B.CPP e \EXERC\CAP4\C++\EX20\_B.EXE

**21.** Em uma fábrica trabalham homens e mulheres divididos em três classes:

- ◆ trabalhadores que fazem até 30 peças por mês – classe 1;
- ◆ trabalhadores que fazem de 31 a 35 peças por mês – classe 2;
- ◆ trabalhadores que fazem mais de 35 peças por mês – classe 3.

A classe 1 recebe salário mínimo. A classe 2 recebe salário mínimo mais 3% do salário mínimo por peça, acima das 30 peças iniciais. A classe 3 recebe salário mínimo mais 5% do salário mínimo por peça, acima das 30 peças iniciais.

Faça um programa que receba o número do operário, o número de peças fabricadas no mês, o sexo do operário, e que também calcule e mostre:

- ◆ o número do operário e seu salário;
- ◆ o total da folha de pagamento da fábrica;
- ◆ o número total de peças fabricadas no mês;
- ◆ a média de peças fabricadas pelos homens;
- ◆ a média de peças fabricadas pelas mulheres;
- ◆ o número do operário ou operária de maior salário.

A fábrica possui 15 operários.

**ALGORITMO****SOLUÇÃO:**

```

ALGORITMO
DECLARE num_op, pecas_op, num_maior, cont_m, cont_f NUMÉRICO
tot_pecas, cont, media_m NUMÉRICO
    media_f, salario_op, tot_folha NUMÉRICO
    sexo_op LITERAL
tot_folha ← 0
tot_pecas ← 0
media_m ← 0
media_f ← 0
cont_m ← 0
cont_f ← 0
PARA cont ← 1 ATÉ 15 FAÇA
    INÍCIO
        ESCREVA "Digite o número do ", cont, "º operário "
        LEIA num_op
        ESCREVA "Digite o sexo do operário (M ou F) "
        LEIA sexo_op
        ESCREVA "Digite o total de peças fabricadas pelo ", cont, "º
        operário "
        LEIA pecas_op
        SE pecas_op <= 30
            ENTÃO salario_op ← 150
        SE (pecas_op > 30) E (pecas_op <= 35)

```

```

ENTÃO salario_op ← 150 + ((pecas_op - 30) * 3 / 100 * 150)
SE pecas_op > 35
    ENTÃO salario_op ← 150 + ((pecas_op - 35) * 5 / 100 * 150)
ESCREVA "O operário de número ", num_op, " recebe salário = ",
    ↪ salario_op
tot_folha ← tot_folha + salario_op
tot_pecas ← tot_pecas + pecas_op
SE sexo_op = 'M'
    ENTÃO INÍCIO
        media_m ← media_m + pecas_op
        cont_m ← cont_m + 1
    FIM
SENÃO INÍCIO
    media_f ← media_f + pecas_op
    cont_f ← cont_f + 1
FIM
SE cont = 1
    ENTÃO INÍCIO
        salario_maior ← salario_op
        num_maior ← num_op
    FIM
SENÃO INÍCIO
    SE (salario_op > salario_maior)
        ENTÃO INÍCIO
            salario_maior ← salario_op
            num_maior ← num_op
        FIM
    FIM
ESCREVA "Total da folha de pagamento = ", tot_folha
ESCREVA "Total de peças fabricadas no mês = ", tot_pecas
SE cont_m = 0
    ENTÃO media_m ← 0
SENÃO media_m ← media_m / cont_m
SE cont_f = 0
    ENTÃO media_f ← 0
SENÃO media_f ← media_f / cont_f
ESCREVA "Média de peças fabricadas por mulheres = ", media_f
ESCREVA "Média de peças fabricadas por homens = ", media_m
ESCREVA "O número do operário com maior salário é ", num_maior
FIM_ALGORITMO.

```



#### 1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP4\PASCAL\EX21\_A.PAS e \EXERC\CAP4\PASCAL\EX21\_A.EXE

#### 2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP4\PASCAL\EX21\_B.PAS e \EXERC\CAP4\PASCAL\EX21\_B.EXE



#### 1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP4\C++\EX21\_A.CPP e \EXERC\CAP4\C++\EX21\_A.EXE

#### 2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP4\C++\EX21\_B.CPP e \EXERC\CAP4\C++\EX21\_B.EXE

- 22.** Foi feita uma pesquisa para determinar o índice de mortalidade infantil em um certo período. Faça um programa que:

- ◆ leia o número de crianças nascidas no período;
- ◆ o sexo (M ou F) e o tempo de vida para cada criança nascida.

Calcule e mostre:

- ◆ a percentagem de crianças do sexo feminino mortas no período;
- ◆ a percentagem de crianças do sexo masculino mortas no período;
- ◆ a percentagem de crianças que viveram 24 meses ou menos no período.

### ALGORITMO

#### SOLUÇÃO:

```

ALGORITMO
DECLARE i, num_cri, meses, porc_f, proc_m, tot_f, NUMÉRICO
      tot_m, tot24, porc24 NUMÉRICO
      sexo LITERAL
ESCREVA "Digite o número de crianças nascidas no período "
LEIA num_cri
tot_m ← 0
tot_f ← 0
tot24 ← 0
PARA i=1 ATÉ num_cri FAÇA
    INÍCIO
        ESCREVA "Digite o sexo da ", i, "ª criança"
        LEIA sexo
        ESCREVA "Digite o tempo de vida (em meses) da ", i, "ª
        ↪ criança"
        LEIA meses
        SE sexo = "M"
            ENTÃO tot_m ← tot_m + 1
        SE sexo = "F"
            ENTÃO tot_f ← tot_f + 1
        SE meses <= 24
            ENTÃO tot_24 ← tot_24 + 1
        FIM
        SE num_cri = 0
        ENTÃO INÍCIO
            perc_m ← 0
            perc_f ← 0
            perc_24 ← 0
        FIM
        SENÃO INÍCIO
            perc_m ← tot_m * 100 / num_cri
            perc_f ← tot_f * 100 / num_cri
            perc_24 ← tot_24 * 100 / num_cri
        FIM
        ESCREVA "Percentual de crianças do sexo feminino mortas ", por_f
        ESCREVA "Percentual de crianças do sexo masculino mortas ", por_m
        ESCREVA "Percentual de crianças com 24 meses ou menos mortas no
        ↪ período ", por_24
    FIM_ALGORITMO.

```



#### 1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP4\PASCAL\EX22\_A.PAS e \EXERC\CAP4\PASCAL\EX22\_A.EXE

#### 2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP4\PASCAL\EX22\_B.PAS e \EXERC\CAP4\PASCAL\EX22\_B.EXE



#### 1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP4\C++\EX22\_A.CPP e \EXERC\CAP4\C++\EX22\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP4\C++\EX22\_B.CPP e \EXERC\CAP4\C++\EX22\_B.EXE

- 23.** Faça um programa que receba o valor de uma dívida e mostre uma tabela com os seguintes dados: valor da dívida, valor dos juros, quantidade de parcelas e valor da parcela.

Os juros e a quantidade de parcelas seguem a tabela abaixo:

QUANTIDADE DE PARCELAS	% DE JUROS SOBRE O VALOR INICIAL DA DÍVIDA
1	0
3	10
6	15
9	20
12	25

Exemplo de saída do programa:

VALOR DA DÍVIDA	VALOR DOS JUROS	QUANTIDADE DE PARCELAS	VALOR DA PARCELA
R\$ 1.000,00	0	1	R\$ 1.000,00
R\$ 1.100,00	100	3	R\$ 366,67
R\$ 1.150,00	150	6	R\$ 191,67

**ALGORITMO****SOLUÇÃO:**

```

ALGORITMO
DECLARE valor_inicial, juros, valor_parc NUMÉRICO
total, valor_juros, num_parc, i NUMÉRICO
ESCREVA "Digite o valor inicial da dívida"
LEIA valor_inicial
juros ← 0
num_parc ← 1
total ← valor_inicial
valor_parc ← valor_inicial
ESCREVA total
ESCREVA juros
ESCREVA num_parc
ESCREVA valor_parc
juros ← juros + 10
num_parc ← num_parc + 2
PARA i ← 1 ATÉ 4 FAÇA
    INÍCIO
        valor_juros ← valor_inicial * juros / 100
        total ← valor_inicial + valor_juros
        valor_parc ← total / num_parc
        ESCREVA total
        ESCREVA valor_juros
        ESCREVA num_parc
        ESCREVA valor_parc
        juros ← juros + 5
        num_parc ← num_parc + 3
    FIM
FIM_ALGORITMO.

```

**1<sup>A</sup> SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP4\PASCAL\EX23\_A.PAS e \EXERC\CAP4\PASCAL\EX23\_A.EXE

**2<sup>A</sup> SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP4\PASCAL\EX23\_B.PAS e \EXERC\CAP4\PASCAL\EX23\_B.EXE

**1<sup>A</sup> SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP4\C++\EX23\_A.CPP e \EXERC\CAP4\C++\EX23\_A.EXE

**2<sup>A</sup> SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:**

\EXERC\CAP4\C++\EX23\_B.CPP e \EXERC\CAP4\C++\EX23\_B.EXE

**24.** Faça um programa que receba o preço unitário, a refrigeração (S para os produtos que necessitem de refrigeração e N para os produtos que não necessitem de refrigeração) e a categoria (A – Alimentação, L – Limpeza e V – Vestuário) de 12 produtos. Calcule e mostre:

- ◆ o custo de estocagem calculado de acordo com a tabela a seguir.

<b>PREÇO UNITÁRIO</b>	<b>REFRIGERAÇÃO</b>	<b>CATEGORIA</b>	<b>CUSTO DE ESTOCAGEM</b>
Até R\$ 20,00		A	R\$ 2,00
		L	R\$ 3,00
		V	R\$ 4,00
Entre R\$ 20,00 e R\$ 50,00 (inclusive)	S		R\$ 6,00
	N		R\$ 0,00
		A	R\$ 5,00
	S	L	R\$ 2,00
Maior que R\$ 50,01		V	R\$ 4,00
		A ou V	R\$ 0,00
	N	L	R\$ 1,00

- ◆ o imposto calculado de acordo com as regras a seguir:

Se o produto **não preencher** nenhum dos requisitos abaixo, seu imposto será de 2% sobre o preço unitário; caso contrário, será de 4%.

Os requisitos são: Categoria – A e Refrigeração – S

- ◆ o preço final calculado observando as informações anteriores;
- ◆ a classificação calculada usando a tabela a seguir.

<b>PREÇO FINAL</b>	<b>CLASSIFICAÇÃO</b>
Até R\$ 20,00	Barato
Entre R\$ 20,00 e R\$ 100,00	Normal
Acima de R\$ 100,00	Caro

- ◆ a média dos valores adicionais;

- ◆ o maior preço final;
- ◆ o menor preço final;
- ◆ o total dos impostos;
- ◆ a quantidade de produtos com classificação Barato;
- ◆ a quantidade de produtos com classificação Caro;
- ◆ a quantidade de produtos com classificação Normal.

**ALGORITMO****SOLUÇÃO:**

```

ALGORITMO
DECLARE i, preco, custo_est, imp, preco_final, adicional NUMÉRICO
      maior_p, menor_p, tot_imp, qtd_b, qtd_n, qtd_c NUMÉRICO
      refri, categ LITERAL
adicional ← 0
tot_imp ← 0
qtd_b ← 0
qtd_n ← 0
qtd_c ← 0
PARA i ← 1 ATÉ 12 FAÇA
INÍCIO
    LEIA preco
    LEIA refri
    LEIA categ
    SE (preco <= 20)
        ENTÃO INÍCIO
            SE (categ = "A")
                ENTÃO custo_est ← 2
            SE (categ = "L")
                ENTÃO custo_est ← 3
            SE (categ = "V")
                ENTÃO custo_est ← 4
        FIM
        SE (preco > 20) E (preco <= 50)
            ENTÃO INÍCIO
                SE (refri = "S")
                    ENTÃO custo_est ← 6
                SENÃO custo_est ← 0
            FIM
            SE (preco > 50)
                ENTÃO INÍCIO
                    SE (refri = "S")
                        ENTÃO INÍCIO
                            SE (categ = "A")
                                ENTÃO custo_est ← 5
                            SE (categ = "L")
                                ENTÃO custo_est ← 2
                            SE (categ = "V")
                                ENTÃO custo_est ← 4
                        FIM
                    SENÃO INÍCIO
                        SE (categ = "A") OU (categ = "V")
                            ENTÃO custo_est ← 0
                        SE (categ = "L")
                            ENTÃO custo_est ← 1
                        FIM
                    FIM
                SE (categ ≠ "A") E (refri ≠ "S")
                    ENTÃO imp ← preco * 2 / 100
                    SENÃO imp ← preco * 4 / 100
                preco_final ← preco + custo_est + imp
                ESCREVA preco
            FIM
        FIM
    FIM
FIM

```

```

    ESCREVA custo_est
    ESCREVA imp
    ESCREVA preco_final
    SE (preco_final <= 20)
        ENTÃO INÍCIO
            qtd_b ← qtd_b + 1
            ESCREVA "Classificação Barato"
        FIM
    SE (preco_final > 20) E (preco_final < 100)
        ENTÃO INÍCIO
            qtd_n ← qtd_n + 1
            ESCREVA "Classificação Normal"
        FIM
    SE (preco_final > 100)
        ENTÃO INÍCIO
            qtd_c ← qtd_c + 1
            ESCREVA "Classificação Caro"
        FIM
    adicional ← adicional + custo_est + imp
    tot_imp ← tot_imp ← imp
    SE (i = 1)
        ENTÃO INÍCIO
            Maior_p ← preco_final
            Menor_p ← preco_final
        FIM
    SENÃO INÍCIO
        SE (preco_final > maior_p)
            ENTÃO maior_p ← preco_final
        SE (preco_final < menor_p)
            ENTÃO menor_p ← preco_final
        FIM
    FIM
    adicional ← adicional / 12
    ESCREVA adicional
    ESCREVA maior_p
    ESCREVA menor_p
    ESCREVA tot_imp
    ESCREVA qtd_b
    ESCREVA qtd_n
    ESCREVA qtd_c
    FIM_ALGORITMO.

```



#### 1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP4\PASCAL\EX24\_A.PAS e \EXERC\CAP4\PASCAL\EX24\_A.EXE

#### 2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA REPEAT:

\EXERC\CAP4\PASCAL\EX24\_B.PAS e \EXERC\CAP4\PASCAL\EX24\_B.EXE



#### 1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA FOR:

\EXERC\C++\EX24\_A.CPP e \EXERC\C++\EX24\_A.EXE

#### 2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\C++\EX24\_B.CPP e \EXERC\C++\EX24\_B.EXE

**25.** Faça um programa que leia uma quantidade indeterminada de números positivos e conte quantos deles estão nos seguintes intervalos: [0-25], [26-50], [51-75] e [76-100]. A entrada de dados deverá terminar quando for lido um número negativo.

**ALGORITMO****SOLUÇÃO:**

```

ALGORITMO
DECLARE num, int1, int2, int3, int4 NUMÉRICO
int1 ← 0
int2 ← 0
int3 ← 0
LEIA num
ENQUANTO (num >= 0) FAÇA
INÍCIO
    SE (num >= 0) E (num <= 25)
        ENTÃO int1 ← int1 + 1
    SE (num >= 26) E (num <= 50)
        ENTÃO int2 ← int2 + 1
    SE (num >= 51) E (num <= 75)
        ENTÃO int3 ← int3 + 1
    SE (num >= 76) E (num <= 100)
        ENTÃO int4 ← int4 + 1
    LETA num
FIM
ESCREVA int1
ESCREVA int2
ESCREVA int3
ESCREVA int4
FIM_ALGORITMO.

```

**1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP4\PASCAL\EX25\_A.PAS e \EXERC\CAP4\PASCAL\EX25\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA REPEAT:**

\EXERC\CAP4\PASCAL\EX25\_B.PAS e \EXERC\CAP4\PASCAL\EX25\_B.EXE

**1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP4\C++\EX25\_A.CPP e \EXERC\CAP4\C++\EX25\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA DO-WHILE:**

\EXERC\CAP4\C++\EX25\_B.CPP e \EXERC\CAP4\C++\EX25\_B.EXE

**26.** Faça um programa que determine e mostre os cinco primeiros múltiplos de 3, considerando números > 0.

**ALGORITMO****SOLUÇÃO:**

```

ALGORITMO
DECLARE qtd, num NUMÉRICO
qtd ← 0
num ← 1
ENQUANTO (qtd < 5) FAÇA
INÍCIO
    SE (RESTO(num/3) = 0)
        ENTÃO INÍCIO

```

```

    ESCREVA num
    qtd ← qtd + 1
    FIM
    num ← num + 1
FIM
FIM_ALGORITMO.

```

**1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP4\PASCAL\EX26\_A.PAS e \EXERC\CAP4\PASCAL\EX26\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA REPEAT:**

\EXERC\CAP4\PASCAL\EX26\_B.PAS e \EXERC\CAP4\PASCAL\EX26\_B.EXE

**1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP4\C++\EX26\_A.CPP e \EXERC\CAP4\C++\EX26\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA DO-WHILE:**

\EXERC\CAP4\C++\EX26\_B.CPP e \EXERC\CAP4\C++\EX26\_B.EXE

27. Faça um programa para calcular a área de um triângulo. Esse programa não pode permitir a entrada de dados inválidos, ou seja, medidas menores ou iguais a 0.

**SOLUÇÃO:**

```

ALGORITMO
DECLARE base, altura, area NUMÉRICO
REPITA
    LEIA base
    ATÉ QUE (base > 0)
    REPITA
        LEIA altura
        ATÉ QUE (altura > 0)
        area ← base * altura / 2
        ESCREVA area
    FIM_ALGORITMO

```

**1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP4\PASCAL\EX27\_A.PAS e \EXERC\CAP4\PASCAL\EX27\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA REPEAT:**

\EXERC\CAP4\PASCAL\EX27\_B.PAS e \EXERC\CAP4\PASCAL\EX27\_B.EXE

**1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA DO-WHILE:**

\EXERC\CAP4\C++\EX27\_A.CPP e \EXERC\CAP4\C++\EX27\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP4\C++\EX27\_B.CPP e \EXERC\CAP4\C++\EX27\_B.EXE

28. O cardápio de uma lanchonete é o seguinte:

ESPECIFICAÇÃO	CÓDIGO	PREÇO
Cachorro quente	100	R\$ 1,20
Bauru simples	101	R\$ 1,30
Bauru com ovo	102	R\$ 1,50
Hambúrguer	103	R\$ 1,20
Cheeseburguer	104	R\$ 1,30
Refrigerante	105	R\$ 1,00

Faça um programa que leia o código dos itens pedidos e as quantidades desejadas. Calcule e mostre o valor a ser pago por item (preço \* quantidade) e o total geral do pedido. Considere que o cliente deve informar quando o pedido deve ser encerrado.

#### ALGORITMO

#### SOLUÇÃO:

```

ALGORITMO
DECLARE codigo, qtd, valor_item, valor_total NUMÉRICO
      Resposta LITERAL
valor_total ← 0
REPITA
    LEIA codigo
    LEIA qtd
    SE (codigo = 100)
        ENTÃO INÍCIO
            ESCREVA "Cachorro Quente"
            valor_item ← qtd * 1,20
        FIM
    SE (codigo = 101)
        ENTÃO INÍCIO
            ESCREVA "Bauru Simples"
            valor_item ← qtd * 1,30
        FIM
    SE (codigo = 102)
        ENTÃO INÍCIO
            ESCREVA "Bauru com Ovo"
            valor_item ← qtd * 1,50
        FIM
    SE (codigo = 103)
        ENTÃO INÍCIO
            ESCREVA "Hambúrguer"
            valor_item ← qtd * 1,20
        FIM
    SE (codigo = 104)
        ENTÃO INÍCIO
            ESCREVA "Cheeseburguer"
            valor_item ← qtd * 1,30
        FIM
    SE (codigo = 105)
        ENTÃO INÍCIO
            ESCREVA "Refrigerante"
            valor_item ← qtd * 1,0
        FIM
    ESCREVA valor_item
    valor_total ← valor_total + valor_item
    ESCREVA "Deseja mais alguma coisa (S ou N) ? "
    LEIA resp
ATÉ resp = 'N'
ESCREVA valor_total
FIM_ALGORITMO.

```

**1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA REPEAT:**

\EXERC\CAP4\PASCAL\EX28\_A.PAS e \EXERC\CAP4\PASCAL\EX28\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP4\PASCAL\EX28\_B.PAS e \EXERC\CAP4\PASCAL\EX28\_B.EXE

**1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA DO-WHILE:**

\EXERC\CAP4\C++\EX28\_A.CPP e \EXERC\CAP4\C++\EX28\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP4\C++\EX28\_B.CPP e \EXERC\CAP4\C++\EX28\_B.EXE

- 29.** Faça um programa que receba o salário de um funcionário chamado Carlos. Sabe-se que o funcionário João tem um salário equivalente a um terço do salário de Carlos. Carlos aplicará seu salário integralmente na caderneta de poupança, que está rendendo 2% ao mês e João aplicará seu salário integralmente no fundo de renda fixa, que está rendendo 5% ao mês. Calcule e mostre a quantidade de meses necessários para que o valor pertencente a João iguale ou ultrapasse o valor pertencente a Carlos.

**SOLUÇÃO:**

```

ALGORITMO
DECLARE sal_carlos, sal_joao, meses NUMÉRICO
LEIA sal_carlos
sal_joao ← sal_carlos / 3
meses ← 0
ENQUANTO (sal_joao < sal_carlos) FAÇA
INÍCIO
    sal_carlos ← sal_carlos + (sal_carlos * 2 / 100)
    sal_joao ← sal_joao + (sal_joao * 5 / 100)
    meses ← meses + 1
FIM
ESCREVA meses
FIM_ALGORITMO.

```

**1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP4\PASCAL\EX29\_A.PAS e \EXERC\CAP4\PASCAL\EX29\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA REPEAT:**

\EXERC\CAP4\PASCAL\EX29\_B.PAS e \EXERC\CAP4\PASCAL\EX29\_B.EXE

**1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP4\C++\EX29\_A.CPP e \EXERC\CAP4\C++\EX29\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA DO-WHILE:**

\EXERC\CAP4\C++\EX29\_B.CPP e \EXERC\CAP4\C++\EX29\_B.EXE

-  **30.** Faça um programa que leia um conjunto não determinado de valores, um de cada vez, e escreva uma tabela com cabeçalho, que deve ser repetido a cada 20 linhas. A tabela deverá conter o valor lido, seu quadrado, seu cubo e sua raiz quadrada. Finalizar a entrada de dados com um valor negativo ou zero.


**ALGORITMO**
**SOLUÇÃO:**

```

ALGORITMO
DECLARE linhas, num, quad, cubo, raiz NUMÉRICO
LEIA num
ESCREVA "Valor    Quadrado    Cubo    Raiz"
linhas ← 1
ENQUANTO (num >= 0) FAÇA
INÍCIO
quad ← num * num
cubo ← num * num * num
raiz ← √num
SE (linhas < 20)
ENTÃO INÍCIO
    linhas ← linhas + 1
    ESCREVA (quad, cubo, raiz)
FIM
SENÃO INÍCIO
    LIMPAR A TELA
    linhas ← 1
    ESCREVA "Valor    Quadrado    Cubo    Raiz"
    linhas ← linhas + 1
    ESCREVA quad, cubo, raiz
FIM
LEIA num
FIM
FIM_ALGORITMO.

```


**1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP4\PASCAL\EX30\_A.PAS e \EXERC\CAP4\PASCAL\EX30\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA REPEAT:**

\EXERC\CAP4\PASCAL\EX30\_B.PAS e \EXERC\CAP4\PASCAL\EX30\_B.EXE


**1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP4\C++\EX30\_A.CPP e \EXERC\CAP4\C++\EX30\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA DO-WHILE:**

\EXERC\CAP4\C++\EX30\_B.CPP e \EXERC\CAP4\C++\EX30\_B.EXE

-  **31.** Faça um programa que leia um número não determinado de pares de valores [m,n], todos inteiros e positivos, um par de cada vez e que calcule e mostre a soma de todos os números inteiros entre m e n (inclusive). A digitação de pares termina quando m for maior ou igual a n.


**ALGORITMO**
**SOLUÇÃO:**

ALGORITMO

```

DECLARE m, n, soma, i NUMÉRICO
LEIA m
LEIA n
ENQUANTO (m < n) FAÇA
INÍCIO
    soma ← 0
    PARA i = m ATÉ n FAÇA
    INÍCIO
        soma ← soma + i
    FIM
    ESCREVA soma
    LEIA m
    LEIA n
FIM
FIM_ALGORITMO.

```

**1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP4\PASCAL\EX31\_A.PAS e \EXERC\CAP4\PASCAL\EX31\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA REPEAT:**

\EXERC\CAP4\PASCAL\EX31\_B.PAS e \EXERC\CAP4\PASCAL\EX31\_B.EXE

**1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP4\C++\EX31\_A.CPP e \EXERC\CAP4\C++\EX31\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA DO-WHILE:**

\EXERC\CAP4\C++\EX31\_B.CPP e \EXERC\CAP4\C++\EX31\_B.EXE

- 32.** Faça um programa que leia dois valores inteiros e positivos, X e Y, e que calcule e mostre a potência  $X^Y$ , utilizando uma estrutura de repetição.

**SOLUÇÃO:**

```

ALGORITMO
DECLARE x, y, pot, cont NUMÉRICO
LEIA x
LEIA y
pot ← 1
PARA cont ← 1 ATÉ y FAÇA
INÍCIO
    pot ← pot * x
FIM
ESCREVA pot
FIM_ALGORITMO.

```

**1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP4\PASCAL\EX32\_A.PAS e \EXERC\CAP4\PASCAL\EX32\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA REPEAT:**

\EXERC\CAP4\PASCAL\EX32\_B.PAS e \EXERC\CAP4\PASCAL\EX32\_B.EXE

**1<sup>A</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP4\C++\EX32\_A.CPP e \EXERC\CAP4\C++\EX32\_A.EXE

**2<sup>A</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA DO WHILE:**

\EXERC\CAP4\C++\EX32\_B.CPP e \EXERC\CAP4\C++\EX32\_B.EXE

**33.** Faça um programa para ler o código, o sexo (M – Masculino, F – Feminino) e o número de horas/aula dadas mensalmente pelos professores de uma universidade, sabendo-se que cada hora/aula vale R\$ 18,50. Emite uma listagem contendo o código, o salário bruto e o salário líquido (levando em consideração os descontos explicados a seguir) de todos os professores lidos. Mostre também a média dos salários líquidos dos professores do sexo masculino e a média dos salários brutos dos professores do sexo feminino. Considere:

- ◆ desconto para homens 10% e para mulheres 5%;
- ◆ as informações terminarão quando for lido o código = 99999.

**SOLUÇÃO:**

```

ALGORITMO
DECLARE cod, num_h, sal_b, sal_l, media_m, media_f NUMÉRICO
        cont_m, cont_f NUMÉRICO
        Sexo LITERAL
LEIA cod
cont_m ← 0
cont_f ← 0
ENQUANTO (cod ≠ 99999) FAÇA
INÍCIO
    LEIA sexo
    LEIA num_h
    sal_b ← num_h * 18,50
    SE (sexo = "M")
        ENTÃO INÍCIO
            sal_l ← sal_b - (sal_b * 10 / 100)
            media_m ← media_m + sal_l
            cont_m ← cont_m + 1
        FIM
    SE (sexo = "F")
        ENTÃO INÍCIO
            sal_l ← sal_b - (sal_b * 5 / 100)
            media_f ← media_f + sal_l
            cont_f ← cont_f + 1
        FIM
    ESCREVA cod
    ESCREVA sal_b
    ESCREVA sal_l
    LEIA cod
FIM
media_m ← media_m / cont_m
media_f ← media_f / cont_f
ESCREVA media_m
ESCREVA media_f
FIM_ALGORITMO.

```

**1<sup>A</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP4\PASCAL\EX33\_A.PAS e \EXERC\CAP4\PASCAL\EX33\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA REPEAT:**

\EXERC\CAP4\PASCAL\EX33\_B.PAS e \EXERC\CAP4\PASCAL\EX33\_B.EXE

**1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP4\C++\EX33\_A.CPP e \EXERC\CAP4\C++\EX33\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA DO-WHILE:**

\EXERC\CAP4\C++\EX33\_B.CPP e \EXERC\CAP4\C++\EX33\_B.EXE

- 34.** Faça um programa que leia um número indeterminado de valores para m, todos inteiros e positivos, um de cada vez. Se m for par, verifique quantos divisores possui. Se m for ímpar, calcule a soma dos números inteiros de 1 até m (m não deve entrar nos cálculos). Mostre os cálculos realizados. Finalize a entrada de dados com m zero ou negativo.

**SOLUÇÃO:**

```

ALGORITMO
DECLARE m, calc, i NUMÉRICO
LEIA m
ENQUANTO (m > 0) FAÇA
INÍCIO
    calc ← 0
    SE (RESTO(m/2) = 0)
        ENTÃO INÍCIO
            PARA i ← 1 ATÉ m FAÇA
            INÍCIO
                SE (RESTO(m/i) = 0)
                    ENTÃO calc ← calc + 1
                FIM
                ESCREVA calc
            FIM
        SENÃO INÍCIO
            PARA i ← 1 ATÉ m-1 FAÇA
            INÍCIO
                calc ← calc + i
            FIM
            ESCREVA calc
        FIM
    LEIA m
FIM
FIM_ALGORITMO.

```

**1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP4\PASCAL\EX34\_A.PAS e \EXERC\CAP4\PASCAL\EX34\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA REPEAT:**

\EXERC\CAP4\PASCAL\EX34\_B.PAS e \EXERC\CAP4\PASCAL\EX34\_B.EXE

**1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP4\PASCAL\EX34\_A.CPP e \EXERC\CAP4\PASCAL\EX34\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA DO-WHILE:**

\EXERC\CAP4\C++\EX34\_B.CPP e \EXERC\CAP4\C++\EX34\_B.EXE

- 35.** Faça um programa que receba vários números, calcule e mostre:

- ◆ a soma dos números digitados;
- ◆ a quantidade de números digitados;
- ◆ a média dos números digitados;
- ◆ o maior número digitado;
- ◆ o menor número digitado;
- ◆ a média dos números pares;
- ◆ a percentagem dos números ímpares entre todos os números digitados.

Finalize a entrada de dados com a digitação do número 30000.

**ALGORITMO****SOLUÇÃO:**

```

ALGORITMO
DECLARE num, soma, qtd, maior, menor, qtd_par NUMÉRICO
            soma_par, qtd_impar, media, perc NUMÉRICO
qtd ← 0
qtd_par ← 0
soma_par ← 0
qtd_impar ← 0
LEIA num
ENQUANTO (num <= 30000) FAÇA
INÍCIO
SE (qtd = 0)
    ENTÃO INÍCIO
        maior ← num
        menor ← num
    FIM
SENÃO INÍCIO
    SE (num > maior)
        ENTÃO maior ← num
    SE (num < menor)
        ENTÃO menor ← num
    FIM
soma ← soma + num
qtd ← qtd + 1
SE (RESTO(num/2) = 0)
    ENTÃO INÍCIO
        soma_par ← soma_par + num
        qtd_par ← qtd_par + 1
    FIM
SENÃO qtd_impar ← qtd_impar + 1
LEIA num
FIM
ESCREVA soma
ESCREVA qtd
media ← soma / qtd
ESCREVA media
ESCREVA maior
ESCREVA menor
media ← soma_par / qtd_par
ESCREVA media
perc ← qtd_impar * 100 / qtd
ESCREVA perc
FIM_ALGORITMO.

```

**1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP4\PASCAL\EX35\_A.PAS e \EXERC\CAP4\PASCAL\EX35\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA REPEAT:**

\EXERC\CAP4\PASCAL\EX35\_B.PAS e \EXERC\CAP4\PASCAL\EX35\_B.EXE

**1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP4\C++\EX35\_A.CPP e \EXERC\CAP4\C++\EX35\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA DO-WHILE:**

\EXERC\CAP4\C++\EX35\_B.CPP e \EXERC\CAP4\C++\EX35\_B.EXE

**36.** Faça um programa que:

- ◆ leia um número indeterminado de linhas contendo cada uma a idade de um indivíduo. A última linha, que não entrará nos cálculos, contém o valor da idade igual a zero;
- ◆ calcule e mostre a idade média desse grupo de indivíduos.

**SOLUÇÃO:**

```

ALGORITMO
DECLARE idade, soma, qtd, media NUMÉRICO
soma ← 0
qtd ← 0
LEIA idade
ENQUANTO (idade > 0) FAÇA
INÍCIO
    soma ← soma + idade
    qtd ← qtd + 1
    LEIA idade
FIM
media ← soma / qtd
ESCREVA soma
FIM_ALGORITMO.

```

**1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP4\PASCAL\EX36\_A.PAS e \EXERC\CAP4\PASCAL\EX36\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA REPEAT:**

\EXERC\CAP4\PASCAL\EX36\_B.PAS e \EXERC\CAP4\PASCAL\EX36\_B.EXE

**1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP4\C++\EX36\_A.CPP e \EXERC\CAP4\C++\EX36\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA DO-WHILE:**

\EXERC\CAP4\C++\EX36\_B.CPP e \EXERC\CAP4\C++\EX36\_B.EXE

**37.** Uma empresa decidiu fazer um levantamento em relação aos candidatos que se apresentarem para preenchimento de vagas no seu quadro de funcionários. Supondo que você seja o programador dessa empresa, faça um programa que:

- leia, para cada candidato, a idade, o sexo (M ou F) e a experiência no serviço (S ou N). Para encerrar a entrada de dados digite zero para a idade.

Calcule e mostre:

- ◆ o número de candidatos do sexo feminino;
  - ◆ o número de candidatos do sexo masculino;
  - ◆ a idade média dos homens que já têm experiência no serviço;
  - ◆ a percentagem dos homens com mais de 45 anos entre o total dos homens;
  - ◆ o número de mulheres com idade inferior a 35 anos e com experiência no serviço;
  - ◆ a menor idade entre as mulheres que já têm experiência no serviço.

## ALGORITMO

### SOLUÇÃO:

```

ALGORITMO
DECLARE idade, tot_f, tot_m, somal, cont_m1, cont_m2, cont_m3,
        cont_m4, cont_f1, media_idade, calc NUMÉRICO
        sexo, exp LITERAL
tot_f ← 0
tot_m ← 0
somal ← 0
cont_m1 ← 0
cont_m2 ← 0
cont_f1 ← 0
LEIA idade
ENQUANTO (idade ≠ 0) FAÇA
INÍCIO
    LEIA sexo
    LEIA exp
    SE (sexo = "F") E (exp = "S")
        ENTÃO INÍCIO
            SE (tot_f = 0)
                ENTÃO menor_idade ← idade
                SENÃO SE (idade < menor_idade)
                    ENTÃO Menor_idade ← idade
            FIM
            SE (sexo = "M")
                ENTÃO tot_m ← tot_m + 1
            SE (sexo = "F")
                ENTÃO tot_f ← tot_f + 1
            SE (sexo = "F") E (idade < 35) E (exp = "S")
                ENTÃO cont_f1 ← cont_f1 + 1
            SE (sexo = "M") e (idade > 45)
                ENTÃO cont_m1 ← cont_m1 + 1
            SE (sexo = "M") E (exp = "S")
                ENTÃO INÍCIO
                    somal ← somal + idade
                    cont_m2 ← cont_m2 + 1
                FIM
            LEIA idade
        FIM
    ESCREVA tot_f
    ESCREVA tot_m
    calc ← somal / cont_m2
    ESCREVA calc
    Calc ← cont1 * 100 / tot_m
    ESCREVA calc

```

```
ESCREVA cont_f1
ESCREVA menor_idade
FIM_ALGORITMO.
```

**1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP4\PASCAL\EX37\_A.PAS e \EXERC\CAP4\PASCAL\EX37\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA REPEAT:**

\EXERC\CAP4\PASCAL\EX37\_B.PAS e \EXERC\CAP4\PASCAL\EX37\_B.EXE

**1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP4\C++\EX37\_A.CPP e \EXERC\CAP4\C++\EX37\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA DO-WHILE:**

\EXERC\CAP4\C++\EX37\_B.CPP e \EXERC\CAP4\C++\EX37\_B.EXE

- 38.** Faça um programa que receba o valor do salário mínimo e uma lista contendo a quantidade de quilowatts gasta por consumidor e o tipo de consumidor (1 – Residencial, 2 – Comercial ou 3 – Industrial).

Calcule e mostre:

- ◆ o valor de cada quilowatt, sabendo que o quilowatt custa  $\frac{1}{8}$  do salário mínimo;
- ◆ o valor a ser pago por cada consumidor (conta final mais acréscimo), considerando que o acréscimo é o mesmo da tabela a seguir.

TIPO	% DE ACRÉSCIMO SOBRE O VALOR GASTO
1	5
2	10
3	15

- ◆ o faturamento geral da empresa;
- ◆ a quantidade de consumidores que pagam entre R\$ 500,00 e R\$ 1.000,00.

Termine a entrada de dados com quantidade de quilowats igual a zero.

**SOLUÇÃO:**

```
ALGORITMO
DECLARE sal, qtd, tipo, valor_kw, gasto, acresc NUMÉRICO
          total, tot_geral, qtd_cons NUMÉRICO
tot_geral ← 0
qtd_cons ← 0
LEIA (sal)
valor_kw ← sal / 8
ENQUANTO (qtd ≠ 0) FAÇA
INÍCIO
    gasto ← qtd * valor_kw
    LEIA tipo
    SE (tipo = 1)
        ENTÃO acresc ← gasto * 5 / 100
    SE (tipo = 2)
        ENTÃO acresc ← gasto * 10 / 100
```

```

SE (tipo = 3)
    ENTÃO acresc ← gasto * 15 / 100
    total ← gasto + acresc
    tot_geral ← tot_geral + total
    SE (total >= 500) E (total <= 1000)
        ENTÃO qtd_cons ← qtd_cons + 1
    ESCREVA gasto
    ESCREVA acresc
    ESCREVA total
    LEIA qtd
FIM
ESCREVA tot_geral
ESCREVA qtd_cons
FIM_ALGORITMO.

```



#### 1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP4\PASCAL\EX38\_A.PAS e \EXERC\CAP4\PASCAL\EX38\_A.EXE

#### 2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA REPEAT:

\EXERC\CAP4\PASCAL\EX38\_B.PAS e \EXERC\CAP4\PASCAL\EX38\_B.EXE



#### 1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP4\C++\EX38\_A.CPP e \EXERC\CAP4\C++\EX38\_A.EXE

#### 2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP4\C++\EX38\_B.CPP e \EXERC\CAP4\C++\EX38\_B.EXE

- 39.** Faça um programa que apresente o menu de opções a seguir, permita ao usuário escolher a opção desejada, receba os dados necessários para executar a operação e mostre o resultado. *Verificar a possibilidade de opção inválida e não se preocupar com restrições do tipo salário inválido.*

Menu de opções:

1. Imposto
2. Novo salário
3. Classificação
4. Finalizar o programa

Digite a opção desejada

**Na opção 1:** receber o salário de um funcionário, calcular e mostrar o valor do imposto usando as regras a seguir.

SALÁRIOS	% DO IMPOSTO
Menor que R\$ 500,00	5
De R\$ 500,00 a R\$ 850,00	10
Acima de R\$ 850,00	15

**Na opção 2:** receber o salário de um funcionário, calcular e mostrar o valor do novo salário usando as regras a seguir.

SALÁRIOS	AUMENTO
Maiores que R\$ 1.500,00	R\$ 25,00
De R\$ 750,00 (inclusive) a R\$ 1.500,00 (inclusive)	R\$ 50,00
De R\$ 450,00 (inclusive) a R\$ 750,00	R\$ 75,00
Menores que R\$ 450,00	R\$ 100,00

**Na opção 3:** receber o salário de um funcionário e mostrar sua classificação usando a tabela a seguir.

SALÁRIOS	CLASSIFICAÇÃO
Até R\$ 700,00 (inclusive)	Mal remunerado
Maiores que R\$ 700,00	Bem remunerado

### ALGORITMO

#### SOLUÇÃO:

```

ALGORITMO
DECLARE op, sal, imp, aum, novo_sal NUMÉRICO
REPITA
    ESCREVA "1- Imposto"
    ESCREVA "2- Novo Salário"
    ESCREVA "3- Classificação"
    ESCREVA "4- Finalizar o programa"
    ESCREVA "Digite a opção desejada"
    LEIA op
    SE ((op > 4) OU (op < 1))
        ENTÃO ESCREVA "Opção inválida !"
    SENÃO SE (op = 1)
        ENTÃO INÍCIO
            LEIA sal
            SE (sal < 500)
                ENTÃO imp ← sal * 5 / 100
            SE (sal >= 500) E (sal <= 850)
                ENTÃO imp ← sal * 10 / 100
            SE (sal > 850)
                ENTÃO imp ← sal * 15 / 100
            ESCREVA imp
        FIM
    SE (op = 2)
        ENTÃO INÍCIO
            LEIA sal
            SE (sal > 1500)
                ENTÃO aum ← 25
            SE (sal <= 750) E (sal <= 1500)
                ENTÃO aum ← 50
            SE (sal >= 450) E (sal < 750)
                ENTÃO aum ← 75
            SE (sal < 450)
                ENTÃO aum ← 100
            novo_sal ← sal + aum
            ESCREVA novo_sal
        FIM
    SE (op = 3)
        ENTÃO INÍCIO
            LEIA sal
            SE (sal <= 700)
                ENTÃO ESCREVA "Mal Remunerado"
            SENÃO ESCREVA "Bem Remunerado"
        FIM
    ATÉ op = 4
FIM_ALGORITMO.

```


**1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA REPEAT:**

\EXERC\CAP4\PASCAL\EX39\_A.PAS e \EXERC\CAP4\PASCAL\EX39\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP4\PASCAL\EX39\_B.PAS e \EXERC\CAP4\PASCAL\EX39\_B.EXE


**1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA DO-WHILE:**

\EXERC\CAP4\C++\EX39\_A.CPP e \EXERC\CAP4\C++\EX39\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP4\C++\EX39\_B.CPP e \EXERC\CAP4\C++\EX39\_B.EXE

**40.** Faça um programa que receba os dados a seguir de vários produtos: preço unitário, país de origem (1 – EUA, 2 – México e 3 – outros), meio de transporte (T – Terrestre, F – Fluvial e A – Aéreo), carga perigosa (S – Sim, N – Não). Calcule e mostre:

- ◆ o valor do imposto calculado usando a tabela a seguir.

	<b>PREÇO UNITÁRIO</b>	<b>PERCENTUAL DE IMPOSTO SOBRE O PREÇO UNITÁRIO</b>
	Até R\$ 100,00	5%
	Maior que R\$ 100,00	10%

- ◆ o valor de transporte calculado usando a tabela a seguir.

<b>CARGA PERIGOSA</b>	<b>PAÍS DE ORIGEM</b>	<b>VALOR DO TRANSPORTE</b>
	1	R\$ 50,00
S	2	R\$ 35,00
	3	R\$ 24,00
	1	R\$ 12,00
N	2	R\$ 35,00
	3	R\$ 60,00

- ◆ o valor do seguro, calculado usando a regra a seguir.

Os produtos que vêm do México e os produtos que utilizam transporte aéreo pagam metade do valor do seu preço unitário como seguro.

- ◆ o preço final;
- ◆ o total dos impostos.


**SOLUÇÃO:**

```

ALGORITMO
DECLARE preco, imp, transp, segura, final NUMÉRICO
      total_imp, origem NUMÉRICO
      Meio_t, carga LITERAL
  
```

```

LEIA preco
ENQUANTO (preco > 0) FAÇA
INÍCIO
    LEIA origem
    LEIA meio_t
    LEIA carga
    SE (preco <= 100)
        ENTÃO imp ← preco * 5 / 100
        SENÃO imp ← preco * 10 / 100
    SE (carga = "S")
        ENTÃO INÍCIO
            SE (origem = 1)
                ENTÃO transp ← 50
            SE (origem = 2)
                ENTÃO transp ← 35
            SE (origem = 3)
                ENTÃO transp ← 24
        FIM
    SE (carga = "N")
        ENTÃO INÍCIO
            SE (origem = 1)
                ENTÃO transp ← 12
            SE (origem = 2)
                ENTÃO transp ← 35
            SE (origem = 3)
                ENTÃO transp ← 60
        FIM
    SE (origem = 2) OU (meio_t = "A")
        ENTÃO seguro ← preco/2
        SENÃO seguro ← 0
        final ← preco +imp + transp + seguro
        total_imp ← total_imp + imp
    ESCREVA imp
    ESCREVA transp
    ESCREVA seguro
    ESCREVA final
    LEIA preco
FIM
ESCREVA total_imp
FIM_ALGORITMO.

```



#### 1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP4\PASCAL\EX40\_A.PAS e \EXERC\CAP4\PASCAL\EX40\_A.EXE

#### 2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA REPEAT:

\EXERC\CAP4\PASCAL\EX40\_B.PAS e \EXERC\CAP4\PASCAL\EX40\_B.EXE



#### 1<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP4\C++\EX40\_A.CPP e \EXERC\CAP4\C++\EX40\_A.EXE

#### 2<sup>a</sup> SOLUÇÃO - UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP4\C++\EX40\_B.CPP e \EXERC\CAP4\C++\EX40\_B.EXE

## EXERCÍCIOS PROPOSTOS

**1.** Faça um programa que verifique e mostre os números entre 1.000 e 2.000 (inclusive) que, quando divididos por 11, produzam resto igual a 5.

**2.** Faça um programa que leia um valor  $n$ , inteiro e positivo, calcule e mostre a seguinte soma:

$$S = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \dots + \frac{1}{n}$$

**3.** Faça um programa que calcule e mostre o produto dos números primos entre 92 e 1.478.

**4.** Faça um programa que leia cinco grupos de quatro valores ( $A, B, C, D$ ) e mostre-os na ordem lida. Em seguida, mostre-os em ordem crescente e decrescente.

**5.** Uma loja tem 15 clientes cadastrados e deseja enviar uma correspondência a cada um deles anunciando um bônus especial. Faça um programa que leia o nome do cliente e o valor de suas compras no ano passado. Calcule e mostre um bônus de 10% se o valor das compras for menor que R\$ 1.000,00 e de 15%, caso contrário.

**6.** Uma companhia de teatro deseja dar uma série de espetáculos. A direção calcula que a R\$ 5,00 o ingresso, serão vendidos 120 ingressos, e que as despesas serão R\$ 200,00. Diminuindo-se R\$ 0,50 o preço dos ingressos espera-se que as vendas aumentem em 26 ingressos.

Faça um programa que escreva uma tabela de valores de lucros esperados em função do preço do ingresso, fazendo-se variar esse preço de R\$ 5,00 a R\$ 1,00 de R\$ 0,50 em R\$ 0,50. Escreva, ainda, o lucro máximo esperado, o preço do ingresso e a quantidade de ingressos vendidos para a obtenção desse lucro.

**7.** Faça um programa que receba a idade de dez pessoas e que calcule e mostre a quantidade de pessoas com idade maior ou igual a 18 anos.

**8.** Faça um programa que receba a idade de 15 pessoas e que calcule e mostre:

- ◆ a quantidade de pessoas em cada faixa etária;
- ◆ a percentagem de pessoas na primeira e na última faixa etária, com relação ao total de pessoas.

<b>FAIXA ETÁRIA</b>	<b>IDADE</b>
1 <sup>a</sup>	Até 15 anos
2 <sup>a</sup>	De 16 a 30 anos
3 <sup>a</sup>	De 31 a 45 anos
4 <sup>a</sup>	De 46 a 60 anos
5 <sup>a</sup>	Acima de 61 anos

**9.** Faça um programa que receba um número e que calcule e mostre a tabuada desse número.

**10.** Faça um programa que mostre as tabuadas dos números de 1 a 10.

**11.** Uma loja utiliza o código V para transação à vista e P para transação a prazo. Faça um programa que receba o código e o valor de 15 transações. Calcule e mostre:

- ◆ o valor total das compras à vista;
- ◆ o valor total das compras a prazo;
- ◆ o valor total das compras efetuadas;
- ◆ o valor da primeira prestação das compras a prazo, sabendo-se que essas serão pagas em três vezes.

**12.** Faça um programa que receba a idade, a altura e o peso de 25 pessoas. Calcule e mostre:

- ◆ a quantidade de pessoas com idade superior a 50 anos;
- ◆ a média das alturas das pessoas com idade entre 10 e 20 anos;
- ◆ a percentagem de pessoas com peso inferior a 40 quilos entre todas as pessoas analisadas.

**13.** Faça um programa que receba a idade e o peso de sete pessoas. Calcule e mostre:

- ◆ a quantidade de pessoas com mais de 90 quilos;
- ◆ a média das idades das sete pessoas.

**14.** Faça um programa que receba a idade, o peso, a altura, a cor dos olhos (A – Azul, P – Preto, V – Verde e C – Castanho) e a cor dos cabelos (P – Preto, C – Castanho, L – Louro e R – Ruivo) de 20 pessoas e que calcule e mostre:

- ◆ a quantidade de pessoas com idade superior a 50 anos e peso inferior a 60 quilos;
- ◆ a média das idades das pessoas com altura inferior a 1,50;
- ◆ a percentagem de pessoas com olhos azuis entre todas as pessoas analisadas;
- ◆ a quantidade de pessoas ruivas e que não possuem olhos azuis.

**15.** Faça um programa que receba dez números e que calcule e mostre a quantidade de números entre 30 e 90.

**16.** Faça um programa que receba dez idades, pesos e alturas e que calcule e mostre:

- ◆ a média das idades das dez pessoas;
- ◆ a quantidade de pessoas com peso superior a 90 quilos e altura inferior a 1,50;
- ◆ a percentagem de pessoas com idade entre 10 e 30 anos entre as pessoas que medem mais de 1,90.

**17.** Faça um programa que receba a idade e o sexo de sete pessoas e que calcule e mostre:

- ◆ a idade média do grupo;
- ◆ a idade média das mulheres;
- ◆ a idade média dos homens.

**18.** Faça um programa que receba dez números, calcule e mostre a soma dos números pares e a soma dos números primos.

**19.** Faça um programa que receba o valor de um carro e mostre uma tabela com os seguintes dados: preço final, quantidade de parcelas e valor da parcela. Considere o seguinte:

1. O preço final para compra à vista tem um desconto de 20%.
2. A quantidade de parcelas pode ser: 6, 12, 18, 24, 30, 36, 42, 48, 54 e 60.
3. Os percentuais de acréscimo seguem a tabela a seguir.

QUANTIDADE DE PARCELAS	PERCENTUAL DE ACRÉSCIMO SOBRE O PREÇO FINAL
6	3%
12	6%
18	9%
24	12%
30	15%
36	18%
42	21%
48	24%
54	27%
60	30%

**20.** Faça um programa que receba dez números inteiros e mostre a quantidade de números primos dentre os números que foram digitados.

**21.** Faça um programa para calcular  $n!$  (fatorial de  $n$ ), sendo que o valor inteiro de  $n$  é fornecido pelo usuário.

Sabe-se que:

$$\begin{aligned} n! &= 1 * 2 * 3 * \dots * (n-1) * n; \\ 0! &= 1, \text{ por definição.} \end{aligned}$$

**22.** Faça um programa que receba a idade e o peso de 15 pessoas. Calcule e mostre as médias dos pesos das pessoas da mesma faixa etária. As faixas etárias são: de 1 a 10 anos, de 11 a 20 anos, de 21 a 30 anos e maiores de 31 anos.

**23.** Cada espectador de um cinema respondeu a um questionário no qual constava sua idade e a sua opinião em relação ao filme: ótimo – 3, bom – 2, regular – 1. Faça um programa que receba a idade e a opinião de 15 espectadores e que calcule e mostre:

- ◆ a média das idades das pessoas que responderam ótimo;
- ◆ a quantidade de pessoas que respondeu regular;
- ◆ a percentagem de pessoas que respondeu bom entre todos os espectadores analisados.

**24.** Uma firma fez uma pesquisa de mercado para saber se as pessoas gostaram ou não de um novo produto lançado no mercado. Para isso forneceu o sexo do entrevistado e sua resposta (S – Sim ou N – Não). Sabe-se que foram entrevistadas dez pessoas. Faça um programa que calcule e mostre:

- ◆ o número de pessoas que respondeu sim;
- ◆ o número de pessoas que respondeu não;
- ◆ o número de mulheres que respondeu sim;
- ◆ a percentagem de homens que respondeu não entre todos os homens analisados.

**25.** O sistema de avaliação de uma determinada disciplina obedece aos seguintes critérios:

- ◆ durante o semestre são dadas três notas;
- ◆ a nota final é obtida pela média aritmética das três notas;
- ◆ é considerado aprovado o aluno que obtiver a nota final superior ou igual a 6 e que tiver comparecido a um mínimo de 40 aulas.

Faça um programa que:

- ◆ leia um conjunto de dados contendo o número da matrícula, as três notas e a freqüência (número de aulas freqüentadas) de dez alunos.

Calcule e mostre:

- ◆ para cada aluno o número da matrícula, a nota final e a mensagem (aprovado ou reprovado);
- ◆ a maior e a menor nota da turma;
- ◆ o total de alunos reprovados;
- ◆ a percentagem de alunos reprovados por freqüência abaixo da mínima necessária.

**26.** Faça um programa que receba várias idades e que calcule e mostre a média das idades digitadas. Finalize digitando idade igual a zero.

**27.** Foi feita uma pesquisa de audiência de canal de TV em várias casas de uma cidade, em um determinado dia. Para cada casa consultada foi fornecido o número do canal (4, 5, 7, 12) e o número de pessoas que estavam assistindo àquele canal. Se a televisão estivesse desligada, nada era anotado, ou seja, essa casa não entrava na pesquisa. Faça um programa que:

- a) leia um número indeterminado de dados (número do canal e o número de pessoas que estavam assistindo);
- b) calcule e mostre a percentagem de audiência de cada canal.

Para encerrar a entrada de dados digite o número do canal ZERO.

**28.** A prefeitura de uma cidade fez uma pesquisa entre seus habitantes, coletando dados sobre o salário e o número de filhos. A prefeitura deseja saber:

- a) a média do salário da população;
- b) a média do número de filhos;
- c) o maior salário;
- d) a percentagem de pessoas com salários até R\$ 150,00.

O final da leitura de dados dar-se-á com a entrada de um salário negativo.

**29.** Foi feita uma pesquisa entre os habitantes de uma região. Foram coletados os dados de idade, sexo (M/F) e salário. Faça um programa que calcule e mostre:

- a) a média dos salários do grupo;
- b) a maior e a menor idade do grupo;
- c) a quantidade de mulheres com salário até R\$ 200,00;
- d) a idade e o sexo da pessoa que possui o menor salário.

Finalize a entrada de dados ao ser digitada uma idade negativa.

**30.** Uma empresa deseja aumentar seus preços em 20%. Faça um programa que leia o código e o preço de custo de cada produto e que calcule o novo preço. Calcule também a média dos preços com e sem aumento. Mostre o código e o novo preço de cada produto e, no final, as médias. A entrada de dados deve terminar quando for lido um código de produto negativo.

**31.** Faça um programa que receba o tipo da ação, ou seja, uma letra a ser comercializada na bolsa de valores, o preço de compra e o preço de venda de cada ação e que calcule e mostre:

- ◆ o lucro de cada ação comercializada;
- ◆ a quantidade de ações com lucro superior a R\$ 1.000,00;

- ◆ a quantidade de ações com lucro inferior a R\$ 200,00;
- ◆ o lucro total da empresa.

Finalize com o tipo de ação ‘F’.

**32.** Faça um programa que receba vários números e que calcule e mostre:

- ◆ a quantidade de números inferiores a 35;
- ◆ a média dos números positivos;
- ◆ a percentagem de números entre 50 e 100 entre todos os números digitados;
- ◆ a percentagem de números entre 10 e 20 entre os números menores que 50.

**33.** Faça um programa que apresente o menu de opções a seguir:

Menu de opções:

1. Média aritmética
2. Média ponderada
3. Sair

Digite a opção desejada

**Na opção 1:** receber duas notas, calcular e mostrar a média aritmética.

**Na opção 2:** receber três notas e seus respectivos pesos, calcular e mostrar a média ponderada.

**Na opção 3:** sair do programa.

Verifique a possibilidade de opção inválida, mostrando uma mensagem.

**34.** Em uma eleição presidencial existem quatro candidatos. Os votos são informados por meio de código. Os códigos utilizados são:

1, 2, 3, 4	VOTOS PARA OS RESPECTIVOS CANDIDATOS
5	Voto nulo
6	Voto em branco

Faça um programa que calcule e mostre:

- o total de votos para cada candidato;
- o total de votos nulos;
- o total de votos em branco;
- a percentagem de votos nulos sobre o total de votos;
- a percentagem de votos em branco sobre o total de votos.

Para finalizar o conjunto de votos, tem-se o valor zero.

**35.** Faça um programa que receba como entrada uma lista de números positivos ou negativos, terminada com o número zero. O programa deve fornecer como saída a soma dos números positivos, a soma dos números negativos e a soma das duas somas parciais.

**36.** Faça um programa que receba a idade e a altura de várias pessoas e que calcule e mostre a média das alturas das pessoas com mais de 50 anos. Para encerrar a entrada de dados digite idade menor ou igual a zero.

**37.** Faça um programa que apresente um menu de opções para o cálculo das seguintes operações entre dois números: adição, subtração, multiplicação e divisão. O programa deve possibilitar ao usuário a escolha da operação desejada, a exibição do resultado e a volta ao menu de opções. O programa só termina quando for escolhida a opção de saída.

**38.** Faça um programa que apresente o menu de opções a seguir, que permita ao usuário escolher a opção desejada, receba os dados necessários para executar a operação e mostre o resultado. Verificar a possibilidade de opção inválida e não se preocupar com as restrições, como salário inválido.

Menu de opções:

1. Novo salário
2. Férias
3. Décimo terceiro
4. Sair

Digite a opção desejada

**Na opção 1:** receber o salário de um funcionário, calcular e mostrar o novo salário usando as regras a seguir.

SALÁRIOS	PERCENTAGEM DE AUMENTO
Até R\$ 350,00	15%
De R\$ 350,00 a R\$ 600,00	10%
Acima de R\$ 600,00	5%

**Na opção 2:** receber o salário de um funcionário, calcular e mostrar o valor de suas férias. Sabe-se que as férias equivalem ao seu salário acrescido de  $\frac{1}{3}$ .

**Na opção 3:** receber o salário de um funcionário e o número de meses de trabalho na empresa, no máximo 12, calcular e mostrar o valor do décimo terceiro. Sabe-se que o décimo terceiro equivale ao seu salário multiplicado pelo número de meses de trabalho dividido por 12.

**Na opção 4:** sair do programa.

**39.** Faça um programa que receba um conjunto de valores inteiros e positivos e que calcule e mostre o maior e o menor valor do conjunto. Considere que:

- ◆ para encerrar a entrada de dados, deve ser digitado o valor zero;
- ◆ para valores negativos, deve ser enviada uma mensagem;
- ◆ os valores negativos ou iguais a zero não entrarão nos cálculos.

**40.** Uma agência bancária possui vários clientes que podem fazer investimentos com rendimentos mensais, conforme a tabela a seguir:

TIPO	DESCRIÇÃO	RENDIMENTO MENSAL
1	Poupança	1,5 %
2	Poupança plus	2 %
3	Fundos de renda fixa	4 %

Faça um programa que leia o código do cliente, o tipo da conta e o valor investido e que calcule e mostre o rendimento mensal de acordo com o tipo do investimento. Ao final do programa mostre o total investido e o total de juros pagos.

A leitura terminará quando o código do cliente digitado for menor ou igual a 0.

# CAPÍTULO

# 5

## VETORES

### 5.1 VETOR EM ALGORITMOS

#### 5.1.1 DEFINIÇÃO DE VETOR

Um vetor é uma variável composta homogênea unidimensional formada por uma seqüência de variáveis, todas do mesmo tipo, com o mesmo identificador (mesmo nome) e alocadas seqüencialmente na memória. Uma vez que as variáveis têm o mesmo nome, o que as distingue é um índice, que referencia sua localização dentro da estrutura.

#### 5.1.2 DECLARAÇÃO DE VETOR

DECLARE **nome**[**tamanho**] **tipo**

onde: **nome** é o nome da variável do tipo vetor, **tamanho** é a quantidade de variáveis que vão compor o vetor e **tipo** é o tipo básico de dados que poderá ser armazenado na seqüência de variáveis que formam o vetor.

#### 5.1.3 EXEMPLO DE VETOR

DECLARE X[5] NUMÉRICO



#### 5.1.4 ATRIBUINDO VALORES AO VETOR

X[1] ← 45

X[4] ← 0



#### 5.1.5 CARREGANDO UM VETOR

```
PARA i ← 1 ATÉ 5 FAÇA
INÍCIO
    ESCREVA "Digite o ", i, "º número"
    LEIA X[i]
FIM
```

Simulação:

MEMÓRIA					TELA
i					Digite o 1º número
1					95
2					Digite o 2º número
3					13
4					Digite o 3º número
5					-25
X	95	13	-25	47	Digite o 4º número
	1	2	3	4	47
				5	Digite o 5º número
					0

### 5.1.6 MOSTRANDO OS ELEMENTOS DO VETOR

```

PARA i ← 1 ATÉ 5 FAÇA
INÍCIO
    ESCREVA "Este é o ", i, "º número do vetor"
    ESCREVA X[i]
FIM

```

## 5.2 VETOR EM PASCAL

### 5.2.1 DEFINIÇÃO DE VETOR

As variáveis compostas homogêneas unidimensionais (vetores) são conhecidas na linguagem PASCAL como ARRAY. Uma estrutura do tipo ARRAY é uma seqüência de variáveis, todas do mesmo tipo, com o mesmo identificador (mesmo nome) e alocadas seqüencialmente na memória.

Uma vez que as variáveis têm o mesmo nome, o que as distingue é um índice, que referencia sua localização dentro da estrutura.

### 5.2.2 DECLARAÇÃO DE VETOR

```
VAR nome_da_variável: ARRAY[1..n] OF tipo_dos_dados_do_vetor;
```

onde: **nome\_da\_variável** é o nome da variável do tipo vetor, **n** é a quantidade de variáveis que vão compor o vetor e **tipo\_dos\_dados\_do\_vetor** é o tipo básico de dados que poderá ser armazenado na seqüência de variáveis que formam o vetor.

### 5.2.3 EXEMPLO DE VETOR

```
VAR X:ARRAY[1..10] OF REAL;
```

X	1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	---	----

```
VAR VET: ARRAY[1..5] OF CHAR;
```

VET	1	2	3	4	5
-----	---	---	---	---	---

### 5.2.4 ATRIBUINDO VALORES AO VETOR

X[4]:=5; atribui o valor 5 ao quarto elemento do vetor

VET[1]:='DIA'; atribui a palavra DIA ao primeiro elemento do vetor

### 5.2.5 CARREGANDO UM VETOR

Para ler dados do teclado e atribuir a um vetor:

```
FOR i:= 1 TO 7 DO
BEGIN
  READLN(X[i]);
END;
```

### 5.2.6 MOSTRANDO OS ELEMENTOS DO VETOR

```
FOR i:=1 TO 10 DO
BEGIN
  WRITELN(X[i]);
END;
```

## 5.3 VETOR EM C/C++

### 5.3.1 DEFINIÇÃO DE VETOR

As variáveis compostas homogêneas unidimensionais (vetores) são variáveis capazes de armazenar vários valores. Cada um desses valores é identificado pelo mesmo nome (o nome dado ao vetor), sendo diferenciado apenas por um índice.

Os índices utilizados na linguagem C/C++ para identificar as posições de um vetor começam sempre em 0 (zero) e vão até o tamanho do vetor menos uma unidade.

### 5.3.2 DECLARAÇÃO DE VETOR

Os vetores em C/C++ são identificados pela existência de colchetes logo após o nome da variável no momento da declaração. Dentro dos colchetes deve-se colocar o número de posições do vetor.

### 5.3.3 EXEMPLO DE VETOR

```
int vet[10];
```

No exemplo acima, o vetor chamado **vet** possui dez posições, começando pela posição 0 e indo até a posição 9 (tamanho do vetor – 1). Em cada posição poderão ser armazenados números inteiros, conforme especificado pelo tipo **int** na declaração.

VET	10	5	3	8	1	19	44	21	2	7
	0	1	2	3	4	5	6	7	8	9

char x[5];	X	A	*	2	@	K
	0	1		2	3	4

No exemplo acima, o vetor chamado **x** possui cinco posições, começando pela posição 0 e indo até a posição 4 (tamanho do vetor – 1). Em cada posição poderão ser armazenados caracteres, conforme especificado pelo tipo **char** na declaração.

É importante ressaltar que na linguagem C/C++ não existe o tipo de dado **string**, como ocorre na linguagem PASCAL. Dessa maneira, para poder armazenar uma cadeia de

caracteres, como o nome completo de uma pessoa, deve-se declarar um vetor de char, onde cada posição equivale a um caractere ou a uma letra do nome. Deve-se lembrar, entretanto, que toda vez que se fizer uso de um vetor para armazenar uma cadeia de caracteres, deve-se definir uma posição a mais que a necessária para armazenar a marca de finalização de cadeia (\0).

#### 5.3.4 ATRIBUINDO VALORES AO VETOR

```
vet[0] = 1;           atribui o valor 1 ao primeiro elemento do vetor
strcpy(nome, "Maria"); atribui o nome Maria ao vetor nome
                      (um vetor de seis posições, declarado assim:
                       char nome[6])
```

#### 5.3.5 CARREGANDO UM VETOR

Para ler dados do teclado e atribuí-los a um vetor:

```
for (i = 0; i < 10; i++)
    cin >> vetor[i];
```

#### 5.3.6 IMPRIMINDO UM VETOR

```
for (i=0;i<10;i++)
    cout << vetor[i];
```

## EXERCÍCIOS RESOLVIDOS

- Faça um programa que carregue um vetor de nove elementos numéricos inteiros, calcule e mostre os números primos e suas respectivas posições.

