

## Capítulo 8 – Grupos de dados homogêneos

### Introdução

Em certas situações são necessárias variáveis indexadas, ou subscritas.

Pode-se representar as coordenadas de um ponto no espaço tridimensional, por exemplo, por um vetor contendo valores correspondentes às coordenadas nos eixos estabelecidos :

$$P = [x \ y \ z]$$

Pode-se representar os termos de uma progressão aritmética, por  $a_1, a_2, a_3, \dots, a_n$  em uma matriz de (m) linhas e (n) colunas :

$$A_{m \times n} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & & & \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}$$

Variáveis assim agrupadas permitem representar certas quantidades de dados, similares em tipo, por um mesmo nome. Cada elemento, em particular, é endereçado através de índices, que indicam uma posição (ou endereço), ou número de ordem desse elemento no conjunto. Por exemplo, o primeiro elemento do vetor P indica a coordenada na direção do eixo (x); o elemento  $a_{11}$  da matriz A indica o primeiro termo de uma determinada progressão.

De forma genérica, dados de esma natureza, agrupados, são chamados **arranjos** (ou tabelas; vetores ou matrizes).

x	y = 2x
1	2
2	4
3	6
4	8
5	10

nome	ramal
André	101
Bruna	215
Célio	112
Délia	119
Edir	314

Abaixo encontram-se sugestões de modelos para esses tipos de dados (ou seus agrupamentos) onde cada dado ocupa uma posição (ou endereço), geralmente com início em zero (primeira posição) e terminando em (n-1) (última posição). No caso de agrupamentos maiores serão usados mais índices para linhas (ou grupos) e colunas (posição no grupo), por exemplo.

	0	1	2	...	(n-1)
	dado (1)	dado (2)	dado (3)	...	dado (n)

/	0	1	2	...	(n-1)	coluna
linha						
0	dado (1,1)	dado (1,2)	dado (1,3)	...	dado (1,n)	
1	dado (2,1)	dado (2,2)	dado (2,3)	...	dado (2,n)	
2	dado (1,1)	dado (1,2)	dado (3,3)	...	dado (3,n)	
...						
(m-1)	dado (m,1)	dado (m,2)	dado (m,3)	...	dado (m,n)	

### Definição de dados indexados

A forma geral para se definir uma variável agrupada é a mesma de uma variável simples, seguida pela definição do número de elementos do grupo, por dimensão. Este número costuma ser um valor constante. O primeiro valor ocupará a posição de índice igual a zero (0).

Exemplos :

```
inteiro P [3];           ! vetor com 3 valores inteiros
real A [5] [3], B [5 ][3]; ! matriz com 5 linhas e 3 colunas de valores reais
caractere S1[10], S2[10], S3[10]; ! cadeia de caracteres capaz de armazenar até 9 letras
```

Observação:

No caso de cadeias de caracteres é comum reservar a última posição para guardar um símbolo especial (  $\epsilon = '\text{'0'}$  ), que serve para representar o fim da seqüência de caracteres.

Uma outra forma, mais estruturada e mais simples, de definição é descrever, primeiro, um tipo agrupado homogêneo; e depois usá-lo para se definir variáveis.

Exemplos :

```
tipo VETOR = inteiro [3];      ! vetor com 3 valores inteiros
tipo MATRIZ = real [5] [3];    ! matriz com 5 linhas e 3 colunas de valores reais
tipo PALAVRA = caractere[10], ! cadeia de caracteres capaz de armazenar até 9 letras
```

```
VETOR P;                      ! um vetor com 3 valores inteiros
MATRIZ A, B;                  ! duas matrizes com 5 linhas e 3 colunas cada
PALAVRA S1, S2, S3;          ! três palavras com até 9 letras cada
```

### Acesso a elementos

O acesso individual a cada elemento pode ser feito mediante o uso, entre colchetes, de uma constante, uma variável inteira (ou valor ordinal), ou por expressão, cujo valor resultante também seja inteiro (ou ordinal).

Os índices podem indicar a posição de um elemento em um conjunto linear (vetor), planar (matriz), espacial (cubo) ou multidimensional. Neste caso, cada dimensão é separada das outras por colchetes.

