

Ana Fernanda Gomes Ascencio
Edilene Aparecida Veneruchi de Campos

Fundamentos da programação de computadores

Algoritmos, Pascal, C/C++ e Java

2^a edição

Prentice
Hall



Site com material de apoio
para professores e alunos

Fundamentos da programação de computadores

Algoritmos, Pascal, C/C++ e Java

Considerada obra de referência para alunos iniciantes no estudo da programação de computadores, este livro apresenta as principais técnicas para a construção de algoritmos e sua implementação nas linguagens Pascal, C/C++ e Java — linguagens básicas e fundamentais para o programador profissional.

A estrutura lógica e didática apresentada pelas autoras, que traz a teoria de cada capítulo com diversos exemplos, sugestões de problemas e analogias provenientes do mundo real, facilita a compreensão dos assuntos abordados e torna mais simples para o aluno o processo de aprendizagem dos conceitos em sala de aula.

Livro-texto indicado para cursos de programação de computadores e ciência da computação, também pode ser utilizado por estudantes das séries iniciais nas áreas de exatas que possuem cursos introdutórios de programação para computadores.

www.prenhall.com/ascencio_campobr
Site de apoio com exercícios resolvidos
(fontes e executáveis) para professores e alunos.



Companion
Website



Ana Fernanda Gomes Ascencio
Edilene Aparecida Veneruchi de Campos

Fundamentos da programação de computadores

Algoritmos, Pascal, C/C++ e Java

2^a edição



São Paulo

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

Dedico esta obra ao meu filho, Eduardo Gomes Ascencio,
a maior realização da minha vida, e ao meu marido, Antonio
Eduardo Pagliuso Ascencio, pela compreensão durante
o tempo de escrita deste trabalho.

Ana Fernanda Gomes Ascencio

Mais uma vez aos meus incansáveis, compreensivos e pacientes
companheiros de equipe: Vanderlei, Yanko e Juliana.

Edilene A. Veneruchi de Campos

SUMÁRIO

CAPÍTULO 1

CONCEITOS BÁSICOS 1

1.1	Conceito de algoritmo.....	1
1.2	Método para a construção de algoritmos	3
1.3	Tipos de algoritmos.....	3
1.4	Exemplos de algoritmos.....	4
1.5	Conceito de variável	7
1.6	Tipos de dados.....	8
1.7	Formação de identificadores	9
1.8	Exemplos de identificadores	9
1.9	Linguagem PASCAL.....	10
1.10	Linguagem C/C++	10
1.11	Linguagem JAVA.....	10

CAPÍTULO 2

PARADIGMAS DE PROGRAMAÇÃO 12

CAPÍTULO 3

ESTRUTURA SEQÜENCIAL 16

3.1	Estrutura seqüencial em algoritmos	16
3.2	Estrutura seqüencial em PASCAL	17
3.3	Estrutura seqüencial em C/C++	21
3.4	Estrutura seqüencial em JAVA.....	27

CAPÍTULO 4

ESTRUTURA CONDICIONAL 50

4.1	Estrutura condicional em algoritmos	50
4.2	Estrutura condicional em PASCAL.....	51
4.3	Estrutura condicional em C/C++	53
4.4	Estrutura condicional em JAVA.....	55

CAPÍTULO 5

ESTRUTURA DE REPETIÇÃO 93

5.1	Estrutura de repetição em algoritmo	93
5.2	Estrutura de repetição em PASCAL.....	96

5.3	Estrutura de repetição em C/C++.....	100
5.4	Estrutura de repetição em JAVA.....	104

CAPÍTULO 6**VETORES 145**

6.1	Vetor em algoritmos.....	145
6.2	Vetor em PASCAL.....	146
6.3	Vetor em C/C++	148
6.4	Vetor em JAVA.....	149

CAPÍTULO 7**MATRIZ 187**

7.1	Matriz em algoritmos.....	187
7.2	Matriz em PASCAL.....	190
7.3	Matriz em C/C++	193
7.4	Matriz em JAVA.....	195

CAPÍTULO 8**SUB-ROTINAS 230**

8.1	Sub-rotinas (programação modularizada)	230
8.2	Sub-rotinas em PASCAL (<i>procedures, functions e units</i>).....	231
8.3	Sub-rotinas em C/C++ (funções)	234
8.4	Sub-rotinas em JAVA (métodos).....	241

CAPÍTULO 9**MANIPULANDO CADEIAS DE CARACTERES 270**

9.1	Manipulando cadeias de caracteres em PASCAL.....	270
9.2	Manipulando cadeias de caracteres em C/C++.....	273
9.3	Manipulando cadeias de caracteres em JAVA.....	280

CAPÍTULO 10**REGISTROS 303**

10.1	Definição de registros	303
10.2	Declaração de registros em algoritmos	303
10.3	Declaração de registros em PASCAL	304
10.4	Declaração de registros em C/C++	305
10.5	Declaração de registros em JAVA	307

CAPÍTULO 11**ARQUIVOS 380**

11.1	Definição de arquivos em algoritmo	380
11.2	Trabalhando com arquivos em PASCAL	380
11.3	Trabalhando com arquivos em C/C++	385
11.4	Trabalhando com arquivos em JAVA	395

BIBLIOGRAFIA 429**ÍNDICE REMISSIVO 431**

APRESENTAÇÃO

OBJETIVOS E RESUMO

O livro proposto tem como objetivos:

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

Todos os capítulos apresentam nas seções iniciais conceitos teóricos sobre a utilização de algum recurso de computação em algoritmos e nas linguagens de programação PASCAL, C/C++ e JAVA.

A penúltima seção de cada capítulo apresenta uma série de problemas resolvidos em algoritmos, PASCAL, C/C++ e também em JAVA, e, na última, o leitor encontrará uma série de problemas para serem resolvidos.

RELEVÂNCIA, ATUALIDADE E PÚBLICO-ALVO

Durante alguns anos em que ensinamos fundamentos da programação de computadores, temos observado a grande dificuldade dos alunos em assimilar estes novos conceitos e em adquirir habilidades que lhes permitam resolver problemas reais relacionados à programação.

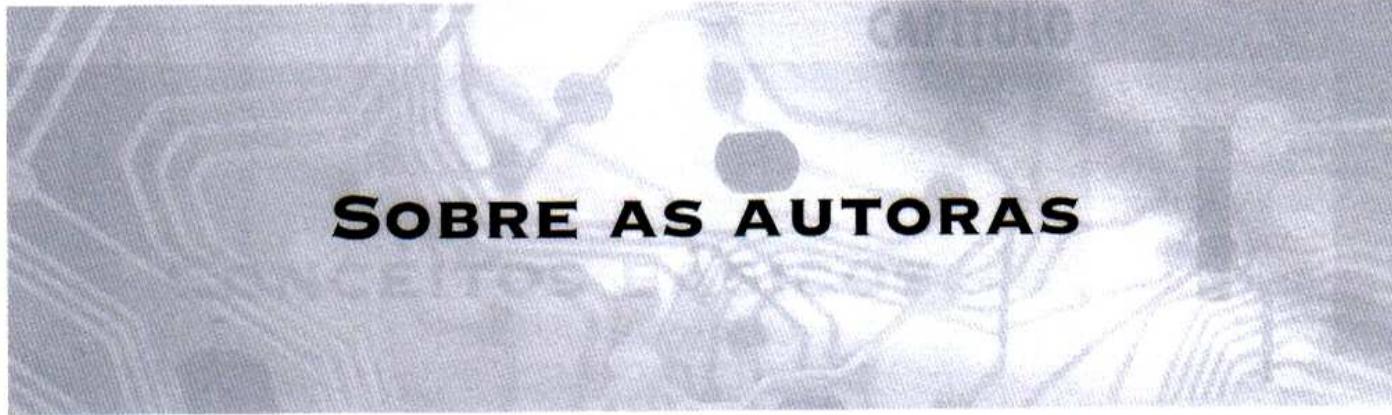
Observamos, também, que através da análise aprofundada de problemas já resolvidos os estudantes conseguem superar parte dessas dificuldades, além de adquirirem maior motivação para os estudos.

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

Este obra se diferencia das demais por possuir uma grande quantidade de exercícios resolvidos e propostos após cada capítulo, o que possibilita sua utilização em aulas de laboratório, uma prática muito comum nas universidades atualmente.



No Companion Website deste livro (http://www.prenhall.com/ascencio_campos_br), professores e alunos obtêm a resolução dos exercícios apresentados em cada capítulo em PASCAL, C/C++ e JAVA, incluindo fontes e executáveis.



SOBRE AS AUTORAS

ANA FERNANDA GOMES ASCENCIO é graduada em ciência da computação pela Pontifícia Universidade Católica de São Paulo (PUC-SP), especialista em sistemas de informação pela Universidade Federal de São Carlos (UFScar), especialista em educação pela Universidade para o Desenvolvimento do Estado e da Região do Pantanal (Uniderp), mestre em ciência da computação pela Universidade Federal do Rio Grande do Sul (UFRGS) e professora universitária na área de programação de computadores desde 1994. Autora dos livros *Lógica de programação com PASCAL*, *Fundamentos da programação de computadores*, *Introdução ao desenvolvimento de aplicações em DELPHI*, *Aplicações das estruturas de dados em DELPHI*, *Desenvolvimento de um sistema usando DELPHI e POSTGRESQL e SQL*.

EDILENE APARECIDA VENERUCHI DE CAMPOS é professora universitária desde 1997 na área de informática. Bacharel em ciência da computação pela Universidade Federal do Mato Grosso do Sul (UFMS), especialista em métodos e técnicas de ensino pela Universidade para o Desenvolvimento do Estado e da Região do Pantanal (Uniderp) e mestre em ciência da computação pela Universidade Federal do Rio Grande do Sul (UFRGS). Atualmente é professora da Uniderp, onde ministra as disciplinas linguagem de programação I e II. Além disso, é coordenadora dos cursos de bacharelado em ciência da computação e engenharia da computação e coordenadora da pós-graduação *lato sensu* em desenvolvimento de aplicações utilizando tecnologia JAVA.

CONCEITOS BÁSICOS

Desde o início de sua existência, o homem procurou criar máquinas que o auxiliassem em seu trabalho, diminuindo esforço e economizando tempo. Dentre essas máquinas, o computador vem se mostrando uma das mais versáteis, rápidas e seguras.

O computador pode auxiliá-lo em qualquer tarefa. É consciente, trabalhador, 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. Visto somente como um gabinete composto de circuitos eletrônicos, cabos e fontes de alimentação, certamente ele parece não ter nenhuma utilidade. 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).

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

Quando queremos criar ou desenvolver um software para realizar 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 criador de software entendam. Essa linguagem é chamada de *linguagem de programação*.

As etapas para o desenvolvimento de um programa são:

- ◆ **Análise** – Nesta etapa estuda-se o enunciado do problema para definir os dados de entrada, o processamento e os dados de saída.
- ◆ **Algoritmo** – Ferramentas do tipo descrição narrativa, fluxograma ou português estruturado são utilizadas para descrever o problema com suas soluções.
- ◆ **Codificação** – 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 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 perceber que executamos no dia-a-dia vários algoritmos, como se pode observar nos exemplos 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 UMA CALÇA JEANS; CASO CONTRÁRIO, PEGAR UM AGASALHO E UMA 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 diversas maneiras, porém, gerando a mesma resposta, ou seja, podem existir vários algoritmos para solucionar o mesmo problema.

1.2 MÉTODO PARA A CONSTRUÇÃO DE ALGORITMOS

Para a construção de qualquer tipo de algoritmo, é necessário seguir estes passos:

- a) Compreender completamente o problema a ser resolvido, destacando os pontos mais importantes e os objetos que o compõem.
- b) Definir os dados de entrada, ou seja, quais dados serão fornecidos e quais objetos fazem parte desse cenário-problema.
- 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. Além disso, deve-se verificar quais objetos são responsáveis pelas atividades.
- 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 sua resolução.

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 pre-definidos (Tabela 1.1), os passos a serem seguidos para sua resolução.

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

Desvantagem: é necessário aprender a simbologia dos fluxogramas e, além disso, o algoritmo resultante não apresenta muitos detalhes, dificultando 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 sua resolução.

Vantagem: a passagem do algoritmo para qualquer linguagem de programação é quase imediata, bastando conhecer as palavras reservadas dessa linguagem que serão utilizadas.

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 utilizado para indicar que deve ser tomada uma decisão, apontando 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.

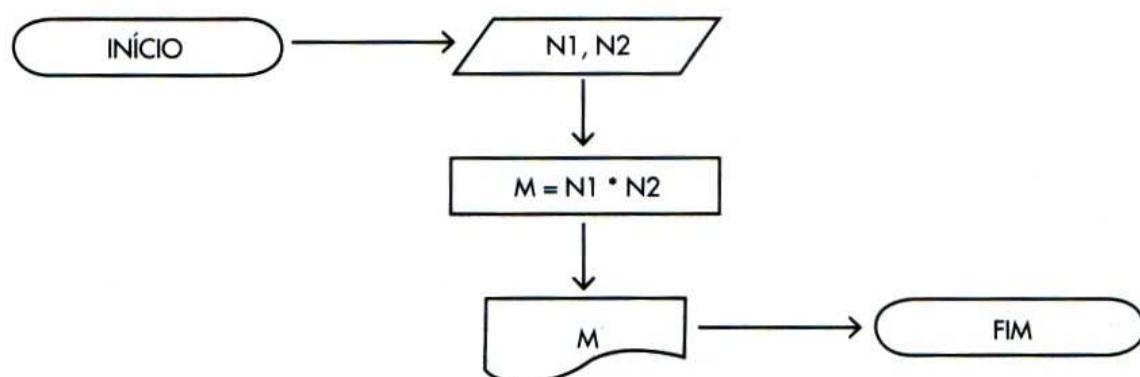
Algoritmo em descrição narrativa:

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:

```

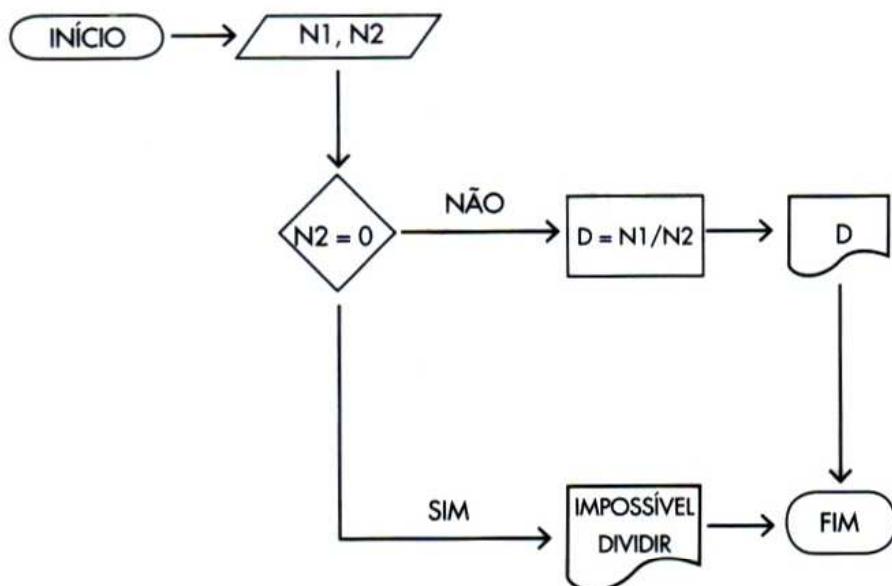
ALGORITMO
DECLARE N1, N2, M NUMÉRICO
ESCREVA "Digite dois números"
LEIA N1, N2
M ← N1 * N2
ESCREVA "Multiplicação = ", M
FIM_ALGORITMO.
```

- 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 SER FEITA A 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 mostrar sua situação, que pode ser aprovado ou reprovado.

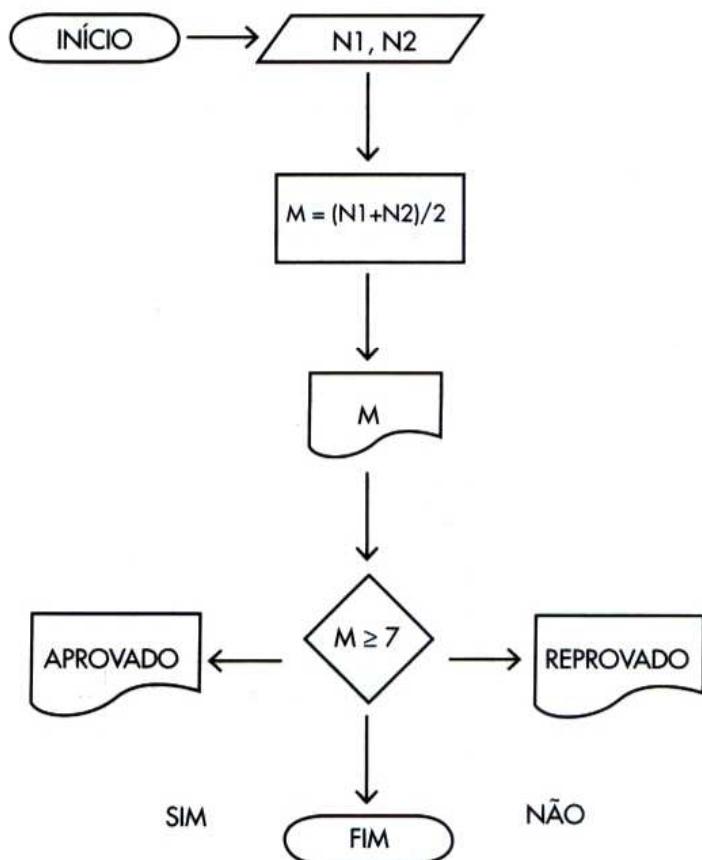
Algoritmo em descrição narrativa:

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 7, 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 recebem atualmente salário de até R\$ 500 terão aumento de 20%; os demais terão aumento de 10%.

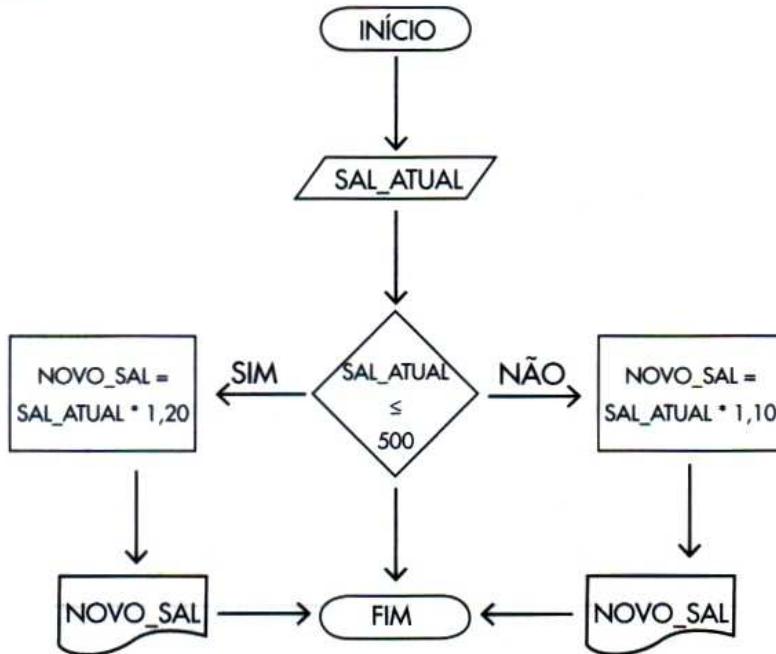
Algoritmo em descrição narrativa:

PASSO 1 — RECEBER O SALÁRIO ATUAL DO FUNCIONÁRIO.

PASSO 2 — SE O SALÁRIO ATUAL DO FUNCIONÁRIO FOR DE ATÉ R\$ 500, 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” e continua: “O primeiro número é 5”.

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

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

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

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

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

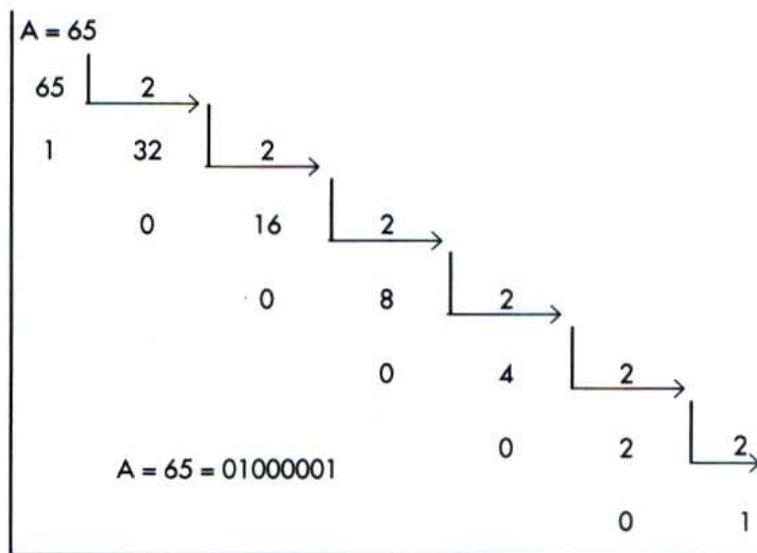
Um algoritmo e, posteriormente, um programa, recebem dados, que precisam ser armazenados no computador para serem utilizados no processamento. Esse armazenamento é feito na memória.

Todos os computadores trabalham com sistema numérico binário. Neste sistema, 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), e cada byte é identificado e acessado por meio de um endereço.

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

Desta maneira, uma variável representa uma posição de memória. Possui nome e tipo e seu 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, seu tipo (para saber quantos bytes ocupará) e 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, busca automaticamente.

1.6 TIPOS DE DADOS

Os tipos de dados mais utilizados são: *numéricos, lógicos e literais ou caracteres*, que descrevemos a seguir.

1.6.1 NUMÉRICOS

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

Os números inteiros podem ser positivos ou negativos e *não possuem parte fracionária*.

Exemplos de dados numéricos inteiros:

```

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

Os números reais podem ser positivos ou negativos e *possuem parte fracionária*.

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 LÓGICOS

São também chamados dados booleanos (por causa da álgebra de Boole) e podem assumir os valores *verdadeiro* ou *falso*.

1.6.3 LITERAIS OU CARACTERES

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 (&, #, @, ?, +).

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 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 pertençam 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 com 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, seja utilizada para qualquer fim.

Por se tratar de uma linguagem estruturada, isto é, uma linguagem que possui regras para a escrita de seus programas, é muito empregada 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:

- ◆ Apresenta um editor que permite ao desenvolvedor do programa digitar, salvar e modificar o código de seus programas.
- ◆ Possui um compilador que converte os códigos de seus programas em instruções de máquina e permite-lhe compilar, ou seja, verificar a existência de erros de sintaxe em seus programas sem retornar ao sistema operacional.
- ◆ Dispõe de um depurador que lhe permite inspecionar um programa durante sua execução, facilitando a localização de erros.
- ◆ Conta com 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 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 e codificação, utilizando o editor do ambiente de desenvolvimento Turbo Pascal.
- ◆ Compilar, utilizando o compilador do ambiente de desenvolvimento Turbo Pascal.
- ◆ Executar.

1.10 LINGUAGEM C/C++

Segundo Schildt (1996), Dennis Ritchie inventou a linguagem C e foi o primeiro a implementá-la usando um computador DEC PDP-11, que utilizava o sistema operacional Unix. Essa linguagem é resultante de um processo evolutivo de linguagens. O marco inicial foi uma linguagem chamada BCPL, desenvolvida por Martin Richards, que teve forte influência em uma linguagem denominada 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.

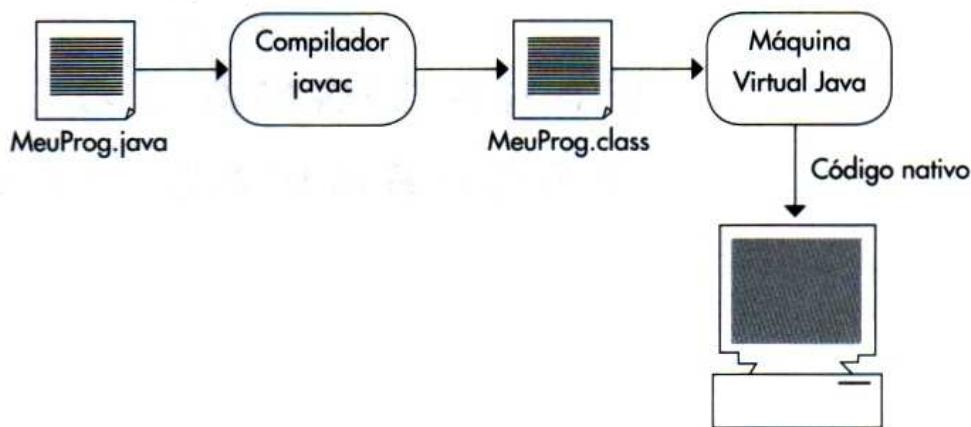
A linguagem C++ é uma extensão da linguagem C. As instruções que fazem parte desta última representam um subconjunto da primeira. Os incrementos encontrados na linguagem C++ foram feitos para dar suporte à programação orientada a objetos. A sintaxe dessa linguagem é basicamente a mesma da linguagem C.

1.11 LINGUAGEM JAVA

A tecnologia JAVA é composta pela linguagem de programação JAVA e pela plataforma de desenvolvimento JAVA.

Essa linguagem de programação possui como principais características: simplicidade, orientação a objetos, portabilidade, alta performance e segurança.

Nessa linguagem, os programas são escritos em arquivos-texto com a extensão .java. Ao serem compilados com o compilador javac, são gerados os arquivos .class. Um arquivo .class é constituído de bytecodes, código interpretado pela Máquina Virtual Java (*Java Virtual Machine*).

FIGURA 1.1 Processo de execução de um programa em JAVA.

Uma plataforma é um ambiente composto por hardware e software, ou seja, um sistema operacional e o hardware com o qual se comunica. A plataforma JAVA, entretanto, é composta apenas de software, uma vez que é a Máquina Virtual Java que faz a interface entre os programas e o sistema operacional.

A plataforma JAVA é composta:

- ◆ Pela Máquina Virtual Java, responsável por fazer a interface entre seu programa e o sistema operacional, transformando os bytecodes (comuns a qualquer ambiente) em código nativo reconhecido pelo hardware.
- ◆ Pela API (*Application Programming Interface*) JAVA, composta por um amplo conjunto de classes já implementadas e testadas que fornecem variados recursos aos desenvolvedores.

2

CAPÍTULO

PARADIGMAS DE PROGRAMAÇÃO

Um paradigma de programação está intimamente relacionado à forma de pensar do programador e como ele busca a solução para os problemas. É o paradigma que permite ou proíbe a utilização de algumas técnicas de programação. Ele é capaz, ainda, de mostrar como o programador analisou e abstraiu o problema a resolver. Existem vários paradigmas de programação: estruturado, orientado a objetos, lógico, funcional, dentre outros. Vamos analisar com mais detalhes os paradigmas estruturado e orientado a objetos.

Pelo *paradigma estruturado* (também conhecido como *imperativo*), qualquer problema pode ser resolvido utilizando três estruturas: seqüencial, condicional e iterativa (repetição). Além disso, procura encontrar uma forma de quebrar um problema complexo em pequenas partes mais simples que, trabalhadas conjuntamente, permitam solucioná-lo.

A idéia é que, utilizando corretamente tais estruturas, o recurso da modularização e a parametrização, seja possível criar programas com menor repetição possível de linhas de comandos.

Já o *paradigma orientado a objetos* comprehende o problema como uma coleção de objetos interagindo por meio de trocas de mensagem. Os objetos são estruturas de dados contendo lógica. Dessa maneira, um conjunto de objetos com informações comuns e com o mesmo comportamento dá origem a uma classe.

Exemplificando, um programador que utiliza o paradigma estruturado analisa o problema tentando relacionar as ações que deverão ser executadas e como poderão ser divididas em módulos. Um programador que utilize o paradigma orientado a objetos analisaria o mesmo problema tentando identificar os objetos que compõem essa realidade e como interagem.

Como o paradigma está ligado à forma de pensar do programador, o simples fato de se utilizar, por exemplo, uma linguagem com suporte nativo à orientação a objetos não implica que a solução apresentada seja orientada a objetos, ou, então, muitas soluções não estruturadas são feitas utilizando linguagens com suporte à estruturação.

Vamos verificar agora, por meio de um exemplo, a aplicação dos dois paradigmas na resolução de um mesmo problema.

O problema é calcular a área e o perímetro de um retângulo. Para isso, deverá existir uma janela, pela qual serão informadas as medidas dos lados do retângulo e poderão ser vistos os cálculos realizados. Trata-se de um problema simples. Como resolvê-lo?

Segundo o paradigma estruturado (ou imperativo), devemos detalhar as ações necessárias para chegar à resposta necessária.

- 1 – Obter o valor da altura do retângulo.
- 2 – Obter o valor da largura do retângulo.
- 3 – Calcular a área.
- 4 – Calcular o perímetro.
- 5 – Mostrar os cálculos realizados.

Posteriormente, devemos analisar a melhor forma de modularizar a solução, para que cada módulo realize uma tarefa bem específica, capaz de ser reutilizada o máximo possível.

Cada módulo poderá receber valores e também devolver um valor a quem solicitou. Neste exemplo, nossa solução será composta por três módulos: o principal, pelo qual a execução começará, o calculaArea, responsável por calcular e devolver o valor da área do retângulo; e o calculaPerimetro, responsável por calcular e devolver o valor do perímetro do retângulo.

Em PASCAL, usando o paradigma estruturado, a solução ficaria da seguinte forma:

```
program exemplo;
uses crt;
var altura, largura, area, perimetro: real;

function calculaArea(var a, b:real):real;
begin
    calculaArea := a * b;
end;

function calculaPerimetro(var a, b:real):real;
begin
    calculaPerimetro := 2*a + 2*b;
end;

begin
    clrscr;
    write('Digite o valor da altura do retângulo: ');
    readln(altura);
    write('Digite o valor da largura do retângulo: ');
    readln(largura);
    area := calculaArea(altura, largura);
    perimetro := calculaPerimetro(altura, largura);
    writeln('O valor da área , ', area:5:2);
    writeln('O valor do perímetro , ', perimetro:5:2);
    readln;
end.
```

Em JAVA, usando o paradigma estruturado, a solução ficaria como se segue:

```
import java.io.*;
import java.util.*;

class Retangulo
{
    public static void main(String[] args)
    {
        float altura, largura, area, perimetro;
        Scanner entrada;
        entrada = new Scanner(System.in);
        System.out.print("Digite o valor da altura do retângulo: ");
        altura = entrada.nextFloat();
        System.out.print("Digite o valor da largura do retângulo: ");
        largura = entrada.nextFloat();
        area = altura * largura;
        perimetro = 2 * altura + 2 * largura;
        System.out.println("O valor da área é "+area);
        System.out.println("O valor do perímetro é "+perimetro);
    }
}
```

Por sua vez, o paradigma orientado a objetos propõe que a solução de qualquer problema pode ser obtida seguindo estas etapas:

- a) Procurar objetos existentes no problema.
- b) Determinar as características e responsabilidades de cada objeto.
- c) Estabelecer como ocorrerá a interação entre os objetos.

Assim, pelo que foi apresentado e analisado no exemplo, observamos a existência de dois objetos: o retângulo e a janela.

O objeto retângulo tem a obrigação de armazenar e manipular o valor da altura e da largura, além de calcular a área e o perímetro.

A janela tem a obrigação de receber os valores iniciais (altura e largura) e enviá-los para o retângulo. Depois disso, deve solicitar os valores da área e do perímetro ao objeto retângulo para mostrá-los.

A comunicação entre os objetos janela e retângulo é conhecida como troca de mensagens.

Em JAVA, usando o paradigma orientado a objetos, a solução ficaria da forma descrita a seguir:

```
public class Retangulo
{
    private float altura;
    private float largura;
    public Retangulo()
    {
        altura = 0;
        largura = 0;
    }
    public float getAltura()
    {
        return altura;
    }
    public void setAltura(float a)
    {
        altura = a;
    }
    public float getLargura()
    {
        return largura;
    }
    public void setLargura(float l)
    {
        largura = l;
    }
    public float calculaArea()
    {
        return altura * largura;
    }
    public float calculaPerimetro()
    {
        return 2*altura + 2*largura;
    }
}
```

Um arquivo Retangulo.java para a classe Retangulo foi descrito anteriormente e um arquivo para a classe Janela, Janela.java, será descrito a seguir.

```
import java.io.*;
import java.util.*;

class Janela
{
    public static void main(String[] args)
    {
        Retangulo r;
        float altura, largura, area, perimetro;
        Scanner entrada;

        entrada = new Scanner(System.in);
        System.out.print("Digite o valor da altura do retângulo: ");
        altura = entrada.nextFloat();
        System.out.println("Digite o valor da largura do retângulo: ");
        largura = entrada.nextFloat();
```

```
r = new Retangulo();
r.setAltura(altura);
r.setLargura(largura);
area = r.calculaArea();
perimetro = r.calculaPerimetro();
System.out.println("O valor da área é "+area);
System.out.println("O valor do perímetro é "+perimetro);
}
}
```

Cada linguagem de programação atende a pelo menos um paradigma. Alguns autores consideram que qualquer paradigma pode ser implementado em qualquer linguagem (inclusive assembly), pois depende apenas da forma de pensar do programador e de sua habilidade de programar. De forma inversa, também se pode afirmar que o fato de a linguagem dar suporte nativo a determinado paradigma não significa que ele foi utilizado.

Além disso, deve-se observar que o paradigma orientado a objetos não exclui o estruturado. Ao contrário, eles trabalham juntos, uma vez que toda a lógica embutida nos objetos segue o pensamento estruturado.

Por fim, uma observação importantíssima precisa ser feita neste momento: este livro não se destina ao estudo da orientação a objetos, nem tem a pretensão de discutir vantagens e desvantagens dos diferentes paradigmas. Temos como objetivo o desenvolvimento da lógica em programadores iniciantes, juntamente com a utilização das estruturas de controle, das estruturas de dados e dos recursos de modularização disponíveis nas linguagens PASCAL, C/C++ e JAVA. Por isso, as soluções aqui apresentadas seguem o paradigma estruturado.

CAPÍTULO

3

ESTRUTURA SEQÜENCIAL

3.1 ESTRUTURA SEQÜENCIAL EM ALGORITMOS

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

3.1.1 DECLARAÇÃO DE VARIÁVEIS EM ALGORITMOS

As *variáveis* são declaradas após a palavra **DECLARE** e os tipos mais utilizados são: **NUMÉRICO** (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 NUMÉRICO
    Y, Z LITERAL
    TESTE LÓGICO
```

3.1.2 COMANDO DE ATRIBUIÇÃO EM ALGORITMOS

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

Exemplo:

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

3.1.3 COMANDO DE ENTRADA EM ALGORITMOS

O *comando de entrada* é utilizado para receber dados digitados pelo usuário, que serão 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**.

3.1.4 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.

3.2 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.

3.2.1 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 podem ser utilizados 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	goto	program
asm	if	record
array	implementation	repeat
begin	in	set
case	inherited	shl
const	inline	shr
constructor	interface	string
destructor	label	then
div	library	to
do	mod	type
downto	nil	unit
else	not	until
end	object	uses
exports	of	var
file	or	while
for	packed	with
function	procedure	xor

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

TIPO	FAIXA DE VALORES	TAMANHO (aproximado)
shortint	-128 a 127	8 bits
integer	-32.768 a 32.767	16 bits
longint	-2.147.483.648 a 2.147.483.647	32 bits
byte	0 a 255	8 bits
word	0 a 65.535	16 bits
real	$2,9 \times 10^{-39}$ a $1,7 \times 10^{38}$ (11 a 12 dígitos com sinal)	6 bytes
single	$1,5 \times 10^{-45}$ a $3,4 \times 10^{38}$ (7 a 8 dígitos com sinal)	4 bytes
double	$5,0 \times 10^{-324}$ a $1,7 \times 10^{308}$ (15 a 16 dígitos com sinal)	8 bytes
extended	$3,4 \times 10^{-4932}$ a $1,1 \times 10^{4932}$ (19 a 20 dígitos com sinal)	10 bytes
comp	$-9,2 \times 10^{18}$ a $9,2 \times 10^{18}$ (19 a 20 dígitos com sinal)	8 bytes
boolean	true ou false	8 bits
wordbool	true ou false	16 bits
longbool	true ou false	32 bits
bytebool	true ou false	8 bits
char	1 caractere qualquer	1 byte
string	cadeia de caracteres (no máximo 255)	tantos bytes quantos forem os caracteres

3.2.2 COMANDO DE ATRIBUIÇÃO EM PASCAL

O comando de atribuição é utilizado para dar valores ou operações a variáveis, sendo representado por := (os sinais de dois-pontos e de igualdade).