### ALGORITMO

### SOLUÇÃO:

```

ALGORITMO
DECLARE num[9] NUMÉRICO
        i, j, cont NUMÉRICO
PARA i ← 1 ATÉ 9 FAÇA
    INÍCIO
        LEIA num[i]
    FIM
PARA i ← 1 ATÉ 9 FAÇA
    INÍCIO
        cont ← 0
        PARA j ← 1 ATÉ num[i] FAÇA
            INÍCIO
                SE RESTO(num[i] / j) = 0
                ENTÃO cont ← cont + 1
            FIM
        SE cont = 2
        ENTÃO INÍCIO
            ESCREVA num[i]
            ESCREVA i
        FIM
    FIM
FIM_ALGORITMO.
```

**SOLUÇÃO:**

\EXERC\CAP5\PASCAL\EX1.PAS e \EXERC\CAP5\PASCAL\EX1.EXE

**SOLUÇÃO:**

\EXERC\CAP5\C++\EX1.CPP e \EXERC\CAP5\C++\EX1.EXE

- 2.** Faça um programa que receba a quantidade de peças vendidas por vendedor e armazene essas quantidades em um vetor. Receba também o preço da peça vendida de cada vendedor e armazene esses preços em outro vetor. Existem apenas dez vendedores e cada vendedor pode vender apenas um tipo de peça, isto é, para cada vendedor existe apenas um preço. Calcule e mostre a quantidade total de peças vendidas por todos os vendedores e para cada vendedor calcule e mostre o valor total da venda, isto é, a quantidade de peças \* o preço da peça.

**SOLUÇÃO:**

```

ALGORITMO
DECLARE qtd[10], preco[10] NUMÉRICO
      i, tot_geral, tot_vend NUMÉRICO
tot_geral ← 0
PARA i ← 1 ATÉ 10 FAÇA
    INÍCIO
        LEIA qtd[i]
        LEIA preco[i]
    FIM
PARA i ← 1 ATÉ 10 FAÇA
    INÍCIO
        tot_vend ← qtd[i] * preco[i]
        ESCREVA tot_vend
    FIM
PARA i ← 1 ATÉ 10 FAÇA
    INÍCIO
        tot_geral ← tot_geral + qtd[i]
    FIM
ESCREVA tot_geral
FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP5\PASCAL\EX2.PAS e \EXERC\CAP5\PASCAL\EX2.EXE

**SOLUÇÃO:**

\EXERC\CAP5\C++\EX2.CPP e \EXERC\CAP5\C++\EX2.EXE

- 3.** Faça um programa que carregue dois vetores de dez elementos numéricos cada um e mostre um vetor resultante da intercalação desses dois vetores.

Vetor 1	3	5	4	2	2	5	3	2	5	9
	1	2	3	4	5	6	7	8	9	10
Vetor 2	3	5	4	2	2	5	3	2	5	9
	1	2	3	4	5	6	7	8	9	10

### Vetor resultante da intercalação

3	3	5	5	4	4	2	2	2	2	5	5	3	3	2	2	5	5	9	9
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

**ALGORITMO**
**SOLUÇÃO:**

```

ALGORITMO
DECLARE vet1[10], vet2[10], vet3[20] NUMÉRICO
      i, j NUMÉRICO
j ← 1
PARA i ← 1 ATÉ 10 FAÇA
    INÍCIO
        LEIA vet1[i]
        vet3[j] ← vet1[i]
        j ← j + 1
        LEIA vet2[i]
        vet3[j] ← vet2[i]
        j ← j + 1
    FIM
PARA i ← 1 ATÉ 20 FAÇA
    INÍCIO
        ESCREVA vet3[i]
    FIM
FIM_ALGORITMO.

```


**SOLUÇÃO:**

\EXERC\CAP5\PASCAL\EX3.PAS e \EXERC\CAP5\PASCAL\EX3.EXE


**SOLUÇÃO:**

\EXERC\CAP5\C++\EX3.CPP e \EXERC\CAP5\C++\EX3.EXE

4. Faça um programa que carregue um vetor com oito números inteiros, calcule e mostre dois vetores resultantes. O primeiro vetor resultante deve conter os números positivos. O segundo vetor resultante deve conter os números negativos. Cada vetor resultante vai ter *no máximo* oito posições, sendo que nem todas devem obrigatoriamente ser utilizadas.

**ALGORITMO**
**SOLUÇÃO:**

```

ALGORITMO
DECLARE num[8], pos[8], neg[8] NUMÉRICO
      cont, cont_n, cont_p NUMÉRICO
cont_n ← 1
cont_p ← 1
PARA i ← 1 ATÉ 8 FAÇA
    INÍCIO
        LEIA num[i]
        SE num[i] >= 0
            ENTÃO INÍCIO
                pos[cont_p] ← num[i]
                cont_p ← cont_p + 1
            FIM
        SENÃO INÍCIO
            neg[cont_n] ← num[i]
            cont_n ← cont_n + 1
        FIM
    FIM

```

```

SE cont_n = 1
ENTÃO ESCREVA "Vetor de negativos vazio"
SENÃO INÍCIO
    PARA i ← 1 ATÉ cont_n - 1 FAÇA
        INÍCIO
            ESCREVA neg[i]
        FIM
    FIM
SE cont_p = 1
ENTÃO ESCREVA "Vetor de positivos vazio"
SENÃO INÍCIO
    PARA i ← 1 ATÉ cont_p - 1 FAÇA
        INÍCIO
            ESCREVA pos[i]
        FIM
    FIM
FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP5\PASCAL\EX4.PAS e \EXERC\CAP5\PASCAL\EX4.EXE

**SOLUÇÃO:**

\EXERC\CAP5\C++\EX4.CPP e \EXERC\CAP5\C++\EX4.EXE

- 5.** Faça um programa que carregue dois vetores, X e Y, com dez números inteiros cada um. Considere que os números de cada vetor digitado, X e Y, não podem estar repetidos. Calcule e mostre os seguintes vetores resultantes:

◆ a união de X com Y

(todos os elementos de X e os elementos de Y que não estejam em X)

X	3	5	4	2	1	6	8	7	11	9
	1	2	3	4	5	6	7	8	9	10
Y	2	1	5	12	3	0	-1	4	7	6
	1	2	3	4	5	6	7	8	9	10
União	3	5	4	2	1	6	8	7	11	9
	1	2	3	4	5	6	7	8	10	12

◆ a diferença entre X e Y

(todos os elementos de X que não existam em Y)

X	3	5	4	2	1	6	8	7	11	9
	1	2	3	4	5	6	7	8	9	10
Y	2	1	5	12	3	0	-1	4	7	6
	1	2	3	4	5	6	7	8	9	10
Diferença	8	11	9							
	1	2	3							

◆ a soma entre X e Y

(soma de cada elemento de X com o elemento de mesma posição em Y)

X	3	5	4	2	1	6	8	7	11	1
	1	2	3	4	5	6	7	8	9	11
Y	2	1	5	12	3	0	-1	4	7	6
	1	2	3	4	5	6	7	8	9	10
Soma	5	6	9	14	4	6	7	11	18	15
	1	2	3	4	5	6	7	8	9	10

◆ produto entre X e Y

(multiplicação de cada elemento de X com o elemento de mesma posição em Y)

X	3	5	4	2	1	6	8	7	11	9
	1	2	3	4	5	6	7	8	9	10
Y	2	1	5	12	3	0	-1	4	7	6
	1	2	3	4	5	6	7	8	9	10
Multiplicação	6	5	20	24	3	0	-8	28	77	54
	1	2	3	4	5	6	7	8	9	10

◆ a interseção entre X e Y

(apenas os elementos que aparecem nos dois vetores)

X	3	5	4	2	1	6	8	7	11	9
	1	2	3	4	5	6	7	8	9	10
Y	2	1	5	12	3	0	-1	4	7	6
	1	2	3	4	5	6	7	8	9	10
Interseção	3	5	4	2	1	6	7	6	7	
	1	2	3	4	5	6	7	6	7	

### ALGORITMO

### SOLUÇÃO:

#### ALGORITMO

```

DECLARE X[10], Y[10], U[20], D[10], S[10], P[10], IT[10] NUMÉRICO
      i, j, cont_u, cont_d, cont_i NUMÉRICO
PARA i ← 1 ATÉ 10 FAÇA
    INÍCIO
      LEIA X[i]
      LEIA Y[i]
    FIM
    cont_u ← 1
    cont_d ← 1
    cont_i ← 1
    PARA i ← 1 ATÉ 10 FAÇA
      INÍCIO
        j ← 1
        ENQUANTO (X[i] ≠ U[j] E j < cont_u) FAÇA

```

```

    INÍCIO
        j ← j + 1
    FIM
    SE (j >= cont_u)
        ENTÃO INÍCIO
            U[cont_u] ← X[i]
            cont ← cont + 1
        FIM
    FIM
PARA i ← 1 ATÉ 10 FAÇA
    INÍCIO
        j ← 1
        ENQUANTO (Y[i] ≠ U[j] E j < cont_u) FAÇA
            INÍCIO
                j ← j + 1
            FIM
        SE (j >= cont_u)
            ENTÃO INÍCIO
                U[cont_u] ← Y[i]
                cont ← cont + 1
            FIM
        FIM
    PARA i ← 1 ATÉ cont_u FAÇA
        INÍCIO
            ESCREVA U[i]
        FIM
    PARA i ← 1 ATÉ 10 FAÇA
        INÍCIO
            j ← 1
            ENQUANTO (X[i] ≠ Y[j] E j <= 10) FAÇA
                INÍCIO
                    j ← j + 1
                FIM
            SE (j > 10)
                ENTÃO INÍCIO
                    D[cond_d] ← X[i]
                    cont_d ← cont_d + 1
                FIM
            FIM
        PARA i ← 1 ATÉ cont_d FAÇA
            INÍCIO
                ESCREVA (D[i])
            FIM
        PARA i ← 1 ATÉ 10 FAÇA
            INÍCIO
                S[i] ← X[i] + Y[i]
                P[i] ← X[i] * Y[i]
            FIM
        PARA i ← 1 ATÉ 10 FAÇA
            INÍCIO
                ESCREVA S[i]
            FIM
        PARA i ← 1 ATÉ 10 FAÇA
            INÍCIO
                ESCREVA P[i]
            FIM
        PARA i ← 1 ATÉ 10 FAÇA
            INÍCIO
                j ← 1
                ENQUANTO (X[i] ≠ Y[j] E j <= 10) FAÇA
                    INÍCIO
                        j ← j + 1
                    FIM
            FIM
        FIM
    FIM

```

```

SE (j > 10)
ENTÃO INÍCIO
    IT[cont_i] ← X[i]
    cont_i ← cont_i + 1
FIM
PARA i ← 1 ATÉ cont_i FAÇA
    INÍCIO
        ESCREVA IT[i]
    FIM
FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP5\PASCAL\EX5.PAS e \EXERC\CAP5\PASCAL\EX5.EXE

**SOLUÇÃO:**

\EXERC\CAP5\C++\EX5.CPP e \EXERC\CAP5\C++\EX5.EXE

6. Faça um programa que carregue um vetor com dez números inteiros. Calcule e mostre um vetor resultante ordenado de maneira decrescente.

X	3	5	4	2	1	6	8	7	11	9
	1	2	3	4	5	6	7	8	9	10

Ordenado	11	9	8	7	6	5	4	3	2	1
	1	2	3	4	5	6	7	8	9	10

**ALGORITMO****SOLUÇÃO:**

```

ALGORITMO
DECLARE vet[10], i, j, aux NUMÉRICO
PARA ← 1 ATÉ 10 FAÇA
    INÍCIO
        LEIA vet[i]
    FIM
PARA i ← 1 ATÉ 10 FAÇA
    INÍCIO
        PARA j ← 1 ATÉ 9 FAÇA
            INÍCIO
                PARA j ← 1 ATÉ 9 FAÇA
                    INÍCIO
                        SE (vet[j] < vet[j+1])
                        ENTÃO INÍCIO
                            aux ← vet[j]
                            vet[j] ← vet[j+1]
                            vet[j+1] ← aux
                        FIM
                    FIM
                PARA i ← 1 ATÉ 10 FAÇA
                    INÍCIO
                        ESCREVA vet[i]
                    FIM
                FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP5\PASCAL\EX6.PAS e \EXERC\CAP5\PASCAL\EX6.EXE

**SOLUÇÃO:**

\EXERC\CAP5\C++\EX6.CPP e \EXERC\CAP5\C++\EX6.EXE

- 7.** Faça um programa que, no momento de carregar um vetor com oito números inteiros, já o carregue de maneira ordenada crescente.

**SOLUÇÃO:**

```

ALGORITMO
DECLARE vet[8], i, j, aux NUMÉRICO
i ← 1
ENQUANTO (i <= 8) FAÇA
INÍCIO
LEIA aux
j ← 1
ENQUANTO (vet[j] < aux) E (j < i) FAÇA
    INÍCIO
        j ← j + 1
    FIM
z ← i
ENQUANTO (z >= j + 1) FAÇA
    INÍCIO
        vet[z] ← vet[z-1]
        z ← z - 1
    FIM
vet[j] ← aux
i ← i + 1
FIM
PARA i ← 1 ATÉ 8 FAÇA
    INÍCIO
        ESCREVA vet[i]
    FIM
FIM_ALGORITMO.

```

**1<sup>a</sup> SOLUÇÃO - UTILIZANDO APENAS WHILE:**

\EXERC\CAP5\PASCAL\EX7\_A.PAS e \EXERC\CAP5\PASCAL\EX7\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO FOR E WHILE:**

\EXERC\CAP5\PASCAL\EX7\_B.CPP e \EXERC\CAP5\PASCAL\EX7\_B.EXE

**1<sup>a</sup> SOLUÇÃO - UTILIZANDO APENAS WHILE:**

\EXERC\CAP5\C++\EX7\_A.CPP e \EXERC\CAP5\C++\EX7\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO FOR E WHILE:**

\EXERC\CAP5\C++\EX7\_B.CPP e \EXERC\CAP5\C++\EX7\_B.EXE

- 8.** Faça um programa que carregue dois vetores com cinco elementos numéricos cada um, depois ordene-os de maneira crescente. Gere um terceiro vetor com dez posições, que será composto pela intercalação dos vetores anteriores, também de maneira crescente.

X	3	5	4	2	1						
	1	2	3	4	5						
X	1	2	3	4	5						
Ordenado	1	2	3	4	5						
Y	11	2	4	1	6						
	1	2	3	4	5						
Y	1	2	4	6	11						
Ordenado	1	2	3	4	5						
Resultado	1	1	2	2	3	3	4	4	5	6	11
	1	2	3	4	5	6	7	8	9	10	

**ALGORITMO****SOLUÇÃO:**

```

ALGORITMO
DECLARE X[5], Y[5], R[10], i, j, z, aux NUMÉRICO
PARA i ← 1 ATÉ 5 FAÇA
    INÍCIO
        LEIA X[i]
    FIM
PARA i ← 1 ATÉ 5 FAÇA
    INÍCIO
        PARA j ← 1 ATÉ 4 FAÇA
            INÍCIO
                SE (X[j] > X[j+1])
                ENTÃO INÍCIO
                    aux ← X[j]
                    X[j] ← X[j+1]
                    X[j+1] ← aux
                FIM
            FIM
        PARA i ← 1 ATÉ 5 FAÇA
            INÍCIO
                LEIA Y[i]
            FIM
        PARA i ← 1 ATÉ 5 FAÇA
            INÍCIO
                PARA j ← 1 ATÉ 4 FAÇA
                    INÍCIO
                        SE (Y[j] > Y[j+1])
                        ENTÃO INÍCIO
                            aux ← Y[j]
                            Y[j] ← Y[j+1]
                            Y[j+1] ← aux
                        FIM
                    FIM
                j ← 1;
            PARA i ← 1 ATÉ 5 FAÇA
                INÍCIO
                    R[j] ← X[i]
                    j ← j + 1
                    R[j] ← Y[i]
                FIM
            FIM
        FIM
    FIM

```

```

j ← j + 1
FIM
PARA i ← 1 ATÉ 10 FAÇA
INÍCIO
PARA j ← 1 ATÉ 9 FAÇA
INÍCIO
SE (R[j] > R[j+1])
ENTÃO INÍCIO
    aux ← R[j]
    R[j] ← R[j+1]
    R[j+1] ← aux
FIM
FIM
PARA i ← 1 ATÉ 5 FAÇA
INÍCIO
ESCREVA X[i]
FIM
PARA i ← 1 ATÉ 5 FAÇA
INÍCIO
ESCREVA Y[i]
FIM
PARA i ← 1 ATÉ 10 FAÇA
INÍCIO
ESCREVA R[i]
FIM
FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP5\PASCAL\EX8.PAS e \EXERC\CAP5\PASCAL\EX8.EXE

**SOLUÇÃO:**

\EXERC\CAP5\C++\EX8.CPP e \EXERC\CAP5\C++\EX8.EXE

- 9.** Faça um programa que efetue reserva de passagens aéreas de uma certa companhia. O programa deverá ler informações sobre os vôos (número, origem e destino) juntamente com o número de lugares disponíveis para 12 aviões (um vetor para cada um desses dados). O programa deverá apresentar um menu com as seguintes opções:

- ◆ Consultar
- ◆ Efetuar reserva
- ◆ Sair

Quando a opção escolhida for *Consultar* deverá ser disponibilizado mais um menu com as seguintes opções:

- ◆ Por número do vôo
- ◆ Por origem
- ◆ Por destino

Quando a opção escolhida for *Efetuar reserva* deverá ser perguntado o número do vôo no qual a pessoa deseja viajar. O programa deverá dar as seguintes respostas:

- ◆ *Reserva confirmada* – caso exista o vôo e lugar disponível, dando baixa nos lugares disponíveis.
- ◆ *Vôo lotado* – caso não exista lugar disponível nesse vôo.
- ◆ *Vôo inexistente* – caso o código do vôo não exista.

A opção *Sair* é a única que permite sair do programa. Sendo assim, após cada operação de consulta ou reserva o programa volta ao menu principal.

## ALGORITMO

### SOLUÇÃO:

```

ALGORITMO
DECLARE voo[12], lugares[12], i, op, op2, num_voo NUMÉRICO
    origem[12], destino[12], local LITERAL
PARA i ← 1 ATÉ 12 FAÇA
    INÍCIO
        LEIA voo[i]
        LEIA origem[i]
        LEIA destino[i]
        LEIA lugares[i]
    FIM
REPITA
    ESCREVA "1- Consultar"
    ESCREVA "2- Reservar"
    ESCREVA "3- Finalizar"
    ESCREVA "Digite sua opção: "
    LEIA op
    SE (op = 1)
        ENTÃO INÍCIO
            ESCREVA "1- Consulta por vôo"
            ESCREVA "2- Consulta por origem"
            ESCREVA "3- Consulta por destino"
            ESCREVA "Digite sua opção: "
            LEIA op2
            SE (op2 = 1)
                ENTÃO INÍCIO
                    ESCREVA "Digite o número de vôo: "
                    LEIA num_voo
                    i ← 1
                    ENQUANTO ((i < 3) E (voo[i] ≠ num_voo)) FAÇA
                        INÍCIO
                            i ← i + 1
                        FIM
                    SE (i = 3)
                        ENTÃO ESCREVA "Vôo inexistente !"
                    SENÃO INÍCIO
                        ESCREVA "Número do vôo: ", voo[i]
                        ESCREVA "Local de origem: ", origem[i]
                        ESCREVA "Local de destino: ",
                            → destino[i]
                        ESCREVA "Lugares disponíveis: ",
                            → lugares[i]
                    FIM
                FIM
            SE (op2 = 2)
                ENTÃO INÍCIO
                    ESCREVA "Digite o local de origem: "
                    LEIA local
                    PARA i ← 1 ATÉ 3 FAÇA
                        INÍCIO
                            SE (local = origem[i])
                                ENTÃO INÍCIO
                                    ESCREVA "Número do vôo: ", voo[i]
                                    ESCREVA "Local de origem: " ,
                                        → origem[i]
                                    ESCREVA "Local de destino: ",
                                        → destino[i]
                                    ESCREVA "Lugares disponíveis: ",
                                        → lugares[i]

```

```

          FIM
          FIM
          SE (op2 = 3)
          ENTÃO INÍCIO
              ESCREVA "Digite o local de destino: "
              LEIA local
              PARA i ← 1 ATÉ 3 FAÇA
                  INÍCIO
                      SE (local = destino[i])
                      ENTÃO INÍCIO
                          ESCREVA "Número do vôo: ", voo[i]
                          ESCREVA "Local de origem: ",
                              ► origem[i]
                          ESCREVA "Local de destino: ",
                              ► destino[i]
                          ESCREVA "Lugares disponíveis: ",
                              ► lugares[i]
                      FIM
                  FIM
                  FIM
              SE (op = 2)
              ENTÃO INÍCIO
                  ESCREVA "Digite o número do vôo desejado: "
                  LEIA num_voo
                  i ← 0
                  ENQUANTO ((i<3) e (voo[i] != num_voo)) FAÇA
                      INÍCIO
                          i = i + 1
                      FIM
                  SE (i = 3)
                      ENTÃO ESCREVA "Número de vôo não encontrado !"
                      SENÃO INÍCIO
                          SE (lugares[i] = 0)
                          ENTÃO ESCREVA "Vôo lotado !"
                          SENÃO INÍCIO
                              lugares[i] = lugares[i] - 1
                              ESCREVA "Reserva confirmada !"
                          FIM
                      FIM
                  FIM
              ATÉ (op = 3)
          FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP5\PASCAL\EX9.PAS e \EXERC\CAP5\PASCAL\EX9.EXE

**SOLUÇÃO:**

\EXERC\CAP5\C++\EX9.CPP e \EXERC\CAP5\C++\EX9.EXE

- 10.** Faça um programa para corrigir provas de múltipla escolha. Cada prova tem dez questões e cada questão vale 1 ponto. O primeiro conjunto de dados a ser lido é o gabarito da prova. Os outros dados serão os números dos alunos e suas respectivas respostas. Existem 15 alunos matriculados. Calcule e mostre:

- ♦ para cada aluno seu número e sua nota;
- ♦ a percentagem de aprovação, sabendo-se que a nota mínima é 6,0.



**ALGORITMO****SOLUÇÃO:**

```

ALGORITMO
DECLARE gab[10], resp[10] LITERAL
      num, pontos, tot_ap, perc_ap, i, j NUMÉRICO
PARA i ← 1 ATÉ 10 FAÇA
    INÍCIO
        ESCREVA "Digite a resposta da questão ", i
        LEIA gab[i]
    FIM
tot_ap ← 0
PARA i ← 1 ATÉ 15 FAÇA
    INÍCIO
        ESCREVA "Digite o número do ", i, "º aluno"
        LEIA num
        pontos ← 0
        PARA j ← 1 ATÉ 10 FAÇA
            INÍCIO
                ESCREVA "Digite a resposta dada pelo aluno ", num, " à "
                ↪ ", j, "ª questão
                LEIA resp[j]
                SE (resp[j] = gab[j])
                    ENTÃO pontos ← pontos + 1
            FIM
        ESCREVA "A nota do aluno ", num, " foi ", pontos
        SE (pontos >= 6)
            ENTÃO tot_ap ← tot_ap + 1
        FIM
    perc_ap ← tot_ap * 100 / 15
    ESCREVA "O percentual de alunos aprovados é ", perc_ap
FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP5\PASCAL\EX10.PAS e \EXERC\CAP5\PASCAL\EX10.EXE

**SOLUÇÃO:**

\EXERC\CAP5\C++\EX10.CPP e \EXERC\CAP5\C++\EX10.EXE

- 11.** Faça um programa que receba a temperatura média de cada mês do ano e armazene-as em um vetor. Calcule e mostre a maior e a menor temperatura do ano e em que mês elas ocorreram (mostrar o mês por extenso: 1 – Janeiro, 2 – Fevereiro, ...).

**OBSERVAÇÃO:**

Desconsiderar empates.

**ALGORITMO****SOLUÇÃO:**

```

ALGORITMO
DECLARE temp[12], cont, maior, menor, maior_mes, menor_mes NUMÉRICO
PARA cont ← 1 ATÉ 12 FAÇA
    INÍCIO
        LEIA temp[cont]
        SE (cont = 1)
            ENTÃO INÍCIO
                maior ← temp[cont]
                menor ← temp[cont]
                maior_mes ← cont
                menor_mes ← cont
            FIM
        SENÃO INÍCIO
            SE (temp[cont] > maior)

```

```

        ENTÃO INÍCIO
            maior ← temp[cont]
            maior_mes ← cont
        FIM
        SE (temp[cont] < menor)
        ENTÃO INÍCIO
            menor ← temp[cont]
            menor_mes ← cont
        FIM
        FIM
    FIM
    ESCREVA maior
    SE (maior_mes = 1)
        ENTÃO ESCREVA "JANEIRO"
    SE (maior_mes = 2)
        ENTÃO ESCREVA "FEVEREIRO"
    SE (maior_mes = 3)
        ENTÃO ESCREVA "MARÇO"
    SE (maior_mes = 4)
        ENTÃO ESCREVA "ABRIL"
    SE (maior_mes = 5)
        ENTÃO ESCREVA "MAIO"
    SE (maior_mes = 6)
        ENTÃO ESCREVA "JUNHO"
    SE (maior_mes = 7)
        ENTÃO ESCREVA "JULHO"
    SE (maior_mes = 8)
        ENTÃO ESCREVA "AGOSTO"
    SE (maior_mes = 9)
        ENTÃO ESCREVA "SETEMBRO"
    SE (maior_mes = 10)
        ENTÃO ESCREVA "OUTUBRO"
    SE (maior_mes = 11)
        ENTÃO ESCREVA "NOVEMBRO"
    SE (maior_mes = 12)
        ENTÃO ESCREVA "DEZEMBRO"
    ESCREVA menor
    SE (menor_mes = 1)
        ENTÃO ESCREVA "JANEIRO"
    SE (menor_mes = 2)
        ENTÃO ESCREVA "FEVEREIRO"
    SE (menor_mes = 3)
        ENTÃO ESCREVA "MARÇO"
    SE (menor_mes = 4)
        ENTÃO ESCREVA "ABRIL"
    SE (menor_mes = 5)
        ENTÃO ESCREVA "MAIO"
    SE (menor_mes = 6)
        ENTÃO ESCREVA "JUNHO"
    SE (menor_mes = 7)
        ENTÃO ESCREVA "JULHO"
    SE (menor_mes = 8)
        ENTÃO ESCREVA "AGOSTO"
    SE (menor_mes = 9)
        ENTÃO ESCREVA "SETEMBRO"
    SE (menor_mes = 10)
        ENTÃO ESCREVA "OUTUBRO"
    SE (menor_mes = 11)
        ENTÃO ESCREVA "NOVEMBRO"
    SE (menor_mes = 12)
        ENTÃO ESCREVA "DEZEMBRO"
FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP5\PASCAL\EX11.PAS e \EXERC\CAP5\PASCAL\EX11.EXE

**SOLUÇÃO:**

\EXERC\CAP5\C++\EX11.CPP e \EXERC\CAP5\C++\EX11.EXE

- 12.** Faça um programa que carregue um vetor com os modelos de cinco carros (exemplos de modelos: FUSCA, GOL, VECTRA etc). Carregue um outro vetor com o consumo desses carros, isto é, quantos quilômetros cada um desses carros faz com um litro de combustível. Calcule e mostre:

- ◆ o modelo do carro mais econômico;
- ◆ quantos litros de combustível cada um dos carros cadastrados consome para percorrer uma distância de 1.000 quilômetros.

**SOLUÇÃO:**

```

ALGORITMO
DECLARE consumo[5], menor_cons, menor_vei, valor, i NUMÉRICO
        veiculo[5] LITERAL
PARA i ← 1 ATÉ 5 FAÇA
    INÍCO
        LEIA veiculo[i]
    FIM
PARA i ← 1 ATÉ 5 FAÇA
    INÍCO
        LEIA consumo[i]
        SE (i = 1)
            ENTÃO INÍCIO
                menor_cons ← consumo
                menor_vei ← i
            FIM
        SENÃO INÍCIO
            SE (consumo[i] > menor_cons)
                ENTÃO INÍCIO
                    menor_cons ← consumo[i]
                    menor_vei ← i
                FIM
            FIM
        valor ← 1000 / consumo[i]
        ESCREVA"Utilizando o ", veiculo[i], "gastaria ", valor, "para"
        ↪ percorrer 1000 Km"
    FIM
FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP5\PASCAL\EX12.PAS e \EXERC\CAP5\PASCAL\EX12.EXE

**SOLUÇÃO:**

\EXERC\CAP5\C++\EX12.CPP e \EXERC\CAP5\C++\EX12.EXE

- 13.** Faça um programa que carregue um vetor com dez números inteiros. Calcule e mostre os números superiores a 50 e suas respectivas posições. Mostrar mensagem se não existir nenhum número nessa condição.

**ALGORITMO****SOLUÇÃO:**

```

ALGORITMO
    DECLARE vet[10] NUMÉRICO
        achou LÓGICO
        i NUMÉRICO
    PARA i ← 1 ATÉ 10 FAÇA
        INÍCIO
        LEIA vet[i]
        FIM
        achou ← falso
    PARA i ← 1 ATÉ 10 FAÇA
        INÍCIO
        SE vet[i] > 50
        ENTÃO INÍCIO
            ESCREVA vet[i], i
            achou ← verdadeiro
        FIM
        FIM
        SE achou = falso
        ENTÃO ESCREVA "Não existe nenhum número superior a 50 no vetor"
    FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP5\PASCAL\EX13.PAS e \EXERC\CAP5\PASCAL\EX13.EXE

**SOLUÇÃO:**

\EXERC\CAP5\C++\EX13.CPP e \EXERC\CAP5\C++\EX13.EXE

- 14.** Faça um programa que carregue três vetores com cinco posições cada um. O primeiro vetor receberá os nomes de cinco funcionários. O segundo vetor receberá os salários dos cinco funcionários e o terceiro vetor receberá a quantidade de anos que cada funcionário trabalha na empresa. Mostre um primeiro relatório apenas com os nomes dos funcionários que não terão aumento. Mostre um segundo relatório apenas com os nomes e os novos salários dos funcionários que terão aumento. Sabe-se que os funcionários que terão direito ao aumento são aqueles que possuem tempo de serviço superior a cinco anos ou salário inferior a R\$ 200,00. Sabe-se, ainda, que se o funcionário satisfizer as duas condições acima (tempo de serviço e salário) o aumento será de 35%; para o funcionário que satisfizer apenas a condição de tempo de serviço, o aumento será de 25%; para o funcionário que satisfizer apenas a condição de salário, o aumento será de 15%.

**ALGORITMO****SOLUÇÃO:**

```

ALGORITMO
    DECLARE nome[5] LITERAL
        sal[5], quant[5] NUMÉRICO
        i, novo_sal NUMÉRICO
    PARA i ← 1 ATÉ 5 FAÇA
        INÍCIO
        LEIA nome[i]
        LEIA sal[i]
        LEIA quant[i]
        FIM
    PARA i ← 1 ATÉ 5 FAÇA
        INÍCIO

```

```

SE (quant[i] <= 5) E (sal[i] >= 200)
ENTÃO ESCREVA nome[i]
FIM
PARA i ← 1 ATÉ 5 FAÇA
INÍCIO
SE (quant[i] > 5) OU (sal[i] < 200)
ENTÃO INÍCIO
    SE (quant[i] > 5) E (sal[i] < 200)
        ENTÃO novo_sal ← sal[i] + 35% * sal[i]
        SENÃO SE (quant[i] > 5)
            ENTÃO novo_sal ← sal[i] + 25% * sal[i]
            SENÃO novo_sal ← sal[i] + 15% * sal[i]
        ESCREVA nome[i], novo_sal
    FIM
FIM
FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP5\PASCAL\EX14.PAS e \EXERC\CAP5\PASCAL\EX14.EXE

**SOLUÇÃO:**

\EXERC\CAP5\C++\EX14.CPP e \EXERC\CAP5\C++\EX14.EXE

- 15.** Faça um programa que carregue um primeiro vetor com dez números inteiros e um segundo vetor com cinco números inteiros. Mostre uma lista dos números do primeiro vetor com seus respectivos divisores armazenados no segundo vetor, bem como as suas posições.

Exemplo de saída do programa:

NUM	5	12	4	7	10	3	2	6	23	16
	1	2	3	4	5	6	7	8	9	10

DIVIS	3		11		5		8		2	
	1		2		3		4		5	

Número 5

Divisível por 5 na posição 3

Número 12

Divisível por 3 na posição 1

Divisível por 2 na posição 5

Número 4

Divisível por 2 na posição 5

Número 7

Não é divisível por nenhum número do segundo vetor

Número 10

Divisível por 5 na posição 3

Divisível por 2 na posição 5

Para saber se um número é divisível por outro, testar o resto.

Exemplo: RESTO(5/5) = 0

**ALGORITMO****SOLUÇÃO:**

```

ALGORITMO
    DECLARE vet1[10], vet2[5] NUMÉRICO
        i, j NUMÉRICO
        achou LÓGICO
    PARA i ← 1 ATÉ 10 FAÇA
        INÍCIO
        LEIA vet1[i]
        FIM
    PARA j ← 1 ATÉ 5 FAÇA
        INÍCIO
        LEIA vet2[j]
        FIM
    PARA i ← 1 ATÉ 10 FAÇA
        INÍCIO
        achou ← falso
        ESCREVA vet1[i]
        PARA j ← 1 ATÉ 5 FAÇA
            INÍCIO
            SE RESTO(vet1[i]/vet2[j]) = 0
            ENTÃO INÍCIO
                ESCREVA vet2[j], j
                achou ← verdadeiro
            FIM
            FIM
            SE achou = falso
            ENTÃO ESCREVA "Não é divisível por nenhum número do segundo
            ↪ vetor"
            FIM
        FIM_ALGORITMO.
    
```

**SOLUÇÃO:**

\EXERC\CAP5\PASCAL\EX15.PAS e \EXERC\CAP5\PASCAL\EX15.EXE

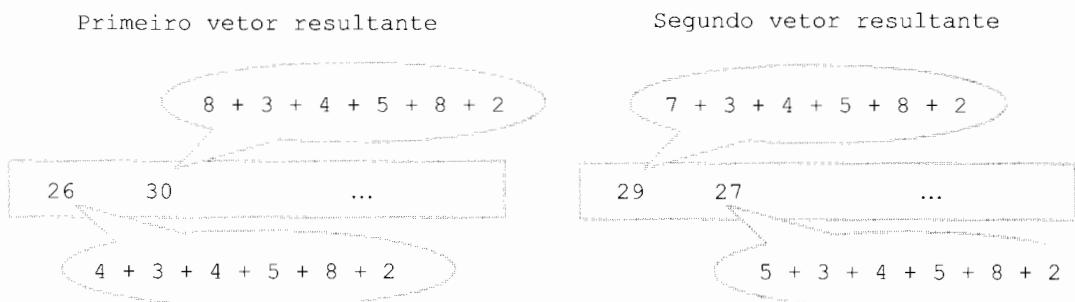
**SOLUÇÃO:**

\EXERC\CAP5\C++\EX15.CPP e \EXERC\CAP5\C++\EX15.EXE

- 16.** Faça um programa que carregue um vetor com dez números inteiros e um segundo vetor com cinco números inteiros. Calcule e mostre dois vetores resultantes. O primeiro vetor resultante será composto pelos números pares gerados pelo elemento do primeiro vetor somado a todos os elementos do segundo vetor. O segundo vetor resultante será composto pelos números ímpares gerados pelo elemento do primeiro vetor somado a todos os elementos do segundo vetor.

Primeiro vetor	4	7	5	8	2	15	9	6	10	11
	1	2	3	4	5	6	7	8	9	10

Segundo vetor	3	4	5	8	2
	1	2	3	4	5

**ALGORITMO****SOLUÇÃO:**

ALGORITMO

```

DECLARE vet1[10], vet2[5] NUMÉRICO
vet_result1[10], vet_result2[10] NUMÉRICO
i, j, poslivre1, poslivre2, soma NUMÉRICO
PARA i ← 1 ATÉ 10 FAÇA
    INÍCIO
    LEIA vet1[i]
    FIM
PARA j ← 1 ATÉ 5 FAÇA
    INÍCIO
    LEIA vet2[j]
    FIM
    poslivre1 ← 1
    poslivre2 ← 1
    PARA i ← 1 ATÉ 10 FAÇA
        INÍCIO
        soma ← vet1[i]
        PARA j ← 1 ATÉ 5 FAÇA
            INÍCIO
            soma ← soma + vet2[j]
            FIM
        SE RESTO(soma/2) = 0
        ENTÃO INÍCIO
            vet_result1[poslivre1] ← soma
            poslivre1 ← poslivre1 + 1
            FIM
        SENÃO INÍCIO
            vet_result2[poslivre2] ← soma
            poslivre2 ← poslivre2 + 1
            FIM
        FIM
    PARA i ← 1 ATÉ (poslivre1 -1) FAÇA
        INÍCIO
        ESCREVA vet_result1[poslivre1]
        FIM
    PARA i ← 1 ATÉ (poslivre2 -1) FAÇA
        INÍCIO
        ESCREVA vet_result2[poslivre2]
        FIM
    FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP5\PASCAL\EX16.PAS e \EXERC\CAP5\PASCAL\EX16.EXE

**SOLUÇÃO:**

\EXERC\CAP5\C++\EX16.CPP e \EXERC\CAP5\C++\EX16.EXE

-  17. Faça um programa que receba seis números inteiros, calcule e mostre a soma dos números pares e a quantidade dos números ímpares, mostrando o relatório a seguir.

Vetor	2	4	5	6	3	7
	1	2	3	4	5	6

#### Relatório

Os números pares são:  
 número 2 na posição 1  
 número 4 na posição 2  
 número 6 na posição 4

Soma dos pares = 12

Os números ímpares são:  
 número 5 na posição 3  
 número 3 na posição 5  
 número 7 na posição 6

Quantidade dos ímpares = 3

#### ALGORITMO

**SOLUÇÃO:**

ALGORITMO

```

DECLARE num[6] NUMÉRICO
      i, soma, qtde NUMÉRICO
      achou LÓGICO
PARA i ← 1 ATÉ 6 FAÇA
  INÍCIO
  LEIA num[i]
  FIM
  soma ← 0
  achou ← falso
PARA i ← 1 ATÉ 6 FAÇA
  INÍCIO
  SE RESTO(num[i]/2) = 0
  ENTÃO INÍCIO
    achou ← verdadeiro
    ESCREVA num[i], i
    soma ← soma + num[i]
    FIM
  FIM
  SE achou = falso
  ENTÃO ESCREVA "Não existe nenhum número par"
  SENÃO ESCREVA soma
  qtde ← 0
  achou ← falso
PARA i ← 1 ATÉ 6 FAÇA
  INÍCIO
  SE RESTO(num[i]/2) ≠ 0
  ENTÃO INÍCIO
    achou ← verdadeiro
    ESCREVA num[i], i
    qtde ← qtde + 1
    FIM
  FIM
  SE achou = falso
  ENTÃO ESCREVA "Não existe nenhum número ímpar"
  SENÃO ESCREVA qtde
FIM_ALGORITMO.
```

**SOLUÇÃO:**

\EXERC\CAP5\PASCAL\EX17.PAS e \EXERC\CAP5\PASCAL\EX17.EXE

**SOLUÇÃO:**

\EXERC\CAP5\C++\EX17.CPP e \EXERC\CAP5\C++\EX17.EXE

18. Faça um programa que receba o número sorteado em um dado durante 20 jogadas, mostre os números sorteados e a freqüência com que apareceram.

**SOLUÇÃO:**

```

ALGORITMO
    DECLARE dado[20] NUMÉRICO
        i, num1, num2, num3, num4, num5, num6 NUMÉRICO
    PARA i ← 1 ATÉ 20 FAÇA
        INÍCIO
        LEIA dado[i]
        ENQUANTO (dado[i] < 1) OU (dado[i] > 6) FAÇA
            INÍCIO
            LEIA dado[i]
            FIM
        FIM
    PARA i ← 1 ATÉ 20 FAÇA
        INÍCIO
        ESCREVA dado[i]
        FIM
    num1 ← 0
    num2 ← 0
    num3 ← 0
    num4 ← 0
    num5 ← 0
    num6 ← 0
    PARA i ← 1 ATÉ 20 FAÇA
        INÍCIO
        SE dado[i] = 1
        ENTÃO num1 ← num1 + 1
        SE dado[i] = 2
        ENTÃO num2 ← num2 + 1
        SE dado[i] = 3
        ENTÃO num3 ← num3 + 1
        SE dado[i] = 4
        ENTÃO num4 ← num4 + 1
        SE dado[i] = 5
        ENTÃO num5 ← num5 + 1
        SE dado[i] = 6
        ENTÃO num6 ← num6 + 1
        FIM
    ESCREVA num1, num2, num3, num4, num5, num6
    FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP5\PASCAL\EX18.PAS e \EXERC\CAP5\PASCAL\EX18.EXE

**SOLUÇÃO:**

\EXERC\CAP5\C++\EX18.CPP e \EXERC\CAP5\C++\EX18.EXE

-  19. Faça um programa que carregue dois vetores de 20 posições de caracteres. A seguir, troque o 1º elemento de A com o 20º de B, o 2º de A com o 19º de B, assim por diante, até trocar o 20º de A com o 1º de B. Mostre os vetores antes e depois da troca.

Vetor 1 – antes da troca

A	G	Y	W	5	V	S	8	6	J	G	A	W	2	M	C	H	Q	6	L
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

Vetor 2 – antes da troca

S	D	4	5	H	G	R	U	8	9	K	S	A	1	2	V	4	D	5	M
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

Vetor 1 – depois da troca

M	5	D	4	V	2	1	A	S	K	9	8	U	R	G	H	5	4	D	S
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

Vetor 2 – depois da troca

L	6	Q	H	C	M	2	W	A	G	J	6	8	S	V	5	W	Y	G	A
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

**ALGORITMO****SOLUÇÃO:**

```

ALGORITMO
DECLARE vet1[20], vet2[20] LITERAL
aux LITERAL
i, j NUMÉRICO
PARA i ← 1 ATÉ 20 FAÇA
    INÍCIO
    LEIA vet1[i]
    FIM
PARA j ← 1 ATÉ 20 FAÇA
    INÍCIO
    LEIA vet2[j]
    FIM
PARA i ← 1 ATÉ 20 FAÇA
    INÍCIO
    ESCREVA vet1[i]
    FIM
PARA j ← 1 ATÉ 20 FAÇA
    INÍCIO
    ESCREVA vet2[j]
    FIM
j ← 20
PARA i ← 1 ATÉ 20 FAÇA
    INÍCIO
    aux ← vet1[i]
    vet1[i] ← vet2[j]
    vet2[j] ← aux
    j ← j -1
    FIM
PARA i ← 1 ATÉ 20 FAÇA
    INÍCIO
    ESCREVA vet1[i]
    FIM
PARA j ← 1 ATÉ 20 FAÇA

```

```

INÍCIO
ESCREVA vet2[j]
FIM
FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP5\PASCAL\EX19.PAS e \EXERC\CAP5\PASCAL\EX19.EXE

**SOLUÇÃO:**

\EXERC\CAP5\C++\EX19.CPP e \EXERC\CAP5\C++\EX19.EXE

- 20.** Faça um programa que leia um código numérico inteiro e um vetor de cinco posições de números reais. Se o código for zero, termine o programa. Se o código for 1, mostre o vetor na ordem direta. Se o código for 2, mostre o vetor na ordem inversa.

**SOLUÇÃO:**

```

ALGORITMO
DECLARE vet[5] NUMÉRICO
          i, cod NUMÉRICO
PARA i ← 1 ATÉ 5 FAÇA
    INÍCIO
    LEIA vet[i]
    FIM
LEIA cod
SE cod = 0
ENTÃO ESCREVA "fim"
SE cod = 1
ENTÃO INÍCIO
    PARA i ← 1 ATÉ 5 FAÇA
        INÍCIO
        ESCREVA vet[i]
        FIM
    FIM
SE cod = 2
ENTÃO INÍCIO
    PARA i ← 5 ATÉ 1 FAÇA
        INÍCIO
        ESCREVA vet[i]
        FIM
    FIM
SE (cod < 0) OU (cod > 2)
ENTÃO ESCREVA "Código inválido"
FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP5\PASCAL\EX20.PAS e \EXERC\CAP5\PASCAL\EX20.EXE

**SOLUÇÃO:**

\EXERC\CAP5\C++\EX20.CPP e \EXERC\CAP5\C++\EX20.EXE

-  **21.** Faça um programa que leia um conjunto de 15 valores e armazene-os em um vetor. A seguir, separe-os em dois outros vetores (P e I) com cinco posições cada. O vetor P armazena números pares e o vetor I armazena números ímpares. Como o tamanho dos vetores pode não ser suficiente para armazenar todos os números, deve-se sempre verificar se os mesmos já estão cheios. Caso P ou I estejam cheios, deve-se mostrá-los e recomeçar o preenchimento da primeira posição. Terminado o processamento, mostrar o conteúdo restante dentro dos vetores P e I.

**ALGORITMO****SOLUÇÃO:**

```

ALGORITMO
    DECLARE vet[15], p[5], i[5] NUMÉRICO
        cont, k, poslivre_p, poslivre_i NUMÉRICO
    PARA cont ← 1 ATÉ 15 FAÇA
        INÍCIO
        LEIA vet[cont]
        FIM
        poslivre_p ← 1
        poslivre_i ← 1
    PARA cont ← 1 ATÉ 15 FAÇA
        INÍCIO
        SE RESTO(vet[cont]/2) = 0
        ENTÃO INÍCIO
            p[poslivre_p] ← vet[cont]
            poslivre_p ← poslivre_p + 1
            FIM
        SENÃO INÍCIO
            i[poslivre_i] ← vet[cont]
            poslivre_i ← poslivre_i + 1
            FIM
        SE poslivre_p = 6
        ENTÃO INÍCIO
            ESCREVA "Vetor de pares cheio"
            PARA k ← 1 ATÉ (poslivre_p - 1) FAÇA
                INÍCIO
                ESCREVA p[k]
                FIM
                poslivre_p ← 1
            FIM
            SE poslivre_i = 6
            ENTÃO INÍCIO
                ESCREVA "Vetor de ímpares cheio"
                PARA k ← 1 ATÉ (poslivre_i - 1) FAÇA
                    INÍCIO
                    ESCREVA i[k]
                    FIM
                    poslivre_i ← 1
                FIM
            FIM
            SE poslivre_p ≠ 1
            ENTÃO INÍCIO
                ESCREVA "Vetor de pares restante"
                PARA k ← 1 ATÉ (poslivre_p - 1) FAÇA
                    INÍCIO
                    ESCREVA p[k]
                    FIM
                FIM
            SE poslivre_i ≠ 1
            ENTÃO INÍCIO
                ESCREVA "Vetor de ímpares restante"
                PARA k ← 1 ATÉ (poslivre_i - 1) FAÇA

```

```

INÍCIO
ESCREVA i[k]
FIM
FIM
FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP5\PASCAL\EX21.PAS e \EXERC\CAP5\PASCAL\EX21.EXE

**SOLUÇÃO:**

\EXERC\CAP5\C++\EX21.CPP e \EXERC\CAP5\C++\EX21.EXE

- 22.** Faça um programa que simule um controle bancário. Para tanto, devem ser lidos os códigos de dez contas e os seus respectivos saldos. Os códigos devem ser armazenados em um vetor de números inteiros (não pode haver mais que uma conta com o mesmo código) e os saldos devem ser armazenados em um vetor de números reais. O saldo deverá ser cadastrado na mesma posição do código. Por exemplo, se a conta 504 foi armazenada na 5<sup>a</sup> posição do vetor de códigos, o seu saldo deverá ficar na 5<sup>a</sup> posição do vetor de saldos. Depois de fazer a leitura dos valores, mostrar o seguinte menu na tela:

1. Efetuar depósito
  2. Efetuar saque
  3. Consultar o ativo bancário (ou seja, o somatório dos saldos de todos os clientes)
  4. Finalizar o programa
- ◆ Para efetuar depósito deve-se solicitar o código da conta e o valor a ser depositado. Se a conta não estiver cadastrada, mostrar a mensagem *Conta não encontrada* e voltar ao menu. Se a conta existir, atualizar o seu saldo.
  - ◆ Para efetuar saque deve-se solicitar o código da conta e o valor a ser sacado. Se a conta não estiver cadastrada, mostrar a mensagem *Conta não encontrada* e voltar ao menu. Se a conta existir, verificar se o seu saldo é suficiente para cobrir o saque. (Estamos supondo que a conta não pode ficar com o saldo negativo). Se o saldo for suficiente, realizar o saque e voltar ao menu. Caso contrário, mostrar a mensagem *Saldo insuficiente* e voltar ao menu.
  - ◆ Para consultar o ativo bancário deve-se somar o saldo de todas as contas do banco. Depois de mostrar esse valor, voltar ao menu.
  - ◆ O programa só termina quando for digitada a opção 4 – *Finalizar o programa*.

**SOLUÇÃO:**

```

ALGORITMO
DECLARE conta[10], saldo[10] NUMÉRICO
      i, j, codigo, valor, soma, op NUMÉRICO
      achou LÓGICO
PARA i ← 1 ATÉ 10 FAÇA
    INÍCIO
    achou ← falso
    REPITA
      LEIA conta[i]
      PARA j ← 1 ATÉ (i-1) FAÇA
        INÍCIO
        SE conta[i] = conta[j]
        ENTÃO achou ← verdadeiro
        FIM
    FIM
  FIM

```

```

ATÉ achou = falso
LEIA saldo[i]
FIM
REPEAT
    LEIA op
    achou ← falso
    SE op = 1
        ENTÃO INÍCIO
            LEIA codigo, valor
            PARA i ← 1 ATÉ 10 FAÇA
                INÍCIO
                    SE codigo = conta[i]
                        ENTÃO INÍCIO
                            saldo[i] ← saldo[i] + valor
                            achou ← verdadeiro
                            ESCREVA "Depósito efetuado"
                            FIM
                FIM
                SE achou = falso
                    ENTÃO ESCREVA "Conta não cadastrada"
                    FIM
                SE op = 2
                    ENTÃO INÍCIO
                        LEIA codigo, valor
                        PARA i ← 1 ATÉ 10 FAÇA
                            INÍCIO
                                SE codigo = conta[i]
                                    ENTÃO INÍCIO
                                        SE saldo[i] < valor
                                            ENTÃO INÍCIO
                                                ESCREVA "Saldo insuficiente"
                                                achou ← verdadeiro
                                            FIM
                                    FIM
                                    SENÃO INÍCIO
                                        saldo[i] ← saldo[i] - valor
                                        achou ← verdadeiro
                                        ESCREVA "Saque efetuado"
                                        FIM
                                FIM
                                SE achou = falso
                                    ENTÃO ESCREVA "Conta não cadastrada"
                                    FIM
                                SE op = 3
                                    ENTÃO INÍCIO
                                        PARA i ← 1 ATÉ 10 FAÇA
                                            INÍCIO
                                                soma ← soma + saldo[i]
                                            FIM
                                        ESCREVA soma
                                        FIM
                                    SE (op < 1) OU (op > 4)
                                        ENTÃO ESCREVA "Opção inválida"
                                ATÉ op = 4
                                FIM_ALGORITMO.

```

SOLUÇÃO:

\EXERC\CAP5\PASCAL\EX22.PAS e \EXERC\CAP5\PASCAL\EX22.EXE

**SOLUÇÃO:**

\EXERC\CAP5\C++\EX22.CPP e \EXERC\CAP5\C++\EX22.EXE

- 23.** Uma empresa possui ônibus com 48 lugares (24 nas janelas e 24 no corredor). Faça um programa que utilize dois vetores para controlar as poltronas ocupadas no corredor e na janela. Considere que zero representa poltrona desocupada e um representa poltrona ocupada.

Janela	0	1	0	0	...	1	0	0
	1	2	3	4	...	21	22	24
Corredor	0	0	0	1	...	1	0	0
	1	2	3	4	...	21	22	24

Esse programa deve controlar a venda de passagens da seguinte maneira:

- ◆ o cliente informa se deseja poltrona no corredor ou na janela e, depois, o programa deve informar quais poltronas estão disponíveis para a venda;
- ◆ quando não existirem poltronas livres no corredor, nas janelas ou, ainda, quando o ônibus estiver completamente cheio, deve ser mostrada uma mensagem.

**SOLUÇÃO:**

## ALGORITMO

```

DECLARE corredor[24], janela[24] NUMÉRICO
achou LÓGICO
posi LITERAL
i NUMÉRICO
PARA i ← 1 ATÉ 24 FAÇA
    INÍCIO
        corredor[i] ← 0
        janela[i] ← 0
    FIM
LEIA posi
achou ← falso
PARA i ← 1 ATÉ 24 FAÇA
    INÍCIO
        SE (corredor[i] = 0) OU (janela[i] = 0)
        ENTÃO achou ← verdadeiro
    FIM
SE achou = falso
ENTÃO ESCREVA "Ônibus lotado"
SENÃO INÍCIO
    SE posi = "J"
    ENTÃO INÍCIO
        achou ← falso
        PARA i ← 1 ATÉ 24 FAÇA
            INÍCIO
                SE janela[i] = 0
                ENTÃO INÍCIO
                    ESCREVA janela[i]
                    achou ← verdadeiro
                FIM
            FIM
        SE achou = falso
        ENTÃO ESCREVA "Não tem lugar disponível na janela"
    FIM

```

```

SE pos1 = "C"
ENTÃO INÍCIO
    achou ← falso
    PARA i ← 1 ATÉ 24 FAÇA
        INÍCIO
            SE corredor[i] = 0
            ENTÃO INÍCIO
                ESCREVA corredor[i]
                achou ← verdadeiro
            FIM
        FIM
        SE achou = falso
        ENTÃO ESCREVA "Não tem lugar disponível no corredor"
        FIM
    FIM
FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP5\PASCAL\EX23.PAS e \EXERC\CAP5\PASCAL\EX23.EXE

**SOLUÇÃO:**

\EXERC\CAP5\C++\EX23.CPP e \EXERC\CAP5\C++\EX23.EXE

- 24.** Faça um programa que leia um vetor A de dez posições contendo números inteiros. Determine e mostre, a seguir, quais os elementos de A que estão repetidos e quantas vezes cada um se repete.

Exemplo:

Vetor A	5	4	3	18	5	3	4	18	4	18	
	1	2	3	4	5	6	7	8	9	10	

Caso sejam digitados valores como os apresentados no vetor A acima, o programa deverá mostrar ao final as seguintes informações:

- ◆ o número 5 aparece duas vezes;
- ◆ o número 4 aparece três vezes;
- ◆ o número 3 aparece duas vezes;
- ◆ o número 18 aparece três vezes.

**SOLUÇÃO:**

```

ALGORITMO
DECLARE a[10], repetidos[10], vezes[10] NUMÉRICO
        i, j, qtde, cont, cont_r NUMÉRICO
PARA i ← 1 ATÉ 10 FAÇA
    INÍCIO
        LEIA a[i]
    FIM
    cont_r ← 1
PARA i ← 1 ATÉ 10 FAÇA
    INÍCIO
        qtde ← 1
        PARA j ← 1 ATÉ 10 FAÇA
            INÍCIO
                SE i ≠ j
                ENTÃO SE a[i] = a[j]
                    ENTÃO qtde ← qtde + 1
            FIM
    FIM

```

```

SE qtde > 1
ENTÃO INÍCIO
    cont ← 1
    ENQUANTO ((cont < cont_r) E (a[i] ≠
        repetidos[cont])) FAÇA
        INÍCIO
            cont ← cont + 1
        FIM
    SE cont = cont_r
    ENTÃO INÍCIO
        repetidos[cont_r] ← a[i]
        vezes[cont_r] ← qtde
        cont_r ← cont_r + 1
    FIM
FIM
PARA i ← 1 ATÉ cont_r - 1 FAÇA
ESCREVA "O número ",repetidos[i], " apareceu ",vezes[i]," vezes"
FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP5\PASCAL\EX24.PAS e \EXERC\CAP5\PASCAL\EX24.EXE

**SOLUÇÃO:**

\EXERC\CAP5\C++\EX24.CPP e \EXERC\CAP5\C++\EX24.EXE

- 25.** Faça um programa que gere os dez primeiros números primos acima de 100 e armazene-os em um vetor, escrevendo ao final o vetor resultante.

**SOLUÇÃO:**

```

ALGORITMO
    DECLARE primos[10] NUMÉRICO
        i, qtde, num, divi NUMÉRICO
    num ← 101
    qtde ← 1
    REPITA
        divi ← 0
        PARA i ← 1 ATÉ num FAÇA
            INÍCIO
                SE RESTO(num/i) = 0
                ENTÃO divi ← divi + 1
            FIM
        SE divi = 2
        ENTÃO INÍCIO
            primos[qtde] ← num
            qtde ← qtde + 1
        FIM
        num ← num + 1
    ATÉ qtde = 11
    PARA i ← 1 ATÉ 10 FAÇA
        INÍCIO
            ESCREVA primos[i]
        FIM
    FIM_ALGORITMO.

```

RESOLUÇÃO  
PASCAL

SOLUÇÃO:

\EXERC\CAP5\PASCAL\EX25.PAS e \EXERC\CAP5\PASCAL\EX25.EXE

RESOLUÇÃO  
C/C++

SOLUÇÃO:

\EXERC\CAP5\C++\EX25.CPP e \EXERC\CAP5\C++\EX25.EXE

## EXERCÍCIOS PROPOSTOS

1. Faça um programa que carregue um vetor de seis elementos numéricos inteiros, calcule e mostre:
  - ◆ a quantidade de números pares;
  - ◆ quais os números pares;
  - ◆ a quantidade de números ímpares;
  - ◆ quais os números ímpares.
2. Faça um programa que carregue um vetor com sete números inteiros, calcule e mostre:
  - ◆ os números múltiplos de 2;
  - ◆ os números múltiplos de 3;
  - ◆ os números múltiplos de 2 e de 3.
3. Faça um programa para controlar o estoque de mercadorias de uma empresa. Inicialmente o programa deverá ler dois vetores com dez posições cada, onde o primeiro corresponde ao código do produto e o segundo corresponde ao total desse produto em estoque. Logo após, o programa deverá ler um conjunto indeterminado de dados contendo o código de um cliente, o código do produto que este deseja comprar juntamente com a quantidade. Código do cliente igual a zero indica fim do programa. O programa deverá verificar:
  - ◆ se o código do produto solicitado existe. Se existir, tentar atender o pedido; caso contrário, exibir mensagem *Código inexistente*.
  - ◆ cada pedido feito por um cliente só pode ser atendido integralmente. Caso isso não seja possível, escrever a mensagem *Não temos estoque suficiente desta mercadoria*. Se puder atendê-lo, escrever a mensagem *Pedido atendido. Obrigado e volte sempre*;
  - ◆ efetuar a atualização do estoque somente se o pedido for atendido integralmente;
  - ◆ no final do programa, escrever os códigos dos produtos com seus respectivos estoques já atualizados.
4. Faça um programa que carregue um vetor com 15 elementos inteiros e verifique a existência de elementos iguais a 30, mostrando as posições em que esses elementos apareceram.
5. Uma escola deseja saber se existem alunos cursando, simultaneamente, as disciplinas Lógica e Linguagem de Programação. Coloque os números das matrículas dos alunos que cursam Lógica em um vetor, no máximo 15 alunos. Coloque os números das matrículas dos alunos que cursam Linguagem de Programação em outro vetor, no máximo dez alunos. Mostre o número da matrícula que aparece nos dois vetores.

**6.** Faça um programa que receba o total das vendas de cada vendedor e armazene-as em um vetor. Receba também o percentual de comissão de cada vendedor a armazene-os em outro vetor. Receba os nomes desses vendedores e armazene-os em um terceiro vetor. Existem apenas dez vendedores. Calcule e mostre:

- ◆ um relatório com os nomes dos vendedores e os valores a receber;
- ◆ o total das vendas de todos os vendedores;
- ◆ o maior valor a receber e quem o receberá;
- ◆ o menor valor a receber e quem o receberá.

**7.** Faça um programa que carregue um vetor com dez números reais, calcule e mostre a quantidade de números negativos e a soma dos números positivos desse vetor.

**8.** Faça um programa que carregue um vetor com os nomes de sete alunos. Carregue um outro vetor com a média final desses alunos. Calcule e mostre:

- ◆ o nome do aluno com maior média (desconsiderar empates);
- ◆ para cada aluno que ainda não está aprovado, isto é, com média menor que 7,0, mostrar quanto esse aluno precisa tirar no exame para ser aprovado. Considerar que a média para aprovação no exame é 5,0.

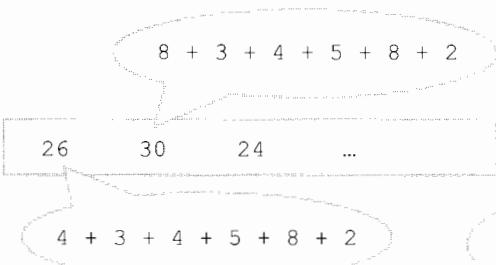
**9.** Faça um programa que carregue três vetores com dez posições cada um. O primeiro vetor com os nomes de dez produtos. O segundo vetor com os códigos dos dez produtos e o terceiro vetor com os preços dos produtos. Mostre um relatório *apenas* com o nome, o código, o preço e o novo preço *dos produtos que sofrerão aumento*. Sabe-se que os produtos que sofrerão aumento são aqueles que possuem código par ou preço superior a R\$ 1.000,00. Sabe-se ainda que se o produto satisfaz as duas condições acima (código e preço), o aumento de preço será de 20% se satisfaz apenas a condição de código; o aumento será de 15%; se satisfaz apenas a condição de preço, o aumento será de 10%.

**10.** Faça um programa que carregue um vetor com dez números inteiros e um segundo vetor com cinco números inteiros. Calcule e mostre dois vetores resultantes. O primeiro vetor resultante será composto pelo número par do primeiro vetor somado aos números do segundo vetor. O segundo vetor resultante será composto pela quantidade de divisores de cada número ímpar do primeiro vetor pelo segundo vetor.

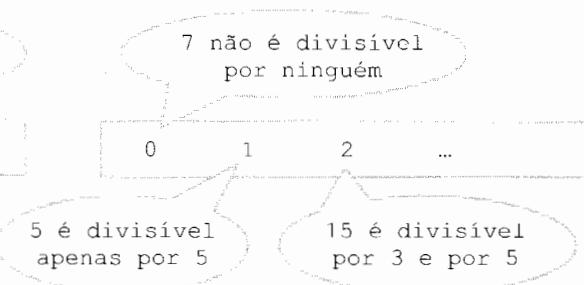
Primeiro vetor	4	7	5	8	2	15	9	6	10	11
	1	2	3	4	5	6	7	8	9	10

Segundo vetor	3	4	5	8	2
	1	2	3	4	5

Primeiro vetor resultante



Segundo vetor resultante



**11.** Faça um programa que receba dez números inteiros e armazene-os em um vetor. Calcule e mostre dois vetores resultantes. O primeiro com os números pares e o segundo com os números ímpares.

**12.** Faça um programa que receba cinco números e mostre a saída a seguir:

```
Digite o 1º número
5
Digite o 2º número
3
Digite o 3º número
2
Digite o 4º número
0
Digite o 5º número
2
Os números digitados foram:
5 + 3 + 2 + 0 + 2 = 12
```

**13.** Faça um programa que receba o nome e a nota de oito alunos e mostre o relatório a seguir:

```
Digite o nome do 1º aluno
Carlos
Digite a nota do Carlos
8
Digite o nome do 2º aluno
Pedro
Digite a nota do Pedro
5
Relatórios de notas
Aluno Nota
Carlos 8.0
Pedro 5.0
..
..
Média da classe = ??
```

**14.** Faça um programa que receba o nome e duas notas de seis alunos e mostre o relatório abaixo:

Relatório de notas:

ALUNO	1ª PROVA	2ª PROVA	MÉDIA	SITUAÇÃO
Carlos	8,0	9,0	8,5	Aprovado
Pedro	4,0	5,0	4,5	Reprovado

- ◆ média da classe = ?
- ◆ quantidade de aprovados = ?%
- ◆ quantidade de alunos de exames = ?%
- ◆ quantidade de reprovados = ?%

**15.** Faça um programa que receba o nome de oito clientes e armazene-os em um vetor. Em um segundo vetor armazene a quantidade de fitas locadas em 1999 por um dos oito clientes. Sabe-se que para cada dez locações o cliente tem direito a uma locação grátis. Faça um programa que mostre o nome de todos os clientes com a quantidade de locações grátis a que ele tem direito.

**16.** Faça um programa que receba o nome de cinco produtos e seus respectivos preços, calcule e mostre:

- ◆ a quantidade de produtos com preço inferior a R\$ 50,00;
- ◆ o nome dos produtos com preço entre R\$ 50,00 e R\$ 100,00;
- ◆ a média dos preços dos produtos com preço superior a R\$ 100,00.

**17.** Faça um programa que carregue dois vetores de dez posições cada um, calcule e mostre um terceiro vetor que contenha os elementos dos dois vetores anteriores ordenados de maneira decrescente.

**18.** Faça um programa que carregue um vetor com 15 posições, calcule e mostre:

- ◆ o maior elemento do vetor e em que posição esse elemento se encontra;
- ◆ o menor elemento do vetor e em que posição esse elemento se encontra.

**19.** Faça um programa que leia dois vetores de dez posições e faça a multiplicação dos elementos de mesmo índice, colocando o resultado em um terceiro vetor. Mostre o vetor resultante.