Exemplos :

Com as definições anteriores, os acessos individuais a elementos podem ser:

```
P [ 0 ] = 1;                ! para atribuir valor ao primeiro elemento do vetor
```

```
tela ← P [ 0 ];             ! para exibir  valor do primeiro elemento do vetor
```

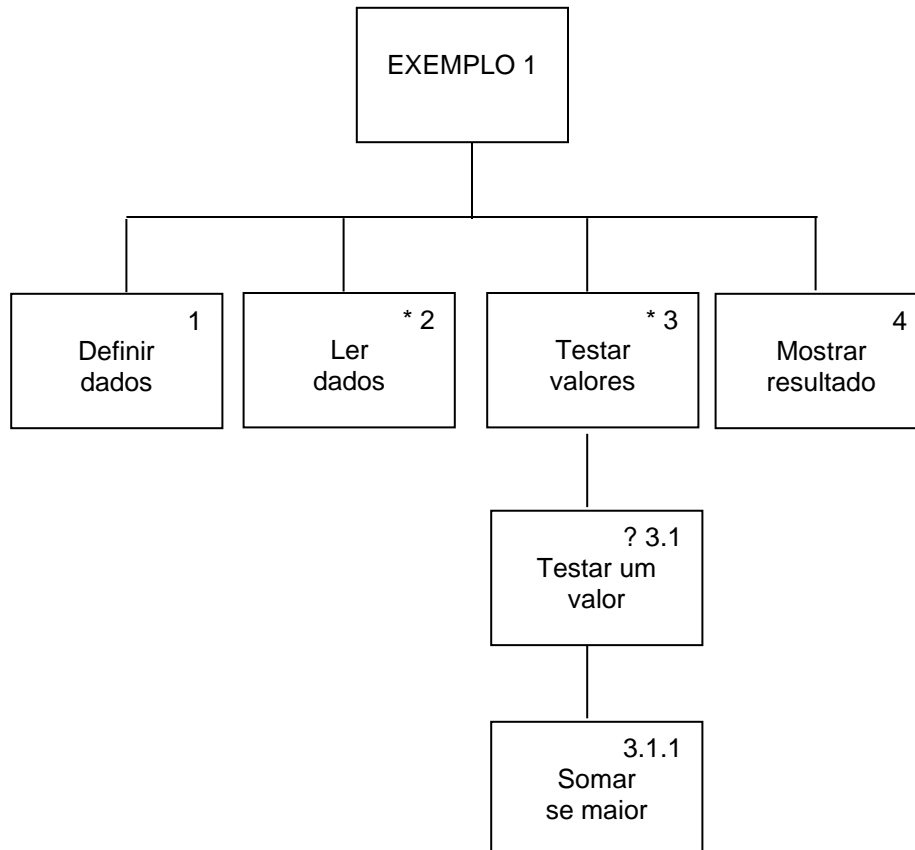
```
repetir para ( X = 0 : 1 : 5 )
  repetir para ( Y = 0 : 1 : 3 )
    A [ X ] [ Y ] = 0.0;     ! para atribuir zero a todos os elementos da matriz
  fim repetir para ! Y
fim repetir para ! X
```

```
S1 [ 0 ] =  $\epsilon$  ;        ! para indicar uma palavra vazia
```

## Exemplo 1.

Montar uma tabela de, no máximo 10, resistores. Os valores serão lidos do teclado, e deseja-se verificar quantos valores são superiores a 10 ohms.

Diagrama funcional:



Análise de dados :

- Dados do problema :

Dado	Tipo	Valor inicial	Função
RESISTOR	inteiro (10)	-	armazenar valores de 10 resistores
I	inteiro	-	índice

- Resultados do problema :

Dado	Tipo	Valor inicial	Função
CONTADOR	inteiro	0	contador de quantos são maiores que 10 ohms

- Avaliação da solução :

Índice	Dado	Resultado
0	1	
1	2	
2	3	
3	5	
4	12	√
5	10	
6	15	√
7	5	
8	25	√
9	30	√
		4

Algoritmo:

Esboço:

Primeira versão, só comentários.

Exemplo 1	v.1
Ação	Bloco
! definir dados	1
! ler dados	2
! testar valores	3
! testar um valor	3.1
! somar se maior que 10 ohms	3.1.1
! mostrar o resultado	4

Segunda versão, com refinamento do primeiro bloco.

Exemplo 1	v.2
Ação	Bloco
! definir dados	1
inteiro RESISTOR[10], ! tabela de resistores X, ! índice CONTADOR=0; ! para os maiores que 10 ohms	
! ler dados	2
! testar valores	3
! testar um valor	3.1
! somar se maior que 10 ohms	3.1.1
! mostrar o resultado	4

Terceira versão, com refinamento do segundo bloco.

Exemplo 1	v.3
Ação	Bloco
! definir dados	1
inteiro RESISTOR[10], ! tabela de resistores X, ! índice CONTADOR=0; ! para os maiores que 10 ohms	
! ler dados	2
repetir para ( X = 1 : 10 : 1 )	2.1
! ler e guardar o valor de um resistor	2.1.1
! testar valores	3
! testar um valor	3.1
! somar se maior que 10 ohms	3.1.1
! mostrar o resultado	4

Quarta versão, com refinamento do terceiro bloco.