Exemplo:

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

Em PASCAL, os caracteres literais são representados entre apóstrofos; os números reais utilizam o ponto como separador decimal; cada comando é finalizado com o sinal de ponto-e-vírgula.

3.2.3 COMANDO DE ENTRADA EM PASCAL

O comando de entrada é utilizado para receber dados digitados pelo usuário. Esses dados 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.

3.2.4 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 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.

3.2.5 COMENTÁRIOS EM PASCAL

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

3.2.6 OPERADORES E FUNÇÕES PREDEFINIDAS EM PASCAL

A linguagem PASCAL possui operadores e funções predefinidas destinados a cálculos matemáticos. Alguns são apresentados a seguir.

OPERADOR	EXEMPLO	COMENTÁRIO
<code>:=</code>	<code>x := y</code>	O conteúdo da variável Y é atribuído à variável X. (A uma variável pode ser atribuído o conteúdo de outra, um valor constante ou, ainda, o resultado de uma função.)
<code>+</code>	<code>x + y</code>	Soma o conteúdo de X e de Y.
<code>-</code>	<code>x - y</code>	Subtrai o conteúdo de Y do conteúdo de X.
<code>*</code>	<code>x * y</code>	Multiplica o conteúdo de X pelo conteúdo de Y.
<code>/</code>	<code>x / y</code>	Obtém o quociente da divisão de X por Y.
<code>div</code>	<code>x div y</code>	Obtém o quociente inteiro da divisão de X por Y.
<code>mod</code>	<code>x mod y</code>	Obtém o resto da divisão de X por Y.

OBSERVAÇÕES

- a) Os operadores `div` e `mod` só podem ser aplicados com operandos do tipo inteiro.
- b) O operador `/` sempre conduz a um resultado real.
- c) Com os operadores `+, -, * e /`, se pelo menos um dos operandos for real, então o resultado será real.

OPERADOR	EXEMPLO	COMENTÁRIO
<code>=</code>	<code>x = y</code>	O conteúdo de X é igual ao conteúdo de Y.
<code><></code>	<code>x <> y</code>	O conteúdo de X é diferente do conteúdo de Y.
<code><=</code>	<code>x <= y</code>	O conteúdo de X é menor ou igual ao conteúdo de Y.
<code>>=</code>	<code>x >= y</code>	O conteúdo de X é maior ou igual ao conteúdo de Y.
<code><</code>	<code>x < y</code>	O conteúdo de X é menor que o conteúdo de Y.
<code>></code>	<code>x > y</code>	O conteúdo de X é maior que o conteúdo de Y.

FUNÇÕES MATEMÁTICAS		
FUNÇÃO	EXEMPLO	COMENTÁRIO
<code>abs</code>	<code>abs(x)</code>	Obtém o valor absoluto de X.
<code>exp</code>	<code>exp(x)</code>	Obtém o logaritmo natural e elevado à potência X.
<code>log</code>	<code>log(x)</code>	Obtém o logaritmo natural de X.
<code>trunc</code>	<code>trunc(x)</code>	Obtém a parte inteira do número real armazenado em X.
<code>frac</code>	<code>frac(x)</code>	Obtém a parte fracionária do número real armazenado em X.
<code>round</code>	<code>round(x)</code>	Arredonda X.
<code>sin</code>	<code>sin(x)</code>	Calcula o seno de X (X deve estar representado em radianos).

continua

continuação

FUNÇÕES MATEMÁTICAS

FUNÇÃO	EXEMPLO	COMENTÁRIO
cos	cos (x)	Calcula o co-seno de X (X deve estar representado em radianos).
pi	Pi	Retorna o valor de π .
sqrt	sqrt (x)	Calcula a raiz quadrada de X.
sqr	sqr (x)	Calcula X elevado ao quadrado.
inc	inc (x, y)	Incrementa a variável X com o valor da variável Y.
dec	dec (x, y)	Decrementa a variável X com o valor da variável Y.

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 e COS 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/co-seno.

Exemplo com variável para o valor de π :

```
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) +, -

Quando se tem uma expressão em que os operadores têm a mesma prioridade, a expressão é resolvida da esquerda para a direita.

Exemplos:

$$2 + 3 - 4 = 5 - 4 = 1$$

$$2 * 4 / 2 = 8 / 2 = 4$$

3.3 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 na 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.

3.3.1 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 os valores verdadeiro ou falso), pois considera verdadeiro qualquer valor diferente de 0 (zero). 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 tratados no Capítulo 6.

Exemplo:

```
float x;
```

Declara uma variável chamada `x` em que pode ser armazenado um número real.

```
float y, z;
```

Declara duas variáveis chamadas `y` e `z` em que podem ser armazenados dois números reais.

```
char SEXO;
```

Declara uma variável chamada `SEXO` em que pode ser armazenado um caractere.

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

Declara uma variável chamada `NOME` em que podem ser armazenados até 40 caracteres.

A linguagem C/C++ possui cinco tipos básicos que podem ser utilizados na declaração das variáveis: `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)
<code>char</code>	-128 a 127	8 bits
<code>unsigned char</code>	0 a 255	8 bits
<code>int</code>	-32.768 a 32.767	16 bits
<code>unsigned int</code>	0 a 65.535	16 bits
<code>short int</code>	-32.768 a 32.767	16 bits
<code>long</code>	-2.147.483.648 a 2.147.483.647	32 bits
<code>unsigned long</code>	0 a 4.294.967.295	32 bits
<code>float</code>	3.4×10^{-38} a 3.4×10^{38}	32 bits
<code>double</code>	1.7×10^{-308} a 1.7×10^{308}	64 bits
<code>long double</code>	3.4×10^{-4932} a 1.1×10^{4932}	80 bits

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

3.3.2 COMANDO DE ATRIBUIÇÃO EM C/C++

O comando de atribuição é utilizado para conceder 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 no programa, por meio da diretiva `include`, a biblioteca `string.h`. As funções de manipulação de strings serão abordadas no Capítulo 9.

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

3.3.3 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 mais 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). Os comandos `gets` e `scanf` armazenam toda a cadeia até que seja pressionada a tecla ENTER. Tanto um comando quanto outro exigem a inclusão da biblioteca `stdio.h`, ou seja, `#include <stdio.h>`.

3.3.4 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.

No comando `printf` é necessário indicar o tipo de variável que será mostrada: `%f` para variáveis que armazenam números reais, `%d` para variáveis que armazenam números inteiros, `%c` para variáveis que armazenam um único caractere e `%s` para variáveis que armazenam um conjunto de caracteres.

3.3.5 COMENTÁRIOS EM C/C++

Comentários são textos que podem ser inseridos em programas com o objetivo de documentá-los. Eles não são analisados pelo compilador.

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

Exemplo:

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

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

```
// comentário
```

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

3.3.6 OPERADORES E FUNÇÕES PREDEFINIDAS EM C/C++

A linguagem C/C++ possui operadores e funções predefinidas destinados a cálculos matemáticos. Alguns são apresentados a seguir.

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, um valor constante ou, ainda, o resultado de uma função.)
+	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.

continua

continuação

OPERADOR	EXEMPLO	COMENTÁRIO
/	x / y	Obtém o quociente da divisão de X por Y.
%	x % y	Obtém o resto da divisão de X por Y.

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

OPERADOR	EXEMPLO	COMENTÁRIO
+=	x += y	Equivale a $X = X + Y$.
-=	x -= y	Equivale a $X = X - Y$.
*=	x *= y	Equivale a $X = X * Y$.
/=	x /= y	Equivale a $X = X / Y$.
%=	x %= y	Equivale a $X = X \% Y$.
++	x++	Equivale a $X = X + 1$.
++	y = ++x	Equivale a $X = X + 1$ e depois $Y = X$.
++	y = x++	Equivale a $Y = X$ e depois $X = X + 1$.
--	x--	Equivale a $X = X - 1$.
--	y = --x	Equivale a $X = X - 1$ e depois $Y = X$.
--	y = x--	Equivale a $Y = X$ e depois $X = X - 1$.

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`.

OPERADOR	EXEMPLO	COMENTÁRIO
==	x == y	O conteúdo de X é igual ao conteúdo de Y.
!=	x != y	O conteúdo de X é diferente do conteúdo de Y.
<=	x <= y	O conteúdo de X é menor ou igual ao conteúdo de Y.
>=	x >= y	O conteúdo de X é maior ou igual ao conteúdo de Y.
<	x < y	O conteúdo de X é menor que o conteúdo de Y.
>	x > y	O conteúdo de X é maior que o conteúdo de Y.

FUNÇÕES MATEMÁTICAS		
FUNÇÃO	EXEMPLO	COMENTÁRIO
ceil	ceil(x)	Arredonda um número real para cima. Por exemplo, <code>ceil(3.2)</code> é 4.
cos	cos(x)	Calcula o co-seno de X (X deve estar representado em radianos).
exp	exp(x)	Obtém o logaritmo natural e elevado à potência X.
abs	abs(x)	Obtém o valor absoluto de X.
floor	floor(x)	Arredonda um número real para baixo. Por exemplo, <code>floor(3.2)</code> é 3.

continua

continuação

FUNÇÕES MATEMÁTICAS		
FUNÇÃO	EXEMPLO	COMENTÁRIO
log	log (X)	Obtém o logaritmo natural de X.
log10	log10 (X)	Obtém o logaritmo de base 10 de X.
modf	z = modf (X, &Y)	Decompõe o número real armazenado em X em duas partes: Y recebe a parte fracionária e z, a parte inteira do número.
pow	pow (X, Y)	Calcula a potência de X elevado a Y.
sin	sin (X)	Calcula o seno de X (X deve estar representado em radianos).
sqrt	sqrt (X)	Calcula a raiz quadrada de X.
tan	tan (X)	Calcula a tangente de X (X deve estar representado em radianos).
M_PI	M_PI	Retorna o valor de π .

OBSERVAÇÃO As funções sin, cos e tan esperam receber argumentos no formato de radianos; para receberem argumentos em graus, siga o próximo exemplo.

Exemplo com variável para o valor de π :

```
VALORPI = 3.1415;
cin>> X; //X EM GRAUS
Y = SIN ((VALORPI * X) / 180);
```

Exemplo utilizando a função M_PI:

```
cin>>X; //X EM GRAUS
Y = SIN ((M_PI * X) / 180);
```

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

3.4 ESTRUTURA SEQÜENCIAL EM JAVA

```
import nome_do_pacote_das_classes;
public class nome
{
    public static void main (String args[])
    {
        bloco de comandos;
    }
}
```

Os pacotes de classes são arquivos contendo diferentes classes que possuem vários métodos, ou seja, funções, os quais podem ser utilizados nos programas escritos em JAVA. A diretiva `import` permite que o programa reconheça as classes do pacote e, consequentemente, a utilização de seus métodos.

É importante salientar que a linguagem JAVA é sensível a letras maiúsculas e minúsculas, ou seja, considera letras maiúsculas diferentes de minúsculas (por exemplo, `a` é diferente de `A`). Sendo assim, cada comando tem sua própria sintaxe, que às vezes é somente com letras minúsculas e outras vezes com letras maiúsculas e minúsculas.

3.4.1 DECLARAÇÃO DE VARIÁVEIS EM JAVA

As *variáveis* são declaradas após a especificação de seus tipos. Os tipos de dados mais utilizados são: `int` (para números inteiros), `float` e `double` (para números reais), `char` (para um caractere), `String` (para vários caracteres) e `boolean` (para verdadeiro ou falso).

Exemplo:

```
float X;
```

Declara uma variável chamada `X` em que pode ser armazenado um número real.

```
double Y, Z;
```

Declara duas variáveis chamadas `Y` e `Z` em que podem ser armazenados dois números reais.

```
char SEXO;
```

Declara uma variável chamada `SEXO` em que pode ser armazenado um caractere.

```
String NOME;
```

Declara uma variável chamada `NOME` em que podem ser armazenados vários caracteres.

A linguagem JAVA possui os tipos primitivos de dados listados a seguir.

TIPO	FAIXA DE VALORES	TAMANHO (aproximado)
<code>byte</code>	-128 a 127	8 bits
<code>char</code>	0 a 65.535	16 bits
<code>short</code>	-32.768 a 32.767	16 bits
<code>int</code>	-2.147.483.648 a 2.147.483.647	32 bits
<code>long</code>	-9.223.372.036.854.775.808 a 9.223.372.036.854.775.807	64 bits
<code>float</code>	-3.4×10^{-38} a 3.4×10^{38}	32 bits
<code>double</code>	-1.7×10^{-308} a 1.7×10^{308}	64 bits

3.4.2 COMANDO DE ATRIBUIÇÃO EM JAVA

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

Exemplo:

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

Em JAVA, os caracteres são representados entre apóstrofos ('). As cadeias de caracteres devem ser representadas entre aspas (").

Nessa linguagem, cada comando é finalizado com o sinal de ponto-e-vírgula.

3.4.3 COMANDO DE ENTRADA EM JAVA

O comando de entrada é utilizado para receber dados digitados pelo usuário. Os dados recebidos são armazenados em variáveis. Uma das formas de entrada utilizada na linguagem JAVA é por meio da classe Scanner, que requer a importação do pacote java.util.

Exemplos:

```
int n1;
Scanner dado;
dado = new Scanner(System.in);
n1 = dado.nextInt();
```

Um valor inteiro digitado pelo usuário será armazenado na variável n1.

```
float x;
Scanner dado;
dado = new Scanner(System.in);
x = dado.nextFloat();
```

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

```
String nome;
Scanner dado;
dado = new Scanner(System.in);
nome = dado.next();
```

Um valor literal digitado pelo usuário será armazenado na variável nome.

É importante salientar que todas as entradas são recebidas pela linguagem JAVA como um conjunto de caracteres. Assim, estes caracteres deverão ser convertidos por funções de conversão de tipos. Seguem algumas dessas funções.

FUNÇÃO	FUNCIONALIDADE
next()	Aguarda uma entrada em formato String com uma única palavra.
nextLine()	Aguarda uma entrada em formato String com uma ou várias palavras.
nextInt()	Aguarda uma entrada em formato inteiro.
nextByte()	Aguarda uma entrada em formato inteiro.
nextLong()	Aguarda uma entrada em formato inteiro.
nextFloat()	Aguarda uma entrada em formato número fracionário.
nextDouble()	Aguarda uma entrada em formato número fracionário.

O tratamento de cadeia de caracteres será mais detalhado no Capítulo 9.

3.4.4 COMANDO DE SAÍDA EM JAVA

O comando de saída é utilizado para mostrar dados na tela ou na impressora. Os comandos de saída mais utilizados na linguagem JAVA são `System.out.println` e `System.out.print`.

Exemplo:

```
System.out.println(X);
```

Mostra o valor armazenado na variável x.

```
System.out.println("Conteúdo de X = " + X);
```

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

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

3.4.5 COMENTÁRIOS EM JAVA

Comentários são textos que podem ser inseridos em programas com o objetivo de documentá-los. Eles são ignorados pelo interpretador.

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

Exemplo:

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

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

```
// comentário
```

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

3.4.6 OPERADORES E FUNÇÕES PREDEFINIDAS EM JAVA

A linguagem JAVA possui operadores e funções predefinidas destinados a cálculos matemáticos. Alguns são apresentados a seguir.

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.)
+	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 += y	Equivale a $X = X + Y$.
-=	x -= y	Equivale a $X = X - Y$.
*=	x *= y	Equivale a $X = X * Y$.

continua

continuação

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++</code>	Equivale a $X = X + 1$.
<code>++</code>	<code>y = ++x</code>	Equivale a $X = X + 1$ e depois $Y = X$.
<code>++</code>	<code>y = x++</code>	Equivale a $Y = X$ e depois $X = X + 1$.
<code>--</code>	<code>x--</code>	Equivale a $X = X - 1$.
<code>--</code>	<code>y = --x</code>	Equivale a $X = X - 1$ e depois $Y = X$.
<code>--</code>	<code>y = x--</code>	Equivale a $Y = X$ e depois $X = X - 1$.

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 acima, o operador `+=` está sendo usado para realizar a operação $x + y$ e, posteriormente, atribuir o resultado obtido à variável `x`.

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 de <code>Y</code> .
<code><=</code>	<code>x <= y</code>	O conteúdo de <code>X</code> é menor ou igual ao conteúdo de <code>Y</code> .
<code>>=</code>	<code>x >= y</code>	O conteúdo de <code>X</code> é maior ou igual ao conteúdo de <code>Y</code> .
<code><</code>	<code>x < y</code>	O conteúdo de <code>X</code> é menor que o conteúdo de <code>Y</code> .
<code>></code>	<code>x > 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>Math.ceil(X)</code>	Arredonda um número real para cima. Por exemplo, <code>ceil(3.2)</code> é 4.
<code>cos</code>	<code>Math.cos(X)</code>	Calcula o co-seno de <code>X</code> (<code>X</code> deve estar representado em radianos).
<code>exp</code>	<code>Math.exp(X)</code>	Obtém o logaritmo natural e elevado à potência <code>X</code> .
<code>abs</code>	<code>Math.abs(X)</code>	Obtém o valor absoluto de <code>X</code> .
<code>floor</code>	<code>Math.floor(X)</code>	Arredonda um número real para baixo. Por exemplo, <code>floor(3.2)</code> é 3.
<code>log</code>	<code>Math.log(X)</code>	Obtém o logaritmo natural de <code>X</code> .
<code>log10</code>	<code>Math.log10(X)</code>	Obtém o logaritmo de base 10 de <code>X</code> .
<code>pow</code>	<code>Math.pow(X, Y)</code>	Calcula a potência de <code>X</code> elevado a <code>Y</code> .
<code>sin</code>	<code>Math.sin(X)</code>	Calcula o seno de <code>X</code> (<code>X</code> deve estar representado em radianos).
<code>sqrt</code>	<code>Math.sqrt(X)</code>	Calcula a raiz quadrada de <code>X</code> .
<code>cbrt</code>	<code>Math.cbrt(X)</code>	Calcula a raiz cúbica de <code>X</code> .
<code>tan</code>	<code>Math.tan(X)</code>	Calcula a tangente de <code>X</code> (<code>X</code> deve estar representado em radianos).
<code>PI</code>	<code>Math.PI</code>	Retorna o valor de π .
<code>toDegrees</code>	<code>Math.toDegrees(X)</code>	Converte a medida de <code>X</code> de radianos para graus.
<code>toRadians</code>	<code>Math.toRadians(X)</code>	Converte a medida de <code>X</code> de graus para radianos.

OBSERVAÇÃO Os métodos `sin`, `cos` e `tan` esperam receber argumentos no formato de radianos; para receberem argumentos em graus, siga o próximo exemplo.

```
dado = new Scanner(System.in);
x = dado.nextDouble();
y = Math.sin(Math.toRadians(x));
```

A linguagem JAVA possui muitas outras funções matemáticas que podem ser observadas detalhadamente na documentação da classe `Math`.

PALAVRAS RESERVADAS dessa linguagem são nomes utilizados pelo compilador para representar comandos de controle do programa, operadores e diretivas.

abstract	continue	for	new	switch
assert	default	if	package	synchronized
boolean	do	goto	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp	volatile
const	float	native	super	while

EXERCÍCIOS RESOLVIDOS

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

ALGORITMO

SOLUÇÃO:

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



1ª SOLUÇÃO:

\EXERC\CAP3\PASCAL\EX1_A.PAS e \EXERC\CAP3\PASCAL\EX1_A.EXE

2ª SOLUÇÃO:

\EXERC\CAP3\PASCAL\EX1_B.PAS e \EXERC\CAP3\PASCAL\EX1_B.EXE



1ª SOLUÇÃO:

\EXERC\CAP3\C++\EX1_A.CPP e \EXERC\CAP3\C++\EX1_A.EXE

2^a SOLUÇÃO:

```
\EXERC\CAP3\C++\EX1_B.CPP e \EXERC\CAP3\C++\EX1_B.EXE
```

**1^a SOLUÇÃO:**

```
\EXERC\CAP3\JAVA\EX1_A.java e \EXERC\CAP3\JAVA\EX1_A.class
```

2^a SOLUÇÃO:

```
\EXERC\CAP3\JAVA\EX1_B.java e \EXERC\CAP3\JAVA\EX1_BA.class
```

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

ALGORITMO**1^a SOLUÇÃO:****ALGORITMO**

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

2^a 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^a SOLUÇÃO:**

```
\EXERC\CAP3\PASCAL\EX2_A.PAS e \EXERC\CAP3\PASCAL\EX2_A.EXE
```

2^a SOLUÇÃO:

```
\EXERC\CAP3\PASCAL\EX2_B.PAS e \EXERC\CAP3\PASCAL\EX2_B.EXE
```

Quando estamos trabalhando com tipos de dados reais, precisamos fazer a formatação desses números. Se isso não for feito, eles 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, das quais: duas casas para a parte fracionária, uma casa para o ponto e as três casas restantes para a parte inteira.
- Y:8:3 A variável y será mostrada com oito casas, das quais: três casas para a parte fracionária, 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 corresponde ao total de casas ocupadas pela parte fracionária. O ponto, que é o separador entre a parte inteira e fracionária, também ocupa uma casa.

3^a SOLUÇÃO:

```
\EXERC\CAP3\PASCAL\EX2_C.PAS e \EXERC\CAP3\PASCAL\EX2_C.EXE
```



1^a SOLUÇÃO:

```
\EXERC\CAP3\C++\EX2_A.CPP e \EXERC\CAP3\C++\EX2_A.EXE
```

Quando estamos trabalhando com tipos de dados reais, precisamos fazer a formatação desses números para definir quantas casas decimais 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 no 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, destas, uma será usada para mostrar o ponto e três outras, para mostrar sua parte fracionária).

2^a SOLUÇÃO:

```
\EXERC\CAP3\C++\EX2_B.CPP e \EXERC\CAP3\C++\EX2_B.EXE
```

3^a SOLUÇÃO:

```
\EXERC\CAP3\C++\EX2_C.CPP e \EXERC\CAP3\C++\EX2_C.EXE
```



1^a SOLUÇÃO:

```
\EXERC\CAP3\JAVA\EX2_A.java e \EXERC\CAP3\JAVA\EX2_A.class
```

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

Deve-se utilizar o método `DecimalFormat`, conforme apresentado a seguir:

```
DecimalFormat casas;
casas = new DecimalFormat("0.00");
System.out.println("Média = "+casas.format(media));
```

No exemplo anterior, a formatação permitirá que sejam mostradas duas casas decimais para o valor da variável média. Para a utilização do método `DecimalFormat`, deve-se incluir o pacote de classes `text`, ou seja, `import java.text.*;`

2^a SOLUÇÃO:

```
\EXERC\CAP3\JAVA\EX2_B.java e \EXERC\CAP3\JAVA\EX2_B.class
```

3^a SOLUÇÃO:

\EXERC\CAP3\JAVA\EX2_C.java e \EXERC\CAP3\JAVA\EX2_C.class

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

ALGORITMO**1^a 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^a SOLUÇÃO:**ALGORITMO**

```
DECLARE nota1, nota2, nota3, peso1, peso2, peso3 NUMÉRICO
      soma1, soma2, soma3, total, media NUMÉRICO
LEIA nota1, nota2, nota3, peso1, peso2, peso3
soma1 ← nota1 * peso1
soma2 ← nota2 * peso2
soma3 ← nota3 * peso3
total ← peso1 + peso2 + peso3
media ← (soma1 + soma2 + soma3) / total
ESCREVA media
FIM_ALGORITMO.
```

**1^a SOLUÇÃO:**

\EXERC\CAP3\PASCAL\EX3_A.PAS e \EXERC\CAP3\PASCAL\EX3_A.EXE

2^a SOLUÇÃO:

\EXERC\CAP3\PASCAL\EX3_B.PAS e \EXERC\CAP3\PASCAL\EX3_B.EXE

3^a SOLUÇÃO:

\EXERC\CAP3\PASCAL\EX3_C.PAS e \EXERC\CAP3\PASCAL\EX3_C.EXE

**1^a SOLUÇÃO:**

\EXERC\CAP3\C++\EX3_A.CPP e \EXERC\CAP3\C++\EX3_A.EXE

2^a SOLUÇÃO:

\EXERC\CAP3\C++\EX3_B.CPP e \EXERC\CAP3\C++\EX3_B.EXE

3^a SOLUÇÃO:

\EXERC\CAP3\C++\EX3_C.CPP e \EXERC\CAP3\C++\EX3_C.EXE

**1^a SOLUÇÃO:**

\EXERC\CAP3\JAVA\EX3_A.java e \EXERC\CAP3\JAVA\EX3_A.class

2^a SOLUÇÃO:

```
\EXERC\CAP3\JAVA\EX3_B.java e \EXERC\CAP3\JAVA\EX3_B.class
```

3^a SOLUÇÃO:

```
\EXERC\CAP3\JAVA\EX3_C.java e \EXERC\CAP3\JAVA\EX3_C.class
```

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%.

ALGORITMO**1^a SOLUÇÃO:**

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

2^a SOLUÇÃO:**ALGORITMO**

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

**1^a SOLUÇÃO:**

```
\EXERC\CAP3\PASCAL\EX4_A.PAS e \EXERC\CAP3\PASCAL\EX4_A.EXE
```

2^a SOLUÇÃO:

```
\EXERC\CAP3\PASCAL\EX4_B.PAS e \EXERC\CAP3\PASCAL\EX4_B.EXE
```

**1^a SOLUÇÃO:**

```
\EXERC\CAP3\C++\EX4_A.CPP e \EXERC\CAP3\C++\EX4_A.EXE
```

2^a SOLUÇÃO:

```
\EXERC\CAP3\C++\EX4_B.CPP e \EXERC\CAP3\C++\EX4_B.EXE
```

**1^a SOLUÇÃO:**

```
\EXERC\CAP3\JAVA\EX4_A.java e \EXERC\CAP3\JAVA\EX4_A.class
```

2^a SOLUÇÃO:

```
\EXERC\CAP3\JAVA\EX4_B.java e \EXERC\CAP3\JAVA\EX4_B.class
```

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

**SOLUÇÃO:**

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

**SOLUÇÃO:**

\EXERC\CAP3\JAVA\EX5.java e \EXERC\CAP3\JAVA\EX5.class

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

ALGORITMO**SOLUÇÃO:**

```

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

**SOLUÇÃO:**

\EXERC\CAP3\JAVA\EX6.java e \EXERC\CAP3\JAVA\EX6.class

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

ALGORITMO**SOLUÇÃO:**

```

ALGORITMO
DECLARE sal, salreceiveber, imp NUMÉRICO
LEIA sal
imp ← sal * 10/100
salreceiveber ← sal + 50 - imp
ESCREVA salreceiveber
FIM_ALGORITMO.

```

**SOLUÇÃO:**

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

**SOLUÇÃO:**

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

**SOLUÇÃO:**

\EXERC\CAP3\JAVA\EX7.java e \EXERC\CAP3\JAVA\EX7.class

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.

ALGORITMO**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\CAP3\PASCAL\EX8.PAS e \EXERC\CAP3\PASCAL\EX8.EXE

**SOLUÇÃO:**

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

**SOLUÇÃO:**

\EXERC\CAP3\JAVA\EX8.java e \EXERC\CAP3\JAVA\EX8.class

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

Sabe-se que: Área = (base * altura)/2

ALGORITMO**SOLUÇÃO:**

```

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

SOLUÇÃO:

\EXERC\CAP3\JAVA\EX9.java e \EXERC\CAP3\JAVA\EX9.class

-  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 \leftarrow 3.1415 * raio²
 ESCREVA area
 FIM_ALGORITMO.

1^a SOLUÇÃO:

\EXERC\CAP3\PASCAL\EX10_A.PAS e \EXERC\CAP3\PASCAL\EX10_A.EXE

2^a SOLUÇÃO:

\EXERC\CAP3\PASCAL\EX10_B.PAS e \EXERC\CAP3\PASCAL\EX10_B.EXE

3^a SOLUÇÃO:

\EXERC\CAP3\PASCAL\EX10_C.PAS e \EXERC\CAP3\PASCAL\EX10_C.EXE

Esse programa usou algumas funções predefinidas da linguagem PASCAL que estão descritas na Seção 3.2.6.

1^a SOLUÇÃO:

\EXERC\CAP3\C++\EX10_A.CPP e \EXERC\CAP3\C++\EX10_A.EXE

2^a SOLUÇÃO:

\EXERC\CAP3\C++\EX10_B.CPP e \EXERC\CAP3\C++\EX10_B.EXE

3^a SOLUÇÃO:

\EXERC\CAP3\C++\EX10_C.CPP e \EXERC\CAP3\C++\EX10_C.EXE

Esse programa usou algumas funções predefinidas da linguagem C/C++ que estão descritas na Seção 3.3.6.

1^a SOLUÇÃO:

\EXERC\CAP3\JAVA\EX10_A.java e \EXERC\CAP3\JAVA\EX10_A.class

2^a SOLUÇÃO:

```
\EXERC\CAP3\JAVA\EX10_B.java e \EXERC\CAP3\JAVA\EX10_B.class
```

3^a SOLUÇÃO:

```
\EXERC\CAP3\JAVA\EX10_C.java e \EXERC\CAP3\JAVA\EX10_C.class
```

Esse programa usou algumas funções predefinidas da linguagem Java que estão descritas na Seção 3.4.6.

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

- a) o número digitado ao quadrado;
- b) o número digitado ao cubo;
- c) a raiz quadrada do número digitado;
- d) a raiz cúbica do número digitado.

ALGORITMO**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\CAP3\PASCAL\EX11.PAS e \EXERC\CAP3\PASCAL\EX11.EXE
```

Esse programa usou algumas funções predefinidas da linguagem PASCAL que estão descritas na Seção 3.2.6.

**SOLUÇÃO:**

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

Esse programa usou algumas funções predefinidas da linguagem C/C++ que estão descritas na Seção 3.3.6.

**SOLUÇÃO:**

```
\EXERC\CAP3\JAVA\EX11.java e \EXERC\CAP3\JAVA\EX11.class
```

Esse programa usou algumas funções predefinidas da linguagem JAVA que estão descritas na Seção 3.4.6.

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 ← num1num2
```

```
r2 ← num2num1
ESCREVA r1, r2
FIM_ALGORITMO.
```

**SOLUÇÃO:**

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

Esse programa usou algumas funções predefinidas da linguagem PASCAL que estão descritas na Seção 3.2.6.

**SOLUÇÃO:**

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

Esse programa usou algumas funções predefinidas da linguagem C/C++ que estão descritas na Seção 3.3.6.

**SOLUÇÃO:**

\EXERC\CAP3\JAVA\EX12.java e \EXERC\CAP3\JAVA\EX12.class

Esse programa usou algumas funções predefinidas da linguagem JAVA que estão descritas na Seção 3.4.6.

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.

- a) polegadas;
- b) jardas;
- c) milhas.

ALGORITMO**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\CAP3\PASCAL\EX13.PAS e \EXERC\CAP3\PASCAL\EX13.EXE

**SOLUÇÃO:**

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

**SOLUÇÃO:**

\EXERC\CAP3\JAVA\EX13.java e \EXERC\CAP3\JAVA\EX13.class

-  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 ela terá em 2050.

ALGORITMO**SOLUÇÃO:****ALGORITMO**

```
DECLARE ano_atual, ano_nascimento, idade_atual, idade_2050 NUMÉRICO
LEIA ano_atual
LEIA ano_nascimento
idade_atual ← ano_atual - ano_nascimento
idade_2050 ← 2050 - ano_nascimento
ESCREVA idade_atual
ESCREVA idade_2050
FIM_ALGORITMO.
```

**SOLUÇÃO:**

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

**SOLUÇÃO:**

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

**SOLUÇÃO:**

\EXERC\CAP3\JAVA\EX14.java e \EXERC\CAP3\JAVA\EX14.class

-  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.

ALGORITMO**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\CAP3\PASCAL\EX15.PAS e \EXERC\CAP3\PASCAL\EX15.EXE

**SOLUÇÃO:**

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

SOLUÇÃO:

\EXERC\CAP3\JAVA\EX15.java e \EXERC\CAP3\JAVA\EX15.class

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 estas regras:

- 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.

ALGORITMOSOLUÇÃO:

ALGORITMO

```

DECLARE horas_t, vlr_sal_min, vlr_hora_t NUMÉRICO
    vlr_sal_bru, imp, vlr_sal_liq NUMÉRICO
LEIA horas_t
LEIA vlr_sal_min
vlr_hora_t ← vlr_sal_min / 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\CAP3\PASCAL\EX16.PAS e \EXERC\CAP3\PASCAL\EX16.EXE

SOLUÇÃO:

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

SOLUÇÃO:

\EXERC\CAP3\JAVA\EX16.java e \EXERC\CAP3\JAVA\EX16.class

17. Um trabalhador recebeu seu salário e o depositou em sua conta 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.

ALGORITMOSOLUÇÃO:

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

**SOLUÇÃO:**

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

**SOLUÇÃO:**

```
\EXERC\CAP3\JAVA\EX17.java e \EXERC\CAP3\JAVA\EX17.class
```

18. Pedro comprou um saco de ração com peso em quilos. Ele possui dois gatos, para os quais fornece a quantidade de ração em gramas. A quantidade diária de ração fornecida para cada gato é sempre a mesma. 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.