**20.** Faça um programa que leia um vetor de 50 posições de números inteiros e mostre somente os números positivos.

**21.** Faça um programa que leia um vetor inteiro de 30 posições e crie um segundo vetor, substituindo os valores nulos por 1. Mostre os dois vetores.

**22.** Faça um programa que leia um vetor de dez posições. Em seguida, compacte o vetor, retirando os valores nulos e negativos. Coloque o resultado no vetor B, mostrando o vetor resultante.

**23.** Faça um programa que leia dois vetores (A e B) de cinco posições de números inteiros. O programa deve, então, subtrair o primeiro elemento de A do último de B, acumulando o valor, subtrair o segundo elemento de A do penúltimo de B, acumulando o valor e assim por diante. Mostre o resultado da soma de todas as subtrações.

**24.** Faça um programa que leia um vetor de 15 posições com números inteiros. Crie, a seguir, um vetor resultante que contenha todos os números primos do vetor digitado. Escreva o vetor resultante.

**25.** Faça um programa que leia um vetor de 15 posições de números inteiros e divida todos os seus elementos pelo maior valor do vetor. Mostre o vetor após os cálculos.

# CAPÍTULO

# 6

# MATRIZ

## 6.1 MATRIZ EM ALGORITMOS

### 6.1.1 DEFINIÇÃO DE MATRIZ

Uma matriz é uma variável composta homogênea bidimensional formada por uma seqüência de variáveis, todas do mesmo tipo, com o mesmo identificador (mesmo nome) e alocadas seqüencialmente na memória. Uma vez que as variáveis têm o mesmo nome, o que as distingue são índices que referenciam sua localização dentro da estrutura. Uma variável do tipo matriz é composta por linhas e colunas.

### 6.1.2 DECLARAÇÃO DE MATRIZ

```
DECLARE nome[linha,coluna] tipo
```

onde: **nome** é o nome da variável do tipo matriz, **linha** é a quantidade de linhas que vão compor a matriz, **coluna** é a quantidade de colunas que vão compor a matriz e **tipo** é o tipo de dados que poderá ser armazenado na seqüência de variáveis que formam a matriz.

### 6.1.3 EXEMPLO DE MATRIZ

```
DECLARE X[3,5] NUMÉRICO
```

	1	2	3	4	5
X	1				
	2				
	3				

X [1,1]    X [3,4]

### 6.1.4 ATRIBUINDO VALORES À MATRIZ

```
X[1,2] ← 45                                    X[2,1] ← 13  
X[2,5] ← 0                                    X[3,3] ← -4
```

	1	2	3	4	5
X	1	45			
	2	13			0
	3		-4		

### 6.1.5 CARREGANDO UMA MATRIZ

```

PARA i ← 1 ATÉ 3 FAÇA
INÍCIO
  PARA j ← 1 ATÉ 5 FAÇA
  INÍCIO
    ESCREVA "Digite o número da linha ", i, " e coluna ", j
    LEIA X[i, j]
  FIM
FIM

```

Simulação:

MEMÓRIA		TELA				
i	j					
1	1					Digite o número da linha 1 e coluna 1 12
	2					Digite o número da linha 1 e coluna 2 9
	3					Digite o número da linha 1 e coluna 3 3
	4					Digite o número da linha 1 e coluna 4 7
	5					Digite o número da linha 1 e coluna 5 -23
2	1					Digite o número da linha 2 e coluna 1 15
	2					Digite o número da linha 2 e coluna 2 4
	3					Digite o número da linha 2 e coluna 3 2
	4					Digite o número da linha 2 e coluna 4 34
	5					Digite o número da linha 2 e coluna 5 -4
3	1					Digite o número da linha 3 e coluna 1 3
	2					Digite o número da linha 3 e coluna 2 45
	3					Digite o número da linha 3 e coluna 3 3
	4					Digite o número da linha 3 e coluna 4 0
	5					Digite o número da linha 3 e coluna 5 -3
	1	2	3	4	5	
X	1	12	9	3	7	-23
	2	15	4	2	34	-4
	3	3	45	3	0	-3

### 6.1.6 MOSTRANDO OS ELEMENTOS DE UMA MATRIZ

```

PARA i ← 1 ATÉ 3 FAÇA
INÍCIO
  PARA j ← 1 ATÉ 5 FAÇA
  INÍCIO
    ESCREVA X[i, j]
  FIM
FIM

```

## 6.2 MATRIZ EM PASCAL

### 6.2.1 DEFINIÇÃO DE MATRIZ

As variáveis compostas homogêneas bidimensionais (matrizes) são conhecidas na linguagem PASCAL como ARRAY. Uma estrutura do tipo ARRAY é uma seqüência de

variáveis, todas do mesmo tipo, com o mesmo identificador (mesmo nome) e alocadas seqüencialmente na memória.

Uma vez que as variáveis têm o mesmo nome, o que as distingue são dois índices que referenciam sua localização dentro da estrutura.

### 6.2.2 DECLARAÇÃO DE MATRIZ

```
VAR nome_da_variável: ARRAY[1..m,1..n] OF tipo_dos_dados_da
  ↪ matriz;
```

onde: **nome da variável** é o nome da variável do tipo matriz, **m** é a quantidade de linhas da matriz, **n** é a quantidade de colunas da matriz e **tipo dos dados da matriz** é o tipo básico de dados que poderá ser armazenado na seqüência de variáveis que formam a matriz.

### 6.2.3 EXEMPLO DE MATRIZ

```
VAR X: ARRAY[1..2,1..6] OF REAL;
```

	1	2	3	4	5	6
X	1					
	2					

```
VAR MAT: ARRAY[1..4,1..2] OF CHAR;
```

	1	2
MAT	1	
	2	
	3	
	4	

### 6.2.4 ATRIBUINDO VALORES À MATRIZ

`X[1, 4]:=5` atribui o valor 5 para a posição referente à linha 1 coluna 4 da matriz

	1	2	3	4	5	6
X	1					
	2					

`MAT[3, 2]:='D'` atribui a letra D para a posição referente à linha 3 coluna 2 da matriz

	1	2
MAT	1	
	2	
	3	
	4	

### 6.2.5 CARREGANDO UMA MATRIZ

Para ler dados do teclado e atribuir a uma matriz.

```
FOR i:= 1 TO 7 DO
  BEGIN
    FOR j:=1 TO 3 DO
```

```

BEGIN
  READLN(MAT[i,j]);
END;
END;

```

### 6.2.6 MOSTRANDO OS ELEMENTOS DE UMA MATRIZ

```

FOR i:=1 TO 10 DO
BEGIN
  FOR j:= 1 TO 6 DO
  BEGIN
    WRITELN(X[i,j]);
  END;
END;

```

## 6.3 MATRIZ EM C/C++

### 6.3.1 DEFINIÇÃO DE MATRIZ

Uma matriz pode ser definida como um conjunto de variáveis de mesmo tipo e identificadas pelo mesmo nome. Essas variáveis são diferenciadas por meio da especificação de suas posições dentro dessa estrutura.

A linguagem C/C++ permite a declaração de matrizes unidimensionais (mais conhecidas como *vetores* – descritos no capítulo anterior), bidimensionais e multidimensionais. O padrão ANSI prevê, no mínimo, 12 dimensões; entretanto, o limite de dimensões fica por conta da quantidade de recurso disponível ao compilador. Apesar disso, as matrizes mais utilizadas possuem duas dimensões.

### 6.3.2 DECLARAÇÃO DE MATRIZ

```

tipo_dos_dados nome_variável [dimensão1] [dimensão2] [...]
[dimensão n];

```

onde:

**tipo\_dos\_dados** é o tipo de dados que poderá ser armazenado na seqüência de variáveis que formam a matriz; **nome\_variável** é o nome dado à variável do tipo matriz; [**dimensão 1**] [**dimensão 2**] [...] [**dimensão n**] representam as possíveis dimensões da matriz.

### 6.3.3 EXEMPLO DE MATRIZ

```
float X[2][6];
```

Da mesma maneira como ocorre com os vetores, os índices começam sempre em 0 (zero). Sendo assim, com a declaração anterior, criou-se uma variável chamada X contendo duas linhas (0 e 1) com seis colunas cada (0 a 5), capazes de armazenar números reais, como pode ser observado a seguir.

	0	1	2	3	4	5
X	0	.	.	.	.	.
	1	.	.	.	.	.

```
char MAT [4][3];
```

A declaração anterior criou uma variável chamada MAT contendo quatro linhas (0 a 3) com três colunas cada (0 e 2), capazes de armazenar símbolos, como pode ser observado a seguir.

	0	1	2	
0				
MAT	1			
2				
3				

### 6.3.4 ATRIBUINDO VALORES À MATRIZ

`X[1][4]=5` atribui o valor 5 à posição referente à linha 1 (2ª linha) coluna 4 (5ª coluna) da matriz.

	0	1	2	3	4	5
x	0					
	1					

`MAT[3][2] = 'D'` atribui a letra D à posição referente à linha 3 (4ª linha) coluna 2 (3ª coluna) da matriz.

	0	1	2			
0						
MAT	1					
2						
3			D			

### 6.3.5 CARREGANDO UMA MATRIZ

Para ler dados do teclado e atribuir a uma matriz, supondo que a mesma tenha sido declarada como `int MAT[7][3]`, pode-se executar os comandos a seguir.

```
for (i=0;i<7;i++)
    { for (j=0;j<3;j++)
        cin >> MAT[i][j];
    }
```

Como a matriz possui sete linhas, o `for` externo deve variar de 0 a 6 (percorrendo, assim, as sete linhas da matriz) e o `for` interno deve variar de 0 a 2 (percorrendo, assim, as três colunas da matriz).

### 6.3.6 MOSTRANDO OS ELEMENTOS DE UMA MATRIZ

Para mostrar os valores armazenados dentro de uma matriz, supondo que a mesma tenha sido declarada como `float X[10][6]`, pode-se executar os comandos a seguir.

```
for (i=0;i<10;i++)
    { for (j=0;j<6;j++)
        cout << X[i][j];
    }
```

Como a matriz possui dez linhas, o `for` externo deve variar de 0 a 9 (percorrendo, assim, as dez linhas da matriz) e o `for` interno deve variar de 0 a 5 (percorrendo, assim, as seis colunas da matriz).

## EXERCÍCIOS RESOLVIDOS

1. Faça um programa que carregue uma matriz  $2 \times 2$ , calcule e mostre uma matriz resultante que será a matriz digitada multiplicada pelo maior elemento da matriz.

**ALGORITMO****SOLUÇÃO:**

```

ALGORITMO
    DECLARE mat[2,2], resultado[2,2] NUMÉRICO
        i, j, maior NUMÉRICO
    PARA i ← 1 ATÉ 2 FAÇA
        INÍCIO
        PARA j ← 1 ATÉ 2 FAÇA
            INÍCIO
            LEIA mat[i,,j]
            FIM
        FIM
        maior ← mat[1,1]
    PARA i ← 1 ATÉ 2 FAÇA
        INÍCIO
        PARA j ← 1 ATÉ 2 FAÇA
            INÍCIO
            SE mat[i,,j] > maior
            ENTÃO maior ← mat[i,,j]
            FIM
        FIM
    PARA i ← 1 ATÉ 2 FAÇA
        INÍCIO
        PARA j ← 1 ATÉ 2 FAÇA
            INÍCIO
            resultado[i,,j] ← maior * mat[i,,j]
            FIM
        FIM
    PARA i ← 1 ATÉ 2 FAÇA
        INÍCIO
        PARA j ← 1 ATÉ 2 FAÇA
            INÍCIO
            ESCREVA resultado[i,,j]
            FIM
        FIM
    FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP6\PASCAL\EX1.PAS e \EXERC\CAP6\PASCAL\EX1.EXE

**SOLUÇÃO:**

\EXERC\CAP6\C++\EX1.CPP e \EXERC\CAP6\C++\EX1.EXE

2. Faça um programa que carregue uma matriz  $10 \times 3$  com as notas de dez alunos em três provas. Mostre um relatório com o número do aluno (número da linha) e a prova em que cada aluno obteve menor nota. Ao final do relatório, mostre quantos alunos tiveram menor nota na prova 1, quantos alunos tiveram menor nota na prova 2 e quantos alunos tiveram menor nota na prova 3.

**SOLUÇÃO:**

```

ALGORITMO
DECLARE notas[10,3] NUMÉRICO
    q1, q2, q3, menor, prova_menor, i, j NUMÉRICO
PARA i ← 1 ATÉ 10 FAÇA
    INÍCIO
        PARA j ← 1 ATÉ 3 FAÇA
            INÍCIO
                LEIA notas[i,,j]
            FIM
        FIM
    FIM
    q1 ← 0
    q2 ← 0
    q3 ← 0
PARA i ← 1 ATÉ 10 FAÇA
    INÍCIO
        ESCREVA i
        menor ← notas[i,1]
        prova_menor ← 1
        PARA j ← 1 ATÉ 3 FAÇA
            INÍCIO
                SE notas[i,,j] < menor
                ENTÃO INÍCIO
                    menor ← notas[i, j]
                    prova_menor ← j
                FIM
            FIM
        ESCREVA prova_menor
        SE prova_menor = 1
        ENTÃO q1 ← q1 + 1
        SE prova_menor = 2
        ENTÃO q2 ← q2 + 1
        SE prova_menor = 3
        ENTÃO q3 ← q3 + 1
    FIM
    ESCREVA q1, q2, q3
FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP6\PASCAL\EX2.PAS e \EXERC\CAP6\PASCAL\EX2.EXE

**SOLUÇÃO:**

\EXERC\CAP6\C++\EX2.CPP e \EXERC\CAP6\C++\EX2.EXE

**3.** Faça um programa que carregue:

- ◆ um vetor com oito posições com os nomes das lojas;
- ◆ um outro vetor com quatro posições com os nomes dos produtos;
- ◆ uma matriz com os preços de todos os produtos em cada loja.

O programa deve mostrar todas as relações (nome do produto – nome da loja) nas quais o preço não ultrapasse R\$ 120,00.

**SOLUÇÃO:**

```

ALGORITMO
DECLARE lojas[8] LITERAL
    produtos[4] LITERAL

```

```

        pre[4,8] NUMÉRICO
        i, j NUMÉRICO
PARA j ← 1 ATÉ 8 FAÇA
    INÍCIO
    LEIA lojas[j]
    FIM
PARA i ← 1 ATÉ 4 FAÇA
    INÍCIO
    LEIA produtos[i]
    FIM
PARA i ← 1 ATÉ 4 FAÇA
    INÍCIO
    PARA j ← 1 ATÉ 8 FAÇA
        INÍCIO
        LEIA pre[i, j]
        FIM
    FIM
PARA i ← 1 ATÉ 4 FAÇA
    INÍCIO
    PARA j ← 1 ATÉ 8 FAÇA
        INÍCIO
        SE pre[i, j] < 120
        ENTÃO ESCREVA produtos[i], lojas[j]
        FIM
    FIM
FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP6\PASCAL\EX3.PAS e \EXERC\CAP6\PASCAL\EX3.EXE

**SOLUÇÃO:**

\EXERC\CAP6\C++\EX3.CPP e \EXERC\CAP6\C++\EX3.EXE

- 4.** Faça um programa que carregue uma matriz  $10 \times 20$  com números inteiros e some cada uma das linhas, armazenando o resultado das somas em um vetor. A seguir, multiplique cada elemento da matriz pela soma da linha e mostre a matriz resultante.

**SOLUÇÃO:**

```

ALGORITMO
    DECLARE mat[10,20] NUMÉRICO
                soma[10] NUMÉRICO
                i, j NUMÉRICO
PARA i ← 1 ATÉ 10 FAÇA
    INÍCIO
    PARA j ← 1 ATÉ 20 FAÇA
        INÍCIO
        LEIA mat[i, j]
        FIM
    FIM
PARA i ← 1 ATÉ 10 FAÇA
    INÍCIO
    PARA j ← 1 ATÉ 20 FAÇA
        INÍCIO
        soma[i] ← soma[i] + mat[i, j]
        FIM
    FIM

```

```

PARA i ← 1 ATÉ 10 FAÇA
    INÍCIO
        PARA j ← 1 ATÉ 20 FAÇA
            INÍCIO
                mat[i, j] ← mat[i, j] * soma[i]
            FIM
        FIM
    PARA i ← 1 ATÉ 10 FAÇA
        INÍCIO
            PARA j ← 1 ATÉ 20 FAÇA
                INÍCIO
                    ESCREVA mat[i, j]
                FIM
            FIM
        FIM
    FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP6\PASCAL\EX4.PAS e \EXERC\CAP6\PASCAL\EX4.EXE

**SOLUÇÃO:**

\EXERC\CAP6\C++\EX4.CPP e \EXERC\CAP6\C++\EX4.EXE

5. Na teoria dos sistemas define-se o elemento MINMAX de uma matriz como sendo o maior elemento da linha onde se encontra o menor elemento da matriz. Faça um programa que carregue uma matriz  $4 \times 7$  com números reais, calcule e mostre seu MINMAX e sua posição (linha e coluna).

**SOLUÇÃO:**

```

ALGORITMO
    DECLARE mat[4, 7] NUMÉRICO
    menor, maior, i, j, linha_menor, coluna NUMÉRICO
    PARA i ← 1 ATÉ 4 FAÇA
        INÍCIO
            PARA j ← 1 ATÉ 7 FAÇA
                INÍCIO
                    LEIA mat[i, j]
                FIM
            FIM
        menor ← mat[1, 1]
        linha_menor ← 1
        PARA i ← 1 ATÉ 4 FAÇA
            INÍCIO
                PARA j ← 1 ATÉ 7 FAÇA
                    INÍCIO
                        SE mat[i, j] < menor
                        ENTÃO INÍCIO
                            menor ← mat[i, j]
                            linha_menor ← i
                        FIM
                    FIM
                FIM
            maior ← mat[linha_menor, 1]
            coluna ← 1
            PARA j ← 1 ATÉ 7 FAÇA
                INÍCIO
                    SE mat[linha_menor, j] > maior
                    ENTÃO INÍCIO
                        maior ← mat[linha_menor, j]

```

```

        coluna ← j
    FIM
FIM
ESCREVA maior, linha_menor, coluna
FIM_ALGORITMO.
```

**SOLUÇÃO:**

\EXERC\CAP6\PASCAL\EX5.PAS e \EXERC\CAP6\PASCAL\EX5.EXE

**SOLUÇÃO:**

\EXERC\CAP6\C++\EX5.CPP e \EXERC\CAP6\C++\EX5.EXE

6. Faça um programa que carregue uma primeira matriz de ordem  $4 \times 5$  e uma segunda matriz  $5 \times 2$ , calcule e mostre a matriz resultante do produto matricial das duas matrizes anteriores, armazenando-o em uma terceira matriz de ordem  $4 \times 2$ .

**SOLUÇÃO:**

```

ALGORITMO
DECLARE a[4, 5] NUMÉRICO
        b[5, 2] NUMÉRICO
        c[4, 2] NUMÉRICO
        i, j, k, soma, mult NUMÉRICO
PARA i ← 1 ATÉ 4 FAÇA
    INÍCIO
        PARA j ← 1 ATÉ 5 FAÇA
            INÍCIO
                LEIA a[i, j]
            FIM
        FIM
    PARA i ← 1 ATÉ 5 FAÇA
        INÍCIO
            PARA j ← 1 ATÉ 2 FAÇA
                INÍCIO
                    LEIA b[i, j]
                FIM
            FIM
        FIM
    soma ← 0
    PARA i ← 1 ATÉ 4 FAÇA
        INÍCIO
            PARA k ← 1 ATÉ 2 FAÇA
                INÍCIO
                    PARA j ← 1 ATÉ 5 FAÇA
                        INÍCIO
                            mult ← a[i, j] * b[j, k]
                            soma ← soma + mult
                        FIM
                    c[i, k] ← soma
                    soma ← 0
                FIM
            FIM
        FIM
    PARA i ← 1 ATÉ 4 FAÇA
        INÍCIO
            PARA j ← 1 ATÉ 2 FAÇA
                INÍCIO
                    ESCREVA c[i, j]
                FIM
            FIM
        FIM
FIM_ALGORITMO.
```

**SOLUÇÃO:**

```
\EXERC\CAP6\PASCAL\EX6.PAS e \EXERC\CAP6\PASCAL\EX6.EXE
```

**SOLUÇÃO:**

```
\EXERC\CAP6\C++\EX6.CPP e \EXERC\CAP6\C++\EX6.EXE
```

- 7.** Um elemento  $A_{ij}$  de uma matriz é dito ponto de sela da matriz A se, e somente se,  $A_{ij}$  for ao mesmo tempo o menor elemento da linha  $i$  e o maior elemento da coluna  $j$ . Faça um programa que carregue uma matriz de ordem  $5 \times 7$ , verifique se a matriz possui ponto de sela e, se possuir, mostre seu valor e sua localização.

**SOLUÇÃO:**

```

ALGORITMO
DECLARE mat[5,.7] NUMÉRICO
      i, j, maior, menor, linha, coluna, cont NUMÉRICO
PARA i ← 1 ATÉ 5 FAÇA
    INÍCIO
        PARA j ← 1 ATÉ 7 FAÇA
            INÍCIO
                LEIA mat[i, j]
            FIM
        FIM
    cont ← 0
    PARA i ← 1 ATÉ 5 FAÇA
        INÍCIO
            linha ← i
            coluna ← 0
            menor ← mat[i, 1]
            PARA j ← 1 ATÉ 7 FAÇA
                INÍCIO
                    SE mat[i, j] < menor
                    ENTÃO INÍCIO
                        menor ← mat[i, j]
                        linha ← i
                        coluna ← j
                    FIM
                FIM
            PARA j ← 1 ATÉ 5 FAÇA
                INÍCIO
                    SE mat[j, coluna] > maior
                    ENTÃO INÍCIO
                        maior ← mat[j, coluna]
                    FIM
                FIM
            SE menor = maior
            ENTÃO INÍCIO
                ESCREVA "Ponto de sela = ",maior," na posição ",linha,
                ↪ coluna
                cont ← cont + 1
            FIM
        FIM
    SE cont = 0
    ENTÃO ESCREVA "Não existe ponto de sela nesta matriz"
FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP6\PASCAL\EX7.PAS e \EXERC\CAP6\PASCAL\EX7.EXE

**SOLUÇÃO:**

\EXERC\CAP6\C++\EX7.CPP e \EXERC\CAP6\C++\EX7.EXE

- 8.** Faça um programa que carregue uma matriz  $6 \times 4$  com números inteiros, calcule e mostre quantos elementos dessa matriz são maiores que 30 e, em seguida, monte uma segunda matriz com os elementos diferentes de 30. No lugar do número 30 da segunda matriz coloque o número zero.

**SOLUÇÃO:**

```

ALGORITMO
    DECLARE mat1[6,4], mat2[6,4] NUMÉRICO
        i, j, qtde NUMÉRICO
    PARA i ← 1 ATÉ 6 FAÇA
        INÍCIO
        PARA j ← 1 ATÉ 4 FAÇA
            INÍCIO
            LEIA mat1[i, j]
            FIM
        FIM
        qtde ← 0
    PARA i ← 1 ATÉ 6 FAÇA
        INÍCIO
        PARA j ← 1 ATÉ 4 FAÇA
            INÍCIO
            SE mat1[i, j] > 30
            ENTÃO qtde ← qtde + 1
            FIM
        FIM
    PARA i ← 1 ATÉ 6 FAÇA
        INÍCIO
        PARA j ← 1 ATÉ 4 FAÇA
            INÍCIO
            SE mat1[i, j] = 30
            ENTÃO mat2[i, j] ← 0
            SENÃO mat2[i, j] ← mat1[i, j]
            FIM
        FIM
    ESCREVA qtde
    PARA i ← 1 ATÉ 6 FAÇA
        INÍCIO
        PARA j ← 1 ATÉ 4 FAÇA
            INÍCIO
            ESCREVA mat2[i, j]
            FIM
        FIM
    FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP6\PASCAL\EX8.PAS e \EXERC\CAP6\PASCAL\EX8.EXE



## SOLUÇÃO:

\EXERC\CAP6\C++\EX8.CPP e \EXERC\CAP6\C++\EX8.EXE

9. Faça um programa que carregue uma matriz  $15 \times 5$  com números inteiros, calcule e mostre quais os elementos da matriz que se repetem e quantas vezes cada um está repetido.

## ALGORITMO

## SOLUÇÃO:

```

ALGORITMO
DECLARE mat[15,5], rep[15,5], vezes[15,5] NUMÉRICO
    i, j, k, l, lin, lin2, col, x, num, qtde, achou NUMÉRICO
PARA i ← 1 ATÉ 15 FAÇA
    INÍCIO
        PARA j ← 1 ATÉ 5 FAÇA
            INÍCIO
                LEIA mat[i,j]
            FIM
        FIM
        lin ← 1
        col ← 1
    PARA i ← 1 ATÉ 15 FAÇA
        INÍCIO
            PARA j ← 1 ATÉ 5 FAÇA
                INÍCIO
                    qtde ← 1
                    num ← mat[i,j]
                PARA k ← 1 ATÉ 15 FAÇA
                    INÍCIO
                        PARA l ← 1 ATÉ 5 FAÇA
                            INÍCIO
                                SE (i <> k) OU (j <>l)
                                ENTÃO SE mat[k,l] = num
                                    ENTÃO qtde ← qtde + 1
                            FIM
                        FIM
                    SE qtde > 1
                    ENTÃO INÍCIO
                        achou ← 0
                        SE (col = 1)
                        ENTÃO lin2 ← lin-1
                        SENÃO lin2 ← lin
                    PARA k ← 1 ATÉ lin2 FAÇA
                        INÍCIO
                            SE (k < lin2)
                            ENTÃO x ← 4
                            SENÃO x ← col-1
                        PARA l ← 1 ATÉ x FAÇA
                            INÍCIO
                                SE num = rep[k,l]
                                ENTÃO achou ← 1
                            FIM
                        FIM
                    SE achou = 0
                    ENTÃO INÍCIO
                        rep[lin,col] ← num
                        vezes[lin,col] ← qtde
                        col ← col + 1
                    SE col > 4
                    ENTÃO INÍCIO
                        lin ← lin + 1
                FIM
            FIM
        FIM
    FIM
FIM

```

```

        col ← 1
        FIM
        FIM
        FIM
    FIM
PARA i ← 1 ATÉ lin FAÇA
    INÍCIO
    SE i < lin
    ENTÃO x ← 4
    SENÃO x ← col-1
    PARA j ← 1 ATÉ x FAÇA
        INÍCIO
        ESCREVA "O número ",rep[i,j], " repetiu ",vezes[i,j],""
        ↪ vezes"
        FIM
    FIM
FIM
FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP6\PASCAL\EX9.PAS e \EXERC\CAP6\PASCAL\EX9.EXE

**SOLUÇÃO:**

\EXERC\CAP6\C++\EX9.CPP e \EXERC\CAP6\C++\EX9.EXE

- 10.** Faça um programa que carregue uma matriz  $10 \times 10$  com números inteiros, execute as trocas especificadas a seguir e mostre a matriz resultante.

- ◆ a linha 2 com a linha 8;
- ◆ a coluna 4 com a coluna 10;
- ◆ a diagonal principal com a diagonal secundária;
- ◆ a linha 5 com a coluna 10.

**SOLUÇÃO:**

```

ALGORITMO
DECLARE mat[10,10] NUMÉRICO
aux[10] NUMÉRICO
i, j NUMÉRICO
PARA i ← 1 ATÉ 10 FAÇA
    INÍCIO
    PARA j ← 1 ATÉ 10 FAÇA
        INÍCIO
        LEIA mat[i, j]
        FIM
    FIM
PARA j ← 1 ATÉ 10 FAÇA
    INÍCIO
    aux[j] ← mat[2, j]
    FIM
PARA j ← 1 ATÉ 10 FAÇA
    INÍCIO
    mat[2, j] ← mat[8, j]
    FIM
PARA j ← 1 ATÉ 10 FAÇA
    INÍCIO
    mat[8, j] ← aux[j]
    FIM

```

```

PARA i ← 1 ATÉ 10 FAÇA
    INÍCIO
    aux[i] ← mat[i, 4]
    FIM
PARA i ← 1 ATÉ 10 FAÇA
    INÍCIO
    mat[i, 4] ← mat[i, 10]
    FIM
PARA i ← 1 ATÉ 10 FAÇA
    INÍCIO
    mat[i, 10] ← aux[i]
    FIM
PARA i ← 1 ATÉ 10 FAÇA
    INÍCIO
    aux[i] ← mat[i, j]
    FIM
j ← 10
PARA i ← 1 ATÉ 10 FAÇA
    INÍCIO
    mat[i, i] ← mat[i, j]
    j ← j -1
    FIM
j ← 10
PARA i ← 1 ATÉ 10 FAÇA
    INÍCIO
    mat[i, j] ← aux[i]
    j ← j-1
    FIM
PARA j ← 1 ATÉ 10 FAÇA
    INÍCIO
    aux[j] ← mat[5, j]
    FIM
PARA j ← 1 ATÉ 10 FAÇA
    INÍCIO
    mat[5, j] ← mat[j, 10]
    FIM
PARA i ← 1 ATÉ 10 FAÇA
    INÍCIO
    mat[i, 10] ← aux[i]
    FIM
PARA i ← 1 ATÉ 10 FAÇA
    INÍCIO
    PARA j ← 1 ATÉ 10 FAÇA
        INÍCIO
        ESCREVA mat[i, j]
        FIM
    FIM
FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP6\PASCAL\EX10.PAS e \EXERC\CAP6\PASCAL\EX10.EXE

**SOLUÇÃO:**

\EXERC\CAP6\C++\EX10.CPP e \EXERC\CAP6\C++\EX10.EXE

11. Faça um programa que carregue uma matriz  $8 \times 8$  com números inteiros e mostre uma mensagem dizendo se a matriz digitada é simétrica. Uma matriz simétrica possui  $A[i, j] = A[j, i]$ .

**ALGORITMO****SOLUÇÃO:**

```

ALGORITMO
DECLARE mat[8,8] NUMÉRICO
      i, j NUMÉRICO
      achou LÓGICO
PARA i ← 1 ATÉ 8 FAÇA
    INÍCIO
    PARA j ← 1 ATÉ 8 FAÇA
      INÍCIO
      LEIA mat[i, j]
      FIM
    FIM
    achou ← falso
  PARA i ← 1 ATÉ 8 FAÇA
    INÍCIO
    PARA j ← 1 ATÉ 8 FAÇA
      INÍCIO
      SE mat[i, j] ≠ mat[j, i]
      ENTÃO achou ← verdadeiro
      FIM
    FIM
    SE achou = falso
    ENTÃO ESCREVA "Matriz Simétrica"
    SENÃO ESCREVA "Matriz Assimétrica"
  FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP6\PASCAL\EX11.PAS e \EXERC\CAP6\PASCAL\EX11.EXE

**SOLUÇÃO:**

\EXERC\CAP6\C++\EX11.CPP e \EXERC\CAP6\C++\EX11.EXE

-  12. Faça um programa que carregue uma matriz  $4 \times 4$  com números inteiros e verifique se essa matriz forma o chamado quadrado mágico. Um quadrado mágico é formado quando a soma dos elementos de cada linha é igual à soma dos elementos de cada coluna e igual à soma dos elementos da diagonal principal e igual, também, à soma dos elementos da diagonal secundária.

**SOLUÇÃO:**

```

ALGORITMO
DECLARE mat[4,4] NUMÉRICO
      soma_linha[4] NUMÉRICO
      soma_coluna[4] NUMÉRICO
      soma_diag, i, j, compara NUMÉRICO
      achou LÓGICO
PARA i ← 1 ATÉ 4 FAÇA
  INÍCIO
  PARA j ← 1 ATÉ 4 FAÇA
    INÍCIO
    LEIA mat[i, j]
    FIM
  FIM
  achou ← falso
  PARA i ← 1 ATÉ 4 FAÇA
    INÍCIO

```

```

PARA j ← 1 ATÉ 4 FAÇA
    INÍCIO
        soma_linha[i] ← soma_linha[i] + mat[i, j]
    FIM
FIM
compara ← soma_linha[1]
PARA i ← 2 ATÉ 4 FAÇA
    INÍCIO
        SE soma_linha[i] ≠ compara
        ENTÃO achou ← verdadeiro
    FIM
PARA j ← 1 ATÉ 4 FAÇA
    INÍCIO
        PARA i ← 1 ATÉ 4 FAÇA
            INÍCIO
                soma_coluna[j] ← soma_coluna[j] + mat[i, j]
            FIM
        FIM
        compara ← soma_coluna[1]
    PARA j ← 2 ATÉ 4 FAÇA
        INÍCIO
            SE soma_coluna[j] ≠ compara
            ENTÃO achou ← verdadeiro
        FIM
        SE soma_linha[1] ≠ soma_coluna[1]
        ENTÃO achou ← verdadeiro
        soma_diag ← 0
    PARA i ← 1 ATÉ 4 FAÇA
        INÍCIO
            soma_diag ← soma_diag + mat[i, i]
        FIM
        SE soma_diag ≠ soma_linha[1]
        ENTÃO achou ← verdadeiro
        soma_diag ← 0
        j ← 4
    PARA i ← 1 ATÉ 4 FAÇA
        INÍCIO
            soma_diag ← soma_diag + mat[i, j]
            j ← j-1
        FIM
        SE soma_diag ≠ soma_linha[1]
        ENTÃO achou ← verdadeiro
        SE achou = verdadeiro
        ENTÃO ESCREVA "Não forma quadrado mágico"
        SENÃO ESCREVA "Forma quadrado mágico"
    FIM_ALGORITMO.

```



**SOLUÇÃO:**

\EXERC\CAP6\PASCAL\EX12.PAS e \EXERC\CAP6\PASCAL\EX12.EXE



**SOLUÇÃO:**

\EXERC\CAP6\C++\EX12.CPP e \EXERC\CAP6\C++\EX12.EXE

### 13. Faça um programa que carregue:

- ◆ um vetor com os nomes de cinco produtos;
- ◆ uma matriz  $5 \times 4$  com os preços dos cinco produtos em quatro lojas diferentes;
- ◆ um outro vetor com o custo do transporte dos cinco produtos.

Calcule uma segunda matriz  $5 \times 4$  com os valores dos impostos de cada produto, sendo que esses obedecem à tabela a seguir.

PREÇO	% DE IMPOSTO
Até R\$ 50,00	5
Entre R\$ 50,01 e R\$ 100,00 (inclusive)	10
Acima de R\$ 100,00	20

Mostre:

- ◆ um relatório com o nome do produto, o número da loja onde o produto se encontra, o valor do imposto a pagar, o custo de transporte, o preço e o preço final (preço acrescido do valor do imposto e do custo do transporte).

#### ALGORITMO

#### SOLUÇÃO:

```

ALGORITMO
DECLARE nome[5] LITERAL
      preco, imp[5,4] NUMÉRICO
      custo[5] NUMÉRICO
      i, j, final NUMÉRICO
PARA i ← 1 ATÉ 5 FAÇA
    INÍCIO
    LEIA nome[i]
    FIM
PARA i ← 1 ATÉ 5 FAÇA
    INÍCIO
    PARA j ← 1 ATÉ 4 FAÇA
        INÍCIO
        LEIA preco[i,j]
        FIM
    FIM
PARA i ← 1 ATÉ 5 FAÇA
    INÍCIO
    LEIA custo[i]
    FIM
PARA i ← 1 ATÉ 5 FAÇA
    INÍCIO
    PARA j ← 1 ATÉ 4 FAÇA
        INÍCIO
        SE preco[i,j] ≤ 50
            ENTÃO imp[i,j] ← preco[i,j] * 5 / 100
        SENÃO SE (preco[i,j] > 50) E (preco[i,j] ≥ 100)
            ENTÃO imp[i,j] ← preco[i,j] * 10 / 100
        SENÃO imp[i,j] ← preco[i,j] * 15 / 100
        FIM
    FIM
PARA i ← 1 ATÉ 5 FAÇA
    INÍCIO
    ESCREVA "Nome do produto ", nome[i]
    ESCREVA "Custo = ", custo[i]
    PARA j ← 1 ATÉ 4 FAÇA
        INÍCIO
        final ← preco[i,j] + imp[i,j] + custo[i]
        ESCREVA "Imposto na loja ", j, " = ", imp[i,j]
        ESCREVA "Preço na loja ", j, " = ", preco[i,j]
        ESCREVA "Preço final na loja ", j, " = ", final
    FIM
FIM
FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP6\PASCAL\EX13.PAS e \EXERC\CAP6\PASCAL\EX13.EXE

**SOLUÇÃO:**

\EXERC\CAP6\C++\EX13.CPP e \EXERC\CAP6\C++\EX13.EXE

**14.** Faça um programa que receba:

- ◆ um vetor com o nome de cinco cidades diferentes;
- ◆ uma matriz  $5 \times 5$  com a distância entre as cidades, sendo que na diagonal principal, deve ser colocada automaticamente distância zero, ou seja, não deve ser permitida a digitação.

Calcule e mostre:

- ◆ os percursos que não ultrapassam 250 quilômetros (os percursos são compostos pelos nomes das cidades de origem e pelos nomes das cidades de destino);
- ◆ o consumo de um veículo, ou seja, quantos quilômetros o veículo faz por litro de combustível e mostre um relatório com a quantidade de combustível necessária para percorrer cada percurso citando o mesmo (nome da cidade de origem e nome da cidade de destino);
- ◆ a maior distância e em que percurso se encontra (nome da cidade de origem e nome da cidade de destino).

**SOLUÇÃO:**

```

ALGORITMO
DECLARE cidade[5] LITERAL
        distancia[5,5], i, j, consumo, qtde NUMÉRICO
PARA i ← 1 ATÉ 5 FAÇA
    INÍCIO
        LEIA cidade[i]
    FIM
PARA i ← 1 ATÉ 5 FAÇA
    INÍCIO
        PARA j ← 1 ATÉ 5 FAÇA
            INÍCIO
                SE (i = j)
                    ENTÃO distancia[i, j] ← 0
                SENÃO distancia[i, j]
            FIM
        FIM
    PARA i ← 1 ATÉ 5 FAÇA
        INÍCIO
            PARA j ← 1 ATÉ 5 FAÇA
                INÍCIO
                    SE (distancia[i, j] <= 250) E (distancia[i, j] >0)
                        ENTÃO ESCREVA "Distancia: ", distancia[i, j], "
                            entre ", cidade[i], " e ", cidade[j]
                FIM
            FIM
        LEIA consumo
    PARA i ← 1 ATÉ 5 FAÇA
        INÍCIO
            PARA j ← 1 ATÉ 5 FAÇA
                INÍCIO

```

```

        SE (i ≠ j)
        ENTÃO INÍCIO
            qtde ← distancia[i, j] / consumo
            ESCREVA "Consumo entre ", cidade[i], " e ",
            • cidade[j], " = ", qtde
        FIM
    FIM
FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP6\PASCAL\EX14.PAS e \EXERC\CAP6\PASCAL\EX14.EXE

**SOLUÇÃO:**

\EXERC\CAP6\C++\EX14.CPP e \EXERC\CAP6\C++\EX14.EXE

**15.** Faça um programa que carregue:

- ◆ um vetor com cinco números inteiros;
- ◆ um outro vetor com dez números inteiros;
- ◆ uma matriz  $4 \times 3$ , também com números inteiros.

Calcule e mostre:

- ◆ o maior elemento do primeiro vetor multiplicado pelo menor elemento do segundo vetor. O resultado dessa multiplicação adicionado aos elementos digitados na matriz dará origem a uma segunda matriz (resultante);
- ◆ a soma dos elementos pares de cada linha da matriz resultante;
- ◆ a quantidade de elementos entre 1 e 5 em cada coluna da matriz resultante.

**SOLUÇÃO:**

```

ALGORITMO
DECLARE vet1[5], vet2[10], mat[4,3], mat_result[4,3] NUMÉRICO
        i, j, maior, menor, mult, soma, qtde NUMÉRICO
PARA i ← 1 ATÉ 5 FAÇA
    INÍCIO
        LEIA vet1[i]
    FIM
PARA i ← 1 ATÉ 10 FAÇA
    INÍCIO
        LEIA vet2[i]
    FIM
PARA i ← 1 ATÉ 4 FAÇA
    INÍCIO
        PARA j ← 1 ATÉ 3 FAÇA
            INÍCIO
                LEIA mat[i,j]
            FIM
        FIM
    FIM
    maior ← vet1[1]
    PARA i ← 1 ATÉ 5 FAÇA
        INÍCIO
            SE vet1[i] > maior
            ENTÃO maior ← vet1[i]
        FIM
    menor ← vet2[1]
    PARA i ← 1 ATÉ 10 FAÇA

```

```

INÍCIO
SE vet2[i] < menor
ENTÃO menor ← vet2[i]
FIM
mult ← maior * menor
PARA i ← 1 ATÉ 4 FAÇA
    INÍCIO
        PARA j ← 1 ATÉ 3 FAÇA
            INÍCIO
                mat_result[i, j] ← mat[i, j] + mult
            FIM
        FIM
    ESCREVA "Matriz resultante"
    PARA i ← 1 ATÉ 4 FAÇA
        INÍCIO
            PARA j ← 1 ATÉ 3 FAÇA
                INÍCIO
                    ESCREVA mat_result[i, j]
                FIM
            FIM
        PARA i ← 1 ATÉ 4 FAÇA
            INÍCIO
                soma ← 0
                PARA j ← 1 ATÉ 3 FAÇA
                    INÍCIO
                        SE RESTO(mat_result[i, j]/2) = 0
                            ENTÃO soma ← soma + mat_result[i, j]
                    FIM
                ESCREVA "Soma dos elementos pares da linha ", i, " da matriz
                → resultante = ", soma
                FIM
            PARA j ← 1 ATÉ 3 FAÇA
                INÍCIO
                    qtdc ← 0
                    PARA i ← 1 ATÉ 4 FAÇA
                        INÍCIO
                            SE (mat_result[i, j] > 1) E (mat_result[i, j] < 5)
                                ENTÃO qtdc ← qtdc + 1
                            FIM
                        ESCREVA "Quantidade de números entre 1 e 5 na coluna
                        → ", j, " da matriz resultante = ", qtdc
                    FIM
                FIM_ALGORITMO.

```



## SOLUÇÃO:

\EXERC\CAP6\PASCAL\EX15.PAS e \EXERC\CAP6\PASCAL\EX15.EXE



## SOLUÇÃO:

\EXERC\CAP6\C++\EX15.CPP e \EXERC\CAP6\C++\EX15.EXE

- 16.** Faça um programa que carregue uma matriz  $7 \times 7$  de números inteiros e crie dois vetores de sete posições cada um e que contenham, respectivamente, o maior elemento de cada uma das linhas e o menor elemento de cada uma das colunas. Escreva a matriz A e os dois vetores.

**SOLUÇÃO:**

```

ALGORITMO
DECLARE mat[7,7], vet1[7], vet2[7], i, j, maior, menor NUMÉRICO
PARA i ← 1 ATÉ 7 FAÇA
    INÍCIO
    PARA j ← 1 ATÉ 7 FAÇA
        INÍCIO
        LEIA mat[i, j]
        FIM
    FIM
    PARA i ← 1 ATÉ 7 FAÇA
        INÍCIO
        maior ← mat[i,1]
        PARA j ← 2 ATÉ 7 FAÇA
            INÍCIO
            SE (mat[i, j] > maior)
                ENTÃO maior ← mat[i, j]
            FIM
        vet1[i] ← maior
        FIM
    PARA i ← 1 ATÉ 7 FAÇA
        INÍCIO
        menor ← mat[1,i]
        PARA j ← 2 ATÉ 7 FAÇA
            INÍCIO
            SE (mat[j, i] < menor)
                ENTÃO menor ← mat[j, i]
            FIM
        vet2[i] ← menor
        FIM
    PARA i ← 1 ATÉ 7 FAÇA
        INÍCIO
        PARA j ← 1 ATÉ 7 FAÇA
            INÍCIO
            ESCREVA mat[i, j]
            FIM
        FIM
    PARA i ← 1 ATÉ 7 FAÇA
        INÍCIO
        ESCREVA vet1[i]
        FIM
    PARA i ← 1 ATÉ 7 FAÇA
        INÍCIO
        ESCREVA vet2[i]
        FIM
    FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP6\PASCAL\EX16.PAS e \EXERC\CAP6\PASCAL\EX16.EXE

**SOLUÇÃO:**

\EXERC\CAP6\C++\EX16.CPP e \EXERC\CAP6\C++\EX16.EXE

17. Faça um programa que utilize uma matriz  $5 \times 5$  que aceite três tipos de valores: múltiplos de 5, múltiplos de 11 e múltiplos de 13. Devem ser lidos apenas valores maiores que zero. Após a leitura, os números devem ser distribuídos da seguinte maneira:

- ◆ os múltiplos de 5 devem ocupar a diagonal principal;
- ◆ os múltiplos de 11 devem ficar acima da diagonal principal;
- ◆ os múltiplos de 13 devem ficar abaixo da diagonal principal.

Como alguns números podem ser múltiplos de 5, de 11 e também de 13 (por exemplo, 55 é múltiplo de 5 e de 11; 65 é múltiplo de 5 e de 13), deve-se, primeiro, verificar se o número digitado é múltiplo de 5. Caso não seja, deve-se verificar se é múltiplo de 11. Caso não seja, deve-se verificar se é múltiplo de 13. Caso não seja, mostrar a mensagem *Número inválido* (por exemplo, o número 55 deverá ser considerado múltiplo de 5, pois essa é a comparação que será feita primeiro).

A seguir pode ser observado um exemplo:

5	44	11	33	55
26	15	77	99	88
39	13	10	121	22
52	78	65	40	132
91	117	104	143	25

Esse programa deve observar as seguintes situações:

- ◆ quando o usuário digitar um múltiplo de 5 e não houver mais espaço na diagonal principal, mostre a mensagem *Diagonal totalmente preenchida*;
- ◆ quando o usuário digitar um múltiplo de 11 e não houver mais espaço disponível na matriz, mostre a mensagem *Não existe espaço acima da diagonal principal*;
- ◆ quando o usuário digitar um múltiplo de 13 e não houver mais espaço disponível na matriz, mostre a mensagem *Não existe espaço abaixo da diagonal principal*;
- ◆ quando a matriz estiver totalmente preenchida, mostre todos os elementos da matriz, juntamente com suas posições (linha e coluna).

## ALGORITMO

### SOLUÇÃO:

```

ALGORITMO
DECLARE mat[5,.5] NUMÉRICO
      i, j, dp, lin_ac, col_ac, lin_ab NUMÉRICO
      col_ab, num, r, cont_dp, cont_ac, cont_ab NUMÉRICO
      dp ← 1
      lin_ac ← 1
      col_ac ← 2
      lin_ab ← 2
      col_ab ← 1
      cont_dp ← 0
      cont_ac ← 0 cont_ab ← 0
ENQUANTO ((cont_ac < 10) OU (cont_ab < 10) OU (cont_dp < 5)) FAÇA
    INÍCIO
      LEIA num
      r ← RESTO (num/5)
      SE r = 0
        ENTÃO INÍCIO
          SE cont_dp < 5
            ENTÃO INÍCIO
              mat[dp,dp] ← num
              dp ← dp + 1
              cont_dp ← cont_dp + 1
            FIM
          FIM
        FIM
      FIM
    FIM
  FIM
FIM

```

```

        FIM
SENÃO ESCREVA "Não existe mais espaço para múltiplos
    de 5"
        FIM
SENÃO INÍCIO
    r ← RESTO (num/11)
    SE r = 0
        ENTÃO INÍCIO
            SE cont_ac < 10
                ENTÃO INÍCIO
                    mat[lin_ac,col_ac] ← num
                    col_ac ← col_ac + 1
                    SE col_ac > 5
                        ENTÃO INÍCIO
                            lin_ac ← lin_ac + 1
                            col_ac ← lin_ac + 1
                        FIM
                    cont_ac ← cont_ac + 1
                FIM
            SENÃO ESCREVA "Não existe mais espaço para múltiplos
                de 11"
            FIM
SENÃO INÍCIO
    r ← RESTO (num/13)
    SE r = 0
        ENTÃO INÍCIO
            SE cont_ab < 10
                ENTÃO INÍCIO
                    mat[lin_ab,col_ab] ← num
                    col_ab ← col_ab + 1
                    SE col_ab >= lin_ab
                        ENTÃO INÍCIO
                            lin_ab ← lin_ab + 1
                            col_ab ← 1
                        FIM
                    cont_ab ← cont_ab + 1
                FIM
            SENÃO ESCREVA "Não existe mais espaço para
                múltiplos de 13"
            FIM
SENÃO ESCREVA "Digite um número múltiplo de 5 ou de 11 ou
    de 13"
    FIM
    FIM
PARA i ← 1 ATÉ 5 FAÇA
INÍCIO
PARA j ← 1 ATÉ 5 FAÇA
INÍCIO
    ESCREVA mat[i,j]
FIM
FIM
FIM_ALGORITMO.

```


**SOLUÇÃO:**

\EXERC\CAP6\PASCAL\EX17.PAS e \EXERC\CAP6\PASCAL\EX17.EXE


**SOLUÇÃO:**

\EXERC\CAP6\C++\EX17.CPP e \EXERC\CAP6\C++\EX17.EXE

**18.** Faça um programa que leia um vetor V contendo 18 elementos. A seguir, distribua esses elementos em uma matriz  $3 \times 6$ . Ao final, mostre a matriz gerada.

Veja a seguir um exemplo do que o seu programa deve fazer:

V	3	25	1	58	97	43	65	32	27	19	10	6	88	13	34	57	89	87
M		3	25	1	58	97	43											
	65		32	27	19	10	6											
	88		13	34	57	89	87											

**ALGORITMO****SOLUÇÃO:**

```

ALGORITMO
DECLARE vet [18], mat [3,6] NUMÉRICO
      i, j, lin, col NUMÉRICO
PARA i ← 1 ATÉ 18 FAÇA
    INÍCIO
    LEIA vet[i]
    FIM
lin ← 1
col ← 1
PARA i ← 1 ATÉ 18 FAÇA
    INÍCIO
    mat[lin,col] ← vet[i]
    col ← col + 1
    SE col > 6
    ENTÃO INÍCIO
        col ← 1
        lin ← lin + 1
    FIM
    FIM
PARA i ← 1 ATÉ 3 FAÇA
    INÍCIO
    PARA j ← 1 ATÉ 6 FAÇA
        INÍCIO
        ESCREVA "Elemento ", i , " - ", j , mat[i,j]
        FIM
    FIM
FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP6\PASCAL\EX18.PAS e \EXERC\CAP6\PASCAL\EX18.EXE

**SOLUÇÃO:**

\EXERC\CAP6\C++\EX18.CPP e \EXERC\CAP6\C++\EX18.EXE

**19.** Faça um programa que utilize uma matriz com dimensões máximas de cinco linhas e quatro colunas.

Solicite que sejam digitados os números que serão armazenados na matriz da seguinte maneira:

- ◆ se o número digitado for par, deve ser armazenado em uma linha de índice par;
- ◆ se o número digitado for ímpar, deve ser armazenado em uma linha de índice ímpar;

- ◆ as linhas devem ser preenchidas de cima para baixo (por exemplo, os números pares digitados devem ser armazenados inicialmente na primeira linha par; quando essa linha for totalmente preenchida, deve ser utilizada a segunda linha par e assim sucessivamente. O mesmo procedimento deve ser adotado para os números ímpares);
- ◆ quando não couberem mais números pares ou ímpares, deve ser mostrada uma mensagem ao usuário;
- ◆ quando a matriz estiver totalmente preenchida, deve-se encerrar a leitura dos números e mostrar todos os elementos armazenados na matriz.

**ALGORITMO****SOLUÇÃO:**

```

ALGORITMO
DECLARE mat[5,4] NUMÉRICO
    i, j, num, r, lin_p, col_p, lin_i, col_i, tot NUMÉRICO
    lin_p ← 2
    col_p ← 1
    lin_i ← 1
    col_i ← 1
    tot ← 0
REPITA
SE tot ≠ 20
ENTÃO INÍCIO
    LEIA num
    r ← RESTO (num/2)
    SE r = 0
        ENTÃO INÍCIO
            SE lin_p > 4
                ENTÃO ESCREVA "Sem espaço para números pares"
            SENÃO INÍCIO
                mat[lin_p,col_p] ← num
                tot ← tot + 1
                col_p ← col_p + 1
                SE col_p > 4
                    ENTÃO INÍCIO
                        lin_p ← lin_p + 2
                        col_p ← 1
                    FIM
                FIM
            FIM
        FIM
    SENÃO INÍCIO
        SE lin_i > 5
            ENTÃO ESCREVA "Sem espaço para números ímpares"
        SENÃO INÍCIO
            mat[lin_i,col_i] ← num
            tot ← tot + 1
            col_i ← col_i + 1
            SE col_i > 4
                ENTÃO INÍCIO
                    lin_i ← lin_i + 2
                    col_i ← 1
                FIM
            FIM
        FIM
    FIM
ATÉ tot = 20
ESCREVA "Matriz totalmente preenchida"
PARA i ← 1 ATÉ 5 FAÇA
    INÍCIO
        PARA j ← 1 ATÉ 4 FAÇA
            INÍCIO

```

```

    ESCREVA mat[i,j]
    FIM
    FIM
FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP6\PASCAL\EX19.PAS e \EXERC\CAP6\PASCAL\EX19.EXE

**SOLUÇÃO:**

\EXERC\CAP6\C++\EX19.CPP e \EXERC\CAP6\C++\EX19.EXE

- 20.** Faça um programa que utilize uma matriz com dimensões máximas de cinco linhas e quatro colunas e solicite que sejam digitados os números (desordenadamente) e armazene-os ordenadamente na matriz.

Observe o exemplo: supondo que sejam digitados os seguintes números: 10-1-2-20-30-17-98-65-24-12-5-8-73-55-31-100-120-110-114-130, estes deverão ser armazenados na matriz da seguinte maneira:

1	2	5	8	10
12	17	20	24	30
31	55	65	73	98
100	110	114	120	130

**SOLUÇÃO:**

```

ALGORITMO
DECLARE num[5,4] NUMÉRICO
    num_aux, i, j, k, l, m, n, lin, col NUMÉRICO
PARA i ← 1 ATÉ 5 FAÇA
    INÍCIO
    PARA j ← 1 ATÉ 4 FAÇA
        INÍCIO
        LEIA num_aux
        SE (i = 1) E (j = 1)
        ENTÃO num[i,j] ← num_aux
        SENÃO INÍCIO
            k ← 1
            l ← 1
            ENQUANTO ((num[k,l] < num_aux) E
            ((k <> i) OU (l <> j))) FAÇA
                INÍCIO
                l ← l + 1
                SE l > 4
                ENTÃO INÍCIO
                    k ← k + 1
                    l ← 1
                FIM
            FIM
            m ← i
            n ← j
            ENQUANTO ((m <> k) OU (n <> l)) FAÇA

```

```

INÍCIO
SE n-1 ≤ 0
ENTÃO INÍCIO
    lin ← m-1
    col ← 4
    FIM
SENÃO INÍCIO
    lin ← m
    col ← n-1
    FIM
    num[m,n] ← num[lin,col]
    n ← n-1
    SE n<=0
        ENTÃO INÍCIO
            n ← 4
            m ← m-1
            FIM
        FIM
        num[k,1] ← num_aux
    FIM
    FIM
PARA i ← 1 ATÉ 5 FAÇA
    INÍCIO
        PARA j ← 1 ATÉ 4 FAÇA
            INÍCIO
                ESCREVA "Elemento da posição ", i, " - ", j , " = " ,
                num[i,j]
            FIM
        FIM
    FIM_ALGORITMO.

```



#### SOLUÇÃO:

\EXERC\CAP6\PASCAL\EX20.PAS e \EXERC\CAP6\PASCAL\EX20.EXE



#### SOLUÇÃO:

\EXERC\CAP6\C++\EX20.CPP e \EXERC\CAP6\C++\EX20.EXE

- 21.** Faça um programa que utilize uma matriz com as dimensões fornecidas pelo usuário e execute as solicitações a seguir.

Para a realização dessa tarefa, a matriz deve ser obrigatoriamente quadrada (número igual de linhas e colunas). Sendo assim, solicite que seja informada a dimensão da matriz.

Posteriormente, realize a leitura dos elementos que vão compor a matriz.

Finalmente, some e mostre os elementos que estão abaixo da diagonal secundária.

Exemplo:

Supondo a matriz a seguir, o resultado do problema seria: 98 + 32 + 87 + 36 + 65 + 25

20	10	1	8
17	42	11	98
19	45	32	87
12	36	65	25

**ALGORITMO****SOLUÇÃO:**

```

ALGORITMO
DECLARE num[100,100] NUMÉRICO
        dim, l, c, soma, cont NUMÉRICO
REPITA
    ESCREVA "Digite a dimensão da matriz (quadrada) - no máximo
    ↪ 100"
    LEIA dim
ATÉ (dim >= 1) E (dim <= 100)
l ← 1
c ← 1
ENQUANTO l ≤ dim FAÇA
INÍCIO
    ENQUANTO c ≤ dim FAÇA
    INÍCIO
        LEIA num[l,c]
        c ← c + 1
        FIM
    l ← l + 1
    c ← 1
    FIM
    soma ← 0
    cont ← 0
    l ← 2
    c ← dim
    ENQUANTO l ≤ dim FAÇA
    INÍCIO
        ENQUANTO c ≥ dim-cont FAÇA
        INÍCIO
            ESCREVA num[l,c]
            soma ← soma + num[l,c]
            c ← c - 1
            FIM
        cont ← cont + 1
        l ← l + 1
        c ← dim
        FIM
        ESCREVA soma
    FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP6\PASCAL\EX21.PAS e \EXERC\CAP6\PASCAL\EX21.EXE

**SOLUÇÃO:**

\EXERC\CAP6\C++\EX21.CPP e \EXERC\CAP6\C++\EX21.EXE

**22.** Faça um programa que receba o estoque atual de três produtos que estão armazenados em quatro armazéns e coloque esses dados em uma matriz  $5 \times 3$ . Sendo que a última linha da matriz contém o custo de cada produto, calcule e mostre:

- ◆ a quantidade de itens armazenados em cada armazém;
- ◆ qual o armazém possui maior estoque do produto 2;
- ◆ qual o armazém possui menor estoque;
- ◆ qual o custo total de cada produto;
- ◆ qual o custo total de cada armazém.

Desconsiderar empates.

**ALGORITMO****SOLUÇÃO:**

```

ALGORITMO
DECLARE prod[5,3] NUMÉRICO
        tot_arm, maior_e, custo_p, custo_a, i, j, ind_a NUMÉRICO
PARA i ← 1 ATÉ 4 FAÇA
INÍCIO
PARA j ← 1 ATÉ 3 FAÇA
INÍCIO
LEIA prod[i,j]
FIM
FIM
PARA i ← 1 ATÉ 3 FAÇA
INÍCIO
LEIA prod[5,i]
FIM
PARA i ← 1 ATÉ 4 FAÇA
INÍCIO
tot_arm ← 0
PARA j ← 1 ATÉ 3 FAÇA
INÍCIO
tot_arm ← tot_arm + prod[i,j]
FIM
ESCREVA "O total de itens no armazém ",i, " = ",tot_arm
FIM
PARA i ← 1 ATÉ 4 FAÇA
INÍCIO
SE i = 1
ENTÃO INÍCIO
    maior_e ← prod[i,1]
    ind_a ← i
    FIM
SENÃO INÍCIO
    SE prod[i,1] > maior_e
    ENTÃO INÍCIO
        maior_e ← prod[i,1]
        ind_a ← i
        FIM
    FIM
    FIM
ESCREVA "O maior estoque do produto 2 está no armazém" ,ind_a
PARA j ← 1 ATÉ 3 FAÇA
INÍCIO
custo_p ← 0
PARA i ← 1 ATÉ 4 FAÇA
INÍCIO
custo_p ← custo_p + prod[i,j]
FIM
custo_p ← custo_p * prod[5,j]
ESCREVA "O custo total do produto ", j , " = ",custo_p
FIM
PARA i ← 1 ATÉ 4 FAÇA
INÍCIO
custo_a ← 0
PARA j ← 1 ATÉ 3 FAÇA
INÍCIO
    custo_a ← custo_a + (prod[i,j] * prod[5,j])
FIM
ESCREVA "O custo total do armazém ", i , " = ", custo_a
FIM
FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP6\PASCAL\EX22.PAS e \EXERC\CAP6\PASCAL\EX22.EXE

**SOLUÇÃO:**

\EXERC\CAP6\C++\EX22.CPP e \EXERC\CAP6\C++\EX22.EXE

- 23.** Faça um programa que receba as vendas semanais (de um mês) de cinco vendedores de uma loja e armazene essas vendas em uma matriz. Calcule e mostre:

- ◆ o total de vendas do mês de cada vendedor;
- ◆ o total de vendas de cada semana (todos os vendedores juntos);
- ◆ o total de vendas do mês.

**SOLUÇÃO:**

```

ALGORITMO
DECLARE vendas[4, 5], tot_ven, tot_sem, tot_geral, i, j NUMÉRICO
PARA i ← 1 ATÉ 4 FAÇA
    INÍCIO
        PARA j ← 1 ATÉ 5 FAÇA
            INÍCIO
                LEIA vendas[i, j]
                FIM
            FIM
        PARA i ← 1 ATÉ 5 FAÇA
            INÍCIO
                tot_ven ← 0
                PARA j ← 1 ATÉ 4 FAÇA
                    INÍCIO
                        tot_ven ← tot_ven + vendas[j, i]
                    FIM
                ESCREVA tot_ven
                FIM
            PARA i ← 1 ATÉ 4 FAÇA
                INÍCIO
                    tot_sem ← 0
                    PARA j ← 1 ATÉ 5 FAÇA
                        INÍCIO
                            tot_sem ← tot_sem + vendas[i, j]
                        FIM
                    ESCREVA tot_sem
                    FIM
                tot_geral ← 0
                PARA i ← 1 ATÉ 4 FAÇA
                    INÍCIO
                        PARA j ← 1 ATÉ 5 FAÇA
                            INÍCIO
                                tot_geral ← tot_geral + vendas[i, j]
                            FIM
                        ESCREVA tot_geral
                    FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP6\PASCAL\EX23.PAS e \EXERC\CAP6\PASCAL\EX23.EXE

**SOLUÇÃO:**

\EXERC\CAP6\C++\EX23.CPP e \EXERC\CAP6\C++\EX23.EXE

**24.** Faça um programa que:

- ◆ receba dez nomes de produtos e armazene-os em um vetor;
- ◆ receba o estoque desses dez produtos em cada um dos cinco armazéns (matriz  $5 \times 10$ );
- ◆ receba o custo dos dez produtos e armazene-os em um outro vetor;

Calcule e mostre:

- ◆ o total de itens armazenados em cada armazém;
- ◆ o total de itens armazenados de cada produto;
- ◆ o custo total de cada armazém;
- ◆ o nome do produto e o número do armazém que possui maior número de itens estocados;
- ◆ o nome do produto que possui menor custo.

**ALGORITMO****SOLUÇÃO:**

```

ALGORITMO
DECLARE prod[10], nome LITERAL
        estoque[10,5], i, j, ind_a, ind_p NUMÉRICO
        tot_a, tot_p, maior_e, custo[10], custo_a, menor_c NUMÉRICO
PARA i ← 1 ATÉ 10 FAÇA
    INÍCIO
        LEIA prod[i]
    FIM
PARA i ← 1 ATÉ 10 FAÇA
    INÍCIO
        LEIA custo[i]
    FIM
PARA i ← 1 ATÉ 10 FAÇA
    INÍCIO
        PARA j ← 1 ATÉ 5 FAÇA
            INÍCIO
                LEIA (estoque[i, j])
            FIM
        FIM
    PARA i ← 1 ATÉ 5 FAÇA
        INÍCIO
            tot_a ← 0
            custo_a ← 0
            PARA j ← 1 ATÉ 10 FAÇA
                INÍCIO
                    tot_a ← tot_a + estoque[j, i]
                    custo_a ← custo_a + estoque[j, i] * custo[j]
                FIM
            ESCREVA tot_a, custo_a
        FIM
    PARA i ← 1 ATÉ 10 FAÇA
        INÍCIO
            tot_p ← 0
            PARA j ← 1 ATÉ 5 FAÇA
                INÍCIO
                    tot_p ← tot_p + estoque[i, j]
                FIM
            ESCREVA tot_p.

```

```

        FIM
PARA i ← 1 ATÉ 10 FAÇA
    INÍCIO
        PARA j ← 1 ATÉ 5 FAÇA
            INÍCIO
                SE (i = 1) E (j = 1)
                ENTÃO INÍCIO
                    maior_e ← estoque[i, j]
                    ind_a ← j
                    ind_p ← i
                    FIM
                SENÃO INÍCIO
                    SE (estoque[i, j] > maior_e)
                    ENTÃO INÍCIO
                        maior_e ← estoque[i, j]
                        ind_a ← j
                        ind_p ← i
                        FIM
                FIM
            FIM
        FIM
    FIM
ESCREVA prod[ind_p], ind_a
PARA i ← 1 ATÉ 10 FAÇA
    INÍCIO
        SE (i = 1)
        ENTÃO INÍCIO
            menor_c ← custo[i]
            ind_p ← i
        FIM
    SENÃO SE (custo[i] < menor_c)
    ENTÃO INÍCIO
        menor_c ← custo[i]
        ind_p ← i
    FIM
    FIM
    ESCREVA prod[ind_p]
FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP6\PASCAL\EX24.PAS e \EXERC\CAP6\PASCAL\EX24.EXE

**SOLUÇÃO:**

\EXERC\CAP6\C++\EX24.CPP e \EXERC\CAP6\C++\EX24.EXE

**25.** Faça um programa que receba as vendas de cinco produtos em três lojas diferentes e em dois meses consecutivos. Armazene essas vendas em duas matrizes  $5 \times 3$ . O bimestre é uma matriz  $5 \times 3$ , resultado da soma das duas matrizes anteriores. Calcule e mostre:

- ◆ as vendas de cada produto em cada loja no bimestre;
- ◆ a maior venda do bimestre;
- ◆ o total vendido por loja no bimestre;
- ◆ o total vendido de cada produto no bimestre.