Exemplo 1	v.4
Ação	Bloco
! definir dados	1
inteiro RESISTOR[10], ! tabela de resistores X, ! índice CONTADOR=0; ! para os maiores que 10 ohms	
! ler dados	2
X = 1 : 10 : 1	2.1
! ler e guardar o valor de um resistor	2.1.1
! testar valores	3
X = 1 : 10 : 1	
! testar um valor	3.1
RESISTOR [X] > 10 ?	3.1.1
! somar se maior que 10 ohms CONTADOR=CONTADOR+1;	
! mostrar o resultado	4

Quinta versão, com refinamento do quarto bloco.

Exemplo 1	v.5
Ação	Bloco
! definir dados	1
inteiro RESISTOR[10], ! tabela de resistores X, ! índice CONTADOR=0; ! para os maiores que 10 ohms	
! ler dados	2
X = 1 : 10 : 1	2.1
! ler e guardar o valor de um resistor	2.1.1
! testar valores	3
X = 1 : 10 : 1	
! testar um valor	3.1
RESISTOR [X] > 10 ?	3.1.1
! somar se maior que 10 ohms CONTADOR=CONTADOR+1;	
! mostrar o resultado	4
! mostrar todos os valores	4.1
X = 1 : 10 : 1	
! mostrar cada valor	
! mostrar quantos são maiores que 10 ohms	4.2

Sexta versão, com refinamento do segundo, terceiro e quarto blocos.

Exemplo 1	v.6
Ação	Bloco
! definir dados	1
inteiro RESISTOR[10], ! tabela de resistores X, ! índice CONTADOR=0; ! para os maiores que 10 ohms	
! ler dados	2
repetir para ( X = 0 : 10 : 1 )	2.1
! ler e guardar o valor de um resistor tela ← ("Qual o valor do resistor ", X , " ? "); RESISTOR [ X ] ← teclado;	2.1.1
fim repetir	
! testar valores	3
repetir para ( X = 0 : 10 : 1 )	3.1
! testar um valor se ( RESISTOR [ X ] > 10 )	
! somar se maior que 10 ohms CONTADOR=CONTADOR+1;	3.1.1
fim se	
fim repetir	
! mostrar o resultado	4
! mostrar todos os valores repetir para ( X = 0 : 10 : 1 )	4.1
! ler e guardar o valor de um resistor tela ← ("n", X , " ", RESISTOR [ X ] );	
fim repetir	
! mostrar quantos são maiores que 10 ohms tela ← ("Maiores que 10 = ", CONTADOR);	4.2

Programa em SCILAB:

```
// Exemplo 1.
// Calcular quantos resistores, em uma tabela, são maiores que 10 ohms.
//
// 1. definir dados
RESISTOR = [ 0 0 0 0 0 0 0 0 0 0 ]; // tabela de resistores
X = 0; // índice
CONTADOR = 0; // para os maiores que 10 ohms
//
// 2. ler dados
for ( X = 1 : 1 : 10 )
    // 2.1 ler e guardar o valor de um resistor
    printf ( " \nQual o valor do resistor %d ? ", X );
    RESISTOR ( X ) = input ( " " );
end // for

// 3. testar valores
for ( X = 1 : 1 : 10 )
    // 3.1 testar um valor
    if ( RESISTOR ( X ) > 10 )
        // 3.1.1 somar se maior que 10 ohms
        CONTADOR=CONTADOR+1;
    end // if
end // for

// 4. mostrar o resultado
// 4.1 mostrar todos os valores
for ( X = 1 : 1 : 10 )
    // 4.1.1 mostrar cada valor de resistor
    printf ( " \n %d \t %f ", X, RESISTOR ( X ) );
end // for

// 4.2 mostrar quantos são maiores que 10 ohms
printf ( " \nMaiores que 10 = %d ", CONTADOR );

// pausa para terminar
printf ( " \nPressionar ENTER para terminar. " );
halt;
// fim do programa
```

