CAPÍTULO

6 Vetor

6.1 Vetor em algoritmos

6.1.1 Definição de vetor

Vetor também é conhecido como variável composta homogênea unidimensional. Isso quer dizer que se trata de um conjunto de variáveis de mesmo tipo, que possuem o mesmo identificador (nome) e são alocadas sequencialmente na memória. Como as variáveis têm o mesmo nome, o que as distingue é um índice que referencia sua localização dentro da estrutura.

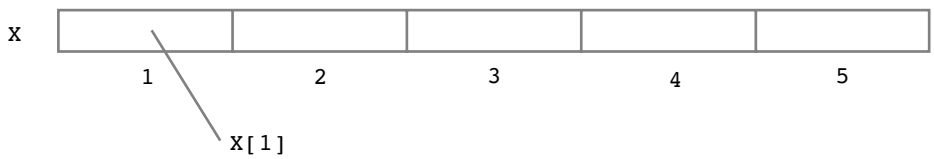
6.1.2 Declaração de vetor

```
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 Exemplos de vetor




```
DECLARE X[5] NUMÉRICO
```

Acima podemos observar a criação de um vetor chamado X, que possui cinco posições. Ou seja, foram alocadas cinco porções de memória para armazenamento de números. Essas porções de memória são contíguas, isto é, seus endereços são sequenciais.

6.1.4 Atribuindo valores ao vetor

Uma vez que todas as posições de um vetor possuem o mesmo nome, as atribuições exigem que seja informada em qual de suas posições o valor ficará armazenado.

```
X[1] ← 45
X[4] ← 0
```

No primeiro exemplo dado, o número 45 será armazenado na posição de índice 1 do vetor. Já no segundo 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 faça uma variável assumir todos os valores possíveis para o índice.

```
PARA i ← 1 ATÉ 5 FAÇA
INÍCIO
   ESCREVA "Digite o ", i, "° 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 entre 1 e 5 (posições válidas para o vetor X). Assim, para cada execução da repetição, será utilizada uma posição diferente do vetor.

Simulação:

	MEMÓRIA						TELA
i = 1	X	95					Digite o 1º número 95
		1	2	3	4	5	
i = 2	X	95	13				Digite o 2º número 13
		1	2	3	4	5	
i = 3	X	95	13	−25			Digite o 3º número −25
		1	2	3	4	5	
i = 4	X	95	13	−25	47		Digite o 4º número 47
		1	2	3	4	5	
i = 5	X	95	13	−25	47	0	Digite o 5º número 0
		1	2	3	4	5	

6.1.6 Mostrando os elementos do vetor

Mostrar os valores contidos em um vetor também implica na utilização do índice.

```
PARA i ← 1 ATÉ 5 FAÇA
INÍCIO
   ESCREVA "Este é o ", i, "° número do vetor"
   ESCREVA X[i]
FIM
```

Nesse exemplo, a estrutura de repetição PARA também foi utilizada para garantir que a variável i assuma todos os valores possíveis para o índice do vetor. Com isso, a cada execução da repetição, um valor de posição diferente será mostrado.

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 sequencialmente 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 valor correspondente ao índice da primeira posição do vetor;
`índice_final` é o valor correspondente ao índice da última posição do vetor;
`tipo_dos_dados_do_vetor` é o tipo básico dos dados que serão armazenados no vetor.

É importante salientar que o valor do `índice_inicial` deve ser maior ou igual ao valor do `índice_final`. As posições são identificadas com valores dentro desse intervalo.

Exemplo 1:

```
VAR vetor1: ARRAY [1..10] OF INTEGER;
```

Nesse caso, o índice poderá assumir valores inteiros que vão de 1 a 10.

Exemplo 2:

```
VAR vetor1: ARRAY [5..9] OF REAL;
```

Nesse caso, o índice poderá assumir valores inteiros que vão de 5 a 9.

Outro ponto importante a ser destacado é que os índices também podem ser representados por valores alfabéticos. Com isso, é permitido o uso de caracteres para representar o `valor_inicial` e o `valor_final`. Obviamente, a regra que obriga o `valor_final` ser maior ou igual ao `valor_inicial` continua valendo. O exemplo 3, a seguir, ilustra essa possibilidade.