**SOLUÇÃO:**

```

ALGORITMO
DECLARE mes1[5,3], mes2[5,3], bim[5,3] NUMÉRICO
i, j, tot_prod, tot_loja, maior NUMÉRICO

```

```

PARA i ← 1 ATÉ 5 FAÇA
    INÍCIO
        PARA j ← 1 ATÉ 3 FAÇA
            INÍCIO
                LEIA mes1[i, j]
            FIM
        FIM
    PARA i ← 1 ATÉ 5 FAÇA
        INÍCIO
            PARA j ← 1 ATÉ 3 FAÇA
                INÍCIO
                    LEIA mes2[i, j]
                FIM
            FIM
        FIM
    PARA i ← 1 ATÉ 5 FAÇA
        INÍCIO
            PARA j ← 1 ATÉ 3 FAÇA
                INÍCIO
                    bim[i, i] ← mes1[i, j] + mes2[i, j]
                    ESCREVA bim[i, j]
                    SE (i=1) E (j=1)
                        ENTÃO maior ← bim[i, j]
                    SENÃO SE (bim[i, j] > maior)
                        ENTÃO maior ← bim[i, j]
                FIM
            FIM
        FIM
        ESCREVA maior
    PARA i ← 1 ATÉ 3 FAÇA
        INÍCIO
            tot_loja ← 0
            PARA j ← 1 ATÉ 5 FAÇA
                INÍCIO
                    tot_loja ← tot_loja + bim[j, i]
                FIM
            FIM
            ESCREVA tot_loja
        FIM
    PARA i ← 1 ATÉ 5 FAÇA
        INÍCIO
            tot_prod ← 0
            PARA j ← 1 ATÉ 3 FAÇA
                INÍCIO
                    tot_prod ← tot_prod + bim[i, j]
                FIM
            FIM
            ESCREVA tot_prod
        FIM
    FIM_ALGORITMO.

```



#### SOLUÇÃO:

\EXERC\CAP6\PASCAL\EX25.PAS e \EXERC\CAP6\PASCAL\EX25.EXE



#### SOLUÇÃO:

\EXERC\CAP6\C++\EX25.CPP e \EXERC\CAP6\C++\EX25.EXE

## EXERCÍCIOS PROPOSTOS

1. Faça um programa que carregue uma matriz  $3 \times 5$  com números inteiros, calcule e mostre a quantidade de elementos entre 15 e 20.
2. Faça um programa que carregue uma matriz  $2 \times 4$  com números inteiros, calcule e mostre:
  - ◆ a quantidade de elementos entre 12 e 20 em cada linha;
  - ◆ a média dos elementos pares da matriz.
3. Faça um programa que carregue uma matriz  $6 \times 3$ , calcule e mostre:
  - ◆ o maior elemento da matriz e sua respectiva posição, ou seja, linha e coluna;
  - ◆ o menor elemento da matriz e sua respectiva posição, ou seja, linha e coluna.
4. Faça um programa que receba:
  - ◆ as notas de 15 alunos em cinco provas diferentes e armazene-as em uma matriz  $15 \times 5$ ;
  - ◆ os nomes dos 15 alunos e armazene-os em um vetor de 15 posições.

Calcule e mostre:

  - ◆ para cada aluno, o nome, a média aritmética das cinco provas e a situação (Aprovado, Reprovado ou Exame);
  - ◆ a média da classe.
5. Faça um programa que carregue uma matriz  $12 \times 4$  com os valores das vendas de uma loja, onde cada linha representa um mês do ano e cada coluna representa uma semana do mês. Calcule e mostre:
  - ◆ o total vendido em cada mês do ano, mostrando o nome do mês por extenso;
  - ◆ o total vendido em cada semana durante todo o ano;
  - ◆ o total vendido pela loja no ano.
6. Faça um programa que carregue uma matriz  $20 \times 10$  com números inteiros e some cada uma das colunas, armazenando o resultado da soma em um vetor. A seguir, multiplique cada elemento da matriz pela soma da coluna e mostre a matriz resultante.
7. Faça um programa que carregue uma matriz M de ordem  $4 \times 6$  e uma segunda matriz N de ordem  $6 \times 4$ , calcule e imprima a soma das linhas de M com as colunas de N.
8. Faça um programa que carregue duas matrizes  $3 \times 8$  com números inteiros, calcule e mostre:
  - ◆ a soma das duas matrizes, resultando em uma terceira matriz também de ordem  $3 \times 8$ ;
  - ◆ a diferença das duas matrizes, resultando em uma quarta matriz também de ordem  $3 \times 8$ .
9. Faça um programa que carregue uma matriz  $3 \times 3$  com números reais e receba um valor, número digitado pelo usuário, calcule e mostre a matriz resultante da multiplicação do número digitado por elemento da matriz.
10. Faça um programa que carregue uma matriz  $5 \times 5$  com números inteiros, calcule e mostre a soma:

- ◆ dos elementos da linha 4;
- ◆ dos elementos da coluna 2;
- ◆ dos elementos da diagonal principal;
- ◆ dos elementos da diagonal secundária;
- ◆ de todos os elementos da matriz.

**11.** Faça um programa que:

- ◆ receba a idade de oito alunos e armazene-as em um vetor, em um outro vetor armazene o código de cinco disciplinas e em uma matriz armazene a quantidade de provas que cada aluno fez em cada disciplina.

Calcule e mostre:

- a) a quantidade de alunos com idade entre 18 e 25 anos e que fizeram mais de duas provas em uma disciplina com código digitado pelo usuário. O usuário pode digitar um código que não está cadastrado; nesse caso, mostrar mensagem.
- b) uma listagem com o número do aluno e o código da disciplina dos alunos que fizeram menos de três provas. Analisar cada disciplina.
- c) a média de idade dos alunos que não fizeram nenhuma prova em alguma disciplina. Cuidado para não contar duas vezes o mesmo aluno.

**12.** Faça um programa que carregue uma matriz  $6 \times 4$ . Recalcule a matriz digitada, onde cada linha será multiplicada pelo maior elemento da linha em questão. Mostre a matriz resultante.

**13.** Faça um programa que carregue uma matriz  $2 \times 3$ , calcule e mostre a quantidade de elementos da matriz que não pertencem ao intervalo  $[5,15]$ .

**14.** Faça um programa que carregue uma matriz  $12 \times 13$  e divida todos os elementos de cada linha pelo maior elemento em módulo daquela linha. Escreva a matriz lida e a modificada.

**15.** Faça um programa que carregue uma matriz  $5 \times 5$  e crie dois vetores de cinco posições cada um, que contenham, respectivamente, as somas das linhas e das colunas da matriz. Escreva a matriz e os vetores criados.

**16.** Faça um programa que calcule e mostre a média dos elementos da diagonal principal de uma matriz  $10 \times 10$ .

**17.** Faça um programa que carregue uma matriz  $5 \times 5$  de números reais, calcule e mostre a soma dos elementos da diagonal secundária.

**18.** Faça um programa que carregue uma matriz  $8 \times 6$  de inteiros, calcule e mostre a média dos elementos das linhas pares da matriz.

**19.** Faça um programa que carregue uma matriz  $5 \times 5$  com números reais e encontre o maior valor da matriz. A seguir, multiplique cada elemento da diagonal principal pelo maior valor encontrado. Mostre a matriz resultante após as multiplicações.

**20.** Faça um programa que carregue uma matriz  $5 \times 5$  de números reais. A seguir, multiplique cada linha pelo elemento da diagonal principal daquela linha. Mostre a matriz após as multiplicações.

**21.** Faça um programa que carregue uma matriz  $6 \times 10$ , some as colunas individualmente e acumule as somas na 7<sup>a</sup> linha da matriz. Mostre o resultado de cada coluna.

**22.** Faça um programa que carregue uma matriz  $3 \times 4$ , calcule e mostre:

- ◆ a quantidade de elementos pares;
- ◆ a soma dos elementos ímpares;
- ◆ a média de todos os elementos.

**23.** Faça um programa que carregue uma matriz  $4 \times 5$ , calcule e mostre um vetor com cinco posições, onde cada posição contém a soma dos elementos de cada coluna da matriz. Mostre apenas os elementos do vetor maiores que dez. Se não existir nenhum elemento maior que dez mostre uma mensagem.

**24.** Faça um programa que:

- ◆ receba o preço de dez produtos e armazene-os em um vetor;
- ◆ receba a quantidade estocada de cada um desses produtos em cinco armazéns diferentes, utilizando uma matriz  $5 \times 10$ .

Calcule e mostre:

- ◆ a quantidade de produtos estocados em cada um dos armazéns;
- ◆ a quantidade de cada um dos produtos estocados em todos os armazéns juntos;
- ◆ o preço do produto que possui maior estoque em um único armazém;
- ◆ o menor estoque armazenado;
- ◆ o custo de cada armazém.

**25.** Faça um programa que receba os preços de 20 produtos em cinco lojas diferentes e armazene-os em uma matriz  $20 \times 5$ . Desconsiderando empates, mostre o número do produto e o número da loja do produto mais caro.

# FUNÇÕES DE TRATAMENTO DE CARACTERES

## 7.1 PRINCIPAIS FUNÇÕES DE TRATAMENTO DE CARACTERES EM PASCAL

COPY (**cadeia**, **posição**, **número**)

Copia da **cadeia**, a partir da **posição** dada, o **número** de caracteres estipulados.

LENGTH (**cadeia**)

Mostra o número de caracteres da **cadeia**.

POS(**cadeia1**, **cadeia2**)

Mostra, a partir de que posição, a **cadeia1** aparece dentro da **cadeia2**.

DELETE(**cadeia**, **posição**, **número**)

Apaga da **cadeia**, a partir da **posição** dada, o **número** de caracteres estipulados.

INSERT(**cadeia1**, **cadeia2**, **posição**)

Insere a **cadeia1** na **cadeia2** a partir da **posição** dada.

CONCAT(**cadeia1**,**cadeia2**,**cadeia3**) OU **cadeia1+cadeia2+cadeia3**

Soma as cadeias.

## 7.2 PRINCIPAIS FUNÇÕES DE TRATAMENTO DE CARACTERES EM C/C++

A linguagem C/C++ não possui um tipo de dados similar à string da linguagem PASCAL. Em vez disso, para armazenar uma cadeia de caracteres utilize vetores (matrizes unidimensionais), onde cada posição representa um caractere.

É importante ressaltar que os compiladores da linguagem C/C++ identificam o fim de uma cadeia por meio do caractere nulo, ou seja, por meio de '\0'. Sendo assim, deve-se declarar sempre o vetor com uma posição a mais para armazenar o caractere nulo (esse caractere não precisa ser armazenado manualmente, isso é feito automaticamente pelo compilador). Por exemplo, para armazenar a palavra CADEIA deve-se declarar um vetor do tipo **char** com sete posições (que ocuparão posições contíguas na memória).

char <b>palavra</b> [7];								
índice	...	0	1	2	3	4	5	6
valor	...	C	A	D	E	I	A	\0
posição memória	...	863	864	865	866	867	868	869

Conforme pode ser observado na tabela anterior, a variável **palavra**, quando é declarada, pode ocupar qualquer posição disponível na memória; entretanto, todas as posições do vetor ocupam espaços de memória adjacentes, sendo que cada caractere ocupa 1 byte. Por isso, conforme o exemplo, se a letra C estiver armazenada na posição de memória de endereço 863 (e este endereço equivale a 1 byte), a letra A deverá ocupar o endereço 864 e assim por diante.

### 7.2.1 MANIPULANDO CADEIAS DE CARACTERES

Como todas as cadeias de caracteres são variáveis compostas homogêneas (vetor ou matriz), deve-se utilizar funções específicas. Algumas delas são descritas a seguir (as funções descritas fazem parte da biblioteca `string.h`).

`strcat(str1, str2)`

Concatena a cadeia identificada por **str2** à cadeia **str1**.

`strchr(str1, ch)`

Retorna um ponteiro para a posição da cadeia **str1**, onde o caractere **ch** é encontrado pela primeira vez.

`strcpy(str1, str2)`

Copia a cadeia **str2** para a cadeia **str1**.

`strcmp(str1, str2)`

Compara duas cadeias de caracteres e retorna um número inteiro, que pode ser:

- ◆ zero se as duas cadeias forem iguais;
- ◆ um número menor que 0 se a cadeia **str1** for alfabeticamente menor que **str2**;
- ◆ um número maior que 0 se a cadeia **str1** for alfabeticamente maior que **str2**.

Essa função considera letras maiúsculas como sendo símbolos diferentes de letras minúsculas.

`stricmp(str1, str2)`

Compara duas cadeias de caracteres e retorna um número inteiro, que pode ser:

- ◆ zero se as duas cadeias forem iguais;
- ◆ um número menor que 0 se a cadeia **str1** for alfabeticamente menor que **str2**;
- ◆ um número maior que 0 se a cadeia **str1** for alfabeticamente maior que **str2**.

Essa função considera letras maiúsculas ou minúsculas como sendo símbolos iguais.

`strlen(str1)`

Obtém o tamanho de uma cadeia de caracteres **str1** (o caractere nulo não é considerado).

`strstr(str1, str2)`

Retorna um ponteiro para a posição da cadeia **str1**, onde a cadeia **str2** é encontrada pela primeira vez.

### 7.2.2 INICIALIZANDO CADEIAS DE CARACTERES

As variáveis que armazenam cadeias de caracteres podem ser inicializadas automaticamente pelo programa ou podem receber um valor por meio do teclado. A seguir exemplificamos alguns casos:

- a) inicialização no momento da declaração:

```
char nome[] = {'P', 'r', 'o', 'g', 'r', 'a', 'm', 'a', '\0'};  
ou  
char nome[] = "Programa";
```

No primeiro caso, a variável **nome** recebeu as letras separadamente (inclusive o caractere nulo). Por isso, cada uma das letras estava envolvida por apóstrofos (' ') – esta é a maneira de identificar um caractere isoladamente.

No segundo caso, a variável **nome** foi inicializada com uma palavra, recebendo automaticamente o caractere nulo. Por isso, a palavra **Programa** estava envolvida por aspas ("") – essa é a maneira de identificar uma cadeia de caracteres.

Em ambos os casos, não houve necessidade de expressar o número de posições dentro dos colchetes ([ ]), pois o mesmo é definido automaticamente em função da inicialização.

- b) inicialização por meio da atribuição (depois da declaração)

```
char vet1[10], vet2[5];  
  
strcpy(vet1, "Programa");  
ou  
strcpy (str1, str2);
```

No primeiro caso, a variável **vet1** recebe um valor constante (a palavra **Programa**). No segundo caso, o conteúdo da variável **str2** é copiado na variável **str1**.

- c) inicialização por meio do teclado

```
char vet[10];  
  
cin >> vet;
```

O comando **cin** consegue armazenar valores vindos do teclado. No caso de uma cadeia de caracteres, o comando **cin** consegue armazenar todos os símbolos digitados até a ocorrência do primeiro espaço em branco. Por exemplo, se for digitado o nome Maria da Silva, a variável **vet** armazenará apenas Maria (o que vem depois do espaço em branco é perdido). Para resolver esse problema, utiliza-se a função **gets**.

```
gets(vet);
```

A função **gets** armazena dentro da variável **vet** todos os símbolos digitados até a ocorrência do ENTER (a função **gets** faz parte da biblioteca **stdio.h**).

## EXERCÍCIOS RESOLVIDOS

1. Faça um programa que receba uma frase, calcule e mostre a quantidade de palavras da frase digitada.

- Digitar uma frase
- Pegar o tamanho da frase

- Percorrer a frase pegando caractere a caractere
- Comparar cada caractere com espaço em branco
- Quando encontrar espaço, encontrou o fim de uma palavra
- Como após a última palavra não tem espaço, acrescenta-se um na  
→ quantidade.

**SOLUÇÃO:**

```
\EXERC\CAP7\PASCAL\EX1.PAS e \EXERC\CAP7\PASCAL\EX1.EXE
```

**SOLUÇÃO:**

```
\EXERC\CAP7\C++\EX1.CPP e \EXERC\CAP7\C++\EX1.EXE
```

- 2.** Faça um programa que receba uma frase e uma palavra. Caso a frase contenha a palavra ESCOLA, substitua-a pela palavra digitada.

Exemplo:

Frase: EU MORO PERTO DE UMA ESCOLA. MAS ESSA ESCOLA NÃO É A MELHOR.

Palavra: PADARIA

Resposta: EU MORO PERTO DE UMA PADARIA. MAS ESSA PADARIA NÃO É A MELHOR.

**SOLUÇÃO:**

- Digitar uma frase
- Digitar a palavra que substituirá a palavra ESCOLA
- Pegar o tamanho da frase
- Percorrer a frase pegando de 6 em 6 caracteres, tendo em vista que  
→ a palavra ESCOLA possui 6 caracteres
- Comparar os caracteres extraídos da frase com a palavra ESCOLA
- Quando encontrar, apagar a palavra ESCOLA e no seu lugar inserir a  
→ palavra digitada
- Depois de substituir, atualizar o tamanho da frase a ser  
→ percorrida.

**SOLUÇÃO:**

```
\EXERC\CAP7\PASCAL\EX2.PAS e \EXERC\CAP7\PASCAL\EX2.EXE
```

**SOLUÇÃO:**

```
\EXERC\CAP7\C++\EX2.CPP e \EXERC\CAP7\C++\EX2.EXE
```

- 3.** Faça um programa que receba uma frase. Caso a frase possua meses por extenso, substitua-os pelo seu número correspondente, como mostra o exemplo a seguir.

Exemplo:

Frase: NO MÊS DE JANEIRO FAZ CALOR.

Nova frase: NO MÊS 01 FAZ CALOR.

**SOLUÇÃO:**

- Digitar uma frase
- Pegar o tamanho da frase

- Percorrer a frase pegando o número de caracteres dos meses do ano,
- ↳ por exemplo, para janeiro pegar de 7 em 7 caracteres, para
- ↳ fevereiro pegar de 9 em 9 caracteres, etc.
- Comparar os caracteres extraídos da frase com cada mês do ano.
- Quando encontrar, apagar a quantidade de caracteres encontrados e
- ↳ no seu lugar inserir o número que corresponde ao mês
- Depois de substituir, atualizar o tamanho da frase a ser
- ↳ percorrida.

**SOLUÇÃO:**

```
\EXERC\CAP7\PASCAL\EX3.PAS e \EXERC\CAP7\PASCAL\EX3.EXE
```

**SOLUÇÃO:**

```
\EXERC\CAP7\C++\EX3.CPP e \EXERC\CAP7\C++\EX3.EXE
```

4. Faça um programa para criptografar uma frase dada pelo usuário, ou seja, a criptografia troca as vogais da frase por \*.

Exemplo:

Frase: EU ESTOU NA ESCOLA

Saída: \*\* \* ST\*\* N\* \*SC\*L\*

**SOLUÇÃO:**

- Digitar uma frase
- Pegar o tamanho da frase
- Percorrer a frase pegando 1 caractere de cada vez
- Comparar o caractere extraído com as vogais
- Quando encontrar, apagar a vogal e no seu lugar inserir o
- ↳ asterisco.

**SOLUÇÃO:**

```
\EXERC\CAP7\PASCAL\EX4.PAS e \EXERC\CAP7\PASCAL\EX4.EXE
```

**SOLUÇÃO:**

```
\EXERC\CAP7\C++\EX4.CPP e \EXERC\CAP7\C++\EX4.EXE
```

5. Faça um programa que se comporte como vírus, ou seja, que duplica cada uma das palavras digitadas pelo usuário.

Exemplo:

Frase: EU ESTOU NA ESCOLA

Saída: EU EU ESTOU ESTOU NA NA ESCOLA ESCOLA

**SOLUÇÃO:**

- Digitar uma frase
- Pegar o tamanho da frase
- Percorrer a frase pegando caractere a caractere
- Comparar cada caractere com espaço em branco
- Quando encontrar espaço, encontrou o fim de uma palavra, então
- ↳ copia a palavra encontrada
- Inserir a palavra encontrada na frase
- Atualizar o tamanho da frase.

**SOLUÇÃO:**

\EXERC\CAP7\PASCAL\EX5.PAS e \EXERC\CAP7\PASCAL\EX5.EXE

**SOLUÇÃO:**

\EXERC\CAP7\C++\EX5.CPP e \EXERC\CAP7\C++\EX5.EXE

- 6.** Faça um programa que receba o nome completo de uma pessoa e mostre os nomes intermediários abreviados.

Exemplo:

Nome: Maria Silva Costa

Saída: Maria S. Costa

Nome: João Carlos Gomes Marques

Saída: João C. G. Marques

**SOLUÇÃO:**

- Digitar uma frase
- Pegar o tamanho da frase
- Percorrer a frase pegando caractere a caractere
- Comparar cada caractere com espaço em branco
- Quando encontrar o primeiro espaço em branco, encontrou o fim da
  - ↳ primeira palavra, então copia a palavra encontrada, pois a
  - ↳ primeira palavra não será abreviada
- Percorrer a frase pegando caractere a caractere, do final para o
  - ↳ começo
- Comparar cada caractere com espaço em branco, para achar a posição
  - ↳ inicial da última palavra, pois esta também não será abreviada
- Percorrer a frase do final da primeira palavra até a posição
  - ↳ inicial da última palavra
- Juntar a primeira palavra com as abreviaturas e depois com a última
  - ↳ palavra.

**SOLUÇÃO:**

\EXERC\CAP7\PASCAL\EX6.PAS e \EXERC\CAP7\PASCAL\EX6.EXE

**SOLUÇÃO:**

\EXERC\CAP7\C++\EX6.CPP e \EXERC\CAP7\C++\EX6.EXE

- 7.** Faça um programa que receba o nome completo de uma pessoa e reescreva-o de acordo com o exemplo a seguir.

Exemplo:

Nome: Maria Silva Costa

Saída: Costa, M. S.

Nome: João Carlos Gomes Marques

Saída: Marques, J. C. G.

**SOLUÇÃO:**

- Digitar uma frase
- Pegar o tamanho da frase

- Percorrer a frase pegando caractere a caractere, do final da frase
- ↳ para o começo da frase
- Comparar cada caractere com espaço em branco
- Quando encontrar o primeiro espaço em branco, encontrou o começo da última palavra, então copia a palavra encontrada, pois a última palavra será a primeira
- Juntar a primeira letra com um ponto e um espaço em branco com a última palavra que será a primeira palavra da resposta
- Percorrer a frase pegando caractere a caractere, do início da frase
- ↳ até a posição inicial da última palavra
- Comparar cada caractere com espaço em branco e, quando encontrar,
- ↳ juntar a letra da próxima posição com um ponto e um branco com a resposta anteriormente obtida.

**SOLUÇÃO:**

```
\EXERC\CAP7\PASCAL\EX7.PAS e \EXERC\CAP7\PASCAL\EX7.EXE
```

**SOLUÇÃO:**

```
\EXERC\CAP7\C++\EX7.CPP e \EXERC\CAP7\C++\EX7.EXE
```

- 8.** Faça um programa que receba duas frases e gere uma terceira que represente a combinação das palavras das duas frases lidas.

Exemplo:

Frase 1: Hoje está um belo dia

Frase 2: Talvez chova amanhã

Saída: Hoje talvez está chova um amanhã belo dia

**SOLUÇÃO:**

- Digitar duas frases
- Pegar o tamanho das frases digitadas
- Percorrer a primeira frase pegando caractere a caractere até chegar a um espaço em branco ou até chegar ao fim da frase
- Quando encontrar o espaço em branco ou o fim da primeira frase,
- ↳ conseguiu pegar uma palavra completa, concatene-a, então, à nova frase
- Percorrer a segunda frase pegando caractere a caractere até chegar a um espaço em branco ou até chegar ao fim da frase
- Quando encontrar o espaço em branco ou o fim da segunda frase,
- ↳ conseguiu pegar uma palavra completa, concatene-a, então, à nova frase
- Repita os procedimentos anteriores até percorrer as duas frases
- ↳ digitadas até o fim.

**SOLUÇÃO:**

```
\EXERC\CAP7\PASCAL\EX8.PAS e \EXERC\CAP7\PASCAL\EX8.EXE
```

**SOLUÇÃO:**

```
\EXERC\CAP7\C++\EX8.CPP e \EXERC\CAP7\C++\EX8.EXE
```

- 9.** Faça um programa que receba uma frase e coloque as palavras em ordem crescente.

Exemplo:

Entrada: A informática está em constante evolução

Saída: A constante em está evolução informática



#### SOLUÇÃO:

- Digitar uma frase
- Pegar o tamanho da frase digitada
- Percorrer a frase, pegando caractere a caractere
- Cada vez que encontrar um espaço em branco ou o fim da primeira frase, conseguiu pegar uma palavra completa
- Se a nova frase (onde as palavras ficarão em ordem crescente)
  - ↳ estiver vazia, copie nela a palavra obtida
  - Caso contrário, percorrer a nova frase, até encontrar a posição adequada para colocar a palavra extraída da frase digitada
  - Caso a palavra extraída da frase digitada seja maior do que todas as que já estão armazenadas na nova frase, concatene essa palavra ao final da nova frase.



#### SOLUÇÃO:

\EXERC\CAP7\PASCAL\EX9.PAS e \EXERC\CAP7\PASCAL\EX9.EXE



#### SOLUÇÃO:

\EXERC\CAP7\C++\EX9.CPP e \EXERC\CAP7\C++\EX9.EXE

10. Faça um programa que receba uma frase e mostre as letras que se repetem, juntamente com o número de repetições.

Exemplo: A PROVA FOI ADIADA

- ◆ a letra A apareceu cinco vezes
- ◆ a letra O apareceu duas vezes
- ◆ a letra I apareceu duas vezes
- ◆ a letra D apareceu duas vezes



#### SOLUÇÃO:

- Digitar uma frase
- Pegar o tamanho da frase digitada
- Percorrer a frase pegando caractere a caractere
- Verificar se é a primeira vez que esse caractere aparece na frase
- Caso seja a primeira vez, atribua 1 ao contador de aparições
- Caso contrário, incremente o contador de aparições em uma unidade
- Mostre todas as letras que apareceram mais que uma vez (que se repetiram), juntamente com o total de repetições.



#### SOLUÇÃO:

\EXERC\CAP7\PASCAL\EX10.PAS e \EXERC\CAP7\PASCAL\EX10.EXE



#### SOLUÇÃO:

\EXERC\CAP7\C++\EX10.CPP e \EXERC\CAP7\C++\EX10.EXE

**EXERCÍCIOS PROPOSTOS**

1. Faça um programa que receba uma frase, calcule e mostre a quantidade de vogais da frase digitada.
2. Faça um programa que receba uma frase, calcule e mostre a quantidade de consoantes da frase digitada.
3. Faça um programa que receba uma frase, calcule e mostre a quantidade de vezes que a palavra AULA aparece na frase digitada.
4. Faça um programa que receba uma frase e uma palavra, calcule e mostre a quantidade de vezes que a palavra digitada aparece na frase.

Exemplo:

Frase: EU ESTOU NA ESCOLA, GOSTO MUITO DE ESTUDAR E ACHO QUE A ESCOLA É LEGAL.

Palavra: ESCOLA

Resposta: A palavra ESCOLA apareceu duas vezes na frase.

5. Faça um programa que receba uma frase e troque a palavra ALUNO por ESTUDANTE e a palavra ESCOLA por UNIVERSIDADE.

Exemplo: EU SOU ALUNO DA ESCOLA

Saída: EU SOU ESTUDANTE DA UNIVERSIDADE

6. Faça um programa que receba uma frase e, a cada ocorrência da palavra TECLADO, inserir o texto OU MOUSE.

Exemplo:

Frase: PODE-SE UTILZAR O TECLADO PARA ENTRADA DE DADOS.

Resposta: PODE-SE UTILZAR O TECLADO OU MOUSE PARA ENTRADA DE DADOS.

7. Faça um programa para criptografar uma frase dada pelo usuário, ou seja, a criptografia inverte a frase.

Exemplo:

Frase: EU ESTOU NA ESCOLA

Saída: ALOCSE AN UOTSE UE

8. Faça um programa para criptografar uma frase dada pelo usuário, ou seja, a criptografia inverte a frase e troca as consoantes por #.

Exemplo:

Frase: EU ESTOU NA ESCOLA

Saída: A#O##E A# UO##E EU

9. Faça um programa que receba uma frase e mostre cada palavra dessa frase em linha separada.

Exemplo: COMPUTADORES SÃO MÁQUINAS POTENTES

Saída:

COMPUTADORES  
SÃO  
MÁQUINAS  
POTENTES

10. Faça um programa que receba uma frase com letras minúsculas e converta a primeira letra de cada palavra para maiúscula.

Exemplo:

Entrada: fazer exercícios faz bem.

Saída: Fazer Exercícios Faz Bem.

## CAPÍTULO

# 8

## REGISTROS

### 8.1 DEFINIÇÃO DE REGISTROS

Registros são estruturas que podem agregar diferentes informações. Dessa maneira, podem ser feitas diferentes combinações, gerando novos tipos de dados. Um registro é uma coleção de campos, em que cada campo pode ser de um tipo de dado diferente. Por isso, os registros são conhecidos como variáveis compostas heterogêneas.

### 8.2 DECLARAÇÃO DE REGISTROS EM ALGORITMOS

Para que um registro possa ser acessado deve existir uma variável do tipo do registro declarada, sendo esta uma variável simples, um vetor ou uma matriz.

```
Declare nome_da_variável_registro REGISTRO (nome_campo  
    ↪ TIPO_DO_CAMPO)
```

Exemplo:

```
Declare conta REGISTRO (num, saldo NUMÉRICO, nome LITERAL)
```

O registro declarado é denominado **conta** e possui três campos, os campos **num** e **saldo** são campos numéricos e o campo **nome** que é do tipo literal.

conta	num	conta.num
	saldo	conta.saldo
	nome	conta.nome

Exemplo:

```
Declare conta[3] REGISTRO (num, saldo NUMÉRICO, nome LITERAL)
```

O registro declarado é um vetor denominado **conta** e possui três campos, os campos **num** e **saldo** são campos numéricos e o campo **nome** é do tipo literal.

<b>conta</b>		
	<b>num</b>	<b>conta[1].num</b>
	<b>saldo</b>	<b>conta[1].saldo</b>
	<b>nome</b>	<b>conta[1].nome</b>
	<b>num</b>	<b>conta[2].num</b>
	<b>saldo</b>	<b>conta[2].saldo</b>
	<b>nome</b>	<b>conta[2].nome</b>
	<b>num</b>	<b>conta[3].num</b>
	<b>saldo</b>	<b>conta[3].saldo</b>
	<b>nome</b>	<b>conta[3].nome</b>

### 8.3 DECLARAÇÃO DE REGISTROS EM PASCAL

Os registros em PASCAL são definidos pelas palavras TYPE e RECORD, conforme sintaxe apresentada a seguir.

```
TYPE nome_da_variável_registro = RECORD
    campo:tipo;
END;
VAR nome_da_variável: nome_da_variável_registro;
```

Exemplo:

```
TYPE REGISTRO = RECORD
    num : INTEGER;
    nome : STRING[35];
    saldo : REAL;
END;
VAR conta: REGISTRO;
```

Exemplo:

```
TYPE REGISTRO = RECORD
    num : INTEGER;
    nome : STRING[35];
    saldo : REAL;
END;
VAR conta: ARRAY[1..15] OF REGISTRO;
```

#### 8.3.1 ACESSO AOS CAMPOS DE UM REGISTRO EM PASCAL

**nome\_da\_variável\_do\_tipo\_registro.nome\_do\_campo**

Exemplos:

**conta.num:= 12;**

coloca o número 12 no campo **num** do registro denominado **conta**.

**conta[2].num:= 13;**

coloca o número 13 no campo **num** do registro denominado **conta**, sendo esse um vetor, o número 13 será colocado na posição 2.

### 8.4 DECLARAÇÃO DE REGISTROS EM C/C++

Os registros em C/C++ são definidos pela palavra reservada **struct**, conforme apresentado a seguir.

```
struct produto
{ int codigo;
  char descricao[30];
};
```

A partir da estrutura definida anteriormente, o programa poderá considerar que existe um novo tipo de dado para ser utilizado, chamado **produto**. Esse novo tipo de dado é capaz de armazenar um **codigo** (número inteiro) e uma **descricao** (literal de tamanho 30). O **codigo** e a **descricao** são chamados de *campos da estrutura*.

#### 8.4.1 DECLARAÇÃO DE VARIÁVEIS DO TIPO REGISTROS EM C/C++

É importante ressaltar que para o programa utilizar esse novo tipo de dado, deve-se declarar uma variável como sendo do tipo **produto**.

```
produto x;
```

Considerando que estruturas representam novos tipos de dados, todas as operações e declarações realizadas com os tipos predefinidos da linguagem também poderão ser realizadas com as estruturas. Dessa maneira, variáveis, vetores e matrizes também podem ser declaradas como sendo do tipo **estrutura**.

```
produto var[10];
```

A variável **var** é um vetor de dez posições e, em cada posição, serão armazenados um código e uma descrição.

```
produto var[10][6];
```

A variável **var** é uma matriz bidimensional com dez linhas e seis colunas, onde cada célula armazena um código e uma descrição.

Em alguns momentos é possível fazer a declaração da variável junto com a definição da estrutura, no mesmo bloco de comandos. Observe o exemplo a seguir:

```
struct produto
{ int codigo;
  char descricao[30];
}x[5][8];
```

Nesse caso, está sendo declarada uma variável **x** com cinco linhas e oito colunas, onde cada posição é do tipo **produto** e contém um **codigo** e uma **descricao**.

#### 8.4.2 ACESSO A MEMBROS DE ESTRUTURAS

Depois que uma variável é declarada, o programa poderá manipular o seu conteúdo, ou seja, os valores armazenados em cada membro da estrutura.

Para que tais manipulações sejam feitas é preciso informar o nome da variável e o campo que será manipulado. Por exemplo:

- para armazenar os valores 1 e ‘geladeira’ na variável **x** deve-se fazer atribuições assim:

```
produto x; //declaração da variável x
x.codigo = 1;
strcpy(x.descricao, "geladeira");
```

- para armazenar os valores 5 e ‘fogão’ na 4<sup>a</sup> posição de um vetor, deve-se fazer atribuições assim:

```
produto eletro[12];
```

```
eletro[3].codigo = 5;
strcpy(eletro[3].descricao,"fogão");
```

**OBSERVAÇÃO:** Considerando que o vetor começa na posição 0, a quarta posição é a apontada pelo índice 3.

- c) para armazenar os valores 11 e ‘televisão’ na 3<sup>a</sup> coluna da 6<sup>a</sup> linha de uma matriz. deve-se fazer atribuições assim:

```
produto eletro[8][4]
eletro[5][2].codigo = 11;
strcpy(eletro[5][2].descricao,"televisão");
```

**OBSERVAÇÃO:** Considerando que tanto as linhas como as colunas de uma matriz começam na posição 0, a 6<sup>a</sup> linha é representada pelo índice 5 e a 3<sup>a</sup> coluna é representada pelo índice 2.

## EXERCÍCIOS RESOLVIDOS

1. Faça um programa que realize o cadastro de contas bancárias com as seguintes informações: número da conta, nome do cliente e saldo. O banco permitirá o cadastramento de apenas 15 contas e não pode haver mais de uma conta com o mesmo número. Crie o menu de opções a seguir:

Menu de opções:

1. Cadastrar contas
2. Visualizar todas as contas de um determinado cliente
3. Excluir a conta com menor saldo (supondo a não existência de saldos iguais)
4. Sair

### ALGORITMO

#### SOLUÇÃO:

```
ALGORITMO
DECLARE conta[15] REGISTRO (num, saldo NUMÉRICO, nome LITERAL)
      i, op, pos, achou, num_conta NUMÉRICO
      saldo_cliente, menor_saldo NUMÉRICO
      nome_cliente LITERAL
PARA i ← 1 ATÉ 15 FAÇA
INÍCIO
conta[i].num ← 0
conta[i].nome ← ''
conta[i].saldo ← 0
FIM
REPITA
ESCREVA "Menu de Opções"
ESCREVA "1 - Cadastrar contas"
ESCREVA "2 - Visualizar todas as contas de um determinado cliente"
ESCREVA "3 - Excluir conta de menor saldo"
ESCREVA "4 - Sair"
ESCREVA "Digite sua opção"
LEIA op
SE (op < 1) OU (op > 4)
ENTÃO ESCREVA "Opção Inválida"
SE op = 1
ENTÃO INÍCIO
achou ← 0
ESCREVA "Digite o número da conta a ser incluída"
LEIA num_conta
PARA i ← 1 ATÉ 15 FAÇA
```

```

INÍCIO
SE num_conta = conta[i].num
ENTÃO achou ← 1
FIM
SE achou = 1
ENTÃO ESCREVA "Já existe conta cadastrada com esse número"
SENÃO INÍCIO
    posi ← 0
    i ← 1
    ENQUANTO i <= 15 FAÇA
        INÍCIO
            SE conta[i].num = 0
            ENTÃO INÍCIO
                posi ← i
                i ← 16
            FIM
            i ← i + 1
        FIM
        SE posi = 0
        ENTÃO ESCREVA "Impossível cadastrar novas contas"
        SENÃO INÍCIO
            ESCREVA "Digite o nome do cliente"
            LEIA nome_cliente
            ESCREVA "Digite o saldo do cliente"
            LEIA saldo_cliente
            conta[posi].num ← num_conta
            conta[posi].nome ← nome_cliente
            conta[posi].saldo ← saldo_cliente
            ESCREVA "Conta cadastrada com sucesso"
        FIM
    FIM
FIM
SE op = 2
ENTÃO INÍCIO
    ESCREVA "Digite o nome do cliente a ser consultado"
    LEIA nome_cliente
    achou ← 0
    PARA i ← 1 ATÉ 15 FAÇA
        INÍCIO
            SE conta[i].nome = nome_cliente
            ENTÃO INÍCIO
                ESCREVA conta[i].num, conta[i].saldo
                achou ← 1
            FIM
        FIM
    SE achou = 0
    ENTÃO ESCREVA "Não existe conta cadastrada para este
    cliente"
FIM
SE op = 3
ENTÃO INÍCIO
    i ← 1
    achou ← 0
    ENQUANTO i <= 15 FAÇA
        INÍCIO
            SE conta[i].num ≠ 0
            ENTÃO INÍCIO
                menor_saldo ← conta[i].saldo
                achou ← 1
                posi ← i
                i ← 16
            FIM
        i ← i + 1
    
```

```

FIM
SE achou = 0
ENTÃO ESCREVA "Nenhuma conta foi cadastrada"
SENÃO INÍCIO
    PARA i ← 1 ATÉ 15 FAÇA
    INÍCIO
        SE (conta[i].saldo < menor_saldo)
        E (conta[i].num ≠ 0)
        ENTÃO posi ← i
    FIM
    conta[i].num ← 0
    conta[i].nome ← "
    conta[i].saldo ← 0
    ESCREVA "Conta excluída com sucesso"
    FIM
FIM
ATÉ op = 4
FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP8\PASCAL\EX1.PAS e \EXERC\CAP8\PASCAL\EX1.EXE

**SOLUÇÃO:**

\EXERC\CAP8\C++\EX1.CPP e \EXERC\CAP8\C++\EX1.EXE

- 2.** Uma empresa prestadora de serviços armazena informações sobre os serviços prestados. Sabe-se que essa empresa pode atender a apenas três clientes em cada dia, ou seja, realiza, no máximo, três serviços diariamente. É de interesse da direção dessa empresa manter um histórico mensal (30 dias) sobre os serviços prestados.

A empresa realiza quatro tipos de serviços: 1 – Pintura; 2 – Jardinagem; 3 – Faxina e 4 – Reforma em geral.

Cada serviço desenvolvido deve ser cadastrado com as seguintes informações: número do serviço, valor do serviço, código do serviço e código do cliente.

Cadastre todos os tipos de serviços (código e descrição) que a empresa poderá realizar. Para isso, utilize um vetor de quatro posições capaz de armazenar as informações de cada um dos quatro tipos de serviços.

Mostre o seguinte menu de opções:

1. Cadastrar os tipos de serviços
2. Cadastrar os serviços prestados
3. Mostrar os serviços prestados em um determinado dia
4. Mostrar os serviços prestados dentro de um intervalo de valor
5. Mostrar um relatório geral (separado por dia), que exibe, inclusive, a descrição do tipo do serviço
6. Finalizar

**Para a opção 1:** deve-se cadastrar os tipos de serviços oferecidos pela empresa, com código e descrição.

**Para a opção 2:** deve-se considerar que deverão ser cadastrados os serviços prestados ao longo do mês. Em cada dia podem ser cadastrados, no máximo, três serviços prestados.

Utilize uma matriz capaz de armazenar em cada posição todas as informações referentes a um serviço prestado. Cada linha representa um dia do mês. Dessa maneira, considere a matriz com dimensão  $30 \times 3$ .

Solicite o dia em que o serviço foi prestado e as demais informações.

Lembre-se de que a empresa só pode prestar os serviços que já tenham sido cadastrados no vetor de tipo de serviços.

Caso o usuário digite um código de tipo de serviço inválido, mostre uma mensagem de erro.

Quando o usuário tentar cadastrar mais de três serviços prestados em um mesmo dia, mostre uma mensagem de erro.

**Para a opção 3:** receba o dia que deseja consultar e mostre os respectivos serviços prestados.

**Para a opção 4:** receba o valor mínimo e o valor máximo e mostre os serviços prestados que estiverem nesse intervalo.

**Para a opção 5:** mostre todos os serviços prestados, conforme o exemplo a seguir:

#### DIA - 01

Nº DO SERVIÇO	VALOR DO SERVIÇO	CÓDIGO DO SERVIÇO	DESCRIÇÃO	CÓDIGO DO CLIENTE
100	R\$ 200,00	1	Pintura	1
150	R\$ 100,00	3	Faxina	5

#### DIA - 02

Nº DO SERVIÇO	VALOR DO SERVIÇO	CÓDIGO DO SERVIÇO	DESCRIÇÃO	CÓDIGO DO CLIENTE
301	R\$ 600,00	4	Reforma em geral	3
280	R\$ 352,00	1	Pintura	2

#### ALGORITMO

#### SOLUÇÃO:

```

ALGORITMO
DECLARE tipos[4] REGISTRO (cod NUMÉRICO, desc LITERAL)
      serv[30,3] REGISTRO (num, valor, cod_serv, cod_cliente
      ↪ NUMÉRICO)
      i, j, op, codigo_serv, achou NUMÉRICO
      dia, codigo_cliente, valor_serv, num_serv, valida NUMÉRICO
      valor_inicial, valor_final, k NUMÉRICO
      desc_serv LITERAL
PARA i ← 1 ATÉ 4 FAÇA
INÍCIO
tipos[i].cod ← 0
tipos[i].desc ← "
FIM
PARA i ← 1 ATÉ 30 FAÇA
INÍCIO
PARA j ← 1 ATÉ 3 FAÇA
INÍCIO
serv[i, j].num ← 0
serv[i, j].valor ← 0
serv[i, j].cod_serv ← 0

```

```

        serv[i, j].cod_cliente ← 0
    FIM
REPITA
    ESCREVA "Menu de Opções"
    ESCREVA "1 - Cadastrar tipos de serviços"
    ESCREVA "2 - Cadastrar serviços prestados"
    ESCREVA "3 - Mostrar os serviços prestados em um determinado dia"
    ESCREVA "4 - Mostrar os serviços prestados dentro de um intervalo de
    ↪ valor"
    ESCREVA "5 - Mostrar um relatório geral, separado por dia"
    ESCREVA "6 - Sair"
    ESCREVA "Digite sua opção"
LEIA op
SE (op < 1) OU (op > 6)
ENTÃO ESCREVA "Opção Inválida"
SE op = 1
ENTÃO INÍCIO
    ESCREVA "Digite o código do serviço a ser cadastrado"
    LEIA codigo_serv
    achou ← 0
    PARA i ← 1 ATÉ 4 FAÇA
    INÍCIO
        SE tipos[i].cod = 0
        ENTÃO achou ← i
    FIM
    SE achou = 0
    ENTÃO ESCREVA "Cadastro de tipos de serviços lotado"
    SENÃO INÍCIO
        PARA i ← 1 ATÉ 4 FAÇA
        INÍCIO
            SE tipos[i].cod = codigo_serv
            ENTÃO achou ← 0
        FIM
        SE achou = 0
        ENTÃO ESCREVA "Já existe tipo de serviço cadastrado com
        ↪ esse código"
        SENÃO INÍCIO
            ESCREVA "Digite a descrição do tipo de serviço a
            ↪ ser cadastrado"
            LEIA desc_serv
            tipos[achou].cod ← codigo_serv
            tipos[achou].desc ← desc_serv
            ESCREVA "Tipo de serviço cadastrado com sucesso"
        FIM
    FIM
    FIM
SE op = 2
ENTÃO INÍCIO
    ESCREVA "Digite o dia em que deseja cadastrar o serviço
    ↪ prestado"
    LEIA dia
    achou ← 0
    PARA j ← 1 ATÉ 3 FAÇA
    INÍCIO
        SE serv[dia, j] = 0
        ENTÃO achou ← j
    FIM
    SE achou = 0
    ENTÃO ESCREVA "Todos os serviços prestados neste dia já foram
    ↪ cadastrados"
    SENÃO INÍCIO

```

```
ESCREVA "Digite o código do serviço a ser
        ↪ cadastrado"
LEIA codigo_serv
valida ← 0
PARA i ← 1 ATÉ 4 FAÇA
INÍCIO
SE tipos[i] = codigo_serv
ENTÃO valida ← 1
FIM
SE valida = 0
ENTÃO ESCREVA "Código de serviço inválido"
SENÃO INÍCIO
    ESCREVA "Digite o número do serviço"
    LEIA num_serv
    ESCREVA "Digite o valor do serviço"
    LEIA valor_serv
    ESCREVA "Digite o código do cliente"
    LEIA codigo_cliente
    serv[dia, achou].num ← num_serv
    serv[dia, achou].valor ← valor_serv
    serv[dia, achou].cod_serv ← codigo_serv
    serv[dia, achou].cod_cliente ←
        ↪ codigo_cliente
    ESCREVA "Serviço prestado cadastrado com
        ↪ sucesso"
    FIM
FIM
SE op = 3
ENTÃO INÍCIO
    ESCREVA "Digite o dia em que deseja consultar os serviços
        ↪ prestados"
    LEIA dia
    achou ← 0
    PARA j ← 1 ATÉ 3 FAÇA
    INÍCIO
    SE serv[dia, j].num ≠ 0
    ENTÃO achou ← 1
    FIM
    SE achou = 0
    ENTÃO ESCREVA "Nenhum serviço foi prestado neste dia"
    SENÃO INÍCIO
        ESCREVA "Serviços prestados no dia", dia
        PARA j ← 1 ATÉ 3 FAÇA
        INÍCIO
        SE serv[dia, j].num = 0
        ENTÃO INÍCIO
            ESCREVA serv[dia, j].num, serv[dia, j].valor
            ESCREVA serv[dia, j].cod_serv
            PARA i ← 1 ATÉ 4 FAÇA
            INÍCIO
            SE tipos[i].cod = serv[dia, j].cod_serv
            ENTÃO ESCREVA tipos[i].desc
            FIM
            ESCREVA serv[dia, j].cod_cliente
            FIM
        FIM
        FIM
    FIM
    FIM
FIM
SE op = 4
ENTÃO INÍCIO
    ESCREVA "Digite o valor inicial"
```

```

LEIA valor_inicial
ESCREVA "Digite o valor final"
LEIA valor_final
achou ← 0
PARA i ← 1 ATÉ 30 FAÇA
INÍCIO
PARA j ← 1 ATÉ 3 FAÇA
INÍCIO
SE (serv[i, j].valor >= valor_inicial)
  E (serv[i, j].valor <= valor_final)
ENTÃO INÍCIO
  achou ← 1
  ESCREVA serv[i, j].num, serv[i, j].valor
  ESCREVA serv[i, j].cod_serv
  PARA k ← 1 ATÉ 4 FAÇA
  INÍCIO
    SE tipos[k].cod = serv[i, j].cod_serv
    ENTÃO ESCREVA tipos[k].desc
    FIM
    ESCREVA serv[i, j].cod_cliente
    FIM
  FIM
  FIM
SE achou = 0
ENTÃO ESCREVA "Nenhum serviço prestado está entre os valores
➥ citados"
FIM
SE op = 5
ENTÃO INÍCIO
  achou ← 0
  PARA i ← 1 ATÉ 30 FAÇA
  INÍCIO
    PARA j ← 1 ATÉ 3 FAÇA
    INÍCIO
      SE (serv[i, j].num ≠ 0)
    ENTÃO INÍCIO
      achou ← 1
      ESCREVA "Dia ", i
      ESCREVA serv[i, j].num, serv[i, j].valor
      ESCREVA serv[i, j].cod_serv
      PARA k ← 1 ATÉ 4 FAÇA
      INÍCIO
        SE tipos[k].cod = serv[i, j].cod_serv
        ENTÃO ESCREVA tipos[k].desc
        FIM
        ESCREVA serv[i, j].cod_cliente
        FIM
      FIM
      FIM
    SE achou = 0
    ENTÃO ESCREVA "Nenhum serviço prestado foi cadastrado"
    FIM
  ATÉ op = 6
  FIM_ALGORITMO.

```



#### SOLUÇÃO:

\EXERC\CAP8\PASCAL\EX2.PAS e \EXERC\CAP8\PASCAL\EX2.EXE



#### SOLUÇÃO:

\EXERC\CAP8\C++\EX2.CPP e \EXERC\CAP8\C++\EX2.EXE

3. Faça um programa que utilize os registros a seguir:

CLIENTES	DOCUMENTOS
cod_cli	num_doc
nome	cod_cli
fone	data_venc
endereco	data_pag
	valor
	juros

Sabe-se que um documento só pode ser cadastrado para um cliente que já existe. Considere que podem existir no máximo 15 clientes e 10 documentos. Crie um vetor para clientes e outro para documentos. Crie um menu para a realização de cada uma das operações especificadas a seguir.

- Cadastrar clientes – não pode existir mais de um cliente com o mesmo código.
- Cadastrar documentos – ao cadastrar um documento, se a data de pagamento for maior que a data de vencimento, calcular o campo 'juros' do registro documentos (5% sobre o valor original do documento).
- Excluir clientes – um cliente só pode ser excluído se não existir documento algum associado a ele.
- Excluir documentos individuais – por meio de seu número. Caso o documento não exista, mostre a mensagem *Documento não encontrado*.
- Excluir documentos por cliente – informar o código do cliente e excluir todos os seus documentos. Caso o cliente não exista, mostre a mensagem *Cliente não encontrado*.
- Excluir documentos por período – informar a data inicial e a data final e excluir todos os documentos que possuírem data de vencimento nesse período.
- Alterar as informações sobre os clientes – só não pode ser alterado o código do cliente.
- Mostrar o total de documentos de um determinado cliente.

**OBSERVAÇÃO:**

Quando forem excluídos clientes *ou* documentos, os vetores devem ser reorganizados, ou seja, todas as posições não preenchidas dos vetores devem ficar no final. Exemplo: se for necessário excluir o número 8 do vetor a seguir, tanto o 9 quanto o 1 devem ser movidos uma casa para a esquerda e a última posição deve ficar livre para uma nova inclusão.

Vetor inicial



Vetor reordenado com uma posição livre ao final

**SOLUÇÃO:**

## ALGORITMO

```
DECLARE clientes[15] REGISTRO (cod_cli NUMÉRICO, nome, fone, ende
    ↪ LITERAL)
docs[30] REGISTRO (num_doc, cod_cli, dv, mv, av, dp, mp, ap,
    ↪ valor, juros NUMÉRICO)
```

```

    posi, op, i, cliente_livre, doc_livre NUMÉRICO
    achou, codigo, numero, diav, mesv, anov NUMÉRICO
    diap, mesp, anop, valor, juros, total NUMÉRICO
    nome, fone, ende LITERAL
PARA i ← 1 ATÉ 15 FAÇA
INÍCIO
    clientes[i].cod_cli ← 0
    clientes[i].nome ← "
    clientes[i].fone ← "
    clinetes[i].ende ← "
FIM
    cliente_livre ← 1
PARA i ← 1 ATÉ 30 FAÇA
INÍCIO
    docs[i].num_doc ← 0
    docs[i].cod_cli ← 0
    docs[i].dv ← 0
    docs[i].mv ← 0
    docs[i].av ← 0
    docs[i].dp ← 0
    docs[i].mp ← 0
    docs[i].ap ← 0
    docs[i].valor ← 0
    docs[i].juros ← 0
FIM
    doc_livre ← 1
REPITA
    ESCREVA "Menu de Opções"
    ESCREVA "1 - Cadastrar clientes"
    ESCREVA "2 - Cadastrar documentos"
    ESCREVA "3 - Excluir clientes"
    ESCREVA "4 - Excluir documentos individuais"
    ESCREVA "5 - Excluir documentos por cliente"
    ESCREVA "6 - Excluir documentos por período"
    ESCREVA "7 - Alterar clientes"
    ESCREVA "8 - Totalizar documentos"
    ESCREVA "9 - Sair"
    ESCREVA "Digite sua opção"
LEIA op
SE (op < 1) OU (op > 9)
ENTÃO ESCREVA "Opção inválida"
SE op = 1
ENTÃO INÍCIO
    SE cliente_livre = 16
    ENTÃO ESCREVA "Cadastro de clientes lotado"
    SENÃO INÍCIO
        ESCREVA "Digite o código do cliente a ser cadastrado"
        LEIA codigo
        achou ← 0
        PARA i ← 1 ATÉ 15 FAÇA
        INÍCIO
            SE clientes[i].cod_cli = codigo
            ENTÃO achou ← 1
            FIM
            SE achou = 1
            ENTÃO ESCREVA "Já existe cliente cadastrado com esse
            ↪ código"
            SENÃO INÍCIO
                ESCREVA "Digite o nome do cliente"
                LEIA nome
                ESCREVA "Digite o telefone do cliente"
                LEIA fone

```

```

        ESCREVA "Digite o endereço do cliente"
        LEIA ende
        clientes[cliente_livre].cod_cli ← codigo
        clientes[cliente_livre].nome ← nome
        clientes[cliente_livre].fone ← fone
        clientes[cliente_livre].ende ← ende
        ESCREVA "Cliente cadastrado com sucesso"
        cliente_livre ← cliente_livre + 1
        FIM
    FIM
SE op = 2
ENTÃO INÍCIO
SE doc_livre = 31
ENTÃO ESCREVA "Cadastro de documentos lotado"
SENÃO INÍCIO
    ESCREVA "Digite o número do documento a ser cadastrado"
    LEIA numero
    achou ← 0
    PARA i ← 1 ATÉ 30 FAÇA
    INÍCIO
        SE docs[i].num_doc = numero
        ENTÃO achou ← 1
    FIM
    SE achou = 1
        ENTÃO ESCREVA "Já existe um documento cadastrado com
        esse código"
    SENÃO INÍCIO
        ESCREVA "Digite o código do cliente dono do
        documento"
        LEIA codigo
        achou ← 0
        PARA i ← 1 ATÉ 15 FAÇA
        INÍCIO
            SE clientes[i].cod_cli = codigo
            ENTÃO achou ← 1
        FIM
        SE achou = 0
            ENTÃO ESCREVA "Não existe cliente cadastrado com
            esse código"
        SENÃO INÍCIO
            ESCREVA "Digite a data do vencimento do
            documento"
            LEIA diav, mesv,anov
            ESCREVA "Digite a data do pagamento do
            documento"
            LEIA diap, mesp,anop
            ESCREVA "Digite o valor do documento"
            LEIA valor
            SE anop > anov
            ENTÃO juros ← valor * 5%
            SENÃO SE mesp > mesv
                ENTÃO juros ← valor * 5%
                SENÃO SE diap > diav
                    ENTÃO juros ← valor * 5%
                    SENÃO juros ← 0
            docs[doc_livre].num_doc ← numero
            docs[doc_livre].cod_cli ← codigo
            docs[doc_livre].dv ← diav
            docs[doc_livre].mv ← mesv
            docs[doc_livre].av ← anov
            docs[doc_livre].dp ← diap

```

```

        docs[doc_livre].mp ← mesp
        docs[doc_livre].ap ← anop
        docs[doc_livre].valor ← valor
        docs[doc_livre].juros ← juros
        ESCREVA "Documento cadastrado com sucesso"
        doc_livre ← doc_livre + 1
        FIM
    FIM
FIM
SE op = 3
ENTÃO INÍCIO
    ESCREVA "Digite o código do cliente a ser excluído"
    LEIA codigo
    achou ← 0
    PARA i ← 1 ATÉ 15 FAÇA
    INÍCIO
        SE clientes[i].cod_cli = codigo
        ENTÃO INÍCIO
            achou ← 1
            posi ← i
            FIM
        FIM
    SE achou = 0
    ENTÃO ESCREVA "Não existe cliente cadastrado com esse código"
    SENÃO INÍCIO
        achou ← 0
        PARA i ← 1 ATÉ 30 FAÇA
        INÍCIO
            SE docs[i].cod_cli = codigo
            ENTÃO achou ← 1
            FIM
        SE achou = 1
        ENTÃO ESCREVA "Este cliente não pode ser excluído,
        → possui documento"
        SENÃO INÍCIO
            PARA i ← posi ATÉ (cliente_livre-2) FAÇA
            INÍCIO
                clientes[i].cod_cli ← clientes[i+1].cod_cli
                clientes[i].nome ← clientes[i+1].nome
                clientes[i].fone ← clientes[i+1].fone
                clientes[i].ende ← clientes[i+1].ende
            FIM
            clientes[cliente_livre - 1].cod_cli ← 0
            clientes[cliente_livre - 1].nome ← "
            clientes[cliente_livre - 1].fone ← "
            clientes[cliente_livre - 1].ende ← "
            cliente_livre ← cliente_livre - 1
            ESCREVA "Cliente excluído com sucesso"
            FIM
        FIM
    FIM
SE op = 4
ENTÃO INÍCIO
    ESCREVA "Digite o número do documento a ser excluído"
    LEIA numero
    achou ← 0
    PARA i ← 1 ATÉ 30 FAÇA
    INÍCIO
        SE docs[i].num_doc = numero
        ENTÃO INÍCIO
            achou ← 1

```



```

        docs[i].mv ← docs[i+1].mv
        docs[i].av ← docs[i+1].av
        docs[i].dp ← docs[i+1].dp
        docs[i].mp ← docs[i+1].mp
        docs[i].ap ← docs[i+1].ap
        docs[i].valor ← docs[i+1].valor
        docs[i].juros ← docs[i+1].juros
    FIM
    docs[doc_livre - 1].num_doc ← 0
    docs[doc_livre - 1].cod_cli ← 0
    docs[doc_livre - 1].dv ← 0
    docs[doc_livre - 1].mv ← 0
    docs[doc_livre - 1].av ← 0
    docs[doc_livre - 1].dp ← 0
    docs[doc_livre - 1].mp ← 0
    docs[doc_livre - 1].ap ← 0
    docs[doc_livre - 1].valor ← 0
    docs[doc_livre - 1].juros ← 0
    doc_livre ← doc_livre - 1
    k ← 1
    FIM
SENÃO k ← k + 1
FIM
SE achou = 1
ENTÃO ESCREVA "Documentos excluídos com sucesso"
SENÃO ESCREVA "Não existe documento para este
    cliente"
FIM
FIM
SE op = 6
ENTÃO INÍCIO
    ESCREVA "Digite a data inicial dos documentos que serão
    ↪ excluídos"
    LEIA dia_inicial, mes_inicial, ano_inicial
    ESCREVA "Digite a data final dos documentos que serão
    ↪ excluídos"
    LEIA dia_final, mes_final, ano_final
    achou ← 0
    PARA i ← 1 ATÉ 30 FAÇA
    INÍCIO
        SE (docs[i].av ≥ ano_inicial) E (docs[i].av ≤ ano_final)
        ENTÃO SE (docs[i].mv ≥ mes_inicial) E (docs[i].mv ≤ mes_final)
            ENTÃO SE (docs[i].dv ≥ dia_inicial)
                E (docs[i].dv ≤ dia_final)
            ENTÃO INÍCIO
                posi ← i
                achou ← 1
            FIM
        FIM
    SE achou = 1
    ENTÃO INÍCIO
        PARA j ← posi ATÉ (doc_livre - 2) FAÇA
        INÍCIO
            docs[j].num_doc ← docs[j+1].num_doc
            docs[j].cod_cli ← docs[j+1].cod_cli
            docs[j].dv ← docs[j+1].dv
            docs[j].mv ← docs[j+1].mv
            docs[j].av ← docs[j+1].av
            docs[j].dp ← docs[j+1].dp
            docs[j].mp ← docs[j+1].mp
            docs[j].ap ← docs[j+1].ap
            docs[j].valor ← docs[j+1].valor

```

```

docs[j].juros ← docs[j+1].juros
FIM
docs[doc_livre - 1].num_doc ← 0
docs[doc_livre - 1].cod_cli ← 0
docs[doc_livre - 1].dv ← 0
docs[doc_livre - 1].mv ← 0
docs[doc_livre - 1].av ← 0
docs[doc_livre - 1].dp ← 0
docs[doc_livre - 1].mp ← 0
docs[doc_livre - 1].ap ← 0
docs[doc_livre - 1].valor ← 0
docs[doc_livre - 1].juros ← 0
doc_livre ← doc_livre - 1
FIM
SE achou = 0
ENTÃO ESCREVA "Não existe documento cadastrado neste período"
SENÃO ESCREVA "Documentos do período excluídos com sucesso"
FIM
SE op = 7
ENTÃO INÍCIO
    ESCREVA "Digite o código do cliente a ser alterado"
    LEIA codigo
    achou ← 0
    PARA i ← 1 ATÉ 15 FAÇA
    INÍCIO
        SE clientes[i].cod_cli = codigo
        ENTÃO INÍCIO
            achou ← 1
            posi ← i
            FIM
        FIM
        SE achou = 0
        ENTÃO ESCREVA "Não existe cliente cadastrado com esse código
        ➔ para ser alterado"
        SENÃO INÍCIO
            ESCREVA "Digite o novo nome do cliente"
            LEIA nome
            ESCREVA "Digite o novo telefone do cliente"
            LEIA fone
            ESCREVA "Digite o novo endereço do cliente"
            LEIA ende
            clientes[posi].nome ← nome
            clientes[posi].fone ← fone
            clientes[posi].ende ← ende
            ESCREVA "Cliente alterado com sucesso"
            FIM
        FIM
    SE op = 8
    ENTÃO INÍCIO
        ESCREVA "Digite o código do cliente do qual deseja totalizar
        ➔ os documentos"
        LEIA codigo
        achou ← 0
        PARA i ← 1 ATÉ 15 FAÇA
        INÍCIO
            SE clientes[i].cod_cli = codigo
            ENTÃO achou ← 1
            FIM
        SE achou = 0
        ENTÃO ESCREVA "Não existe cliente cadastrado com esse código"
        SENÃO INÍCIO
            total ← 0
            PARA i ← 1 ATÉ 30 FAÇA

```

```

INÍCIO
SE docs[i].cod_cli = codigo
ENTÃO INÍCIO
    total ← total + docs[i].valor
    total ← total + docs[i].juros
    FIM
    FIM
    ESCREVA "Total dos documentos do cliente de código ",
    ↪ código, " = ", total
    FIM
    FIM
ATÉ op = 9
FIM_ALGORITMO.

```



#### SOLUÇÃO:

\EXERC\CAP8\PASCAL\EX3.PAS e \EXERC\CAP8\PASCAL\EX3.EXE



#### SOLUÇÃO:

\EXERC\C++\EX3.CPP e \EXERC\C++\EX3.EXE

- 4.** Faça um programa que efetue reserva de passagens aéreas de uma determinada companhia. O programa deverá ler os números dos aviões e o número de lugares disponíveis em cada avião. Utilize um vetor de quatro posições, onde cada posição representa um avião, e um outro vetor também de quatro posições para armazenar os lugares disponíveis.

Mostre o seguinte menu de opções:

1. Cadastrar os números dos aviões
2. Cadastrar o número de lugares disponíveis em cada avião
3. Reservar passagem
4. Consultar por avião
5. Consultar por passageiro
6. Finalizar

Imagine que poderão ser registradas até 60 reservas e que cada reserva deverá possuir o número do avião e o nome do passageiro.

**Para realizar a opção 1,** deverá ser solicitado ao usuário o número dos quatro aviões disponíveis.

**Para realizar a opção 2,** deverá ser solicitado ao usuário o número de lugares disponíveis em cada avião cadastrado na opção 1.

**Para realizar a opção 3,** deverá ser verificado se o número do avião digitado é válido. Posteriormente, deverá ser verificado se, no avião escolhido, ainda existe lugar disponível. Caso exista, diminua o total de vagas e mostre a mensagem *Reserva confirmada*. Caso contrário, mostre a mensagem *Vôo lotado*. Observe que não podem ser feitas mais de 600 reservas.

**Para realizar a opção 4,** deverá ser solicitado o número do avião desejado e, posteriormente, exibidas todas as suas reservas.

**Para realizar a opção 5,** deverá ser solicitado o nome do passageiro e, posteriormente, exibidas todas as reservas feitas em seu nome.

**A opção 6,** termina o programa.

**ALGORITMO****SOLUÇÃO:**