ALGORITMO**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\CAP3\PASCAL\EX18.PAS e \EXERC\CAP3\PASCAL\EX18.EXE
```

**SOLUÇÃO:**

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

**SOLUÇÃO:**

```
\EXERC\CAP3\JAVA\EX18.java e \EXERC\CAP3\JAVA\EX18.class
```

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 ele deverá subir para atingir seu objetivo, sem se preocupar com a altura do usuário. Todas as medidas fornecidas devem estar em metros.

ALGORITMO**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\CAP3\PASCAL\EX19.PAS e \EXERC\CAP3\PASCAL\EX19.EXE
```

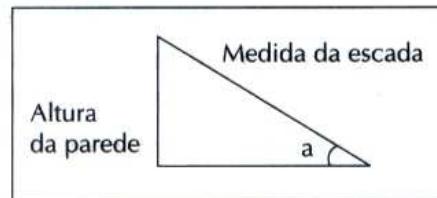
SOLUÇÃO:

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

SOLUÇÃO:

\EXERC\CAP3\JAVA\EX19.java e \EXERC\CAP3\JAVA\EX19.class

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

SOLUÇÃO:

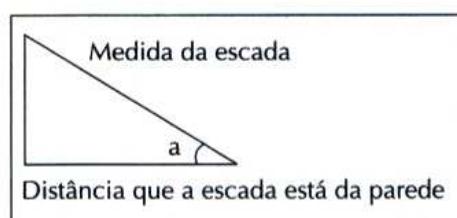
\EXERC\CAP3\C++\EX20.CPP e \EXERC\CAP3\C++\EX20.EXE

SOLUÇÃO:

\EXERC\CAP3\JAVA\EX20.java e \EXERC\CAP3\JAVA\EX20.class

21. Uma pessoa deseja pregar um quadro em uma parede. Faça um programa para calcular e mostrar a que distância a escada deve estar da parede. A pessoa 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 – Ramanho da escada

ALGORITMOSOLUÇÃO:

```

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

```

**SOLUÇÃO:**

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

**SOLUÇÃO:**

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

**SOLUÇÃO:**

\EXERC\CAP3\JAVA\EX21.java e \EXERC\CAP3\JAVA\EX21.class

- 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 de cada quilowatt;
 - o valor a ser pago por essa residência;
 - o valor 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\CAP3\PASCAL\EX22.PAS e \EXERC\CAP3\PASCAL\EX22.EXE

**SOLUÇÃO:**

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

**SOLUÇÃO:**

\EXERC\CAP3\JAVA\EX22.java e \EXERC\CAP3\JAVA\EX22.class

- 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 ← parte inteira de num
f ← num - i
a ← arredonda (num)
```

```

ESCREVA i
ESCREVA f
ESCREVA a
FIM_ALGORITMO.

```



SOLUÇÃO (ARREDONDANDO O NÚMERO COMO NA MATEMÁTICA):

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



1^a SOLUÇÃO (ARREDONDANDO O NÚMERO PARA CIMA):

\EXERC\CAP3\C++\EX23_A.CPP e \EXERC\CAP3\C++\EX23_A.EXE

2^a SOLUÇÃO (ARREDONDANDO O NÚMERO PARA BAIXO):

\EXERC\CAP3\C++\EX23_B.CPP e \EXERC\CAP3\C++\EX23_B.EXE



1^a SOLUÇÃO (ARREDONDANDO O NÚMERO PARA CIMA):

\EXERC\CAP3\JAVA\EX23_A.java e \EXERC\CAP3\JAVA\EX23_A.class

2^a SOLUÇÃO (ARREDONDANDO O NÚMERO PARA BAIXO):

\EXERC\CAP3\JAVA\EX23_B.java e \EXERC\CAP3\JAVA\EX23_B.class

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 59.



SOLUÇÃO:

```

ALGORITMO
DECLARE hora, h, m, conversao NUMÉRICO
LEIA hora
h ← pegar a parte inteira da variável hora
m ← hora - h
conversao ← (h * 60) + (m * 100)
ESCREVA conversao
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



SOLUÇÃO:

\EXERC\CAP3\JAVA\EX24.java e \EXERC\CAP3\JAVA\EX24.class

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

ALGORITMO**SOLUÇÃO:**

```

ALGORITMO
DECLARE custo, convite, qtd NUMÉRICO
LEIA custo
LEIA convite
qtd ← custo / convite
ESCREVA qtd
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

**SOLUÇÃO:**

\EXERC\CAP3\JAVA\EX25.java e \EXERC\CAP3\JAVA\EX25.class

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 e peso 3 para a segunda.
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 do funcionário e o valor de suas vendas, calcule e mostre a comissão e seu salário final.
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: $A = ((base\ maior + base\ menor) * altura)/2$
10. Faça um programa que calcule e mostre a área de um quadrado.
Sabe-se que: $A = lado * lado$

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 esse funcionário ganha.

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 atrasadas. Por causa do atraso, ele deverá pagar multa de 2% sobre cada conta. Faça um programa que calcule e mostre quanto restará do salário de João.

16. Faça um programa que receba o valor dos catetos de um triângulo, calcule e mostre o valor da hipotenusa.

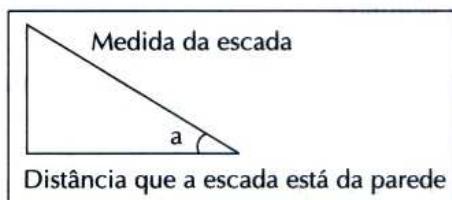
17. Faça um programa que receba o raio, calcule e mostre:

- a) o comprimento de uma esfera; sabe-se que $C = 2 * \pi * R$;
- b) a área de uma esfera; sabe-se que $A = \pi * R^2$;
- c) 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 em que a escada está da parede, calcule e mostre a medida da escada para que se possa alcançar sua ponta.



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 abaixo:

- a) a hora trabalhada vale $\frac{1}{8}$ do salário mínimo;
- b) a hora extra vale $\frac{1}{4}$ do salário mínimo;
- c) o salário bruto equivale ao número de horas trabalhadas multiplicado pelo valor da hora trabalhada;
- d) a quantia a receber pelas horas extras equivale ao número de horas extras trabalhadas multiplicado pelo valor da hora extra;
- e) 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. Sabe-se que $ND = N * (N - 3)/2$, onde N é o número de lados do polígono.

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 graus.

24. Faça um programa que receba a quantidade de dinheiro em reais que uma pessoa que vai viajar possui. Ela 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) a hora digitada convertida em minutos;
- b) o total dos minutos, ou seja, os minutos digitados mais a conversão anterior;
- c) o total dos minutos convertidos em segundos.

CAPÍTULO

4

ESTRUTURA CONDICIONAL

4.1 ESTRUTURA CONDICIONAL EM ALGORITMOS

4.1.1 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.

4.1.2 ESTRUTURA CONDICIONAL COMPOSTA

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

Se a condição for verdadeira, será executado o comando1; caso contrário, 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, o comando3 e o comando4 serão executados.

4.2 ESTRUTURA CONDICIONAL EM PASCAL

4.2.1 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.

4.2.2 ESTRUTURA CONDICIONAL COMPOSTA

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

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

OBSERVAÇÃO Antes do comando ELSE não existe ponto-e-vírgula.

4.2.3 ESTRUTURA CASE

Em alguns programas, existem situações mutuamente exclusivas, isto é, se uma situação for executada, as demais não serão. Quando for este o caso, um comando seletivo será o mais indicado, e esse comando, 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 atingir a lista de alvos2, o comando2 será executado; se atingir o alvo3, o comando3 será executado; se 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 atingir a lista de alvos2, o comando3 será executado. Se nenhum alvo for atingido, será executado o comando4.

Os alvos podem ser valores únicos, listas de valores (separados por vírgulas) ou faixas de valores, como no exemplo que se segue.

Exemplo:

```
program teste;
uses crt;
var i: integer;
begin
clrscr;
writeln('Digite um número');
readln(i);
case i of
    1: writeln('Número 1');
    2,5,6:writeln('Número 2 ou número 5 ou número 6');
    7..10:writeln('Número entre 7 e 10');
    else writeln('outro número');
end;
readln;
end.
```

A restrição da estrutura case é que o seletor só pode ser uma variável do tipo char, integer ou boolean.

4.2.4 OPERADORES LÓGICOS

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

TABELA E	TABELA OU	TABELA NÃO
V e V = V	V ou V = V	Não V = F
V e F = F	V ou F = V	Não F = V
F e V = F	F ou V = V	
F e F = F	F ou F = F	

OBSERVAÇÃO

Na linguagem PASCAL, quando existe mais de uma condição, elas devem estar entre parênteses.

Exemplos:

```
IF x = 3
THEN WRITELN('Número igual a 3');
```

No exemplo, existe apenas uma condição, logo, os parênteses são opcionais.

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

No exemplo anterior, existe mais de uma condição, logo, os parênteses são obrigatórios, ou seja, cada condição deve estar entre parênteses.

```
IF ((X = 5) AND (Y = 2)) OR (Y = 3)
THEN WRITELN('X é igual a 5 e Y é igual a 2, ou Y é igual a 3');
```

No exemplo acima, existe mais de uma condição e mais de um tipo de operador lógico, logo, além dos parênteses de cada condição, devem existir ainda parênteses que indiquem a prioridade de execução das condições. Nesse exemplo, as condições com o operador AND, ou seja, $((X = 5) \text{ AND } (Y = 2))$, serão testadas, e seu resultado será testado com a condição OR $(Y = 3)$.

```
IF (X = 5) AND ((Y = 2) OR (Y = 3))
THEN WRITELN('X é igual a 5, e Y é igual a 2 ou Y é igual a 3');
```

Neste exemplo, existe mais de uma condição e mais de um tipo de operador lógico, logo, além dos parênteses de cada condição, devem existir ainda parênteses que indiquem a prioridade de execução das condições. Aqui, as condições com o operador OR, ou seja, $((Y = 2) \text{ OR } (Y = 3))$, serão testadas, e seu resultado será testado com a condição AND $(X = 5)$.

4.3 ESTRUTURA CONDICIONAL EM C/C++

4.3.1 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 existe mais de um comando a executar. Os comandos entre chaves {} só serão executados se a condição for verdadeira.

4.3.2 ESTRUTURA CONDICIONAL COMPOSTA

```
if (condição)
    comando1;
else
    comando2;
```

Se a condição for verdadeira, será executado o comando1; se 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, o comando3 e o comando4 serão executados.

4.3.3 ESTRUTURA CASE

Em alguns programas, existem situações mutuamente exclusivas, isto é, se uma situação for executada, as demais não serão. Quando for este o caso, um comando seletivo é o mais indicado, e esse comando 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, que deve ser, obrigatoriamente, do tipo `char`, `unsigned char`, `int`, `unsigned int`, `short int`, `long` OU `unsigned long`.

O comando `break` deve ser utilizado para impedir a execução dos comandos definidos nos `cases` subsequentes.

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

```
#include <iostream.h>
#include <conio.h>
void main()
{
    int i;
    clrscr();
    cout<<"Digite um número ";
    cin>>i;
    switch (i)
    {
        case 1:cout<<"Número 1";
        break;
        case 2:cout<<"Número 2";
        break;
        default:cout<<"Número diferente de 1 e de 2";
    }
    getch();
}
```

4.3.4 OPERADORES LÓGICOS

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

TABELA E	TABELA OU	TABELA NÃO
V e V = V	V ou V = V	Não V = F
V e F = F	V ou F = V	Não F = V
F e V = F	F ou V = V	
F e F = F	F ou F = F	

Na linguagem C/C++, todas as condições devem estar entre parênteses.

Exemplos:

```
if (x == 3)
cout<<"Número igual a 3";
```

No exemplo acima, existe apenas uma condição que, obrigatoriamente, deve estar entre parênteses.

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

No exemplo acima, existe mais de uma condição, as quais, obrigatoriamente, devem estar entre parênteses.

```
if ((X == 5 && Y == 2) || Y == 3)
cout<<"X é igual a 5 e Y é igual a 2, ou Y é igual a 3";
```

No exemplo acima, existe mais de uma condição e mais de um tipo de operador lógico, logo, além dos parênteses que envolvem todas as condições, devem existir ainda parênteses que indiquem a prioridade de execução das condições. Aqui, as condições com o operador `&&`, ou seja, `(X == 5 && Y == 2)`, serão testadas, e seu resultado será testado com a condição `|| Y == 3`.

```
if (X == 5 && (Y == 2 || Y == 3))
cout<<"X é igual a 5, e Y é igual a 2 ou Y é igual a 3";
```

No exemplo acima, existe mais de uma condição e mais de um tipo de operador lógico, logo, além dos parênteses que envolvem todas as condições, devem existir ainda parênteses que indiquem a prioridade de execução das condições. Nesse exemplo, as condições com o operador `||`, ou seja, `(Y == 2 || Y == 3)`, serão testadas, e seu resultado será testado com a condição `&& X == 5`.

4.4 ESTRUTURA CONDICIONAL EM JAVA

4.4.1 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 JAVA, torna-se obrigatória a utilização de chaves quando existe mais de um comando a executar. Os comandos entre chaves `{ }` só serão executados se a condição for verdadeira.

4.4.2 ESTRUTURA CONDICIONAL COMPOSTA

```
if (condição)
    comando1;
else
    comando2;
```

Se a condição for verdadeira, será executado o `comando1`; caso contrário, será executado o `comando2`.

```
if condição
{
    comando1;
```

```

        comando2;
    }
else
{
    comando3;
    comando4;
}

```

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

4.4.3 ESTRUTURA CASE

Em alguns programas, existem situações mutuamente exclusivas, isto é, se uma situação for executada, as demais não serão. Quando for este o caso, um comando seletivo é o mais indicado, e esse comando, em JAVA, 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)` analisa o valor de uma variável para decidir qual `case` será executado.

Cada `case` está associado a UM possível valor da variável, que deve ser obrigatoriamente do tipo `int`, `short`, `byte` ou `char`.

O comando `break` deve ser utilizado para impedir a execução dos comandos definidos nos `cases` subsequentes.

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

Exemplo:

```

import java.io.*;
import java.util.*;
class teste
{
    public static void main(String args[])
    {
        int x;
        Scanner dado;

        System.out.println("Digite um número ");
        dado = new Scanner(System.in);
        x = dado.nextInt();
        switch (x)
        {
            case 1: System.out.println("Número 1");
                break;
            case 2: System.out.println("Número 2");
                break;
            default: System.out.println("Outro número");
        }
    }
}

```

4.4.4 OPERADORES LÓGICOS

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

TABELA E	TABELA OU	TABELA NÃO
V e V = V	V ou V = V	Não V = F
V e F = F	V ou F = V	Não F = V
F e V = F	F ou V = V	
F e F = F	F ou F = F	

Na linguagem JAVA, todas as condições devem estar entre parênteses.

Exemplos:

```
if (x == 3)
    System.out.println("Número igual a 3");
```

No exemplo acima, existe apenas uma condição que, obrigatoriamente, deve estar entre parênteses.

```
if (X > 5 && X < 10)
    System.out.println("Número entre 5 e 10");
```

No exemplo acima, existe mais de uma condição, as quais, obrigatoriamente, devem estar entre parênteses.

```
if ((X == 5 && Y == 2) || Y == 3)
    System.out.println("X é igual a 5 e Y é igual a 2, ou Y é igual a 3");
```

No exemplo acima, existe mais de uma condição e mais de um tipo de operador lógico, logo, além dos parênteses que envolvem todas as condições, devem existir ainda parênteses que indiquem a prioridade de execução das condições. Nesse exemplo, as condições com o operador `&&`, ou seja, `(X == 5 && Y == 2)`, serão testadas, e seu resultado será testado com a condição `|| Y == 3`.

```
if (X == 5 && (Y == 2 || Y == 3))
    System.out.println("X é igual a 5, e Y é igual a 2 ou Y é igual a 3");
```

No exemplo acima, existe mais de uma condição e mais de um tipo de operador lógico, logo, além dos parênteses que envolvem todas as condições, devem existir ainda parênteses que indiquem a prioridade de execução das condições. Nesse exemplo, as condições com o operador `||`, ou seja, `(Y == 2 || Y == 3)`, serão testadas, e seu resultado será testado com a condição `&& X == 5`.

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 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:

MÉDIA PONDERADA	CONCEITO
8,0 ●—● 10,0	A
7,0 ●—○ 8,0	B
6,0 ●—○ 7,0	C
5,0 ●—○ 6,0	D
0,0 ●—○ 5,0	E

ALGORITMO**SOLUÇÃO:**

```

ALGORITMO
DECLARE nota_trab, aval_sem, exame, media NUMÉRICO
ESCREVA "Digite a nota do trabalho de 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^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL SIMPLES:**

\EXERC\CAP4\PASCAL\EX1_A.PAS e \EXERC\CAP4\PASCAL\EX1_A.EXE

2^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:

\EXERC\CAP4\PASCAL\EX1_B.PAS e \EXERC\CAP4\PASCAL\EX1_B.EXE

**1^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL SIMPLES:**

\EXERC\CAP4\C++\EX1_A.CPP e \EXERC\CAP4\C++\EX1_A.EXE

2^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:

\EXERC\CAP4\C++\EX1_B.CPP e \EXERC\CAP4\C++\EX1_B.EXE

**1^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL SIMPLES:**

\EXERC\CAP4\JAVA\EX1_A.java e \EXERC\CAP4\JAVA\EX1_A.class

2^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:

\EXERC\CAP4\JAVA\EX1_B.java e \EXERC\CAP4\JAVA\EX1_B.class

2. Faça um programa que receba três notas de um aluno, calcule e mostre a média aritmética e a mensagem constante na tabela a seguir. Aos alunos que ficaram para exame, calcule e mostre a nota que deverão tirar para serem aprovados, considerando que a média exigida é 6,0.

MÉDIA ARITMÉTICA		MENSAGEM
0,0	● - ○	3,0
3,0	● - ○	7,0
7,0	● - ●	10,0
		Aprovado

ALGORITMO**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.

```

**1ª SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL SIMPLES:**

\EXERC\CAP4\PASCAL\EX2_A.PAS e \EXERC\CAP4\PASCAL\EX2_A.EXE

2ª SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:

\EXERC\CAP4\PASCAL\EX2_B.PAS e \EXERC\CAP4\PASCAL\EX2_B.EXE

**1ª SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL SIMPLES:**

\EXERC\CAP4\C++\EX2_A.CPP e \EXERC\CAP4\C++\EX2_A.EXE

2ª SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:

\EXERC\CAP4\C++\EX2_B.CPP e \EXERC\CAP4\C++\EX2_B.EXE

**1ª SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL SIMPLES:**

\EXERC\CAP4\JAVA\EX2_A.java e \EXERC\CAP4\JAVA\EX2_A.class

2ª SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:

\EXERC\CAP4\JAVA\EX2_B.java e \EXERC\CAP4\JAVA\EX2_B.class

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

ALGORITMO**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^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL SIMPLES:**

\EXERC\CAP4\PASCAL\EX3_A.PAS e \EXERC\CAP4\PASCAL\EX3_A.EXE

2^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:

\EXERC\CAP4\PASCAL\EX3_B.PAS e \EXERC\CAP4\PASCAL\EX3_B.EXE

**1^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL SIMPLES:**

\EXERC\CAP4\C++\EX3_A.CPP e \EXERC\CAP4\C++\EX3_A.EXE

2^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:

\EXERC\CAP4\C++\EX3_B.CPP e \EXERC\CAP4\C++\EX3_B.EXE

**1^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL SIMPLES:**

\EXERC\CAP4\JAVA\EX3_A.java e \EXERC\CAP4\JAVA\EX3_A.class

2^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:

\EXERC\CAP4\JAVA\EX3_B.java e \EXERC\CAP4\JAVA\EX3_B.class

-  4. Faça um programa que receba três números e mostre-os em ordem crescente. Suponha que o usuário digitará três números diferentes.

ALGORITMO**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^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL SIMPLES:**

\EXERC\CAP4\PASCAL\EX4_A.PAS e \EXERC\CAP4\PASCAL\EX4_A.EXE

2^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:

\EXERC\CAP4\PASCAL\EX4_B.PAS e \EXERC\CAP4\PASCAL\EX4_B.EXE

**1^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL SIMPLES:**

\EXERC\CAP4\C++\EX4_A.CPP e \EXERC\CAP4\C++\EX4_A.EXE

2^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:

\EXERC\CAP4\C++\EX4_B.CPP e \EXERC\CAP4\C++\EX4_B.EXE

**1^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL SIMPLES:**

\EXERC\CAP4\JAVA\EX4_A.java e \EXERC\CAP4\JAVA\EX4_A.class

2^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:

\EXERC\CAP4\JAVA\EX4_B.java e \EXERC\CAP4\JAVA\EX4_B.class

5. Faça um programa que receba três números obrigatoriamente em ordem crescente e um quarto número que não siga essa regra. Mostre, em seguida, os quatro números em ordem decrescente. Suponha que o usuário digitará quatro números diferentes.

ALGORITMO**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^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL SIMPLES:**

\EXERC\CAP4\PASCAL\EX5_A.PAS e \EXERC\CAP4\PASCAL\EX5_A.EXE

2^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:

\EXERC\CAP4\PASCAL\EX5_B.PAS e \EXERC\CAP4\PASCAL\EX5_B.EXE

**1^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL SIMPLES:**

\EXERC\CAP4\C++\EX5_A.CPP e \EXERC\CAP4\C++\EX5_A.EXE

2^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:

\EXERC\CAP4\C++\EX5_B.CPP e \EXERC\CAP4\C++\EX5_B.EXE

**1^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL SIMPLES:**

\EXERC\CAP4\JAVA\EX5_A.java e \EXERC\CAP4\JAVA\EX5_A.class

2^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:

\EXERC\CAP4\JAVA\EX5_B.java e \EXERC\CAP4\JAVA\EX5_B.class

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

ALGORITMO**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.

```

**1^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL SIMPLES:**

\EXERC\CAP4\PASCAL\EX6_A.PAS e \EXERC\CAP4\PASCAL\EX6_A.EXE

2^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:

\EXERC\CAP4\PASCAL\EX6_B.PAS e \EXERC\CAP4\PASCAL\EX6_B.EXE

**1^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL SIMPLES:**

\EXERC\CAP4\C++\EX6_A.CPP e \EXERC\CAP4\C++\EX6_A.EXE

2^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:

\EXERC\CAP4\C++\EX6_B.CPP e \EXERC\CAP4\C++\EX6_B.EXE

**1^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL SIMPLES:**

\EXERC\CAP4\JAVA\EX6_A.java e \EXERC\CAP4\JAVA\EX6_A.class

**2^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:**

\EXERC\CAP4\JAVA\EX6_B.java e \EXERC\CAP4\JAVA\EX6_B.class

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

Suponha que o valor digitado para I seja sempre um valor válido, ou seja, 1, 2 ou 3, e que os números digitados sejam diferentes um do outro.

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.

ALGORITMO**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<C
            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>C
            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^ª SOLUÇÃO:**

\EXERC\CAP4\PASCAL\EX7_A.PAS e \EXERC\CAP4\PASCAL\EX7_A.EXE

2^ª SOLUÇÃO:

\EXERC\CAP4\PASCAL\EX7_B.PAS e \EXERC\CAP4\PASCAL\EX7_B.EXE

3^ª SOLUÇÃO:

\EXERC\CAP4\PASCAL\EX7_C.PAS e \EXERC\CAP4\PASCAL\EX7_C.EXE

**1^ª SOLUÇÃO:**

\EXERC\CAP4\C++\EX7_A.CPP e \EXERC\CAP4\C++\EX7_A.EXE

2^ª SOLUÇÃO:

\EXERC\CAP4\C++\EX7_B.CPP e \EXERC\CAP4\C++\EX7_B.EXE

3^ª SOLUÇÃO:

\EXERC\CAP4\C++\EX7_C.CPP e \EXERC\CAP4\C++\EX7_C.EXE

**1^ª SOLUÇÃO:**

\EXERC\CAP4\JAVA\EX7_A.java e \EXERC\CAP4\JAVA\EX7_A.class

2^ª SOLUÇÃO:

\EXERC\CAP4\JAVA\EX7_B.java e \EXERC\CAP4\JAVA\EX7_B.class

3^ª SOLUÇÃO:

\EXERC\CAP4\JAVA\EX7_C.java e \EXERC\CAP4\JAVA\EX7_C.class

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.

ALGORITMO

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^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL SIMPLES:

\EXERC\CAP4\PASCAL\EX8_A.PAS e \EXERC\CAP4\PASCAL\EX8_A.EXE

2^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:

\EXERC\CAP4\PASCAL\EX8_B.PAS e \EXERC\CAP4\PASCAL\EX8_B.EXE

3^a SOLUÇÃO – UTILIZANDO ESTRUTURA SELETORA:

\EXERC\CAP4\PASCAL\EX8_C.PAS e \EXERC\CAP4\PASCAL\EX8_C.EXE



1^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL SIMPLES:

\EXERC\CAP4\C++\EX8_A.CPP e \EXERC\CAP4\C++\EX8_A.EXE

2^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:

\EXERC\CAP4\C++\EX8_B.CPP e \EXERC\CAP4\C++\EX8_B.EXE

3^a SOLUÇÃO – UTILIZANDO ESTRUTURA SELETORA:

\EXERC\CAP4\C++\EX8_C.CPP e \EXERC\CAP4\C++\EX8_C.EXE

**1^ª SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL SIMPLES:**

\EXERC\CAP4\JAVA\EX8_A.java e \EXERC\CAP4\JAVA\EX8_A.class

2^ª SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:

\EXERC\CAP4\JAVA\EX8_B.java e \EXERC\CAP4\JAVA\EX8_B.class

3^ª SOLUÇÃO – UTILIZANDO ESTRUTURA SELETORA:

\EXERC\CAP4\JAVA\EX8_C.java e \EXERC\CAP4\JAVA\EX8_C.class

9. Faça um programa que mostre a data e a hora do sistema nos seguintes formatos: DD/MM/AAAA – mês por extenso e hora:minuto.

ALGORITMO**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"
t ← OBTENHA_HORÁRIO;
hora ← OBTENHA_HORA(t)
min ← OBTENHA_MINUTO(t)
ESCREVA "Hora Atual: "
ESCREVA hora, ":" , min
FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP4\PASCAL\EX9.PAS e \EXERC\CAP4\PASCAL\EX9.EXE

Na solução com a linguagem PASCAL foram utilizados os comandos `getdate` e `gettime`, para obter a data e a hora do sistema operacional, respectivamente. O comando `getdate` retorna os valores do ano, mês, dia do mês e dia da semana da data do sistema operacional, e as variáveis que receberão esses valores devem ser do tipo `word`. O comando `gettime` retorna os valores da hora, minuto, segundo e centésimo de segundo da hora do sistema operacional e as variáveis que receberão esses valores devem ser do tipo `word`. Para a utilização dos comandos `getdate` e `gettime` faz-se necessária a utilização da biblioteca `DOS`, ou seja, `USES DOS`.

OBSERVAÇÃO

O dia da semana é um número em que domingo vale 0; segunda-feira, 1; terça-feira, 2; quarta-feira, 3; quinta-feira, 4; sexta-feira, 5; e sábado, 6.

Exemplo:

```
GETDATE(ano, mes, dia, dia_semana);
GETTIME(hora, min, seg, cen_seg);
```

**SOLUÇÃO:**

\EXERC\CAP4\C++\EX9.CPP e \EXERC\CAP4\C++\EX9.EXE

Na solução com a linguagem C/C++ foram utilizados os comandos `getdate()` e `gettime()`, para obter a data e a hora do sistema operacional, respectivamente. O comando `getdate()` retorna os valores do ano, mês e dia da data do sistema operacional. O comando `gettime()` retorna os valores da hora e minutos da hora do sistema operacional. Para a utilização dos comandos `getdate()` e `gettime()` faz-se necessária a utilização da biblioteca `time.h`, ou seja, `#include <time.h>`.

Exemplo:

```
struct time t;
struct date d;
getdate(&d);
dia = d.da_day;
mes = d.da_mon;
ano = d.da_year;
gettime(&t);
hora = t.ti_hour;
minuto = t.ti_min;
```

**SOLUÇÃO:**

\EXERC\CAP4\JAVA\EX9.java e \EXERC\CAP4\JAVA\EX9.class

Na solução com a linguagem JAVA foram utilizadas as classes `Calendar` e `Date` para empregar a data e a hora do sistema operacional.

Exemplo:

```
int dia, mes, ano, hora, min;
Calendar cal = Calendar.getInstance();
Date d = new Date();
cal.setTime(d);
dia = cal.get(Calendar.DAY_OF_MONTH);
mes = cal.get(Calendar.MONTH) + 1;
ano = cal.get(Calendar.YEAR);
hora = cal.get(Calendar.HOUR);
min = cal.get(Calendar.MINUTE);
```

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

ALGORITMO**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\CAP4\PASCAL\EX10.PAS e \EXERC\CAP4\PASCAL\EX10.EXE

**SOLUÇÃO:**

\EXERC\CAP4\C++\EX10.CPP e \EXERC\CAP4\C++\EX10.EXE

**SOLUÇÃO:**

\EXERC\CAP4\JAVA\EX10.java e \EXERC\CAP4\JAVA\EX10.class

11. Faça um programa que receba a hora do início de um jogo e a hora final (cada hora é composta por duas variáveis inteiras: 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 ele pode iniciar-se em um dia e terminar no dia seguinte.

ALGORITMO**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\CAP4\PASCAL\EX11.PAS e \EXERC\CAP4\PASCAL\EX11.EXE

**SOLUÇÃO:**

\EXERC\CAP4\C++\EX11.CPP e \EXERC\CAP4\C++\EX11.EXE

**SOLUÇÃO:**

\EXERC\CAP4\JAVA\EX11.java e \EXERC\CAP4\JAVA\EX11.class

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 abaixo.

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
        ESCRIVA "O cargo é Secretário"
        aumento ← salario * 35 / 100
        ESCRIVA "O valor do aumento é: ", aumento
        novo_sal ← salario + aumento
        ESCRIVA "O novo salário é: ", novo_sal
        FIM
SENÃO SE cargo = 3
    ENTÃO INÍCIO
        ESCRIVA "O cargo é Caixa"
        aumento ← salario * 20 / 100
        ESCRIVA "O valor do aumento é: ", aumento
        novo_sal ← salario + aumento
        ESCRIVA "O novo salário é: ", novo_sal
        FIM
SENÃO SE cargo = 4
    ENTÃO INÍCIO
        ESCRIVA "O cargo é Gerente"
        aumento ← salario * 10 / 100
        ESCRIVA "O valor do aumento é: ",
        ↪ aumento
        novo_sal ← salario + aumento
        ESCRIVA "O novo salário é: ",
        ↪ novo_sal
        FIM
SENÃO SE cargo = 5
    ENTÃO INÍCIO
        ESCRIVA "O cargo é Diretor"
        aumento ← salario * 0 / 100
        ESCRIVA "O valor do aumento
        ↪ é: ", aumento
        novo_sal ← salario +
        ↪ aumento
        ESCRIVA "O novo salário é:
        ↪ ", novo_sal
        FIM
SENÃO ESCRIVA "Cargo Inexistente !"
FIM_ALGORITMO.

```



1^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:

\EXERC\CAP4\PASCAL\EX12_A.PAS e \EXERC\CAP4\PASCAL\EX12_A.EXE

2^a SOLUÇÃO – UTILIZANDO ESTRUTURA SELETORA:

\EXERC\CAP4\PASCAL\EX12_B.PAS e \EXERC\CAP4\PASCAL\EX12_B.EXE



1^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:

\EXERC\CAP4\C++\EX12_A.CPP e \EXERC\CAP4\C++\EX12_A.EXE

2^a SOLUÇÃO – UTILIZANDO ESTRUTURA SELETORA:

\EXERC\CAP4\C++\EX12_B.CPP e \EXERC\CAP4\C++\EX12_B.EXE

**1^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:**

\EXERC\CAP4\JAVA\EX12_A.java e \EXERC\CAP4\JAVA\EX12_A.class

2^a SOLUÇÃO – UTILIZANDO ESTRUTURA SELETORA:

\EXERC\CAP4\JAVA\EX12_B.java e \EXERC\CAP4\JAVA\EX12_B.class

13. Faça um programa que apresente o menu 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:

SALÁRIO	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
Maior 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
Menor 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**SOLUÇÃO:**

```

ALGORITMO
DECLARE op, sal, imp, aum, novo_sal NUMÉRICO
LEIA op
SE op = 1
ENTÃO INÍCIO
LEIA sal
SE sal < 500
ENTÃO imp ← sal * 5/100

```

```

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"
SE sal > 700
    ENTÃO ESCREVA "Bem Remunerado"
FIM

SE op < 1 OU op > 3
    ENTÃO ESCREVA "Opção Inválida"
FIM_ALGORITMO.

```



1^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL SIMPLES:

\EXERC\CAP4\PASCAL\EX13_A.PAS e \EXERC\CAP4\PASCAL\EX13_A.EXE

2^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:

\EXERC\CAP4\PASCAL\EX13_B.PAS e \EXERC\CAP4\PASCAL\EX13_B.EXE

3^a SOLUÇÃO – UTILIZANDO ESTRUTURA SELETORA:

\EXERC\CAP4\PASCAL\EX13_C.PAS e \EXERC\CAP4\PASCAL\EX13_C.EXE



1^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL SIMPLES:

\EXERC\CAP4\C++\EX13_A.CPP e \EXERC\CAP4\C++\EX13_A.EXE

2^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:

\EXERC\CAP4\C++\EX13_B.CPP e \EXERC\CAP4\C++\EX13_B.EXE

3^a SOLUÇÃO – UTILIZANDO ESTRUTURA SELETORA:

\EXERC\CAP4\C++\EX13_C.CPP e \EXERC\CAP4\C++\EX13_C.EXE



1^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL SIMPLES:

\EXERC\CAP4\JAVA\EX13_A.java e \EXERC\CAP4\JAVA\EX13_A.class

2^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:

```
\EXERC\CAP4\JAVA\EX13_B.java e \EXERC\CAP4\JAVA\EX13_B.class
```

3^a SOLUÇÃO – UTILIZANDO ESTRUTURA SELETORA:

```
\EXERC\CAP4\JAVA\EX13_C.java e \EXERC\CAP4\JAVA\EX13_C.class
```

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

SALÁRIO	BONIFICAÇÃO
Até R\$ 500,00	5% do salário
Entre R\$ 500,00 e R\$ 1.200,00	12% do salário
Acima de R\$ 1.200,00	Sem bonificação

SALÁRIO	AUXÍLIO ESCOLA
Até R\$ 600,00	R\$ 150,00
Mais que R\$ 600,00	R\$ 100,00

ALGORITMO**SOLUÇÃO:**

```

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

**1^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL SIMPLES:**

```
\EXERC\CAP4\PASCAL\EX14_A.PAS e \EXERC\CAP4\PASCAL\EX14_A.EXE
```

2^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:

```
\EXERC\CAP4\PASCAL\EX14_B.PAS e \EXERC\CAP4\PASCAL\EX14_B.EXE
```

**1^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL SIMPLES:**

```
\EXERC\CAP4\C++\EX14_A.CPP e \EXERC\CAP4\C++\EX14_A.EXE
```

2^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:

```
\EXERC\CAP4\C++\EX14_B.CPP e \EXERC\CAP4\C++\EX14_B.EXE
```

**1^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL SIMPLES:**

```
\EXERC\CAP4\JAVA\EX14_A.java e \EXERC\CAP4\JAVA\EX14_A.class
```

2^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:

```
\EXERC\CAP4\JAVA\EX14_B.java e \EXERC\CAP4\JAVA\EX14_B.class
```

 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:

IRRF	SALÁRIO BRUTO
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:

SALÁRIO LÍQUIDO	GRATIFICAÇÃO
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 * 10/100
SE sbruto > 500
  ENTÃO imp ← sbruto * 20/100
sliq ← sbruto - imp
SE sliq <= 350
  ENTÃO grat ← 100
SE sliq > 350
  ENTÃO grat ← 50
sal_receber ← sliq + grat
ESCREVA sal_receber
FIM_ALGORITMO.

```

**1^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL SIMPLES:**

\EXERC\CAP4\PASCAL\EX15_A.PAS e \EXERC\CAP4\PASCAL\EX15_A.EXE

2^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:

\EXERC\CAP4\PASCAL\EX15_B.PAS e \EXERC\CAP4\PASCAL\EX15_B.EXE

**1^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL SIMPLES:**

\EXERC\CAP4\C++\EX15_A.CPP e \EXERC\CAP4\C++\EX15_A.EXE

2^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:

\EXERC\CAP4\C++\EX15_B.CPP e \EXERC\CAP4\C++\EX15_B.EXE

**1^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL SIMPLES:**

\EXERC\CAP4\JAVA\EX15_A.java e \EXERC\CAP4\JAVA\EX15_A.class

2^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:

\EXERC\CAP4\JAVA\EX15_B.java e \EXERC\CAP4\JAVA\EX15_B.class

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

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 média mensal do produto, calcule e mostre o novo preço.

ALGORITMO**SOLUÇÃO:**

```

ALGORITMO
DECLARE pre, venda, novo_pre NUMÉRICO
LEIA pre, venda
SE venda<500 OU pre<30
ENTÃO novo_pre ← pre + 10/100 * pre
SENÃO SE (venda>=500 E venda<1200) OU (pre>=30 E pre<80)
    ENTÃO novo_pre ← pre + 15/100 * pre
    SENÃO SE venda>=1200 OU pre>=80
        ENTÃO novo_pre ← pre - 20/100 * pre
ESCREVA novo_pre
FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP4\PASCAL\EX16.PAS e \EXERC\CAP4\PASCAL\EX16.EXE

SOLUÇÃO:

\EXERC\CAP4\C++\EX16.CPP e \EXERC\CAP4\C++\EX16.EXE

SOLUÇÃO:

\EXERC\CAP4\JAVA\EX16.java e \EXERC\CAP4\JAVA\EX16.class

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

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

A variável a deve ser diferente de zero.

$$\Delta = b^2 - 4 * a * c$$

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

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

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

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

$$x_1 = (-b + \sqrt{\Delta}) / (2 * a)$$

$$x_2 = (-b - \sqrt{\Delta}) / (2 * a)$$

ALGORITMOSOLUÇÃO:

```

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 + √Δ) / (2 * a)
                    x2 ← (-b - √Δ) / (2 * a)
                    ESCREVA x1, x2
                    FIM
            FIM
    FIM_ALGORITMO.