Exemplo 3:

```
VAR vetor1: ARRAY ['C'..'G'] OF REAL;
```

Nesse caso, o índice poderá assumir valores que vão de C a G.

Observação

Os valores que indicam o `índice_inicial` e o `índice_final` devem representar valores fixos (literais[1] ou constantes), não podendo ser substituídos por variáveis.

6.2.3 Exemplos 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;
```

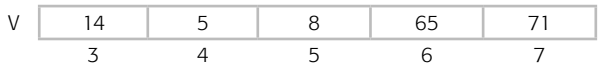
VET	E	*	m	J	k
	5	6	7	8	9

```
VAR X:ARRAY[´D´..´G´] OF INTEGER;
```

X	5	10	8	3
	D	E	F	G

1 Literal é um valor fixo, definido quando se escreve o programa. Por exemplo: `x:=10.3;` onde 10.3 é um literal. `vet: array [1..18] of char;` onde 1 e 18, escritos dentro dos colchetes, são literais.

```
CONST MIN = 3;
CONST MAX = 7;
VAR V:ARRAY[MIN..MAX] OF INTEGER;
```



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.
`Y['d']:=4.1;` atribui o valor 4.1 à posição do vetor cujo índice é o caractere d.

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.

Exemplo 1:

```
FOR i:= 1 TO 7 DO
BEGIN
   READLN(X[i]);
END;
```

Exemplo 2:

```
FOR i:= 'C' TO 'E' DO
BEGIN
   READLN(X[i]);
END;
```

O exemplo 1 apresentou uma estrutura de repetição `FOR`, que foi utilizada para garantir que a variável `i` assumisse todos os valores possíveis para o índice do vetor (de 1 a 7). Já o exemplo 2 utilizou uma estrutura de repetição `for` para garantir que a variável `i` assumisse todos os valores possíveis entre os caracteres `C` e `E`. Assim, para cada execução da repetição, uma posição diferente dos vetores será preenchida por um valor digitado pelo usuário.

6.2.6 Mostrando os elementos do vetor

Mostrar os valores contidos em um vetor também exige a utilização do índice.

Exemplo 1:

```
FOR i:=1 TO 10 DO
BEGIN
   WRITELN(X[i]);
END;
```

Exemplo 2:

```
FOR i:= 'C' TO 'E' DO
BEGIN
   WRITELN(X[i]);
END;
```

O exemplo 1 apresentou uma estrutura de repetição FOR, que foi utilizada para garantir que a variável i assumisse todos os valores possíveis para o índice do vetor (de 1 a 10). Já no exemplo 2, a estrutura de repetição FOR garantiu que a variável i assumisse todos os valores possíveis entre os caracteres C e E. Assim, para cada execução da repetição, foi utilizada uma posição diferente do vetor e, dessa forma, todos os valores armazenados foram mostrados.

6.3 Vetor em C/C++

6.3.1 Definição de vetor

As variáveis compostas homogêneas unidimensionais (ou, simplesmente, vetores) são capazes de armazenar diversos valores. Cada um desses valores é identificado pelo mesmo nome (o nome dado ao vetor), sendo diferenciados entre si 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. O índice de um vetor em C/C++ deve sempre ser representado por um dos tipos inteiros disponíveis na linguagem.

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 elementos que o vetor poderá armazenar.

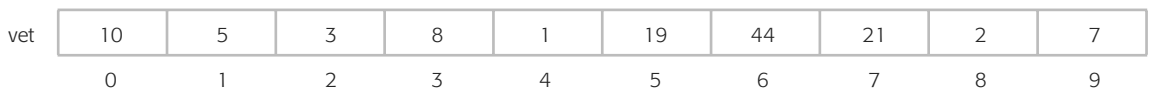
Em C/C++, a indicação do tamanho do vetor (ou seja, a quantidade de elementos que o vetor poderá armazenar) deve ser feita por um valor inteiro fixo (representado por um literal[2] ou uma constante). Se houver necessidade de definir o tamanho do vetor em tempo de execução, deve-se fazê-lo através de ponteiros (o Capítulo 8 apresentará o conceito de ponteiro).

6.3.3 Exemplo de vetor

A seguir, são apresentadas algumas formas de criação de vetores.

Exemplo 1:

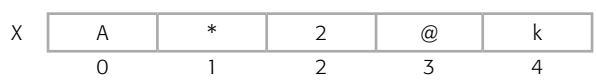
```
int vet[10];
```



No exemplo 1, 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.

Exemplo 2:

```
char x[5];
```

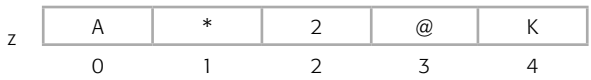


No exemplo 2, o vetor chamado x possui cinco posições, começando pela posição 0 e indo até a posição 4 (tamanho do vetor – 1). Em cada posição poderão ser armazenados caracteres, conforme especificado pelo tipo char na declaração.

2 Literal é um valor fixo, definido quando se escreve o programa. Por exemplo: `double x=10.3;` onde 10.3 é um literal. `char vet [18];` onde 18, escrito dentro dos colchetes, é um literal.

Exemplo 3:

```
#define tam 5;
char z[tam];
```



No exemplo 3, o vetor z tem tamanho 5, exatamente o valor definido para a constante chamada `tam`.

É importante ressaltar que, na linguagem C, não existe o tipo de dado `string` (que será visto em detalhes no Capítulo 9), como ocorre na linguagem PASCAL. Dessa maneira, por exemplo, para poder armazenar o nome completo de uma pessoa, em uma cadeia de caracteres, deve-se declarar um vetor de `char`, em que cada posição equivale a um caractere ou a uma letra do nome. Além disso, 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, pois esta armazenará a marca de finalização de cadeia, representada pelo caractere `'\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. Lembre-se: sempre 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 de que o vetor começa na posição zero).
`x[3] = 'b';`	atribui a letra b à quarta posição do vetor (lembre-se de que o vetor começa na posição zero).

6.3.5 Preenchendo um vetor

Preencher um vetor significa atribuir valores às suas posições. Assim, deve-se implementar um mecanismo que controle o valor do índice.

```
for (i=0; i<10; i++)
  scanf("%d%*c", &vetor[i]);
```

Nesse exemplo, a estrutura de repetição `FOR` foi utilizada para garantir que a variável `i` assumisse 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 de um índice.

```
for (i=0; i<10; i++)
  printf("%d", vetor[i];
```

Nesse exemplo, a estrutura de repetição `FOR` foi utilizada para garantir que a variável `i` assumisse todos os valores possíveis para o índice do vetor (de 0 a 9). Assim, para cada execução da repetição, foi utilizada uma posição diferente e, dessa forma, todos os valores do vetor foram mostrados.

6.4 Vetor em JAVA

6.4.1 Definição de vetor

As variáveis compostas homogêneas unidimensionais (vetores) são variáveis capazes de armazenar diversos valores. Cada um desses valores é identificado pelo mesmo nome (o nome dado ao vetor), sendo diferenciados entre si 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.

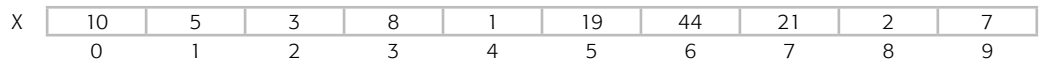
Em JAVA, a indicação do tamanho do vetor (ou seja, a quantidade de elementos que o vetor poderá armazenar) pode ser feita por um valor inteiro fixo (representado por um literal[3] ou uma constante) ou por uma variável cujo valor é definido em tempo de execução.

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:

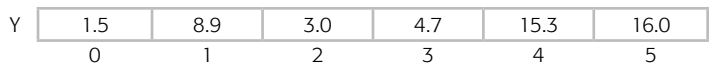
```
int x[];
x = new int[10];
```



Na primeira linha do exemplo 1, 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 estabelece que o vetor `x` terá tamanho 10 (ou seja, posições de 0 a 9).

Exemplo 2:

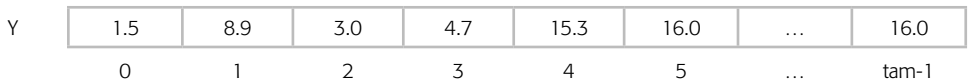
```
final int tam=6;
float []y;
y = new float[tam];
```



Na primeira linha do exemplo 2, foi definida a constante `tam`, com valor igual a 6. Na segunda linha, os colchetes vazios antes do nome definem que `y` será um vetor. O tipo `float` determina o tipo do conteúdo que poderá ser armazenado em todas as suas posições. A terceira linha estabelece que o vetor `y` terá tamanho 6, exatamente o valor da constante `tam` (ou seja, o vetor terá posições de 0 a 5).

Exemplo 3:

```
double w[];
int tam;
tam = ent.nextInt();
w = new double[tam];
```



No exemplo 3, tem-se um vetor cujo tamanho dependerá de um valor fornecido no momento da execução do programa. Na primeira linha do exemplo 3, os colchetes vazios depois do nome, definem que `w` será um vetor. O tipo `double`, determina o tipo do dado que poderá ser armazenado em todas as suas posições. Na segunda linha é declarada a variável `tam`, que, após receber um valor externo (terceira linha), indicará o tamanho do vetor `w` (quarta linha).

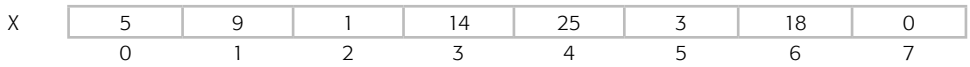
Já, nos exemplos apresentados a seguir, utilizou-se a forma condensada, onde a declaração e o dimensionamento do vetor são feitos utilizando-se uma única linha.

---

3 Literal é um valor fixo, definido quando se escreve o programa. Por exemplo: `double x=10.3;` onde 10.3 é um literal. `char vet []= new char[18];` onde 18, escrito dentro dos colchetes, é um literal.

Exemplo 4:

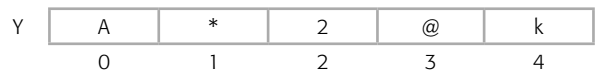
```
final int tam = 8;
int x[] = new int[tam];
```



O exemplo 4 faz uso da constante `tam` para especificar o tamanho do vetor `x`. Na segunda linha do exemplo, 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 (posições de 0 a 7).

Exemplo 5:

```
char []y = new char[5];
```



O exemplo 5 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 (posições de 0 a 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 de que o vetor começa na posição 0).
`x[3] = 'b';`	atribui o valor b à quarta posição do vetor (lembre-se de 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` assumisse 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 na 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` assumisse 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.
```

PASCAL Solução:

```
\EXERC\CAP6\PASCAL\EX1.PAS e \EXERC\CAP6\PASCAL\EX1.EXE
```

C/C++ Solução:

```
\EXERC\CAP6\C++\EX1.CPP e \EXERC\CAP6\C++\EX1.EXE
```

JAVA 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$ 545,00, 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, deverão ser mostrados o valor geral das vendas e o valor da comissão que será paga ao vendedor; e
- o valor do objeto mais vendido e sua posição no vetor (não se preocupe com empates).

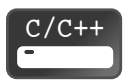
ALGORITMO Solução:

```
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	4	4	5	6	7	8	9	10

Vetor 2	7	15	20	0	18	4	55	23	8	6
	1	2	4	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

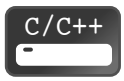
ALGORITMO Solução:

```
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 e o segundo, os números negativos. Cada vetor resultante vai ter, no máximo, oito posições, que não poderão ser completamente utilizadas.

A L G O R I T M O 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 intersecção 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

Intersecção	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
                 INÍCIO
                        k  ← k + 1
                 FIM
                SE k >= cont_i
                   ENTÃO INÍCIO
                           IT[cont_i] ← X[i]
```

```
                          cont_i ← cont_i + 1
                        FIM

              FIM

   FIM
PARA i ← 1 ATÉ cont_i – 1 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
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:

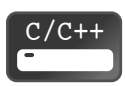
```
ALGORITMO
DECLARE vet[8], i, j, z, aux NUMÉRICO
i ← 1
ENQUANTO (i <= 8) FAÇA
INÍCIO
LEIA aux
j ←  1
ENQUANTO (j < i  E vet[j] < aux) FAÇA
   INÍCIO
      j ← j + 1
   FIM
z ← i
ENQUANTO (z >= j+1) FAÇA
   INÍCIO
      vet[z] ← vet[z-1]
      z ← z - 1
   FIM
vet[j] ← aux
i ← i + 1
FIM
PARA i ← 1 ATÉ 8 FAÇA
   INÍCIO
      ESCREVA vet[i]
   FIM
FIM_ALGORITMO.
```

1ª solução – utilizando apenas WHILE:

\EXERC\CAP6\PASCAL\EX7_A.PAS **e** \EXERC\CAP6\PASCAL\EX7_A.EXE

2ª solução – utilizando FOR e WHILE:

\EXERC\CAP6\PASCAL\EX7_B.PAS **e** \EXERC\CAP6\PASCAL\EX7_B.EXE



1ª solução – utilizando apenas WHILE:

\EXERC\CAP6\C++\EX7_A.CPP **e** \EXERC\CAP6\C++\EX7_A.EXE

2ª solução – utilizando FOR e WHILE:

\EXERC\CAP6\C++\EX7_B.CPP **e** \EXERC\CAP6\C++\EX7_B.EXE

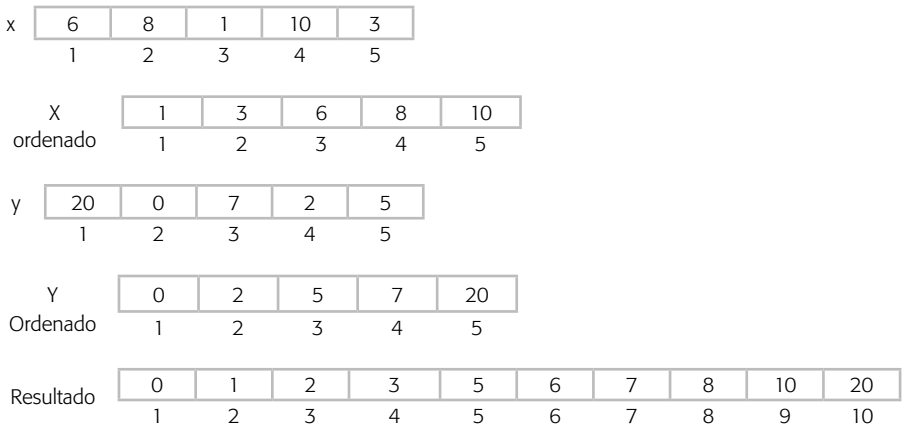
1ª solução – utilizando apenas WHILE:

\EXERC\CAP6\JAVA\EX7_A.java **e** \EXERC\CAP6\JAVA\EX7_A.class

2ª 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.



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
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
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
   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 voos (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; e
- sair.

Quando a opção escolhida for *Consultar*, deverá ser disponibilizado mais um menu com as seguintes opções:

- por número do voo;
- por origem; e
- por destino.

Quando a opção escolhida for *Efetuar reserva*, deverá ser perguntado o número do voo em que a pessoa deseja viajar. O programa deverá dar as seguintes respostas:

- reserva confirmada — caso exista o voo e lugar disponível, dando baixa nos lugares disponíveis;
- voo lotado — caso não exista lugar disponível nesse voo;
- voo inexistente — caso o código do voo 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 voo"
            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 voo: "
                     LEIA num_voo
                     i ← 1
                     ENQUANTO (i <= 12 E voo[i] ≠ num_voo) FAÇA
                     INÍCIO
                        i ← i + 1
                     FIM
                     SE i > 12
                     ENTÃO ESCREVA "Voo inexistente "
                     SENÃO 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
            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 voo: ", 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
             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 "Voo lotado "
                        SENÃO INÍCIO
                                 lugares[i] = lugares[i] - 1
                                 ESCREVA "Reserva confirmada !"
                              FIM
                     FIM
         FIM
ATÉ (op = 3)
FIM_ALGORITMO.
```

Observações

A comparação de duas cadeias de caracteres (como dois nomes, por exemplo) em PASCAL é feita utilizando-se o sinal de = . As funções de manipulação de `strings` desta linguagem serão abordadas no Capítulo 9.

Exemplo:

```
{faz distinção entre maiúsculas e minúsculas}
if (nome1 = nome2)
then writeln('Nomes iguais');
```

A comparação de duas cadeias de caracteres (como dois nomes, por exemplo) em C é feita utilizando-se algumas funções da biblioteca `string.h`. As funções de manipulação de `strings` desta linguagem serão abordadas no Capítulo 9.

Exemplos:

```
// faz distinção entre maiúsculas e minúsculas
if (strcmp(nome1,nome2)==0)
 printf("Nomes iguais");

// NÃO faz distinção entre maiúsculas e minúsculas
if (strcmpi(nome1,nome2)==0)
 printf("Nomes iguais");
```

A comparação de duas cadeias de caracteres (como dois nomes, por exemplo) em JAVA é feita utilizando-se alguns métodos da classe `String`. Os métodos de manipulação de `strings` desta linguagem serão abordados no Capítulo 9.

Exemplos:

```
// faz distinção entre maiúsculas e minúsculas
if (nome1.equals(nome2))
 System.out.println("Nomes iguais");

// NÃO faz distinção entre maiúsculas e minúsculas
if (nome1.equalsIgnoreCase(nome2))
 System.out.println("Nomes iguais");
```

PASCAL

Solução:

`\EXERC\CAP6\PASCAL\EX9.PAS` **e** `\EXERC\CAP6\PASCAL\EX9.EXE`

C/C++

Solução:

`\EXERC\CAP6\C++\EX9.CPP` **e** `\EXERC\CAP6\C++\EX9.EXE`

JAVA

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:

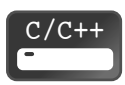
- o número e a nota de cada aluno; e
- a porcentagem de aprovação, sabendo-se que a nota mínima é 6.

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
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; e
- quantos litros de combustível cada um dos carros cadastrados consome para percorrer uma distância de 1.000 km.

ALGORITMO Solução:

```
ALGORITMO
DECLARE consumo[5], menor_cons, menor_vei, valor, i NUMÉRICO
        veiculo[5] LITERAL
PARA i ← 1 ATÉ 5 FAÇA
   INÍCIO
      LEIA veiculo[i]
   FIM
PARA i ← 1 ATÉ 5 FAÇA
   INÍCIO
      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

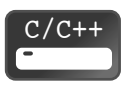
13. Faça um programa que preencha um vetor com dez números inteiros, calcule e mostre os números superiores a cinquenta e suas respectivas posições. O programa deverá mostrar mensagem se não existir nenhum número nessa condição.

ALGORITMO SOLUÇÃO:

```
ALGORITMO
   DECLARE vet[10], 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$ 800,00. 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 satisfazer apenas à condição tempo de serviço, o aumento será de 25%; para aquele que satisfazer 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] >= 800)
   ENTÃO ESCREVA nome[i]
   FIM
PARA i ← 1 ATÉ 5 FAÇA
   INÍCIO
   SE (quant[i] > 5) OU (sal[i] < 800)
   ENTÃO INÍCIO
            SE (quant[i] > 5) E (sal[i] < 800)
            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.
```

PASCAL Solução:

\EXERC\CAP6\PASCAL\EX14.PAS **e** \EXERC\CAP6\PASCAL\EX14.EXE

C/C++ Solução:

\EXERC\CAP6\C++\EX14.CPP **e** \EXERC\CAP6\C++\EX14.EXE

JAVA 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], 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.
```

PASCAL Solução:

\EXERC\CAP6\PASCAL\EX15.PAS **e** \EXERC\CAP6\PASCAL\EX15.EXE

C/C++ Solução:

\EXERC\CAP6\C++\EX15.CPP **e** \EXERC\CAP6\C++\EX15.EXE

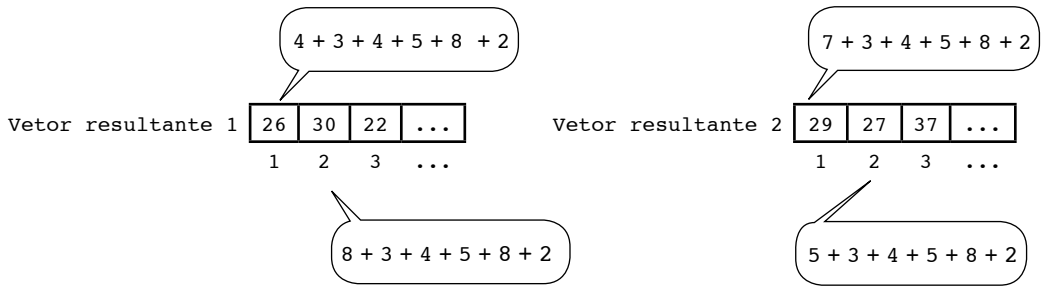
JAVA Solução:

\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



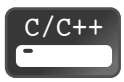
ALGORITMO Solução:

```
ALGORITMO
   DECLARE vet1[10], vet2[5] NUMÉRICO
           vet_result1[10], vet_result2[10] NUMÉRICO
           i, j, poslivre1, poslivre2, soma NUMÉRICO
PARA i ← 1 ATÉ 10 FAÇA
   INÍCIO
   LEIA vet1[i]
   FIM
PARA j ← 1 ATÉ 5 FAÇA
   INÍCIO
   LEIA vet2[j]
   FIM
poslivre1 ← 1
poslivre2 ← 1
PARA i ← 1 ATÉ 10 FAÇA
   INÍCIO
   soma ← vet1[i]
   PARA j ← 1 ATÉ 5 FAÇA
      INÍCIO
      soma ← soma + vet2[j]
      FIM
   SE RESTO(soma/2) = 0
   ENTÃO INÍCIO
         vet_result1[poslivre1] ← soma
         poslivre1 ← poslivre1 + 1
         FIM
```

```
   SENÃO INÍCIO
         vet_result2[poslivre2] ← soma
         poslivre2 ← poslivre2 + 1
         FIM
   FIM
PARA i ← 1 ATÉ (poslivre1 -1) FAÇA
   INÍCIO
   ESCREVA vet_result1[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; e
- 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
```

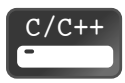
ALGORITMO Solução:

```
ALGORITMO
   DECLARE num[6], 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 dos pares = ",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 "Quantidade de ímpares = ",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 frequência com que apareceram.

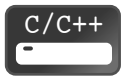
ALGORITMO Solução:

```
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 "O número 1 foi sorteado ", num1, "vez(es)"
ESCREVA "O número 2 foi sorteado ", num2, "vez(es)"
ESCREVA "O número 3 foi sorteado ", num3, "vez(es)"
ESCREVA "O número 4 foi sorteado ", num4, "vez(es)"
ESCREVA "O número 5 foi sorteado ", num5, "vez(es)"
ESCREVA "O número 6 foi sorteado ", num6, "vez(es)"
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.
```

PASCAL Solução:

\EXERC\CAP6\PASCAL\EX19.PAS **e** \EXERC\CAP6\PASCAL\EX19.EXE

C/C++ Solução:

\EXERC\CAP6\C++\EX19.CPP **e** \EXERC\CAP6\C++\EX19.EXE

JAVA 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.

ALGORITMO Solução:

```
ALGORITMO
      DECLARE vet[5], 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.
```

PASCAL Solução:

\EXERC\CAP6\PASCAL\EX20.PAS **e** \EXERC\CAP6\PASCAL\EX20.EXE

C/C++ Solução:

\EXERC\CAP6\C++\EX20.CPP **e** \EXERC\CAP6\C++\EX20.EXE

JAVA 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 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 o seu saldo é suficiente para cobrir o saque. (Estamos supondo que a conta não possa 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*.