```

ALGORITMO
DECLARE avi[4], lug[4] NUMÉRICO
      reservas[60] REGISTRO (num_avi NUMÉRICO, nome LITERAL)
      i, pos_livre, op, achou, numero, posi NUMÉRICO
      nome LITERAL
PARA i ← 1 ATÉ 4 FAÇA
INÍCIO
avi[i] ← 0
lug[i] ← 0
FIM
PARA i ← 1 ATÉ 60 FAÇA
INÍCIO
reservas[i].num_avi ← 0
reservas[i].nome ← ''
FIM
pos_livre ← 1
REPITA
ESCREVA "Menu de Opções"
ESCREVA "1 - Cadastrar os números dos aviões"
ESCREVA "2 - Cadastrar os lugares disponíveis em cada avião"
ESCREVA "3 - Reservar passagem"
ESCREVA "4 - Consultar pelo número do avião"
ESCREVA "5 - Consultar pelo nome do passageiro"
ESCREVA "6 - Finalizar"
ESCREVA "Digite a opção desejada"
LEIA op
SE op = 1
ENTÃO INÍCIO
PARA i ← 1 ATÉ 4 FAÇA
INÍCIO
ESCREVA "Digite o número do ", i, "º avião"
LEIA avi[i]
FIM
FIM
SE op = 2
ENTÃO INÍCIO
PARA i ← 1 ATÉ 4 FAÇA
INÍCIO
ESCREVA "Digite o número de lugares disponíveis no ", i, "º
    avião"
LEIA lug[i]
FIM
FIM
SE op = 3
ENTÃO INÍCIO
ESCREVA "Digite o número do avião no qual deseja efetuar a
    reserva"
LEIA numero
SE pos_livre > 60
ENTÃO ESCREVA "Reservas em todos os aviões esgotadas"
SENÃO INÍCIO
    achou ← 0
    PARA i ← 1 ATÉ 4 FAÇA
    INÍCIO
        SE avi[i] = numero
        ENTÃO INÍCIO
            achou ← 1
            posi ← i
            FIM
        FIM
        SE achou = 0

```

```

ENTÃO ESCREVA "Não existe este avião"
SENÃO SE lug[posi] = 0
    ENTÃO ESCREVA "Avião lotado"
    SENÃO INÍCIO
        ESCREVA "Digite o nome do passageiro"
        LEIA nome
        reservas[pos_livre].num_avi ← numero
        reservas[pos_livre].nome ← nome
        ESCREVA "Reserva efetuada com sucesso"
        pos_livre ← pos_livre + 1
        lug[posi] ← lug[posi] - 1
    FIM
FIM
SE op = 4
ENTÃO INÍCIO
    ESCREVA "Digite o número do avião do qual deseja consultar as"
    ↪ reservas"
    LEIA numero
    achou ← 0
    PARA i ← 1 ATÉ 4 FAÇA
    INÍCIO
        SE avi[i] = numero
        ENTÃO achou ← 1
    FIM
    SE achou = 0
        ENTÃO ESCREVA "Não existe este avião"
    SENÃO INÍCIO
        achou ← 0
        PARA i ← 1 ATÉ (pos_livre - 1) FAÇA
        INÍCIO
            SE reservas[i].num_avi = numero
            ENTÃO INÍCIO
                ESCREVA reservas[i].nome
                achou ← 1
            FIM
        FIM
        SE achou = 0
            ENTÃO ESCREVA "Nenhuma reserva está cadastrada para este"
            ↪ avião"
        FIM
    FIM
SE op = 5
ENTÃO INÍCIO
    ESCREVA "Digite o nome do passageiro do qual deseja consultar"
    ↪ as reservas"
    LEIA nome
    achou ← 0
    PARA i ← 1 ATÉ (pos_livre - 1) FAÇA
    INÍCIO
        SE reservas[i].nome = nome
        ENTÃO INÍCIO
            ESCREVA reservas[i].num_avi
            achou ← 1
        FIM
    FIM
    SE achou = 0
        ENTÃO ESCREVA "Nenhuma reserva está cadastrada para este nome"
    FIM
ATÉ op = 6
FIM_ALGORITMO.

```

**SOLUÇÃO:**

```
\EXERC\CAP8\PASCAL\EX4.PAS e \EXERC\CAP8\PASCAL\EX4.EXE
```

**SOLUÇÃO:**

```
\EXERC\CAP8\C++\EX4.CPP e \EXERC\CAP8\C++\EX4.EXE
```

- 5.** Uma empresa possui 18 funcionários com as seguintes características: nome, número de horas trabalhadas no mês, turno de trabalho (pode ser M – Matutino, V – Vespertino ou N – Noturno), categoria (pode ser O – Operário ou G – Gerente) e valor da hora trabalhada. Sabendo-se que essa empresa deseja informatizar sua folha de pagamento, faça um programa que leia o nome, o número de horas trabalhadas no mês, o turno e a categoria dos funcionários, não permitindo que sejam informados turnos e categorias inexistentes. Calcule o valor da hora trabalhada, conforme a tabela a seguir, e adotando o valor de R\$ 112,00 para o salário mínimo.

CATEGORIA	TURNO	VALOR DA HORA TRABALHADA
G	N	18% do salário mínimo
G	M ou V	15% do salário mínimo
O	N	13% do salário mínimo
O	M ou V	10% do salário mínimo

Calcule o salário inicial dos funcionários, com base no valor da hora e no número de horas trabalhadas. Todos os funcionários recebem um auxílio-alimentação, de acordo com o seu salário inicial, conforme a tabela a seguir:

SALÁRIO INICIAL	AUXÍLIO-ALIMENTAÇÃO
≤ R\$ 300,00	20% do salário inicial
> R\$ 300,00 e < R\$ 600,00	15% do salário inicial
≥ R\$ 600,00	5% do salário inicial

Mostre o nome, o número de horas trabalhadas, o valor da hora trabalhada, o salário inicial, o auxílio-alimentação e o salário final (salário inicial + auxílio-alimentação) de todos os funcionários.

O programa deve apresentar o seguinte menu de opções:

1. Cadastrar funcionários
2. Mostrar folha de pagamento
3. Sair

**ALGORITMO**  
**SOLUÇÃO:**

```

ALGORITMO
DECLARE func[18] REGISTRO (num_horas_trab, valor_hora NUMÉRICO,
                            nome, turno, cat LITERAL)
                            i, pos_livre, op, sal_minimo, sal_inicial
                            aux_alim, sal_final NUMÉRICO
PARA i ← 1 ATÉ 18 FAÇA
INÍCIO
func[i].num_horas_trab ← 0
func[i].valor_hora ← 0
func[i].nome ← "

```

```

func[i].turno ← "
func[i].cat ← "
FIM
pos_livre ← 1
REPITA
ESCREVA "Menu de Opções"
ESCREVA "1 - Cadastrar funcionários"
ESCREVA "2 - Mostrar folha de pagamento"
ESCREVA "3 - Sair"
ESCREVA "Digite a opção desejada"
LEIA op
SE (op ≠ 1) E (op ≠ 2) E (op ≠ 3)
ENTÃO ESCREVA "Opção Inválida"
SE op = 1
ENTÃO INÍCIO
    SE pos_livre = 19
        ENTÃO ESCREVA "Cadastro de funcionários lotado"
        SENÃO INÍCIO
            sal_minimo ← 112
            ESCREVA "Digite o nome do funcionário que deseja"
            ↪ incluir"
            LEIA func[pos_livre].nome
            ESCREVA "Digite o número de horas trabalhadas"
            LEIA func[pos_livre].num_horas_trab
            ESCREVA "Digite o turno de trabalho"
            LEIA func[pos_livre].turno
            ENQUANTO (func[pos_livre].turno ≠ 'M')
                E (func[pos_livre].turno ≠ 'V')
                E (func[pos_livre].turno ≠ 'N') FAÇA
            INÍCIO
                ESCREVA "Turno inválido. Digite novamente"
                LEIA func[pos_livre].turno
            FIM
            ESCREVA "Digite a categoria"
            LEIA func[pos_livre].cat
            ENQUANTO (func[pos_livre].cat ≠ "O") E
                ↪ (func[pos_livre].cat ≠ "G") FAÇA
            INÍCIO
                ESCREVA "Categoria inválida. Digite novamente"
                LEIA func[pos_livre].cat
            FIM
            SE func[pos_livre].cat = "G"
            ENTÃO SE func[pos_livre].turno = "N"
                ENTÃO func[pos_livre].valor_hora ← sal_minimo * 18%
                SENÃO func[pos_livre].valor_hora ← sal_minimo * 15%
            SE func[pos_livre].cat = "O"
            ENTÃO SE func[pos_livre].turno = "N"
                ENTÃO func[pos_livre].valor_hora ← sal_minimo * 13%
                SENÃO func[pos_livre].valor_hora ← sal_minimo * 10%
            ESCREVA "Funcionário cadastrado com sucesso"
            pos_livre ← pos_livre + 1
            FIM
        FIM
    SE op = 2
    ENTÃO INÍCIO
        ESCREVA "Folha de Pagamento"
        SE pos_livre = 1
        ENTÃO ESCREVA "Não existe funcionário cadastrado"
        SENÃO INÍCIO
            PARA i ← 1 ATÉ (pos_livre - 1) FAÇA
            INÍCIO
                ESCREVA func[i].nome, func[i].num_horas_trab,

```

```

    ↵ func[i].valor_hora
    sal_inicial ← func[i].num_horas_trab *
    ↵ func[i].valor_hora
    ESCREVA sal_inicial
    SE sal_inicial ≤ 300
        ENTÃO aux_alim ← sal_inicial * 20%
        SENÃO SE sal_inicial < 600
            ENTÃO aux_alim ← sal_inicial * 15%
            SENÃO aux_alim ← sal_inicial * 5%
    ESCREVA aux_alim
    sal_final ← sal_inicial + aux_alim
    ESCREVA sal_final
    FIM
    FIM
    FIM
    FIM
ATÉ op = 3
FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP8\PASCAL\EX5.PAS e \EXERC\CAP8\PASCAL\EX5.EXE

**SOLUÇÃO:**

\EXERC\CAP8\C++\EX5.CPP e \EXERC\CAP8\C++\EX5.EXE

**6.** Uma empresa contratou 15 funcionários temporários. De acordo com o valor das vendas mensais, os funcionários adquirem pontos que determinarão seus salários ao final de cada mês. Sabe-se que esses funcionários trabalharão nos meses de novembro de 2000 a janeiro de 2001. Faça um programa que:

- cadastre os nomes dos funcionários e suas respectivas vendas mensais;
- calcule e mostre a pontuação geral de cada funcionário nos três meses. Sabe-se que R\$ 100,00 equivalem a 1 ponto;
- calcule e mostre a pontuação geral de todos os funcionários a cada mês;
- determine e mostre a maior pontuação atingida nos três meses, mostrando o nome do funcionário. Desconsiderar empates;
- determine e mostre o valor total vendido.

**SOLUÇÃO:**

```

ALGORITMO
DECLARE func[15] REGISTRO (nome LITERAL, venda_nov, venda_dez,
    ↵ venda_jan NUMÉRICO)
    i, pontos, maior, pos_maior, mes, valor_total NUMÉRICO
ESCREVA "CADASTRANDO OS FUNCIONÁRIOS"
PARA i ← 1 ATÉ 15 FAÇA
INÍCIO
ESCREVA "Digite o nome do ",i, "º funcionário"
LEIA func[i].nome
ESCREVA "Digite o valor vendido no mês de novembro pelo ",i, "º :
    ↵ funcionário"
LEIA func[i].venda_nov
ESCREVA "Digite o valor vendido no mês de dezembro pelo ",i, "º :
    ↵ funcionário"
LEIA func[i].venda_dez
ESCREVA "Digite o valor vendido no mês de janeiro pelo ",i, "º :
    ↵ funcionário"

```

```
LEIA func[i].venda_jan
FIM
ESCREVA "MOSTRANDO AS PONTUAÇÕES MENSAIS DE CADA FUNCIONÁRIO"
PARA i ← 1 ATÉ 15 FAÇA
INÍCIO
ESCREVA "Funcionário: ", func[i].nome
pontos ← func[i].venda_nov/100
ESCREVA "Pontos de novembro = ", pontos
pontos ← func[i].venda_dez/100
ESCREVA "Pontos de dezembro = ", pontos
pontos ← func[i].venda_jan/100
ESCREVA "Pontos de janeiro = ", pontos
pontos ← func[i].venda_nov/100 + func[i].venda_dez/100 +
➥ func[i].venda_jan/100
ESCREVA "Total de pontos = ", pontos
FIM
ESCREVA "MOSTRANDO A PONTUAÇÃO TOTAL DO MÊS DE NOVEMBRO"
pontos ← 0
PARA i ← 1 ATÉ 15 FAÇA
INÍCIO
pontos ← pontos + func[i].venda_nov/100
FIM
ESCREVA pontos
ESCREVA "MOSTRANDO A PONTUAÇÃO TOTAL DO MÊS DE DEZEMBRO"
pontos ← 0
PARA i ← 1 ATÉ 15 FAÇA
INÍCIO
pontos ← pontos + func[i].venda_dez/100
FIM
ESCREVA pontos
ESCREVA "MOSTRANDO A PONTUAÇÃO TOTAL DO MÊS DE JANEIRO"
pontos ← 0
PARA i ← 1 ATÉ 15 FAÇA
INÍCIO
pontos ← pontos + func[i].venda_jan/100
FIM
ESCREVA pontos
ESCREVA "MOSTRANDO A MAIOR PONTUAÇÃO"
maior ← 0
PARA i ← 1 ATÉ 15 FAÇA
INÍCIO
SE func[i].venda_nov/100 > maior
ENTÃO INÍCIO
    maior ← func[i].venda_nov/100
    pos_maior ← i
    mes ← 1
    FIM
SE func[i].venda_dez/100 > maior
ENTÃO INÍCIO
    maior ← func[i].venda_dez/100
    pos_maior ← i
    mes ← 2
    FIM
SE func[i].venda_jan/100 > maior
ENTÃO INÍCIO
    maior ← func[i].venda_jan/100
    pos_maior ← i
    mes ← 3
    FIM
FIM
ESCREVA "Funcionário: ", func[pos_maior].nome
ESCREVA "Maior pontuação: ", maior
```

```

SE mes = 1
ENTÃO ESCREVA "No mês de novembro"
SE mes = 2
ENTÃO ESCREVA "No mês de dezembro"
SE mes = 3
ENTÃO ESCREVA "No mês de janeiro"
ESCREVA "MOSTRANDO O VALOR TOTAL VENDIDO"
pontos ← 0
PARA i ← 1 ATÉ 15 FAÇA
INÍCIO
pontos ← pontos + func[i].venda_nov/100 + func[i].venda_dez/100 +
func[i].venda_jan/100
FIM
ESCREVA "Total vendido = ", pontos
FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP8\PASCAL\EX6.PAS e \EXERC\CAP8\PASCAL\EX6.EXE

**SOLUÇÃO:**

\EXERC\CAP8\C++\EX6.CPP e \EXERC\CAP8\C++\EX6.EXE

7. Faça um programa para ler o código, o sexo (M – Masculino; F – Feminino) e o número de horas/aula dadas no mês dos professores de uma escola, sabendo que um professor ganha R\$ 12,50 hora/aula e que a escola possui dez professores. Após a leitura, mostre:

- uma listagem contendo o código, o salário bruto, o desconto e o salário líquido de todos os professores;
- a média aritmética dos salários brutos dos professores do sexo masculino;
- a média aritmética dos salários brutos dos professores do sexo feminino.

Os descontos devem ser assim calculados:

<b>SEXO</b>	<b>ATÉ 70 HORAS/</b>		<b>MAIS QUE 70 HORAS/</b>	
	<b>AULA AO MÊS</b>		<b>AULA AO MÊS</b>	
Masculino	10%			8%
Feminino		7%		5%

**SOLUÇÃO:**

```

ALGORITMO
DECLARE prof[10] REGISTRO (cod, num_aula NUMÉRICO, sexo LITERAL)
    i, sal_bruto, desc, sal_liq, ma_masc, ma_fem NUMÉRICO
    soma_masc, soma_fem, qt_masc, qt_fem NUMÉRICO
PARA i ← 1 ATÉ 10 FAÇA
INÍCIO
prof[i].cod ← 0
prof[i].num_aula ← 0
prof[i].sexo ← "
FIM
soma_masc ← 0
soma_fem ← 0
qt_masc ← 0
qt_fem ← 0

```

```

ESCREVA "Digitando os dados dos 10 professores"
PARA i ← 1 ATÉ 10 FAÇA
INÍCIO
  ESCREVA i, "º professor"
  ESCREVA "Digite o código"
  LEIA prof[i].cod
  ESCREVA "Digite o número de aulas"
  LEIA prof[i].num_aula
  ESCREVA "Digite o sexo"
  LEIA prof[i].sexo
  FIM
  ESCREVA "Mostrando a listagem com os salários dos professores"
  PARA i ← 1 ATÉ 10 FAÇA
  INÍCIO
    ESCREVA prof[i].cod
    sal_bruto ← 12.50 * prof[i].num_aula
    ESCREVA sal_bruto
    SE prof[i].sexo = "F"
      ENTÃO INÍCIO
        SE prof[i].num_aula ≤ 70
        ENTÃO desc ← sal_bruto * 7%
        SENÃO desc ← sal_bruto * 5%
        FIM
      SENÃO INÍCIO
        SE prof[i].num_aula ≤ 70
        ENTÃO desc ← sal_bruto * 10%
        SENÃO desc ← sal_bruto * 8%
        FIM
      ESCREVA desc
      sal_liq ← sal_bruto - desc
      ESCREVA sal_liq
      SE prof[i].sexo = "F"
      ENTÃO INÍCIO
        soma_fem ← soma_fem + sal_bruto
        qt_fem ← qt_fem + 1
        FIM
      SENÃO INÍCIO
        soma_masc ← soma_masc + sal_bruto
        qt_masc ← qt_masc + 1
        FIM
      FIM
    SE qt_fem = 0
    ENTÃO ma_fem ← 0
    SENÃO ma_fem ← soma_fem / qt_fem
    SE qt_masc = 0
    ENTÃO ma_masc ← 0
    SENÃO ma_masc ← soma_masc / qt_masc
    ESCREVA "Média dos salários brutos dos professores do sexo feminino"
    ↳ = ",ma_fem
    ESCREVA "Média dos salários brutos dos professores do sexo masculino"
    ↳ = ",ma_masc
  FIM_ALGORITMO.

```



#### SOLUÇÃO:

\EXERC\CAP8\PASCAL\EX7.PAS e \EXERC\CAP8\PASCAL\EX7.EXE



#### SOLUÇÃO:

\EXERC\CAP8\C++\EX7.CPP e \EXERC\CAP8\C++\EX7.EXE

8. A seguir, vê-se os campos de alguns registros:

Professor

(número de registro, nome, cod\_título, total h/a semanal)

Título

(cod\_título, descrição, valor hora/aula)

- ◆ Crie uma rotina para cadastrar informações relacionadas a título. Sabe-se que nessa escola existem cinco títulos.
- ◆ Crie uma rotina para cadastrar os professores. Sabe-se que nessa escola existem 14 professores, cada um deve estar associado a um título previamente cadastrado.
- ◆ Crie uma rotina para mostrar a relação de professores, conforme o layout a seguir.

Nº DO REGISTRO	NOME	TITULAÇÃO (DESCRIÇÃO)	VALOR HORA/AULA	TOTAL H/A SEMANAL	TOTAL GERAL SEMANAL
111	João da Silva	Mestre	R\$ 15,50	10	R\$ 155,00
113	Maria Oliveira	Especialista	R\$ 11,20	8	R\$ 89,60

### ALGORITMO

#### SOLUÇÃO:

```

ALGORITMO
DECLARE prof[14] REGISTRO (reg, cod_titulo, total_semanal NUMÉRICO,
                            ↪ nome LITERAL)
                    titulo[5] REGISTRO (cod_titulo, valor NUMÉRICO,
                            ↪ desc LITERAL)
                            i, j, total_geral, achou NUMÉRICO
PARA i ← 1 ATÉ 14 FAÇA
INÍCIO
prof[i].reg ← 0
prof[i].cod_titulo ← 0
prof[i].total_semanal ← 0
prof[i].nome ← "
FIM
PARA j ← 1 ATÉ 5 FAÇA
INÍCIO
titulo[j].cod_titulo ← 0
titulo[j].valor ← 0
titulo[j].desc ← "
FIM
ESCREVA "Cadastrando os 5 títulos"
PARA j ← 1 ATÉ 5 FAÇA
INÍCIO
ESCREVA "Digite o código do ", j , " º título"
LEIA titulo[j].cod_titulo
ESCREVA "Digite a descrição do ", j , " º título"
LEIA titulo[j].desc
ESCREVA "Digite o valor da hora aula do ", j , " º título "
LEIA titulo[j].valor
FIM
ESCREVA "Cadastrando os 14 professores"
PARA i ← 1 ATÉ 14 FAÇA
INÍCIO
ESCREVA "Digite o registro do ", i , " º professor"
LEIA prof[i].reg
ESCREVA "Digite o título do ", i , " º professor"
LEIA prof[i].cod_titulo
achou ← 0

```

```

ENQUANTO achou = 0 FAÇA
    INÍCIO
    PARA j ← 1 ATÉ 5 FAÇA
        INÍCIO
        SE titulo[j].cod_titulo = prof[i].cod_titulo
        ENTÃO achou ← 1
        FIM
        SE achou = 0
        ENTÃO INÍCIO
            ESCREVA "Título não cadastrado, digite novo título"
            LEIA prof[i].cod_titulo
            FIM
        FIM
    ESCREVA "Digite a carga horária semanal do ", i , " ° professor"
    LEIA prof[i].total_semanal
    ESCREVA "Digite o nome do ", i , " professor"
    LEIA prof[i].nome
    FIM
    ESCREVA "Mostrando a relação de professores"
    PARA i ← 1 ATÉ 14 FAÇA
    INÍCIO
        ESCREVA prof[i].reg, prof[i].nome
        PARA j ← 1 ATÉ 5 FAÇA
        INÍCIO
            SE prof[i].cod_titulo = titulo[j].cod_titulo
            ENTÃO INÍCIO
                ESCREVA titulo[j].desc, titulo[j].valor
                total_geral ← titulo[j].valor * prof[i].total_semanal
            FIM
        FIM
        ESCREVA prof[i].total_semanal, total_geral
    FIM
    FIM_ALGORITMO.

```


**SOLUÇÃO:**

\EXERC\CAP8\PASCAL\EX8.PAS e \EXERC\CAP8\PASCAL\EX8.EXE


**SOLUÇÃO:**

\EXERC\CAP8\C++\EX8.CPP e \EXERC\CAP8\C++\EX8.EXE

- 9.** Crie um pequeno sistema para controle automatizado de estoque, com os seguintes registros:

CLIENTES	NOTAS	ITENS_NOTAS	PRODUTOS
Cod_cliente	Numero_NF	Numero_NF	Cod_produto
Endereco	Cod_cliente	Cod_produto	Descricao
Telefone	Total_Geral	Quantidade	Unidade
		Preco_Venda	Preco_unitario
			Qtdade_estoque

O sistema deverá conter os seguintes módulos: CADASTROS, MOVIMENTAÇÕES, CONSULTAS, além de uma opção para SAÍDA.

1. O módulo CADASTROS deverá fazer manutenção das informações sobre clientes e produtos (seis produtos e três clientes):

- a) manutenção de CLIENTES – **inclusão**, tomando cuidado de não cadastrar dois clientes com o mesmo código; **alteração**, o único campo que não pode ser alterado é o código; **exclusão**, tomando cuidado para não permitir a exclusão de clientes que possuam nota fiscal.
- b) manutenção de PRODUTOS – **inclusão**, tomando cuidado de não cadastrar dois produtos com o mesmo código; **alteração**, o único campo que não pode ser alterado é o código; **exclusão**, tomando cuidado para não permitir a exclusão de produtos pertencentes a alguma nota fiscal.
2. O módulo MOVIMENTAÇÕES deverá permitir a **digitação** de notas fiscais de saída, de acordo com as especificações a seguir, supondo que poderão ser gravadas até cinco notas fiscais contendo dois itens em cada uma:
- não cadastrar duas notas com o mesmo número;
  - uma nota só pode ser emitida a um cliente que já exista;
  - todos os produtos da nota devem estar previamente cadastrados; caso contrário, emitir mensagem de erro;
  - não cadastrar duas vezes um produto na mesma nota;
  - quando um produto for confirmado, baixar sua quantidade em estoque e gravar um registro em ITENS\_NOTAS.
3. O módulo CONSULTAS deverá permitir as **consultas** descritas a seguir:
- todos os produtos com preços entre dois valores digitados pelo usuário;
  - todas as notas e os itens da nota de um cliente escolhido pelo usuário;
  - todas as notas e os itens da nota com total geral superior a um valor escolhido pelo usuário.

**ALGORITMO****SOLUÇÃO:**

```

ALGORITMO
DECLARE cliente[3] REGISTRO (cod_cliente NUMÉRICO, ende, fone
                                ↪ LITERAL)
        produto[6] REGISTRO (cod_produto, preco_unit, qtde_est
                                ↪ NUMÉRICO, desc, unid LITERAL)
        nota[5] REGISTRO (numero_nf, cod_cliente, total NUMÉRICO)
        itens_nota[10] REGISTRO (numero_nf, cod_prod, qtde,
                                ↪ preco_vend NUMÉRICO)
        i, j, k, h, cont, achou, op1, op2, posi, codigo_cli NUMÉRICO
        livre_cliente, livre_produto, livre_nota NUMÉRICO
        livre_item, codigo, pre, qtde NUMÉRICO
        valor, valor_inicial, valor_final NUMÉRICO
        endere, telefone, desc, unidade, resp LITERAL
PARA i ← 1 ATÉ 3 FAÇA
INÍCIO
cliente[i].cod_cliente ← 0
cliente[i].ende ← "
cliente[i].fone ← "
FIM
PARA i ← 1 ATÉ 6 FAÇA
INÍCIO
produto[i].cod_produto ← 0
produto[i].preco_unit ← 0
produto[i].qtde_est ← 0
produto[i].desc ← "
produto[i].unid ← "
FIM
PARA i ← 1 ATÉ 5 FAÇA

```

```

INÍCIO
nota[i].numero_nf ← 0
nota[i].cod_cliente ← 0
nota[i].total ← 0
FIM
PARA i ← 1 ATÉ 10 FAÇA
INÍCIO
itens_nota[i].numero_nf ← 0
itens_nota[i].cod_prod ← 0
itens_nota[i].qtde ← 0
itens_nota[i].preco_vend ← 0
FIM
livre_cliente ← 1
livre_produto ← 1
livre_nota ← 1
livre_item ← 1
REPITA
ESCREVA "Menu de Opções"
ESCREVA "1 - Cadastros"
ESCREVA "2 - Movimentações"
ESCREVA "3 - Consultas"
ESCREVA "4 - Sair"
ESCREVA "Digite sua opção"
LEIA op1
SE (op1 < 1) OU (op1 > 4)
ENTÃO ESCREVA "Opção inválida, digite novamente"
SE op1 = 1
ENTÃO BEGIN
    REPITA
        ESCREVA "Sub-menu de Opções"
        ESCREVA "1 - Incluir clientes"
        ESCREVA "2 - Alterar clientes"
        ESCREVA "3 - Excluir clientes"
        ESCREVA "4 - Incluir produtos"
        ESCREVA "5 - Alterar produtos"
        ESCREVA "6 - Excluir produtos"
        ESCREVA "7 - Sair"
        ESCREVA "Digite sua opção"
        LEIA op2
        SE (op2 < 1) OU (op2 > 7)
        ENTÃO ESCREVA "Opção inválida, digite novamente"
        SE op2 = 1
        ENTÃO INÍCIO
            ESCREVA "Inclusão de Clientes"
            SE livre_cliente = 4
            ENTÃO ESCREVA "Cadastro de clientes lotado"
            SENÃO INÍCIO
                ESCREVA "Digite o código do cliente a ser
                ↪ incluído"
                LEIA codigo
                achou ← 0
                PARA i ← 1 ATÉ 3 FAÇA
                INÍCIO
                    SE cliente[i].cod_cliente = codigo
                    ENTÃO achou ← 1
                    FIM
                    SE achou = 1
                    ENTÃO ESCREVA "Já existe cliente com este
                    ↪ código"
                    SENÃO INÍCIO
                        ESCREVA "Digite o endereço do cliente"
                        LEIA endere
                        ESCREVA "Digite o telefone do cliente"

```

```
LEIA telefone
cliente[livre_cliente].cod_cliente ←
↳ código
cliente[livre_cliente].ende ← endere
cliente[livre_cliente].fone ←
↳ telefone
ESCREVA "Cliente cadastrado com
↳ sucesso!"
livre_cliente ← livre_cliente + 1
FIM
FIM
FIM
SE op2 = 2
ENTÃO INÍCIO
    ESCREVA "Alteração de Clientes"
    SE livre_cliente = 1
        ENTÃO ESCREVA "Cadastro de clientes vazio"
        SENÃO INÍCIO
            ESCREVA "Digite o código do cliente a ser
            ↳ alterado"
            LEIA codigo
            achou ← 0
            PARA i ← 1 ATÉ 3 FAÇA
            INÍCIO
                SE cliente[i].cod_cliente = codigo
                ENTÃO INÍCIO
                    achou ← 1
                    posi ← i
                FIM
            FIM
            SE achou = 0
                ENTÃO ESCREVA "Não existe cliente com esse
                ↳ código"
                SENÃO INÍCIO
                    ESCREVA "Digite o novo endereço do
                    ↳ cliente"
                    LEIA endere
                    ESCREVA "Digite o novo telefone do
                    ↳ cliente"
                    LEIA telefone
                    cliente[posi].ende ← endere
                    cliente[posi].fone ← telefone
                    ESCREVA "Cliente alterado com
                    ↳ sucesso!"
                    FIM
                FIM
            FIM
        FIM
    FIM
    SE op2 = 3
        ENTÃO INÍCIO
            ESCREVA "Exclusão de Clientes"
            SE livre_cliente = 1
                ENTÃO ESCREVA "Cadastro de clientes vazio"
                SENÃO INÍCIO
                    ESCREVA "Digite o código do cliente a ser
                    ↳ excluído"
                    LEIA codigo
                    achou ← 0
                    PARA i ← 1 ATÉ 3 FAÇA
                    INÍCIO
                        SE cliente[i].cod_cliente = codigo
                        ENTÃO INÍCIO
                            achou ← 1
                            posi ← i
```

```

        FIM
        FIM
        SE achou = 0
        ENTÃO ESCREVA "Não existe cliente com este
        ↪ código"
        SENÃO INÍCIO
            achou ← 0
            PARA j ← 1 ATÉ (livre_nota - 1) FAÇA
            INÍCIO
                SE nota[j].cod_cliente = codigo
                ENTÃO achou ← 1
                FIM
                SE achou = 1
                ENTÃO INÍCIO
                    ESCREVA "Cliente não pode ser
                    ↪ excluído"
                    ESCREVA "possui notas"
                    FIM
                SENÃO INÍCIO
                    SE posi = 3
                    ENTÃO INÍCIO
                        cliente[j].cod_cliente
                        ↪ ← 0
                        cliente[j].ende ← "
                        cliente[j].fone ← "
                        FIM
                    PARA j ← posi ATÉ
                    ↪ (livre_cliente - 2) FAÇA
                    INÍCIO
                        cliente[j].cod_cliente←cliente[j+1].cod_cliente
                        cliente[j].ende ←
                        ↪ cliente[j+1].ende
                        cliente[j].fone ←
                        ↪ cliente[j+1].fone
                        FIM
                        cliente[livre_cliente-
                        ↪ 1].cod_cliente ← 0
                        cliente[livre_cliente-1].ende ←
                        ↪ ''
                        cliente[livre_cliente-1].fone ←
                        ↪ ''
                        ESCREVA "Cliente excluído com
                        ↪ sucesso!"
                        livre_cliente ← livre_cliente -
                        ↪ 1
                        FIM
                    FIM
                FIM
            FIM
        FIM
        SE op2 = 4
        ENTÃO INÍCIO
            ESCREVA "Inclusão de Produtos"
            SE livre_produto = 7
            ENTÃO ESCREVA "Cadastro de produtos lotado"
            SENÃO INÍCIO
                ESCREVA "Digite o código do produto a ser
                ↪ incluído"
                LEIA codigo
                achou ← 0
                PARA i ← 1 ATÉ 6 FAÇA
                INÍCIO
                    SE produto[i].cod_produto = codigo
                    ENTÃO achou ← 1

```

```
FIM
SE achou = 1
ENTÃO ESCREVA "Já existe produto com este
↳ código"
SENÃO INÍCIO
    ESCREVA "Digite a descrição do
    ↳ produto"
    LEIA desc
    ESCREVA "Digite a unidade do produto"
    LEIA unidade
    ESCREVA "Digite o preço unitário do
    ↳ produto"
    LEIA pre
    ESCREVA "Digite a quantidade de
    ↳ estoque do produto"
    LEIA qtde
    produto[livre_produto].cod_produto ←
    ↳ codigo
    produto [livre_produto].desc ← desc
    produto [livre_produto].unid ←
    ↳ unidade
    produto [livre_produto].preco_unit ←
    ↳ pre
    produto [livre_produto].qtde_est ←
    ↳ qtde
    ESCREVA "Produto cadastrado com
    ↳ sucesso!"
    livre_produto ← livre_produto + 1
FIM
FIM
FIM
SE op2 = 5
ENTÃO INÍCIO
    ESCREVA "Alteração de Produtos"
    SE livre_produto = 1
    ENTÃO ESCREVA "Cadastro de produtos vazio"
    SENÃO INÍCIO
        ESCREVA "Digite o código do produto a ser
        ↳ alterado"
        LEIA codigo
        achou ← 0
        PARA i ← 1 ATÉ 6 FAÇA
        INÍCIO
            SE cliente[i].cod_cliente = codigo
            ENTÃO INÍCIO
                achou ← 1
                posi ← i
            FIM
        FIM
        SE achou = 0
        ENTÃO ESCREVA "Não existe produto com este
        ↳ código"
        SENÃO INÍCIO
            ESCREVA "Digite a nova descrição do
            ↳ produto"
            LEIA desc
            ESCREVA "Digite a nova unidade do
            ↳ produto"
            LEIA unidade
            ESCREVA "Digite o novo preço unitário"
            LEIA pre
            ESCREVA "Digite a nova quantidade em
            ↳ estoque"
```

```

LEIA qtde
produto [livre_produto].desc ← desc
produto [livre_produto].unid ←
    ↪ unidade
produto [livre_produto].preco_unit ←
    ↪ pre
produto [livre_produto].qtde_est ←
    ↪ qtde
ESCREVA "Produto alterado com
    ↪ sucesso!"
FIM
FIM
FIM
SE op2 = 6
ENTÃO INÍCIO
    ESCREVA "Exclusão de Produtos"
    SE livre_produto = 1
        ENTÃO ESCREVA "Cadastro de produtos vazio"
        SENÃO INÍCIO
            ESCREVA "Digite o código do produto a ser
                ↪ excluído"
            LEIA codigo
            achou ← 0
            PARA i ← 1 ATÉ 6 FAÇA
            INÍCIO
                SE produto[i].cod_produto = codigo
                ENTÃO INÍCIO
                    achou ← 1
                    posi ← i
                FIM
            FIM
            SE achou = 0
                ENTÃO ESCREVA "Não existe produto com este
                    ↪ código"
                SENÃO INÍCIO
                    achou ← 0
                    PARA j ← 1 ATÉ (livre_item - 1) FAÇA
                    INÍCIO
                        SE itens_nota[j].cod_prod = codigo
                        ENTÃO achou ← 1
                    FIM
                    SE achou = 1
                    ENTÃO INÍCIO
                        ESCREVA "Não pode excluir
                            ↪ produto"
                        ESCREVA "está em notas"
                    FIM
                SENÃO INÍCIO
                    SE posi = 6
                    ENTÃO INÍCIO
                        produto [posi].cod_produto ← 0
                        produto [posi].desc ← "
                        produto [posi].unid ← "
                        produto [posi].preco_unit ← 0
                        produto [posi].qtde_est ← 0
                    FIM
                    PARA j ← posi ATÉ
                        ↪ (livre_produto - 2) FAÇA
                    INÍCIO

```

```

        produto [j].cod_produto ← produto
        ↪ [j+1].cod_produto
        produto [j].desc ← produto
        ↪ [j+1].desc
        produto [j].unid ← produto
        ↪ [j+1].unid
        produto [j].preco_unit ← produto
        ↪ [j+1].preco_unit
        produto [j].qtde_est ← produto
        ↪ [j+1].qtde_est
            FIM
        produto [livre_produto -
        ↪ 1].cod_produto ← 0
        produto [livre_produto - 1].desc ←
        ↪ "
        produto [livre_produto - 1].unid ←
        ↪ "
        produto [livre_produto -
        ↪ 1].preco_unit ← 0
        produto [livre_produto -
        ↪ 1].qtde_est ← 0
        ESCREVA "Produto excluído com
        ↪ sucesso!"
        livre_produto ← livre_produto - 1
    FIM
    FIM
    FIM
    ATÉ op2 = 7
    FIM
SE op1 = 2
ENTÃO INÍCIO
    ESCREVA "Cadastro de notas de saída"
    ESCREVA "Digite o número da nota"
    LEIA num_nota
    SE livre_nota = 6
    ENTÃO ESCREVA "Cadastro de notas lotado"
    SENÃO INÍCIO
        achou ← 0
        PARA i ← 1 TATÉ 5 FAÇA
        INÍCIO
            SE nota[i].numero_nf = num_nota
            ENTÃO achou ← 1
        FIM
        SE achou = 1
        ENTÃO ESCREVA "Já existe nota fiscal cadastrada com esse
        ↪ número"
        SENÃO INÍCIO
            ESCREVA "Digite o código do cliente"
            LEIA codigo_cli
            achou ← 0
            ENQUANTO achou = 0 FAÇA
            INÍCIO
                PARA i ← 1 ATÉ 3 FAÇA
            INÍCIO
                SE cliente[i].cod_cliente = codigo_cli
                ENTÃO achou ← 1
            FIM
            SE achou = 0
            ENTÃO INÍCIO
                ESCREVA "Este cliente não está cadastrado"
                ESCREVA "Digite outro cliente"

```

```

        LEIA codigo_cli
        FIM
    FIM
    cont ← 0
    resp ← 's'
    ENQUANTO (cont < 2) E (resp = 's') FAÇA
    INÍCIO
        ESCREVA "Digite o código do produto"
        LEIA codigo
        achou ← 0
        PARA k ← 1 ATÉ 6 FAÇA
        INÍCIO
            SE produto[k].cod_produto = codigo
            ENTÃO INÍCIO
                SE produto[k].qtde_est = 0
                ENTÃO achou ← 2
                SENÃO INÍCIO
                    achou ← 1
                    posi ← k
                    PARA h ← 1 ATÉ 10 FAÇA
                    INÍCIO
                        SE (itens_nota[h].numero_nf =
                            num_nota)
                            E (itens_nota[h].cod_prod = codigo)
                        ENTÃO INÍCIO
                            ESCREVA "Produto já existe
                                em esta nota"
                            achou ← 0
                        FIM
                    FIM
                FIM
            FIM
        FIM
    FIM
    SE achou = 0
    ENTÃO INÍCIO
        ESCREVA "Este produto não está cadastrado"
        FIM
    SE achou = 2
    ENTÃO INÍCIO
        ESCREVA "Este produto está com estoque
            zerado"
        FIM
    SE achou = 1
    ENTÃO INÍCIO
        ESCREVA "Digite a quantidade"
        LEIA qtde
        ENQUANTO qtde > produto[posi].qtde_est FAÇA
        INÍCIO
            ESCREVA "Estoque insuficiente"
            ESCREVA "Digite outra quantidade"
            LEIA qtde
        FIM
        nota[livre_nota].numero_nf ← num_nota
        nota[livre_nota].cod_cliente ← codigo_cli
        nota[livre_nota].total ←
        nota[livre_nota].total + (qtde *
        produto[posi].preco_unit)
        itens_nota[livre_item].numero_nf ← num_nota
        itens_nota[livre_item].cod_prod ← codigo
        itens_nota[livre_item].qtde ← qtde
        itens_nota[livre_item].preco_vend ← qtde *
        produto[posi].preco_unit

```

```

        livre_item ← livre_item + 1
        ESCREVA "Produto incluído na nota com
        ↪ sucesso"
        produto[posi].qtde_est ←
        ↪ produto[posi].qtde_est - qtde
        cont ← cont + 1
        FIM
    SE cont < 2
    ENTÃO INÍCIO
        ESCREVA "Deseja cadastrar outro produto
        ↪ nesta nota"
        ESCREVA "s - sim ou n - não"
        LEIA resp
        FIM
    FIM
    SE cont >= 1
    ENTÃO INÍCIO
        livreNota ← livreNota + 1
        ESCREVA "Nota cadastrada com sucesso"
        FIM
    FIM
    FIM
SE op1 = 3
ENTÃO INÍCIO
REPITA
    ESCREVA "Submenu de Opções"
    ESCREVA "1 - Consultar todos os produtos com preços entre dois
    ↪ valores"
    ESCREVA "2 - Consultar todas as notas de um cliente"
    ESCREVA "3 - Consultar todas as notas com total superior a um
    ↪ determinado valor"
    ESCREVA "4 - Sair"
    ESCREVA "Digite sua opção"
    LEIA op2
    SE (op2 < 1) OU (op2 > 4)
    ENTÃO ESCREVA "Opção inválida, digite novamente"
    SE op2 = 1
    ENTÃO INÍCIO
        ESCREVA "Consultar todos os produtos com preços entre
        ↪ dois valores"
        ESCREVA "Digite o valor inicial"
        LEIA valor_inicial
        ESCREVA "Digite o valor final"
        LEIA valor_final
        SE livre_produto = 1
        ENTÃO ESCREVA "Nenhum produto está cadastrado"
        SENÃO INÍCIO
            achou ← 0
            PARA i ← 1 ATÉ (livre_produto - 1) FAÇA
                INÍCIO
                    SE (produto[i].preco_unit >= valor_inicial)
                    E (produto[i].preco_unit <= valor_final)
                    ENTÃO INÍCIO
                        achou ← 1
                        ESCREVA produto[i].cod_produto
                        ESCREVA produto[i].desc
                        FIM
                FIM
            SE achou = 0
            ENTÃO ESCREVA "Nenhum produto foi cadastrado com
            ↪ estes preços
            FIM

```

```

        FIM
SE op2 = 2
ENTÃO INÍCIO
    ESCREVA "Consultar todas as notas e itens da nota de um
    ↪ cliente"
    ESCREVA "Digite o código do cliente"
    LEIA codigo
    SE livre_cliente = 1
    ENTÃO ESCREVA "Não existe cliente cadastrado"
    SENÃO INÍCIO
        achou ← 0
        PARA i ← 1 ATÉ (livre_cliente - 1) FAÇA
        INÍCIO
            SE cliente[i].cod_cliente = codigo
            ENTÃO INÍCIO
                achou ← 1
                posi ← i
                FIM
            FIM
        SE achou = 0
        ENTÃO ESCREVA "Este cliente não está cadastrado"
        SENÃO INÍCIO
            SE livre_nota = 1
            ENTÃO ESCREVA "Nenhuma nota cadastrada"
            SENÃO INÍCIO
                achou ← 0
                PARA i ← 1 ATÉ (livre_nota - 1) FAÇA
                INÍCIO
                    SE nota[i].cod_cliente = codigo
                    ENTÃO INÍCIO
                        achou ← 1
                        ESCREVA nota[i].numero_nf
                        ESCREVA nota[i].total
                        PARA j ← 1 ATÉ (livre_item - 1) FAÇA
                        INÍCIO
                            SE itens_nota[j].numero_nf =
                            ↪ nota[i].numero_nf
                            ENTÃO INÍCIO
                                ESCREVA itens_nota[j].cod_prod
                                ESCREVA itens_nota[j].qtde
                                ESCREVA itens_nota[j].preco_vend
                            FIM
                        FIM
                    FIM
                FIM
            SE achou = 0
            ENTÃO ESCREVA "Nenhuma nota cadastrada"
            FIM
        FIM
    FIM
SE op2 = 3
ENTÃO INÍCIO
    ESCREVA "Consultar todas as notas e itens da nota com total
    ↪ superior a um valor"
    ESCREVA "Digite o valor"
    LEIA valor
    SE livre_nota = 1
    ENTÃO ESCREVA "Não existe nota cadastrada"
    SENÃO INÍCIO
        achou ← 0
        PARA i ← 1 ATÉ (livre_nota - 1) FAÇA

```

```

INÍCIO
SE nota[i].total > valor
ENTÃO INÍCIO
    achou ← 1
    ESCREVA nota[i].numero_nf
    ESCREVA nota[i].total
    PARA j ← 1 ATÉ (livre_item - 1) FAÇA
        INÍCIO
            SE (itens_nota[j].numero_nf =
                nota[i].numero_nf)
            ENTÃO INÍCIO
                ESCREVA itens_nota[j].cod_prod
                ESCREVA itens_nota[j].qtde
                ESCREVA itens_nota[j].preco_vend
            FIM
        FIM
    FIM
    SE achou = 0
    ENTÃO ESCREVA "Nenhuma nota cadastrada"
    FIM
FIM
ATÉ op2 = 4
FIM
ATÉ op1 = 4
FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP8\PASCAL\EX9.PAS e \EXERC\CAP8\PASCAL\EX9.EXE

**SOLUÇÃO:**

\EXERC\CAP8\C++\EX9.CPP e \EXERC\CAP8\C++\EX9.EXE

- 10.** Uma empresa do ramo de material esportivo deseja ter um controle automatizado dos funcionários que trabalham em cada uma de suas filiais. Sabe-se que essa empresa possui quatro filiais com quatro vendedores e um gerente em cada uma delas. Todos devem ser cadastrados como funcionários.

Faça um programa que realize esse controle, com as seguintes rotinas:

- cadastrar filial, observando que não podem existir duas filiais com o mesmo número;
- cadastrar funcionário, observando que: (i) não podem existir dois funcionários com o mesmo número; (ii) cada funcionário deve ser cadastrado em uma filial e (iii) cada filial pode ter apenas um gerente e no máximo quatro vendedores;
- criar uma consulta a todas as filiais, mostrando o nome do gerente e dos vendedores, o valor total gasto com pagamento de salários por filial e o valor gasto com pagamento de salário geral.

<b>FILIAL</b>	<b>FUNCIONÁRIO</b>
Numero_filial	Numero_filial
Nome_filial	Codigo_funcionario
	Nome_funcionario
	Cargo
	Salario

**ALGORITMO****SOLUÇÃO:**

```

ALGORITMO
DECLARE filial[4] REGISTRO (numero_filiai NÚMERO nome_filiai
                                ↪ LITERAL)
                                func[20] REGISTRO (numero_filiai, cod_func, salario NUMÉRICO
                                ↪ cargo, nome_func LITERAL)
                                i, j, livre_filiai, livre_func NUMÉRICO
                                total_geral, total_filiai NUMÉRICO
                                sal, cont, filial, op, achou, numero NUMÉRICO
                                cargo, nome LITERAL
PARA i ← 1 ATÉ 4 FAÇA
INÍCIO
filial[i].numero_filiai ← 0
filial[i].nome_filiai ← "
FIM
PARA i ← 1 ATÉ 20 FAÇA
INÍCIO
func[i].numero_filiai ← 0
func[i].cod_func ← 0
func[i].salario ← 0
func[i].cargo ← "
func[i].nome_func ← "
FIM
livre_func ← 1
livre_filiai ← 1
total_geral ← 0
REPITA
ESCREVA "Menu de Opções"
ESCREVA "1 - Cadastrar filial"
ESCREVA "2 - Cadastrar funcionário"
ESCREVA "3 - Consultar filiais"
ESCREVA "4 - Sair"
ESCREVA "Digite a opção desejada"
LEIA op
SE op = 1
ENTÃO INÍCIO
    ESCREVA "Cadastro de filiais"
    SE livre_filiai = 5
        ENTÃO ESCREVA "Cadastro de filiais lotado"
        SENÃO INÍCIO
            ESCREVA "Digite o número da filial"
            LEIA numero
            achou ← 0
            PARA i ← 1 ATÉ (livre_filiai - 1) FAÇA
            INÍCIO
                SE filial[i].numero_filiai = numero
                ENTÃO achou ← 1
                FIM
                SE achou = 1
                ENTÃO ESCREVA "Já existe filial cadastrada com este
                                ↪ número"
                SENÃO INÍCIO
                    ESCREVA "Digite o nome da filial"
                    LEIA nome
                    filial[livre_filiai].numero_filiai ← numero
                    filial[livre_filiai].nome_filiai ← nome
                    ESCREVA "Filial cadastrado com sucesso!"
                    livre_filiai ← livre_filiai + 1
                    FIM
            FIM
    FIM
SE op = 2

```



```

ENTÃO INÍCIO
    cont ← 0
    PARA i ← 1 ATÉ (livre_func - 1)
        ↪ FAÇA
        INÍCIO
            SE (func[i].numero_filial =
                ↪ filial)
                E (func[i].cargo = cargo)
            ENTÃO cont ← cont + 1
            FIM
            SE cont = 4
            ENTÃO INÍCIO
                ESCREVA "Filial já tem 4
                ↪ vendedores"
                FIM
            SENÃO cont ← 0
            FIM
            SE cont = 0
            ENTÃO INÍCIO
                ESCREVA "Digite o salário"
                LEIA sal
                ESCREVA "Digite o nome do
                ↪ funcionário"
                LEIA nome
                func[livre_func].numero_filial
                ↪ ← filial
                func[livre_func].cod_func ←
                ↪ numero
                func[livre_func].salario ← sal
                func[livre_func].cargo ← cargo
                func[livre_func].nome_func ←
                ↪ nome
                ESCREVA "Funcionário cadastrado
                ↪ com sucesso!"
                livre_func ← livre_func + 1
            FIM
            FIM
        FIM
    FIM
FIM
FIM
FIM
FIM
SE op = 3
ENTÃO INÍCIO
    SE livre_filial = 1
    ENTÃO ESCREVA "Cadastro de filiais vazio"
    SENÃO SE livre_func = 1
        ENTÃO ESCREVA "Cadastro de funcionários vazio"
        SENÃO INÍCIO
            PARA i ← 1 ATÉ (livre_filial - 1) FAÇA
            INÍCIO
                ESCREVA "Filial número: ", filial[i].numero_filial
                ESCREVA " nome: ", filial[i].nome_filial
                total_filial ← 0
            PARA j ← 1 ATÉ (livre_func - 1) FAÇA
            INÍCIO
                SE func[j].numero_filial =
                    ↪ filial[i].numero_filial
                ENTÃO INÍCIO
                    ESCREVA "Funcionário: ",
                    ↪ func[j].nome_func
                    ESCREVA "Cargo: ", func[j].cargo
                    total_filial ← total_filial +

```

```

        ↪ func[j].salario
    FIM
    FIM
    ESCREVA "Total de salário da filial = ",
    ↪ total_filial
    total_geral ← total_geral + total_filial
    FIM
    ESCREVA "Total dos salários de todas as filiais =
    ↪ ", total_geral
    FIM
    FIM
ATÉ op = 4
FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP8\PASCAL\EX10.PAS e \EXERC\CAP8\PASCAL\EX10.EXE

**SOLUÇÃO:**

\EXERC\CAP8\C++\EX10.CPP e \EXERC\CAP8\C++\EX10.EXE

11. Crie um controle de matrícula anual de alunos em uma escola. Sabe-se que nessa escola é permitida a matrícula por disciplinas (o aluno monta o seu horário). Esse controle deverá armazenar as informações pertinentes a apenas um ano. A escola oferece a cada ano seis disciplinas e sabe-se que existem dez alunos e que cada um pode matricular-se em, no máximo, três disciplinas a cada ano.

As informações devem estar estruturadas conforme os registros a seguir:

- ◆ Aluno (código do aluno, nome do aluno, série).
- ◆ Matrícula (código do aluno, código da disciplina, total de faltas, nota final).
- ◆ Disciplina (código da disciplina, descrição, carga horária).

Seu programa deve seguir as especificações abaixo:

- ◆ cadastrar todas as disciplinas que poderão ser oferecidas no ano (não permita duas disciplinas com o mesmo código);
- ◆ cadastrar alunos (não permita dois alunos com o mesmo código e os valores válidos para a série vão de 5 a 8);
- ◆ realizar a matrícula do aluno (nesse momento, o aluno está apenas se inscrevendo na disciplina que ainda não foi cursada. Dessa maneira, os campos 'total de faltas' e 'nota final' não devem ser preenchidos);
- ◆ lançamento dos resultados finais (a secretaria, ao final do ano letivo, informa o código do aluno e o código da disciplina e preenche os campos 'total de faltas' e 'nota final' que estavam vazios);
- ◆ criar uma consulta a todos os alunos reprovados nas disciplinas (reprovado se a nota for menor que 7,0 ou se o total de faltas ultrapassar 25% da carga horária);
- ◆ criar uma rotina que mostre o nome das disciplinas cursadas por um determinado aluno, juntamente com o total de faltas, a nota final e o resultado (aprovado ou reprovado).

**SOLUÇÃO:**

```

ALGORITMO
DECLARE alunos[10] REGISTRO (cod_a, serie NÚMERO nome LITERAL)
                                disciplinas[6] REGISTRO (cod_d, carga_hor NÚMERO descr

```

```

    ↪ LITERAL)
matriculas[30] REGISTRO (cod_a, cod_d, faltas, nota
    ↪ NUMÉRICO)
cont_a, cont_d, cont_m, cont, alu_aux NUMÉRICO
dis_aux, achou, i, j, k, op, perc NUMÉRICO
cont_a ← 1
cont_d ← 1
cont_m ← 1
REPITA
ESCREVA "1-Cadastrar disciplinas"
ESCREVA "2-Cadastrar alunos"
ESCREVA "3-Realizar matrículas"
ESCREVA "4-Lançar notas e faltas"
ESCREVA "5-Consultar alunos reprovados"
ESCREVA "6-Mostrar disciplinas cursadas por aluno"
ESCREVA "7-Finalizar"
ESCREVA "Digite sua opção: "
LEIA op
SE (op = 1)
ENTÃO INÍCIO
    SE (cont_d < 6)
        ENTÃO INÍCIO
            LEIA dis_aux
            i ← 1
            ENQUANTO (i < cont_d) E
                (dis_aux ≠ disciplinas[i].cod_d) FAÇA
                    INÍCIO
                    i ← i + 1
                    FIM
            SE (i < cont_d)
                ENTÃO ESCREVA "Disciplina já cadastrada! "
                SENÃO INÍCIO
                    disciplinas[cont_d].cod_d ← dis_aux
                    LEIA disciplinas[cont_d].descr
                    LEIA disciplinas[cont_d].carga_hor
                    cont_d ← cont_d + 1
                    FIM
            FIM
            SENÃO ESCREVA "Já foram cadastradas as 6 disciplinas! "
        FIM
    SE (op = 2)
    ENTÃO INÍCIO
        SE (cont_a < 10)
            ENTÃO INÍCIO
                LEIA alu_aux
                i ← 1
                ENQUANTO (i < cont_a) E (alu_aux ≠ alunos[i].cod_a) FAÇA
                    INÍCIO
                    i ← i + 1
                    FIM
                SE (i < cont_a)
                    ENTÃO ESCREVA "Aluno já cadastrado! "
                    SENÃO INÍCIO
                        alunos[cont_a].cod_a ← alu_aux
                        LEIA alunos[cont_a].nome
                        REPITA
                            LEIA alunos[cont_a].serie
                            ATÉ (alunos[cont_a].serie >= 5) E
                                ↪ (alunos[cont_a].serie <= 8)
                            cont_a ← cont_a + 1
                            FIM
            FIM
            SENÃO ESCREVA "Já foram cadastrados os 10 alunos! "

```

```

        cont_m = cont_m + 1
    FIM
    FIM
    FIM
    FIM
    SENÃO ESCREVA "Não existe espaço para mais
    ↪ matrículas!"
FIM
SE (op = 4)
ENTÃO INÍCIO
    LEIA alu_aux
    i ← 1
    ENQUANTO (i < cont_a) E (alunos[i].cod_a ≠ alu_aux) FAÇA
        INÍCIO
            i ← i + 1
        FIM
    SE (i = cont_a)
    ENTÃO ESCREVA "Não existe aluno com este código! "
    SENÃO INÍCIO
        LEIA dis_aux
        i ← 1
        ENQUANTO (i < cont_d) E
            (disciplinas[i].cod_d ≠ dis_aux) FAÇA
                INÍCIO
                    i ← i + 1
                FIM
        SE (i = cont_d)
        ENTÃO ESCREVA "Não existe disciplina com este código! "
        SENÃO INÍCIO
            i ← 1
            ENQUANTO (i < cont_m) E
                ((matriculas[i].cod_a ≠ alu_aux) OU
                 (matriculas[i].cod_d ≠ dis_aux)) FAÇA
            INÍCIO
                i ← i + 1
            FIM
            SE (i = cont_m)
            ENTÃO ESCREVA "Matrícula inválida! "
            SENÃO INÍCIO
                LEIA matriculas[i].faltas
                LEIA matriculas[i].nota
            FIM
        FIM
    FIM
SE (op = 5)
ENTÃO INÍCIO
    PARA i ← 1 ATÉ cont_m - 1 FAÇA
        INÍCIO
        j ← 1
        ENQUANTO (j < cont_d) E
            (matriculas[i].cod_d ≠ disciplinas[j].cod_d) FAÇA
                INÍCIO
                j ← j + 1
            FIM
        perc ← disciplinas[j].carga_hor * 25 / 100
        SE (matriculas[i].faltas > perc) OU
            (matriculas[i].nota < 7)
            ENTÃO INÍCIO
                k ← 1
                ENQUANTO (k < cont_a)
                    E (matriculas[i].cod_a ≠ alunos[k].cod_a) FAÇA

```

```

        INÍCIO
        k ← k + 1
        FIM
        ESCREVA alunos[k].nome
        ESCREVA disciplinas[j].descr
        ESCREVA matriculas[i].faltas
        ESCREVA matriculas[i].nota
        FIM
    FIM
SE (op = 6)
ENTÃO INÍCIO
    LEIA alu_aux
    i ← 1
    ENQUANTO (i < cont_a) E (alunos[i].cod_a ≠ alu_aux) FAÇA
        INÍCIO
        i ← i + 1
        FIM
    SE (i = cont_a)
    ENTÃO ESCREVA "Este aluno não está cadastrado!"
    SENÃO INÍCIO
        ESCREVA alunos[i].nome
        PARA i ← 1 ATÉ cont_m - 1 FAÇA
        INÍCIO
        SE (matriculas[i].cod_a = alu_aux)
        ENTÃO INÍCIO
            j ← 1
            ENQUANTO (j < cont_d) E
                (disciplinas[j].cod_d ≠ matriculas[i].cod_d) FAÇA
                INÍCIO
                j ← j + 1
                FIM
            ESCREVA disciplinas[j].descr
            ESCREVA matriculas[i].faltas
            ESCREVA matriculas[i].nota
            perc ← disciplinas[j].carga_hor * 25 / 100
            SE ((matriculas[i].faltas > perc) OU
                (matriculas[i].nota < 7))
            ENTÃO ESCREVA "Reprovado"
            SENÃO ESCREVA "Aprovado"
        FIM
        FIM
    FIM
ATÉ (op = 7)
FIM_ALGORITMO.

```



## SOLUÇÃO:

\EXERC\CAP8\PASCAL\EX11.PAS e \EXERC\CAP8\PASCAL\EX11.EXE



## SOLUÇÃO:

\EXERC\CAP8\C++\EX11.CPP e \EXERC\CAP8\C++\EX11.EXE

-  12. Faça um programa que receba a hora de início e a hora de término de um jogo, ambas subdivididas em dois valores distintos: horas e minutos, e determine a duração do jogo expressa em horas e minutos, apenas em minutos e apenas em segundos, considerando que o tempo máximo de duração de um jogo é de 24 horas e que o jogo pode começar em um dia e terminar no outro.

**SOLUÇÃO:**

```

ALGORITMO
DECLARE tempo REGISTRO ( hora_ini, min_ini, hora_fim, min_fim
                           ↪ NUMÉRICO)
               min, seg, hora_total, min_total NUMÉRICO
REPITA
    LEIA tempo.hora_ini
ATÉ (tempo.hora_ini <= 24) E (tempo.hora_ini >= 0)
REPITA
    LEIA tempo.min_ini
ATÉ (tempo.min_ini <= 59) E (tempo.min_ini >= 0)
REPITA
    LEIA tempo.hora_fim
ATÉ (tempo.hora_fim <= 24) OU (tempo.hora_fim >= 0)
REPITA
    LEIA tempo.min_fim
ATÉ (tempo.min_fim <= 59) E (tempo.min_fim >= 0)
SE (tempo.min_fim < tempo.min_ini)
    ENTÃO INÍCIO
        tempo.min_fim ← tempo.min_fim + 60
        tempo.hora_fim ← tempo.hora_fim - 1
    FIM
SE (tempo.hora_fim < tempo.hora_ini)
    ENTÃO INÍCIO
        tempo.hora_fim ← tempo.hora_fim + 24
    FIM
hora_total ← tempo.hora_fim - tempo.hora_ini
min_total ← tempo.min_fim - tempo.min_ini
ESCREVA hora_total, min_total
min ← hora_total * 60 + min_total
ESCREVA min
seg ← min * 60
ESCREVA seg
FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP8\PASCAL\EX12.PAS e \EXERC\CAP8\PASCAL\EX12.EXE

**SOLUÇÃO:**

\EXERC\CAP8\C++\EX12.CPP e \EXERC\CAP8\C++\EX12.EXE

**13.** Faça um programa que manipule uma lista contendo informações sobre dez pacientes (nome do paciente, nome do médico, data de nascimento e sexo). Esse programa deverá implementar as seguintes rotinas:

1. cadastrar pacientes;
  2. mostrar elementos na ordem de cadastramento;
  3. mostrar elementos em ordem crescente (ordenar pelo nome do paciente);
  4. mostrar elementos em ordem decrescente (ordenar pelo nome do paciente);
  5. excluir pacientes individualmente;
  6. excluir pacientes por médico.
- ◆ os elementos podem ser inseridos sem qualquer ordenação (utilizar um vetor e observar que não poderão ser cadastrados mais de dez pacientes);
  - ◆ quando a lista for mostrada em ordem crescente ou decrescente, deve-se utilizar alguma maneira que não destrua a ordem original de cadastramento;

```

        FIM
SE (op = 3)
ENTÃO INÍCIO
    SE (cont_m < 30)
        ENTÃO INÍCIO
            LEIA alu_aux
            i ← 1
            ENQUANTO (i < cont_a) E (alunos[i].cod_a ≠ alu_aux) FAÇA
                INÍCIO
                i ← i + 1
                FIM
            SE (i = cont_a)
                ENTÃO ESCREVA "Aluno não cadastrado! "
            SENÃO INÍCIO
                cont ← 0
                PARA i ← 1 ATÉ cont_m-1 FAÇA
                    INÍCIO
                    SE (matriculas[i].cod_a = alu_aux)
                        ENTÃO cont ← cont + 1
                    FIM
                SE (cont >= 3)
                    ENTÃO ESCREVA "Aluno já está matriculado
                    ↪ em 3 disciplinas "
                SENÃO INÍCIO
                    LEIA dis_aux
                    j ← 1
                    ENQUANTO (j < cont_d)
                        E (disciplinas[j].cod_d ≠ dis_aux)
                        ↪ FAÇA
                    INÍCIO
                    j ← j + 1
                    FIM
                SE (j = cont_d)
                    ENTÃO INÍCIO
                        ESCREVA "Disciplina não
                        ↪ cadastrada! "
                        achou ← 1
                    FIM
                SENÃO INÍCIO
                    j ← 1
                    achou ← 0
                    ENQUANTO (j < cont_m) E
                        (achou = 0) FAÇA
                    INÍCIO
                    SE (matriculas[j].cod_a =
                        ↪ alu_aux) E
                        (matriculas[j].cod_d =
                            ↪ dis_aux)
                            ENTÃO achou ← 1
                    j ← j + 1
                    FIM
                SE (achou = 1)
                    ENTÃO ESCREVA "Aluno já está
                    ↪ nesta disc."
                SENÃO INÍCIO
                    matriculas[cont_m].
                    ↪ cod_a ← alu_aux
                    matriculas[cont_m].
                    ↪ cod_d ← dis_aux
                    matriculas[cont_m].
                    ↪ faltas ← 0
                    matriculas[cont_m].
                    ↪ nota ← 0

```

- ◆ para realizar a exclusão de pacientes individualmente deve-se informar o nome do paciente que se deseja remover;
- ◆ para realizar a exclusão de pacientes por médico, deve-se informar o nome do médico cujos pacientes serão excluídos da lista.