```

SOLUÇÃO:

\EXERC\CAP4\PASCAL\EX17.PAS e \EXERC\CAP4\PASCAL\EX17.EXE

SOLUÇÃO:

\EXERC\CAP4\C++\EX17.CPP e \EXERC\CAP4\C++\EX17.EXE

SOLUÇÃO:

\EXERC\CAP4\JAVA\EX17.java e \EXERC\CAP4\JAVA\EX17.class

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

- ◆ O comprimento de cada lado de um triângulo é menor do que a soma dos outros dois lados.
- ◆ Chama-se eqüilátero o triângulo que tem três lados iguais.
- ◆ Denomina-se isósceles o triângulo que tem o comprimento de dois lados iguais.
- ◆ Recebe o nome de 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\CAP4\PASCAL\EX18.PAS e \EXERC\CAP4\PASCAL\EX18.EXE

**SOLUÇÃO:**

\EXERC\CAP4\C++\EX18.CPP e \EXERC\CAP4\C++\EX18.EXE

**SOLUÇÃO:**

\EXERC\CAP4\JAVA\EX18.java e \EXERC\CAP4\JAVA\EX18.class

 19. Faça um programa que receba a altura e o peso de uma pessoa. De acordo com a tabela a seguir, verifique e mostre 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

ALGORITMO**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^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL SIMPLES:

\EXERC\CAP4\PASCAL\EX19_A.PAS e \EXERC\CAP4\PASCAL\EX19_A.EXE

2^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:

\EXERC\CAP4\PASCAL\EX19_B.PAS e \EXERC\CAP4\PASCAL\EX19_B.EXE



1^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL SIMPLES:

\EXERC\CAP4\C++\EX19_A.CPP e \EXERC\CAP4\C++\EX19_A.EXE

2^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:

\EXERC\CAP4\C++\EX19_B.CPP e \EXERC\CAP4\C++\EX19_B.EXE



1^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL SIMPLES:

\EXERC\CAP4\JAVA\EX19_A.java e \EXERC\CAP4\JAVA\EX19_A.class

2^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:

\EXERC\CAP4\JAVA\EX19_B.java e \EXERC\CAP4\JAVA\EX19_B.class



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 seja sempre válida, ou seja, um número inteiro entre 1 e 3.

Tabelas:

CÓDIGO DO PAÍS DE ORIGEM	IMPOSTO
1	0%
2	15%
3	25%

CÓDIGO DO PRODUTO	PREÇO POR GRAMA
1 a 4	10
5 a 7	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 ele é cobrado sobre o preço total do produto comprado e depende do país de origem.
- ◆ O valor total, preço total do produto mais imposto.

ALGORITMO

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 * 15/100
SE cod_pais = 3
  ENTÃO imposto ← pre_total * 25/100
ESCREVA imposto
valor_total ← pre_total + imposto
ESCREVA valor_total
FIM ALGORITMO.

```



1ª SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL SIMPLES:

\EXERC\CAP4\PASCAL\EX20_A.PAS e \EXERC\CAP4\PASCAL\EX20_A.EXE

2ª SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:

\EXERC\CAP4\PASCAL\EX20_B.PAS e \EXERC\CAP4\PASCAL\EX20_B.EXE

3ª SOLUÇÃO – UTILIZANDO ESTRUTURA SELETORA:

\EXERC\CAP4\PASCAL\EX20_C.PAS e \EXERC\CAP4\PASCAL\EX20_C.EXE



1ª SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL SIMPLES:

\EXERC\CAP4\C++\EX20_A.CPP e \EXERC\CAP4\C++\EX20_A.EXE

2^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:

\EXERC\CAP4\C++\EX20_B.CPP e \EXERC\CAP4\C++\EX20_B.EXE

3^a SOLUÇÃO – UTILIZANDO ESTRUTURA SELETORA:

\EXERC\CAP4\C++\EX20_C.CPP e \EXERC\CAP4\C++\EX20_C.EXE

**1^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL SIMPLES:**

\EXERC\CAP4\JAVA\EX20_A.java e \EXERC\CAP4\JAVA\EX20_A.class

2^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:

\EXERC\CAP4\JAVA\EX20_B.java e \EXERC\CAP4\JAVA\EX20_B.class

3^a SOLUÇÃO – UTILIZANDO ESTRUTURA SELETORA:

\EXERC\CAP4\JAVA\EX20_C.java e \EXERC\CAP4\JAVA\EX20_C.class

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 seja sempre válida, ou seja, um número inteiro entre 10 e 40.

Tabelas:

CÓDIGO DO ESTADO	IMPOSTO
1	35%
2	25%
3	15%
4	5%
5	Isento

CÓDIGO DA CARGA	PREÇO POR QUILO
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 depende do estado de origem.
- ◆ O valor total transportado pelo caminhão, preço da carga mais imposto.

ALGORITMO**SOLUÇÃO:****ALGORITMO**

```

DECLARE cod_est, cod_carga, peso_toneladas NUMÉRICO
      peso_quilos 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 ← 35/100 * pre_carga
SE cod_est = 2
  ENTÃO imposto ← 25/100 * pre_carga
SE cod_est = 3
  ENTÃO imposto ← 15/100 * pre_carga
SE cod_est = 4
  ENTÃO imposto ← 5/100 * pre_carga
SE cod_est = 5
  ENTÃO imposto ← 0
ESCREVA imposto
valor_total ← pre_carga + imposto
ESCREVA valor_total
FIM_ALGORITMO.

```

**1^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL SIMPLES:**

\EXERC\CAP4\PASCAL\EX21_A.PAS e \EXERC\CAP4\PASCAL\EX21_A.EXE

2^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:

\EXERC\CAP4\PASCAL\EX21_B.PAS e \EXERC\CAP4\PASCAL\EX21_B.EXE

3^a SOLUÇÃO – UTILIZANDO ESTRUTURA SELETORA:

\EXERC\CAP4\PASCAL\EX21_C.PAS e \EXERC\CAP4\PASCAL\EX21_C.EXE

**1^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL SIMPLES:**

\EXERC\CAP4\C++\EX21_A.CPP e \EXERC\CAP4\C++\EX21_A.EXE

2^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:

\EXERC\CAP4\C++\EX21_B.CPP e \EXERC\CAP4\C++\EX21_B.EXE .

3^a SOLUÇÃO – UTILIZANDO ESTRUTURA SELETORA:

\EXERC\CAP4\C++\EX21_C.CPP e \EXERC\CAP4\C++\EX21_C.EXE

**1^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL SIMPLES:**

\EXERC\CAP4\JAVA\EX21_A.java e \EXERC\CAP4\JAVA\EX21_A.class

2^a SOLUÇÃO – UTILIZANDO ESTRUTURA CONDICIONAL COMPOSTA:

\EXERC\CAP4\JAVA\EX21_B.java e \EXERC\CAP4\JAVA\EX21_B.class

3^a SOLUÇÃO – UTILIZANDO ESTRUTURA SELETORA:

\EXERC\CAP4\JAVA\EX21_C.java e \EXERC\CAP4\JAVA\EX21_C.class

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

- ◆ O imposto, apresentado 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\$ 450,00 e R\$ 700,00	8%
=> R\$ 700,00	12%

- ◆ A gratificação, que se encontra na tabela abaixo.

SALÁRIO BASE	TEMPO DE SERVIÇO	GRATIFICAÇÃO
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.

SALÁRIO LÍQUIDO	CLASSIFICAÇÃO
Até R\$ 350,00	A
Entre R\$ 350,00 e R\$ 600,00	B
De R\$ 600,00 para cima	C

ALGORITMO

SOLUÇÃO:

```

ALGORITMO
DECLARE sal_base, tempo, imposto, grat NUMÉRICO
    sal_liq NUMÉRICO
LEIA sal_base, tempo
SE sal_base < 200
ENTÃO imposto ← 0
SENÃO SE sal_base <= 450
    ENTÃO imposto ← 3/100 * sal_base
    SENÃO SE sal_base < 700
        ENTÃO imposto ← 8/100 * sal_base
        SENÃO imposto ← 12/100 * 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\CAP4\PASCAL\EX22.PAS e \EXERC\CAP4\PASCAL\EX22.EXE

**SOLUÇÃO:**

\EXERC\CAP4\C++\EX22.CPP e \EXERC\CAP4\C++\EX22.EXE

**SOLUÇÃO:**

\EXERC\CAP4\JAVA\EX22.java e \EXERC\CAP4\JAVA\EX22.class

-  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. Suponha a digitação apenas de dados válidos e, quando houver digitação de letras, utilize 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 que se seguem.

Se o funcionário preencher **todos** os requisitos a seguir, 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 seguintes regras.

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 NUMÉRICO
        imposto, grat, auxilio, sal_liq NUMÉRICO
        turno, categoria LITERAL
LEIA sal_min, turno, categoria, nht
SE turno = "M"
    ENTÃO coeficiente ← 10/100 * sal_min
SE turno = "V"
    ENTÃO coeficiente ← 15/100 * sal_min
SE turno = "N"
    ENTÃO coeficiente ← 12/100 * sal_min
ESCREVA coeficiente
sal_bruto ← nht * coeficiente
ESCREVA sal_bruto
SE categoria = "O"
    ENTÃO INÍCIO
        SE sal_bruto >= 300
            ENTÃO imposto ← 5/100 * sal_bruto
            SENÃO imposto ← 3/100 * sal_bruto
        FIM
    SENÃO INÍCIO
        SE sal_bruto >= 400
            ENTÃO imposto ← 6/100 * sal_bruto
            SENÃO imposto ← 4/100 * sal_bruto
        FIM
    ESCREVA 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 ESCREVA "Mal Remunerado"
SE sal_liq >= 350 E sal_liq <= 600
  ENTÃO ESCREVA "Normal"
SE sal_liq > 600
  ENTÃO ESCREVA "Bem Remunerado"
FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP4\PASCAL\EX23.PAS e \EXERC\CAP4\PASCAL\EX23.EXE

**SOLUÇÃO:**

\EXERC\CAP4\C++\EX23.CPP e \EXERC\CAP4\C++\EX23.EXE

**SOLUÇÃO:**

\EXERC\CAP4\JAVA\EX23.java e \EXERC\CAP4\JAVA\EX23.class

- 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 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
		>= R\$ 10,00	R\$ 2,50
	V	< R\$ 30,00	R\$ 3,00
		>= 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 de custo 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 ← 5/100 * pre
  SENÃO imposto ← 8/100 * pre
ESCREVA imposto

```

```

pre_custo ← pre + imposto
ESCREVA pre_custo
SE tipo ≠ "A" E refrig ≠ "S"
    ENTÃO desconto ← 3/100 * pre_custo
    SENÃO desconto ← 0
ESCREVA desconto
novo_pre ← pre_custo + 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\CAP4\PASCAL\EX24.PAS e \EXERC\CAP4\PASCAL\EX24.EXE

**SOLUÇÃO:**

\EXERC\CAP4\C++\EX24.CPP e \EXERC\CAP4\C++\EX24.EXE

**SOLUÇÃO:**

\EXERC\CAP4\JAVA\EX24.java e \EXERC\CAP4\JAVA\EX24.class

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° ou menores que -360° , 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 "1° Quadrante"
SE (angulo > 90 E angulo < 180) OU (angulo < -180 E angulo > -270)
    ENTÃO ESCREVA "2° Quadrante"
SE (angulo > 180 E angulo < 270) OU (angulo < -90 E angulo > -180)
    ENTÃO ESCREVA "3° Quadrante"
SE (angulo > 270 E angulo < 360) OU (angulo < 0 E angulo > -90)
    ENTÃO ESCREVA "4° Quadrante"
ESCREVA voltas, " volta(s) no sentido "
SE angulo < 0
    ENTÃO ESCREVA "horário"
    SENÃO ESCREVA "anti-horário"
FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP4\PASCAL\EX25.PAS e \EXERC\CAP4\PASCAL\EX25.EXE

**SOLUÇÃO:**

\EXERC\CAP4\C++\EX25.CPP e \EXERC\CAP4\C++\EX25.EXE

**SOLUÇÃO:**

\EXERC\CAP4\JAVA\EX25.java e \EXERC\CAP4\JAVA\EX25.class

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 se encontra na tabela a seguir:

MÉDIA ARITMÉTICA	MENSAGEM
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, mostre uma mensagem de erro e termine 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, mostre mensagem de erro e termine a execução do programa. As opções são:

1. O primeiro número elevado ao segundo número.
2. Raiz quadrada de cada um dos números.
3. Raiz cúbica de cada um dos 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 ele não tenha direito ao aumento.

8. Faça um programa para calcular e mostrar o salário reajustado de um funcionário. O percentual de aumento encontra-se na 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
R\$ 300,00	25% do saldo médio
R\$ 200,00	20% do saldo médio
Até R\$ 200,00	10% do saldo médio

10. O preço, 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. As porcentagens encontram-se na tabela a seguir. Faça um programa que receba o custo de fábrica de um carro e mostre o preço 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 atual 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
R\$ 600,00	10%
R\$ 900,00	5%
Acima de R\$ 900,00	0%

12. Faça um programa que receba o salário bruto 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\$ 75,00
R\$ 600,00	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 ━━━━ R\$ 500,00	40%
R\$ 500,00 ━━━━ R\$ 700,00	30%
R\$ 700,00 ━━━━ R\$ 800,00	20%
R\$ 800,00 ━━━━ 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 seu valor 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 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 maioridade ou não.

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

- ◆ 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 sua categoria, usando as regras a seguir. Para idade inferior a 5, deverá mostrar mensagem.

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 seu código de origem e mostre 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 que 20	9	8	7
De 20 a 50	6	5	4
Maiores que 50	3	2	1

23. Faça um programa que receba:

- ◆ O código do produto comprado;
- ◆ A quantidade comprada do 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

PREÇO TOTAL DA NOTA	% DE DESCONTO
Até R\$ 250,00	5%
Entre R\$ 250,00 e R\$ 500,00	10%
Acima de R\$ 500,00	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 que se seguem.

PREÇO	CATEGORIA	PERCENTUAL DE AUMENTO
< = 25	1	5%
	2	8%
	3	10%
> 25	1	12%
	2	15%
	3	18%

- ◆ O valor do imposto, usando as regras a seguir.

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

Categoria: 2

Situação: R

- ◆ O novo preço, ou seja, o preço mais aumento menos imposto.
- ◆ A classificação, usando as regras a seguir.

NOVO PREÇO	CLASSIFICAÇÃO
< = R\$ 50,00	Barato
Entre R\$ 50,00 e R\$ 120,00	Normal
> = R\$ 120,00	Caro

25. Uma empresa decidiu dar uma gratificação de Natal a 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 à tabela que se segue, na qual:

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

H (MINUTOS)	PRÊMIO (R\$)
> 2.400	500,00
1.800 ──○ 2.400	400,00
1.200 ──● 1.800	300,00
600 ──○ 1.200	200,00
< 600	100,00

ESTRUTURA DE REPETIÇÃO

5.1 ESTRUTURA DE REPETIÇÃO EM ALGORITMO

Uma estrutura de repetição é utilizada quando um trecho do algoritmo ou até mesmo o algoritmo inteiro precisa ser repetido. O número de repetições pode ser fixo ou estar atrelado a uma condição. Assim, existem estruturas para tais situações, descritas a seguir.

5.1.1 ESTRUTURA DE REPETIÇÃO PARA NÚMERO DEFINIDO DE REPETIÇÕES (ESTRUTURA PARA)

Essa estrutura de repetição é utilizada quando se sabe o número de vezes que um trecho do algoritmo deve ser repetido. O formato geral dessa estrutura é:

```
PARA I ← valor inicial ATÉ valor final FAÇA [PASSO n]
INÍCIO
    comando1
    comando2
    ...
    comandom
FIM
```

O comando1, o comando2 e o comandom serão executados utilizando-se a variável I como controle, e seu conteúdo vai variar do valor inicial até o valor final. A informação do PASSO está entre colchetes porque é opcional. O PASSO indica como será a variação da variável de controle. Por exemplo, quando for indicado PASSO 2, a variável de controle será aumentada em 2 unidades a cada iteração até atingir o valor final. Quando a informação do PASSO for suprimida, isso significa que o incremento ou o decremento da variável de controle será de 1 unidade.

Quando houver apenas um comando a ser repetido, os marcadores de bloco INÍCIO e FIM poderão ser suprimidos.

Exemplos:

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

O comando ESCREVA I será executado dez vezes, ou seja, para I variando de 1 a 10. Assim, os valores de I serão: 1, 2, 3, 4, 5, 6, 7, 8, 9 e 10.

```
PARA J ← 1 ATÉ 9 FAÇA PASSO 2
    ESCREVA J
```

O comando ESCREVA J será executado cinco vezes, ou seja, para J variando de 1 a 10, de 2 em 2. Assim, os valores de J serão: 1, 3, 5, 7 e 9.

```
PARA I ← 10 ATÉ 5 FAÇA
    ESCREVA I
```

O comando **ESCREVA I** será executado seis vezes, ou seja, para **I** variando de 10 a 5. Assim, os valores de **I** serão: 10, 9, 8, 7, 6 e 5.

```
PARA J ← 15 ATÉ 1 FAÇA PASSO -2
      ESCREVA J
```

O comando **ESCREVA J** será executado oito vezes, ou seja, para **J** variando de 15 a 1, de 2 em 2. Assim, os valores de **J** serão: 15, 13, 11, 9, 7, 5, 3 e 1.

5.1.2 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 que um trecho do algoritmo deve ser repetido, embora também possa ser utilizada quando se conhece esse número.

Essa estrutura baseia-se na análise de uma condição. A repetição será feita enquanto a condição mostrar-se verdadeira.

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, o comando1, o comando2 e o comando3 serão executados.
Exemplos:

```
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	

No trecho do algoritmo anterior, portanto, os comandos que estão dentro da estrutura de repetição serão repetidos quatro vezes.

```

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

Simulação:

Y	X	
1	1	Valores iniciais
1	2	
2	3	
6	4	Valores obtidos dentro da estrutura de repetição
24	5	
120	6	

No trecho do algoritmo anterior, portanto, os comandos que se localizam na estrutura de repetição serão repetidos cinco vezes. Nesse exemplo, a estrutura ENQUANTO é utilizada para repetir o trecho do algoritmo um número definido de vezes.

5.1.3 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 que um trecho do algoritmo deve ser repetido, embora também possa ser utilizada quando se conhece esse número.

Essa estrutura baseia-se na análise de uma condição. A repetição será feita até a condição tornar-se verdadeira.

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

```

REPITA
  comandos
  ATÉ condição
  
```

Repita os comandos até a condição se tornar verdadeira.

Exemplos:

```

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	

No trecho do algoritmo anterior, portanto, os comandos de dentro da estrutura de repetição serão repetidos quatro vezes.

```
X ← 1
Y ← 1
REPITA
  Y ← Y * X
  X ← X + 1
ATÉ X = 6
```

Simulação:

Y	X	
1	1	Valores iniciais
1	2	
2	3	
6	4	Valores obtidos dentro da estrutura de repetição
24	5	
120	6	

No trecho do algoritmo anterior, portanto, os comandos que se localizam dentro da estrutura de repetição serão repetidos cinco vezes. Nesse exemplo, a estrutura REPITA é utilizada para repetir o trecho do algoritmo um número definido de vezes.

5.2 ESTRUTURA DE REPETIÇÃO EM PASCAL

5.2.1 ESTRUTURA DE REPETIÇÃO FOR

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

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

O comando será executado utilizando-se a variável I como controle, e seu 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;
```

O comando1 e o comando2 serão executados utilizando-se a variável J como controle, e seu 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-se a variável K como controle, e seu 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;
```

O comando1, o comando2 e o comando3 serão executados utilizando-se a variável h como controle, e seu conteúdo vai variar do valor inicial até o valor final, de 1 em 1, decrementando automaticamente.

OBSERVAÇÃO Na linguagem PASCAL, a estrutura de repetição FOR funciona obrigatoriamente e automaticamente de 1 em 1, incrementando ou decrementando.

Exemplos:

```
FOR i := 1 TO 5 DO
  WRITELN(i);
```

No trecho de programa acima, o comando WRITELN(i); será executado cinco vezes, ou seja, para i valendo 1, 2, 3, 4 e 5.

```
FOR i := 10 DOWNTO 1 DO
  WRITELN(i);
```

No trecho de programa acima, o comando WRITELN(i); será executado dez vezes, ou seja, para i valendo 10, 9, 8, 7, 6, 5, 4, 3, 2 e 1.

5.2.2 ESTRUTURA DE REPETIÇÃO WHILE

A estrutura de repetição WHILE é utilizada quando o número de repetições necessárias não é fixo, apesar de também poder ser utilizada quando se conhece a quantidade de repetições.

Nessa estrutura, os comandos serão repetidos enquanto a condição for verdadeira e 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 DO
  comando;
```

Enquanto a condição for verdadeira, o comando será executado.

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

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

Exemplos:

```
X = 0;
WHILE X <> 5 DO
BEGIN
  WRITELN('Valor de X = ',X);
  X = X + 1;
END;
WRITELN('Valor de X depois que sair da estrutura = ',X);
```

No trecho de programa acima, os comandos WRITELN('Valor de X = ',X) e X = X + 1; serão executados cinco vezes. O teste condicional avaliará x valendo 0, 1, 2, 3, 4 e 5.

Simulação:

TELA	X	VALOR INICIAL
	0	
Valor de X = 0	1	Valores obtidos dentro da estrutura de repetição

continua

continuação

TELA	X	VALOR INICIAL
Valor de X = 1	2	
Valor de X = 2	3	Valores obtidos dentro da estrutura de repetição
Valor de X = 3	4	
Valor de X = 4	5	Valor obtido dentro da estrutura de repetição, que torna a condição falsa e interrompe a repetição
Valor de X depois que sair da estrutura = 5		

```

X = 1;
Y = 10;
WHILE Y > X DO
BEGIN
WRITELN('Valor de Y = ',Y);
Y = Y - 2;
END;
WRITELN('Valor de Y depois que sair da estrutura = ',Y);

```

No trecho de programa acima, os comandos WRITELN('Valor de Y = ',Y); e Y = Y - 2; serão executados cinco vezes. O teste condicional avaliará Y valendo 10, 8, 6, 4, 2 e 0.

Simulação:

TELA	X	Y	VALORES INICIAIS
	1	10	
Valor de Y = 10	1	8	
Valor de Y = 8	1	6	Valores obtidos dentro da estrutura de repetição
Valor de Y = 6	1	4	
Valor de Y = 4	1	2	
Valor de Y = 2	1	0	Valor obtido dentro da estrutura de repetição, que torna a condição falsa e interrompe a repetição
Valor de Y depois que sair da estrutura = 0			

```

X = 1;
Y = 1;
WHILE X < Y DO
BEGIN
WRITELN('Valor de X = ',X);
X = X + 1;
END;

```

No trecho de programa acima, os comandos WRITELN('Valor de X = ',X); e X = X + 1; não serão executados, pois com os valores iniciais de X e Y a condição é falsa, logo, não ocorrerá a entrada na estrutura de repetição para execução de seus comandos.

5.2.3 ESTRUTURA DE REPETIÇÃO REPEAT

A estrutura de repetição REPEAT é utilizada quando o número de repetições necessárias não é fixo, apesar de também poder ser utilizada quando se conhece o número de repetições.

Nessa estrutura, os comandos serão repetidos até a condição tornar-se verdadeira e o teste condicional ocorre no final, o que significa que a repetição será executada no mínimo uma vez.

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

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

Exemplos:

```
X = 0;
REPEAT
    WRITELN('Valor de X = ', X);
    X = X + 1;
UNTIL X = 5;
WRITELN('Valor de X depois que sair da estrutura = ', X);
```

No trecho de programa acima, os comandos `WRITELN('Valor de X = ', X);` e `X = X + 1;` serão executados cinco vezes. O teste condicional avaliará `X` valendo 1, 2, 3, 4 e 5.

Simulação:

TELA	X	
	0	VALOR INICIAL
Valor de X = 0	1	
Valor de X = 1	2	
Valor de X = 2	3	Valores obtidos dentro da estrutura de repetição
Valor de X = 3	4	
Valor de X = 4	5	Valor obtido dentro da estrutura de repetição, que torna a condição verdadeira e interrompe a repetição
Valor de X depois que sair da estrutura = 5		

No trecho do algoritmo anterior, portanto, os comandos que se localizam na estrutura de repetição serão repetidos cinco vezes. Nesse exemplo, a estrutura `REPITA` está sendo utilizada para repetir o trecho do algoritmo um número definido de vezes.

```
X = 1;
Y = 10;
REPEAT
    WRITELN('Valor de Y = ', Y);
    Y = Y - 2;
UNTIL Y <= X;
WRITELN('Valor de Y depois que sair da estrutura = ', Y);
```

No trecho de programa acima, os comandos `WRITELN('Valor de Y = ', Y);` e `Y = Y - 2;` serão executados cinco vezes. O teste condicional avaliará `Y` valendo 8, 6, 4, 2 e 0.

Simulação:

TELA	X	Y	
	1	10	VALORES INICIAIS
Valor de Y = 10	1	8	Valores obtidos dentro da estrutura de repetição
Valor de Y = 8	1	6	

continua

continuação

TELA	X	Y	VALORES INICIAIS
Valor de Y = 6	1	4	Valores obtidos dentro da estrutura de repetição
Valor de Y = 4	1	2	
Valor de Y = 2	1	0	Valor obtido dentro da estrutura de repetição, que torna a condição verdadeira e interrompe a repetição
Valor de Y depois que sair da estrutura = 0			

5.3 ESTRUTURA DE REPETIÇÃO EM C/C++

5.3.1 ESTRUTURA DE REPETIÇÃO FOR

Essa estrutura de repetição é utilizada quando se sabe o número de vezes 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 decremento 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 decremento) 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, à variável `a` é atribuído inicialmente o valor 1 e depois vai sendo incrementada em uma unidade. 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).

Se for 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. A cada decremento, o bloco de comando que está entre 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).

Exemplos:

```

for (i = 1; i <= 5; i++)
cout << i;
ou
for (i = 1; i <= 5; i=i+1)
cout << i;

```

Nos trechos de programa acima, que expressam a mesma coisa, o comando `cout << i;` será executado cinco vezes, ou seja, para `i` valendo 1, 2, 3, 4 e 5.

```

for (i = 10; i >= 1; i--)
cout << i;
ou
for (i = 10; i >= 1; i=i-1)
cout << i;

```

Nos trechos de programa acima, que são exatamente a mesma coisa, o comando `cout << i;` será executado dez vezes, ou seja, para `i` valendo 10, 9, 8, 7, 6, 5, 4, 3, 2 e 1.

```

for (i = 0; i <= 10; i=i+2)
cout << i;

```

No trecho de programa acima, o comando `cout << i;` será executado seis vezes, ou seja, para `i` valendo 0, 2, 4, 6, 8 e 10.

```

for (i = 100; i >= 0; i=i-20)
cout << i;

```

No trecho de programa acima, o comando `cout << i;` será executado seis vezes, ou seja, para `i` valendo 100, 80, 60, 40, 20 e 0.

5.3.2 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. Isto 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...`).

Exemplos:

```

X = 0;
while (X != 5)
{
    cout << "Valor de X = " << X;
    X = X + 1;
}
cout << "Valor de X depois que sair da estrutura = " << X;

```

No trecho de programa acima, os comandos `cout << "Valor de X = " << X;` e `X = X + 1;` serão executados cinco vezes. O teste condicional avaliará `X` valendo 0, 1, 2, 3, 4 e 5.

Simulação:

TELA	X	VALOR INICIAL
	0	
Valor de X = 0	1	
Valor de X = 1	2	Valores obtidos dentro da estrutura de repetição
Valor de X = 2	3	
Valor de X = 3	4	
Valor de X = 4	5	Valor obtido dentro da estrutura de repetição, que torna a condição falsa e interrompe a repetição
Valor de X depois que sair da estrutura = 5		

```

X = 1;
Y = 10;
while (Y > X)
{
    cout << "Valor de Y = " << Y;
    Y = Y - 2;
}
cout << "Valor de Y depois que sair da estrutura = " << Y;

```

No trecho de programa acima, os comandos `cout << "Valor de Y = " << Y;` e `Y = Y - 2;` serão executados cinco vezes. O teste condicional avaliará `Y` valendo 10, 8, 6, 4, 2 e 0.

Simulação:

TELA	X	Y	VALORES INICIAIS
	1	10	
Valor de Y = 10	1	8	
Valor de Y = 8	1	6	Valores obtidos dentro da estrutura de repetição
Valor de Y = 6	1	4	
Valor de Y = 4	1	2	
Valor de Y = 2	1	0	Valor obtido dentro da estrutura de repetição, que torna a condição verdadeira e interrompe a repetição
Valor de Y depois que sair da estrutura = 0			

```

X = 1;
Y = 1;
while (X < Y)
{
    cout << "Valor de X = " << X;
    X = X + 1;
}

```

No trecho de programa acima, os comandos `cout << "Valor de X = " << X;` e `X = X + 1;` não serão executados, pois com os valores iniciais de `X` e `Y` a condição é falsa, logo, não ocorrerá a entrada na estrutura de repetição para execução de seus comandos.

5.3.3 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.

Exemplos:

```

X = 0;
do
{
    cout << "Valor de X = " << X;
    X = X + 1;
}
while (X != 5);
cout << "Valor de X depois que sair da estrutura = " << X;

```

No trecho de programa acima, os comandos `cout << "Valor de X = " << X;` e `X = X + 1;` serão executados cinco vezes. O teste condicional avaliará `X` valendo 0, 1, 2, 3, 4 e 5.

Simulação:

TELA	X	
	0	VALOR INICIAL
Valor de X = 0	1	
Valor de X = 1	2	
Valor de X = 2	3	Valores obtidos dentro da estrutura de repetição
Valor de X = 3	4	
Valor de X = 4	5	Valor obtido dentro da estrutura de repetição, que torna a condição falsa e interrompe a repetição
Valor de X depois que sair da estrutura = 5		

```

X = 1;
Y = 10;
do
{
    cout << "Valor de Y = " << Y;
    Y = Y - 2;
}
while (Y > X);
cout << "Valor de Y depois que sair da estrutura = " << Y;

```

No trecho de programa acima, os comandos `cout << "Valor de Y = " << Y;` e `Y = Y - 2;` serão executados cinco vezes. O teste condicional avaliará `Y` valendo 10, 8, 6, 4, 2 e 0.

Simulação:

TELA	X	Y	
	1	10	VALORES INICIAIS
Valor de Y = 10	1	8	
Valor de Y = 8	1	6	
Valor de Y = 6	1	4	Valores obtidos dentro da estrutura de repetição
Valor de Y = 4	1	2	

continua

continuação

TELA	X	Y	VALORES INICIAIS
Valor de Y = 2	1	10	Valor obtido dentro da estrutura de repetição, que torna a condição verdadeira e interrompe a repetição
Valor de Y depois que sair da estrutura = 0			

5.4 ESTRUTURA DE REPETIÇÃO EM JAVA

5.4.1 ESTRUTURA DE REPETIÇÃO FOR

Essa estrutura de repetição é utilizada quando se sabe o número de vezes 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 decremento 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 decremento) 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++)
    System.out.println("O valor de a é: " +a);
```

No exemplo acima, a variável *a* é inicializada com o valor 1 e é incrementada em uma unidade. A cada incremento, o comando `System.out.println` 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).

Se for necessária a repetição de mais de um comando, a linguagem 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)
{
    System.out.println("Digite um número: ");
    dado = new Scanner(System.in);
    x = dado.nextInt();
}
```

No exemplo anterior, a variável *a* começa com o valor 15 e é decrementada em duas unidades. A cada decremento, o bloco de comando que está entre 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).

Exemplos:

```
for (i = 1; i <= 5; i++)
    System.out.println(i);
ou
for (i = 1; i <= 5; i=i+1)
    System.out.println(i);
```

Nos trechos de programa anteriores, que expressam a mesma coisa, o comando `System.out.println(i);` será executado cinco vezes, ou seja, para `i` valendo 1, 2, 3, 4 e 5.

```
for (i = 10; i >= 1; i--)
    System.out.println(i);
ou
for (i = 10; i >= 1; i=i-1)
    System.out.println(i);
```

Nos trechos de programa acima, que são exatamente a mesma coisa, o comando `System.out.println(i);` será executado dez vezes, ou seja, para `i` valendo 10, 9, 8, 7, 6, 5, 4, 3, 2 e 1.

```
for (i = 0; i <= 10; i=i+2)
    System.out.println(i);
```

No trecho de programa acima, o comando `System.out.println(i);` será executado seis vezes, ou seja, para `i` valendo 0, 2, 4, 6, 8 e 10.

```
for (i = 100; i >= 0; i=i-20)
    System.out.println(i);
```

No trecho de programa acima, o comando `System.out.println(i);` será executado seis vezes, ou seja, para `i` valendo 100, 80, 60, 40, 20 e 0.

5.4.2 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, o que significa que existe a possibilidade de a 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...`).

Exemplos:

```
X = 0;
while (X != 5)
{
    System.out.println("Valor de X = "+X);
    X = X + 1;
}
System.out.println("Valor de X depois que sair da estrutura = "+X);
```

No trecho de programa acima, os comandos `System.out.println("Valor de X = "+X);` e `X = X + 1;` serão executados cinco vezes. O teste condicional avaliará `X` valendo 0, 1, 2, 3, 4 e 5.

Simulação:

TELA	X	VALOR INICIAL
Valor de X = 0	0	
Valor de X = 1	1	
Valor de X = 2	2	Valores obtidos dentro da estrutura de repetição
Valor de X = 3	3	
Valor de X = 4	4	
Valor de X depois que sair da estrutura = 5	5	Valor obtido dentro da estrutura de repetição, que torna a condição falsa e interrompe a repetição

```

X = 1;
Y = 10;
while (Y > X)
{
    System.out.println("Valor de Y = "+Y);
    Y = Y - 2;
}
System.out.println("Valor de Y depois que sair da estrutura = "+Y);

```

No trecho de programa acima, os comandos `System.out.println("Valor de Y = "+Y);` e `Y = Y - 2;` serão executados cinco vezes. O teste condicional avaliará `Y` valendo 10, 8, 6, 4, 2 e 0.

Simulação:

TELA	X	Y	VALORES INICIAIS
Valor de Y = 10	1	10	
Valor de Y = 8	1	8	
Valor de Y = 6	1	6	Valores obtidos dentro da estrutura de repetição
Valor de Y = 4	1	4	
Valor de Y = 2	1	2	
Valor de Y depois que sair da estrutura = 0	1	0	Valor obtido dentro da estrutura de repetição, que torna a condição verdadeira e interrompe a repetição

```

X = 1;
Y = 1;
while (X < Y)
{
    System.out.println("Valor de X = "+X);
    X = X + 1;
}

```

No trecho de programa, os comandos `System.out.println("Valor de X = "+X);` e `X = X + 1;` não serão executados, pois com os valores iniciais de `X` e `Y` a condição é falsa. Logo, não ocorrerá a entrada na estrutura de repetição para execução dos seus comandos.

5.4.3 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.

Exemplos:

```

X = 0;
do
{
    System.out.println("Valor de X = " + X);
    X = X + 1;
}
while (X != 5);
System.out.println("Valor de X depois que sair da estrutura = " + X);

```

No trecho de programa acima, os comandos `System.out.println ("Valor de X = "+X);` e `X = X + 1;` serão executados cinco vezes. O teste condicional avaliará x valendo 1, 2, 3, 4 e 5.

Simulação:

TELA	X	
	0	VALOR INICIAL
Valor de X = 0	1	
Valor de X = 1	2	Valores obtidos dentro da estrutura de repetição
Valor de X = 2	3	
Valor de X = 3	4	
Valor de X = 4	5	Valor obtido dentro da estrutura de repetição, que torna a condição falsa e interrompe a repetição
Valor de X depois que sair da estrutura = 5		

```

X = 1;
Y = 10;
do
{
    System.out.println("Valor de Y = " + Y);
    Y = Y - 2;
}
while (Y > X);
System.out.println("Valor de Y depois que sair da estrutura = " + Y);

```

No trecho de programa acima, os comandos `System.out.println ("Valor de Y = "+Y);` e `Y = Y - 2;` serão executados cinco vezes. O teste condicional avaliará y valendo 8, 6, 4, 2 e 0.

Simulação:

TELA	X	Y	
	1	10	VALORES INICIAIS
Valor de Y = 10	1	8	
Valor de Y = 8	1	6	Valores obtidos dentro da estrutura de repetição
Valor de Y = 6	1	4	
Valor de Y = 4	1	2	

continua

continuação

TELA	X	Y	VALORES INICIAIS
			1
Valor de Y = 2	1	0	Valor obtido dentro da estrutura de repetição, que torna a condição verdadeira e interrompe a repetição
Valor de Y depois que sair da estrutura = 0			

EXERCÍCIOS RESOLVIDOS

1. Um funcionário de uma empresa recebe aumento salarial anualmente. Sabe-se que:

- a) Esse funcionário foi contratado em 2005, com salário inicial de R\$ 1.000,00.
- b) Em 2006, ele recebeu aumento de 1,5% sobre seu salário inicial.
- c) A partir de 2007 (inclusive), os aumentos salariais sempre corresponderam ao dobro do percentual do ano anterior.