ALGORITMO Solução:

```
ALGORITMO
   DECLARE conta[10], saldo[10] NUMÉRICO
           i, j, codigo, valor, soma, op NUMÉRICO
           achou LÓGICO
PARA i ← 1 ATÉ 10 FAÇA
       INÍCIO
       achou ← falso
       REPITA
       LEIA conta[i]
       PARA j ← 1 ATÉ (i-1) FAÇA
              INÍCIO
              SE conta[i] = conta[j]
              ENTÃO achou ← verdadeiro
              FIM
       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.
```

PASCAL Solução:

\EXERC\CAP6\PASCAL\EX22.PAS **e** \EXERC\CAP6\PASCAL\EX22.EXE

C/C++ Solução:

\EXERC\CAP6\C++\EX22.CPP **e** \EXERC\CAP6\C++\EX22.EXE

JAVA Solução:

\EXERC\CAP6\JAVA\EX22.java **e** \EXERC\CAP6\C++\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
   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
      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
   ATÉ op = 3
   FIM_ALGORITMO.
```

PASCAL Solução:

```
\EXERC\CAP6\PASCAL\EX23.PAS e \EXERC\CAP6\PASCAL\EX23.EXE
```

C/C++ Solução:

```
\EXERC\CAP6\C++\EX23.CPP e \EXERC\CAP6\C++\EX23.EXE
```

JAVA 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.

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 2 vezes;
- o número 4 aparece 3 vezes;
- o número 3 aparece 2 vezes;
- o número 18 aparece 3 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.

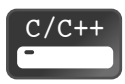
ALGORITMO Solução:

```
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 dessa 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, quinze alunos.

Coloque os números das matrículas dos alunos que cursam Linguagem de Programação em outro vetor, 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.

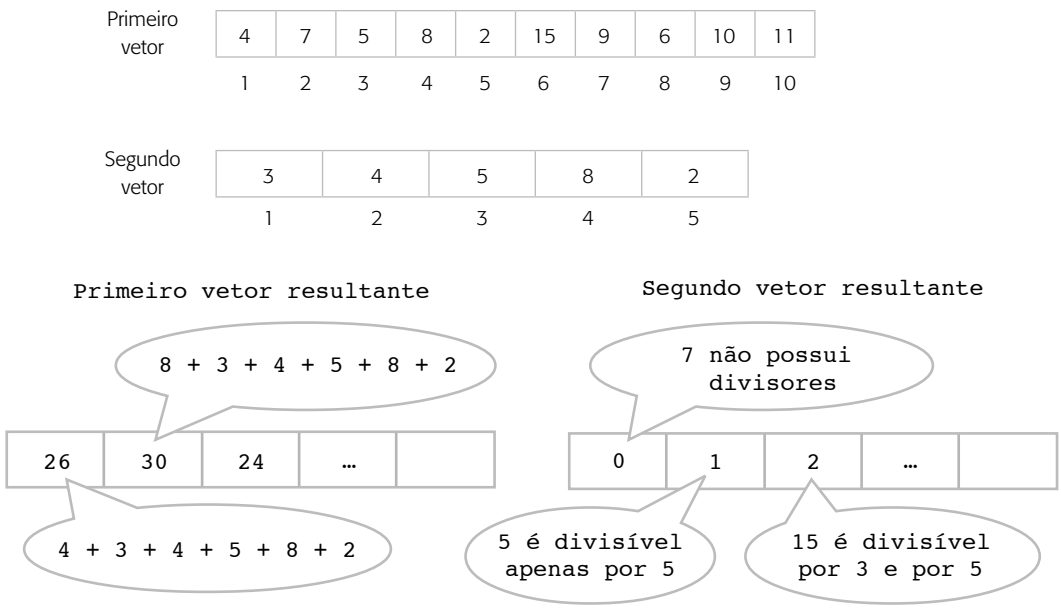
8. Faça um programa que preencha um vetor com os nomes de sete alunos e carregue outro vetor com a média final desses alunos. Calcule e mostre:

- o nome do aluno com maior média (desconsiderar empates);
- para cada aluno não aprovado, isto é, com média menor 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 satisfazem as duas condições anteriores, código e preço, o aumento será de 20%; para aqueles que satisfazem apenas a condição de código, o aumento será de 15%; e para aqueles que satisfazem apenas a 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.



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
Carlos 8.0
Pedro 5.0
..
..
..
Média da classe = ??

14. Faça um programa que receba o nome e duas notas de seis alunos e mostre o relatório 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 2011 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, 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 dez posições para números inteiros e mostre somente os números positivos.

21. Faça um programa que leia um vetor com dez 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.