**ALGORITMO****SOLUÇÃO:**

```

ALGORITMO
DECLARE pacientes[10], pac_cres[10], pac_decrecs[10] REGISTRO (nome,
    nome_med, data_nasc, sexo LITERAL)
cont, cont_p, i, j, op NUMÉRICO
pac_aux, med_aux LITERAL
cont_p ← 1
REPITA
    ESCREVA "1-Cadastrar paciente "
    ESCREVA "2-Mostrar pacientes em ordem de cadastramento "
    ESCREVA "3-Mostrar pacientes em ordem crescente "
    ESCREVA "4-Mostrar pacientes em ordem decrescente "
    ESCREVA "5-Excluir paciente individualmente "
    ESCREVA "6-Excluir pacientes por médico "
    ESCREVA "7-Finalizar "
    ESCREVA "Digite sua opção: "
LEIA op
SE (op = 1)
ENTÃO INÍCIO
    SE (cont_p <= 10)
        ENTÃO INÍCIO
            LEIA pac_aux
            i ← 1
            ENQUANTO (i < cont_p) E (pacientes[i].nome ≠ pac_aux) FAÇA
                INÍCIO
                i ← i + 1
                FIM
            SE (i < cont_p)
                ENTÃO ESCREVA "Paciente já cadastrado! "
                SENÃO INÍCIO
                    pacientes[cont_p].nome ← pac_aux;
                    LEIA pacientes[cont_p].nome_med
                    LEIA pacientes[cont_p].data_nasc
                    LEIA pacientes[cont_p].sexo
                    i ← 1
                    ENQUANTO (i < cont_p) E
                        pacientes[cont_p].nome > pac_cres[i].nome
                            FAÇA
                    INÍCIO
                    i ← i + 1
                    FIM
                SE (i = cont_p)
                    ENTÃO pac_cres[cont_p] ← pacientes[cont_p]
                    SENÃO INÍCIO
                        j ← cont_p - 1
                        ENQUANTO j >= i FAÇA
                            INÍCIO
                            pac_cres[j+1] ← pac_cres[j]
                            j ← j - 1
                            FIM
                        pac_cres[i] ← pacientes[cont_p]
                        FIM
                    i ← 1
                    ENQUANTO (i < cont_p) E
                        (pacientes[cont_p].nome < pac_decrecs[i].nome)

```

```

    ➔ FAÇA
INÍCIO
i ← i + 1
FIM
SE (i = cont_p)
ENTÃO pac_decres[cont_p] ← pacientes[cont_p]
SENÃO INÍCIO
    j ← cont_p - 1
    ENQUANTO j >= i FAÇA
        INÍCIO
            pac_decres[j+1] ← pac_decres[j]
            j ← j - 1
        FIM
        pac_decres[i] ← pacientes[cont_p]
        FIM
    cont_p ← cont_p + 1
    FIM
FIM
SENÃO ESCREVA "Já foram cadastrados 10 pacientes! "
    FIM
SE (op = 2)
ENTÃO INÍCIO
    PARA i ← 1 ATÉ cont_p - 1 FAÇA
        INÍCIO
            ESCREVA pacientes[i].nome
            ESCREVA pacientes[i].nome_med
            ESCREVA pacientes[i].data_nasc
            ESCREVA pacientes[i].sexo
        FIM
    FIM
SE (op = 3)
ENTÃO INÍCIO
    PARA i ← 1 ATÉ cont_p - 1 FAÇA
        INÍCIO
            ESCREVA pac_cres[i].nome
            ESCREVA pac_cres[i].nome_med
            ESCREVA pac_cres[i].data_nasc
            ESCREVA pac_cres[i].sexo
        FIM
    FIM
SE (op = 4)
ENTÃO INÍCIO
    PARA i ← 1 ATÉ cont_p - 1 FAÇA
        INÍCIO
            ESCREVA pac_decres[i].nome
            ESCREVA pac_decres[i].nome_med
            ESCREVA pac_decres[i].data_nasc
            ESCREVA pac_decres[i].sexo
        FIM
    FIM
SE (op = 5)
ENTÃO INÍCIO
    LEIA pac_aux
    i ← 1
    ENQUANTO (pacientes[i].nome ≠ pac_aux) E (i < cont_p) FAÇA
        INÍCIO
            i ← i + 1
        FIM
    SE (i = cont_p)
        ENTÃO ESCREVA "Paciente não cadastrado!"
        SENÃO INÍCIO
            ESCREVA "Paciente excluído com sucesso"
            PARA j ← i ATÉ cont_p - 2 FAÇA

```

```

    INÍCIO
    pacientes[j] ← pacientes[j+1]
    FIM
    i ← 1
    ENQUANTO (pac_cres[i].nome ≠ pac_aux) E (i < cont_p) FAÇA
        INÍCIO
        i ← i + 1
        FIM
        SE (i = cont_p)
        ENTÃO ESCREVA "Paciente não cadastrado!"
        SENÃO INÍCIO
            PARA j ← i ATÉ cont_p - 2 FAÇA
            INÍCIO
                pac_cres[j] = pac_cres[j+1]
            FIM
        FIM
        i ← 1
    ENQUANTO (pac_decres[i].nome ≠ pac_aux) E (i < cont_p) FAÇA
        INÍCIO
        i ← i + 1
        FIM
        SE (i = cont_p)
        ENTÃO ESCREVA "Paciente não cadastrado!"
        SENÃO INÍCIO
            PARA j ← i ATÉ cont_p - 2 FAÇA
            INÍCIO
                pac_decres[j] ← pac_decres[j+1]
            FIM
        FIM
        cont_p ← cont_p - 1
    FIM
    FIM
    SE (op = 6)
    ENTÃO INÍCIO
        LEIA med_aux
        i ← 1
        cont ← cont_p
        ENQUANTO (i < cont) FAÇA
            INÍCIO
            SE (pacientes[i].nome_med = med_aux)
            ENTÃO INÍCIO
                PARA j ← i ATÉ cont - 1 FAÇA
                INÍCIO
                    pacientes[j] ← pacientes[j+1]
                FIM
                cont ← cont - 1
            FIM
            SENÃO i ← i + 1
        FIM
        SE i = cont_p
        ENTÃO ESCREVA "Médico não cadastrado"
        SENÃO INÍCIO
            ESCREVA "Pacientes excluídos com sucesso"
            i ← 1
            cont ← cont_p
            ENQUANTO (i < cont) FAÇA
            INÍCIO
            SE (pac_cres[i].nome_med = med_aux)
            ENTÃO INÍCIO
                PARA j ← i ATÉ cont - 1 FAÇA
                INÍCIO
                    pac_cres[j] ← pac_cres[j+1]
                FIM

```

```

        cont ← cont - 1
        FIM
        SENÃO i ← i + 1
        FIM
        i ← 1
        cont ← cont_p
        ENQUANTO (i < cont) FAÇA
        INÍCIO
            SE (pac_decres[i].nome_med = med_aux)
            ENTÃO  INÍCIO
                PARA j ← i ATÉ cont - 1 FAÇA
                INÍCIO
                    pac_decres[j] ← pac_decres[j+1]
                FIM
                cont ← cont - 1
                FIM
            SENÃO i ← i + 1
            FIM
            cont_p ← cont
        FIM
    FIM
    ATÉ (op = 7)
FIM_ALGORITMO.

```



#### SOLUÇÃO:

\EXERC\CAP8\PASCAL\EX13.PAS e \EXERC\CAP8\PASCAL\EX13.EXE



#### SOLUÇÃO:

\EXERC\CAP8\C++\EX13.CPP e \EXERC\CAP8\C++\EX13.EXE

**14.** Uma empresa patrocinadora de regatas deseja ter um controle preciso sobre os participantes e resultados, a fim de realizar adequadamente o pagamento dos prêmios. Dessa maneira, precisa cadastrar algumas informações, conforme apresentado a seguir:

**Regata** (número da regata, data, hora de início, código do barco vencedor).

**Barco/Regata** (número da regata, número do barco participante, hora de chegada).

**Barco** (número do barco, nome do barco, ano de fabricação).

- ◆ cadastre os barcos, não permitindo dois barcos com o mesmo número (defina espaço para seis barcos);
- ◆ cadastre as regatas, não permitindo duas regatas com o mesmo número (defina espaço para três regatas);
- ◆ cadastre os participantes (não permitindo cadastrar mais que uma vez um barco para a mesma regata, levando-se em consideração que em cada regata podem participar, no máximo, quatro barcos);
- ◆ mostre uma relação das regatas realizadas, juntamente com o nome do barco vencedor;
- ◆ mostre uma relação de todos os barcos que participaram de uma determinada regata, mostrando o tempo que levaram para cumprir a regata (considere que todas as regatas duram, no máximo, 24 horas).



#### SOLUÇÃO:

```

ALGORITMO
DECLARE barcos[6] REGISTRO (num_b, ano_f NUMÉRICO nome LITERAL)
    regatas[3] REGISTRO (num_r, cod_venc, hora_i NUMÉRICO data

```

```

    ↪ LITERAL)
barco_reg[12] REGISTRO (num_r, num_b, hora_c NUMÉRICO)
i, j, k, cont_b, cont_r, cont_rb NUMÉRICO
cont, reg_aux, bar_aux, op NUMÉRICO
cont_b ← 1
cont_r ← 1
cont_rb ← 1
REPITA
ESCREVA "1-Cadastrar barco"
ESCREVA "2-Cadastrar regata"
ESCREVA "3-Cadastrar participantes"
ESCREVA "4-Cadastrar barco vencedor"
ESCREVA "5-Mostrar regatas com seus vencedores"
ESCREVA "6-Mostrar participantes de uma regata"
ESCREVA "7-Finalizar"
ESCREVA "Digite sua opção: "
LEIA op
SE (op = 1)
ENTÃO INÍCIO
    SE (cont_b < 6)
        ENTÃO INÍCIO
            LEIA bar_aux
            i ← 1
            ENQUANTO (i < cont_b) E (barcos[i].num_b ≠ bar_aux) FAÇA
                INÍCIO
                    i ← i + 1
                FIM
            SE (i < cont_b)
                ENTÃO ESCREVA "Barco já cadastrado!"
            SENÃO INÍCIO
                barcos[cont_b].num_b ← bar_aux
                LEIA barcos[cont_b].nome
                LEIA barcos[cont_b].ano_f
                cont_b ← cont_b + 1
                ESCREVA "Barco cadastrado com sucesso!"
            FIM
        FIM
    SENÃO ESCREVA "Já foram cadastrados 6 barcos!"
    FIM
SE (op = 2)
ENTÃO INÍCIO
    SE (cont_r < 3)
        ENTÃO INÍCIO
            LEIA reg_aux
            i ← 1
            ENQUANTO (i < cont_r) E (regatas[i].num_r ≠ reg_aux) FAÇA
                INÍCIO
                    i ← i + 1
                FIM
            SE (i < cont_r)
                ENTÃO ESCREVA "Regata já cadastrada!"
            SENÃO INÍCIO
                regatas[cont_r].num_r ← reg_aux
                LEIA (regatas[cont_r].data)
                LEIA (regatas[cont_r].hora_i)
                regatas[cont_r].cod_venc ← 0
                cont_r ← cont_r + 1
            FIM
        FIM
    SENÃO ESCREVA "Já foram cadastradas 3 regatas!"
    FIM
SE (op = 3)

```

```

ENTÃO INÍCIO
LEIA reg_aux
i ← 1
ENQUANTO (i < cont_r) E (regatas[i].num_r ≠ reg_aux) FAÇA
    INÍCIO
        i ← i + 1
    FIM
SE (i = cont_r)
ENTÃO ESCREVA "Regata não cadastrada!"
SENÃO INÍCIO
    cont ← 0
PARA i ← 1 ATÉ cont_rb - 1 FAÇA
    INÍCIO
        SE (barco_reg[i].num_r = reg_aux)
            ENTÃO cont ← cont + 1
        FIM
    SE (cont = 4)
    ENTÃO ESCREVA "Já foram cadastrados 4 participantes
    ↪ nesta regata!"
    SENÃO INÍCIO
        LEIA bar_aux
        i ← 1
        ENQUANTO (i < cont_b) E
            (barcos[i].num_b ≠ bar_aux) FAÇA
                INÍCIO
                    i ← i + 1
                FIM
        SE (i = cont_b)
        ENTÃO ESCREVA "Barco não cadastrado!"
        SENÃO INÍCIO
            i ← 1
            ENQUANTO (i < cont_rb) FAÇA
                INÍCIO
                    SE (bar_aux = barco_reg[i].num_b)
                    E (barco_reg[i].num_r = reg_aux)
                    ENTÃO i ← cont_rb + 1
                    SENÃO i ← i + 1
                FIM
            SE (i > cont_rb)
                ENTÃO ESCREVA "Este barco já está
                ↪ participando desta regata!"
                SENÃO INÍCIO
                    barco_reg[cont_rb].num_r ←
                    ↪ reg_aux
                    barco_reg[cont_rb].num_b ←
                    ↪ bar_aux
                    LEIA barco_reg[cont_rb].hora_c
                    cont_rb ← cont_rb + 1
                FIM
            FIM
        FIM
    FIM
FIM
SE (op = 4)
ENTÃO INÍCIO
    LEIA reg_aux
    i ← 0
    ENQUANTO (i < cont_r) E (regatas[i].num_r ≠ reg_aux) FAÇA
        INÍCIO
            i ← i + 1
        FIM
    SE (i = cont_r)

```

```

ENTÃO ESCREVA "Regata não cadastrada!"
SENÃO INÍCIO
    LEIA bar_aux
    j ← 1
    ENQUANTO (j < cont_b) E
        (barco_reg[j].num_b ≠ bar_aux) FAÇA
            INÍCIO
                j ← j + 1
            FIM
        SE (j = cont_b)
            ENTÃO ESCREVA "Este barco não participou desta regata!"
            SENÃO regatas[i].cod_venc ← bar_aux
        FIM
    FIM
SE (op = 5)
ENTÃO INÍCIO
    PARA i ← 1 ATÉ cont_r - 1 FAÇA
        INÍCIO
            ESCREVA (regatas[i].num_r, regatas[i].data)
            SE (regatas[i].cod_venc = 0)
                ENTÃO ESCREVA "Ainda não foi cadastrado o barco vencedor! "
                SENÃO INÍCIO
                    j ← 1
                    ENQUANTO (j < cont_b)
                        E (regatas[i].cod_venc ≠ barcos[j].num_b) FAÇA
                            INÍCIO
                                j ← j + 1
                            FIM
                        SE (j = cont_b)
                            ENTÃO ESCREVA "Barco vencedor não cadastrado!"
                            SENÃO ESCREVA barcos[j].nome
                        FIM
                    FIM
                FIM
            FIM
        FIM
    FIM
SE (op = 6)
ENTÃO INÍCIO
    LEIA reg_aux
    i ← 1
    ENQUANTO (i < cont_r) E (regatas[i].num_r ≠ reg_aux) FAÇA
        INÍCIO
            i ← i + 1
        FIM
    SE (i = cont_r)
        ENTÃO ESCREVA "Regata não cadastrada!"
        SENÃO INÍCIO
            ESCREVA (regatas[i].num_r, regatas[i].data)
            PARA j ← 1 ATÉ cont_rb - 1 FAÇA
                INÍCIO
                SE (barco_reg[j].num_r = reg_aux)
                    ENTÃO INÍCIO
                        k ← 1
                        ENQUANTO (k < cont_b) E
                            (barcos[k].num_b ≠ barco_reg[j].num_b) FAÇA
                                INÍCIO
                                    k ← k + 1
                                FIM
                            SE (k < cont_b)
                                ENTÃO ESCREVA barcos[k].num_b, barcos[k].ano_f
                            FIM
                        FIM
                    FIM
                FIM
            FIM
        FIM
    FIM
FIM

```

ATÉ op = 7  
FIM\_ALGORITMO.



#### SOLUÇÃO:

\EXERC\CAP8\PASCAL\EX14.PAS e \EXERC\CAP8\PASCAL\EX14.EXE



#### SOLUÇÃO:

\EXERC\CAP8\C++\EX14.CPP e \EXERC\CAP8\C++\EX14.EXE

15. Uma pizzaria será informatizada e percebeu-se a necessidade de um controle eficaz do sistema de entrega em domicílio. Dessa maneira, serão utilizadas as informações a seguir:

- ◆ Clientes (número do telefone, nome do cliente, endereço, complemento, CEP).
- ◆ Pizzas (código da pizza, nome da pizza, valor).
- ◆ Pedido (número do pedido, número do telefone, código da pizza, código do motoqueiro, situação).

Por simplicidade será assumido que:

- (i) existem cinco clientes cadastrados;
  - (ii) existem três pizzas cadastradas;
  - (iii) podem existir seis pedidos;
  - (iv) o cliente é identificado pelo número de telefone;
  - (v) o campo 'situação do pedido' refere-se às fases pelas quais o pedido passa até chegar à residência do cliente (1 – Em preparo, 2 – A caminho, 3 – Entregue).
- ◆ crie uma rotina para o cadastramento das pizzas. Não pode haver mais de uma pizza com o mesmo código;
  - ◆ crie uma rotina para o cadastramento dos clientes. Não pode haver mais de um cliente com o mesmo número de telefone;
  - ◆ crie uma rotina para o cadastramento dos pedidos, na qual se supõe que o cliente e a pizza envolvidos já tenham sido cadastrados. Quando o pedido é cadastrado, o campo 'situação' recebe o valor 1 e o campo 'código do motoqueiro' recebe o valor zero;
  - ◆ crie uma rotina que faça o despacho da pizza, definindo o campo 'situação' para 2 e atribuindo o código do motoqueiro (que pode ser qualquer valor entre 1 e 5, inclusive);
  - ◆ crie uma rotina que faça o recebimento da pizza, alterando o campo 'situação' para 3;
  - ◆ mostre o total gasto com pizzas por cliente;
  - ◆ mostre todas as pizzas com uma determinada situação;
  - ◆ mostre o código do motoqueiro que mais fez entregas.



#### SOLUÇÃO:

```
ALGORITMO
DECLARE clientes[5] REGISTRO (fone, nome_cli, endereco, comple, cep
    ↪ LITERAL)
    pizzas[3] REGISTRO (cod_piz, valor NUMÉRICO nome_piz
    ↪ LITERAL)
    pedidos[6] REGISTRO (num_ped, cod_piz, cod_mot, situacao
    ↪ NUMÉRICO
                                fone LITERAL)
```

```

cont_piz, cont_ped, cont_cli NUMÉRICO
i, j, k, op, ped_aux, piz_aux, total_mot[5], NUMÉRICO
maior, sit_aux, total_cli[5] NUMÉRICO
fone_aux LITERAL
cont_piz ← 1
cont_ped ← 1
cont_cli ← 1
REPITA
ESCREVA "1-Cadastro de cliente"
ESCREVA "2-Cadastro de pizza"
ESCREVA "3-Cadastro de pedido"
ESCREVA "4-Despachar pizza"
ESCREVA "5-Receive pizza"
ESCREVA "6-Mostrar total gasto por cliente"
ESCREVA "7-Mostrar todas os pedidos com uma determinada situação"
ESCREVA "8-Mostrar motoqueiro que fez mais entregas"
ESCREVA "9-Finalizar"
ESCREVA "Digite sua opção: "
LEIA op
SE (op = 1)
ENTÃO INÍCIO
    SE (cont_cli <= 5)
        ENTÃO INÍCIO
            LEIA fone_aux
            i ← 1
            ENQUANTO (i <= cont_cli) E
                (clientes[i].fone ≠ fone_aux) FAÇA
                    INÍCIO
                    i ← i + 1
                    FIM
            SE (i <= cont_cli)
                ENTÃO ESCREVA "Cliente já cadastrado!"
            SENÃO INÍCIO
                clientes[cont_cli].fone ← fone_aux
                LEIA (clientes[cont_cli].nome_cli)
                LEIA (clientes[cont_cli].endereco)
                LEIA (clientes[cont_cli].comple)
                LEIA (clientes[cont_cli].cep)
                cont_cli ← cont_cli + 1
                FIM
            FIM
            SENÃO ESCREVA "Já foram cadastrados 5 clientes!"
        FIM
    SE (op = 2)
    ENTÃO INÍCIO
        SE (cont_piz <= 3)
            ENTÃO INÍCIO
                LEIA piz_aux
                i ← 1
                ENQUANTO (i <= cont_piz) E
                    (pizzas[i].cod_piz ≠ piz_aux) FAÇA
                        INÍCIO
                        i ← i + 1
                        FIM
                SE (i <= cont_piz)
                    ENTÃO ESCREVA "Pizza já cadastrada!"
                SENÃO INÍCIO
                    pizzas[cont_piz].cod_piz ← piz_aux
                    LEIA (pizzas[cont_piz].nome_piz)
                    LEIA (pizzas[cont_piz].valor)
                    cont_piz ← cont_piz + 1
                    FIM
            FIM
    
```

```

SENÃO ESCREVA "Já foram cadastradas 3 pizzas!"
FIM
SE (op = 3)
ENTÃO INÍCIO
    SE (cont_ped <= 6)
        ENTÃO INÍCIO
            LEIA ped_aux
            i ← 1
            ENQUANTO (i <= cont_ped) E
                (pedidos[i].num_ped ≠ ped_aux) FAÇA
                    INÍCIO
                    i ← i + 1
                    FIM
            SE (i <= cont_ped)
                ENTÃO ESCREVA "Pedido já cadastrado!"
            SENÃO INÍCIO
                LEIA (pedidos[cont_ped].fone)
                i ← 1
                ENQUANTO (i <= cont_cli) E
                    (clientes[i].fone ≠ pedidos[cont_ped].fone) FAÇA
                    INÍCIO
                    i ← i + 1
                    FIM
                SE (i > cont_cli)
                    ENTÃO ESCREVA "Cliente não cadastrado!"
                SENÃO INÍCIO
                    LEIA pedidos[cont_ped].cod_piz
                    i ← 1
                    ENQUANTO (i <= cont_piz) E
                        (pizzas[i].cod_piz ≠
                         pedidos[cont_ped].cod_piz) FAÇA
                        INÍCIO
                        i ← i + 1
                        FIM
                    SE (i > cont_piz)
                        ENTÃO ESCREVA "Pizza não cadastrada!"
                    SENÃO INÍCIO
                        pedidos[cont_ped].num_ped ← ped_aux
                        pedidos[cont_ped].cod_mot ← 0
                        pedidos[cont_ped].situacao ← 1
                        cont_ped ← cont_ped + 1
                        FIM
                    FIM
                FIM
            SENÃO ESCREVA "Já foram cadastrados 6 pedidos!"
            FIM
        SE (op = 4)
        ENTÃO INÍCIO
            LEIA ped_aux
            i ← 1
            ENQUANTO (i <= cont_ped) E
                (pedidos[i].num_ped ≠ ped_aux) FAÇA
                    INÍCIO
                    i ← i + 1
                    FIM
            SE (i > cont_ped)
                ENTÃO ESCREVA "Pedido não cadastrado!"
            SENÃO INÍCIO
                SE (pedidos[i].situacao ≠ 1)
                    ENTÃO ESCREVA "Pedido já foi despachado!"
                SENÃO INÍCIO
                    REPITA

```

```

        LEIA pedidos[i].cod_mot
        ATÉ (pedidos[i].cod_mot >= 1) E
            (pedidos[i].cod_mot <= 5)
            pedidos[i].situacao ← 2
            FIM
        FIM
    FIM
SE (op = 5)
ENTÃO INÍCIO
    LEIA ped_aux
    i ← 1
    ENQUANTO (i <= cont_ped) E (pedidos[i].num_ped ≠ ped_aux) FAÇA
        INÍCIO
        i ← i + 1
        FIM
    SE (i > cont_ped)
    ENTÃO ESCREVA "Pedido não cadastrado!"
    SENÃO INÍCIO
        SE (pedidos[i].situacao = 1)
        ENTÃO ESCREVA "Pedido ainda não foi despachado!"
        SENÃO SE (pedidos[i].situacao = 3)
            ENTÃO ESCREVA "Pedido já foi entregue!"
            SENÃO pedidos[i].situacao ← 3
        FIM
    FIM
SE (op = 6)
ENTÃO INÍCIO
    PARA i ← 1 ATÉ 5 FAÇA
        INÍCIO
        total_cli[i] ← 0
        FIM
    PARA i ← 1 ATÉ cont_ped - 1 FAÇA
        INÍCIO
        j ← 1
        ENQUANTO (j <= cont_cli) E
            (clientes[j].fone ≠ pedidos[i].fone) FAÇA
            INÍCIO
            j ← j + 1
            FIM
        k ← 1
        ENQUANTO (k <= cont_piz) E
            (pizzas[k].cod_piz ≠ pedidos[i].cod_piz) FAÇA
            INÍCIO
            k ← k + 1
            FIM
        total_cli[j] ← total_cli[j] + pizzas[k].valor
    FIM
    PARA i ← 1 ATÉ cont_cli - 1 FAÇA
        INÍCIO
        ESCREVA (clientes[i].nome_cli)
        ESCREVA (total_cli[i])
    FIM
FIM
SE (op = 7)
ENTÃO INÍCIO
    SE cont_ped = 0
    ENTÃO ESCREVA "Nenhum pedido foi cadastrado"
    SENÃO INÍCIO
        LEIA sit_aux
        SE (sit_aux <> 1) E (sit_aux <> 2) E (sit_aux <> 3)
        ENTÃO ESCREVA "Situação inexistente"
        SENÃO INÍCIO
            j := 0;

```

```

PARA i ← 1 ATÉ cont_ped - 1 FAÇA
INÍCIO
    SE (pedidos[i].situacao = sit_aux)
    ENTÃO INÍCIO
        ESCREVA (pedidos[i].num_ped)
        j ← 1
        ENQUANTO (j <= cont_piz) E
            (pizzas[j].cod_piz ≠ pedidos[i].cod_piz)
            • FAÇA
            INÍCIO
                j ← j + 1
            FIM
            ESCREVA (pizzas[j].nome_piz)
            FIM
        FIM
        SE j = 0
            ESCREVA "Nenhum pedido nesta situação"
        FIM
    FIM
SE (op = 8)
ENTÃO INÍCIO
    PARA i ← 1 ATÉ 5 FAÇA
        INÍCIO
            total_mot[i] ← 0
        FIM
    PARA i ← 1 ATÉ cont_ped - 1 FAÇA
        INÍCIO
            total_mot[pedidos[i].cod_mot] ←
            • total_mot[pedidos[i].cod_mot] + 1
        FIM
    PARA i ← 1 ATÉ cont_cli - 1 FAÇA
        INÍCIO
            SE (i = 1)
            ENTÃO INÍCIO
                maior ← total_mot[i]
                j ← i
            FIM
            SENÃO INÍCIO
                SE (total_mot[i] > maior)
                ENTÃO INÍCIO
                    maior ← total_mot[i]
                    j ← i
                FIM
            FIM
            FIM
            SE maior = 0
            ENTÃO ESCREVA "Nenhuma entrega foi realizada"
            SENÃO INÍCIO
                ESCREVA "O motoqueiro ", j, " fez mais entregas"
                ESCREVA "Total geral de entregas: ", maior
            FIM
        FIM
    ATÉ (op = 9)
    FIM_ALGORITMO.

```



#### SOLUÇÃO:

\EXERC\CAP8\PASCAL\EX15.PAS e \EXERC\CAP8\PASCAL\EX15.EXE



#### SOLUÇÃO:

\EXERC\CAP8\C++\EX15.CPP e \EXERC\CAP8\C++\EX15.EXE

 **16.** Faça um programa que solucione o problema de preenchimento das vagas nos cursos de uma universidade. Cada aluno que prestou vestibular em uma determinada universidade originou um registro com os seguintes campos: número de inscrição, idade, pontuação alcançada (de 0 a 5.000) e código do curso pretendido.

A universidade oferece seis cursos com 40 vagas cada. O problema consiste em distribuir os candidatos entre os cursos, de acordo com a nota final e com a opção apresentada pelo candidato. Em caso de empate, será atendido, primeiro, o candidato com maior idade.

Sabe-se que o final da leitura dos dados será determinado pelo campo de inscrição negativo.

Durante a leitura, os dados deverão ser repassados aos vetores (um vetor para cada curso).

Como existe limite de vagas e não há ordem na leitura, o programa deverá criar vetores ordenados de maneira decrescente pela pontuação. Assim, aqueles que não conseguiram as melhores 40 pontuações serão descartados.

Mostre a lista de alunos que foram aprovados em cada curso.

**ALGORITMO****SOLUÇÃO:**

```

ALGORITMO
DECLARE cand[6,40] REGISTRO (num_insc, idade, pontos, cod_curso
    ↪ NUMÉRICO)
    cont[6], i, j, k, insc_aux, idade_aux NUMÉRICO
    pontos_aux, curso_aux NUMÉRICO
PARA i ← 1 ATÉ 6 FAÇA
    INÍCIO
        cont[i] ← 40
    FIM
LEIA insc_aux
ENQUANTO insc_aux > 0 FAÇA
    INÍCIO
        LEIA idade_aux
        REPITA
            LEIA pontos_aux
        ATÉ (pontos_aux >= 0) E (pontos_aux <= 5000)
        REPITA
            LEIA curso_aux
        ATÉ (curso_aux >= 1) E (curso_aux <= 6)
        i ← curso_aux
        SE cont[i] = 40
        ENTÃO INÍCIO
            cand[i,40].num_insc ← insc_aux
            cand[i,40].idade ← idade_aux
            cand[i,40].pontos ← pontos_aux
            cand[i,40].cod_curso ← curso_aux
            cont[i] ← cont[i] - 1
        FIM
        SENÃO INÍCIO
            j ← 40
            ENQUANTO (cand[i][j].pontos < pontos_aux) E
                (j>cont[i]) FAÇA
                INÍCIO
                    j ← j - 1
                FIM
            SE cont[i] <= 0
            ENTÃO INÍCIO
                PARA k ← 40 ATÉ j+1 FAÇA

```

```

        INÍCIO
        cand[i,k] ← cand[i,k-1]
        FIM
        cand[i,j+1].num_insc ← insc_aux
        cand[i,j+1].idade ← idade_aux
        cand[i,j+1].pontos ← pontos_aux
        cand[i,j+1].cod_curso ← curso_aux
        cont[i] ← cont[i] - 1
        FIM
SENÃO SE j = cont[i]
ENTÃO INÍCIO
        cand[i,j].num_insc ← insc_aux
        cand[i,j].idade ← idade_aux;
        cand[i,j].pontos ← pontos_aux
        cand[i,j].cod_curso ← curso_aux
        cont[i] ← cont[i] - 1
        FIM
SENÃO INÍCIO
    ENQUANTO (cand[i,j].pontos = pontos_aux)
    E (cand[i,j].idade < idade_aux) E
    ↪ (j > cont[i]) FAÇA
    INÍCIO
        j ← j - 1
    FIM
    SE j >= cont[i]
    ENTÃO INÍCIO
        PARA k ← cont[i] ATÉ j-1 FAÇA
        INÍCIO
            cand[i,k] ← cand[i,k+1]
        FIM
        cand[i,j].num_insc ← insc_aux
        cand[i,j].idade ← idade_aux
        cand[i,j].pontos ← pontos_aux
        cand[i,j].cod_curso ← curso_aux
        cont[i] ← cont[i] - 1
        FIM
    SENÃO INÍCIO
        cand[i,j].num_insc ← insc_aux
        cand[i,j].idade ← idade_aux
        cand[i,j].pontos ← pontos_aux
        cand[i,j].cod_curso ← curso_aux
        cont[i] ← cont[i] - 1
        FIM
    FIM
    FIM
LEIA insc_aux
FIM
PARA i ← 1 ATÉ 6 FAÇA
INÍCIO
SE cont[i] ≠ 40
ENTÃO INÍCIO
    ESCREVA "Lista de aprovados no curso ",i
    SE cont[i] <= 0
    ENTÃO INÍCIO
        PARA j ← 1 ATÉ 40 FAÇA
        INÍCIO
            ESCREVA cand[i][j].num_insc
            ESCREVA cand[i][j].idade
            ESCREVA cand[i][j].pontos
        FIM
    FIM
SE cont[i] > 0

```

```

ENTÃO INÍCIO
PARA j ← cont[i]+1 ATÉ 40 FAÇA
INÍCIO
ESCREVA cand[i][j].num_insc
ESCREVA cand[i][j].idade
ESCREVA cand[i][j].pontos
FIM
FIM
FIM
FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP8\PASCAL\EX16.PAS e \EXERC\CAP8\PASCAL\EX16.EXE

**SOLUÇÃO:**

\EXERC\CAP8\C++\EX16.CPP e \EXERC\CAP8\C++\EX16.EXE

**17.** Faça um programa que gerencie uma locadora de fitas de videocassete. Como é uma locadora nova, ela não deve ter mais de dez clientes. Cada cliente tem os seguintes dados: código, nome, sexo, data de nascimento, RG, CPF, endereço, cidade, estado, número total de fitas já locadas e número de fitas que estão locadas atualmente. O código do cliente é o seu número no registro da locadora. Faça o seguinte:

- crie uma rotina que inclua um novo cliente. O número total de fitas locadas e o número de fitas que estão locadas atualmente devem ser zero;
- crie uma rotina que mostre os clientes cadastrados;
- crie uma rotina que remova um cliente, desde que ele não esteja com fitas locadas no momento;
- crie uma rotina que faça a locação de novas fitas a um cliente, desde que ele não esteja com nenhuma em seu poder. Deve-se entrar com o código do cliente e solicitar o número de fitas que deseja locar (nesse momento o campo 'fitas locadas' atualmente deve ser atualizado);
- crie uma rotina para devolução de fitas. Deve-se solicitar o código do cliente e, se esse for encontrado, deve-se perguntar quantas fitas estão sendo devolvidas (o cliente não pode devolver mais fitas que o valor do campo 'número de fitas locadas atualmente'). Quando efetivar a devolução, os campos 'fitas locadas atualmente' e 'total de fitas já locadas' devem ser atualizados;
- mostre os clientes que estão com fitas locadas.

**SOLUÇÃO:**

```

ALGORITMO
DECLARE clientes[10] REGISTRO (cod, fitas_loc, tot_fitas_loc
                                ↪ NUMÉRICO
                                nome, sexo, data_nas, RG, CPF,
                                ↪ endereço, cidade, estado LITERAL)
                                cont_c, i, j, cod_aux, qtd, op NUMÉRICO
cont_c ← 1
REPITA
    ESCREVA "1-Cadastrar cliente "
    ESCREVA "2-Mostrar clientes "
    ESCREVA "3-Remover cliente "

```

```

ESCREVA "4-Devolução de fitas "
ESCREVA "5-Efetuar locação "
ESCREVA "6-Mostrar clientes com fitas locadas"
ESCREVA "7-Finalizar "
ESCREVA "Digite sua opção: "
LEIA op
SE (op = 1)
ENTÃO INÍCIO
    SE (cont_c <= 10)
        ENTÃO INÍCIO
            LEIA cod_aux
            i ← 1
            ENQUANTO (i <= cont_c) E
                (clientes[i].cod ≠ cod_aux) FAÇA
                    INÍCIO
                        i ← i + 1
                    FIM
            SE (i <= cont_c)
                ENTÃO ESCREVA "Cliente já cadastrado! "
            SENÃO INÍCIO
                clientes[cont_c].cod ← cod_aux
                LEIA clientes[cont_c].nome
                LEIA clientes[cont_c].sexo
                LEIA clientes[cont_c].data_nas
                LEIA clientes[cont_c].RG
                LEIA clientes[cont_c].CPF
                LEIA clientes[cont_c].endereco
                LEIA clientes[cont_c].cidade
                LEIA clientes[cont_c].estado
                clientes[cont_c].fitas_loc ← 0
                clientes[cont_c].tot_fitas_loc ← 0
                cont_c ← cont_c + 1
                ESCREVA "Cliente cadastrado com sucesso"
            FIM
        FIM
    SENÃO ESCREVA "10 clientes já foram cadastrados! "
FIM
SE (op = 2)
ENTÃO INÍCIO
    PARA i ← 1 ATÉ cont_c - 1 FAÇA
        INÍCIO
            ESCREVA clientes[i].cod
            ESCREVA clientes[i].nome
            ESCREVA clientes[i].sexo
            ESCREVA clientes[i].data_nas
            ESCREVA clientes[i].RG
            ESCREVA clientes[i].CPF
            ESCREVA clientes[i].endereco
            ESCREVA clientes[i].cidade
            ESCREVA clientes[i].estado
            ESCREVA clientes[i].fitas_loc
            ESCREVA clientes[i].tot_fitas_loc
        FIM
    FIM
SE (op = 3)
ENTÃO INÍCIO
    LEIA cod_aux
    i ← 1
    ENQUANTO (i <= cont_c) E (clientes[i].cod ≠ cod_aux) FAÇA
        INÍCIO
            i ← i + 1
        FIM
    SE (i <= cont_c)

```

```

ENTÃO INÍCIO
    SE (clientes[i].fitas_loc > 0)
        ENTÃO ESCREVA "Cliente não pode ser excluído, possui
        ↪ fitas locadas! "
    SENÃO INÍCIO
        PARA j ← i+1 ATÉ cont_c - 1 FAÇA
            INÍCIO
                clientes[j-1] ← clientes[j]
            FIM
        cont_c ← cont_c - 1
        ESCREVA "Cliente removido com sucesso"
        FIM
    FIM
    SENÃO ESCREVA "Cliente não cadastrado! "
    FIM
SE (op = 4)
ENTÃO INÍCIO
    LEIA cod_aux
    i ← 1
    ENQUANTO (i <= cont_c) E (clientes[i].cod ≠ cod_aux) FAÇA
        INÍCIO
            i ← i + 1
        FIM
    SE (i <= cont_c)
    ENTÃO INÍCIO
        SE clientes[i].fitas_loc = 0
        ENTÃO ESCREVA "Este cliente não possui fitas locadas"
        SENÃO INÍCIO
            REPITA
                LEIA qtd
                SE (qtd > clientes[i].fitas_loc)
                    ENTÃO INÍCIO
                        ESCREVA "Cliente possui apenas",
                        ↪ clientes[i].fitas_loc, "locadas"
                        ESCREVA "Digite a quantidade correta"
                        LEIA qtd
                    FIM
                    ATÉ (qtd <= clientes[i].fitas_loc)
                    clientes[i].fitas_loc ← clientes[i].fitas_loc -
                    ↪ qtd
                    ESCREVA "Devolução efetuada com sucesso"
                    FIM
                FIM
            SENÃO ESCREVA "Cliente não cadastrado! "
            FIM
    FIM
SE (op = 5)
ENTÃO INÍCIO
    LEIA cod_aux
    i ← 1
    ENQUANTO (i <= cont_c) E (clientes[i].cod ≠ cod_aux) FAÇA
        INÍCIO
            i ← i + 1
        FIM
    SE (i <= cont_c)
    ENTÃO INÍCIO
        SE (clientes[i].fitas_loc > 0)
            ENTÃO INÍCIO
                ESCREVA "Este cliente não pode fazer novas
                ↪ locações"
                ESCREVA "pois possui fitas em seu poder"
            FIM
        SENÃO INÍCIO
            LEIA qtd

```

```

        clientes[i].fitas_loc ← qtd
        clientes[i].tot_fitas_loc ←
        ↪ clientes[i].tot_fitas_loc + qtd
        ESCREVA "Locação efetuada com sucesso"
        FIM
    FIM
    SENÃO  ESCREVA "Cliente não cadastrado! "
    FIM
SE (op = 6)
ENTÃO INÍCIO
    PARA i ← 1 ATÉ cont_c - 1 FAÇA
        INÍCIO
            SE (clientes[i].fitas_loc > 0)
                ENTÃO INÍCIO
                    ESCREVA clientes[i].cod
                    ESCREVA clientes[i].nome
                    ESCREVA clientes[i].sexo
                    ESCREVA clientes[i].data_nas
                    ESCREVA clientes[i].RG
                    ESCREVA clientes[i].CPF
                    ESCREVA clientes[i].endereco
                    ESCREVA clientes[i].cidade
                    ESCREVA clientes[i].estado
                    ESCREVA clientes[i].fitas_loc
                    ESCREVA clientes[i].tot_fitas_loc
                FIM
            FIM
        ATÉ (op = 7)
    FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP8\PASCAL\EX17.PAS e \EXERC\CAP8\PASCAL\EX17.EXE

**SOLUÇÃO:**

\EXERC\CAP8\C++\EX17.CPP e \EXERC\CAP8\C++\EX17.EXE

- 18.** Considere que um médico armazena algumas informações sobre os seus 20 pacientes (nome, idade, sexo, altura e peso). Crie um programa que leia essas informações e determine:

- ◆ o nome da pessoa mais pesada;
- ◆ nomes e idades das pessoas que estejam acima do seu peso ideal;
- ◆ nomes das pessoas que estejam abaixo do seu peso ideal, mostrando, ainda, o peso que essas pessoas deverão adquirir para atingirem esse peso ideal.

**OBSERVAÇÃO:**

Utilize esses cálculos para determinar o peso ideal.

Homens:  $(72.7 * \text{altura}) - 58$

Mulheres:  $(62.1 * \text{altura}) - 44.7$

**SOLUÇÃO:**

```

ALGORITMO
DECLARE pac[20] REGISTRO (nome, sexo LITERAL; idade, altura, peso
    ↪ NUMÉRICO)
    op, i, j, cont_p, maior_peso, peso_ideal, dif NUMÉRICO
    nome_aux[30] LITERAL

```

```

cont_p ← 1
REPITA
    ESCREVA "1-Cadastrar paciente"
    ESCREVA "2-Determinar paciente mais pesada"
    ESCREVA "3-Mostrar pacientes acima do peso ideal"
    ESCREVA "4-Mostrar pacientes abaixo do peso ideal"
    ESCREVA "5-Finalizar"
    ESCREVA "Digite sua opção: "
    LEIA op
    SE op = 1
        ENTÃO INÍCIO
            SE cont_p < 20
                ENTÃO INÍCIO
                    ESCREVA "Digite o nome do paciente: "
                    LEIA pac[cont_p].nome
                    REPITA
                        ESCREVA "Digite o sexo do paciente (M ou F): "
                        LEIA pac[cont_p].sexo
                        ATÉ (pac[cont_p].sexo = 'F') OU
                            (pac[cont_p].sexo = 'M');
                        ESCREVA "Digite a idade do paciente: "
                        LEIA pac[cont_p].idade
                        ESCREVA "Digite a altura do paciente (em metros): "
                        LEIA pac[cont_p].altura
                        ESCREVA "Digite o peso do paciente (em Kg): "
                        LEIA pac[cont_p].peso
                        cont_p ← cont_p + 1
                        ESCREVA "Paciente cadastrado com sucesso"
                    FIM
                SENÃO ESCREVA "Já foram cadastrados 20 pacientes!"
                FIM
            SE op = 2
                ENTÃO INÍCIO
                    SE cont_p = 1
                    ENTÃO ESCREVA "Nenhum paciente foi cadastrado"
                    SENÃO INÍCIO
                        PARA i ← 1 ATÉ cont_p-1 FAÇA
                            INÍCIO
                                SE i = 1
                                ENTÃO INÍCIO
                                    maior_peso ← pac[i].peso
                                    nome_aux ← pac[i].nome
                                    FIM
                                SENÃO INÍCIO
                                    SE (pac[i].peso > maior_peso)
                                    ENTÃO INÍCIO
                                        maior_peso ← pac[i].peso
                                        nome_aux ← pac[i].nome
                                        FIM
                                    FIM
                                FIM
                            FIM
                        ESCREVA "Nome do paciente mais pesado: ", nome_aux
                        ESCREVA "Peso: ", maior_peso
                        FIM
                    FIM
                SE op = 3
                ENTÃO INÍCIO
                    SE cont_p = 1
                    ENTÃO ESCREVA "Nenhum paciente foi cadastrado"
                    SENÃO INÍCIO
                        PARA i ← 1 ATÉ cont_p -1 FAÇA
                            INÍCIO
                                SE pac[i].sexo = 'F'

```

```

ENTÃO peso_ideal ← (62.1 * pac[i].altura) - 44.7
SENÃO peso_ideal ← (72.7 * pac[i].altura) - 58
SE (pac[i].peso > peso_ideal)
ENTÃO INÍCIO
    ESCREVA "Nome do paciente: ",pac[i].nome
    ESCREVA "Sexo do paciente: ",pac[i].nome
    ESCREVA "Idade do paciente: ",pac[i].idade
    ESCREVA "Altura do paciente (em metros):
    ↪ ",pac[i].altura
    ESCREVA "Peso do paciente (em Kg): ",pac[i].peso
FIM
FIM
FIM
SE op = 4
ENTÃO INÍCIO
    SE cont_p = 1
        ENTÃO ESCREVA "Nenhum paciente foi cadastrado"
        SENÃO INÍCIO
            j: ← 0;
            PARA i ← 1 ATÉ cont_p - 1 FAÇA
            INÍCIO
                SE pac[i].sexo = 'F'
                    ENTÃO peso_ideal ← (62.1 * pac[i].altura) - 44.7
                    SENÃO peso_ideal ← (72.7 * pac[i].altura) - 58
                    SE (pac[i].peso < peso_ideal)
                        ENTÃO INÍCIO
                            j: ← 1;
                            dif ← peso_ideal - pac[i].peso
                            ESCREVA "Nome do paciente: ",pac[cont_p].nome
                            ESCREVA "Peso atual: ",pac[i].peso
                            ESCREVA "Precisa adquirir ",dif, " Kg para
                            ↪ atingir seu peso ideal"
                        FIM
                    FIM
                SE j = 0
                    ENTÃO ESCREVA "Nenhuma pessoa está abaixo do peso ideal"
                FIM
            FIM
        ATÉ op = 5
    FIM_ALGORITMO.

```



#### SOLUÇÃO:

\EXERC\CAP8\PASCAL\EX18.PAS e \EXERC\CAP8\PASCAL\EX18.EXE



#### SOLUÇÃO:

\EXERC\CAP8\C++\EX18.CPP e \EXERC\CAP8\C++\EX18.EXE

**19.** Faça um programa que controle o estoque de uma loja de brinquedos. Atualmente existem 40 itens, cada um contendo código, descrição, preço de compra, preço de venda, quantidade em estoque e estoque mínimo.

- crie uma rotina para cadastrar os produtos;
- crie uma rotina para mostrar o valor do lucro obtido com a venda de cada produto e o percentual que esse valor representa;
- crie uma rotina que mostre os produtos com quantidade em estoque abaixo do estoque mínimo permitido.

**ALGORITMO****SOLUÇÃO:**

```

ALGORITMO
DECLARE brinquedos[40] REGISTRO (cod, qtd_est, est_min, p_compra,
                                     p_venda NÚMERO
                                     descr LITERAL)
      i, cont_b, op, cod_aux, lucro, perc, achou NUMÉRICO
cont_b ← 1
REPITA
  ESCREVA "1-Cadastrar brinquedo"
  ESCREVA "2-Mostrar lucro"
  ESCREVA "3-Mostrar produtos com estoque abaixo do estoque mínimo"
  ESCREVA "4-Finalizar"
  ESCREVA "Digite sua opção "
  LEIA op
  SE (op = 1)
    ENTÃO INÍCIO
      SE (cont_b = 41)
        ENTÃO ESCREVA "Já foram cadastrados os 40 brinquedos!"
        SENÃO INÍCIO
          LEIA cod_aux
          i ← 1
          ENQUANTO (i <= cont_b) E
            (brinquedos[i].cod ≠ cod_aux) FAÇA
              INÍCIO
                i ← i + 1
              FIM
            SE (i <= cont_b)
              ENTÃO ESCREVA "Já existe brinquedo com este código!"
              SENÃO INÍCIO
                brinquedos[cont_b].cod ← cod_aux
                LEIA brinquedos[cont_b].descr
                LEIA brinquedos[cont_b].qtd_est
                LEIA brinquedos[cont_b].est_min
                LEIA brinquedos[cont_b].p_compra
                LEIA brinquedos[cont_b].p_venda
                cont_b ← cont_b + 1
                ESCREVA "Brinquedo cadastrado com sucesso"
              FIM
            FIM
          FIM
        FIM
      SE (op = 2)
        ENTÃO INÍCIO
          LEIA cod_aux
          i ← 1
          ENQUANTO (i <= cont_b) E (brinquedos[i].cod ≠ cod_aux) FAÇA
            INÍCIO
              i ← i + 1
            FIM
          SE (i <= cont_b)
            ENTÃO INÍCIO
              lucro ← brinquedos[i].p_venda - brinquedos[i].p_compra
              perc ← lucro / brinquedos[i].p_compra * 100
              ESCREVA lucro, perc
            FIM
          SENÃO ESCREVA "Brinquedo não cadastrado!"
        FIM
      SE (op = 3)
        ENTÃO INÍCIO
          achou ← 0
          PARA i ← 1 ATÉ cont_b - 1 FAÇA
            INÍCIO
              SE (brinquedos[i].qtd_est < brinquedos[i].est_min)

```

```

ENTÃO INÍCIO
achou ← 1
ESCREVA brinquedos[i].cod
ESCREVA brinquedos[i].descr
ESCREVA brinquedos[i].qtd_est
ESCREVA brinquedos[i].est_min
FIM
FIM
SE achou = 0
ENTÃO ESCREVA "Nenhum brinquedo está com estoque abaixo do
➥ estoque mínimo"
FIM
ATÉ (op = 4)
FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP8\PASCAL\EX19.PAS e \EXERC\CAP8\PASCAL\EX19.EXE

**SOLUÇÃO:**

\EXERC\CAP8\C++\EX19.CPP e \EXERC\CAP8\C++\EX19.EXE

- 20.** A prefeitura de uma cidade fez uma pesquisa entre os seus habitantes, coletando dados sobre o salário, idade, sexo e número de filhos.

Crie um programa que leia os dados de um número indeterminado de pessoas e, ao final, mostre:

- a média de idade das mulheres com salário inferior a R\$ 300,00;
- a média de salário da população;
- a média do número de filhos;
- o maior salário;
- a menor idade.

**OBSEVAÇÃO:**

A leitura termina quando for digitada idade igual a zero.

**SOLUÇÃO:**

```

ALGORITMO
DECLARE pessoa REGISTRO (idade, filhos, salario NUMÉRICO sexo
➥ LITERAL)
media_id, media_sa1, media_sa2 NUMÉRICO
media_fi, maior_sa, soma_sa1, soma_sa2, soma_fi NUMÉRICO
menor_id, cont1, cont2 NUMÉRICO
soma_sa1 ← 0
soma_sa2 ← 0
soma_fi ← 0
cont1 ← 0
cont2 ← 0
menor_id ← 0
maior_sa ← 0
LEIA pessoa.idade
ENQUANTO (pessoa.idade > 0) FAÇA
INÍCIO
LEIA pessoa.salario
LEIA pessoa.sexo
LEIA pessoa.filhos
soma_sa1 ← soma_sa1 + pessoa.salario
cont1 ← cont1 + 1

```

```

soma_fi ← soma_fi + pessoa.filhos
SE (pessoa.sexo = 'F') E (pessoa.salario < 300)
    ENTÃO INÍCIO
        soma_sa2 ← soma_sa2 + pessoa.salario
        cont2 ← cont2 + 1
    FIM
SE (cont1 = 1)
    ENTÃO maior_sa ← pessoa.salario
    SENÃO SE (pessoa.salario > maior_sa)
        ENTÃO maior_sa ← pessoa.salario
    SE (cont1 = 1)
        ENTÃO menor_id ← pessoa.idade
        SENÃO SE (pessoa.idade < menor_id)
            ENTÃO menor_id ← pessoa.idade
LEIA pessoa.idade
FIM
SE (cont2 > 0)
    ENTÃO media_sa2 ← soma_sa2 / cont2
    SENÃO media_sa2 ← 0
ESCREVA media_sa2
SE (cont1 > 0)
    ENTÃO INÍCIO
        media_sal ← soma_sal / cont1
        media_fi ← soma_fi / cont1
    FIM
SENÃO INÍCIO
    media_sal ← 0
    media_fi ← 0
FIM
ESCREVA media_sal
ESCREVA media_fi
ESCREVA maior_sa
ESCREVA menor_id
FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP8\PASCAL\EX20.PAS e \EXERC\CAP8\PASCAL\EX20.EXE

**SOLUÇÃO:**

\EXERC\CAP8\C++\EX20.CPP e \EXERC\CAP8\C++\EX20.EXE

## EXERCÍCIOS PROPOSTOS

**1.** Uma empresa deseja controlar as vendas realizadas por seus vendedores a cada mês, durante um ano inteiro. Sabe-se que nessa empresa existem quatro vendedores.

É importante que esse controle seja automatizado porque muitas consultas devem ser respondidas imediatamente. O gerente precisa de um meio para cadastrar as vendas de todos os vendedores e, depois, precisa ver um menu, contendo as seguintes opções:

1. cadastrar vendedor;
2. cadastrar venda;
3. consultar as vendas de um funcionário em um determinado mês;
4. consultar o total das vendas de um determinado vendedor;
5. mostrar o número do vendedor que mais vendeu em um determinado mês;

6. mostrar o número do mês com mais vendas;
7. finalizar o programa.

**Na opção 1:** deve-se cadastrar vendedores, sendo que não pode existir dois vendedores com o mesmo código.

**Na opção 2:** deve-se cadastrar vendas, informar o código do vendedor e o mês das vendas, mas não podem existir duas vendas para o mesmo vendedor no mesmo mês.

**Na opção 3:** deve-se informar o número do vendedor e o número do mês que se deseja consultar, para então descobrir e mostrar esse valor.

**Na opção 4:** deve-se informar o número do vendedor desejado, calcular e mostrar o total de suas vendas.

**Na opção 5:** deve-se informar o número do mês que se deseja pesquisar, para então descobrir e mostrar o número do vendedor que mais vendeu nesse mês.

**Na opção 6:** deve-se descobrir e mostrar o mês com maior venda.

**Na opção 7:** o programa termina.

**2.** A prefeitura de uma cidade fez uma pesquisa entre os seus habitantes, coletando dados sobre o salário, idade e número de filhos. Faça um programa que leia esses dados de 20 pessoas, calcule e mostre:

- ◆ a média de salário da população;
- ◆ a média do número de filhos;
- ◆ o maior salário;
- ◆ o percentual de mulheres com salário superior a R\$ 1.000,00.

**3.** Foi realizada uma pesquisa de algumas características físicas de 50 habitantes de uma certa região. De cada habitante foram coletados os seguintes dados: sexo, altura, idade e cor dos olhos (A – Azuis, V – Verdes ou C – Castanhos).

Faça um programa que leia esses dados e armazene-os em um registro do tipo vetor. Determine:

- ◆ a média de idade das pessoas com olhos castanhos e altura superior a 1,60 m;
- ◆ a maior idade entre os habitantes;
- ◆ a quantidade de indivíduos do sexo feminino cuja idade esteja entre 20 e 45 anos (inclusive) ou que tenham olhos verdes e altura inferior a 1,70 m;
- ◆ o percentual de homens.

**4.** Foi realizada uma pesquisa entre 20 habitantes de uma cidade. De cada habitante foram coletados os dados: idade, sexo, renda familiar e número de filhos. Faça um programa que leia esses dados, armazenando-os em um vetor. Calcule e mostre a média de salário entre os habitantes, a menor e a maior idade do grupo e a quantidade de mulheres com mais de dois filhos e com renda familiar inferior a R\$ 600,00.

**5.** Faça um programa que leia o código, a descrição, o valor unitário e a quantidade em estoque dos 50 produtos comercializados por uma papelaria. Essas informações devem ser armazenadas em um registro do tipo vetor em ordem crescente de código.

Depois da leitura faça:

- ◆ uma rotina que permita alterar a descrição, o valor unitário e a quantidade em estoque de um determinado produto, que deverá ser localizado por meio da informação do seu código;

- ◆ uma rotina que mostre todos os produtos cuja descrição comecem com uma determinada letra (informada pelo usuário);
- ◆ mostre todos os produtos com quantidade em estoque inferior a cinco unidades.

**6.** Observe as informações a seguir:

CLIENTE	CONTA_BANCÁRIA
Número do cliente	Número da conta
Nome	Número do cliente
Telefone	Saldo
Endereço	

Crie um programa que faça o cadastramento de contas, verificando se o número do cliente titular dessa conta já foi previamente cadastrado em Clientes. Se existir, permitir a inclusão. Caso contrário, mostrar a mensagem *Cliente não cadastrado* e abrir uma tela que permita o cadastramento desse cliente. Mostre ao final, todas as contas cadastradas.

**7.** Considere que existe um registro com os seguintes atributos: `codigo_cliente` e `nome_cliente` e um outro registro com os seguintes atributos: `Nº_conta`, `valor_compra`, `codigo_cliente`. Faça um programa que:

- ◆ inclua clientes, não permitindo que dois clientes possuam o mesmo código;
- ◆ inclua contas, verificando se o código do cliente informado já está cadastrado. Caso não esteja, não permita a inclusão;
- ◆ remova um determinado cliente. Antes de executar a remoção, verifique se o cliente possui alguma compra. Se possuir, mostrar a mensagem *Exclusão não permitida*. Caso contrário, proceder à exclusão.

**8.** Foi feita uma estatística em 15 estados brasileiros para coletar dados sobre acidentes de trânsito. Em cada estado observou-se os seguintes aspectos:

- ◆ nome do estado;
- ◆ número de veículos que circulam nesse estado (em 2000);
- ◆ número de acidentes de trânsito (em 2000).

Deseja-se saber:

- qual o maior e o menor índice de acidentes de trânsito e o nome dos estados em que eles ocorreram;
- qual o percentual de veículos em cada estado;
- qual a média de acidentes em cada um dos estados.

**9.** Um funcionário recebe um salário fixo mais 6% de comissão sobre suas vendas. Crie um algoritmo que leia o salário do funcionário, o valor total de suas vendas, calcule a comissão e o salário final. Mostre todos os valores calculados.

**10.** Uma empresa armazena informações sobre as contas a receber de seus clientes. Cada uma dessas contas tem as seguintes informações: número do documento, código do cliente, data de vencimento, data de pagamento, valor da conta e juros. Faça um programa para cadastrar um documento. Se a data de pagamento for maior que a data de vencimento, o programa deve calcular o campo 'juros' da tabela documentos (a cada dia de atraso, deve-se aplicar 0,02% de multa). O programa deve ler informações sobre 15 documentos e, depois, mostrar todos os documentos lidos e o total geral a receber (valor das contas + juros) e a média dos juros.

**11.** Faça um programa que utilize as informações a seguir:

<b>MÉDICOS</b>		<b>PACIENTES</b>		<b>CONSULTAS</b>	
Cod_Medico		Cod_Pac		Num_Prontuario	
Nome		Nome_Pac		Data_Consulta	
Endereco		Endereco		Diagnostico	
Salario		idade		Cod_Medico	
				Cod_Paciente	

- crie uma rotina para realizar inclusão e alteração no cadastro de Pacientes;
- crie uma rotina para excluir um médico (lembre-se: se existir alguma consulta realizada por esse médico, o mesmo não poderá ser excluído);
- crie uma rotina para mostrar todas as consultas realizadas em uma data qualquer, escolhida pelo usuário (lembre-se de mostrar também o nome do médico e o nome do paciente envolvidos na consulta).

**12.** Utilizando os registros a seguir, faça um programa que:

<b>CLIENTE</b>		<b>RECEBIMENTOS</b>	
Cod_Cli		Num_doc	
Nome		Valor_doc	
Endereco		Data_Emissao	
Fone		Data_Vencimento	
		Cod_Cli	

- inclua Recebimentos (deve verificar se o cliente já se encontra cadastrado);
- altere o cadastro de clientes (o usuário deve informar o código do cliente que será alterado);
- mostre todos os recebimentos com data de vencimento dentro de um período qualquer. Não esqueça de mostrar também o nome do cliente e o total de dias em atraso (quando não houver atraso mostrar zero);
- considere que poderão ser cadastrados no máximo três recebimentos para cada cliente.

**13.** Considere as informações a seguir:

- ◆ Estilista (código do estilista, nome do estilista, salário);
- ◆ Roupa (código da roupa, descrição da roupa, código do estilista, código da estação, ano);
- ◆ Estação (código da estação, nome da estação);
- ◆ sabe-se que nessa indústria de confecção existem três estilistas. Crie uma rotina para cadastrá-los;
- ◆ crie uma rotina para cadastrar as estações climáticas (sabendo que são duas, primavera-verão e outono-inverno);
- ◆ crie uma rotina para cadastrar as roupas (lembre-se de que estilista e estação devem ter sido previamente cadastrados) – no máximo dez roupas por estação;
- ◆ crie um relatório que mostre todas as roupas de uma determinada estação (informando, inclusive, o nome do estilista que a desenhou).

- 14.** Utilize as informações a seguir para criar um controle automatizado de uma clínica médica. Sabe-se que essa clínica deseja ter um controle semanal (de 2<sup>a</sup> a 6<sup>a</sup> feira) das consultas realizadas. A cada dia podem ser realizadas, no máximo, duas consultas para cada médico. Considere que serão cadastrados três médicos e cinco pacientes.

```
Paciente(cod_pac, nome, endereço, fone)
Médico(cod_med, nome, fone, endereço)
Consulta(num_consulta, dia semana, hora, cod_med, cod_pac)
```

Crie rotinas para:

- a) cadastrar os pacientes, não permitindo dois pacientes com o mesmo código;
- b) cadastrar os médicos, não permitindo dois pacientes com o mesmo código;
- c) cadastrar as consultas, obedecendo às especificações apresentadas acima;
- d) consultar as consultas de um determinado médico em um certo dia da semana (2<sup>a</sup> a 6<sup>a</sup> feira);
- e) mostrar um relatório contendo todas as consultas realizadas em um dia.

- 15.** Um restaurante deseja criar um controle de qualidade sobre os pratos que oferece a seus clientes. Dessa maneira, deseja cadastrar algumas informações sobre as receitas, ingredientes e cozinheiros. As informações necessárias são descritas a seguir:

```
Receita(codigo da receita, nome da receita, total de calorias a cada
↳ 100g, codigo_cozinheiro)
Ingredientes (codigo do ingrediente, descrição)
Ingredientes/Receita(codigo do ingrediente, codigo da receita,
↳ quantidade, unidade de medida)

◆ cadastre os cozinheiros (existem apenas três nesse restaurante);
◆ cadastre os ingredientes (existem, no máximo, 15);
◆ cadastre as receitas (existem 20 receitas que utilizam, no máximo, três ingredientes cada);
◆ mostre todas as receitas de um determinado cozinheiro;
◆ mostre todas as receitas cujo total de calorias esteja dentro de um intervalo especificado;
◆ mostre o total de receitas elaboradas por cada um dos cozinheiros;
```

- 16.** O acervo de uma biblioteca precisa ser informatizado. Para tanto, as principais informações das obras foram assim estruturadas:

```
Obra(numero do tombo, numero do exemplar, data compra)
Tombo(numero do tombo, nome da obra, nome do autor, nome da editora,
↳ codigo da área)
```

Sabe-se que existem 20 tombos e, para cada um, existem, no máximo, três exemplares.

Defina vetores de registros para armazenar tais informações.

Defina um menu de opções a seguir:

1. Cadastrar tombos
2. Cadastrar obras
3. Mostrar obras por área
4. Mostrar obras por autor
5. Mostrar obras por editora
6. Encerrar o programa

**OBSERVAÇÃO:** Existem três áreas: 1 – Exatas, 2 – Humanas e sociais e 3 – Biomédicas

**17.** Um banco está informatizando seus controles de clientes e contas. Cada cliente tem os seguintes dados: nome, idade, endereço, número de suas contas (15 no máximo) e CGC. As contas válidas têm número diferente de 0. Cada conta possui um só cliente. As informações das contas são as seguintes: número da conta, cliente e saldo atual. (Se existem 12 clientes com quatro contas no máximo, então devem existir 48 contas).

- a) cadastre os clientes e suas contas;
- b) mostre todas as contas cadastradas;
- c) mostre todas as contas de um determinado cliente (identificadas pelo código);
- d) mostre o somatório das contas de um determinado cliente;
- e) mostre todas as contas com saldo negativo;
- f) mostre o ativo bancário (soma de todos os saldos).

**18.** Uma loja de eletrodomésticos está fazendo uma promoção entre seus 15.000 clientes. Todos os clientes que gastarem mais de R\$ 5.000,00 em compras passarão a ser considerados como clientes VIP, tendo 15% de desconto em todas as suas compras posteriores. Esse valor é cumulativo, mas precisa atingir R\$ 5.000,00 dentro de seis meses a partir da primeira compra ou será zerado. Faça um programa que:

- a) cadastre os clientes dessa loja. Para cada cliente devem ser cadastrados: nome do cliente, CPF, RG, endereço, data da primeira compra, o total gasto desde sua primeira compra e um campo que diz se o cliente é VIP ou não. O campo que guarda o total gasto pelo cliente deve sempre iniciar como zero e o campo que diz se o cliente é VIP deve começar como FALSO;
- b) atualize o total gasto por um determinado cliente. Deve-se ler um RG e, caso o RG seja encontrado na lista de clientes, deve-se entrar com um novo valor que atualizará o campo total gasto por esse cliente.  
Depois de entrar com o novo total gasto, deve-se fazer um teste para ver se o valor chegou a R\$ 5.000,00. Em caso positivo, o cliente passará a ser VIP. Esse teste só será realizado caso o cliente ainda não seja VIP.
- c) teste se o total gasto de cada cliente não-VIP deve ser zerado. Se o tempo entre a data da primeira compra de um cliente e a data atual exceder seis meses, o total gasto por esse cliente deve ser zerado. Lembre-se que isso só vale para clientes não-VIP.

**19.** Uma empresa de eletrodomésticos está realizando um sorteio de uma Ferrari F-50, do qual estão participando todos os que comprarem pelo menos cinco produtos de uma vez só, nas lojas autorizadas. Faça um programa que:

- a) leia os dados dos clientes como nome, data de nascimento, CPF, RG, cidade em que mora, endereço e a quantidade de eletrodomésticos adquiridos por esse cliente. Deve-se incluir um campo para o número do registro, que vai de 1 até 9999.
- b) faça o sorteio entre os clientes participantes. Somente os clientes que compraram mais de cinco equipamentos devem participar. Será sorteado um número e o cliente que tiver o registro com esse número será o ganhador. O cliente sorteado só será o ganhador caso tenha comprado pelo menos cinco equipamentos.

**20.** Faça um programa contendo os serviços que uma oficina mecânica pode realizar:  
Ordem de serviço (número da OS, data, valor, serviço realizado, cliente).

Leia as informações sobre várias ordens de serviço e determine, ao final, a média dos valores, o nome do cliente que realizou o serviço mais caro, juntamente com a descrição desse serviço e a data de sua realização.

## ARQUIVOS

### 9.1 DEFINIÇÃO DE ARQUIVOS EM ALGORIMTO

Estruturas de dados manipuladas fora do ambiente do programa são conhecidas como *arquivos*. Considera-se como ambiente do programa a memória principal, onde nem sempre é possível ou conveniente manter certas estruturas de dados.

Um arquivo, que é armazenado em um dispositivo de memória secundária, como discos, por exemplo, pode ser lido ou escrito por um programa.