Faça um programa que determine o salário atual desse funcionário.

ALGORITMO

SOLUÇÃO:

```

ALGORITMO
DECLARE i, ano_atual, salario NUMÉRICO
        novo_salario, percentual NUMÉRICO
LEIA ano_atual
salario ← 1000
percentual ← 1,5/100
novo_salario ← salario + percentual * salario
PARA i ← 2007 ATÉ ano_atual FAÇA
INÍCIO
percentual ← 2 * percentual
novo_salario ← novo_salario + percentual * novo_salario
FIM
ESCREVA novo_salario
FIM_ALGORITMO.

```



1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP5\PASCAL\EX1_A.PAS e \EXERC\CAP5\PASCAL\EX1_A.EXE

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\PASCAL\EX1_B.PAS e \EXERC\CAP5\PASCAL\EX1_B.EXE



1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP5\C++\EX1_A.CPP e \EXERC\CAP5\C++\EX1_A.EXE

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\C++\EX1_B.CPP e \EXERC\CAP5\C++\EX1_B.EXE

**1^a SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP5\JAVA\EX1_A.java e \EXERC\CAP5\JAVA\EX1_A.class

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\JAVA\EX1_B.java e \EXERC\CAP5\JAVA\EX1_B.class

- 2.** Faça um programa que leia um valor N inteiro e positivo, calcule e mostre o valor de E, conforme a fórmula a seguir:

$$E = 1 + 1/1! + 1/2! + 1/3! + \dots + 1/N!$$

ALGORITMO**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^a SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP5\PASCAL\EX2_A.PAS e \EXERC\CAP5\PASCAL\EX2_A.EXE

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA REPEAT:

\EXERC\CAP5\PASCAL\EX2_B.PAS e \EXERC\CAP5\PASCAL\EX2_B.EXE

**1^a SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP5\C++\EX2_A.CPP e \EXERC\CAP5\C++\EX2_A.EXE

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\C++\EX2_B.CPP e \EXERC\CAP5\C++\EX2_B.EXE

**1^a SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP5\JAVA\EX2_A.java e \EXERC\CAP5\JAVA\EX2_A.class

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\JAVA\EX2_B.java e \EXERC\CAP5\JAVA\EX2_B.class

-  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.

ALGORITMO

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^a SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP5\PASCAL\EX3_A.PAS e \EXERC\CAP5\PASCAL\EX3_A.EXE

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\PASCAL\EX3_B.PAS e \EXERC\CAP5\PASCAL\EX3_B.EXE



1^a SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP5\C++\EX3_A.CPP e \EXERC\CAP5\C++\EX3_A.EXE

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\C++\EX3_B.CPP e \EXERC\CAP5\C++\EX3_B.EXE



1^a SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP5\JAVA\EX3_A.java e \EXERC\CAP5\JAVA\EX3_A.class

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\JAVA\EX3_B.java e \EXERC\CAP5\JAVA\EX3_B.class

-  4. 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 2007);
- c) número de acidentes de trânsito com vítimas (em 2007).

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.

ALGORITMO **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
            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 ESCREVA "Não foi digitada nenhuma cidade com menos de 2000
        ↪ veículos"
        SENÃO INÍCIO
            media_acid ← soma_acid/cont_acid
            ESCREVA media_acid
            FIM
    FIM_ALGORITMO.

```

**1^a SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP5\PASCAL\EX4_A.PAS e \EXERC\CAP5\PASCAL\EX4_A.EXE

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA REPEAT:

\EXERC\CAP5\PASCAL\EX4_B.PAS e \EXERC\CAP5\PASCAL\EX4_B.EXE

**1^a SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP5\C++\EX4_A.CPP e \EXERC\CAP5\C++\EX4_A.EXE

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\C++\EX4_B.CPP e \EXERC\CAP5\C++\EX4_B.EXE

**1^a SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP5\JAVA\EX4_A.java e \EXERC\CAP5\JAVA\EX4_A.class

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\JAVA\EX4_B.java e \EXERC\CAP5\JAVA\EX4_B.class

5. Faça um programa que leia o número de termos e um valor positivo para X, calcule e mostre o valor da série a seguir:

$$S = \frac{-X^2}{1!} + \frac{+X^3}{2!} - \frac{-X^4}{3!} + \frac{+X^5}{4!} - \frac{-X^6}{3!} + \frac{+X^7}{2!} - \frac{-X^8}{1!} + \frac{+X^9}{2!} - \frac{-X^{10}}{3!} + \frac{+X^{11}}{4!} - \dots$$

ALGORITMO 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^a SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP5\PASCAL\EX5_A.PAS e \EXERC\CAP5\PASCAL\EX5_A.EXE

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

```
\EXERC\CAP5\PASCAL\EX5_B.PAS e \EXERC\CAP5\PASCAL\EX5_B.EXE
```

**1^a SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:**

```
\EXERC\CAP5\C++\EX5_A.CPP e \EXERC\CAP5\C++\EX5_A.EXE
```

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

```
\EXERC\CAP5\C++\EX5_B.CPP e \EXERC\CAP5\C++\EX5_B.EXE
```

**1^a SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:**

```
\EXERC\CAP5\JAVA\EX5_A.java e \EXERC\CAP5\JAVA\EX5_A.class
```

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

```
\EXERC\CAP5\JAVA\EX5_B.java e \EXERC\CAP5\JAVA\EX5_B.class
```

6. 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:

- a) Leia as informações dos funcionários, exceto o valor da hora trabalhada, não permitindo que sejam informados turnos nem categorias inexistentes. Trabalhe sempre com a digitação de letras maiúsculas.
- b) Calcule o valor da hora trabalhada, conforme a tabela a seguir. Adote o valor de R\$ 450,00 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

- c) 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 seu salário inicial, conforme a tabela a seguir.

SALÁRIO INICIAL	AUXÍLIO-AUMENTAÇÃO
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 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 ← 450
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 ← sal_min * 18/100
                SENÃO valor ← sal_min * 15/100
            FIM
        SENÃO INÍCIO
            SE turno = "N"
                ENTÃO valor ← sal_min * 13/100
                SENÃO valor ← sal_min * 10/100
            FIM
        sal_inicial ← nht * valor
    SE sal_inicial <= 300
        ENTÃO aux ← sal_inicial * 20/100
    SENÃO SE sal_inicial < 600
        ENTÃO aux ← sal_inicial * 15/100
        SENÃO aux ← sal_inicial * 5/100
    sal_final ← sal_inicial + aux
    ESCREVA codigo, nht, valor, sal_inicial, aux, sal_final
    FIM
FIM_ALGORITMO.

```

**1^a SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP5\PASCAL\EX6_A.PAS e \EXERC\CAP5\PASCAL\EX6_A.EXE

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA REPEAT:

\EXERC\CAP5\PASCAL\EX6_B.PAS e \EXERC\CAP5\PASCAL\EX6_B.EXE

**1^a SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP5\C++\EX6_A.CPP e \EXERC\CAP5\C++\EX6_A.EXE

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\C++\EX6_B.CPP e \EXERC\CAP5\C++\EX6_B.EXE

**1^a SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP5\JAVA\EX6_A.java e \EXERC\CAP5\JAVA\EX6_A.class

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\JAVA\EX6_B.java e \EXERC\CAP5\JAVA\EX6_B.class

7. Faça um programa que monte os oito primeiros termos da seqüência de Fibonacci.

0-1-1-2-3-5-8-13-21-34-55...

ALGORITMO**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^a SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP5\PASCAL\EX7_A.PAS e \EXERC\CAP5\PASCAL\EX7_A.EXE

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA REPEAT:

\EXERC\CAP5\PASCAL\EX7_B.PAS e \EXERC\CAP5\PASCAL\EX7_B.EXE

**1^a SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP5\C++\EX7_A.CPP e \EXERC\CAP5\C++\EX7_A.EXE

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\C++\EX7_B.CPP e \EXERC\CAP5\C++\EX7_B.EXE

**1^a SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP5\JAVA\EX7_A.java e \EXERC\CAP5\JAVA\EX7_A.class

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\JAVA\EX7_B.java e \EXERC\CAP5\JAVA\EX7_B.class

8. Faça um programa que leia o número de termos, determine e mostre os valores de acordo com a série a seguir:

Série = 2, 7, 3, 4, 21, 12, 8, 63, 48, 16, 189, 192, 32, 567, 768...

ALGORITMO **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 * 3
            ESCREVA num2
            i ← i + 1
            SE i ≠ num_termos
            ENTÃO INÍCIO
                num3 ← num3 * 4
                ESCREVA num3
                i ← i + 1
            FIM
        FIM
    FIM
FIM_ALGORITMO.

```


1^a SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\PASCAL\EX8_A.PAS e \EXERC\CAP5\PASCAL\EX8_A.EXE

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA REPEAT:

\EXERC\CAP5\PASCAL\EX8_B.PAS e \EXERC\CAP5\PASCAL\EX8_B.EXE


1^a SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\C++\EX8_A.CPP e \EXERC\CAP5\C++\EX8_A.EXE

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\C++\EX8_B.CPP e \EXERC\CAP5\C++\EX8_B.EXE


1^a SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\JAVA\EX8_A.java e \EXERC\CAP5\JAVA\EX8_A.class

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\JAVA\EX8_B.java e \EXERC\CAP5\JAVA\EX8_B.class

9. 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:

MÉDIA ARITMÉTICA	MENSAGEM
Até 3	Reprovado
Entre 3 e 7	Exame
De 7 para cima	Aprovado

- ◆ o total de alunos aprovados;
- ◆ o total de alunos de exame;
- ◆ o total de alunos reprovados;
- ◆ a média da classe.

ALGORITMO**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^a SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP5\PASCAL\EX9_A.PAS e \EXERC\CAP5\PASCAL\EX9_A.EXE

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\PASCAL\EX9_B.PAS e \EXERC\CAP5\PASCAL\EX9_B.EXE

**1^a SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP5\C++\EX9_A.CPP e \EXERC\CAP5\C++\EX9_A.EXE

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\C++\EX9_B.CPP e \EXERC\CAP5\C++\EX9_B.EXE

**1^a SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP5\JAVA\EX9_A.java e \EXERC\CAP5\JAVA\EX9_A.class

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\JAVA\EX9_B.java e \EXERC\CAP5\JAVA\EX9_B.class

10. Em um campeonato de futebol existem cinco times e cada um 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, tot80 NUMÉRICO
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^a SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP5\PASCAL\EX10_A.PAS e \EXERC\CAP5\PASCAL\EX10_A.EXE

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\PASCAL\EX10_B.PAS e \EXERC\CAP5\PASCAL\EX10_B.EXE

**1^a SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP5\C++\EX10_A.CPP e \EXERC\CAP5\C++\EX10_A.EXE

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\C++\EX10_B.CPP e \EXERC\CAP5\C++\EX10_B.EXE

**1^a SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP5\JAVA\EX10_A.java e \EXERC\CAP5\JAVA\EX10_A.class

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\JAVA\EX10_B.java e \EXERC\CAP5\JAVA\EX10_B.class

 11. Faça um programa que receba um número inteiro maior que 1, verifique se o número fornecido é primo ou não e mostre mensagem de número primo ou de número não primo.

Um número é primo quando é divisível apenas por 1 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.

**1^a SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP5\PASCAL\EX11_A.PAS e \EXERC\CAP5\PASCAL\EX11_A.EXE

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA REPEAT:

\EXERC\CAP5\PASCAL\EX11_B.PAS e \EXERC\CAP5\PASCAL\EX11_B.EXE

**1^a SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP5\C++\EX11_A.CPP e \EXERC\CAP5\C++\EX11_A.EXE

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\C++\EX11_B.CPP e \EXERC\CAP5\C++\EX11_B.EXE

**1^a SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP5\JAVA\EX11_A.java e \EXERC\CAP5\JAVA\EX11_A.class

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\JAVA\EX11_B.java e \EXERC\CAP5\JAVA\EX11_B.class

 12. 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 21 peças por mês – classe 2;
- ◆ trabalhadores que fazem mais de 21 peças por mês – classe 3.

A classe 1 recebe salário mínimo. A classe 2 recebe salário mínimo mais 3% deste salário por peça, acima das 30 peças iniciais. A classe 3 recebe salário mínimo mais 5% deste salário 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.

ALGORITMOSOLUÇÃO:

```

ALGORITMO
DECLARE num_op, pecas_op, num_maior, cont_m, cont_f NUMÉRICO
      tot_pecas, cont, media_m, salario_maior 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 ← 450
        SE pecas_op > 30 E pecas_op <= 21
            ENTÃO salario_op ← 450 + (pecas_op - 30 * 3 / 100 * 450)
        SE pecas_op > 21
            ENTÃO salario_op ← 450 + (pecas_op - 21 * 5 / 100 * 450)
        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
    FIM
ESCREVA "Total da folha de pagamento = ", tot_folha
ESCREVA "Total de peças fabricadas no mês = ", tot_pecas
SE cont_m = 0
ENTÃO ESCREVA "NENHUM HOMEM"
SENÃO INÍCIO
    media_m ← media_m / cont_m
    ESCREVA "Média de peças fabricadas por homens = ", media_m
    FIM
SE cont_f = 0
ENTÃO ESCREVA "NENHUMA MULHER"
SENÃO INÍCIO
    media_f ← media_f / cont_f
    ESCREVA "Média de peças fabricadas por mulheres = ", media_f
    FIM
ESCREVA "O número do operário com maior salário é ", num_maior
FIM_ALGORITMO.

```

**1^a SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP5\PASCAL\EX12_A.PAS e \EXERC\CAP5\PASCAL\EX12_A.EXE

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\PASCAL\EX12_B.PAS e \EXERC\CAP5\PASCAL\EX12_B.EXE

**1^a SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP5\C++\EX12_A.CPP e \EXERC\CAP5\C++\EX12_A.EXE

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\C++\EX12_B.CPP e \EXERC\CAP5\C++\EX12_B.EXE

**1^a SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP5\JAVA\EX12_A.java e \EXERC\CAP5\JAVA\EX12_A.class

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\JAVA\EX12_B.java e \EXERC\CAP5\JAVA\EX12_B.class

 13. Foi feita uma pesquisa para determinar o índice de mortalidade infantil em certo período. Faça um programa que:

- ◆ leia o número de crianças nascidas no período;
- ◆ identifique o sexo (M ou F) e o tempo de vida de cada criança nascida.

O programa deve calcular e mostrar:

- ◆ 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, porc_m, tot_f NUMÉRICO
      tot_m, tot_24, porc_24 NUMÉRICO
      sexo LITERAL
ESCREVA "Digite o número de crianças nascidas no período "
LEIA num_cri
tot_m ← 0
tot_f ← 0
tot_24 ← 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 ESCREVA "NENHUMA CRIANÇA DIGITADA"
SENÃO INÍCIO
  porc_m ← tot_m * 100 / num_cri
  porc_f ← tot_f * 100 / num_cri
  porc_24 ← tot_24 * 100 / num_cri
  ESCREVA "Percentual de crianças do sexo feminino mortas ", porc_f
  ESCREVA "Percentual de crianças do sexo masculino mortas ", porc_m
  ESCREVA "Percentual de crianças com 24 meses ou menos mortas no período ",
  porc_24
  FIM
FIM_ALGORITMO.

```



1^a SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP5\PASCAL\EX13_A.PAS e \EXERC\CAP5\PASCAL\EX13_A.EXE

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\PASCAL\EX13_B.PAS e \EXERC\CAP5\PASCAL\EX13_B.EXE



1^a SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP5\C++\EX13_A.CPP e \EXERC\CAP5\C++\EX13_A.EXE

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\C++\EX13_B.CPP e \EXERC\CAP5\C++\EX13_B.EXE

**1^a SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP5\JAVA\EX13_A.java e \EXERC\CAP5\JAVA\EX13_A.class

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\JAVA\EX13_B.java e \EXERC\CAP5\JAVA\EX13_B.class

- 14.** 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:

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
  FIM
FIM

```

```

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^a SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP5\PASCAL\EX14_A.PAS e \EXERC\CAP5\PASCAL\EX14_A.EXE

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\PASCAL\EX14_B.PAS e \EXERC\CAP5\PASCAL\EX14_B.EXE



1^a SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP5\C++\EX14_A.CPP e \EXERC\CAP5\C++\EX14_A.EXE

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\C++\EX14_B.CPP e \EXERC\CAP5\C++\EX14_B.EXE



1^a SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP5\JAVA\EX14_A.java e \EXERC\CAP5\JAVA\EX14_A.class

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\JAVA\EX14_B.java e \EXERC\CAP5\JAVA\EX14_B.class

15. Faça um programa que receba o preço unitário, a refrigeração (S para os produtos que necessitem de refrigeração e N para os que não necessitem) e a categoria (A – alimentação, L – limpeza e V – vestuário) de doze produtos, e que calcule e mostre:

- ◆ O custo de estocagem, calculado de acordo com a tabela a seguir.

PREÇO UNITÁRIO	REFRIGERAÇÃO	CATEGORIA	CUSTO DE ESTOCAGEM
Até 20		A	R\$ 2,00
		L	R\$ 3,00
		V	R\$ 4,00
Entre 20 e 50 (inclusive)	S		R\$ 6,00
			R\$ 0,00
		A	R\$ 5,00
Maior que 50	N	S	R\$ 2,00
		V	R\$ 4,00
		A ou V	R\$ 0,00
		N	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, ou seja, preço unitário mais custo de estocagem mais imposto.
- ◆ A classificação calculada usando a tabela a seguir.

PREÇO FINAL	CLASSIFICAÇÃO
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, ou seja, a média dos custos de estocagem e dos impostos dos doze produtos.
- ◆ 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 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.

```



1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:

\EXERC\CAP5\PASCAL\EX15_A.PAS e \EXERC\CAP5\PASCAL\EX15_A.EXE

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA REPEAT:

\EXERC\CAP5\PASCAL\EX15_B.PAS e \EXERC\CAP5\PASCAL\EX15_B.EXE

**1^a SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP5\C++\EX15_A.CPP e \EXERC\CAP5\C++\EX15_A.EXE

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\C++\EX15_B.CPP e \EXERC\CAP5\C++\EX15_B.EXE

**1^a SOLUÇÃO – UTILIZANDO A ESTRUTURA FOR:**

\EXERC\CAP5\JAVA\EX15_A.java e \EXERC\CAP5\JAVA\EX15_A.class

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\JAVA\EX15_B.java e \EXERC\CAP5\JAVA\EX15_B.class

16. Faça um programa para calcular a área de um triângulo, que não permita a entrada de dados inválidos, ou seja, medidas menores ou iguais a 0.

ALGORITMO SOLUÇÃO:

```

ALGORITMO
DECLARE base, altura, area NUMÉRICO
REPITA
    LEIA base
    ATÉ base > 0
    REPITA
        LEIA altura
        ATÉ altura > 0
        area ← base * altura / 2
        ESCREVA area
    FIM_ALGORITMO.

```

**1^a SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP5\PASCAL\EX16_A.PAS e \EXERC\CAP5\PASCAL\EX16_A.EXE

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA REPEAT:

\EXERC\CAP5\PASCAL\EX16_B.PAS e \EXERC\CAP5\PASCAL\EX16_B.EXE

**1^a SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP5\C++\EX16_A.CPP e \EXERC\CAP5\C++\EX16_A.EXE

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\C++\EX16_B.CPP e \EXERC\CAP5\C++\EX16_B.EXE

**1^a SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP5\JAVA\EX16_A.java e \EXERC\CAP5\JAVA\EX16_A.class

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\JAVA\EX16_B.java e \EXERC\CAP5\JAVA\EX16_B.class

17. Faça um programa que receba o salário de um funcionário chamado Carlos. Sabe-se que outro funcionário, João, tem 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. O programa deverá calcular e mostrar a quantidade de meses necessários para que o valor pertencente a João iguale ou ultrapasse o valor pertencente a Carlos.

ALGORITMO 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^a SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP5\PASCAL\EX17_A.PAS e \EXERC\CAP5\PASCAL\EX17_A.EXE

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA REPEAT:

\EXERC\CAP5\PASCAL\EX17_B.PAS e \EXERC\CAP5\PASCAL\EX17_B.EXE

**1^a SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP5\C++\EX17_A.CPP e \EXERC\CAP5\C++\EX17_A.EXE

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\C++\EX17_B.CPP e \EXERC\CAP5\C++\EX17_B.EXE

**1^a SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP5\JAVA\EX17_A.java e \EXERC\CAP5\JAVA\EX17_A.class

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\JAVA\EX17_B.java e \EXERC\CAP5\JAVA\EX17_B.class

18. 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 vinte linhas. A tabela deverá conter o valor lido, seu quadrado, seu cubo e sua raiz quadrada. Finalize 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^a SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP5\PASCAL\EX18_A.PAS e \EXERC\CAP5\PASCAL\EX18_A.EXE

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA REPEAT:

\EXERC\CAP5\PASCAL\EX18_B.PAS e \EXERC\CAP5\PASCAL\EX18_B.EXE

**1^a SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP5\C++\EX18_A.CPP e \EXERC\CAP5\C++\EX18_A.EXE

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\C++\EX18_B.CPP e \EXERC\CAP5\C++\EX18_B.EXE

**1^a SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP5\JAVA\EX18_A.java e \EXERC\CAP5\JAVA\EX18_A.class

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\JAVA\EX18_B.java e \EXERC\CAP5\JAVA\EX18_B.class

- 19.** 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 terminará 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^a SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP5\PASCAL\EX19_A.PAS e \EXERC\CAP5\PASCAL\EX19_A.EXE

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA REPEAT:

\EXERC\CAP5\PASCAL\EX19_B.PAS e \EXERC\CAP5\PASCAL\EX19_B.EXE

**1^a SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP5\C++\EX19_A.CPP e \EXERC\CAP5\C++\EX19_A.EXE

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\C++\EX19_B.CPP e \EXERC\CAP5\C++\EX19_B.EXE

**1^a SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP5\JAVA\EX19_A.java e \EXERC\CAP5\JAVA\EX19_A.class

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\JAVA\EX19_B.java e \EXERC\CAP5\JAVA\EX19_B.class

-  20. 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\$ 30,00. 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. Mostre também a média dos salários líquidos dos professores do sexo masculino e a média dos salários líquidos dos professores do sexo feminino. Considere:

- ◆ desconto para homens, 10% e, para mulheres, 5%;
- ◆ as informações terminarão quando for lido o código = 99999.

ALGORITMO 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 * 30
    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
SE cont_m = 0
ENTÃO ESCREVA "Nenhum professor do sexo masculino"
SENÃO INÍCIO
    media_m ← media_m / cont_m
    ESCREVA media_m
    FIM
SE cont_f = 0
ENTÃO ESCREVA "Nenhum professor do sexo feminino"
SENÃO INÍCIO
    media_f ← media_f / cont_f
    ESCREVA media_f
    FIM
FIM_ALGORITMO.

```

**1^a SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP5\PASCAL\EX20_A.PAS e \EXERC\CAP5\PASCAL\EX20_A.EXE

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA REPEAT:

\EXERC\CAP5\PASCAL\EX20_B.PAS e \EXERC\CAP5\PASCAL\EX20_B.EXE

**1^a SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP5\C++\EX20_A.CPP e \EXERC\CAP5\C++\EX20_A.EXE

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\C++\EX20_B.CPP e \EXERC\CAP5\C++\EX20_B.EXE

**1^a SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP5\JAVA\EX20_A.java e \EXERC\CAP5\JAVA\EX20_A.class

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\JAVA\EX20_B.java e \EXERC\CAP5\JAVA\EX20_B.class

21. 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 30.000.

ALGORITMO **SOLUÇÃO:**

```

ALGORITMO
DECLARE num, soma, qtd, maior, menor, qtd_par NUMÉRICO
        media_par, 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
SE qtd = 0
ENTÃO ESCREVA "Nenhum número digitado"
SENÃO INÍCIO
    ESCREVA soma
    ESCREVA qtd
    media ← soma / qtd
    ESCREVA media
    ESCREVA maior
    ESCREVA menor
    SE qtd_par = 0
        ENTÃO ESCREVA "nenhum par"
    SENÃO INÍCIO
        media_par ← soma_par / qtd_par
        ESCREVA media_par
    FIM
    perc ← qtd_impar * 100 / qtd

```

```

    ESCREVA perc
    FIM
FIM_ALGORITMO.

```

**1^a SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP5\PASCAL\EX21_A.PAS e \EXERC\CAP5\PASCAL\EX21_A.EXE

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA REPEAT:

\EXERC\CAP5\PASCAL\EX21_B.PAS e \EXERC\CAP5\PASCAL\EX21_B.EXE

**1^a SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP5\C++\EX21_A.CPP e \EXERC\CAP5\C++\EX21_A.EXE

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\CPP\EX21_B.CPP e \EXERC\CAP5\CPP\EX21_B.EXE

**1^a SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP5\JAVA\EX21_A.java e \EXERC\CAP5\JAVA\EX21_A.class

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\JAVA\EX21_B.java e \EXERC\CAP5\JAVA\EX21_B.class

22. Uma empresa decidiu fazer um levantamento em relação aos candidatos que se apresentarem para preenchimento de vagas em 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.

O programa também deve calcular e mostrar:

- ◆ 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 21 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, tot NUMÉRICO
        cont_f1, media_idade, perc, menor_idade NUMÉRICO
        sexo, exp LITERAL
tot ← 0
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 = 0
            ENTÃO INÍCIO
                menor_idade ← idade
                tot ← 1
                FIM
            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 < 21 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
        SE cont_m2 = 0
            ENTÃO ESCREVA "Nenhum homem com experiência"
        SENÃO INÍCIO
            media_idade ← somal / cont_m2
            ESCREVA media_idade
            FIM
        SE tot_m = 0
            ENTÃO ESCREVA "Nenhum homem"
        SENÃO INÍCIO
            perc ← cont_m1 * 100 / tot_m
            ESCREVA perc
            FIM
        ESCREVA cont_f1
        ESCREVA menor_idade
    FIM_ALGORITMO.

```



1^a SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\PASCAL\EX22_A.PAS e \EXERC\CAP5\PASCAL\EX22_A.EXE

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA REPEAT:

\EXERC\CAP5\PASCAL\EX22_B.PAS e \EXERC\CAP5\PASCAL\EX22_B.EXE



1^a SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\C++\EX22_A.CPP e \EXERC\CAP5\C++\EX22_A.EXE

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\C++\EX22_B.CPP e \EXERC\CAP5\C++\EX22_B.EXE

**1^a SOLUÇÃO - UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP5\JAVA\EX22_A.java e \EXERC\CAP5\JAVA\EX22_A.class

2^a SOLUÇÃO - UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\JAVA\EX22_B.java e \EXERC\CAP5\JAVA\EX22_B.class

23. Faça um programa que receba o valor do salário mínimo, uma lista contendo a quantidade de quilowatts gasta por consumidor e o tipo de consumidor (1 – residencial, 2 – comercial ou 3 – industrial) e que calcule e mostre:

- ◆ o valor de cada quilowatt, sabendo que o quilowatt custa um oitavo do salário mínimo;
- ◆ o valor a ser pago por cada consumidor (conta final mais acréscimo). O acréscimo encontra-se na 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.

ALGORITMO**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, qtd
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^a SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP5\PASCAL\EX23_A.PAS e \EXERC\CAP5\PASCAL\EX23_A.EXE

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA REPEAT:

\EXERC\CAP5\PASCAL\EX23_B.PAS e \EXERC\CAP5\PASCAL\EX23_B.EXE

**1^a SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP5\C++\EX23_A.CPP e \EXERC\CAP5\C++\EX23_A.EXE

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\C++\EX23_B.CPP e \EXERC\CAP5\C++\EX23_B.EXE

**1^a SOLUÇÃO - UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP5\JAVA\EX23_A.java e \EXERC\CAP5\JAVA\EX23_A.class

2^a SOLUÇÃO - UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\JAVA\EX23_B.java e \EXERC\CAP5\JAVA\EX23_B.class

- 24.** 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 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 esta tabela:

SALÁRIOS	CLASSIFICAÇÃO
Até R\$ 700,00	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 !"
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.
```

**1ª SOLUÇÃO – UTILIZANDO A ESTRUTURA REPEAT:**

\EXERC\CAP5\PASCAL\EX24_A.PAS e \EXERC\CAP5\PASCAL\EX24_A.EXE

2ª SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\PASCAL\EX24_B.PAS e \EXERC\CAP5\PASCAL\EX24_B.EXE

**1^a SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:**

\EXERC\CAP5\C++\EX24_A.CPP e \EXERC\CAP5\C++\EX24_A.EXE

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\C++\EX24_B.CPP e \EXERC\CAP5\C++\EX24_B.EXE

**1^a SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:**

\EXERC\CAP5\JAVA\EX24_A.java e \EXERC\CAP5\JAVA\EX24_A.class

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:

\EXERC\CAP5\JAVA\EX24_B.java e \EXERC\CAP5\JAVA\EX24_B.class

- 25.** Faça um programa que receba os dados a seguir de vários produtos: preço unitário, país de origem (1 – Estados Unidos, 2 – México e 3 – outros), meio de transporte (T – terrestre, F – fluvial e A – aéreo), carga perigosa (S – sim, N – não), finalize a entrada de dados com um preço inválido, ou seja, menor ou igual a zero e que calcule e mostre:

- ◆ O valor do imposto, usando a tabela a seguir.

PREÇO UNITÁRIO	PERCENTUAL DE IMPOSTO SOBRE O PREÇO UNITÁRIO
Até R\$ 100,00	5%
Maior que R\$ 100,00	10%

- ◆ O valor do transporte usando a tabela a seguir.

CARGA PERIGOSA	PAÍS DE ORIGEM	VALOR DO TRANSPORTE
S	1	R\$ 50,00
	2	R\$ 21,00
	3	R\$ 24,00
N	1	R\$ 12,00
	2	R\$ 21,00
	3	R\$ 60,00

- ◆ O valor do seguro, 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, ou seja, preço unitário mais imposto mais valor do transporte mais valor do seguro.
- ◆ O total dos impostos.

ALGORITMO**SOLUÇÃO:**

```

ALGORITMO
DECLARE preco, imp, transp, seguro, 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 ← 21
        SE origem = 3
        ENTÃO transp ← 24
    FIM
SE carga = "N"
    ENTÃO INÍCIO
        SE origem = 1
        ENTÃO transp ← 12
        SE origem = 2
        ENTÃO transp ← 21
        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^a SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP5\PASCAL\EX25_A.PAS e \EXERC\CAP5\PASCAL\EX25_A.EXE

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA REPEAT:

\EXERC\CAP5\PASCAL\EX25_B.PAS e \EXERC\CAP5\PASCAL\EX25_B.EXE

**1^a SOLUÇÃO – UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP5\C++\EX25_A.CPP e \EXERC\CAP5\C++\EX25_A.EXE

2^a SOLUÇÃO – UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\C++\EX25_B.CPP e \EXERC\CAP5\C++\EX25_B.EXE

**1^a SOLUÇÃO - UTILIZANDO A ESTRUTURA WHILE:**

\EXERC\CAP5\JAVA\EX25_A.java e \EXERC\CAP5\JAVA\EX25_A.class

2^a SOLUÇÃO - UTILIZANDO A ESTRUTURA DO-WHILE:

\EXERC\CAP5\JAVA\EX25_B.java e \EXERC\CAP5\JAVA\EX25_B.class

EXERCÍCIOS PROPOSTOS

1. Faça um programa que leia cinco grupos de quatro valores (A, B, C, D) e mostre-os na ordem lida. Em seguida, organize-os em ordem crescente e decrescente.
2. Uma companhia de teatro deseja montar uma série de espetáculos. A direção calcula que, a R\$ 5,00 o ingresso, serão vendidos 120 ingressos, e que as despesas serão de R\$ 200,00. Diminuindo-se em 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, para cada novo preço de ingresso, o lucro máximo esperado, o preço do ingresso e a quantidade de ingressos vendidos para a obtenção desse lucro.
3. Faça um programa que receba a idade de 15 pessoas, 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.

FAIXA ETÁRIA	IDADE
1º	Até 15 anos
2º	De 16 a 30 anos
3º	De 31 a 45 anos
4º	De 46 a 60 anos
5º	Acima de 60 anos

4. Faça um programa que receba um número, calcule e mostre a tabuada desse número.
5. Faça um programa que mostre as tabuadas dos números de 1 a 10.
6. 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 quinze 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 juntas, sabendo-se que serão pagas em três vezes.
7. 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.
8. 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 vinte 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 metro;
 - ◆ a percentagem de pessoas com olhos azuis entre todas as pessoas analisadas;
 - ◆ a quantidade de pessoas ruivas e que não possuem olhos azuis.

9. Faça um programa que receba dez idades, pesos e alturas, 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 metro;
- ◆ a percentagem de pessoas com idade entre 10 e 30 anos entre as pessoas que medem mais de 1,90 metro.

10. Faça um programa que receba dez números, calcule e mostre a soma dos números pares e dos números primos.

11. 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:

- ◆ o preço final para compra à vista tem desconto de 20%;
- ◆ a quantidade de parcelas pode ser: 6, 12, 18, 24, 30, 36, 42, 48, 54 e 60;
- ◆ os percentuais de acréscimo encontram-se na 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%

12. Faça um programa que receba dez números inteiros e mostre a quantidade de números primos dentre os números que foram digitados.

13. Faça um programa que receba a idade e o peso de quinze pessoas, e que 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 de 31 anos para cima.

14. Cada espectador de um cinema respondeu a um questionário no qual constava sua idade e 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, calcule e mostre:

- ◆ a média das idades das pessoas que responderam ótimo;
- ◆ a quantidade de pessoas que responderam regular;
- ◆ a percentagem de pessoas que responderam bom, entre todos os espectadores analisados.

15. Uma empresa 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 responderam sim;
- ◆ o número de pessoas que responderam não;

- ◆ o número de mulheres que responderam sim;
- ◆ a percentagem de homens que responderam não, entre todos os homens analisados.

16. Faça um programa que receba várias idades, calcule e mostre a média das idades digitadas. Finalize digitando idade igual a zero.

17. Foi feita uma pesquisa sobre a audiência de canal de TV em várias casas de uma cidade, em 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:

- ◆ leia um número indeterminado de dados (número do canal e número de pessoas que estavam assistindo);
- ◆ calcule e mostre a percentagem de audiência de cada canal.

Para encerrar a entrada de dados, digite o número do canal ZERO.

18. 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 média dos salários do grupo;
- ◆ a maior e a menor idade do grupo;
- ◆ a quantidade de mulheres com salário até R\$ 200,00;
- ◆ a idade e o sexo da pessoa que possui o menor salário.

Finalize a entrada de dados ao ser digitada uma idade negativa.

19. 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'.

20. 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. Neste caso, o programa deverá mostrar uma mensagem.

21. 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 e, para códigos inválidos, o programa deverá mostrar uma mensagem.

22. Elabore um programa que receba a idade e a altura de várias pessoas, calcule e mostre a média das alturas daquelas com mais de 50 anos. Para encerrar a entrada de dados, digite idade menor ou igual a zero.

23. 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. Verifique a possibilidade de opção inválida e não se preocupe 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\$ 210,00	15%
De R\$ 210,00 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 um terço do salário.

Na opção 3: receber o salário de um funcionário e o número de meses de trabalho na empresa, no máximo doze, 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.

24. Faça um programa que receba um conjunto de valores inteiros e positivos, 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.

25. 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 do investimento e o valor investido, e que calcule e mostre o rendimento mensal de acordo com o tipo do investimento. No final, o programa deverá mostrar o total investido e o total de juros pagos.

A leitura terminará quando o código do cliente digitado for menor ou igual a 0.

VETORES

6.1 VETOR EM ALGORITMOS

6.1.1 DEFINIÇÃO DE VETOR

Vetor também é conhecido como variável composta homogênea unidimensional. Isto quer dizer que se trata de um conjunto de variáveis de mesmo tipo, que possuem o mesmo identificador (nome) e são alocadas seqüencialmente na memória. Como as variáveis têm o mesmo nome, o que as distingue é um índice que referencia sua localização dentro da estrutura.

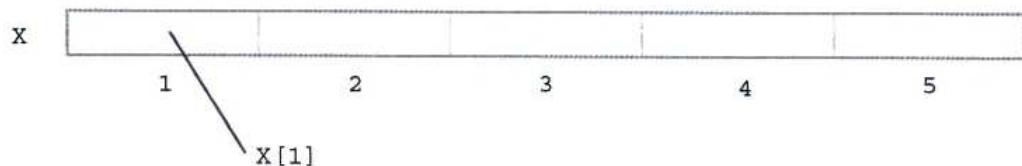
6.1.2 DECLARAÇÃO DE VETOR

```
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; tipo é o tipo básico dos dados que serão armazenados no vetor.