Um arquivo é formado por uma coleção de registros, cada registro é composto por campos e cada campo possui suas características específicas. Um ou mais campos desse registro é considerado , que é o campo que diferencia um registro dos demais, evitando duplicidade de informações.

Um sistema de banco de dados é composto por um ou vários arquivos, onde cada arquivo possui programas de manutenção que são: *inclusão, exclusão lógica ou exclusão física, alteração, consulta geral, consulta específica e relatórios*.

Existem dois tipos de exclusão de registros: a *exclusão física*, em que após a eliminação de um registro, os demais registros são deslocados, e a *exclusão lógica*, em que os registros possuem um campo adicional, identificando se os mesmos estão ativos ou inativos, isto é, se foram excluídos. Nos exemplos apresentados neste capítulo, convencionou-se que se o registro possuir campo ATIVO com valor 0 significa que foi excluído.

### 9.2 DECLARAÇÃO DE ARQUIVOS EM PASCAL

```
TYPE nome do registro = RECORD
  nome do campo: tipo do dado;
  nome do campo : tipo do dado;
END;
Nome do arquivo = FILE OF nome do registro;
VAR  variável do arquivo : nome do arquivo;
     variável do registro : nome do registro;
```

A seguir temos a declaração de dois arquivos, primeiro o arquivo AGENDA que é composto pelos campos nome, endereço e telefone, depois o arquivo DETRAN que é composto pelos campos placa, marca, ano.

Exemplo do arquivo AGENDA:

```
TYPE registro = RECORD
  Nome : string[30];
  Endereco : string[20];
  Telefone : string[10];
END;
Arquivo = FILE OF registro;
VAR    AGENDA : arquivo;
REG : registro;
```

Exemplo do arquivo DETRAN:

```
TYPE carro = RECORD
  placa : string[7];
  marca : string[20];
  ano : integer;
END;
frota = FILE OF carro;
VAR    DETRAN : frota;
CARROS : carro;
```

## 9.3 COMANDOS DE ARQUIVOS EM PASCAL

### 9.3.1 COMANDO ASSIGN

Esse comando é utilizado para associar nomes de arquivos físicos a variáveis locais, isto é, variáveis do tipo arquivo com o arquivo do dispositivo de memória secundária, por exemplo, disco.

Exemplo do comando ASSIGN:

```
ASSIGN(nome do arquivo no programa, 'caminho do arquivo no
disco:\nome do arquivo no disco');
ASSIGN(AGENDA, 'AGENDA.DAT');
ASSIGN(DETRAN, 'CARROS.DAT');
```

#### OBSERVAÇÃO:

Se nenhum caminho for especificado, o PASCAL procura o arquivo no caminho default.

### 9.3.2 COMANDO REWRITE

Esse comando é utilizado para abrir novos arquivos, pois antes de posicionar o ponteiro no registro de número zero o comando REWRITE apagará todo o conteúdo do arquivo.

Exemplo do comando REWRITE:

```
REWRITE(nome do arquivo no programa);
REWRITE(AGENDA);
REWRITR(DETRAN);
```

### 9.3.3 COMANDO RESET

Esse comando é utilizado para abrir arquivos e posicionar o ponteiro no registro de número zero, sem destruir os dados já existentes no arquivo.

Exemplo do comando RESET:

```
RESET(nome do arquivo no programa);
RESET(AGENDA);
RESET(DETRAN);
```

### 9.3.4 COMANDO CLOSE

Esse comando é utilizado para fechar arquivos que foram abertos pelo comando REWRITE ou pelo comando RESET. As atualizações só serão efetuadas em um arquivo assim que ele for fechado.

Exemplo do comando CLOSE:

```
CLOSE(nome do arquivo no programa);
CLOSE(AGENDA);
CLOSE(DETRAN);
```

### 9.3.5 COMANDO READ

Esse comando é utilizado para ler os dados que estão armazenados nos registros de um arquivo.

Exemplo do comando READ:

```
READ(variável do arquivo, variável do registro);
READ(AGENDA, REG);
READ(DETRAN, CARROS);
```

### 9.3.6 COMANDO WRITE

Esse comando é utilizado para gravar dados nos registros de um arquivo.

Exemplo do comando WRITE:

```
WRITE(variável do arquivo, variável do registro);
WRITE(AGENDA, REG);
WRITE(DETRAN, CARROS);
```

### 9.3.7 COMANDO SEEK

Esse comando é utilizado para posicionar o ponteiro no registro desejado. O primeiro registro do arquivo é sempre o de número zero.

Exemplo do comando SEEK:

```
SEEK(variável do arquivo, número do registro);
SEEK(AGENDA, 2);
SEEK(DETRAN, 0);
```

### 9.3.8 COMANDO FILESIZE

Esse comando é utilizado para retornar o número de registros presentes em um arquivo.

Exemplo do comando FILESIZE:

```
FILESIZE(variável do arquivo);
FILESIZE(AGENDA);
FILESIZE(DETRAN);
```

### 9.3.9 COMANDO FILEPOS

Esse comando é utilizado para retornar o número do registro onde o ponteiro está localizado.

Exemplo do comando FILEPOS:

```
FILEPOS(variável do arquivo);
FILEPOS(AGENDA);
FILEPOS(DETRAN);
```

### 9.3.10 COMANDO NOT EOF

Esse comando é utilizado para verificar o final do arquivo.

Exemplo do comando NOT EOF:

```
WHILE NOT EOF(variável do arquivo) DO
  BEGIN
    comandos;
  END;
```

```

WHILE NOT EOF (DETRAN) DO
BEGIN
comandos;
END;

```

## 9.4 DECLARAÇÃO DE ARQUIVOS EM C/C++

Um arquivo em C/C++ pode representar diversas coisas, como arquivos em disco, uma impressora, um teclado, ou qualquer dispositivo de entrada ou saída. Neste capítulo estamos considerando apenas arquivos em disco. Entretanto, ressaltamos que, caso o leitor queira manipular outros dispositivos, a interface é a mesma.

A linguagem C/C++ dá suporte à utilização de arquivos por meio da biblioteca `stdlib.h`. Essa biblioteca fornece várias funções para manipulação de arquivos, define novos tipos de dados para serem usados especificamente com arquivos e, ainda, oferece algumas macros.

Os tipos de dados definidos na biblioteca `stdlib.h` são: `size_t`, `fpos_t` (ambos assumem os mesmos valores que o tipo `unsigned`) e `FILE`. Uma variável quando é do tipo ponteiro para `FILE` é capaz de identificar um arquivo no disco, direcionando-lhe todas as operações (você poderá obter maiores detalhes sobre ponteiros no próximo capítulo; por enquanto, basta saber que um ponteiro armazena um endereço de memória). Essas variáveis são declaradas como qualquer outro tipo de ponteiro:

```
FILE *arq, *pont;
```

## 9.5 COMANDOS DE ARQUIVOS EM C/C++

### 9.5.1 FUNÇÃO `fopen()`

Essa função abre um arquivo, retornando o ponteiro associado a esse arquivo. O protótipo da função `fopen()` é:

```
FILE *fopen(nome_do_arquivo, modo_de_abertura);
```

onde:

**nome\_do\_arquivo** representa o nome do arquivo que se deseja abrir, podendo conter, inclusive, o caminho (*path*) da pesquisa;

**modo\_de\_abertura** representa como o arquivo será aberto (a tabela a seguir descreve todos os modos).

- r Abre um arquivo de texto onde poderão ser realizadas apenas leituras.
- w Cria um arquivo de texto onde poderão ser realizadas apenas operações de escrita.
- a Anexa novos dados a um arquivo de texto.
- rb Abre um arquivo binário onde poderão ser realizadas apenas leituras.
- wb Cria um arquivo binário onde poderão ser realizadas apenas operações de escrita.
- ab Anexa novos dados a um arquivo binário.
- r+ Abre um arquivo de texto onde poderão ser realizadas operações de leitura e de escrita.
- w+ Cria um arquivo de texto onde poderão ser realizadas operações de escrita e de leitura.
- a+ Anexa novos dados ou cria um arquivo de texto para operações de leitura e de escrita.
- rb+ Abre um arquivo binário onde poderão ser realizadas operações de leitura e de escrita.
- wb+ Cria um arquivo binário onde poderão ser realizadas operações de escrita e de leitura.
- ab+ Anexa novos dados a um arquivo binário para operações de leitura e de escrita.

### Exemplo 1:

```
FILE *arq;
arq = fopen("arquivo1.dat", "w");
```

No Exemplo 1, a função `fopen()` cria um arquivo chamado `arquivo1`, onde poderão ser realizadas operações de escrita (gravação). Se a função `fopen()` for executada sem problemas, a variável `arq` receberá o endereço de memória ocupado pelo arquivo. Caso algum erro ocorra, a variável `arq` receberá o valor `NULL`. Sendo assim, é recomendável a realização de um teste para verificar se o arquivo foi aberto adequadamente. Observe o Exemplo 2.

### Exemplo 2:

```
FILE *arq;
arq = fopen("arquivo1.dat", "w");
if (arq == NULL)
    { cout << "\nOcorreu um erro. O arquivo não foi aberto."
      getch();
    }
else
    { /* As demais instruções do programa só poderão ser
       executadas se o arquivo foi aberto de forma correta.
    */
    }
```

**OBSERVAÇÕES:** Quando a função `fopen()` é utilizada para abrir um arquivo no modo escrita (`w` e `wb`), duas situações podem ser encontradas:

1. se o arquivo não existir, o mesmo será criado;
2. se o arquivo já existir, o mesmo será sobreposto por um novo arquivo vazio.

Quando a função `fopen()` é utilizada para abrir um arquivo já existente, deverá ser utilizado o modo para leitura ou para leitura/escrita. Dessa maneira, caso o arquivo já exista, seu conteúdo será preservado.

### 9.5.2 FUNÇÃO `fclose()`

Essa função fecha um arquivo. Quando ocorrer algum erro durante a execução dessa, poderá haver perda de dados ou, até mesmo, a perda do arquivo. O protótipo da função `fclose()` é:

```
int fclose(FILE *arq);
```

onde:

**arq** é o ponteiro para um arquivo (**arq** é o ponteiro obtido quando o arquivo foi aberto com a função `fopen()`).

Quando a função `fclose()` é executada ela gera como resultado um número inteiro. Quando o resultado for igual a 0 (zero) significa que o arquivo foi fechado corretamente. Quando o resultado for qualquer valor diferente de 0 (zero) significa que houve erro na operação.

### 9.5.3 FUNÇÃO `ferror()`

A função `ferror()` detecta se ocorreu algum erro durante uma operação com arquivos. O protótipo dessa função é:

```
int ferror(FILE *arq);
```

A função `ferror()` retorna um número inteiro. Se esse número for diferente de zero significa que ocorreu um erro durante a última operação realizada com o arquivo apontado por `arq`. Se esse número for 0 (zero) não ocorreu erro.

### 9.5.4 FUNÇÃO `fputc()`

Essa função escreve um caractere em um arquivo. O protótipo da função `fputc()` é:

```
int fputc(char ch, FILE *arq);
```

onde:

**ch** é o caractere que será escrito;

**arq** representa o endereço do arquivo onde o caractere será escrito.

Se a execução da função `fputc()` for bem-sucedida, vai gerar como retorno o valor do caractere escrito; caso contrário, devolverá o valor EOF.

### 9.5.5 FUNÇÃO `fgetc()`

Essa função lê um caractere em um arquivo. O protótipo da função `fgetc()` é:

```
int fgetc(FILE *arq);
```

onde:

**arq** representa o endereço do arquivo de onde o caractere será lido.

Se a execução da função `fgetc()` for bem-sucedida, vai gerar como retorno o valor do caractere lido; caso contrário, devolverá o valor EOF.

### 9.5.6 FUNÇÃO `fputs()`

Essa função escreve uma cadeia de caracteres em um arquivo. O protótipo da função `fputs()` é:

```
char *fputs(char *cadeia, FILE *arq);
```

onde:

**cadeia** armazena a cadeia de caracteres que será escrita no arquivo.

**arq** é o endereço do arquivo onde a cadeia de caracteres será escrita.

### 9.5.7 FUNÇÃO `fgets()`

Essa função lê uma cadeia de caracteres armazenada (até que seja encontrado um caractere de nova linha ou até que **tam-1** caracteres já tenham sido lidos) em um arquivo. O protótipo dessa função é:

```
char *fgets(char *cadeia, int tam, FILE *arq);
```

onde:

**cadeia** armazena a cadeia de caracteres obtida do arquivo.

**tam** é o tamanho da cadeia.

**arq** é o endereço do arquivo.

### 9.5.8 FUNÇÃO `fwrite()`

A função `fwrite()` pode escrever qualquer tipo dado e não apenas caracteres ou cadeias de caracteres. O protótipo da função `fwrite()` é:

```
size_t fwrite(void *mem, size_t qtd_bytes, size_t cont, FILE *arq);
```

onde:

**mem** representa a variável que armazena o conteúdo para ser gravado no arquivo.

**qtd\_bytes** representa o total em bytes que será escrito no arquivo.

**cont** representa o número de espaços de memória do tamanho especificado por **qtd\_bytes** que serão escritos no arquivo.

**arq** é o endereço do arquivo onde as informações serão escritas.

Quando a função `fwrite()` for bem-sucedida, vai gerar como retorno um valor igual ao número de gravações realizadas (o parâmetro **cont** descrito anteriormente). Caso contrário, quando ocorrer algum erro, o valor retornado será menor que **cont**.

Exemplo:

```
fwrite(&var, sizeof(double), 1, arq);
```

### 9.5.9 FUNÇÃO `fread()`

A função `fread()` pode ler qualquer tipo dado e não apenas caracteres ou cadeias de caracteres. O protótipo da função `fread()` é:

```
size_t fread(void *mem, size_t qtd_bytes, size_t cont, FILE *arq);
```

onde:

**mem** representa a variável que vai receber o conteúdo lido do arquivo.

**qtd\_bytes** representa o total em bytes que será lido do arquivo.

**cont** representa o número de espaços de memória do tamanho especificado por **qtd\_bytes** que serão lidos.

**arq** é o endereço do arquivo que será lido.

Quando a função `fread()` for bem-sucedida, vai gerar como retorno um valor igual ao número de leituras realizadas (o parâmetro **cont** descrito anteriormente). Caso contrário, quando ocorrer algum erro ou quando o final do arquivo for encontrado, o valor retornado será menor que **cont**.

Exemplo:

```
fread(&var, sizeof(double), 1, arq);
```

### 9.5.10 FUNÇÃO `fseek()`

Posiciona o cursor (ponteiro) em um endereço específico, tornando possível leituras e escritas aleatórias. O protótipo da função `fseek()` é:

```
int fseek(FILE *arq, long qtd_bytes, int posicao);
```

onde:

**arq** representa o arquivo que será percorrido pela função `fseek`.

**qtd\_bytes** representa a quantidade de bytes que será percorrida a partir de **posicao** para encontrar a informação desejada dentro do arquivo.

**posicao** representa o ponto a partir do qual a busca será executada – a variável **posicao** poderá receber três valores (SEEK\_SET, SEEK\_CUR, SEEK\_END).

SEEK\_SET – permite a movimentação de **qtd\_bytes** a partir da posição inicial do arquivo.

SEEK\_CUR – permite a movimentação de **qtd\_bytes** a partir da posição atual do arquivo.

SEEK\_END – permite a movimentação de **qtd\_bytes** a partir da posição final do arquivo.

### 9.5.11 FUNÇÃO `feof()`

Essa função determina se o final do arquivo foi encontrado. A função `feof()` retorna um número inteiro. Quando esse número for 0 (zero) significa que o fim do arquivo ainda não foi atingido. Qualquer outro valor significa que o fim do arquivo foi encontrado.

### 9.5.12 FUNÇÃO `rewind()`

Posiciona o cursor (indicador da posição atual) de volta ao início do arquivo. O protótipo da função `rewind()` é:

```
void rewind(FILE *arq);
```

onde:

`arq` representa o arquivo que terá o cursor reposicionado.

### 9.5.13 FUNÇÃO `remove()`

Essa função apaga um arquivo. O protótipo da função `remove()` é:

```
int remove(char *nome_arq);
```

onde:

`nome_arq` indica o nome do arquivo que será removido (podendo ser incluído o path).

Se a função `remove()` for executada com êxito, será devolvido o número 0 (zero). Caso contrário, será devolvido qualquer outro valor.

### 9.5.14 FUNÇÃO `fflush()`

A função `fflush()` escreve o conteúdo armazenado no buffer dentro de um determinado arquivo. O protótipo da função `fflush()` é:

```
inf fflush(FILE *arq);
```

onde:

`arq` especifica o endereço do arquivo onde serão efetivadas as gravações.

Caso a função `fflush()` não indique um arquivo especificamente, as gravações associadas a todos os arquivos abertos no momento serão efetivadas.

## EXERCÍCIOS RESOLVIDOS

1. Faça um programa para criar um arquivo chamado `ALUNOS.DAT`, onde cada registro será composto pelos seguintes campos: `numero`, `nome`, `curso`, `nota1`, `nota2`.

### ALGORITMO

#### SOLUÇÃO:

- Abrir o arquivo
- Fechar o arquivo



1<sup>a</sup> SOLUÇÃO - ARQUIVO QUE TRABALHARÁ COM EXCLUSÃO FÍSICA,  
PORTANTO, SEM O CAMPO ATIVO:

\EXERC\CAP9\PASCAL\EX1\_A.PAS e \EXERC\CAP9\PASCAL\EX1\_A.EXE

**2<sup>a</sup> SOLUÇÃO - ARQUIVO QUE TRABALHARÁ COM EXCLUSÃO LÓGICA,  
PORTANTO, COM O CAMPO ATIVO:**

\EXERC\CAP9\PASCAL\EX1\_B.PAS e \EXERC\CAP9\PASCAL\EX1\_B.EXE

**SOLUÇÃO:**

\EXERC\CAP9\C++\EX1.CPP e \EXERC\CAP9\C++\EX1.EXE

**2.** Faça um programa para incluir alunos no arquivo criado no Exercício 1, lembrando que não podem existir dois alunos com o mesmo número.

**ALGORITMO**

**SOLUÇÃO:**

Os passos para a inclusão seqüencial de registros em um arquivo são:

- Abrir o arquivo que sofrerá inclusões
- Digitar os dados que serão incluídos, fazer a validação dos dados
- Se o arquivo estiver vazio então gravar dados digitados no arquivo
- Se o arquivo não estiver vazio, então percorrer todo o arquivo do ↵ começo ao fim ou percorrer todo o arquivo até encontrar o ↵ campo-chave igual ao que se deseja incluir
- Se encontrar o campo-chave igual ao que se deseja incluir, então ↵ mensagem
- Se não encontrar o campo-chave igual ao que se deseja incluir, ↵ então gravar dados digitados no arquivo
- Fechar o arquivo

Os passos para a inclusão ordenada de registros em um arquivo são:

- Abrir o arquivo que sofrerá inclusões
- Digitar os dados que serão incluídos, fazer a validação dos dados
- Se o arquivo estiver vazio, então gravar dados digitados no arquivo
- Se o arquivo não estiver vazio, então percorrer todo o arquivo do ↵ começo ao fim ou percorrer todo o arquivo até encontrar o ↵ campo-chave igual ao que se deseja incluir
- Se encontrar o campo-chave que se está querendo incluir, então ↵ mensagem
- Se não encontrar o campo-chave igual ao que se deseja incluir, ↵ então ocorrerá o deslocamento de registros para gravar os dados ↵ digitados no arquivo na posição ordenada
- Fechar o arquivo

**RESOLUÇÃO  
PASCAL**

**1<sup>a</sup> SOLUÇÃO - INCLUSÃO SEQÜENCIAL - SEM O CAMPO ATIVO:**

\EXERC\CAP9\PASCAL\EX2\_A.PAS e \EXERC\CAP9\PASCAL\EX2\_A.EXE

**2<sup>a</sup> SOLUÇÃO - INCLUSÃO SEQÜENCIAL - COM O CAMPO ATIVO:**

\EXERC\CAP9\PASCAL\EX2\_B.PAS e \EXERC\CAP9\PASCAL\EX2\_B.EXE

**3<sup>a</sup> SOLUÇÃO - INCLUSÃO ORDENADA - SEM O CAMPO ATIVO:**

\EXERC\CAP9\PASCAL\EX2\_C.PAS e \EXERC\CAP9\PASCAL\EX2\_C.EXE

**4<sup>a</sup> SOLUÇÃO - INCLUSÃO ORDENADA - COM O CAMPO ATIVO:**

\EXERC\CAP9\PASCAL\EX2\_D.PAS e \EXERC\CAP9\PASCAL\EX2\_D.EXE


**1<sup>A</sup> SOLUÇÃO - INCLUSÃO SEQÜENCIAL - SEM O CAMPO ATIVO:**

\EXERC\CAP9\C++\EX2\_A.CPP e \EXERC\CAP9\C++\EX2\_A.EXE

**2<sup>A</sup> SOLUÇÃO - INCLUSÃO SEQÜENCIAL - COM O CAMPO ATIVO:**

\EXERC\CAP9\C++\EX2\_B.CPP e \EXERC\CAP9\C++\EX2\_B.EXE

**3<sup>A</sup> SOLUÇÃO - INCLUSÃO ORDENADA - SEM O CAMPO ATIVO:**

\EXERC\CAP9\C++\EX2\_C.CPP e \EXERC\CAP9\C++\EX2\_C.EXE

**4<sup>A</sup> SOLUÇÃO - INCLUSÃO ORDENADA - COM O CAMPO ATIVO:**

\EXERC\CAP9\C++\EX2\_D.CPP e \EXERC\CAP9\C++\EX2\_D.EXE

3. Faça um programa para alterar as notas dos alunos do arquivo criado no Exercício 1.


**SOLUÇÃO:**

- Abrir o arquivo que sofrerá alterações
- Digitar o campo-chave do registro que sofrerá alterações
- Se o arquivo estiver vazio, então mensagem
- Se o arquivo não estiver vazio, então percorrer todo o arquivo do começo ao fim ou percorrer todo o arquivo até encontrar o campo-chave que possui os dados que se está querendo alterar
- Se não encontrar o campo-chave que se está querendo alterar os dados, então mensagem
- Se encontrar o campo-chave que se está querendo alterar os dados, então mostrar os dados do registro que sofrerá alterações, digitar e validar os novos dados, posicionar no registro que sofrerá alterações e gravar
- Fechar o arquivo


**1<sup>A</sup> SOLUÇÃO - SEM O CAMPO ATIVO:**

\EXERC\CAP9\PASCAL\EX3\_A.PAS e \EXERC\CAP9\PASCAL\EX3\_A.EXE

**2<sup>A</sup> SOLUÇÃO - COM O CAMPO ATIVO:**

\EXERC\CAP9\PASCAL\EX3\_B.PAS e \EXERC\CAP9\PASCAL\EX3\_B.EXE


**1<sup>A</sup> SOLUÇÃO - SEM O CAMPO ATIVO:**

\EXERC\CAP9\C++\EX3\_A.CPP e \EXERC\CAP9\C++\EX3\_A.EXE

**2<sup>A</sup> SOLUÇÃO - COM O CAMPO ATIVO:**

\EXERC\CAP9\C++\EX3\_B.CPP e \EXERC\CAP9\C++\EX3\_B.EXE

4. Faça um programa para alterar o curso dos alunos do arquivo criado no Exercício 1.


**SOLUÇÃO:**

- Abrir o arquivo que sofrerá alterações
- Digitar o campo-chave do registro que sofrerá alterações

- Se o arquivo estiver vazio, então mensagem
- Se o arquivo não estiver vazio, então percorrer todo o arquivo do **█** começo ao fim ou percorrer todo o arquivo até encontrar o **█** campo-chave que se está querendo alterar
- Se não encontrar o campo-chave que contém os dados que se está **█** querendo alterar, então mensagem
- Se encontrar o campo-chave que contêm os dados que se está **█** querendo alterar então mostrar os dados do registro que sofrerá **█** alterações,
- **█** digitar e validar os novos dados, posicionar no registro que **█** sofrerá alterações e gravar
- Fechar o arquivo



#### 1<sup>a</sup> SOLUÇÃO - SEM O CAMPO ATIVO:

\EXERC\CAP9\PASCAL\EX4\_A.PAS e \EXERC\CAP9\PASCAL\EX4\_A.EXE

#### 2<sup>a</sup> SOLUÇÃO - COM O CAMPO ATIVO:

\EXERC\CAP9\PASCAL\EX4\_B.PAS e \EXERC\CAP9\PASCAL\EX4\_B.EXE



#### 1<sup>a</sup> SOLUÇÃO - SEM O CAMPO ATIVO:

\EXERC\CAP9\C++\EX4\_A.CPP e \EXERC\CAP9\C++\EX4\_A.EXE

#### 2<sup>a</sup> SOLUÇÃO - COM O CAMPO ATIVO:

\EXERC\CAP9\C++\EX4\_B.CPP e \EXERC\CAP9\C++\EX4\_B.EXE

### 5. Faça um programa para excluir os alunos do arquivo criado no Exercício 1.



#### SOLUÇÃO:

Conforme apresentado anteriormente, existem dois tipos de exclusão:  
**█** física e lógica.

Os passos para a exclusão física de registros em um arquivo são:

- Abrir o arquivo que sofrerá exclusão
- Digitar o campo-chave do registro que será excluído
- Se o arquivo estiver vazio, então mensagem
- Se o arquivo não estiver vazio, então percorrer todo o arquivo do **█** começo ao fim ou percorrer todo o arquivo até encontrar o **█** campo-chave que se está querendo excluir
- Se não encontrar o campo chave que se está querendo excluir, então **█** mensagem
- Se encontrar o campo-chave que se está querendo excluir, então **█** ocorrerá o deslocamento de registros para sobrepor o registro que **█** será excluído
- Fechar o arquivo

Os passos para a exclusão lógica de registros em um arquivo são:

- Abrir o arquivo que sofrerá exclusão
- Digitar o campo-chave do registro que será excluído
- Se o arquivo estiver vazio, então mensagem
- Se o arquivo não estiver vazio então percorrer todo o arquivo do **█** começo ao fim ou percorrer todo o arquivo até encontrar o **█** campo-chave que se está querendo excluir
- Se não encontrar o campo-chave que se está querendo excluir, então **█** mensagem

- Se encontrar o campo-chave que se está querendo excluir, então o campo de marcação (campo ativo) do registro será marcado para exclusão
- Fechar o arquivo


**1<sup>a</sup> SOLUÇÃO - EXCLUSÃO FÍSICA - SEM O CAMPO ATIVO:**

\EXERC\CAP9\PASCAL\EX5\_A.PAS e \EXERC\CAP9\PASCAL\EX5\_A.EXE

**2<sup>a</sup> SOLUÇÃO - EXCLUSÃO LÓGICA - COM O CAMPO ATIVO:**

\EXERC\CAP9\PASCAL\EX5\_B.PAS e \EXERC\CAP9\PASCAL\EX5\_B.EXE


**1<sup>a</sup> SOLUÇÃO - EXCLUSÃO FÍSICA - SEM O CAMPO ATIVO:**

\EXERC\CAP9\C++\EX5\_A.CPP e \EXERC\CAP9\C++\EX5\_A.EXE

**2<sup>a</sup> SOLUÇÃO - EXCLUSÃO LÓGICA - COM O CAMPO ATIVO:**

\EXERC\CAP9\C++\EX5\_B.CPP e \EXERC\CAP9\C++\EX5\_B.EXE

6. Faça um programa para consultar o número, o nome e a média de todos os alunos cadastrados no arquivo do Exercício 1.

**SOLUÇÃO:**

Os passos para a consulta geral de registros em um arquivo são:

- Abrir o arquivo que sofrerá a consulta
- Se o arquivo estiver vazio, então mensagem
- Se o arquivo não estiver vazio, então percorrer todo o arquivo do começo ao fim mostrando todos os campos solicitados de cada um dos registros e calculando o necessário, neste caso, a média
- Fechar o arquivo


**1<sup>a</sup> SOLUÇÃO - SEM O CAMPO ATIVO:**

\EXERC\CAP9\PASCAL\EX6\_A.PAS e \EXERC\CAP9\PASCAL\EX6\_A.EXE

**2<sup>a</sup> SOLUÇÃO - COM O CAMPO ATIVO:**

\EXERC\CAP9\PASCAL\EX6\_B.PAS e \EXERC\CAP9\PASCAL\EX6\_B.EXE


**1<sup>a</sup> SOLUÇÃO - SEM O CAMPO ATIVO:**

\EXERC\CAP9\C++\EX6\_A.CPP e \EXERC\CAP9\C++\EX6\_A.EXE

**2<sup>a</sup> SOLUÇÃO - COM O CAMPO ATIVO:**

\EXERC\CAP9\C++\EX6\_B.CPP e \EXERC\CAP9\C++\EX6\_B.EXE

7. Faça um programa para consultar o número, o nome e a média de todos os alunos cadastrados no arquivo do Exercício 1 e que estejam aprovados, ou seja, com média no mínimo 7,0.

**SOLUÇÃO:**

Os passos para a consulta de registros em um arquivo são:

- Abrir o arquivo que sofrerá a consulta
- Se o arquivo estiver vazio, então mensagem
- Se o arquivo não estiver vazio, então percorrer todo o arquivo do  $\blacktriangleright$  começo ao fim calculando a média e, se esta for maior ou igual a 7  $\blacktriangleright$  (sete), mostrando todos os campos solicitados
- Fechar o arquivo

**1<sup>a</sup> SOLUÇÃO - SEM O CAMPO ATIVO:**

```
\EXERC\CAP9\PASCAL\EX7_A.PAS e \EXERC\CAP9\PASCAL\EX7_A.EXE
```

**2<sup>a</sup> SOLUÇÃO - COM O CAMPO ATIVO:**

```
\EXERC\CAP9\PASCAL\EX7_B.PAS e \EXERC\CAP9\PASCAL\EX7_B.EXE
```

**1<sup>a</sup> SOLUÇÃO - SEM O CAMPO ATIVO:**

```
\EXERC\CAP9\C++\EX7_A.CPP e \EXERC\CAP9\C++\EX7_A.EXE
```

**2<sup>a</sup> SOLUÇÃO - COM O CAMPO ATIVO:**

```
\EXERC\CAP9\C++\EX7_B.CPP e \EXERC\CAP9\C++\EX7_B.EXE
```

**8.** Faça um programa para consultar o número, o nome e o curso de todos os alunos cadastrados no arquivo do Exercício 1 e que estejam reprovados, ou seja, com média inferior a 3,0.

**SOLUÇÃO:**

Os passos para a consulta de registros em um arquivo são:

- Abrir o arquivo que sofrerá a consulta
- Se o arquivo estiver vazio, então mensagem
- Se o arquivo não estiver vazio, então percorrer todo o arquivo do  $\blacktriangleright$  começo ao fim calculando a média e, se esta for menor que 3  $\blacktriangleright$  (três), mostrando todos os campos solicitados
- Fechar o arquivo

**1<sup>a</sup> SOLUÇÃO - SEM O CAMPO ATIVO:**

```
\EXERC\CAP9\PASCAL\EX8_A.PAS e \EXERC\CAP9\PASCAL\EX8_A.EXE
```

**2<sup>a</sup> SOLUÇÃO - COM O CAMPO ATIVO:**

```
\EXERC\CAP9\PASCAL\EX8_B.PAS e \EXERC\CAP9\PASCAL\EX8_B.EXE
```

**1<sup>a</sup> SOLUÇÃO - SEM O CAMPO ATIVO:**

```
\EXERC\CAP9\C++\EX8_A.CPP e \EXERC\CAP9\C++\EX8_A.EXE
```

**2<sup>a</sup> SOLUÇÃO - COM O CAMPO ATIVO:**

```
\EXERC\CAP9\C++\EX8_B.CPP e \EXERC\CAP9\C++\EX8_B.EXE
```

 9. Faça um programa para consultar o nome de todos os alunos cadastrados no arquivo do Exercício 1 e que estejam de exame, ou seja, com média entre 3,0 (inclusive) e 7,0.

 **SOLUÇÃO:**

Os passos para a consulta de registros em um arquivo são:

- Abrir o arquivo que sofrerá a consulta
- Se o arquivo estiver vazio, então mensagem
- Se o arquivo não estiver vazio, então percorrer todo o arquivo do **█** começo ao fim calculando a média e, se esta for maior ou igual a 3 **█** (três) e menor que 7 (sete), mostrando todos os campos solicitados
- Fechar o arquivo

 **1<sup>a</sup> SOLUÇÃO - SEM O CAMPO ATIVO:**

\EXERC\CAP9\PASCAL\EX9\_A.PAS e \EXERC\CAP9\PASCAL\EX9\_A.EXE

**2<sup>a</sup> SOLUÇÃO - COM O CAMPO ATIVO:**

\EXERC\CAP9\PASCAL\EX9\_B.PAS e \EXERC\CAP9\PASCAL\EX9\_B.EXE

 **1<sup>a</sup> SOLUÇÃO - SEM O CAMPO ATIVO:**

\EXERC\CAP9\C++\EX9\_A.CPP e \EXERC\CAP9\C++\EX9\_A.EXE

**2<sup>a</sup> SOLUÇÃO - COM O CAMPO ATIVO:**

\EXERC\CAP9\C++\EX9\_B.CPP e \EXERC\CAP9\C++\EX9\_B.EXE

 10. Faça um programa para consultar o nome de todos os alunos de um determinado curso.

 **SOLUÇÃO:**

Os passos para a consulta de registros em um arquivo são:

- Abrir o arquivo que sofrerá a consulta
- Se o arquivo estiver vazio, então mensagem
- Se o arquivo não estiver vazio, então percorrer todo o arquivo do **█** começo ao fim mostrando todos os campos solicitados de cada um dos **█** registros, neste caso conferindo se o campo curso é igual ao **█** curso digitado
- Fechar o arquivo

 **1<sup>a</sup> SOLUÇÃO - SEM O CAMPO ATIVO:**

\EXERC\CAP9\PASCAL\EX10\_A.PAS e \EXERC\CAP9\PASCAL\EX10\_A.EXE

**2<sup>a</sup> SOLUÇÃO - COM O CAMPO ATIVO:**

\EXERC\CAP9\PASCAL\EX10\_B.PAS e \EXERC\CAP9\PASCAL\EX10\_B.EXE

 **1<sup>a</sup> SOLUÇÃO - SEM O CAMPO ATIVO:**

\EXERC\CAP9\C++\EX10\_A.CPP e \EXERC\CAP9\C++\EX10\_A.EXE

**2<sup>a</sup> SOLUÇÃO - COM O CAMPO ATIVO:**

\EXERC\CAP9\C++\EX10\_B.CPP e \EXERC\CAP9\C++\EX10\_B.EXE

- 11.** Faça um programa para criar um arquivo chamado VENDAS.DAT, onde cada registro será composto pelos seguintes campos: `codigo_vendedor`, `nome_vendedor`, `valor_venda` e `mes`.

**ALGORITMO****SOLUÇÃO:**

- Abrir o arquivo
- Fechar o arquivo

**1<sup>a</sup> SOLUÇÃO - SEM O CAMPO ATIVO:**

\EXERC\CAP9\PASCAL\EX11\_A.PAS e \EXERC\CAP9\PASCAL\EX11\_A.EXE

**2<sup>a</sup> SOLUÇÃO - COM O CAMPO ATIVO:**

\EXERC\CAP9\PASCAL\EX11\_B.PAS e \EXERC\CAP9\PASCAL\EX11\_B.EXE

**SOLUÇÃO:**

\EXERC\CAP9\C++\EX11.CPP e \EXERC\CAP9\C++\EX11.EXE

- 12.** Faça um programa para incluir um vendedor no arquivo criado no Exercício 11, lembrando que não podem existir dois vendedores com o mesmo código e mesmo mês de vendas.

**ALGORITMO****SOLUÇÃO:**

Os passos para a inclusão seqüencial de registros em um arquivo são:

- Abrir o arquivo que sofrerá inclusões
- Digitar os dados que serão incluídos, fazer a validação dos dados
- Se o arquivo estiver vazio, então gravar os dados digitados no arquivo
- Se o arquivo não estiver vazio então percorrer todo o arquivo do começo ao fim ou percorrer todo o arquivo até encontrar um campo-chave igual ao que se está querendo incluir
- Se encontrar o campo chave igual ao que se está querendo incluir, então mensagem
- Se não encontrar o campo-chave igual ao que se está querendo incluir, então gravar os dados digitados no arquivo
- Fechar o arquivo

Os passos para a inclusão ordenada de registros em um arquivo são:

- Abrir o arquivo que sofrerá inclusões
- Digitar os dados que serão incluídos, fazer a validação dos dados
- Se o arquivo estiver vazio, então gravar dados digitados no arquivo
- Se o arquivo não estiver vazio então percorrer todo o arquivo do começo ao fim ou percorrer todo o arquivo até encontrar um campo-chave igual ao que se está querendo incluir
- Se encontrar o campo-chave igual ao que se está querendo incluir, então mensagem
- Se não encontrar o campo-chave igual ao que se está querendo incluir, então ocorrerá o deslocamento de registros para gravar os dados digitados no arquivo na posição ordenada
- Fechar o arquivo

**1<sup>A</sup> SOLUÇÃO - INCLUSÃO SEQÜENCIAL - SEM O CAMPO ATIVO:**

\EXERC\CAP9\PASCAL\EX12\_A.PAS e \EXERC\CAP9\PASCAL\EX12\_A.EXE

**2<sup>A</sup> SOLUÇÃO - INCLUSÃO SEQÜENCIAL - COM O CAMPO ATIVO:**

\EXERC\CAP9\PASCAL\EX12\_B.PAS e \EXERC\CAP9\PASCAL\EX12\_B.EXE

**3<sup>A</sup> SOLUÇÃO - INCLUSÃO ORDENADA - SEM O CAMPO ATIVO:**

\EXERC\CAP9\PASCAL\EX12\_C.PAS e \EXERC\CAP9\PASCAL\EX12\_C.EXE

**4<sup>A</sup> SOLUÇÃO - INCLUSÃO ORDENADA - COM O CAMPO ATIVO:**

\EXERC\CAP9\PASCAL\EX12\_D.PAS e \EXERC\CAP9\PASCAL\EX12\_D.EXE

**1<sup>A</sup> SOLUÇÃO - INCLUSÃO SEQÜENCIAL - SEM O CAMPO ATIVO:**

\EXERC\CAP9\C++\EX12\_A.CPP e \EXERC\CAP9\C++\EX12\_A.EXE

**2<sup>A</sup> SOLUÇÃO - INCLUSÃO SEQÜENCIAL - COM O CAMPO ATIVO:**

\EXERC\CAP9\C++\EX12\_B.CPP e \EXERC\CAP9\C++\EX12\_B.EXE

**3<sup>A</sup> SOLUÇÃO - INCLUSÃO ORDENADA - SEM O CAMPO ATIVO:**

\EXERC\CAP9\C++\EX12\_C.CPP e \EXERC\CAP9\C++\EX12\_C.EXE

**4<sup>A</sup> SOLUÇÃO - INCLUSÃO ORDENADA - COM O CAMPO ATIVO:**

\EXERC\CAP9\C++\EX12\_D.CPP e \EXERC\CAP9\C++\EX12\_D.EXE

**13.** Faça um programa para alterar o valor de uma venda no arquivo criado no Exercício 11.

**SOLUÇÃO:**

- Abrir o arquivo que sofrerá alterações
- Digitar o campo-chave do registro que sofrerá alterações
- Se o arquivo estiver vazio, então mensagem
- Se o arquivo não estiver vazio, então percorrer todo o arquivo do ↵ começo ao fim ou percorrer todo o arquivo até encontrar um ↵ campo chave igual ao que se está querendo alterar
- Se não encontrar o campo-chave igual ao que se está querendo ↵ alterar, então mensagem
- Se encontrar o campo-chave igual ao que se está querendo alterar, ↵ então mostrar os dados do registro que sofrerá alterações, digitar ↵ e validar os novos dados, posicionar no registro que sofrerá ↵ alterações e gravar
- Fechar o arquivo

**1<sup>A</sup> SOLUÇÃO - SEM O CAMPO ATIVO:**

\EXERC\CAP9\PASCAL\EX13\_A.PAS e \EXERC\CAP9\PASCAL\EX13\_A.EXE

**2<sup>A</sup> SOLUÇÃO - COM O CAMPO ATIVO:**

\EXERC\CAP9\PASCAL\EX13\_B.PAS e \EXERC\CAP9\PASCAL\EX13\_B.EXE

**1<sup>A</sup> SOLUÇÃO - SEM O CAMPO ATIVO:**

\EXERC\CAP9\C++\EX13\_A.CPP e \EXERC\CAP9\C++\EX13\_A.EXE

**2<sup>A</sup> SOLUÇÃO - COM O CAMPO ATIVO:**

\EXERC\CAP9\C++\EX13\_B.CPP e \EXERC\CAP9\C++\EX13\_B.EXE

**14.** Faça um programa para um excluir vendedor no arquivo criado no Exercício 11.

**SOLUÇÃO:**

Os passos para a exclusão física de registros em um arquivo são:

- Abrir o arquivo que sofrerá exclusão
- Digitar o campo-chave do registro que será excluído
- Se o arquivo estiver vazio, então mensagem
- Se o arquivo não estiver vazio, então percorrer todo o arquivo até começar ao fim ou percorrer todo o arquivo até encontrar o campo-chave do registro que se está querendo excluir
- Se não encontrar o campo-chave do registro que se está querendo excluir, então mensagem
- Se encontrar o campo-chave do registro que se está querendo excluir, então ocorrerá o deslocamento de registros para sobrepor o registro que será excluído
- Fechar o arquivo

Os passos para a exclusão lógica de registros em um arquivo são:

- Abrir o arquivo que sofrerá exclusão
- Digitar o campo-chave do registro que será excluído
- Se o arquivo estiver vazio, então mensagem
- Se o arquivo não estiver vazio, então percorrer todo o arquivo até começar ao fim ou percorrer todo o arquivo até encontrar o campo-chave do registro que se está querendo excluir
- Se não encontrar o campo-chave do registro que se está querendo excluir, então mensagem
- Se encontrar o campo-chave do registro que se está querendo excluir, então o campo de marcação (campo ativo) do registro será marcado para exclusão
- Fechar o arquivo

**1<sup>A</sup> SOLUÇÃO - EXCLUSÃO FÍSICA - SEM O CAMPO ATIVO:**

\EXERC\CAP9\PASCAL\EX14\_A.PAS e \EXERC\CAP9\PASCAL\EX14\_A.EXE

**2<sup>A</sup> SOLUÇÃO - EXCLUSÃO LÓGICA - COM O CAMPO ATIVO:**

\EXERC\CAP9\PASCAL\EX14\_B.PAS e \EXERC\CAP9\PASCAL\EX14\_B.EXE

**1<sup>A</sup> SOLUÇÃO - EXCLUSÃO FÍSICA - SEM O CAMPO ATIVO:**

\EXERC\CAP9\C++\EX14\_A.CPP e \EXERC\CAP9\C++\EX14\_A.EXE

**2<sup>A</sup> SOLUÇÃO - EXCLUSÃO LÓGICA - COM O CAMPO ATIVO:**

\EXERC\CAP9\C++\EX14\_B.CPP e \EXERC\CAP9\C++\EX14\_B.EXE

 **15.** Faça um programa para consultar o valor da venda de um vendedor em um determinado mês no arquivo criado no Exercício 11.



#### SOLUÇÃO:

Os passos para a consulta de registros em um arquivo são:

- Abrir o arquivo que sofrerá a consulta
- Se o arquivo estiver vazio, então mensagem
- Se o arquivo não estiver vazio, então percorrer todo o arquivo do  começo ao fim mostrando todos os campos solicitados de cada um dos  registros, neste caso conferindo se os campos código do vendedor  e mês são iguais aos valores digitados
- Fechar o arquivo



#### 1<sup>a</sup> SOLUÇÃO - SEM O CAMPO ATIVO:

\EXERC\CAP9\PASCAL\EX15\_A.PAS e \EXERC\CAP9\PASCAL\EX15\_A.EXE

#### 2<sup>a</sup> SOLUÇÃO - COM O CAMPO ATIVO:

\EXERC\CAP9\PASCAL\EX15\_B.PAS e \EXERC\CAP9\PASCAL\EX15\_B.EXE



#### 1<sup>a</sup> SOLUÇÃO - SEM O CAMPO ATIVO:

\EXERC\CAP9\C++\EX15\_A.CPP e \EXERC\CAP9\C++\EX15\_A.EXE

#### 2<sup>a</sup> SOLUÇÃO - COM O CAMPO ATIVO:

\EXERC\CAP9\C++\EX15\_B.CPP e \EXERC\CAP9\C++\EX15\_B.EXE

 **16.** Faça um programa para consultar o total das vendas de um determinado vendedor do arquivo criado no Exercício 11.



#### SOLUÇÃO:

Os passos para a consulta de registros em um arquivo são:

- Abrir o arquivo que sofrerá a consulta
- Se o arquivo estiver vazio, então mensagem
- Se o arquivo não estiver vazio, então percorrer todo o arquivo do  começo ao fim mostrando todos os campos solicitados de cada um dos  registros, neste caso conferindo se o campo código do vendedor é  igual ao código digitado, somando os valores das vendas para  finalmente mostrar o total de vendas
- Fechar o arquivo



#### 1<sup>a</sup> SOLUÇÃO - SEM O CAMPO ATIVO:

\EXERC\CAP9\PASCAL\EX16\_A.PAS e \EXERC\CAP9\PASCAL\EX16\_A.EXE

#### 2<sup>a</sup> SOLUÇÃO - COM O CAMPO ATIVO:

\EXERC\CAP9\PASCAL\EX16\_B.PAS e \EXERC\CAP9\PASCAL\EX16\_B.EXE



#### 1<sup>a</sup> SOLUÇÃO - SEM O CAMPO ATIVO:

\EXERC\CAP9\C++\EX16\_A.CPP e \EXERC\CAP9\C++\EX16\_A.EXE

**2<sup>A</sup> SOLUÇÃO - COM O CAMPO ATIVO:**

\EXERC\CAP9\C++\EX16\_B.CPP e \EXERC\CAP9\C++\EX16\_B.EXE

-  **17.** Faça um programa para consultar o nome e o código do vendedor que mais vendeu em um determinado mês no arquivo criado no Exercício 11.

**SOLUÇÃO:**

Os passos para a consulta de registros em um arquivo são:

- Abrir o arquivo que sofrerá a consulta
- Se o arquivo estiver vazio, então mensagem
- Se o arquivo não estiver vazio, então percorrer todo o arquivo do ↪ começo ao fim verificando se o mês da venda é igual ao digitado e, ↪ neste caso, verificando qual foi a maior venda (armazenando o nome ↪ e o código do vendedor em variáveis auxiliares)
- Quando chegar ao final do arquivo, mostrar o nome e o código ↪ armazenados nas variáveis auxiliares
- Fechar o arquivo

**1<sup>A</sup> SOLUÇÃO - SEM O CAMPO ATIVO:**

\EXERC\CAP9\PASCAL\EX17\_A.PAS e \EXERC\CAP9\PASCAL\EX17\_A.EXE

**2<sup>A</sup> SOLUÇÃO - COM O CAMPO ATIVO:**

\EXERC\CAP9\PASCAL\EX17\_B.PAS e \EXERC\CAP9\PASCAL\EX17\_B.EXE

**1<sup>A</sup> SOLUÇÃO - SEM O CAMPO ATIVO:**

\EXERC\CAP9\C++\EX17\_A.CPP e \EXERC\CAP9\C++\EX17\_A.EXE

**2<sup>A</sup> SOLUÇÃO - COM O CAMPO ATIVO:**

\EXERC\CAP9\C++\EX17\_B.CPP e \EXERC\CAP9\C++\EX17\_B.EXE

-  **18.** Faça um programa para consultar o mês com o maior valor de vendas e qual o nome do vendedor que efetuou essas vendas, no arquivo criado no Exercício 11.

**SOLUÇÃO:**

Os passos para a consulta de registros em um arquivo são:

- Abrir o arquivo que sofrerá a consulta
- Se o arquivo estiver vazio, então mensagem
- Se o arquivo não estiver vazio, então percorrer todo o arquivo do ↪ começo ao fim verificando qual a maior venda (armazenado em ↪ variáveis auxiliares a maior venda e o nome do vendedor ↪ responsável por ela)
- Quando chegar ao final do arquivo, mostrar os valores das ↪ variáveis auxiliares
- Fechar o arquivo

**1<sup>A</sup> SOLUÇÃO - SEM O CAMPO ATIVO:**

\EXERC\CAP9\PASCAL\EX18\_A.PAS e \EXERC\CAP9\PASCAL\EX18\_A.EXE



**2<sup>a</sup> SOLUÇÃO - COM O CAMPO ATIVO:**

\EXERC\CAP9\PASCAL\EX18\_B.PAS e \EXERC\CAP9\PASCAL\EX18\_B.EXE

**1<sup>a</sup> SOLUÇÃO - SEM O CAMPO ATIVO:**

\EXERC\CAP9\C++\EX18\_A.CPP e \EXERC\CAP9\C++\EX18\_A.EXE

**2<sup>a</sup> SOLUÇÃO - COM O CAMPO ATIVO:**

\EXERC\CAP9\C++\EX18\_B.CPP e \EXERC\CAP9\C++\EX18\_B.EXE

**19.** Faça um programa para criar os arquivos a seguir.**CLIENTE**

Número do cliente

Nome

**CONTA\_BANCÁRIA**

Número da conta

Número do cliente

Saldo

**SOLUÇÃO:**

- Abrir o arquivo (para o arquivo cliente)
- Fechar o arquivo (para o arquivo cliente)
- Abrir o arquivo (para o arquivo conta\_bancária)
- Fechar o arquivo (para o arquivo conta\_bancária)

**1<sup>a</sup> SOLUÇÃO - SEM O CAMPO ATIVO:**

\EXERC\CAP9\PASCAL\EX19\_A.PAS e \EXERC\CAP9\PASCAL\EX19\_A.EXE

**2<sup>a</sup> SOLUÇÃO - COM O CAMPO ATIVO:**

\EXERC\CAP9\PASCAL\EX19\_B.PAS e \EXERC\CAP9\PASCAL\EX19\_B.EXE

**SOLUÇÃO:**

\EXERC\CAP9\C++\EX19.CPP e \EXERC\CAP9\C++\EX19.EXE

**20.** Faça um programa que inclua clientes no arquivo criado no Exercício 19.**SOLUÇÃO:**

Os passos para a inclusão seqüencial de registros em um arquivo são:

- Abrir o arquivo que sofrerá inclusões
- Digitar os dados que serão incluídos, fazer a validação dos dados
- Se o arquivo estiver vazio, então gravar dados digitados no arquivo
- Se o arquivo não estiver vazio então percorrer todo o arquivo do **begin** ao **fim** ou percorrer todo o arquivo até encontrar o **campo-chave** igual ao que se está querendo incluir
- Se encontrar o campo-chave igual ao que se está querendo incluir, **então mensagem**
- Se não encontrar o campo-chave igual ao que se está querendo **incluir, então gravar dados digitados no arquivo**
- Fechar o arquivo

Os passos para a inclusão ordenada de registros em um arquivo são:

- Abrir o arquivo que sofrerá inclusões
- Digitar os dados que serão incluídos, fazer a validação dos dados
- Se o arquivo estiver vazio, então gravar dados digitados no arquivo
- Se o arquivo não estiver vazio, então percorrer todo o arquivo do **beginning** ao **end** ou percorrer todo o arquivo até encontrar o **key** campo-chave igual ao que se está querendo incluir
- Se encontrar o campo-chave igual ao que se está querendo incluir, **then** então mensagem
- Se não encontrar o campo-chave igual ao que se está querendo **then** incluir, então ocorrerá o deslocamento de registros para gravar os **new data** digitados no arquivo na posição ordenada
- Fechar o arquivo



#### **1<sup>a</sup> SOLUÇÃO - INCLUSÃO SEQÜENCIAL - SEM O CAMPO ATIVO:**

\EXERC\CAP9\PASCAL\EX20\_A.PAS e \EXERC\CAP9\PASCAL\EX20\_A.EXE

#### **2<sup>a</sup> SOLUÇÃO - INCLUSÃO SEQÜENCIAL - COM O CAMPO ATIVO:**

\EXERC\CAP9\PASCAL\EX20\_B.PAS e \EXERC\CAP9\PASCAL\EX20\_B.EXE



#### **1<sup>a</sup> SOLUÇÃO - INCLUSÃO SEQÜENCIAL - SEM O CAMPO ATIVO:**

\EXERC\CAP9\C++\EX20\_A.CPP e \EXERC\CAP9\C++\EX20\_A.EXE

#### **2<sup>a</sup> SOLUÇÃO - INCLUSÃO SEQÜENCIAL - COM O CAMPO ATIVO:**

\EXERC\CAP9\C++\EX20\_B.CPP e \EXERC\CAP9\C++\EX20\_B.EXE

**21.** Faça um programa que inclua contas para clientes já cadastrados no Exercício 20.



#### **SOLUÇÃO:**

Os passos para a inclusão seqüencial de registros em um arquivo são:

- Abrir o arquivo que sofrerá inclusões
- Digitar os dados que serão incluídos, fazer a validação dos dados (**verificando se o cliente informado foi previamente cadastrado no arquivo de clientes**)
- Se o arquivo estiver vazio, então gravar dados digitados no arquivo
- Se o arquivo não estiver vazio, então percorrer todo o arquivo do **beginning** ao **end** ou percorrer todo o arquivo até encontrar um **key** campo-chave igual ao que se está querendo incluir
- Se encontrar o campo-chave igual ao que se está querendo incluir, **then** então mensagem
- Se não encontrar o campo-chave igual ao que se está querendo **then** incluir, então gravar dados digitados no arquivo
- Fechar o arquivo

Os passos para a inclusão ordenada de registros em um arquivo são:

- Abrir o arquivo que sofrerá inclusões
- Digitar os dados que serão incluídos, fazer a validação dos dados (**verificando se o cliente informado foi previamente cadastrado no arquivo de clientes**)
- Se o arquivo estiver vazio, então gravar dados digitados no arquivo
- Se o arquivo não estiver vazio, então percorrer todo o arquivo do **beginning** ao **end** ou percorrer todo o arquivo até encontrar um **key** campo-chave igual ao que se está querendo incluir
- Se encontrar o campo-chave igual ao que se está querendo incluir, **then** então mensagem

- Se não encontrar o campo-chave igual ao que se está querendo
  - ↳ incluir, então ocorrerá o deslocamento de registros para gravar os dados digitados no arquivo na posição ordenada
  - Fechar o arquivo



**1<sup>a</sup> SOLUÇÃO - INCLUSÃO SEQÜENCIAL - SEM O CAMPO ATIVO:**

\EXERC\CAP9\PASCAL\EX21\_A.PAS e \EXERC\CAP9\PASCAL\EX21\_A.EXE

**2<sup>a</sup> SOLUÇÃO - INCLUSÃO SEQÜENCIAL - COM O CAMPO ATIVO:**

\EXERC\CAP9\PASCAL\EX21\_B.PAS e \EXERC\CAP9\PASCAL\EX21\_B.EXE



**1<sup>a</sup> SOLUÇÃO - INCLUSÃO SEQÜENCIAL - SEM O CAMPO ATIVO:**

\EXERC\CAP9\C++\EX21\_A.CPP e \EXERC\CAP9\C++\EX21\_A.EXE

**2<sup>a</sup> SOLUÇÃO - INCLUSÃO SEQÜENCIAL - COM O CAMPO ATIVO:**

\EXERC\CAP9\C++\EX21\_B.CPP e \EXERC\CAP9\C++\EX21\_B.EXE

22. Faça um programa para consultar o saldo de todas as contas de um determinado cliente cadastrado nos exercícios 20 e 21.



**SOLUÇÃO:**

- Abrir os arquivos que sofrerão consultas
- Digitar o código do cliente a ser consultado
- Se o arquivo de contas não estiver vazio, então percorrer todo o
  - ↳ arquivo do começo ao fim verificando se o código do cliente é
    - ↳ igual ao código informado e, neste caso, mostrando os campos
      - ↳ solicitados
  - Fechar os arquivos



**1<sup>a</sup> SOLUÇÃO - SEM O CAMPO ATIVO:**

\EXERC\CAP9\PASCAL\EX22\_A.PAS e \EXERC\CAP9\PASCAL\EX22\_A.EXE

**2<sup>a</sup> SOLUÇÃO - COM O CAMPO ATIVO:**

\EXERC\CAP9\PASCAL\EX22\_B.PAS e \EXERC\CAP9\PASCAL\EX22\_B.EXE



**1<sup>a</sup> SOLUÇÃO - INCLUSÃO SEQÜENCIAL - SEM O CAMPO ATIVO:**

\EXERC\CAP9\C++\EX22\_A.CPP e \EXERC\CAP9\C++\EX22\_A.EXE

**2<sup>a</sup> SOLUÇÃO - INCLUSÃO SEQÜENCIAL - COM O CAMPO ATIVO:**

\EXERC\CAP9\C++\EX22\_B.CPP e \EXERC\CAP9\C++\EX22\_B.EXE

23. Faça um programa para consultar todos os clientes e suas contas cujos nomes começam por uma letra digitada pelo usuário nos arquivos criados nos exercícios 20 e 21.



**SOLUÇÃO:**

- Abrir os arquivos que sofrerão a consulta (clientes e contas)
- Se o arquivo clientes estiver vazio, então mensagem

- Se o arquivo de clientes não estiver vazio, percorrê-lo do início
- ➔ ao fim e, quando verificar que o nome do cliente
- ➔ começa com a letra digitada, mostrar os dados do cliente e buscar
- ➔ as contas associadas ao cliente
- Fechar os arquivos

**1<sup>a</sup> SOLUÇÃO - SEM O CAMPO ATIVO:**

```
\EXERC\CAP9\PASCAL\EX23_A.PAS e \EXERC\CAP9\PASCAL\EX23_A.EXE
```

**2<sup>a</sup> SOLUÇÃO - COM O CAMPO ATIVO:**

```
\EXERC\CAP9\PASCAL\EX23_B.PAS e \EXERC\CAP9\PASCAL\EX23_B.EXE
```

**1<sup>a</sup> SOLUÇÃO - INCLUSÃO SEQÜENCIAL - SEM O CAMPO ATIVO:**

```
\EXERC\CAP9\C++\EX23_A.CPP e \EXERC\CAP9\C++\EX23_A.EXE
```

**2<sup>a</sup> SOLUÇÃO - INCLUSÃO SEQÜENCIAL - COM O CAMPO ATIVO:**

```
\EXERC\CAP9\C++\EX23_B.CPP e \EXERC\CAP9\C++\EX23_B.EXE
```

**24.** Faça um programa que receba um depósito de um cliente, ou seja, atualize o saldo do cliente no arquivo criado no Exercício 21.

**SOLUÇÃO:**

- Abrir o arquivo que sofrerá consultas e atualizações
- Solicitar número da conta que receberá o depósito e o valor que
- ➔ será depositado
- Se o arquivo contas estiver vazio, então mensagem
- Se o arquivo não estiver vazio, então percorrer todo o arquivo de
- ➔ contas do começo ao fim verificando se o número da conta é igual
- ➔ ao número informado.
- Se encontrar a conta atualizar o saldo. Caso contrário, mensagem
- Fechar o arquivo

**1<sup>a</sup> SOLUÇÃO - SEM O CAMPO ATIVO:**

```
\EXERC\CAP9\PASCAL\EX24_A.PAS e \EXERC\CAP9\PASCAL\EX24_A.EXE
```

**2<sup>a</sup> SOLUÇÃO - COM O CAMPO ATIVO:**

```
\EXERC\CAP9\PASCAL\EX24_B.PAS e \EXERC\CAP9\PASCAL\EX24_B.EXE
```

**1<sup>a</sup> SOLUÇÃO - SEM O CAMPO ATIVO:**

```
\EXERC\CAP9\C++\EX24_A.CPP e \EXERC\CAP9\C++\EX24_A.EXE
```

**2<sup>a</sup> SOLUÇÃO - COM O CAMPO ATIVO:**

```
\EXERC\CAP9\C++\EX24_B.CPP e \EXERC\CAP9\C++\EX24_B.EXE
```

**25.** Faça um programa que receba um saque de um cliente, ou seja, atualize o saldo do cliente no arquivo criado no Exercício 21.

**SOLUÇÃO:**

- Abrir o arquivo
- Solicitar número da conta que receberá o saque e o valor que será sacado
- Se o arquivo contas estiver vazio, então mensagem
- Se o arquivo não estiver vazio, então percorrer todo o arquivo de contas do começo ao fim verificando se o número da conta é igual ao número informado.
- Se encontrar a conta, atualizar o saldo. Caso contrário, mensagem
- Fechar o arquivo

**1<sup>a</sup> SOLUÇÃO - SEM O CAMPO ATIVO:**

\EXERC\CAP9\PASCAL\EX25\_A.PAS e \EXERC\CAP9\PASCAL\EX25\_A.EXE

**2<sup>a</sup> SOLUÇÃO - COM O CAMPO ATIVO:**

\EXERC\CAP9\PASCAL\EX25\_B.PAS e \EXERC\CAP9\PASCAL\EX25\_B.EXE

**1<sup>a</sup> SOLUÇÃO - SEM O CAMPO ATIVO:**

\EXERC\CAP9\C++\EX25\_A.CPP e \EXERC\CAP9\C++\EX25\_A.EXE

**2<sup>a</sup> SOLUÇÃO - COM O CAMPO ATIVO:**

\EXERC\CAP9\C++\EX25\_B.CPP e \EXERC\CAP9\C++\EX25\_B.EXE

## EXERCÍCIOS PROPOSTOS

- 1.** Faça um programa para criar os arquivos a seguir:

### CLIENTES

Cod\_Cli  
Nome  
Endereco  
Fone

### RECEBIMENTOS

Num\_doc  
Valor\_doc  
Data\_Emissao  
Data\_Vencimento  
Cod\_Cli

- 2.** Faça um programa para cadastrar clientes no arquivo criado no Exercício 1.
- 3.** Faça um programa que inclua recebimentos no arquivo criado no Exercício 1, devendo verificar se o cliente já se encontra cadastrado.
- 4.** Faça um programa que exclua clientes e, consequentemente, todos os seus recebimentos, dos arquivos criados no Exercício 1.
- 5.** Faça um programa que altere o cadastro de clientes do Exercício 1. O usuário deve informar o código do cliente que será alterado.

6. Faça um programa que altere um recebimento de um cliente, ou seja, o usuário informa o número do documento e o número do cliente e faz as alterações desejadas.
7. Faça um programa que mostre todos os recebimentos com data de vencimento dentro de um período qualquer. Não esqueça de mostrar também o nome do cliente e o total de dias em atraso. Quando não houver atraso, mostrar zero.
8. Faça um programa que mostre todos os recebimentos de um determinado cliente.
9. Faça um programa que mostre todos os recebimentos com valor acima de um valor dado pelo usuário.
10. Faça um programa que mostre todos os recebimentos com valor abaixo de um valor dado pelo usuário.
11. Faça um programa que mostre todos os recebimentos com valor entre dois valores dados pelo usuário.
12. Faça um programa para criar os arquivos a seguir:
  - ◆ Estilista (código do estilista, nome do estilista, salário)
  - ◆ Roupa (código da roupa, descrição da roupa, código do estilista, código da estação, ano)
  - ◆ Estação (código da estação, nome da estação)
13. Faça um programa para:
  - ◆ Cadastrar as estações climáticas, por exemplo, primavera-verão e outono-inverno;
  - ◆ Cadastrar os estilistas;
  - ◆ Cadastrar as roupas. Lembre-se de que estilista e estação devem ter sido previamente cadastrados;
  - ◆ Mostrar um relatório de todas as roupas de uma determinada estação, informando, inclusive, o nome do estilista que a desenhou.
14. Faça um programa que apresente o seguinte menu de opções:
  1. Criar
  2. Incluir
  3. Sair

Digite a opção desejada

**Na opção 1:** criar um arquivo com os campos: numero, nome, nota1 e nota2.

**Na opção 2:** incluir **todos** os dados digitados, podendo haver repetição. No final da inclusão após pressionar ENTER o programa deve mostrar todos os registros cadastrados, calcular e mostrar a média das notas de cada registro.

- 15. Faça um programa para criar um arquivo chamado PRODUTOS.DAT, onde cada registro será composto pelos seguintes campos: codigo, descricao e preco.
- 16. Faça um programa para incluir produtos no arquivo criado no Exercício 15, lembrando que não podem existir dois produtos com o mesmo código.
- 17. Faça um programa para consultar a descrição de todos os produtos que possuem preço superior a R\$ 500,00.

- 18.** Faça um programa para alterar os preços de todos os produtos em 15%.
- 19.** Faça um programa para alterar os preços dos produtos em R\$ 10,00, mas apenas os produtos que já custam mais de R\$ 100,00.
- 20.** Faça um programa para consultar todos os produtos cujos nomes começem pela letra M.
- 21.** Faça um programa para excluir produtos do arquivo criado no Exercício 15.
- 22.** Faça um programa para consultar os produtos com preços inferiores a R\$ 15,00.
- 23.** Faça um programa para consultar todos os produtos com nomes começados por uma letra digitada pelo usuário e com preços entre dois valores também fornecidos pelo usuário.
- 24.** Faça um programa para excluir todos os produtos com preço superior a R\$ 200,00.
- 25.** Faça um programa para conceder um percentual de desconto dado pelo usuário aos produtos cujos preços estão entre dois valores, também fornecidos pelo usuário.

## CAPÍTULO

# 10

## SUB-ROTINAS

### 10.1 SUB-ROTINAS (PROGRAMAÇÃO MODULARIZADA)

Sub-rotinas, também chamadas de subprogramas, são blocos de instruções que realizam tarefas específicas. O código de uma sub-rotina é carregado uma vez e pode ser executado quantas vezes for necessário. Dessa maneira, os programas tendem a ficar menores e mais organizados, uma vez que o problema pode ser subdividido em pequenas tarefas.