6.1.3 EXEMPLO DE VETOR

```
DECLARE X [5] NUMÉRICO
```



6.1.4 ATRIBUINDO VALORES AO VETOR

As atribuições em vetor exigem que seja informada em qual de suas posições o valor ficará armazenado.

```
X [1] ← 45
```

No exemplo, o número 45 será armazenado na posição de índice 1 do vetor.

```
X [4] ← 0
```

No exemplo, o número 0 será armazenado na posição de índice 4 do vetor.

6.1.5 PREENCHENDO UM VETOR

Preencher um vetor significa atribuir valores a todas as suas posições. Assim, deve-se implementar um mecanismo que controle o valor do índice.

```

PARA i ← 1 ATÉ 5 FAÇA
INÍCIO
  ESCREVA "Digite o ", i, "º número"
  LEIA X[i]
FIM
  
```

Nesse exemplo, a estrutura de repetição PARA foi utilizada para garantir que a variável i assuma todos os valores possíveis para o índice do vetor. Assim, para cada execução da repetição, será utilizada uma posição diferente do vetor.

Simulação:

		MEMÓRIA					TELA
i = 1	X	95					Digite o primeiro número
		1	2	3	4	5	95
i = 2	X	95	13				Digite o segundo número
		1	2	3	4	5	13
i = 3	X	95	13	-25			Digite o terceiro número
		1	2	3	4	5	-25
i = 4	X	95	13	-25	47		Digite o quarto número
		1	2	3	4	5	47
i = 5	X	95	13	-25	47	0	Digite o quinto número
		1	2	3	4	5	0

6.1.6 MOSTRANDO OS ELEMENTOS DO VETOR

Mostrar os valores contidos em um vetor também implica a utilização do índice.

```

PARA i ← 1 ATÉ 5 FAÇA
INÍCIO
  ESCREVA "Este é o ", i, "º número do vetor"
  ESCREVA X[i]
FIM
  
```

Nesse exemplo, a estrutura de repetição PARA foi utilizada para garantir que a variável i assuma todos os valores possíveis para o índice do vetor. Assim, para cada execução da repetição, será utilizada uma posição diferente e, dessa forma, todos os valores do vetor serão mostrados.

6.2 VETOR EM PASCAL

6.2.1 DEFINIÇÃO DE VETOR

As variáveis compostas homogêneas unidimensionais (vetores) são conhecidas na linguagem Pascal como ARRAY. Todas as posições do ARRAY possuem o mesmo identificador (mesmo nome) e são alocadas seqüencialmente na memória.

6.2.2 DECLARAÇÃO DE VETOR

VAR nome da variável: ARRAY[índice inicial..índice final] OF tipo dos dados do vetor;

onde: nome da variável é o nome da variável do tipo vetor; índice inicial é o número correspondente ao índice da primeira posição do vetor; índice final é o número correspondente ao índice da última posição do vetor; tipo dos dados do vetor é o tipo básico dos dados que serão armazenados no vetor.

OBSERVAÇÃO O valor do índice inicial deve, obrigatoriamente, ser maior ou igual ao valor do índice final. As posições serão numeradas com valores inteiros dentro desse intervalo.

Exemplo 1:

```
VAR vetor1: ARRAY [1..10] OF INTEGER
```

Nesse caso, o índice poderá assumir valores inteiros que vão de 1 até 10.

Exemplo 2:

```
VAR vetor1: ARRAY [5..9] OF REAL
```

Neste caso, o índice poderá assumir valores inteiros que vão de 5 até 9.

6.2.3 EXEMPLO DE VETOR

```
VAR X:ARRAY[1..10] OF REAL;
```

X	10.5	20	13.1	14.65	87	1.2	35.6	78.2	15	65.9
	1	2	3	4	5	6	7	8	9	10

```
VAR VET: ARRAY[5..9] OF CHAR;
```

X	E	*	m	J	k
	5	6	7	8	9

6.2.4 ATRIBUINDO VALORES AO VETOR

As atribuições em vetor exigem que seja informada em qual de suas posições o valor ficará armazenado.

```
X[4]:=5;      atribui o valor 5 à posição do vetor cujo índice é 4
VET[3]:='F'; atribui a letra F à posição do vetor cujo índice é 3
```

6.2.5 PREENCHENDO UM VETOR

Preencher um vetor significa atribuir valores a todas as suas posições. Assim, deve-se implementar um mecanismo que controle o valor do índice.

```
FOR i:= 1 TO 7 DO
BEGIN
  READLN(X[i]);
END;
```

Nesse exemplo, a estrutura de repetição FOR foi utilizada para garantir que a variável i assuma todos os valores possíveis para o índice do vetor (de 1 a 7). Assim, para cada execução da repetição, será utilizada uma posição diferente do vetor.

6.2.6 MOSTRANDO OS ELEMENTOS DO VETOR

Mostrar os valores contidos em um vetor também exige a utilização do índice.

```
FOR i:=1 TO 10 DO
BEGIN
  WRITELN(X[i]);
END;
```

Nesse exemplo, a estrutura de repetição `FOR` foi utilizada para garantir que a variável `i` assuma todos os valores possíveis para o índice do vetor (de 1 a 10). Assim, para cada execução da repetição, será utilizada uma posição diferente do vetor e, dessa forma, todos os valores do vetor serão mostrados.

6.3 VETOR EM C/C++

6.3.1 DEFINIÇÃO DE VETOR

As variáveis compostas homogêneas unidimensionais (vetores) são capazes de armazenar vários valores. Cada um desses valores é identificado pelo mesmo nome (o nome dado ao vetor). Eles são diferenciados 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.

6.3.2 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.

6.3.3 EXEMPLO DE VETOR

```
int vet[10];
```

vet	10	5	3	8	1	19	44	21	2	7
	0	1	2	3	4	5	6	7	8	9

Nesse exemplo, 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.

```
char x[5];
```

x	A	*	2	@	k
	0	1	2	3	4

Nesse exemplo, 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` (que será visto em detalhes no Capítulo 8), como ocorre na linguagem Pascal. Dessa maneira, para poder armazenar em uma cadeia de caracteres, por exemplo, o nome completo de uma pessoa, deve-se declarar um vetor de `char`, em que cada posição equivale a um caractere ou a uma letra do nome. É importante lembrar, entretanto, que toda vez que se faz 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`).

6.3.4 ATRIBUINDO VALORES AO VETOR

As atribuições em vetor exigem que seja informada em qual de suas posições o valor ficará armazenado. Deve-se lembrar sempre que a primeira posição de um vetor em C/C++ tem índice 0.

```
vet[0] = 1; atribui o valor 1 à primeira posição do vetor (lembre-se que o vetor começa na posição 0)
x[3] = 'b'; atribui a letra b à quarta posição do vetor (lembre-se que o vetor começa na posição 0)
```

6.3.5 PREENCHENDO UM VETOR

Preencher um vetor significa atribuir valores a todas as suas posições. Assim, deve-se implementar um mecanismo que controle o valor do índice.

```
for (i=0; i<10; i++)
    cin >> vetor[i];
```

Nesse exemplo, a estrutura de repetição FOR foi utilizada para garantir que a variável *i* assuma todos os valores possíveis para o índice do vetor (de 0 a 9). Assim, para cada execução da repetição, será utilizada uma posição diferente do vetor.

6.3.6 MOSTRANDO OS ELEMENTOS DO VETOR

Mostrar os valores contidos em um vetor também exige a utilização do índice.

```
for (i=0; i<10; i++)
    cout << vetor[i];
```

Nesse exemplo, a estrutura de repetição FOR foi utilizada para garantir que a variável *i* assuma todos os valores possíveis para o índice do vetor (de 0 a 9). Assim, para cada execução da repetição, será utilizada uma posição diferente e, dessa forma, todos os valores do vetor serão mostrados.

6.4 VETOR EM JAVA

6.4.1 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). Eles são diferenciados apenas por um índice.

Os índices utilizados na linguagem JAVA para identificar as posições de um vetor começam sempre em 0 (zero) e vão até o tamanho do vetor menos uma unidade.

6.4.2 DECLARAÇÃO DE VETOR

Os vetores em JAVA são definidos pela existência de colchetes vazios antes ou depois do nome da variável no momento da declaração. Logo depois, deve ser feito o dimensionamento do vetor.

6.4.3 EXEMPLO DE VETOR

Nos exemplos a seguir são utilizadas duas linhas de comando: a primeira declara um vetor e a segunda define o seu tamanho.

Exemplo 1:

Na primeira linha deste exemplo, os colchetes vazios após o nome definem que *x* será um vetor. O tipo int determina que todas as suas posições armazenarão valores inteiros. A segunda linha determina que o vetor *x* terá tamanho 10 (ou seja, das posições 0 a 9).

```
int x[];
x = new int[10];
```

x	10	5	3	8	1	19	44	21	2	7
	0	1	2	3	4	5	6	7	8	9

Exemplo 2:

Na primeira linha deste exemplo, os colchetes vazios antes do nome definem que `y` será um vetor. O tipo `float` determina o tipo do número que poderá ser armazenado em todas as suas posições. A segunda linha determina que o vetor `y` terá tamanho 6 (ou seja, das posições 0 a 5).

```
float []y;
y = new float[6];
```

y	1.5	8.9	3.0	4.7	15.3	16.0	
	0	1	2	3	4	5	

Já nos exemplos apresentados a seguir, utilizou-se a forma condensada, onde a declaração e o dimensionamento de um vetor são feitos utilizando-se uma única linha.

Exemplo 3:

Este exemplo define e dimensiona o vetor `x` utilizando uma única linha. Assim, a parte que antecede o sinal de igual informa que `x` é um vetor e que poderá armazenar números inteiros. A parte que sucede o sinal de igual dimensiona o tamanho de `x` em 8 (das posições 0 a 7).

```
int x[] = new int[8];
```

x	5	9	1	14	25	3	18	0
	0	1	2	3	4	5	6	7

Exemplo 4:

Este exemplo define e dimensiona o vetor `y` utilizando uma única linha. Assim, a parte que antecede o sinal de igual informa que `y` é um vetor e que poderá armazenar qualquer caractere. A parte que sucede o sinal de igual dimensiona o tamanho de `y` em 5 (das posições 0 a 4).

```
char []y = new char[5];
```

y	A	*	2	@	k	
	0	1	2	3	4	

6.4.4 ATRIBUINDO VALORES AO VETOR

As atribuições em vetor exigem que seja informada em qual de suas posições o valor ficará armazenado. Deve-se lembrar sempre que a primeira posição de um vetor em JAVA tem índice 0.

```
vet[0] = 1;   atribui o valor 1 à primeira posição do vetor (lembre-se que o vetor começa na posição 0)
x[3] = 'b';  atribui a letra b à quarta posição do vetor (lembre-se que o vetor começa na posição 0)
```

6.4.5 PREENCHENDO UM VETOR

Preencher um vetor significa atribuir valores a todas as suas posições. Assim, deve-se implementar um mecanismo que controle o valor do índice.

```
e = new Scanner(System.in);
for (i=0; i<10; i++)
    vet[i] = e.nextInt();
```

Nesse exemplo, a estrutura de repetição FOR foi utilizada para garantir que a variável `i` assuma todos os valores possíveis para o índice do vetor (de 0 a 9). Assim, para cada execução da repetição, uma posição diferente do vetor será utilizada.

6.4.6 MOSTRANDO OS ELEMENTOS DO VETOR

Mostrar os valores contidos em um vetor também implica a utilização do índice.

```
for (i=0; i<10; i++)
    System.out.println(vet[i]);
```

Nesse exemplo, a estrutura de repetição **FOR** foi utilizada para garantir que a variável *i* assuma todos os valores possíveis para o índice do vetor (de 0 a 9). Assim, para cada execução da repetição, será utilizada uma posição diferente e, dessa forma, todos os valores do vetor serão mostrados.

EXERCÍCIOS RESOLVIDOS

-  1. Faça um programa que preencha um vetor com nove números 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\CAP6\PASCAL\EX1.PAS e \EXERC\CAP6\PASCAL\EX1.EXE



SOLUÇÃO:

\EXERC\CAP6\C++\EX1.CPP e \EXERC\CAP6\C++\EX1.EXE



SOLUÇÃO:

\EXERC\CAP6\JAVA\EX1.java e \EXERC\CAP6\JAVA\EX1.class

 2. Uma pequena loja de artesanato possui apenas um vendedor e comercializa dez tipos de objetos. O vendedor recebe, mensalmente, salário de R\$ 400,00, acrescido de 5% do valor total de suas vendas. O valor unitário dos objetos deve ser informado e armazenado em um vetor; a quantidade vendida de cada peça deve ficar em outro vetor, mas na mesma posição. Crie um programa que receba os preços e as quantidades vendidas, armazenando-os em seus respectivos vetores (ambos com tamanho dez). Depois, determine e mostre:

- ◆ um relatório contendo quantidade vendida, valor unitário e valor total de cada objeto. Ao final, deverá ser mostrado o valor geral das vendas e o valor da comissão que será paga ao vendedor;
- ◆ o valor do objeto mais vendido e sua posição no vetor (não se preocupe com empates).

ALGORITMO**SOLUÇÃO:**

```

ALGORITMO
DECLARE qtd[10], preco[10] NUMÉRICO
    i, tot_geral, tot_vend, comissao, maio, ind 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 qtd[i], preço[i], tot_vend
        tot_geral ← tot_geral + tot_vend
    FIM
comissão ← tot_geral * 5 /100
ESCREVA tot_geral, comissao
maior ← qtd[1]
ind ← 1
PARA i ← 2 ATÉ 10 FAÇA
    INÍCIO
        SE qtd[i] > maior
        ENTÃO INÍCIO
            maior ← qtd[i]
            ind ← i
        FIM
    FIM
    ESCREVA preco [ind], ind
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

**SOLUÇÃO:**

\EXERC\CAP6\JAVA\EX2.java e \EXERC\CAP6\JAVA\EX2.class

 3. Faça um programa que preencha dois vetores de dez elementos numéricos cada um e mostre o vetor resultante da intercalação deles.

Vetor 1	3	5	4	2	2	5	3	2	5	9
	1	2	3	4	5	6	7	8	9	10

Vetor 2	7	15	20	0	18	4	55	23	8	6
	1	2	3	4	5	6	7	8	9	10

Vetor resultante da intercalação

3	7	5	15	4	20	2	0	2	18	5	4	3	55	2	23	5	8	9	6
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\CAP6\PASCAL\EX3.PAS e \EXERC\CAP6\PASCAL\EX3.EXE



SOLUÇÃO:

\EXERC\CAP6\C++\EX3.CPP e \EXERC\CAP6\C++\EX3.EXE



SOLUÇÃO:

\EXERC\CAP6\JAVA\EX3.java e \EXERC\CAP6\JAVA\EX3.class

4. Faça um programa que preencha um vetor com oito números inteiros, calcule e mostre dois vetores resultantes. O primeiro vetor resultante deve conter os números positivos; o segundo deve conter os números negativos. Cada vetor resultante vai ter, no máximo, oito posições, que poderão não ser completamente utilizadas.

ALGORITMO

SOLUÇÃO:

```

ALGORITMO
DECLARE num[8], pos[8], neg[8] NUMÉRICO
```

```

        cont, cont_n, cont_p, i 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\CAP6\PASCAL\EX4.PAS e \EXERC\CAP6\PASCAL\EX4.EXE



SOLUÇÃO:

\EXERC\CAP6\C++\EX4.CPP e \EXERC\CAP6\C++\EX4.EXE



SOLUÇÃO:

\EXERC\CAP6\JAVA\EX4.java e \EXERC\CAP6\JAVA\EX4.class

5. Faça um programa que preencha dois vetores, X e Y, com dez números inteiros cada. Calcule e mostre os seguintes vetores resultantes:

- ◆ A união de X com Y
(todos os elementos de X e de Y sem repetições).

X	3	8	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	5	6
	1	2	3	4	5	6	7	8	9	10

União	3	8	4	2	1	6	7	11	9	5	12	0
	1	2	3	4	5	6	7	8	9	10	11	12

◆ A diferença entre X e Y

(todos os elementos de X que não existam em Y, sem repetições).

X	3	8	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	5	6
	1	2	3	4	5	6	7	8	9	10

Diferença	8	7	11	9
	1	2	3	4

◆ A soma entre X e Y

(soma de cada elemento de X com o elemento de mesma posição em Y).

X	3	8	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	5	6
	1	2	3	4	5	6	7	8	9	10

Soma	5	9	9	14	4	6	9	11	16	15
	1	2	3	4	5	6	7	8	9	10

◆ O produto entre X e Y

(multiplicação de cada elemento de X com o elemento de mesma posição em Y).

X	3	8	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	5	6
	1	2	3	4	5	6	7	8	9	10

Produto	6	8	20	24	3	0	8	28	55	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, sem repetições).

X	3	8	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	5	6
	1	2	3	4	5	6	7	8	9	10

Interseção	3	4	2	1	6
	1	2	3	4	5

ALGORITMO**SOLUÇÃO:**

```

ALGORITMO
DECLARE X[10], Y[10], U[20], D[10], S[10], P[10], IT[10] NUMÉRICO
    i, j, k, 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 (j < cont_u E X[i] ≠ U[j]) FAÇA
            INÍCIO
                j ← j + 1
            FIM
            SE j >= cont_u
                ENTÃO INÍCIO
                    U[cont_u] ← X[i]
                    cont_u ← cont_u + 1
                FIM
            FIM
        PARA i ← 1 ATÉ 10 FAÇA
            INÍCIO
                j ← 1
                ENQUANTO (j < cont_u E Y[i] ≠ U[j]) FAÇA
                    INÍCIO
                        j ← j + 1
                    FIM
                    SE j >= cont_u
                        ENTÃO INÍCIO
                            U[cont_u] ← Y[i]
                            cont_u ← cont_u + 1
                        FIM
                    FIM
                PARA i ← 1 ATÉ cont_u -1 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
                                    k ← 1

```

```

        ENQUANTO (k < cont_d E X[i] ≠ D[k]) FAÇA
            INÍCIO
                k ← k + 1
            FIM
            SE k >= cont_d
                ENTÃO INÍCIO
                    D[cond_d] ← X[i]
                    cont_d ← cont_d + 1
                FIM
            FIM
        FIM

PARA i ← 1 ATÉ cont_d - 1 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 (j <= 10 E X[i] ≠ Y[j]) FAÇA
            INÍCIO
                j ← j + 1
            FIM
            SE j <= 10
                ENTÃO INÍCIO
                    k ← 1
                    ENQUANTO(k < cont_i E IT[k] ≠ X[i])FAÇA
                        INICIO
                            k ← k + 1
                        FIM
                        SE k >= cont_i
                            ENTÃO INÍCIO
                                IT[cont_i] ← X[i]
                                cont_i ← cont_i + 1
                            FIM
                        FIM
                    FIM
                FIM
            FIM
        FIM
    FIM
PARA i ← 1 ATÉ cont_i FAÇA
    INÍCIO
        ESCREVA IT[i]
    FIM
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

**SOLUÇÃO:**

\EXERC\CAP6\JAVA\EX5.java e \EXERC\CAP6\JAVA\EX5.class

6. Faça um programa que preencha um vetor com dez números inteiros, calcule e mostre o vetor resultante de uma ordenação 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
                SE vet[j] < vet[j+1]
                ENTÃO INÍCIO
                    aux ← vet[j]
                    vet[j] ← vet[j+1]
                    vet[j+1] ← aux
                FIM
            FIM
        FIM
    FIM
PARA i ← 1 ATÉ 10 FAÇA
    INÍCIO
        ESCREVA vet[i]
    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

**SOLUÇÃO:**

\EXERC\CAP6\JAVA\EX6.java e \EXERC\CAP6\JAVA\EX6.class

7. Faça um programa que, no momento de preencher um vetor com oito números inteiros, já os armazene de forma crescente.

ALGORITMO**SOLUÇÃO:**

```

ALGORITMO
DECLARE vet[8], i, j, z, aux NUMÉRICO
i ← 1
ENQUANTO (i <= 8) FAÇA
INÍCIO
LEIA aux
j ← 1
ENQUANTO (j < i E vet[j] < aux) 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^a SOLUÇÃO – UTILIZANDO APENAS WHILE:**

\EXERC\CAP6\PASCAL\EX7_A.PAS e \EXERC\CAP6\PASCAL\EX7_A.EXE

2^a SOLUÇÃO – UTILIZANDO FOR E WHILE:

\EXERC\CAP6\PASCAL\EX7_B.PAS e \EXERC\CAP6\PASCAL\EX7_B.EXE

**1^a SOLUÇÃO – UTILIZANDO APENAS WHILE:**

\EXERC\CAP6\C++\EX7_A.CPP e \EXERC\CAP6\C++\EX7_A.EXE

2^a SOLUÇÃO – UTILIZANDO FOR E WHILE:

\EXERC\CAP6\C++\EX7_B.CPP e \EXERC\CAP6\C++\EX7_B.EXE

**1^a SOLUÇÃO – UTILIZANDO APENAS WHILE:**

\EXERC\CAP6\JAVA\EX7_A.java e \EXERC\CAP6\JAVA\EX7_A.class

2^a SOLUÇÃO – UTILIZANDO FOR E WHILE:

\EXERC\CAP6\JAVA\EX7_B.java e \EXERC\CAP6\JAVA\EX7_B.class

- 8.** Faça um programa que preencha dois vetores com cinco elementos numéricos cada e depois ordene os de maneira crescente. Deverá ser gerado um terceiro vetor com dez posições, composto pela junção dos elementos dos vetores anteriores, também ordenado de maneira crescente.

X	6	8	1	10	3
	1	2	3	4	5

X	1	3	6	8	10					
Ordenado	1	2	3	4	5					
Y	20	0	7	2	5					
	1	2	3	4	5					
Y	0	2	5	7	20					
Ordenado	1	2	3	4	5					
Resultado	0	1	2	3	5	6	7	8	10	20
	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
        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
        FIM
    FIM
j ← 1;
PARA i ← 1 ATÉ 5 FAÇA
    INÍCIO
        R[j] ← X[i]
        j ← j + 1
        R[j] ← Y[i]
        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\CAP6\PASCAL\EX8.PAS e \EXERC\CAP6\PASCAL\EX8.EXE

**SOLUÇÃO:**

\EXERC\CAP6\C++\EX8.CPP e \EXERC\CAP6\C++\EX8.EXE

**SOLUÇÃO:**

\EXERC\CAP6\JAVA\EX8.java e \EXERC\CAP6\JAVA\EX8.class

-  9. Faça um programa que efetue reserva de passagens aéreas de uma companhia. O programa deverá ler informações sobre os vôos (número, origem e destino) e o número de lugares disponíveis para doze aviões (um vetor para cada um desses dados). Depois da leitura, 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 em que 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 encerrar a execução 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 <= 12 E voo[i] ≠ num_voo) FAÇA
                        INÍCIO
                            i ← i + 1
                        FIM
                    SE i > 12
                        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 12 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
                    FIM
            SE op2 = 3
                ENTÃO INÍCIO
                    ESCREVA "Digite o local de destino: "
                    LEIA local
                    PARA i ← 1 ATÉ 12 FAÇA

```

```

INÍCIO
  SE local = destino[i]
    ENTÃO INÍCIO
      ESCREVA "Número do voo: ", 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 voo desejado: "
  LEIA num_voo
  i ← 1
  ENQUANTO (i <= 12 E voo[i] = num_voo) FAÇA
    INÍCIO
      i = i + 1
    FIM
  SE i > 12
    ENTÃO ESCREVA "Número de voo 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.

```

OBSERVAÇÃO A comparação de duas cadeias de caracteres (como dois nomes, por exemplo) em JAVA é feita utilizando-se alguns métodos da classe String (que serão vistos em detalhes no Capítulo 8):

- ◆ `equals()` – verifica se duas cadeias são iguais, mas é sensível ao caso (ou seja, considera que uma letra em maiúsculo é diferente da mesma letra em minúsculo).
Exemplo: `nome1.equals(nome2);` //verifica se o nome 1 é igual ao nome 2.
- ◆ `equalsIgnoreCase()` – verifica se duas cadeias são iguais e não faz distinção entre letras maiúsculas e minúsculas.

Exemplo: `nome1.equalsIgnoreCase(nome2);` //verifica se o nome 1 é igual ao nome 2.



SOLUÇÃO:

\EXERC\CAP6\PASCAL\EX9.PAS e \EXERC\CAP6\PASCAL\EX9.EXE



SOLUÇÃO:

\EXERC\CAP6\C++\EX9.CPP e \EXERC\CAP6\C++\EX9.EXE



SOLUÇÃO:

\EXERC\CAP6\JAVA\EX9.java e \EXERC\CAP6\JAVA\EX9.class

 10. Faça um programa para corrigir provas de múltipla escolha. Cada prova tem oito questões e cada questão vale um ponto. O primeiro conjunto de dados a ser lido é o gabarito da prova. Os outros dados são os números dos alunos e as respostas que deram às questões. Existem dez alunos matriculados. Calcule e mostre:

- ◆ o número e a nota de cada aluno;
- ◆ a percentagem de aprovação, sabendo-se que a nota mínima é 6.

ALGORITMO**SOLUÇÃO:**

```

ALGORITMO
DECLARE gabarito[8], resposta 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 gabarito[i]
    FIM
tot_ap ← 0
PARA i ← 1 ATÉ 10 FAÇA
    INÍCIO
        ESCREVA "Digite o número do ", i, ".º aluno"
        LEIA num
        pontos ← 0
        PARA j ← 1 ATÉ 8 FAÇA
            INÍCIO
                ESCREVA "Digite a resposta dada pelo aluno ", num, " à ", j, ".ª questão"
                LEIA resposta[j]
                SE resposta[j] = gabarito[j]
                ENTÃO pontos ← pontos + 1
            FIM
        ESCREVA "A nota do aluno ", num, " foi ", pontos
        SE pontos >= 6
            ENTÃO tot_ap ← tot_ap + 1
        FIM
    FIM
    perc_ap ← tot_ap * 100 / 10
    ESCREVA "O percentual de alunos aprovados é ", perc_ap
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

**SOLUÇÃO:**

\EXERC\CAP6\JAVA\EX10.java e \EXERC\CAP6\JAVA\EX10.class

 11. Faça um programa que receba a temperatura média de cada mês do ano, armazenando-as em um vetor. Calcule e mostre a maior e a menor temperatura do ano e em que mês ocorreram (mostrar o mês por extenso: 1 – janeiro, 2 – fevereiro...). Desconsidere 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\CAP6\PASCAL\EX11.PAS e \EXERC\CAP6\PASCAL\EX11.EXE

**SOLUÇÃO:**

\EXERC\CAP6\C++\EX11.CPP e \EXERC\CAP6\C++\EX11.EXE

**SOLUÇÃO:**

\EXERC\CAP6\JAVA\EX11.java e \EXERC\CAP6\java\EX11.class

12. Faça um programa que preencha um vetor com os modelos de cinco carros (exemplos de modelos: Fusca, Gol, Vectra etc.). Carregue outro vetor com o consumo desses carros, isto é, quantos quilômetros cada um deles faz com um litro de combustível, calcule e mostre:

- ◆ o modelo de carro mais econômico;
- ◆ quantos litros de combustível cada um dos carros cadastrados consome para percorrer uma distância de 1.000 quilômetros.

ALGORITMO**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[i]
                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 " O veículo " , veiculo[i], " consome " , valor, " litro(s) de
        ➔ combustível para percorrer 1000 Km"
    FIM

```

ESCREVA "O veículo mais econômico é ", veiculo[menor_vei]
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

**SOLUÇÃO:**

\EXERC\CAP6\java\EX12.java e \EXERC\CAP6\java\EX12.class

13. Faça um programa que preencha um vetor com dez números inteiros, calcule e mostre os números superiores a cinqüenta e suas respectivas posições. O programa deverá mostrar mensagem se não existir nenhum número nessa condição.

**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 existem números superiores a 50 no vetor"
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

**SOLUÇÃO:**

\EXERC\CAP6\JAVA\EX13.java e \EXERC\CAP6\JAVA\EX13.class

14. Faça um programa que preencha três vetores com cinco posições cada. O primeiro vetor receberá os nomes de cinco funcionários; o segundo e o terceiro vetor receberão, respectivamente, o salário e o

tempo de serviço de cada um. 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\$ 400,00. Sabe-se ainda que, se o funcionário satisfizer às duas condições anteriores, tempo de serviço e salário, o aumento será de 35%; para o funcionário que satisfizer apenas à condição tempo de serviço, o aumento será de 25%; para aquele que satisfizer apenas à condição salário, o aumento será de 15%.

ALGORITMO**SOLUÇÃO:**

```

ALGORITMO
  DECLARE nome[5] LITERAL
    sal[5], quant[5], 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] >= 400)
      ENTÃO ESCREVA nome[i]
    FIM
  PARA i ← 1 ATÉ 5 FAÇA
    INÍCIO
    SE (quant[i] > 5) OU (sal[i] < 400)
      ENTÃO INÍCIO
        SE (quant[i] > 5) E (sal[i] < 400)
          ENTÃO novo_sal ← sal[i] + sal[i] * 35 / 100
        SENÃO SE (quant[i] > 5)
          ENTÃO novo_sal ← sal[i] + sal[i] * 25 / 100
        SENÃO novo_sal ← sal[i] + sal[i] * 15 / 100
        ESCREVA nome[i], novo_sal
      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

**SOLUÇÃO:**

\EXERC\CAP6\JAVA\EX14.java e \EXERC\CAP6\JAVA\EX14.class

15. Faça um programa que preencha um primeiro vetor com dez números inteiros e um segundo vetor com cinco números inteiros. O programa deverá mostrar uma lista dos números do primeiro vetor com seus respectivos divisores armazenados no segundo vetor, bem como 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 possui divisores no 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, deve-se 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 i ← 1 ATÉ 5 FAÇA
    INÍCIO
    LEIA vet2[i]
    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 "Divisível por ", vet2[j], "na posição ", j
        achou ← verdadeiro
      FIM
    FIM
    SE achou = falso
    ENTÃO ESCREVA "Não possui divisores no segundo vetor"
    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

**SOLUÇÃO:**

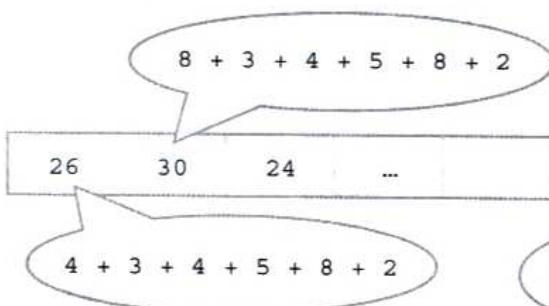
\EXERC\CAP6\JAVA\EX15.java e \EXERC\CAP6\JAVA\EX15.class

16. Faça um programa que preencha 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 será composto pelos números ímpares gerados pelo elemento do primeiro vetor somado a todos os elementos do segundo vetor.

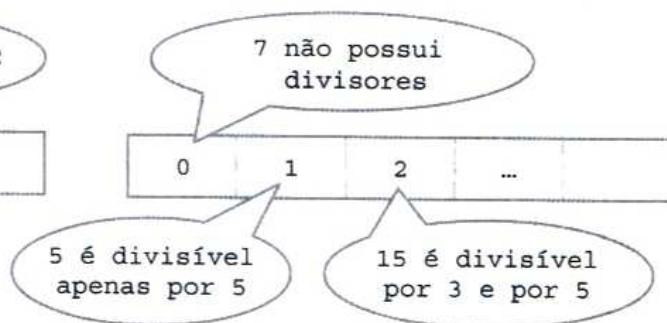
Primeiro vetor	4 7 5 8 2 15 9 6 10 11
	1 2 3 4 5 6 7 8 9 10

Segundo vetor	3 4 5 8 2
	1 2 3 4 5

Primeiro vetor resultante



Segundo vetor resultante

**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
    vet_result1[i] ← soma
    FIM
    poslivre1 ← poslivre1 + 1
    poslivre2 ← poslivre2 + 1
    FIM
    vet_result2[i] ← soma
    FIM
FIM

```

```

    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[i]
    FIM
PARA i ← 1 ATÉ (poslivre2 -1) FAÇA
    INÍCIO
        ESCREVA vet_result2[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

**SOLUÇÃO:**

\EXERC\CAP6\JAVA\EX16.java e \EXERC\CAP6\JAVA\EX16.class

17. Faça um programa que receba seis números inteiros e mostre:

- ◆ os números pares digitados;
- ◆ a soma dos números pares digitados;
- ◆ os números ímpares digitados;
- ◆ a quantidade de números ímpares digitados.

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 de 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 "Nenhum número par foi digitado"
    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 "Nenhum número ímpar foi digitado"
      SENÃO ESCREVA qtde
    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

SOLUÇÃO:

\EXERC\CAP6\JAVA\EX17.java e \EXERC\CAP6\JAVA\EX17.class

18. Faça um programa que receba o número sorteado por um dado em vinte jogadas, mostre os números sorteados e a freqüência com que apareceram.

ALGORITMO 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\CAP6\PASCAL\EX18.PAS e \EXERC\CAP6\PASCAL\EX18.EXE

SOLUÇÃO:

\EXERC\CAP6\C++\EX18.CPP e \EXERC\CAP6\C++\EX18.EXE

SOLUÇÃO:

\EXERC\CAP6\JAVA\EX18.java e \EXERC\CAP6\JAVA\EX18.class

-  19. Faça um programa que preencha dois vetores, A e B, com vinte caracteres cada. A seguir, troque o 1º elemento de A com o 20º de B, o 2º de A com o 19º de B, e 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 i ← 1 ATÉ 20 FAÇA
  INÍCIO
  LEIA vet2[i]
  FIM
PARA i ← 1 ATÉ 20 FAÇA
  INÍCIO
  ESCREVA vet1[i]
  FIM
PARA i ← 1 ATÉ 20 FAÇA
  INÍCIO
  ESCREVA vet2[i]
  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 i ← 1 ATÉ 20 FAÇA
  INÍCIO
  ESCREVA vet2[i]
  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

SOLUÇÃO:

\EXERC\CAP6\JAVA\EX19.java e \EXERC\CAP6\JAVA\EX19.class

20. Faça um programa que leia um vetor com cinco posições para números reais e, depois, um código inteiro. Se o código for zero, finalize o programa; se for 1, mostre o vetor na ordem direta; se for 2, mostre o vetor na ordem inversa.