Os programas em geral são executados linearmente, uma linha após a outra, até o fim. Entretanto, quando são utilizadas sub-rotinas, é possível a realização de desvios na execução natural dos programas. Esses desvios são realizados quando uma função é chamada pelo programa principal. Observe o exemplo a seguir (a numeração das linhas à esquerda está sendo utilizada apenas para facilitar a explicação).

```
1      ALGORITMO
2      DECLARE sal NUMÉRICO
3          LEIA sal
4          aum ← calculo (sal)
5          novo_sal ← sal + aum
6          ESCREVA "Novo salário é ", novo_sal
7      FIM ALGORITMO

8      SUB-ROTINA calculo (sal NUMÉRICO)
9          DECLARE perc, valor
10         LEIA perc
11         valor ← sal * perc / 100
12         RETORNE valor
13     FIM SUB-ROTINA calculo
```

O algoritmo apresentado tem como objetivo geral receber o valor do salário de um funcionário e calcular o novo salário. Para resolver esse problema, utilizou-se o programa principal (representado pelo bloco de instruções entre as linhas 1 e 7) e uma sub-rotina (representada pelo bloco de instruções entre as linhas 8 e 13).

O programa principal é executado linearmente até a linha 4. Nesse ponto (linha 4) existe uma chamada à sub-rotina **calculo** (que recebe como parâmetro o valor do salário inicial do funcionário) e o programa principal fica temporariamente suspenso. A ordem de execução das instruções é, então, desviada para a linha 8, onde começa a sub-rotina **calculo**. A execução só volta ao programa principal quando o comando **RETORNE** for executado (linha 12). Esse comando é responsável também por devolver um valor calculado dentro da

sub-rotina ao programa principal (nesse exemplo, foi devolvido o conteúdo da variável **valor**). A execução do programa principal é retomada exatamente no ponto em que foi interrompida; dessa maneira, o valor devolvido pela sub-rotina é atribuído à variável **aum** (na linha 4). A partir daí, o programa volta a ser executado linearmente até o fim (a linha 7).

## 10.2 SUB-ROTINAS EM PASCAL

Um importante recurso apresentado nas linguagens de programação é a modularização, na qual um programa pode ser particionado em sub-rotinas bastante específicas. A linguagem PASCAL possibilita a modularização por meio de *procedures* (procedimentos), *functions* (funções) e *units* (unidades).

### 10.2.1 PROCEDURES SEM PASSAGEM DE PARÂMETROS

As *procedures* (procedimentos) são rotinas chamadas pelo programa principal para executar alguma operação específica e têm a sintaxe a seguir:

```
PROCEDURE nome_da_procedure;
declaração de variáveis locais;
BEGIN
comandos;
END;
```

A procedure é chamada escrevendo apenas seu nome no programa principal. A execução é desviada do programa principal até a procedure, para essa ser executada, e depois é retornada na linha do programa principal abaixo da linha da chamada. A seguir será apresentado um exemplo de procedure sem parâmetros (a numeração das linhas não faz parte do programa).

```

1   PROGRAM EXEMPLO;
2   USES CRT;
3   VAR I, P, NUM, CONT : INTEGER;

4   PROCEDURE PAR;
5   {Conta quantos nºs pares existem entre 0 e o
6   ➔ número digitado no programa principal}
7   BEGIN
8       CONT:=0;
9       P:=0;
10      WHILE CONT <= NUM DO
11          BEGIN
12              P:=P+1;
13              CONT:=CONT+2;
14          END;

15      PROCEDURE IMPAR;
16      {Conta quantos nºs ímpares existem entre 0 e o
17      ➔ número digitado no programa principal}
18      BEGIN
19          CONT:=1;
20          I:=0;
21          REPEAT
22              IF NUM < > 0
23              THEN BEGIN
24                  I:=I+1;
25                  CONT:=CONT+2;
```

```

25           END;
26       UNTIL CONT > NUM;
27   END;

28   BEGIN
29   {Início do programa principal}
30   WRITELN('DIGITE O NUMERO DO INTERVALO ');
31   READLN (NUM);
32   PAR;
33   IMPAR;
34   WRITELN ('QUANTIDADE DE PARES = ',P);
35   WRITELN ('QUANTIDADE DE ÍMPARES = ',I);
36   READLN;
37   END.

```

O programa começa sua execução no BEGIN principal, no exemplo anterior representado pela linha 28. Posteriormente, executa as linhas 29, 30 e 31. Na linha 32 existe a chamada a uma procedure. O programa principal é desviado para a procedure chamada PAR; portanto, o programa vai para a linha 4, executando toda a procedure, ou seja, as linhas de 4 a 14. Em seguida, o programa retorna à linha 33 que é a linha abaixo da linha onde ocorreu o desvio. Na linha 33 existe a chamada a uma procedure. O programa principal é desviado para a procedure chamada IMPAR, portanto o programa vai para a linha 15, executando toda a procedure, ou seja, as linhas de 15 a 27. Em seguida, o programa retorna à linha 34, que é a linha abaixo da linha onde ocorreu o desvio, executando finalmente as linhas 34, 35, 36 e 37 do programa principal.

### 10.2.2 PROCEDURES COM PASSAGEM DE PARÂMETROS

Pode-se utilizar procedures com passagem de parâmetros, ou seja, a procedure é chamada e, na sua chamada, lhe são atribuídos alguns valores, seguindo a sintaxe:

```

PROCEDURE nome_da_procedure(x,y:tipo_dos_dados);
declaração_de_variáveis_locais;
BEGIN
comandos;
END;

1   PROGRAM EXEMPLO;
2   USES CRT;
3   VAR I, COL1, COL2, LIN1, LIN2, X : INTEGER;

4   PROCEDURE DESENHA(C1,C2,L1, L2 : INTEGER);
5   {Desenha um quadrado com *}
6   {Onde os limites do quadrado estão nas
    ↪ variáveis C1, C2, L1 e L2}
7   BEGIN
8     FOR I:= C1 TO C2 DO
9       BEGIN
10      GOTOXY(I,L1);
11      WRITE('*');
12      END;
13     FOR I:= L1 TO L2 DO
14       BEGIN
15      GOTOXY(C1,I);
16      WRITELN('*');
17      END;
18     END;

19   BEGIN
20   {Programa principal - desenha 5 quadrados com *}

```

```

21      X:= 0;
22      REPEAT
23          CLRSCR;
24          X:= X+1;
25          WRITELN('DIGITE OS VALORES DAS BORDAS');
26          READLN(COL1,COL2,LIN1,LIN2);
27          CLRSCR;
28          DESENHA(COL1,COL2,LIN1,LIN2);
29          READLN;
30      UNTIL X = 5;
31  END.

```

O programa começa sua execução no BEGIN principal, no exemplo anterior representado pela linha 19. Posteriormente, executa as linhas 20, 21, 22, 23, 24, 25, 26 e 27. Na linha 28 existe a chamada a uma procedure. O programa principal é desviado para a procedure chamada DESENHA, que possui em seu cabeçalho quatro variáveis; portanto, o programa vai para a linha 4, executando toda a procedure, ou seja, as linhas de 4 a 18. Em seguida, o programa retorna à linha 29 que é a linha abaixo da linha onde ocorreu o desvio. Executa a linha 29 e a linha 30, que faz o programa voltar para a linha 22 até que a variável x possua o valor 5. Finalmente a linha 31 do programa principal será executada.

Quando a procedure DESENHA é chamada no programa principal lhe são passados os seguintes parâmetros, ou seja, valores, COL1, COL2, LIN1 e LIN2. Valores esses que serão colocados nas variáveis C1, C2, L1 e L2 da procedure.

### 10.2.3 – FUNCTION SEM PASSAGEM DE PARÂMETROS

Uma *function* (função) tem o mesmo objetivo de uma procedure, ou seja, desviar a execução do programa principal para realizar uma tarefa específica, com uma única diferença, uma function sempre retorna um valor. A sintaxe de uma function é:

```

FUNCTION nome da function : tipo de dado do valor retornado;
declaração de variáveis locais;
BEGIN
comandos;
END;

```

A function é chamada quando seu nome é atribuído a uma variável que receberá seu valor de retorno. A execução é desviada do programa principal até a function, para essa ser executada e, depois, é retornada na linha do programa principal, que fez a chamada. A seguir, será apresentado um exemplo de uma function (a numeração das linhas não faz parte do programa).

```

1  PROGRAM EXEMPLO;
2  USES CRT;
3  VAR CALC, X : REAL;v

4  FUNCTION RAIZ : REAL;
5  {Função que calcula a raiz quadrada de um número}
6  BEGIN
7      RAIZ:= SQRT(X);
8      END;

9  BEGIN
10 {Programa principal}
11     CLRSCR;
12     WRITELN('DIGITE UM VALOR PARA CALCULAR A RAIZ');
13     READLN(X);
14     IF X < 0

```

```

15      THEN WRITELN('NÃO EXISTE RAIZ QUADRADA')
16      ELSE BEGIN
17          CALC:= RAIZ;
18          WRITELN(' RAIZ DE ',X:6:2,' = ',CALC:6:2);
19          END;
20          READLN;
21      END.

```

O programa começa sua execução no BEGIN principal, no exemplo anterior representado pela linha 9. Posteriormente, executa as linhas 10, 11, 12, 13, 14, 15, 16 e 17. Na linha 17 existe a chamada a uma function. O programa principal é desviado para a function chamada RAIZ; portanto, o programa vai para a linha 4, executando toda a function, ou seja, as linhas de 4 a 8. Em seguida, o programa retorna à linha 17 que é a linha onde ocorreu o desvio, colocando o valor calculado na variável CALC. Finalmente, as linhas 18, 19, 20 e 21 do programa principal são executadas.

#### 10.2.4 FUNCTION COM PASSAGEM DE PARÂMETROS

Pode-se utilizar functions com passagem de parâmetros, ou seja, a function é chamada e na sua chamada lhe são atribuídos alguns valores, seguindo a sintaxe:

```

FUNCTION nome da function(x,y : tipo dos dados) : tipo da dado do
valor retornado;
declaração de variáveis locais;
BEGIN
comandos;
END;

```

A function é chamada quando seu nome é atribuído a uma variável que receberá seu valor de retorno. A execução é desviada do programa principal até a function, para essa ser executada e, depois, é retornada na linha do programa principal que fez a chamada. A seguir, será apresentado um exemplo de function com passagem de parâmetros (a numeração das linhas não faz parte do programa).

```

1      PROGRAM EXEMPLO;
2      VAR CALC, X : REAL;
3          I, J: INTEGER;

4      FUNCTION RAIZ (NUM:REAL) : REAL;
5          {Função que calcula a raiz quadrada de um número}
6          BEGIN
7              RAIZ := SQRT(NUM);
8          END;

9          BEGIN
10         {Programa principal}
11         CLRSCR;
12         WRITELN('DIGITE UM VALOR ');
13         READLN(I);
14         IF I <= 0
15         THEN WRITELN('VALOR INVALIDO');
16         ELSE BEGIN
17             FOR J := 1 TO I DO
18                 BEGIN
19                     CALC:= RAIZ(J);
20                     WRITELN('RAIZ DE ',J,' = ',CALC:8:3);
21                 END;
22             END;
23             READLN;
24         END.

```

O programa começa sua execução no BEGIN principal, no exemplo anterior representado pela linha 9. Posteriormente, executa as linhas de 10 a 19. Na linha 19 existe a chamada a uma function. O programa principal é desviado para a function chamada RAIZ; portanto, o programa vai para a linha 4, executando toda a function, ou seja, as linhas de 4 a 8. Em seguida, o programa retorna à linha 19 que é a linha onde ocorreu o desvio, colocando o valor calculado na variável CALC. Finalmente, as linhas de 20 a 24 do programa principal são executadas.

Quando a function RAIZ é chamada no programa principal lhe é passado um parâmetro, ou seja, o valor de J. Valor esse que será colocado na variável NUM da function.

#### 10.2.5 UNIT

Na linguagem PASCAL existem várias units prontas para serem utilizadas. Algumas dessas units são: CRT (oferece recursos de I/O), DOS (oferece acesso às funções do sistema operacional), entre outras, mas podemos criar nossas próprias units que podem ser utilizadas por vários programas. A sintaxe de uma unit é:

```
UNIT nome_da_unidade;
INTERFACE
  declaração_de_outras_unidades_utilizadas;
  declaração_das_procedures_e_functions;
IMPLEMENTATION
  desenvolvimento_das_procedures_e_functions;
END.
```

A unit é chamada quando seu nome é declarado na cláusula USES e, dentro do programa, o nome da procedure ou function, que está desenvolvida dentro da unit, é chamado. A execução é desviada do programa principal para o arquivo que contém a unit e, consequentemente, para a procedure ou function chamada, depois é retornada na linha do programa principal abaixo da linha que fez a chamada. A seguir, será apresentado um exemplo de unit (a numeração das linhas não faz parte do programa).

Em um arquivo novo escrever as linhas do seguinte código e salvar como MENU.PAS.

```
1   PROGRAM OPERA;
2   USES CRT, SOMA, SUBTRAI, MULTI, DIVIDI;
3   VAR OPCAO : INTEGER;
4   BEGIN
5     CLRSCR;
6     WRITELN('DIGITE A OPÇÃO DESEJADA ');
7     WRITELN(' 1 - SOMAR');
8     WRITELN(' 2 - SUBTRAIR');
9     WRITELN(' 3 - MULTIPLICAR');
10    WRITELN(' 4 - DIVIDIR');
11    READLN(OPCAO);
12    IF OPCAO = 1
13      THEN OPER1
14    ELSE IF OPCAO = 2
15      THEN OPER2
16    ELSE IF OPCAO = 3
17      THEN OPER3
18    ELSE IF OPCAO = 4
19      THEN OPER4
20    ELSE WRITELN('OPÇÃO INVÁLIDA ');
21    READLN;
22  END.
```

Na linha 2 existe a declaração das units SOMA, SUBTRAI, MULTI e DIVIDI criadas pelo usuário, além da unit CRT que é da linguagem PASCAL. Na linha 13 existe uma

chamada à procedure OPER1 que será construída dentro da unit SOMA. Na linha 15 existe uma chamada à procedure OPER2, que será construída dentro da unit SUBTRAI. Na linha 17 existe uma chamada à procedure OPER3, que será construída dentro da unit MULTI. Na linha 19 existe uma chamada à procedure OPER4, que será construída dentro da unit DIVIDI. Todas são descritas a seguir.

Em um arquivo novo escrever as seguintes linhas de código e salvar como SOMA.PAS.

```

1   UNIT SOMA;
2   INTERFACE
3       USES CRT;
4       PROCEDURE OPER1;
5   IMPLEMENTATION
6       PROCEDURE OPER1;
7           VAR N1,N2, R: INTEGER;
8           BEGIN
9               CLRSCR;
10              WRITELN('DIGITE N1 E N2');
11              READLN(N1,N2);
12              R:= N1+N2;
13              WRITELN('RESULTADO = ',R);
14          END;
15      END.

```

Em um arquivo novo escrever as seguintes linhas de código e salvar como SUBTRAI.PAS.

```

1   UNIT SUBTRAI;
2   INTERFACE
3       USES CRT;
4       PROCEDURE OPER2;
5   IMPLEMENTATION
6       PROCEDURE OPER2;
7           VAR N1,N2,R : INTEGER;
8           BEGIN
9               CLRSCR;
10              WRITELN('DIGITE N1 E N2 ');
11              READLN(N1,N2);
12              R:=N1-N2;
13              WRITELN(' RESULTADO = ',R);
14          END;
15      END.

```

Em um arquivo novo escrever as seguintes linhas de código e salvar como MULTI.PAS.

```

1   UNIT MULTI;
2   INTERFACE
3       USES CRT;
4       PROCEDURE OPER3;
5   IMPLEMENTATION
6       PROCEDURE OPER3;
7           VAR N1,N2,R : INTEGER;
8           BEGIN
9               CLRSCR;
10              WRITELN('DIGITE N1 E N2');
11              READLN(N1,N2);
12              R:= N1*N2;
13              WRITELN('RESULTADO = ',R);
14          END;
15      END.

```

Em um arquivo novo escrever as seguintes linhas de código e salvar como DIVIDI.PAS.

```

1   UNIT DIVIDI;
2   INTERFACE
3       USES CRT;
4       PROCEDURE OPER4;
5   IMPLEMENTATION
6       PROCEDURE OPER4;
7           VAR N1,N2 : INTEGER;
8               R : REAL;
9           BEGIN
10          CLRSCR;
11          WRITELN('DIGITE N1 E N2');
12          READLN(N1,N2);
13          R:= N1/N2;
14          WRITELN('RESULTADO = ',R:6:2);
15      END;
16  END.

```

**IMPORTANTE:**

Compilar um programa para memória, gera arquivo .PAS

Compilar um programa para disco, gera arquivo .EXE

Compilar um programa, que é uma UNIT, para disco gera arquivo .TPU

### 10.3 SUB-ROTIÑAS EM C/C++ (FUNÇÕES)

Um importante recurso apresentado nas linguagens de programação é a modularização, na qual um programa pode ser particionado em sub-rotinas bastante específicas. A linguagem C/C++ possibilita a modularização por meio das funções.

Conforme já pode ser observado, um programa escrito na linguagem C/C++ tem, no mínimo, uma função chamada **main**, por onde a execução começa. Existem também muitas outras funções predefinidas na linguagem C/C++, por exemplo: `clrscr()`, `gets()`, `strcmp()`, `strcpy()` etc. Essas funções são adicionadas aos programas pela diretiva `#include`, no momento da ‘linkedição’.

Além disso, o usuário também pode criar quantas funções quiser, dependendo do problema que está sendo resolvido pelo programa.

#### 10.3.1 PASSAGEM DE PARÂMETROS E TIPOS DE RETORNO

Cada função pode receber vários valores, os parâmetros, e pode devolver um valor, o retorno. Dessa maneira, quando se especifica uma função deve-se deixar claro qual será o tipo de retorno e quais os parâmetros necessários para a execução da função.

Exemplo 1:

```

int soma (int a, int b)
{
    int s;
    s = a + b;
    return s;
}

```

No Exemplo 1, observa-se que foi realizada a soma de dois valores inteiros (**a** e **b**) que foram passados à função, como parâmetros. O resultado dessa soma, armazenado na variável **s** é, então, devolvido ao ponto em que a função foi chamada, para ser utilizado conforme a necessidade do programa. Isso só foi possível porque a função estava preparada para tratar esses valores, conforme especificado na primeira linha da função (cabeçalho).

**Exemplo 2:**

```
float divisao(int dividendo, int divisor)
{ float q;
  q = dividendo / divisor;
  return q;
}
```

No Exemplo 2 foi realizada uma operação de divisão. Pode-se observar que a função recebeu dois valores inteiros para proceder a conta (as variáveis dividendo e divisor). Como a realização de uma divisão pode gerar valores não inteiros, o resultado foi armazenado em uma variável do tipo `float` chamada `q`, que foi devolvida ao ponto em que a função foi chamada. Por causa desse tipo de retorno é que a função de divisão foi definida como sendo `float`.

**Exemplo 3:**

```
void multiplicacao (float multiplicando, float multiplicador)
{ float produto;
  produto = multiplicando * multiplicador;
  cout << "\nO produto é " << produto;
  getch();
}
```

No Exemplo 3 observa-se outra variação de função. Pode-se verificar que a função recebeu dois valores para proceder à multiplicação, `multiplicando` e `multiplicador`. Entretanto, o resultado dessa operação foi mostrado dentro da própria função e nenhum valor foi devolvido ao programa principal. Por isso, como a função não tem retorno, seu tipo é `void`.

**10.3.2 PASSAGEM DE PARÂMETROS POR VALOR**

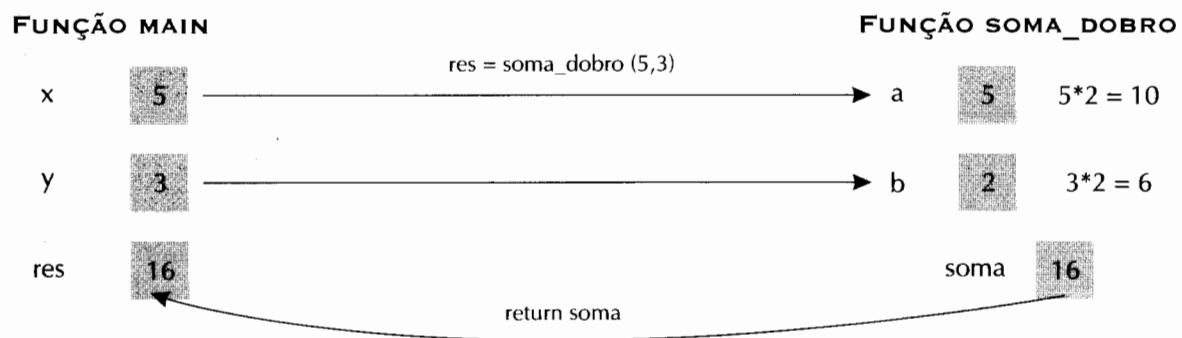
Passagem de parâmetros por valor significa que, para a execução da função, serão geradas cópias dos valores de cada um dos parâmetros. Como exemplo, observe o programa a seguir (a numeração das linhas não faz parte do programa, foram utilizadas apenas para facilitar a explicação).

```
1   #include <iostream.h>
2   #include <conio.h>
3   int soma_dobro(int a, int b);
4   void main()
5   {
6       int x, y, res;
7       cout << "\nDigite o primeiro número: ";
8       cin >> x;
9       cout << "\nDigite o segundo número: ";
10      cin >> y;
11      res = soma_dobro(x,y);
12      cout << "\nA soma do dobro dos números " << x << "
13      " e " << y << " é: " << res;
14      getch(); }

15      int soma_dobro(int a, int b)
16      {
17          int soma;
18          a = 2*a;
19          b= 2*b;
20          soma = a + b;
21          return soma;
22      }
```

**FIGURA 10.1:** Representação gráfica da passagem de parâmetros por valor.

Na Figura 10.1 é feita uma apresentação gráfica de como se dá uma passagem de parâmetros por valor apresentada no programa anterior. Estamos supondo que os valores armazenados nas variáveis **x** e **y**, por meio da execução das linhas 7 e 9 do programa anterior, foram, respectivamente, 5 e 3. Quando a linha 10 é executada, esses valores são copiados para as variáveis **a** e **b** (pertencentes à função **soma\_dobro**). Depois disso, os valores de **a** e **b** são multiplicados por 2 e, depois, é realizada a soma. O resultado dessa soma é devolvido à função **main** pela execução da linha 18, onde o valor calculado (16) recai sobre a variável **res**. No momento em que a função **soma\_dobro** chega ao fim, as variáveis **a**, **b** e **soma** são destruídas e, portanto, as alterações realizadas pelas multiplicações por 2 são perdidas.



### 10.3.3 PASSAGEM DE PARÂMETROS POR REFERÊNCIA

Passagem de parâmetros por referência significa que os parâmetros passados para uma função correspondem a endereços de memória ocupados por variáveis. Dessa maneira, toda vez que for necessário acessar um determinado valor, isso será feito por meio de referência ao seu endereço.

```

1      #include <iostream.h>
2      #include <conio.h>
3      int soma_dobro(int *a, int *b);
4      void main()
5      { int x, y, res;
6          cout << "\nDigite o primeiro número: ";
7          cin >> x;
8          cout << "\nDigite o segundo número: ";
9          cin >> y;
10         res = soma_dobro(&x,&y);
11         cout << "\nA soma dos números " << x << " e " << y <<
12         " é: " << res;
13         getch(); }
14         int soma_dobro(int *a, int *b)
15         { int soma;
16             *a = 2 * (*a);
17             *b = 2 * (*b);
18             soma = *a + *b;
19             return soma; }
```

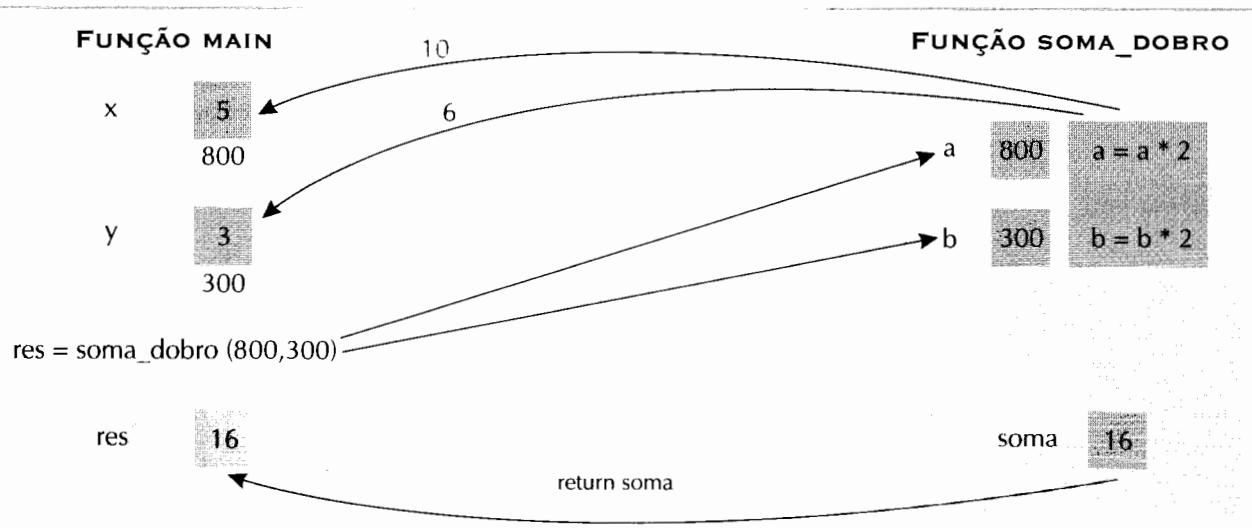


FIGURA 10.2: Representação gráfica da passagem de parâmetros por referência.

A Figura 10.2 está representando graficamente o que acontece durante a execução do programa apresentado, onde ocorre a passagem de parâmetros por referência à função `soma_dobro`.

Nas linhas 7 e 9 são lidos, respectivamente, os valores para as variáveis **x** e **y** (como exemplo, estamos supondo que foram digitados os valores 5 e 3). Entretanto, quando a função `soma_dobro` é chamada, na linha 10, são passados como parâmetros para a função os endereços de memória ocupados pelas variáveis **x** e **y** (isso é feito pelo operador `&` – que obtém o endereço de memória de uma variável), ou seja, conforme nosso exemplo, os valores 800 (endereço ocupado por **x**) e 300 (endereço ocupado por **y**). Dessa maneira, os valores que recaem sobre as variáveis **a** e **b** (da função) são, respectivamente, 800 e 300 (isso é correto, uma vez que **a** e **b** são ponteiros para `int`).

Nas linhas 15 e 16, os valores 5 e 3 são multiplicados por 2. Nesse momento ocorre a ‘referência’ aos endereços de memória 800 e 300, para que sejam obtidos os valores iniciais e, após a realização das multiplicações, os valores sejam alterados. Dessa maneira, no endereço 800 passamos a ter o valor 10 e no endereço 300 passamos a ter o valor 6. Na linha 17, é realizada a soma dos valores que estão nos endereços especificados por **a** e **b** (que já foram multiplicados por 2).

Finalmente, na linha 18, o resultado da soma é devolvido à função `main`, recaindo sobre a variável **res** e encerrando a função `soma_dobro`. Quando a função `soma_dobro` chega ao fim, as variáveis **a**, **b** e **res** são destruídas. Entretanto, as alterações decorrentes das multiplicações feitas são mantidas, pois não geraram duplicatas de valores, mas sim, a todo momento, fizeram referência a endereços de memória que estavam fora da área destinada à função.

#### IMPORTANTE:

A linguagem C/C++ não permite que um vetor ou uma matriz sejam passados na íntegra como parâmetro para uma função. Para resolver esse problema, deve-se passar apenas o endereço da posição inicial do vetor ou da matriz. Esse endereço inicial é obtido utilizando-se o nome do vetor (ou da matriz) sem a utilização do índice entre colchetes. Isso quer dizer que só é possível passar um vetor para uma função se essa passagem for por referência.

## EXERCÍCIOS RESOLVIDOS

1. Faça uma função que retorne 1 se o número digitado for positivo ou 0 se o número for negativo.

**ALGORITMO****SOLUÇÃO:**

```

ALGORITMO
DECLARE num, x NUMÉRICO
LEIA num
x ← verifica(num)
SE x = 0
ENTÃO ESCREVA "Número positivo"
SENÃO ESCREVA "Número negativo"
FIM ALGORITMO
SUB-ROTA  
INA verifica(num NUMÉRICO)
DECLARE res NUMÉRICO
SE num >= 0
ENTÃO res ← 1
SENÃO res ← 0
RETORNE res
FIM SUB-ROTA  
INA verifica

```

**SOLUÇÃO:**

\EXERC\CAP10\PASCAL\EX1.PAS e \EXERC\CAP10\PASCAL\EX1.EXE

**RESOLUÇÃO  
PASCAL****1<sup>a</sup> SOLUÇÃO:**

\EXERC\CAP10\C++\EX1\_A.CPP e \EXERC\CAP10\C++\EX1\_A.EXE

No CD você encontrará uma solução para o Exercício 1 na qual a função `verifica` é escrita antes da função `main`. Entretanto, deve-se lembrar que os programas sempre começam a ser executados pela função `main`.

**2<sup>a</sup> SOLUÇÃO:**

\EXERC\CAP10\C++\EX1\_B.CPP e \EXERC\CAP10\C++\EX1\_B.EXE

O Exercício 1, no CD, é resolvido colocando-se a função `verifica` depois da função `main`. Nesse caso, como o programa será compilado seqüencialmente (do início ao fim), deve-se acrescentar um item chamado protótipo da função. O protótipo é uma espécie de declaração da função, contendo o tipo do retorno, o nome e os parâmetros da função. Dessa maneira, quando o compilador encontrar alguma chamada à função poderá avaliar se a mesma está correta.

2. Faça uma função que receba dois números positivos por parâmetro e retorne a soma dos N números inteiros existentes entre eles.

**ALGORITMO****SOLUÇÃO:**

```

ALGORITMO
DECLARE num1, num2, s NUMÉRICO
LEIA num1, num2
s ← somar(num1, num2)
ESCREVA "soma = ", s
FIM ALGORITMO
SUB-ROTA  
INA somar(num1, num2 NUMÉRICO)

```

```

DECLARE i, s NUMÉRICO
s ← 0
PARA i ← num1+1 ATÉ num2-1 FAÇA
    INÍCIO
        s ← s + i
    FIM
RETORNE s
FIM SUB-ROTINA somar

```

**1<sup>a</sup> SOLUÇÃO - UTILIZANDO FUNCTION:**

\EXERC\CAP10\PASCAL\EX2\_A.PAS e \EXERC\CAP10\PASCAL\EX2\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO PROCEDURE:**

\EXERC\CAP10\PASCAL\EX2\_B.PAS e \EXERC\CAP10\PASCAL\EX2\_B.EXE

**1<sup>a</sup> SOLUÇÃO:**

\EXERC\CAP10\C++\EX2\_A.CPP e \EXERC\CAP10\C++\EX2\_A.EXE

**2<sup>a</sup> SOLUÇÃO:**

\EXERC\CAP10\C++\EX2\_B.CPP e \EXERC\CAP10\C++\EX2\_B.EXE

- 3.** Faça uma função que receba três números inteiros:  $a$ ,  $b$  e  $c$ , onde  $a$  é maior que 1. A função deve somar todos os inteiros entre  $b$  e  $c$  que sejam divisíveis por  $a$  (inclusive  $b$  e  $c$ ) e retornar o resultado para a função principal.

**SOLUÇÃO:**

```

ALGORITMO
DECLARE a, b, c, result NUMÉRICO
REPITA
    LEIA a
    ATÉ a>1
    LEIA b, c
    result ← divisores(a, b, c)
    ESCREVA "Soma dos inteiros = ", result
FIM ALGORITMO
SUB-ROTINA divisores(a, b, c NUMÉRICO)
    DECLARE i, s, r NUMÉRICO
    s ← 0
    PARA i ← b ATÉ c FAÇA
        INÍCIO
            r ← RESTO (i / a)
            SE (r = 0)
                ENTÃO s ← s + i
            FIM
        RETORNE s
    FIM SUB-ROTINA divisores

```

**1<sup>a</sup> SOLUÇÃO - UTILIZANDO FUNCTION:**

\EXERC\CAP10\PASCAL\EX3\_A.PAS e \EXERC\CAP10\PASCAL\EX3\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO PROCEDURE:**

\EXERC\CAP10\PASCAL\EX3\_B.PAS e \EXERC\CAP10\PASCAL\EX3\_B.EXE



**SOLUÇÃO:**

\EXERC\CAP10\C++\EX3.CPP e \EXERC\CAP10\C++\EX3.EXE

-  4. Faça uma função que transforme e mostre segundos em horas, minutos e segundos. Todas as variáveis devem ser passadas como parâmetro, não havendo variáveis globais.

**SOLUÇÃO:**

```
ALGORITMO
DECLARE seg NUMÉRICO
LEIA seg
transformacao(seg);
FIM ALGORITMO
SUB-ROTINA transformacao(segundos NUMÉRICO)
DECLARE h, m, s, r NUMÉRICO
h ← segundos / 3600
r ← RESTO(segundos / 3600)
m ← r / 60
s = RESTO(r / 60)
ESCREVA h, m, s
FIM SUB-ROTINA transformacao
```

**SOLUÇÃO:**

\EXERC\CAP10\PASCAL\EX4.PAS e \EXERC\CAP10\PASCAL\EX4.EXE

**1<sup>a</sup> SOLUÇÃO:**

\EXERC\CAP10\C++\EX4\_A.PAS e \EXERC\CAP10\C++\EX4\_A.EXE

**2<sup>a</sup> SOLUÇÃO:**

\EXERC\CAP10\C++\EX4\_B.CPP e \EXERC\CAP10\C++\EX4\_B.EXE

-  5. O número 3.025 possui a seguinte característica:

$$30 + 25 = 55$$

$$55^2 = 3.025$$

Faça uma função que receba um número inteiro de quatro dígitos e retorne 1 se o número possuir essa característica e 0, caso contrário.

**SOLUÇÃO:**

```
ALGORITMO
DECLARE num, resp NUMÉRICO
REPITA
    LEIA num
    ATÉ (num >= 1000) E (num <= 9999)
    resp ← caracteristica(num)
    SE (resp = 1)
        ENTÃO ESCREVA "O número possui as características descritas"
        SENÃO ESCREVA "O número não possui as características descritas"
    FIM ALGORITMO
SUB-ROTINA caracteristica(num NUMÉRICO)
DECLARE num1, num2, soma, pot, r NUMÉRICO
conv[5], parte1[3], parte2[3] LITERAL
```

Transformar o conteúdo de num em literal e armazenar na variável  
 ↪ conv  
 Separar o conteúdo de conv em duas partes, armazenando-as nas  
 ↪ variáveis partel e parte2  
 converter o conteúdo de partel em inteiro e armazená-lo na variável  
 ↪ num1  
 converter o conteúdo de parte2 em inteiro e armazená-lo na variável  
 ↪ num2  
 $soma \leftarrow num1 + num2$   
 $pot \leftarrow soma * soma$   
 SE (pot = num)  
 ENTÃO r  $\leftarrow 1$   
 SENÃO r  $\leftarrow 0$   
 RETORNE r  
 FIM SUB-ROTINA caracteristica

**1<sup>a</sup> SOLUÇÃO - UTILIZANDO FUNCTION:**

\EXERC\CAP10\PASCAL\EX5\_A.PAS e \EXERC\CAP10\PASCAL\EX5\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO PROCEDURE:**

\EXERC\CAP10\PASCAL\EX5\_B.PAS e \EXERC\CAP10\PASCAL\EX5\_B.EXE

**SOLUÇÃO:**

\EXERC\CAP10\C++\EX5.CPP e \EXERC\CAP10\C++\EX5.EXE

6. Faça uma função que receba como parâmetro um inteiro no intervalo de 1 a 9 e mostre a seguinte tabela de multiplicação (**no exemplo, n = 9**):

1								
2	4							
3	6	9						
4	8	12	16					
5	10	15	20	25				
6	12	18	24	30	36			
7	14	21	28	35	42	49		
8	16	24	32	40	48	56	64	
9	18	27	36	45	54	63	72	81

**SOLUÇÃO:**

```

ALGORITMO
DECLARE num NUMÉRICO
REPITA
  LEIA num
  ATÉ (num >= 1) E (num <= 9)
  multiplicacao(num)
FIM ALGORITMO
  
```

```

SUB-ROTINA multiplicacao(n NUMÉRICO)
DECLARE i, j, l, c NUMÉRICO
PARA i ← 1 ATÉ n FAÇA
    INÍCIO
        PARA j ← 1 ATÉ i FAÇA
            INÍCIO
                ESCREVA i * j
            FIM
        FIM
    FIM
FIM SUB-ROTINA multiplicacao

```

**SOLUÇÃO:**

\EXERC\CAP10\PASCAL\EX6.PAS e \EXERC\CAP10\PASCAL\EX6.EXE

**SOLUÇÃO:**

\EXERC\CAP10\C++\EX6.CPP e \EXERC\CAP10\C++\EX6.EXE

- 7.** Faça um procedimento que receba as três notas de um aluno como parâmetros e uma letra. Se a letra for A o procedimento calcula a média aritmética das notas do aluno, se for P o procedimento calcula a média ponderada com pesos 5, 3 e 2. A média calculada deve ser devolvida ao programa principal para, então, ser mostrada.

**SOLUÇÃO:**

```

ALGORITMO
DECLARE nota1, nota2, nota3, m NUMÉRICO
    Letra LITERAL
LEIA nota1
LEIA nota2
LEIA nota3
REPITA
    LEIA letra
ATÉ (letra = "A") OU (letra = "P")
m ← calcula_media(nota1, nota2, nota3, letra)
SE letra = "A"
ENTÃO ESCREVA "A média aritmética " , m
SENÃO ESCREVA "A média ponderada " , m
FIM ALGORITMO
SUB-ROTINA calcula_media(n1, n2, n3 NUMÉRICO 1 LITERAL)
DECLARE media NUMÉRICO
SE 1 = "A"
ENTÃO media ← (n1+n2+n3)/3
SENÃO media ←(n1*5+n2*3+n3*2) / (5+3+2)
RETORNE media
FIM SUB-ROTINA calcula_media

```

**1<sup>a</sup> SOLUÇÃO - UTILIZANDO PROCEDURE:**

\EXERC\CAP10\PASCAL\EX7\_A.PAS e \EXERC\CAP10\PASCAL\EX7\_A.EXE

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO FUNCTION:**

\EXERC\CAP10\PASCAL\EX7\_B.PAS e \EXERC\CAP10\PASCAL\EX7\_B.EXE

**SOLUÇÃO:**

\EXERC\CAP10\C++\EX7.CPP e \EXERC\CAP10\C++\EX7.EXE

-  8. Faça um procedimento que receba, por parâmetro, a hora de início e a hora de término de um jogo, ambas subdivididas em dois valores distintos: horas e minutos. O procedimento deve retornar a duração expressa em minutos, considerando que o tempo máximo de duração de um jogo é de 24 horas e que o jogo pode começar em um dia e terminar no outro.

**ALGORITMO****SOLUÇÃO:**

```

ALGORITMO
DECLARE hora_i, min_i, hora_f, min_f, minutos NUMÉRICO
LEIA hora_i, min_i
LEIA hora_f, min_f
minutos ← calculo(hora_i, min_i, hora_f, min_f)
ESCREVA minutos
FIM ALGORITMO
SUB-ROTINA calculo(h_i, m_i, h_f, m_f NUMÉRICO)
DECLARE tot_h, tot_m, total NUMÉRICO
SE m_f < m_i
ENTÃO INÍCIO
    m_f ← m_f + 60
    h_f ← h_f - 1
FIM
SE h_f < h_i
ENTÃO INÍCIO
    h_f ← h_f + 24
    tot_h ← h_f - h_i
FIM
tot_m ← m_f - m_i
total ← tot_h * 60 + tot_m
RETORN total
FIM SUB-ROTINA calculo

```

**RESOLUÇÃO  
PASCAL****1<sup>a</sup> SOLUÇÃO - UTILIZANDO FUNCTION:**

```
\EXERC\CAP10\PASCAL\EX8_A.PAS e \EXERC\CAP10\PASCAL\EX8_A.EXE
```

**2<sup>a</sup> SOLUÇÃO - UTILIZANDO PROCEDURE:**

```
\EXERC\CAP10\PASCAL\EX8_B.PAS e \EXERC\CAP10\PASCAL\EX8_B.EXE
```

**RESOLUÇÃO  
C/C++****SOLUÇÃO:**

```
\EXERC\CAP10\C++\EX8.CPP e \EXERC\CAP10\C++\EX8.EXE
```

-  9. Faça um procedimento que leia cinco valores inteiros e retorne o maior e o menor deles.

**ALGORITMO****SOLUÇÃO:**

```

ALGORITMO
DECLARE maior, menor NUMÉRICO;
maior_menor;
ESCREVA "O maior número digitado foi: ",maior
ESCREVA "O menor número digitado foi: ",menor
FIM_ALGORITMO
SUB-ROTINA maior_menor
DECLARE i, num NUMÉRICO
PARA i ← 1 ATÉ 5 FAÇA
INÍCIO
    ESCREVA "Digite o ", i, "º número: "

```

```

LEIA num
SE i =1
ENTÃO INÍCIO
    maior ← num
    menor ← num
    FIM
SENÃO INÍCIO
    SE num > maior
        ENTÃO maior ← num
    SE num < menor
        ENTÃO menor ← num
    FIM
FIM
RETORNE maior, menor
FIM SUB-ROTINA maior_menor

```

**SOLUÇÃO:**

\EXERC\CAP10\PASCAL\EX9.PAS e \EXERC\CAP10\PASCAL\EX9.EXE

**SOLUÇÃO:**

\EXERC\CAP10\C++\EX9.CPP e \EXERC\CAP10\C++\EX9.EXE

10. Faça uma função que receba por parâmetro um valor inteiro e positivo N e retorne o valor de S.

$$S = 1 + 1/1! + 1/2! + 1/3! + \dots + 1/N!$$

**SOLUÇÃO:**

```

ALGORITMO
DECLARE num, s NUMÉRICO
LEIA num
s ← sequencia(num)
ESCREVA s
FIM ALGORITMO
SUB-ROTINA sequencia(n NUMÉRICO)
    DECLARE a, b, f, seq NUMÉRICO
    seq ← 1
    PARA a ← 1 ATÉ n FAÇA
        INÍCIO
            f ← 1
            PARA b ← 1 ATÉ a FAÇA
                INÍCIO
                    f ← f * b
                FIM
                seq ← seq + 1 / f
            FIM
        RETORNE seq
    FIM SUB-ROTINA sequencia

```

**1ª SOLUÇÃO - UTILIZANDO FUNCTION:**

\EXERC\CAP10\PASCAL\EX10\_A.PAS e \EXERC\CAP10\PASCAL\EX10\_A.EXE

**2ª SOLUÇÃO - UTILIZANDO PROCEDURE:**

\EXERC\CAP10\PASCAL\EX10\_B.PAS e \EXERC\CAP10\PASCAL\EX10\_B.EXE

**SOLUÇÃO:**

\EXERC\CAP10\C++\EX10.CPP e \EXERC\CAP10\C++\EX10.EXE

**E** 11. Foi realizada uma pesquisa de algumas características físicas de cinco habitantes de uma certa região. De cada habitante foram coletados os seguintes dados: sexo, cor dos olhos (A – Azuis ou C – Castanhos), cor dos cabelos (L – Louros, P – Pretos ou C – Castanhos) e idade.

- ◆ Faça uma função que leia esses dados em um vetor. Determine, por meio de outra função, a média de idade das pessoas com olhos castanhos e cabelos pretos. Mostre esse resultado no programa principal.
- ◆ Faça uma função que determine e devolva ao programa principal a maior idade entre os habitantes.
- ◆ Faça uma função que determine e devolva ao programa principal a quantidade de indivíduos do sexo feminino cuja idade está entre 18 e 35 (inclusive) e que tenham olhos azuis e cabelos louros.

**SOLUÇÃO:**

```

ALGORITMO
DECLARE vetor[5] REGISTRO DE (sexo, olhos, cabelos LITERAL
                                ↪ idade NUMÉRICO)
      x, i, q, m NUMÉRICO
PARA x ← 1 ATÉ 5 FAÇA
    INÍCIO
    REPITA
        LEIA vetor[x].sexo
        ATÉ (vetor[x].sexo = "F") OU (vetor[x].sexo = "M")
    REPITA
        LEIA vetor[x].olhos
        ATÉ (vetor[x].olhos = "C") OU (vetor[x].olhos = "A")
    REPITA
        LEIA vetor[x].cabelos
        ATÉ (vetor[x].cabelos = "C") OU (vetor[x].cabelos = "L") OU
        ↪ (vetor[x].cabelos = "P")
        LEIA vetor[x].idade
    FIM
    m ← media_idade(vetor)
    ESCREVA m
    i ← maior_idade(vetor)
    ESCREVA i
    q ← quantidade_individuos(vetor)
    ESCREVA q
FIM ALGORITMO
SUB-ROTINA media_idade (v[5] REGISTRO (sexo, olhos, cabelos LITERAL
                                         idade NUMÉRICO))
DECLARE i, cont, soma, media NUMÉRICO
soma ← 0
cont ← 0
PARA i ← 1 ATÉ 5 FAÇA
    INÍCIO
        SE (v[i].olhos = "C") E (v[i].cabelos = "P")
        ENTÃO INÍCIO
            soma ← soma + v[i].idade
            cont = cont + 1
        FIM
    FIM

```

```

SE cont = 0
ENTÃO media ← 0
SENÃO media ← soma / cont
RETORNE media
FIM SUB-ROTA  
INA media_idade
SUB-ROTA  
INA maior_idade (v[5] REGISTRO (sexo, olhos, cabelos LITERAL
→ idade NUMÉRICO)
DECALRE i, maior NUMÉRICO
PARA i ← 1 ATÉ 5 FAÇA
INÍCIO
SE i = 1
ENTÃO maior ← v[i].idade
SENÃO INÍCIO
SE (v[i].idade > maior)
ENTÃO maior ← v[i].idade
FIM
FIM
RETORNE maior
FIM SUB-ROTA  
INA maior_idade
SUB-ROTA  
INA quantidade_individuos(v[5] REGISTRO (sexo, olhos, cabelos
→ LITERAL idade NUMÉRICO)
DECALRE i, qtd NUMÉRICO
qtd ← 0
PARA i ← 1 ATÉ 5 FAÇA
INÍCIO
SE (v[i].sexo = "F") E (v[i].idade >= 18) OU
(v[i].idade <= 35) E
(v[i].olhos = "A") E (v[i].cabelos = "L")
ENTÃO qtd ← qtd + 1
FIM
RETORNE qtd
FIM SUB-ROTA  
INA quantidade_individuos

```



#### SOLUÇÃO:

\EXERC\CAP10\PASCAL\EX11.PAS e \EXERC\CAP10\PASCAL\EX11.EXE



#### SOLUÇÃO:

\EXERC\CAP10\C++\EX11.CPP e \EXERC\CAP10\C++\EX11.EXE

Nesse exercício as funções recebem como parâmetro a variável **v** (um ponteiro que armazena o endereço de memória da 1<sup>a</sup> posição do vetor declarado no programa principal). Veja a solução em C/C++ no CD.

- 12.** Faça uma função que retorne ao programa principal um vetor com os três primeiros números perfeitos. Sabe-se que um número é perfeito quando é igual à soma de seus divisores (exceto ele mesmo). Exemplo: os divisores de 6 são 1, 2 e 3 e  $1 + 2 + 3 = 6$ , logo, 6 é perfeito.



#### SOLUÇÃO:

```

ALGORITMO
DECLARE vet[3], i NUMÉRICO
perfeitos(vet)
PARA i ← 1 ATÉ 3 FAÇA
INÍCIO
ESCREVA vet[i]

```

```

FIM
FIM ALGORITMO
SUB-ROTINA perfeitos(v[3] NUMÉRICO)
  DECLARE a, r, num, soma, cont NUMÉRICO
  cont ← 1
  num ← 1
  ENQUANTO (cont < 4) FAÇA
    INÍCIO
    soma ← 0
    PARA a ← 1 ATÉ num-1 FAÇA
      INÍCIO
        r ← RESTO(num / a)
        SE r = 0
        ENTÃO soma ← soma + a
      FIM
      SE soma = num
      ENTÃO INÍCIO
        v[cont] ← num
        cont ← cont + 1
      FIM
      num ← num + 1
    FIM ENQUANTO
  FIM SUB-ROTINA perfeitos

```

**SOLUÇÃO:**

\EXERC\CAP10\PASCAL\EX12.PAS e \EXERC\CAP10\PASCAL\EX12.EXE

**SOLUÇÃO:**

\EXERC\CAP10\C++\EX12.CPP e \EXERC\CAP10\C++\EX12.EXE

- 13.** Faça uma função que receba um vetor A de dez elementos inteiros, por parâmetro. Ao final dessa função, o vetor B deve conter o fatorial de cada elemento de A. O vetor B deve ser mostrado no programa principal.

A	2	1	0	3	4	...
B	2	1	1	6	24	...

**SOLUÇÃO:**

```

ALGORITMO
DECLARE x, vet1[10], vet2[10] NUMÉRICO
PARA x ← 1 ATÉ 10 FAÇA
  INÍCIO
    LEIA vet1[x]
  FIM
  fatoriais(vet1)
  PARA x ← 1 ATÉ 10 FAÇA
    INÍCIO
      ESCREVA vet2[x]
    FIM
  FIM ALGORITMO
SUB-ROTINA fatoriais(a[10] NUMÉRICO)
  DECLARE i, j , f NUMÉRICO
  PARA i ← 1 ATÉ 10 FAÇA
    INÍCIO
      SE (a[i] = 0) OU (a[i] = 1)

```

```

ENTÃO b[i] ← 1
SENÃO INÍCIO
    b[i] ← 1
    PARA j ← 1 ATÉ a[i] FAÇA
        INÍCIO
            b[i] ← b[i] * j
        FIM
    FIM
FIM
FIM SUB-ROTINA fatoriais

```

**SOLUÇÃO:**

\EXERC\CAP10\PASCAL\EX13.PAS e \EXERC\CAP10\PASCAL\EX13.EXE

**SOLUÇÃO:**

\EXERC\CAP10\C++\EX13.CPP e \EXERC\CAP10\C++\EX13.EXE

- 14.** Faça uma função que receba, por parâmetro, dois vetores de dez elementos inteiros positivos e mostre o vetor união dos dois primeiros.

**SOLUÇÃO:**

```

ALGORITMO
DECLARE x, vet1[10], vet2[10], vet3[20] NUMÉRICO
PARA x ← 1 ATÉ 10 FAÇA
    INÍCIO
        REPITA
            LEIA vet1[x]
            ATÉ vet1[x] >= 0
        FIM
    PARA x ← 1 ATÉ 10 FAÇA
        INÍCIO
            REPITA
                LEIA vet2[x]
                ATÉ vet2[x] >= 0
            FIM
        uniao(vet1,vet2,vet3)
        x ← 1
    ENQUANTO !x<=20) E (vet3[x] ≠ -1) FAÇA
        INÍCIO
            ESCREVA vet3[x]
        FIM
    FIM ALGORITMO
SUB-ROTINA uniao(a[10], b[10], u[20] NUMÉRICO)
    DECLARE i, j, k, cont NUMÉRICO
    PARA i ← 1 ATÉ 20 FAÇA
        INÍCIO
            u[i] ← -1
        FIM
    k = 1
    PARA i ← 1 ATÉ 10 FAÇA
        INÍCIO
            cont ← 1
            ENQUANTO (cont < k) E (a[i] ≠ u[cont]) FAÇA
                INÍCIO
                    cont ← cont + 1
                FIM
            SE (cont = k)

```

```

ENTÃO INÍCIO
    u[k] ← a[i]
    k ← k + 1
FIM
PARA i ← 1 ATÉ 10 FAÇA
INÍCIO
    cont ← 1
    ENQUANTO (cont < k) E (b[i] ≠ u[cont]) FAÇA
INÍCIO
    cont ← cont + 1
FIM
SE (cont = k)
    ENTÃO INÍCIO
        u[k] ← b[i]
        k ← k + 1
    FIM
FIM
FIM SUB-ROTAINA uniao

```

**SOLUÇÃO:**

\EXERC\CAP10\PASCAL\EX14.PAS e \EXERC\CAP10\PASCAL\EX14.EXE

**SOLUÇÃO:**

\EXERC\CAP10\C++\EX14.CPP e \EXERC\CAP10\C++\EX14.EXE

- 15.** Faça um procedimento que receba, por parâmetro, um vetor A com cinco números reais e retorne esses números ordenados em ordem crescente.

**SOLUÇÃO:**

```

ALGORITMO
DECLARE x, vet[5] NUMÉRICO
PARA x ← 1 ATÉ 5 FAÇA
INÍCIO
    LEIA vet[x]
FIM
ordena(vet)
PARA x ← 1 ATÉ 5 FAÇA
INÍCIO
    ESCREVA vet[x]
FIM
FIM ALGORITMO
SUB-ROTAINA ordena(v[5] NUMÉRICO)
DECLARE i, j, aux NUMÉRICO
PARA i ← 1 ATÉ 5 FAÇA
INÍCIO
    PARA j ← 1 ATÉ 4 FAÇA
    INÍCIO
        SE (v[j] > v[j+1])
        ENTÃO INÍCIO
            aux ← v[j]
            v[j] ← v[j+1]
            v[j+1] ← aux
        FIM
    FIM
FIM SUB-ROTAINA ordena

```

**SOLUÇÃO:**

\EXERC\CAP10\PASCAL\EX15.PAS e \EXERC\CAP10\PASCAL\EX15.EXE

**SOLUÇÃO:**

\EXERC\CAP10\C++\EX15.CPP e \EXERC\CAP10\C++\EX15.EXE

- 16.** Faça uma função que receba dois vetores A e B de dez elementos inteiros, por parâmetro. O procedimento deve determinar e mostrar um vetor C que contenha os elementos de A e B em ordem decrescente.

**SOLUÇÃO:**

```

ALGORITMO
DECLARE x, vet1[10], vet2[10], vet3[20] NUMÉRICO
PARA x ← 1 ATÉ 10 FAÇA
    INÍCIO
        LEIA vet1[x]
    FIM
PARA x ← 1 ATÉ 10 FAÇA
    INÍCIO
        LEIA vet2[x]
    FIM
ordena.todos(vet1,vet2,vet3)
PARA x ← 1 ATÉ 20 FAÇA
    INÍCIO
        ESCREVA vet3[x]
    FIM
FIM ALGORITMO
SUB-ROTINA ordena.todos(a[10], b[10], c[20] NUMÉRICO)
DECLARE i, j, k, cont NUMÉRICO
k ← 1
PARA i ← 1 ATÉ 10 FAÇA
    INÍCIO
        cont ← 1
        ENQUANTO (cont < k) E (a[i] < c[cont]) FAÇA
            INÍCIO
                cont ← cont + 1
            FIM
            SE cont = k
                ENTÃO INÍCIO
                    c[k] ← a[i]
                    k ← k + 1
                FIM
            SENÃO INÍCIO
                PARA j ← k ATÉ cont FAÇA
                    INÍCIO
                        c[j] ← c[j-1]
                    FIM
                    c[cont] ← a[i]
                    k ← k + 1
                FIM
            FIM
        PARA i ← 1 ATÉ 10 FAÇA
            INÍCIO
                cont ← 1
                ENQUANTO (cont < k) E (b[i] < c[cont]) FAÇA
                    INÍCIO
                        cont ← cont + 1
                    FIM
                FIM
            FIM
        FIM
    FIM
FIM

```

```

        FIM
        SE cont = k
        ENTÃO INÍCIO
            c[k] ← b[i]
            k ← k + 1
        FIM
        SENÃO INÍCIO
            PARA j ← k ATÉ cont FAÇA
            INÍCIO
                c[j] ← c[j-1]
            FIM
            c[cont] ← b[i]
            k ← k + 1
        FIM
    FIM
FIM SUB-ROTINA ordena.todos

```

**SOLUÇÃO:**

\EXERC\CAP10\PASCAL\EX16.PAS e \EXERC\CAP10\PASCAL\EX16.EXE

**SOLUÇÃO:**

\EXERC\CAP10\C++\EX16.CPP e \EXERC\CAP10\C++\EX16.EXE

17. Faça uma função que receba, por parâmetro, uma matriz A(5,5) e retorne a soma dos seus elementos.

**SOLUÇÃO:**

```

ALGORITMO
DECLARE x, y, s, matriz[5,5] NUMÉRICO
PARA x ← 1 ATÉ 5 FAÇA
    INÍCIO
        PARA y ← 1 ATÉ 5 FAÇA
        INÍCIO
            LEIA matriz[x,y]
        FIM
    FIM
    s ← soma_matriz(matriz)
    ESCREVA s
FIM ALGORITMO
SUB-ROTINA soma_matriz(m[5,5] NUMÉRICO)
DECLARE i, j, soma NUMÉRICO
soma ← 0
PARA i ← 1 ATÉ 5 FAÇA
    INÍCIO
        PARA j ← 1 ATÉ 5 FAÇA
        INÍCIO
            soma ← soma + m[i, j]
        FIM
    FIM
    RETORNE soma
FIM SUB-ROTINA soma_matriz

```

**SOLUÇÃO:**

\EXERC\CAP10\PASCAL\EX17.PAS e \EXERC\CAP10\PASCAL\EX17.EXE

**SOLUÇÃO:**

\EXERC\CAP10\C++\EX17.CPP e \EXERC\CAP10\C++\EX17.EXE

- 18.** Faça uma função que receba, por parâmetro, uma matriz A(6,6) e retorne o menor elemento da sua diagonal secundária.

**SOLUÇÃO:**

```

ALGORITMO
DECLARE x, y, menor, matriz[6,6] NUMÉRICO
PARA x ← 1 ATÉ 6 FAÇA
    INÍCIO
        PARA y ← 1 ATÉ 6 FAÇA
            INÍCIO
                LEIA matriz[x,y]
            FIM
        FIM
    FIM
menor ← menor_elemento(matriz)
ESCREVA menor
FIM ALGORITMO
SUB-ROTEIRA menor_elemento(m[6,6] NUMÉRICO)
    DECLARE i, j, me NUMÉRICO
    me ← m[1,6]
    j ← 6
    PARA i ← 1 ATÉ 6 FAÇA
        INÍCIO
            SE m[i,j] < me
                ENTÃO me ← m[i,j]
                j ← j - 1
            FIM
        RETORNE me
    FIM SUB-ROTEIRA menor_elemento

```

**SOLUÇÃO:**

\EXERC\CAP10\PASCAL\EX18.PAS e \EXERC\CAP10\PASCAL\EX18.EXE

**SOLUÇÃO:**

\EXERC\CAP10\C++\EX18.CPP e \EXERC\CAP10\C++\EX18.EXE

- 19.** Faça uma função que receba, por parâmetro, uma matriz A(6,6) e multiplique cada linha pelo elemento da diagonal principal daquela linha. A função deve retornar a matriz alterada para ser mostrada no programa principal.

**SOLUÇÃO:**

```

ALGORITMO
DECLARE x, y, matriz[6,6] NUMÉRICO
PARA x ← 1 ATÉ 6 FAÇA
    INÍCIO
        PARA y ← 1 ATÉ 6 FAÇA
            INÍCIO
                LEIA matriz[x,y]
            FIM
        FIM
    FIM
multiplica_matriz(matriz)
PARA x ← 1 ATÉ 6 FAÇA

```

```

INÍCIO
PARA y ← 1 ATÉ 6 FAÇA
    INÍCIO
        ESCREVA matriz[x,y]
    FIM
FIM
FIM ALGORITMO
SUB-ROTANA multiplica_matriz(m[6,6] NUMÉRICO)
DECLARE i, j, me, v NUMÉRICO
PARA i ← 1 ATÉ 6 FAÇA
    INÍCIO
        v ← m[i,i]
        PARA j ← 1 ATÉ 6 FAÇA
            INÍCIO
                m[i][j] ← m[i][j] * v
            FIM
        FIM
    FIM
FIM SUB-ROTANA multiplica_matriz

```

**SOLUÇÃO:**

\EXERC\CAP10\PASCAL\EX19.PAS e \EXERC\CAP10\PASCAL\EX19.EXE

**SOLUÇÃO:**

\EXERC\CAP10\C++\EX19.CPP e \EXERC\CAP10\C++\EX19.EXE

20. Faça uma função que receba, por parâmetro, uma matriz A(12,12) e retorne a média aritmética dos elementos abaixo da diagonal principal.

**SOLUÇÃO:**

```

ALGORITMO
DECLARE x, y, matriz[12,12], m NUMÉRICO
PARA x ← 1 ATÉ 12 FAÇA
    INÍCIO
        PARA y ← 1 ATÉ 12 FAÇA
            INÍCIO
                LEIA matriz[x,y]
            FIM
        FIM
        m ← media_aritmetica(matriz)
        ESCREVA m
    FIM
FIM ALGORITMO
SUB-ROTANA media_aritmetica(m[12,12] NUMÉRICO)
DECLARE i, j, cont, soma, media NUMÉRICO
soma ← 0
cont ← 0
PARA i ← 2 ATÉ 10 FAÇA
    INÍCIO
        PARA j ← 12 ATÉ (12 - i ) PASSO -1 FAÇA
            INÍCIO
                soma ← soma+ m[i,j]
                cont ← cont + 1
            FIM
        FIM
        media ← soma/cont
        RETORNE media
    FIM
FIM SUB-ROTANA media_aritmetica

```

**SOLUÇÃO:**

\EXERC\CAP10\PASCAL\EX20.PAS e \EXERC\CAP10\PASCAL\EX20.EXE

**SOLUÇÃO:**

\EXERC\CAP10\C++\EX20.CPP e \EXERC\CAP10\C++\EX20.EXE

## EXERCÍCIOS PROPOSTOS

- 1.** Faça uma função que receba um número inteiro e positivo N como parâmetro e retorne a soma dos N números inteiros existentes entre o número 1 e esse número.
- 2.** Faça uma função que receba três números inteiros como parâmetros, representando horas, minutos e segundos e os converta em segundos. Exemplo: 2 h, 40 min e 10 seg correspondem a 9.610 segundos.
- 3.** Faça uma função que receba duas cadeias de caracteres como parâmetros e retorne 0 se elas forem iguais. Caso contrário, retorne o índice do primeiro caractere não coincidente.
- 4.** Faça uma função que receba, como parâmetro, o raio de uma esfera, calcule e mostre no programa principal o seu volume,  $v = 4/3 * R^3$ .
- 5.** Faça uma função que receba um valor inteiro e verifique se o valor é positivo ou negativo.
- 6.** Faça uma função que receba, por parâmetro, a altura (alt) e o sexo de uma pessoa e retorne o seu peso ideal. Para homens calcular o peso ideal usando a fórmula a seguir: peso ideal =  $72.7 * alt - 58$  e, para mulheres: peso ideal =  $62.1 * alt - 44.7$ .
- 7.** Faça uma função que leia um número não determinado de valores positivos e retorne a média aritmética dos mesmos.
- 8.** Faça uma função que receba um valor inteiro e positivo, calcule e mostre o seu fatorial.
- 9.** Faça uma função que receba, por parâmetro, um valor inteiro e positivo e retorne a soma dos divisores desse valor.
- 10.** Faça uma função que receba por parâmetro um valor inteiro e positivo N e retorne o valor de S, onde a 1ª parcela da soma tem  $N = 1$ , a 2ª parcela tem  $N = 2$ , ..., até N ser igual ao valor digitado.

$$S = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \dots + \frac{1}{N}$$

- 11.** Faça uma função que receba, por parâmetro, um valor inteiro e positivo N e retorne o valor de S, onde a 1ª parcela da soma tem  $N = 1$ , a 2ª parcela tem  $N = 2$ , ..., até N ser igual ao valor digitado.

$$S = \frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \frac{1}{5^2} + \dots + \frac{1}{N^2}$$

**12.** Faça uma função que receba, por parâmetro, dois valores X e Z, calcule e retorne  $X^Z$  (sem utilizar funções ou operadores de potência prontos).

**13.** Foi realizada uma pesquisa entre 15 habitantes de uma certa região. De cada habitante foram coletados os dados: idade, sexo, salário e número de filhos.

Faça uma função que leia esses dados em um vetor. Faça funções que recebam esse vetor, por parâmetro, e retornem a média de salário entre os habitantes, a menor e a maior idade do grupo e a quantidade de mulheres com três filhos que recebe até R\$ 500,00 (utilize uma função para cada cálculo).

**14.** Faça uma função que receba um vetor X de 30 elementos inteiros, por parâmetro, e retorne dois vetores A e B. O vetor A deve conter os elementos pares de X e o vetor B, os elementos ímpares.

**15.** Faça uma função que receba um vetor X de 15 números inteiros, por parâmetro, e retorne a quantidade de valores pares em X.

**16.** Faça uma função que receba um vetor X de 20 de números reais, por parâmetro, e retorne a soma dos elementos de X.

**17.** Faça uma função que receba, por parâmetro, um vetor A de 25 números inteiros e substitua todos os valores negativos de A por zero.

**18.** Faça uma função que gere e mostre os dez primeiros primos acima de 100.

**19.** Faça uma função que receba, por parâmetro, dois vetores de dez números inteiros, determine e mostre o vetor intersecção dos dois vetores.

**20.** A prefeitura de uma cidade fez uma pesquisa entre os seus habitantes, coletando dados sobre o salário e número de filhos. Faça uma função que leia esses dados para um número não determinado de pessoas e retorne a média de salário da população, a média do número de filhos, o maior salário e o percentual de pessoas com salário até R\$ 350,00.