ALGORITMOSOLUÇÃO:

```

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 PASSO - 1 FAÇA
            INÍCIO
            ESCREVA vet[i]
            FIM
        FIM
    SE (cod < 0) OU (cod > 2)
    ENTÃO ESCREVA "Código inválido"
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

SOLUÇÃO:

\EXERC\CAP6\JAVA\EX20.java e \EXERC\CAP6\JAVA\EX20.class

21. Faça um programa que leia um conjunto de quinze 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, números ímpares. Como o tamanho dos vetores pode não ser suficiente para armazenar todos os números, deve-se sempre verificar se já estão cheios. Caso P ou I estejam cheios, deve-se mostrá-los e recomeçar o preenchimento da primeira posição. Terminado o processamento, mostre 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\CAP6\PASCAL\EX21.PAS e \EXERC\CAP6\PASCAL\EX21.EXE

SOLUÇÃO:

\EXERC\CAP6\C++\EX21.CPP e \EXERC\CAP6\C++\EX21.EXE

SOLUÇÃO:

\EXERC\CAP6\JAVA\EX21.java e \EXERC\CAP6\JAVA\EX21.class

22. Faça um programa que simule um controle bancário. Para tanto, devem ser lidos os códigos de dez contas e seus respectivos saldos. Os códigos devem ser armazenados em um vetor de números inteiros (não pode haver mais de 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 quinta posição do vetor de códigos, seu saldo deverá ficar na quinta posição do vetor de saldos. Depois de fazer a leitura dos valores, deverá aparecer 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, deverá aparecer 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, deverá aparecer a mensagem *Conta não encontrada* e voltar ao menu. Se a conta existir, verificar se 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*.

ALGORITMOSOLUÇÃO:

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
    ATÉ achou = falso
    LEIA saldo[i]
    FIM
REPITA
  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"
                    FIM
                SENÃO INÍCIO
                    saldo[i] ← saldo[i] - valor
                    ESCREVA "Saque efetuado"
                    FIM
                achou ← verdadeiro
                FIM
            FIM
            SE achou = falso
                ENTÃO ESCREVA "Conta não cadastrada"
                FIM
            SE op = 3
            ENTÃO INÍCIO
                soma ← 0
                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\CAP6\PASCAL\EX22.PAS e \EXERC\CAP6\PASCAL\EX22.EXE



SOLUÇÃO:

\EXERC\CAP6\C++\EX22.CPP e \EXERC\CAP6\C++\EX22.EXE



SOLUÇÃO:

\EXERC\CAP6\JAVA\EX22.java e \EXERC\CAP6\JAVA\EX22.class

- 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 0 representa poltrona desocupada e 1, poltrona ocupada.

Janela	0	1	0	0	...	1	0	0
	1	2	3	4	...	22	23	24
Corredor	0	0	0	1	...	1	0	0
	1	2	3	4	...	22	23	24

Inicialmente, todas as poltronas estarão livres. Depois disso, o programa deverá apresentar as seguintes opções:

- ◆ vender passagem.
- ◆ mostrar mapa de ocupação do ônibus.
- ◆ encerrar.

Quando a opção escolhida for Vender Passagem, deverá ser perguntado se o usuário deseja janela ou corredor e o número da poltrona. O programa deverá, então, dar uma das seguintes mensagens:

- ◆ Venda efetivada – se a poltrona solicitada estiver livre, marcando-a como ocupada.
- ◆ Poltrona ocupada – se a poltrona solicitada não estiver disponível para venda.
- ◆ Ônibus lotado – quando todas as poltronas já estiverem ocupadas.

Quando a opção escolhida for Mostrar Mapa de Ocupação do Ônibus, deverá ser mostrada uma listagem conforme a seguir:

JANELA	CORREDOR
1- Ocupada	1- Ocupada
2- Ocupada	2- Livre
3- Livre	3- Livre
4- Livre	4- Ocupada
5- Ocupada	5- Livre
...	

Quando for escolhida a opção Encerrar, a execução do programa deverá ser finalizada.

ALGORITMO

SOLUÇÃO:

```

ALGORITMO
  DECLARE corredor[24], janela[24] NUMÉRICO
    achou LÓGICO
    posicao LITERAL
    i, num NUMÉRICO
  PARA i ← 1 ATÉ 24 FAÇA
    INÍCIO
      corredor[i] ← 0
      janela[i] ← 0
    FIM
  REPITA
    ESCREVA "1- Vender passagem"
    ESCREVA "2- Mostrar mapa de ocupação do ônibus"
    ESCREVA "3- Encerrar"
    LEIA op
  FIMREPITA

```

```

SE op = 1
ENTÃO INÍCIO
    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
                    REPITA
                        LEIA posicao
                        ATÉ posicao = "J" OU posicao = "C"
                        REPITA
                            LEIA num
                            ATÉ num >= 1 E num <= 24
                            SE posicao = "J" E janela[num] = 1
                                ENTÃO ESCREVA "Poltrona ocupada"
                                SENÃO INÍCIO
                                    ESCREVA "Venda efetivada"
                                    janela[num] ← 1
                                FIM
                            SE posicao = "C" E corredor[num] = 1
                                ENTÃO ESCREVA "Poltrona ocupada"
                                SENÃO INÍCIO
                                    ESCREVA "Venda efetivada"
                                    corredor[num] ← 1
                                FIM
                            FIM
            FIM
        FIM
    FIM
SE op = 2
ENTÃO INÍCIO
    ESCREVA "JANELA" CORREDOR"
    PARA i ← 1 ATÉ 24 FAÇA
        INÍCIO
            SE janela[i] = 0
                ENTÃO ESCREVA i, "- Livre"
                SENÃO ESCREVA i, "- Ocupada"
            SE corredor[i] = 0
                ENTÃO ESCREVA i, "- Livre"
                SENÃO ESCREVA i, "- Ocupada"
            FIM
        FIM
    FIM
ATÉ QUE op = 3
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



SOLUÇÃO:

\EXERC\CAP6\JAVA\EX23.java e \EXERC\CAP6\JAVA\EX23.class

24. Faça um programa que leia um vetor A de dez posições contendo números inteiros. Determine e mostre, a seguir, quais elementos de A 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, 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.

ALGORITMO

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
            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
            FIM
PARA i ← 1 ATÉ cont_r - 1 FAÇA
ESCREVA "O número ",repetidos[i], " apareceu ",vezes[i]," vezes"
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

**SOLUÇÃO:**

\EXERC\CAP6\JAVA\EX24.java e \EXERC\CAP6\JAVA\EX24.class

25. Faça um programa que gere os dez primeiros números primos acima de 100 e armazene-os em um vetor. Escreva no final o vetor resultante.

ALGORITMO**SOLUÇÃO:**

```

ALGORITMO
    DECLARE primos[10] NUMÉRICO
        i, qtde, num, divisores NUMÉRICO
    num ← 101
    qtde ← 1
    REPITA
        divisores ← 0
        PARA i ← 1 ATÉ num FAÇA
            INÍCIO
            SE RESTO(num/i) = 0
            ENTÃO divisores ← divisores + 1
            FIM
        SE divisores <= 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.

```

**SOLUÇÃO:**

\EXERC\CAP6\PASCAL\EX25.PAS e \EXERC\CAP6\PASCAL\EX25.EXE

**SOLUÇÃO:**

\EXERC\CAP6\C++\EX25.CPP e \EXERC\CAP6\C++\EX25.EXE

**SOLUÇÃO:**

\EXERC\CAP6\JAVA\EX25.java e \EXERC\CAP6\JAVA\EX25.class

EXERCÍCIOS PROPOSTOS

1. Faça um programa que preencha um vetor com seis elementos numéricos inteiros, calcule e mostre:
 - ◆ todos os números pares;
 - ◆ a quantidade de números pares;
 - ◆ todos os números ímpares;
 - ◆ a quantidade de números ímpares.

2. Faça um programa que preencha 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 preencher dois vetores com dez posições cada, onde o primeiro corresponde ao código do produto e o segundo ao total desse produto em estoque. Logo após, o programa deverá ler um conjunto indeterminado de dados contendo o código de um cliente e o código do produto que ele 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 ao 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 preencha um vetor com quinze elementos inteiros e verifique a existência de elementos iguais a 30, mostrando as posições em que 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 quinze alunos. Coloque os números das matrículas dos alunos que cursam Linguagem de Programação em outro vetor, no máximo dez alunos. Mostre o número das matrículas que aparecem nos dois vetores.

6. Faça um programa que receba o total das vendas de cada vendedor de uma loja e armazene-as em um vetor. Receba também o percentual de comissão a que cada vendedor tem direito e armazene-os em outro vetor. Receba os nomes desses vendedores e armazene-os em um terceiro vetor. Existem apenas dez vendedores na loja. Calcule e mostre:

- ◆ um relatório com os nomes dos vendedores e os valores a receber referentes à comissão;
- ◆ o total das vendas de todos os vendedores;
- ◆ o maior valor a receber e o nome de quem o receberá;
- ◆ o menor valor a receber e o nome de quem o receberá.

7. Faça um programa que preencha 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 preencha um vetor com os nomes de sete alunos e carregue outro vetor com a média final desses alunos. Calcule e mostre:

- ◆ o nome do aluno com maior média (desconsiderar empates);
- ◆ para cada aluno não aprovado, isto é, com média menor do que 7, mostrar quanto esse aluno precisa tirar na prova de exame final para ser aprovado. Considerar que a média para aprovação no exame é 5.

9. Faça um programa que preencha 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, para os produtos que satisfizerem às duas condições anteriores, código e preço, o aumento será de 20%; para aqueles que satisfizerem apenas à condição de código, o aumento será de 15%; e para aqueles que satisfizerem apenas à condição de preço, o aumento será de 10%.

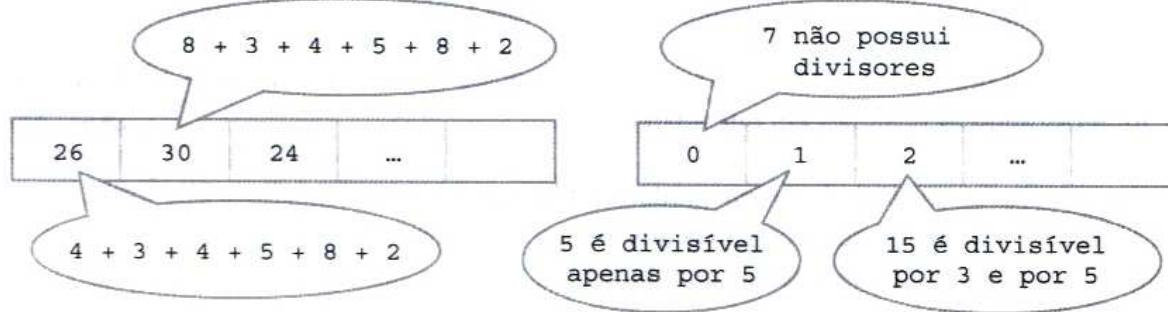
10. Faça um programa que preencha um vetor com dez números inteiros e um segundo vetor com cinco números inteiros, calcule e mostre dois vetores resultantes. O primeiro vetor resultante será composto pela soma de cada número par do primeiro vetor somado a todos os números do segundo vetor. O segundo vetor resultante será composto pela quantidade de divisores que cada número ímpar do primeiro vetor tem no segundo vetor.

Primeiro vetor	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td>4</td><td>7</td><td>5</td><td>8</td><td>2</td><td>15</td><td>9</td><td>6</td><td>10</td><td>11</td></tr> </table>	4	7	5	8	2	15	9	6	10	11
4	7	5	8	2	15	9	6	10	11		
	1 2 3 4 5 6 7 8 9 10										

Segundo vetor	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td>3</td><td>4</td><td>5</td><td>8</td><td>2</td></tr> </table>	3	4	5	8	2
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 os nomes e duas notas de seis alunos e mostre o relatório a seguir.

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 = ?
- ◆ percentual de alunos aprovados = ?%
- ◆ percentual de alunos de exame = ?%
- ◆ percentual de alunos 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 DVDs locados em 2006 por cada 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 preencha dois vetores de dez posições cada um, determine e mostre um terceiro contendo os elementos dos dois vetores anteriores ordenados de maneira decrescente.

18. Faça um programa que preencha um vetor com quinze números, determine e mostre:

- ◆ o maior número e a posição por ele ocupada no vetor;
- ◆ o menor número e a posição por ele ocupada no vetor.

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 com cinqüenta posições para números inteiros e mostre somente os números positivos.

21. Faça um programa que leia um vetor com trinta posições para números inteiros. Crie um segundo vetor, substituindo os valores nulos por 1. Mostre os dois vetores.

22. Faça um programa que leia um vetor A de dez posições. Em seguida, compacte o vetor, retirando os valores nulos e negativos. Armazene esse resultado no vetor B. Mostre o vetor B. (Lembre-se: o vetor B pode não ser completamente preenchido.)
23. Faça um programa que leia dois vetores (A e B) com cinco posições para 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. Ao final, mostre o resultado de todas as subtrações realizadas.
24. Faça um programa que leia um vetor com quinze posições para 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 com quinze posições para números inteiros. Depois da leitura, divida todos os seus elementos pelo maior valor do vetor. Mostre o vetor após os cálculos.

MATRIZ

7.1 MATRIZ EM ALGORITMOS

7.1.1 DEFINIÇÃO DE MATRIZ

Uma matriz é uma variável composta homogênea multidimensional. Ela é 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 precisa de um índice para cada uma de suas dimensões.

7.1.2 DECLARAÇÃO DE MATRIZ

Um algoritmo pode declarar uma matriz, conforme descrito a seguir.

```
DECLARE nome[dimensão1, dimensão2, dimensão3, ..., dimensãoN] tipo
```

onde: nome é o nome da variável do tipo matriz;

dimensão1 é a quantidade de elementos da 1^a dimensão (muitas vezes chamada de linha);

dimensão2 é a quantidade de elementos da 2^a dimensão (muitas vezes chamada de coluna);

dimensão3 é a quantidade de elementos da 3^a dimensão (muitas vezes chamada de profundidade);

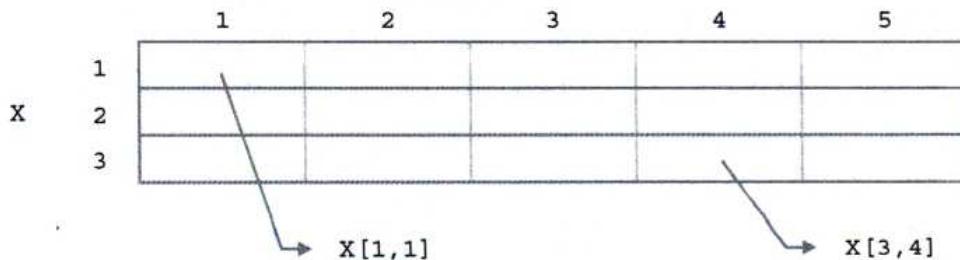
dimensãoN é a quantidade de elementos da n-ésima dimensão;

tipo é o tipo de dados dos elementos da matriz.

7.1.3 EXEMPLO DE MATRIZ

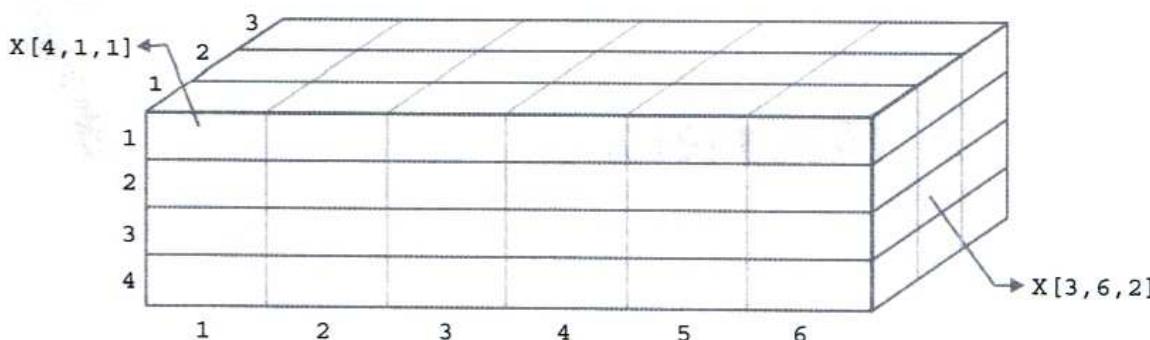
O exemplo a seguir define uma matriz bidimensional, onde o tamanho da 1^a dimensão (linha) é 3 e o da 2^a dimensão (coluna) é 5.

```
DECLARE X [3, 5] NUMÉRICO
```



O exemplo que se segue define uma matriz tridimensional, onde o tamanho da 1^a dimensão (linha) é 4, o tamanho da 2^a dimensão (coluna) é 6 e o tamanho da 3^a dimensão (profundidade) é 3.

DECLARE X[4, 6, 3] NUMÉRICO



Neste livro, abordaremos matrizes com até três dimensões, porém de fácil visualização.

7.1.4 ATRIBUINDO VALORES A UMA MATRIZ

Cada elemento de uma matriz pode armazenar um valor. Para fazer este armazenamento, é necessário executar uma atribuição, informando o número das dimensões.

```
X[2, 4] ← 45      X[3, 1] ← 13
X[4, 2, 1] ← 0    X[3, 5, 3] ← -4
```

7.1.5 PREENCHENDO UMA MATRIZ

Para preencher uma matriz, é necessário identificar todas as suas posições. Isto exige a utilização de um índice para cada dimensão da matriz.

No exemplo a seguir, uma matriz bidimensional com três linhas e cinco colunas é mostrada. Observe que a variável *i* varia dentro do intervalo de 1 a 3, ou seja, exatamente nas linhas. Para cada valor de *i*, a variável *j* varia de 1 a 5, ou seja, as cinco colunas que cada linha possui.

```
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

MEMÓRIA			TELA	
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	

Assim, podemos imaginar os elementos dispostos em uma estrutura bidimensional, como uma tabela.

		1	2	3	4	5
	1	12	9	3	7	-23
x	2	15	4	2	34	-4
	3	3	45	3	0	-3

Já no exemplo que se segue, uma matriz tridimensional com quatro linhas, três colunas e profundidade dois é preenchida. Observe que a variável *i* oscila dentro do intervalo de 1 a 4, ou seja, exatamente nas linhas. Para cada valor de *i*, a variável *j* se movimenta de 1 a 3, ou seja, as três colunas que cada linha possui, e, por fim, a variável *k* se alterna entre 1 e 2, que é a profundidade.

```

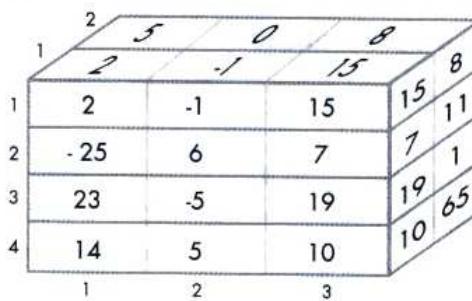
PARA i ← 1 ATÉ 4 FAÇA
INÍCIO
  PARA j ← 1 ATÉ 3 FAÇA
  INÍCIO
    PARA k ← 1 ATÉ 2 FAÇA
    INÍCIO
      ESCREVA "Digite o número da linha ",i, " coluna ", j,
      " e profundidade: ", k
      LEIA X[i,j,k]
    FIM
  FIM
FIM
  
```

Simulação

MEMÓRIA			TELA	
I	J	K		
1	1	1	Digite o número da linha 1 coluna 1 e profundidade 1: 2	
	2		Digite o número da linha 1 coluna 1 e profundidade 2: 5	
2	1	1	Digite o número da linha 1 coluna 2 e profundidade 1: -1	
	2		Digite o número da linha 1 coluna 2 e profundidade 2: 0	
3	1	1	Digite o número da linha 1 coluna 3 e profundidade 1: 15	
	2		Digite o número da linha 1 coluna 3 e profundidade 2: 8	
2	1	1	Digite o número da linha 2 coluna 1 e profundidade 1: -25	
	2		Digite o número da linha 2 coluna 1 e profundidade 2: 3	
2	1	1	Digite o número da linha 2 coluna 2 e profundidade 1: 6	
	2		Digite o número da linha 2 coluna 2 e profundidade 2: 9	

MEMÓRIA			TELA
	3	1	Digite o número da linha 2 coluna 3 e profundidade 1: 7
		2	Digite o número da linha 2 coluna 3 e profundidade 2: 11
3	1	1	Digite o número da linha 3 coluna 1 e profundidade 1: 23
		2	Digite o número da linha 3 coluna 1 e profundidade 2: -2
2	1	1	Digite o número da linha 3 coluna 2 e profundidade 1: -5
		2	Digite o número da linha 3 coluna 2 e profundidade 2: 46
3	1	1	Digite o número da linha 3 coluna 3 e profundidade 1: 19
		2	Digite o número da linha 3 coluna 3 e profundidade 2: 1
4	1	1	Digite o número da linha 4 coluna 1 e profundidade 1: 14
		2	Digite o número da linha 4 coluna 1 e profundidade 2: 27
2	1	1	Digite o número da linha 4 coluna 2 e profundidade 1: 5
		2	Digite o número da linha 4 coluna 2 e profundidade 2: 4
3	1	1	Digite o número da linha 4 coluna 3 e profundidade 1: 10
		2	Digite o número da linha 4 coluna 3 e profundidade 2: 65

Assim, podemos imaginar os elementos dispostos em uma estrutura tridimensional, como um cubo.



7.1.6 MOSTRANDO OS ELEMENTOS DE UMA MATRIZ

Para mostrar os elementos de uma matriz, é necessário identificar as suas posições. Isto exige a utilização de um índice para cada dimensão da matriz.

No exemplo a seguir, uma matriz bidimensional com três linhas e cinco colunas é mostrada. Observe que a variável *i* assume valores seqüenciais no intervalo de 1 a 3, ou seja, exatamente nas linhas da matriz. Para cada valor assumido por *i*, a variável *j* assume valores seqüenciais de 1 a 5, ou seja, as cinco colunas que cada linha possui.

```

PARA i ← 1 ATÉ 3 FAÇA
INÍCIO
  PARA j ← 1 ATÉ 5 FAÇA
  INÍCIO
    ESCREVA X[i,j]
  FIM
FIM
  
```

7.2 MATRIZ EM PASCAL

7.2.1 DEFINIÇÃO DE MATRIZ

As variáveis compostas homogêneas multidimensionais (matrizes) são conhecidas na linguagem PASCAL como **ARRAY**. Uma estrutura do tipo **ARRAY** é uma seqüência de variáveis com o mesmo identificador (mesmo nome) e alocadas seqüencialmente na memória. Todas as variáveis que compõem uma **ARRAY** devem ser do mesmo tipo.

Uma vez que as variáveis recebem o mesmo nome, o que as distingue são os índices que referenciam sua posição em cada dimensão da estrutura. Assim, se a matriz for bidimensional necessitará de dois índices, se for tridimensional necessitará de três índices, e assim por diante.

7.2.2 DECLARAÇÃO DE MATRIZ

```
VAR nome da variável: ARRAY[inicio1..fim1, inicio2..fim2, inicioN..fimN] OF
    tipo dos dados;
```

onde:

nome da variável é o nome da variável do tipo matriz;
 inicio1 é o índice inicial da primeira dimensão da matriz;
 fim1 é o índice final da primeira dimensão da matriz;
 inicio2 é o índice inicial da segunda dimensão da matriz;
 fim2 é o índice final da segunda dimensão da matriz;
 inicioN é o índice inicial da n-ésima dimensão da matriz;
 fimN é o índice final da n-ésima dimensão da matriz;
 tipo dos dados é o tipo básico de dados que serão armazenados na matriz.

7.2.3 EXEMPLO DE MATRIZ

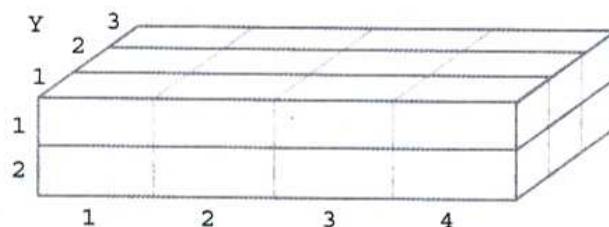
```
VAR X: ARRAY[1..2,1..6] OF REAL;
```

	1	2	3	4	5	6
X	1					
	2					

```
VAR MAT: ARRAY[2..5,3..6] OF CHAR;
```

	3	4	5	6
MAT	2			
	3			
	4			
	5			

```
VAR y: ARRAY[1..2,1..4,1..3] OF REAL;
```



7.2.4 ATRIBUINDO VALORES A UMA MATRIZ

Atribuir valor à matriz significa armazenar uma informação em um de seus elementos, identificado de forma única por meio de seus índices.

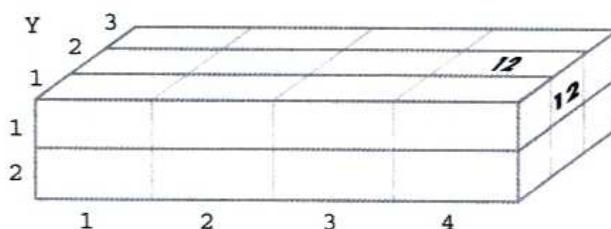
X[1,4] := 5 => Atribui o valor 5 à posição identificada pelos índices 1 (linha) e 4 (coluna).

	1	2	3	4	5	6
X	1				5	
	2					

MAT[4, 5] := 'D' => Atribui a letra D à posição identificada pelos índices 4 (linha) e 5 (coluna).

	3	4	5	6
MAT	2			
	3			
	4		D	
	5			

Y[1, 4, 2] := 12 => Atribui o valor 12 à posição identificada pelos índices 1 (linha), 4 (coluna) e 2 (profundidade).



7.2.5 PREENCHENDO UMA MATRIZ

Preencher uma matriz significa percorrer todos os seus elementos, atribuindo-lhes um valor. Este valor pode ser recebido do usuário, pelo teclado, ou pode ser gerado pelo programa.

No exemplo seguinte, percorrem-se os elementos de uma matriz bidimensional, atribuindo-lhes valores digitados pelo usuário e capturados por meio do comando READLN.

```
FOR i:= 1 TO 7 DO
BEGIN
  FOR j:=1 TO 3 DO
  BEGIN
    READLN(MAT[i,j]);
  END;
END;
```

7.2.6 MOSTRANDO OS ELEMENTOS DE UMA MATRIZ

Pode-se também percorrer todos os elementos da matriz, acessando o seu conteúdo. Para mostrar os valores armazenados em uma matriz, supondo que ela tenha sido declarada como var X:ARRAY [1..10, 1..6] OF REAL, pode-se executar os comandos a seguir.

```
FOR i:=1 TO 10 DO
BEGIN
  FOR j:= 1 TO 6 DO
  BEGIN
    WRITELN(X[i,j]);
  END;
END;
```

7.3 MATRIZ EM C/C++

7.3.1 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 até 12 dimensões. Entretanto, o limite de dimensões fica por conta da quantidade de recursos disponíveis ao compilador. Apesar disso, as matrizes mais utilizadas possuem duas dimensões. Para cada dimensão deve ser utilizado um índice.

7.3.2 DECLARAÇÃO DE MATRIZ

```
tipo_dos_dados nome_variável [dimensão1] [dimensão2] [...] [dimensãoN];
```

onde:

tipo_dos_dados é o tipo de dados que serão armazenados na matriz;

nome_variável é o nome dado à variável do tipo matriz;

[*dimensão1*] representa o tamanho da 1^a dimensão da matriz;

[*dimensão2*] representa o tamanho da 2^a dimensão da matriz;

[*dimensãoN*] representa o tamanho da n-ésima dimensão da matriz.

7.3.3 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					

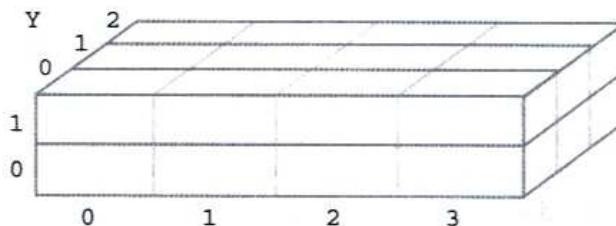
```
char MAT[4][3];
```

A declaração anterior criou uma variável chamada *MAT* contendo quatro linhas (0 a 3) com três colunas cada (0 a 2), capazes de armazenar símbolos, como pode ser observado a seguir.

	0	1	2
MAT	0		
	1		
	2		
	3		

```
float Y[2][4][3];
```

A declaração anterior criou uma variável chamada *Y* contendo duas linhas (0 a 1) com quatro colunas cada (0 a 3) e profundidade três (0 a 2), capazes de armazenar números reais, como pode ser observado a seguir.



7.3.4 ATRIBUINDO VALORES A UMA MATRIZ

Atribuir valor a uma matriz significa armazenar uma informação em um de seus elementos, identificado de forma única por meio de seus índices.

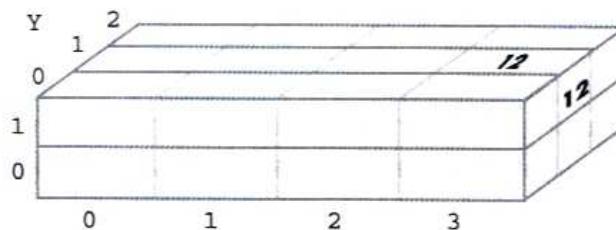
`X[1][4] = 5;` => Atribui o valor 5 à posição identificada pelos índices 1 (2^a linha) e 4 (5^a coluna).

	0	1	2	3	4	5
X	0					
	1				5	

`MAT[3][2] = 'D'` => Atribui a letra D à posição identificada pelos índices 3 (4^a linha) e 2 (3^a coluna).

	0	1	2
MAT	0		
	1		
	2		
	3		D

`Y[0][3][1] = 12` => Atribui o valor 12 à posição identificada pelos índices 0 (1^a linha), 3 (4^a coluna) e 1 (2^a profundidade).



7.3.5 PREENCHENDO UMA MATRIZ

Preencher uma matriz significa percorrer todos os seus elementos, atribuindo-lhes um valor. Este valor pode ser recebido do usuário, por meio do teclado, ou pode ser gerado pelo programa.

No exemplo que se segue, todos os elementos de uma matriz bidimensional são percorridos, atribuindo-lhes valores digitados pelo usuário e capturados pelo comando `cin`.

```
for (i=0;i<7;i++)
{
    for (j=0;j<3;j++)
        cin >> MAT[i][j];
}
```

Como a matriz possui sete linhas e três colunas, 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).

7.3.6 MOSTRANDO OS ELEMENTOS DE UMA MATRIZ

Pode-se também percorrer todos os elementos de uma matriz acessando o seu conteúdo. Para mostrar os valores armazenados dentro de uma matriz, supondo que ela 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 e seis colunas, 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).

7.4 MATRIZ EM JAVA

7.4.1 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 JAVA permite a declaração de matrizes unidimensionais (mais conhecidas como *vetores* – descritos no capítulo anterior), bidimensionais e multidimensionais. As matrizes mais utilizadas possuem duas dimensões. Para cada dimensão deve ser adotado um índice.

7.4.2 DECLARAÇÃO DE MATRIZ

Matrizes em JAVA são definidas pela existência de colchetes vazios antes ou depois do nome da variável no momento da declaração. Logo depois, deve ser feita a definição do tamanho de cada dimensão da matriz.

Para utilizar uma matriz em JAVA, é necessário seguir dois passos:

1º PASSO: DECLARAR UMA VARIÁVEL QUE FARÁ REFERÊNCIA AOS ELEMENTOS.

```
tipo_dos_dados nome_variável[][][]...[]; => Os colchetes vazios após o nome da variável
                                                 definem que a variável será uma estrutura
                                                 multidimensional.
```

2º PASSO: DEFINIR O TAMANHO DAS DIMENSÕES DA MATRIZ.

```
nome_variável = new tipo_dados [dimensão1] [dimensão2] [dimensão3]...[dimensãoN];
```

onde:

`tipo_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;
`[dimensão1]` representa o tamanho da primeira dimensão da matriz;
`[dimensão2]` representa o tamanho da segunda dimensão da matriz;
`[dimensãoN]` representa o tamanho da n-ésima dimensão da matriz.

7.4.3 EXEMPLO DE MATRIZ

Da mesma maneira como ocorre com os vetores, os índices de uma matriz começam sempre em 0 (zero) e podem variar até o tamanho da dimensão menos uma unidade.

É importante ressaltar que, em JAVA, os pares de colchetes podem aparecer todos antes do nome da variável ou todos depois do nome da variável ou, ainda, alguns antes e outros depois. Assim, todos os exemplos a seguir são válidos.

1º exemplo

```
float X[][];  
X = new float[2][6];
```

A declaração anterior criou uma variável chamada `X` contendo duas linhas (0 a 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					

2º exemplo

```
char [] [] MAT;  
MAT = new char[4][3];
```

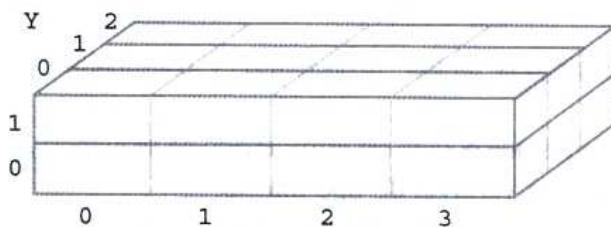
A declaração anterior criou uma variável chamada `MAT` contendo quatro linhas (0 a 3) com três colunas cada (0 a 2), capazes de armazenar símbolos, como pode ser observado a seguir.

	0	1	2
MAT	0		
	1		
	2		
	3		

3º exemplo

```
int [] [] Y[];  
Y = new int[2][4][3];
```

A declaração anterior criou uma variável chamada `Y` contendo duas linhas (0 a 1) com quatro colunas (0 a 3) e três profundidades (0 a 2), capazes de armazenar números inteiros, como pode ser observado a seguir.



4º exemplo

Além das formas descritas nos exemplos anteriores, a linguagem JAVA também permite que os dois passos necessários para utilização de uma matriz (declaração e dimensionamento) sejam realizados em apenas uma linha de comando.

As linhas que se seguem mostram três formas diferentes para criação de uma matriz chamada `mat`, contendo dez linhas e três colunas, onde poderão ser armazenados números inteiros.

A parte à esquerda do sinal de igualdade representa a declaração da variável mat, e a parte à direita, o dimensionamento da matriz.

```
int mat[][] = new int[10][3];
int [][]mat = new int[10][3];
int []mat[] = new int[10][3];
```

7.4.4 ATRIBUINDO VALORES A UMA MATRIZ

Atribuir valor a uma matriz significa armazenar uma informação em um de seus elementos, identificado de forma única por meio de seus índices.

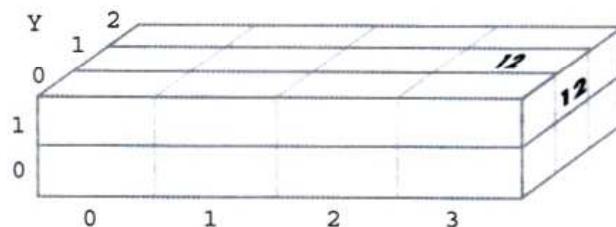
`x[1][4]=5` => Atribui o valor 5 à posição identificada pelos índices 1 (2^a linha) e 4 (5^a coluna).

	0	1	2	3	4	5
x						
	0					5

`MAT[3][2] = 'D'` => Atribui a letra D à posição identificada pelos índices 3 (4^a linha) e 2 (3^a coluna).

	0	1	2
MAT			
	0		
	1		
	2		
	3		D

`Y[0][3][1] = 12` => Atribui o valor 12 à posição identificada pelos índices 0 (1^a linha), 3 (4^a coluna) e 1 (2^a profundidade).



7.4.5 PREENCHENDO UMA MATRIZ

Preencher uma matriz significa percorrer todos os seus elementos, atribuindo-lhes um valor. Este valor pode ser recebido do usuário, por meio do teclado, ou pode ser gerado pelo programa.

No exemplo a seguir, todos os elementos de uma matriz bidimensional são percorridos, atribuindo-lhes valores inteiros digitados pelo usuário e capturados pelo método `nextInt()` da classe `Scanner`.

```
Scanner e = new Scanner(System.in);
for (i=0;i<7;i++)
{
    for (j=0;j<3;j++)
        X[i][j] = e.nextInt();
}
```

Como a matriz possui sete linhas e três colunas, 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).

7.4.6 MOSTRANDO OS ELEMENTOS DE UMA MATRIZ

Pode-se, também, percorrer todos os elementos da matriz, acessando o seu conteúdo. No exemplo que se segue, são mostrados todos os elementos de uma matriz contendo dez linhas e seis colunas. Observe que são usados dois índices, i e j . Estes índices estão atrelados a estruturas de repetição que mantêm a variação de ambos dentro de intervalos permitidos. Ou seja, o índice i , que representa as linhas, varia de 0 a 9 e o índice j , que representa as colunas, varia de 0 a 5.

```
for (i=0; i<10; i++)
{
    for (j=0; j<6; j++)
        System.out.println(X[i][j]);
}
```

EXERCÍCIOS RESOLVIDOS

-  1. Faça um programa que preencha uma matriz M (2×2), calcule e mostre a matriz R , resultante da multiplicação dos elementos de M pelo seu maior elemento.

ALGORITMO

SOLUÇÃO:

```
ALGORITMO
DECLARE mat[2,2], resultado[2,2], 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
        FIM_ALGORITMO.
```

SOLUÇÃO:

\EXERC\CAP7\PASCAL\EX1.PAS e \EXERC\CAP7\PASCAL\EX1.EXE

SOLUÇÃO:

\EXERC\CAP7\C++\EX1.CPP e \EXERC\CAP7\C++\EX1.EXE

SOLUÇÃO:

\EXERC\CAP7\JAVA\EX1.java e \EXERC\CAP7\JAVA\EX1.class

2. Faça um programa que preencha uma matriz 10×3 com as notas de dez alunos em três provas. O programa deverá mostrar um relatório com o número dos alunos (número da linha) e a prova em que cada aluno obteve menor nota. Ao final do relatório, deverá mostrar quantos alunos tiveram menor nota em cada uma das provas: na prova 1, na prova 2 e na prova 3.

ALGORITMOSOLUÇÃO:

```

ALGORITMO
DECLARE notas[10,3],q1, q2, q3, menor, p_menor, i, j NUMÉRICO
PARA i ← 1 ATÉ 10 FAÇA
    INÍCIO
        PARA j ← 1 ATÉ 3 FAÇA
            INÍCIO
                LEIA notas[i,j]
            FIM
        FIM
    q1 ← 0
    q2 ← 0
    q3 ← 0
PARA i ← 1 ATÉ 10 FAÇA
    INÍCIO
        ESCREVA i
        menor ← notas[i,1]
        p_menor ← 1
        PARA j ← 1 ATÉ 3 FAÇA
            INÍCIO
                SE notas[i, j] < menor
                    ENTÃO INÍCIO
                        menor ← notas[i, j]
                        p_menor ← j
                    FIM
                FIM
            ESCREVA p_menor
            SE p_menor = 1
                ENTÃO q1 ← q1 + 1
            SE p_menor = 2
                ENTÃO q2 ← q2 + 1
            SE p_menor = 3
                ENTÃO q3 ← q3 + 1
            FIM
        ESCREVA q1, q2, q3
    FIM_ALGORITMO.

```

SOLUÇÃO:

\EXERC\CAP7\PASCAL\EX2.PAS e \EXERC\CAP7\PASCAL\EX2.EXE

**SOLUÇÃO:**

\EXERC\CAP7\C++\EX2.CPP e \EXERC\CAP7\C++\EX2.EXE

**SOLUÇÃO:**

\EXERC\CAP7\JAVA\EX2.java e \EXERC\CAP7\JAVA\EX2.class

3. Faça um programa que preencha:

- ◆ um vetor com oito posições, contendo nomes de lojas;
- ◆ outro vetor com quatro posições, contendo nomes de produtos;
- ◆ uma matriz com os preços de todos os produtos em cada loja.

O programa deverá mostrar todas as relações (nome do produto – nome da loja) em que o preço não ultrapasse R\$ 120,00.

ALGORITMO**SOLUÇÃO:**

```

ALGORITMO
    DECLARE lojas[8],produtos[4] LITERAL
        preços[4,8], 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 preços[i, j]
                FIM
            FIM
        PARA i ← 1 ATÉ 4 FAÇA
            INÍCIO
                PARA j ← 1 ATÉ 8 FAÇA
                    INÍCIO
                        SE preços[i, j] < 120
                            ENTÃO ESCREVA produtos[i], lojas[j]
                    FIM
                FIM
            FIM
        FIM_ALGORITMO.
    
```

**SOLUÇÃO:**

\EXERC\CAP7\PASCAL\EX3.PAS e \EXERC\CAP7\PASCAL\EX3.EXE

**SOLUÇÃO:**

\EXERC\CAP7\C++\EX3.CPP e \EXERC\CAP7\C++\EX3.EXE

**SOLUÇÃO:**

\EXERC\CAP7\JAVA\EX3.java e \EXERC\CAP7\JAVA\EX3.class

4. Crie um programa que preencha uma matriz 10×20 com números inteiros e some cada uma das linhas, armazenando o resultado das somas em um vetor. A seguir, o programa deverá multiplicar cada elemento da matriz pela soma da linha correspondente e mostrar a matriz resultante.

ALGORITMO**SOLUÇÃO:**

```

ALGORITMO
DECLARE mat[10,20], soma[10], 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
        soma[i] ← 0
    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
    FIM
FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP7\PASCAL\EX4.PAS e \EXERC\CAP7\PASCAL\EX4.EXE

**SOLUÇÃO:**

\EXERC\CAP7\C++\EX4.CPP e \EXERC\CAP7\C++\EX4.EXE

**SOLUÇÃO:**

\EXERC\CAP7\JAVA\EX4.java e \EXERC\CAP7\JAVA\EX4.class

5. Na teoria dos sistemas, define-se o elemento MINMAX de uma matriz como o maior elemento da linha em que se encontra o menor elemento da matriz. Elabore um programa que carregue uma matriz 4×7 com números reais, calcule e mostre seu MINMAX e sua posição (linha e coluna).

ALGORITMO**SOLUÇÃO:**

```

ALGORITMO
DECLARE mat[4,7], menor, maior, i, j, l_menor, col 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]
l_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]
                    l_menor ← i
                FIM
            FIM
        FIM
    FIM
maior ← mat[l_menor, 1]
col ← 1
PARA j ← 1 ATÉ 7 FAÇA
    INÍCIO
        SE mat[l_menor, j] > maior
        ENTÃO INÍCIO
            maior ← mat[l_menor, j]
            col ← j
        FIM
    FIM
ESCREVA maior, l_menor, col
FIM_ALGORITMO.

```

SOLUÇÃO:

\EXERC\CAP7\PASCAL\EX5.PAS e \EXERC\CAP7\PASCAL\EX5.EXE

SOLUÇÃO:

\EXERC\CAP7\C++\EX5.CPP e \EXERC\CAP7\C++\EX5.EXE

SOLUÇÃO:

\EXERC\CAP7\JAVA\EX5.java e \EXERC\CAP7\JAVA\EX5.class

6. Crie um programa que preencha uma primeira matriz de ordem 4×5 e uma segunda matriz 5×2 . O programa deverá, também, calcular e mostrar a matriz resultante do produto matricial das duas matrizes anteriores, armazenando-o em uma terceira matriz de ordem 4×2 .

ALGORITMOSOLUÇÃO:

```

ALGORITMO
DECLARE a[4, 5], b[5, 2], 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
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
PARA i ← 1 ATÉ 4 FAÇA
INÍCIO
    PARA j ← 1 ATÉ 2 FAÇA
    INÍCIO
        ESCREVA c[i, j]
    FIM
FIM
FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP7\PASCAL\EX6.PAS e \EXERC\CAP7\PASCAL\EX6.EXE

**SOLUÇÃO:**

\EXERC\CAP7\C++\EX6.CPP e \EXERC\CAP7\C++\EX6.EXE

**SOLUÇÃO:**

\EXERC\CAP7\JAVA\EX6.java e \EXERC\CAP7\JAVA\EX6.class

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 5×7 , verifique se a matriz possui ponto de sela e, se possuir, mostre seu valor e sua localização.

ALGORITMO**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
    FIM
cont ← 0

```

```

PARA i ← 1 ATÉ 5 FAÇA
    INÍCIO
        menor ← mat[i, 1]
        linha ← i
        coluna ← 0
        PARA j ← 1 ATÉ 7 FAÇA
            INÍCIO
                SE mat[i, j] < menor
                ENTÃO INÍCIO
                    menor ← mat[i, j]
                    linha ← i
                    coluna ← j
                FIM
            FIM
        maior ← mat[1, coluna]
        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\CAP7\PASCAL\EX7.PAS e \EXERC\CAP7\PASCAL\EX7.EXE

SOLUÇÃO:

\EXERC\CAP7\C++\EX7.CPP e \EXERC\CAP7\C++\EX7.EXE

SOLUÇÃO:

\EXERC\CAP7\JAVA\EX7.java e \EXERC\CAP7\JAVA\EX7.class

8. Elabore um programa que preencha uma matriz 6×4 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.

ALGORITMOSOLUÇÃO:

```

ALGORITMO
    DECLARE mat1[6,4], mat2[6,4], 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
            ESCRIVA qtde
        PARA i ← 1 ATÉ 6 FAÇA
            INÍCIO
                PARA j ← 1 ATÉ 4 FAÇA
                    INÍCIO
                        ESCRIVA mat2[i, j]
                    FIM
                FIM
            FIM
        FIM_ALGORITMO.
    
```

**SOLUÇÃO:**

\EXERC\CAP7\PASCAL\EX8.PAS e \EXERC\CAP7\PASCAL\EX8.EXE

**SOLUÇÃO:**

\EXERC\CAP7\C++\EX8.CPP e \EXERC\CAP7\C++\EX8.EXE

**SOLUÇÃO:**

\EXERC\CAP7\JAVA\EX8.java e \EXERC\CAP7\JAVA\EX8.class

- 9.** Crie um programa que preencha uma matriz 15×5 com números inteiros, calcule e mostre quais elementos da matriz se repetem e quantas vezes cada um se repete.

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 ← 5
                                    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 > 5
                                ENTÃO INÍCIO
                                    lin ← lin + 1
                                    col ← 1
                                FIM
                            FIM
                        FIM
                    FIM
                FIM
            FIM
        FIM
    FIM
PARA i ← 1 ATÉ lin FAÇA
    INÍCIO
        SE i < lin
        ENTÃO x ← 5
        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_ALGORITMO.

```



SOLUÇÃO:

\EXERC\CAP7\PASCAL\EX9.PAS e \EXERC\CAP7\PASCAL\EX9.EXE

**SOLUÇÃO:**

\EXERC\CAP7\C++\EX9.CPP e \EXERC\CAP7\C++\EX9.EXE

**SOLUÇÃO:**

\EXERC\CAP7\JAVA\EX9.java e \EXERC\CAP7\JAVA\EX9.class

10. Elabore um programa que preencha uma matriz 10×10 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.

ALGORITMO**SOLUÇÃO:**

```

ALGORITMO
    DECLARE mat[10,10], aux, i, j NUMÉRICO
    PARA i ← 1 ATÉ 10 FAÇA
        INÍCIO
            PARA j ← 1 ATÉ 10 FAÇA
                INÍCIO
                    LEIA mat[i, j]
                FIM
            FIM
        PARA i ← 1 ATÉ 10 FAÇA
            INÍCIO
                aux ← mat[2, i]
                mat[2, i] ← mat[8, i]
                mat[8, i] ← aux
            FIM
        PARA i ← 1 ATÉ 10 FAÇA
            INÍCIO
                aux ← mat[i, 4]
                mat[i, 4] ← mat[i, 10]
                mat[i, 10] ← aux
            FIM
        j ← 10
        PARA i ← 1 ATÉ 10 FAÇA
            INÍCIO
                aux ← mat[i, i]
                mat[i, i] ← mat[i, j]
                mat[i, j] ← aux
                j ← j - 1
            FIM
        PARA j ← 1 ATÉ 10 FAÇA
            INÍCIO
                aux ← mat[5, j]
                mat[5, j] ← mat[j, 10]
                mat[j, 10] ← aux
            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\CAP7\PASCAL\EX10.PAS e \EXERC\CAP7\PASCAL\EX10.EXE

SOLUÇÃO:

\EXERC\CAP7\C++\EX10.CPP e \EXERC\CAP7\C++\EX10.EXE

SOLUÇÃO:

\EXERC\CAP7\JAVA\EX10.java e \EXERC\CAP7\JAVA\EX10.class

11. Crie um programa que preencha uma matriz 8×8 com números inteiros e mostre uma mensagem dizendo se a matriz digitada é simétrica. Uma matriz só pode ser considerada simétrica se $A[i, j] = A[j, i]$.

ALGORITMOSOLUÇÃO:

```

ALGORITMO
  DECLARE mat[8,8], i, j NUMÉRICO
            simetria LÓGICO
  PARA i ← 1 ATÉ 8 FAÇA
    INÍCIO
    PARA j ← 1 ATÉ 8 FAÇA
      INÍCIO
      LEIA mat[i, j]
      FIM
    FIM
  simetria ← verdadeiro
  PARA i ← 1 ATÉ 8 FAÇA
    INÍCIO
    PARA j ← 1 ATÉ 8 FAÇA
      INÍCIO
      SE mat[i, j] ≠ mat[j, i]
        ENTÃO simetria ← falso
      FIM
    FIM
  SE simetria = verdadeiro
    ENTÃO ESCREVA "Matriz Simétrica"
    SENÃO ESCREVA "Matriz Assimétrica"
  FIM_ALGORITMO.
```

SOLUÇÃO:

\EXERC\CAP7\PASCAL\EX11.PAS e \EXERC\CAP7\PASCAL\EX11.EXE

SOLUÇÃO:

\EXERC\CAP7\C++\EX11.CPP e \EXERC\CAP7\C++\EX11.EXE

SOLUÇÃO:

\EXERC\CAP7\JAVA\EX11.java e \EXERC\CAP7\JAVA\EX11.class

12. Elabore um programa que preencha uma matriz 4×4 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 desta linha, é igual à soma dos elementos da diagonal principal e, também, é igual à soma dos elementos da diagonal secundária.

ALGORITMO**SOLUÇÃO:**

```

ALGORITMO
    DECLARE mat[4,4] NUMÉRICO
        soma_linha[4] NUMÉRICO
        soma_coluna[4] NUMÉRICO
        soma_diags, soma_diags, i, j, compara NUMÉRICO
        q_magico LÓGICO
    PARA i ← 1 ATÉ 4 FAÇA
        INÍCIO
        PARA j ← 1 ATÉ 4 FAÇA
            INÍCIO
                LEIA mat[i,j]
            FIM
        FIM
    PARA i ← 1 ATÉ 4 FAÇA
        INÍCIO
        soma_linha[i] ← 0
    PARA j ← 1 ATÉ 4 FAÇA
        INÍCIO
            soma_linha[i] ← soma_linha[i] + mat[i,j]
        FIM
    FIM
    PARA i ← 1 ATÉ 4 FAÇA
        INÍCIO
        soma_coluna[i] ← 0
    PARA j ← 1 ATÉ 4 FAÇA
        INÍCIO
            soma_coluna[i] ← soma_coluna[i] + mat[j,i]
        FIM
    FIM
    soma_diags ← 0
    PARA i ← 1 ATÉ 4 FAÇA
        INÍCIO
            soma_diags ← soma_diags + mat[i,i]
        FIM
    soma_diags ← 0
    j ← 4
    PARA i ← 1 ATÉ 4 FAÇA
        INÍCIO
            soma_diags ← soma_diags + mat[i,j]
            j ← j - 1
        FIM
    q_magico ← verdadeiro
    PARA i ← 1 ATÉ 4 FAÇA
        INÍCIO
            PARA j ← 1 ATÉ 4 FAÇA
                INÍCIO
                    SE soma_linha[i] ≠ soma_coluna[j]
                        ENTÃO q_magico ← falso
                FIM
            FIM
    PARA i ← 1 ATÉ 4 FAÇA
        INÍCIO
            SE soma_linha[i] ≠ soma_diags
                ENTÃO q_magico ← falso
            FIM
    PARA i ← 1 ATÉ 4 FAÇA
        INÍCIO
            SE soma_linha[i] ≠ soma_diags
                ENTÃO q_magico ← falso
            FIM
    SE q_magico = verdadeiro

```

ENTÃO ESCREVA "Forma quadrado mágico"
 SENÃO ESCREVA "Não forma quadrado mágico"
 FIM_ALGORITMO.

SOLUÇÃO:

\EXERC\CAP7\PASCAL\EX12.PAS e \EXERC\CAP7\PASCAL\EX12.EXE

SOLUÇÃO:

\EXERC\CAP7\C++\EX12.CPP e \EXERC\CAP7\C++\EX12.EXE

SOLUÇÃO:

\EXERC\CAP7\JAVA\EX12.java e \EXERC\CAP7\JAVA\EX12.class

13. Faça um programa que preencha:

- ♦ um vetor com os nomes de cinco produtos;
- ♦ uma matriz 5×4 com os preços dos cinco produtos em quatro lojas diferentes;
- ♦ outro vetor com o custo do transporte dos cinco produtos.

O programa deverá preencher uma segunda matriz 5×4 com os valores dos impostos de cada produto, de acordo com a 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

O programa deverá mostrar ainda um relatório com o nome do produto, o número da loja onde o produto é encontrado, 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).

ALGORITMOSOLUÇÃO:

```

ALGORITMO
DECLARE nome[5] LITERAL
      preco, imp[5,4], custo[5], 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
FIM_ALGORITMO.

```

SOLUÇÃO:

\EXERC\CAP7\PASCAL\EX13.PAS e \EXERC\CAP7\PASCAL\EX13.EXE

SOLUÇÃO:

\EXERC\CAP7\C++\EX13.CPP e \EXERC\CAP7\C++\EX13.EXE

SOLUÇÃO:

\EXERC\CAP7\JAVA\EX13.java e \EXERC\CAP7\JAVA\EX13.class

14. Faça um programa que receba:

- ◆ um vetor com o nome de cinco cidades diferentes;
- ◆ uma matriz 5×5 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;
- ◆ o consumo de combustível de um veículo, ou seja, quantos quilômetros este veículo percorre com um litro de combustível.

O programa deverá calcular e mostrar:

- ◆ os percursos que não ultrapassam 250 quilômetros (os percursos são compostos pelos nomes das cidades de origem e pelos nomes das cidades de destino);
- ◆ todos os percursos (nome da cidade de origem e nome da cidade de destino), juntamente com a quantidade de combustível necessária para o veículo percorrê-los;

ALGORITMOSOLUÇÃO:

```

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 LEIA distancia[i, j]
            FIM
        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
    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
    FIM
FIM_ALGORITMO.

```

SOLUÇÃO:

\EXERC\CAP7\PASCAL\EX14.PAS e \EXERC\CAP7\PASCAL\EX14.EXE

SOLUÇÃO:

\EXERC\CAP7\C++\EX14.CPP e \EXERC\CAP7\C++\EX14.EXE

SOLUÇÃO:

\EXERC\CAP7\JAVA\EX14.java e \EXERC\CAP7\JAVA\EX14.class

15. Elabore um programa que preencha:

- ◆ um vetor com cinco números inteiros;
- ◆ outro vetor com dez números inteiros;
- ◆ uma matriz 4×3 , também com números inteiros.

O programa deverá calcular e mostrar:

- ◆ 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.

ALGORITMO**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
    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
    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
        FIM
        ESCREVA "Soma dos elementos pares da linha ", i,
            ↪ " da matriz resultante = ", soma
    FIM
PARA j ← 1 ATÉ 3 FAÇA

```

```

INÍCIO
    qtde ← 0
    PARA i ← 1 ATÉ 4 FAÇA
        INÍCIO
            SE mat_result[i,j] > 1 E mat_result[i,j] < 5
                ENTÃO qtde ← qtde + 1
            FIM
        ESCREVA "Total de números entre 1 e 5 na coluna ",
            → j, " da matriz resultante = ",qtde
        FIM
    FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP7\PASCAL\EX15.PAS e \EXERC\CAP7\PASCAL\EX15.EXE

**SOLUÇÃO:**

\EXERC\CAP7\C++\EX15.CPP e \EXERC\CAP7\C++\EX15.EXE

**SOLUÇÃO:**

\EXERC\CAP7\JAVA\EX15.java e \EXERC\CAP7\JAVA\EX15.class

16. Faça um programa que preencha uma matriz 7×7 de números inteiros e crie dois vetores com sete posições cada um que contenham, respectivamente, o maior elemento de cada uma das linhas e o menor elemento de cada uma das colunas. Escreva a matriz e os dois vetores gerados.

ALGORITMO**SOLUÇÃO:**

```

ALGORITMO
DECLARE mat[7,7], vet1[7], vet2[7] NUMÉRICO
    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
    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\CAP7\PASCAL\EX16.PAS e \EXERC\CAP7\PASCAL\EX16.EXE

SOLUÇÃO:

\EXERC\CAP7\C++\EX16.CPP e \EXERC\CAP7\C++\EX16.EXE

SOLUÇÃO:

\EXERC\CAP7\JAVA\EX16.java e \EXERC\CAP7\JAVA\EX16.class

17. Faça um programa que utilize uma matriz 5×5 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, o programa deverá 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).

Segue-se 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 deverá observar as seguintes situações:

- ◆ quando o usuário digitar um múltiplo de 5 e não houver mais espaço na diagonal principal, deverá mostrar a mensagem *Diagonal totalmente preenchida*;

- ◆ quando o usuário digitar um múltiplo de 11 e não houver mais espaço disponível na matriz, deverá mostrar 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, deverá mostrar a mensagem *Não existe espaço abaixo da diagonal principal*;
- ◆ quando a matriz estiver totalmente preenchida, deverá mostrar 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
                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
            FIM
        FIM
    FIM
FIM

```

```

        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\CAP7\PASCAL\EX17.PAS e \EXERC\CAP7\PASCAL\EX17.EXE

SOLUÇÃO:

\EXERC\CAP7\C++\EX17.CPP e \EXERC\CAP7\C++\EX17.EXE

SOLUÇÃO:

\EXERC\CAP7\JAVA\EX17.java e \EXERC\CAP7\JAVA\EX17.class

- 18.** Crie um programa que leia um vetor V contendo 18 elementos. A seguir, o programa deverá distribuir esses elementos em uma matriz 3×6 e, no final, mostrar a matriz gerada.

Veja a seguir um exemplo do que o seu programa deverá 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

ALGORITMOSOLUÇÃO:

```

ALGORITMO
DECLARE vet [18], mat [3,6], 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]
    FIM

```

```

        col ← col + 1
        SE col > 6
            ENTÃO INÍCIO
                lin ← lin + 1
                col ← 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\CAP7\PASCAL\EX18.PAS e \EXERC\CAP7\PASCAL\EX18.EXE

SOLUÇÃO:

\EXERC\CAP7\C++\EX18.CPP e \EXERC\CAP7\C++\EX18.EXE

SOLUÇÃO:

\EXERC\CAP7\JAVA\EX18.java e \EXERC\CAP7\JAVA\EX18.class

-  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 esta linha estiver 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, o programa deverá mostrar uma mensagem ao usuário;
- ◆ quando a matriz estiver totalmente preenchida, o programa deverá encerrar a leitura dos números e mostrar todos os elementos armazenados na matriz.

SOLUÇÃO:

```

ALGORITMO
DECLARE mat[5,4], i, j, num, r NUMÉRICO
        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
            col_p ← col_p + 1
            SE col_p > 4
                ENTÃO INÍCIO
                    lin_p ← lin_p + 2
                    col_p ← 1
                FIM
            FIM
        FIM
    SENÃO INÍCIO
        SE lin_i > 5
            ENTÃO ESCREVA "Sem espaço para números ímpares"
            SENÃO INÍCIO
                mat[lin_i,col_i] ← num
                col_i ← col_i + 1
                SE col_i > 4
                    ENTÃO INÍCIO
                        lin_i ← lin_i + 2
                        col_i ← 1
                    FIM
                FIM
            FIM
        FIM
    tot ← tot + 1
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\CAP7\PASCAL\EX19.PAS e \EXERC\CAP7\PASCAL\EX19.EXE

**SOLUÇÃO:**

\EXERC\CAP7\C++\EX19.CPP e \EXERC\CAP7\C++\EX19.EXE

**SOLUÇÃO:**

\EXERC\CAP7\JAVA\EX19.java e \EXERC\CAP7\JAVA\EX19.class

- 20.** Crie um programa que utilize uma matriz com dimensões máximas de cinco linhas e quatro colunas e solicite que sejam digitados os números (desordenadamente), armazenando-os, ordenadamente, na matriz.

Observe o exemplo que se segue.

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

ALGORITMO**SOLUÇÃO:**

```

ALGORITMO
DECLARE num[5,4] NUMÉRICO
    num_aux, i, j, k, l, m, n, lin, col NUMÉRICO
PARA i ← 1 ATÉ 5
INÍCIO
PARA j ← 1 ATÉ 4
INÍCIO
LEIA num_aux
IF i = 1 E j = 1
    ENTÃO num[i, j] ← num_aux
    SENÃO INICIO
        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 INICIO
                    k ← k + 1
                    l ← 1
                    FIM
                FIM
            m ← i;
            n ← j;
            ENQUANTO m ≠ k OU n ≠ l FAÇA
                INÍCIO
                SE n-1 < 1
                    ENTÃO INICIO
                    lin ← m-1
                    col ← 4
                    FIM
                SENÃO INICIO
                    lin ← m
                    col ← n-1
                    FIM
                num[m][n] ← num[lin][col]
                n ← n-1
                SE n < 1
                    ENTÃO INICIO
                        n ← 4
                        m ← m-1
                        FIM
                    FIM
                num[k][l] ← num_aux
                FIM
            FIM
        PARA i ← 1 ATÉ 5
        INÍCIO
        PARA j ← 1 ATÉ 4

```

```

INÍCIO
ESCREVA "Elemento da posição ", i, "-", j, " = ", num[i][j]
FIM
FIM
FIM_ALGORITMO.

```

SOLUÇÃO:

\EXERC\CAP7\PASCAL\EX20.PAS e \EXERC\CAP7\PASCAL\EX20.EXE

SOLUÇÃO:

\EXERC\CAP7\C++\EX20.CPP e \EXERC\CAP7\C++\EX20.EXE

SOLUÇÃO:

\EXERC\CAP7\JAVA\EX20.java e \EXERC\CAP7\JAVA\EX20.class

21. Crie 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 deverá ser obrigatoriamente quadrada (número igual de linhas e colunas). Sendo assim, solicite que seja informada a dimensão da matriz.

Posteriormente, o programa deverá realizar a leitura dos elementos que vão compor a matriz.

Finalmente, deverá somar e mostrar os elementos que estão abaixo da diagonal secundária.

Veja o exemplo.

Imagine que sejam informados números, conforme apresentado nesta matriz.

20	10	1	8
17	42	11	98
19	45	32	87
12	36	65	25

O resultado do problema seria: $98 + 32 + 87 + 36 + 65 + 25 = 343$

SOLUÇÃO:

```

ALGORITMO
DECLARE num[100,100],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
    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\CAP7\PASCAL\EX21.PAS e \EXERC\CAP7\PASCAL\EX21.EXE

SOLUÇÃO:

\EXERC\CAP7\C++\EX21.CPP e \EXERC\CAP7\C++\EX21.EXE

SOLUÇÃO:

\EXERC\CAP7\JAVA\EX21.java e \EXERC\CAP7\JAVA\EX21.class

22. Faça um programa que receba o estoque atual de três produtos, armazenados em quatro armazéns, e coloque esses dados em uma matriz 5×3 . Considerando que a última linha dessa matriz contém o custo de cada produto, o programa deverá calcular e mostrar:

- ◆ a quantidade de itens armazenados em cada armazém;
- ◆ qual armazém possui maior estoque do produto 2;
- ◆ qual armazém possui menor estoque;
- ◆ qual o custo total de cada produto;
- ◆ qual o custo total de cada armazém.

Devem ser desconsiderados empates.

ALGORITMOSOLUÇÃO:

```

ALGORITMO
DECLARE prod[5,3], tot_arm, maior_e, menor_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
    SE i=1
        ENTÃO INÍCIO
            menor_e ← tot_arm
            ind_a ← i
        FIM
    SENÃO INÍCIO
        SE tot_arm < menor_e
            ENTÃO INÍCIO
                menor_e ← tot_arm
                ind_a ← i
            FIM
        FIM
    FIM
ESCREVA "Armazém com menor estoque", ind_a
PARA i ← 1 ATÉ 4 FAÇA
    INÍCIO
    SE i = 1
        ENTÃO INÍCIO
            maior_e ← prod[i,2]
            ind_a ← i
        FIM
    SENÃO INÍCIO
        SE prod[i,2] > maior_e
            ENTÃO INÍCIO
                maior_e ← prod[i,2]
                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\CAP7\PASCAL\EX22.PAS e \EXERC\CAP7\PASCAL\EX22.EXE

**SOLUÇÃO:**

\EXERC\CAP7\C++\EX22.CPP e \EXERC\CAP7\C++\EX22.EXE

**SOLUÇÃO:**

\EXERC\CAP7\JAVA\EX22.java e \EXERC\CAP7\JAVA\EX22.class

23. Crie um programa que receba as vendas semanais (de um mês) de cinco vendedores de uma loja e armazene essas vendas em uma matriz. O programa deverá calcular e mostrar:

- ◆ 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.

ALGORITMO**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
            FIM
        ESCREVA tot_geral
    FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP7\PASCAL\EX23.PAS e \EXERC\CAP7\PASCAL\EX23.EXE

SOLUÇÃO:

\EXERC\CAP7\C++\EX23.CPP e \EXERC\CAP7\C++\EX23.EXE

SOLUÇÃO:

\EXERC\CAP7\JAVA\EX23.java e \EXERC\CAP7\JAVA\EX23.class

24. Uma escola deseja controlar as médias das disciplinas cursadas por seus alunos. Sabe-se que nessa escola existem três turmas, com oito alunos cada, e cada aluno cursa quatro disciplinas. Crie um programa que armazene essas médias em uma matriz $3 \times 8 \times 4$. Depois da leitura, ele deverá calcular e mostrar:

- ◆ a média geral de cada aluno;
- ◆ a média de cada turma.

ALGORITMOSOLUÇÃO:

```

ALGORITMO
DECLARE medias[3][8][4], i, j, k, media_aluno, media_turma NUMÉRICO
PARA i ← 1 ATÉ 3 FAÇA
    INÍCIO
        PARA j ← 1 ATÉ 8 FAÇA
            INÍCIO
                PARA k ← 1 ATÉ 4 FAÇA
                    INÍCIO
                        LEIA medias[i][j][k]
                    FIM
                FIM
            FIM
        FIM
    PARA i ← 1 ATÉ 3 FAÇA
        INÍCIO
            PARA j ← 1 ATÉ 8 FAÇA
                INÍCIO
                    media_aluno ← 0
                    PARA k ← 1 ATÉ 4 FAÇA
                        INÍCIO
                            media_aluno ← media_aluno + medias[i][j][k]
                        FIM
                    media_aluno ← media_aluno / 4
                    ESCREVA "A média do aluno ", j, " na turma ",
                        → i, " = ", media_aluno
                FIM
            FIM
        FIM

PARA i ← 1 ATÉ 3 FAÇA
    INÍCIO
        media_turma ← 0
        PARA j ← 1 ATÉ 8 FAÇA
            INÍCIO
                PARA k ← 1 ATÉ 4 FAÇA
                    INÍCIO
                        media_turma ← media_turma + medias[i][j][k]
                    FIM
                FIM
            FIM
        media_turma ← media_turma / (8 * 4)
        ESCREVA "A média da turma ", i, " = ", media_turma
    FIM
FIM_ALGORITMO.

```

SOLUÇÃO:

\EXERC\CAP7\PASCAL\EX24.PAS e \EXERC\CAP7\PASCAL\EX24.EXE

SOLUÇÃO:

\EXERC\CAP7\C++\EX24.CPP e \EXERC\CAP7\C++\EX24.EXE

SOLUÇÃO:

\EXERC\CAP7\JAVA\EX24.java e \EXERC\CAP7\JAVA\EX24.class

25. Elabore um programa que receba as vendas de cinco produtos em três lojas diferentes e em dois meses consecutivos. O programa deverá armazenar essas vendas em duas matrizes 5×3 . O bimestre é uma matriz 5×3 , resultado da soma das duas matrizes anteriores. Deverá ainda calcular e mostrar:

- ◆ 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.

ALGORITMOSOLUÇÃO:

```

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
        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
                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
                    FIM
                FIM
            FIM
        FIM
    FIM
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
    ESCREVA tot_prod
FIM
FIM_ALGORITMO.

```

**SOLUÇÃO:**

\EXERC\CAP7\PASCAL\EX25.PAS e \EXERC\CAP7\PASCAL\EX25.EXE

**SOLUÇÃO:**

\EXERC\CAP7\C++\EX25.CPP e \EXERC\CAP7\C++\EX25.EXE

**SOLUÇÃO:**

\EXERC\CAP7\JAVA\EX25.java e \EXERC\CAP7\JAVA\EX25.class

EXERCÍCIOS PROPOSTOS

1. Faça um programa que preencha uma matriz 3×5 com números inteiros, calcule e mostre a quantidade de elementos entre 15 e 20.
2. Crie um programa que preencha uma matriz 2×4 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. Elabore um programa que preencha uma matriz 6×3 , 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 15×5 ;
 - ◆ os nomes dos 15 alunos e armazene-os em um vetor de 15 posições.
 O programa deverá calcular e mostrar:
 - ◆ para cada aluno, o nome, a média aritmética das cinco provas e a situação (aprovado, reprovado ou exame);
 - ◆ a média da classe.
5. Elabore um programa que preencha uma matriz 12×4 com os valores das vendas de uma loja, em que cada linha representa um mês do ano e cada coluna representa uma semana do mês. O programa deverá calcular e mostrar:
 - ◆ 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 preencha uma matriz 20×10 com números inteiros e some cada uma das colunas, armazenando o resultado da soma em um vetor. A seguir, o programa deverá multiplicar cada elemento da matriz pela soma da coluna e mostrar a matriz resultante.
7. Elabore um programa que preencha uma matriz M de ordem 4×6 e uma segunda matriz N de ordem 6×4 , calcule e imprima a soma das linhas de M com as colunas de N.
8. Crie um programa que preencha duas matrizes 3×8 com números inteiros, calcule e mostre:
 - ◆ a soma das duas matrizes, resultando em uma terceira matriz também de ordem 3×8 ;
 - ◆ a diferença das duas matrizes, resultando em uma quarta matriz também de ordem 3×8 .
9. Faça um programa que preencha uma matriz 3×3 com números reais e outro valor numérico digitado pelo usuário. O programa deverá calcular e mostrar a matriz resultante da multiplicação do número digitado por elemento da matriz.
10. Crie um programa que preencha uma matriz 5×5 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. Elabore um programa que: receba a idade de oito alunos e armazene-as em um vetor; armazene o código de cinco disciplinas em outro vetor; armazene em uma matriz a quantidade de provas que cada aluno fez em cada disciplina.

O programa deverá calcular e mostrar:

 - ◆ a quantidade de alunos com idade entre 18 e 25 anos que fizeram mais de duas provas em uma determinada disciplina, cujo código é digitado pelo usuário. O usuário poderá digitar um código não cadastrado; neste caso, o programa deverá mostrar mensagem de erro;
 - ◆ uma listagem contendo o código dos alunos que fizeram menos que três provas em determinada disciplina, seguido do código da disciplina;
 - ◆ a média de idade dos alunos que não fizeram nenhuma prova em alguma disciplina. Cuidado para não contar duas vezes o mesmo aluno.
12. Elabore um programa que: preencha uma matriz 6×4 ; recalcule a matriz digitada, onde cada linha deverá ser multiplicada pelo maior elemento da linha em questão; mostre a matriz resultante.
13. Faça um programa que preencha uma matriz 2×3 , calcule e mostre a quantidade de elementos da matriz que não pertencem ao intervalo $[5,15]$.
14. Crie um programa que preencha uma matriz 12×13 e divida todos os elementos de cada linha pelo maior elemento em módulo daquela linha. O programa deverá escrever a matriz lida e a modificada.
15. Elabore um programa que preencha uma matriz 5×5 e crie dois vetores de cinco posições cada um, que contenham, respectivamente, as somas das linhas e das colunas da matriz. O programa deverá escrever a matriz e os vetores criados.
16. Faça um programa que preencha e mostre a média dos elementos da diagonal principal de uma matriz 10×10 .
17. Crie um programa que preencha uma matriz 5×5 de números reais, calcule e mostre a soma dos elementos da diagonal secundária.

18. Faça um programa que preencha uma matriz 8×6 de inteiros, calcule e mostre a média dos elementos das linhas pares da matriz.

19. Elabore um programa que preencha uma matriz 5×5 com números reais e encontre o maior valor da matriz. A seguir, o programa deverá multiplicar cada elemento da diagonal principal pelo maior valor encontrado e mostrar a matriz resultante após as multiplicações.

20. Faça um programa que preencha uma matriz 5×5 de números reais. A seguir, o programa deverá multiplicar cada linha pelo elemento da diagonal principal daquela linha e mostrar a matriz após as multiplicações.

21. Crie um programa que preencha uma matriz 6×10 , some as colunas individualmente e acumule as somas na 7^a linha da matriz. O programa deverá mostrar o resultado de cada coluna.

22. Faça um programa que preencha uma matriz 3×4 , calcule e mostre:

- ◆ a quantidade de elementos pares;
- ◆ a soma dos elementos ímpares;
- ◆ a média de todos os elementos.

23. Elabore um programa que preencha uma matriz 4×5 , calcule e mostre um vetor com cinco posições, onde cada posição contém a soma dos elementos de cada coluna da matriz. O programa deverá mostrar apenas os elementos do vetor maiores que dez. Se não existir nenhum elemento maior que dez, deverá mostrar uma mensagem.

24. Crie 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 5×10 .

O programa deverá calcular e mostrar:

- ◆ 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 vinte produtos em cinco lojas diferentes e armazene-os em uma matriz 20×5 . Desconsiderando empates, o programa deverá mostrar o número do produto e o número da loja do produto mais caro.