Programa em C:

```
// Exemplo 1a.
// Calcular quantos resistores, em uma tabela, são maiores que 10 ohms.
//
// bibliotecas necessárias
#include <stdio.h>
#include <stdlib.h>
//
// parte principal
//
int main (void)
{
// 1. definir dados
    int RESISTOR [ 10 ],    // tabela de resistores
        X,                // índice
        CONTADOR = 0;      // para os maiores que 10 ohms
// 2. ler dados
    for ( X = 0; X < 10; X = X + 1 )
    { // 2.1 ler e guardar o valor de um resistor
        printf ( "\nQual o valor do resistor %d%s", X, " ? " );
        scanf ( "%d", &RESISTOR [ X ] );
    } // fim for

// 3. testar valores
    for ( X = 0; X < 10; X = X + 1 )
    { // 3.1 testar um valor
        if ( RESISTOR [ X ] > 10 )
        {
            // 3.1.1 somar se maior que 10 ohms
            CONTADOR=CONTADOR+1;
        } // fim if
    } // fim for

// 4. mostrar o resultado
// 4.1 mostrar todos os valores
    for ( X = 0; X < 10; X = X + 1 )
    { // 4.1.1 mostrar cada valor de resistor
        printf ( "\n%d%s%d", X, " ", RESISTOR [ X ] );
    } // fim for

// 4.2 mostrar quantos são maiores que 10 ohms
    printf ( "\nMaiores que 10 = %d ", CONTADOR;

// pausa para terminar
    printf ( "\nPressionar ENTER para terminar." );
    getchar ( );
    return ( 0 );
} // fim do programa
```



Programa em C++:

```
// Exemplo 1a.
// Calcular quantos resistores, em uma tabela, são maiores que 10 ohms.
//
// bibliotecas necessárias
#include <iostream>
using namespace std;
//
// parte principal
//
int main (void)
{
// 1. definir dados
    int RESISTOR [ 10 ],    // tabela de resistores
        X,                 // índice
        CONTADOR = 0;      // para os maiores que 10 ohms
// 2. ler dados
    for ( X = 0; X < 10; X = X + 1 )
    { // 2.1 ler e guardar o valor de um resistor
        cout << "\nQual o valor do resistor " << X << " ? ";
        cin >> RESISTOR [ X ];
    } // fim for

// 3. testar valores
    for ( X = 0; X < 10; X = X + 1 )
    { // 3.1 testar um valor
        if ( RESISTOR [ X ] > 10 )
        {
            // 3.1.1 somar se maior que 10 ohms
            CONTADOR=CONTADOR+1;
        } // fim if
    } // fim for

// 4. mostrar o resultado
// 4.1 mostrar todos os valores
    for ( X = 0; X < 10; X = X + 1 )
    { // 4.1.1 mostrar cada valor de resistor
        cout << "\n" << X << " " << RESISTOR [ X ];
    } // fim for

// 4.2 mostrar quantos são maiores que 10 ohms
    cout << "\nMaiores que 10 = " << CONTADOR;

// pausa para terminar
    cout << "\nPressionar ENTER para terminar.";
    cin.get ( );
    return EXIT_SUCCESS;
} // fim do programa
```

Outra versão com procedimentos e função.

Programa em C++:

```
// Exemplo 1b.
// Calcular quantos resistores, em uma tabela, sao maiores que 10 ohms.
//
// bibliotecas necessarias
#include <iostream>
using namespace std;
//

// procedimento para ler dados
void LER ( int RESISTOR [ ] )
{
    // definir dado local
    int X;
    // ler dados
    for ( X = 0; X < 10; X = X + 1 )
    { // ler e guardar o valor de um resistor
        cout << "\nQual o valor do resistor " << X << " ? ";
        cin  >> RESISTOR [ X ];
    } // fim for
} // fim do procedimento LER ( )

// procedimento para mostrar dados
void MOSTRAR ( int RESISTOR [ ] )
{
    // definir dado local
    int X;
    // mostrar dados
    for ( X = 0; X < 10; X = X + 1 )
    { // mostrar cada valor de resistor
        cout << "\n" << X << " " << RESISTOR [ X ];
    } // fiim for
} // fim do procedimento MOSTRAR ( )

// funcao para contar os maiores de 10 ohms
int CONTAR ( int RESISTOR [ ] )
{
    // definir dados locais
    int X,
        CONTADOR = 0;
    // testar valores
    for ( X = 0; X < 10; X = X + 1 )
    { // testar um valor
        if ( RESISTOR [ X ] > 10 )
        {
            // somar se maior que 10 ohms
            CONTADOR=CONTADOR+1;
        } // fim if
    } // fim for
    return ( CONTADOR );
} // fim da funcao CONTAR ( )
```

```
// parte principal
//
int main (void)
{
// 1. definir dados
    int RESISTOR [ 10 ];      // tabela de resistores

// 2. ler dados
    LER ( RESISTOR );

// 3. mostrar o resultado
    // 3.1 mostrar todos os valores
        MOSTRAR ( RESISTOR );

    // 3.2 mostrar quantos sao maiores que 10 ohms
        cout << "\nMaiores que 10 = " << CONTAR ( RESISTOR );

// pausa para terminar
    cout << "\nPressionar ENTER para terminar.";
    getchar ( );
    return EXIT_SUCCESS;
} // fim do programa
```

Programa em C#:

```

/*
 * Exemplo 1a
 * Calcular quantos resistores, em uma tabela, sao maiores que 10 ohms.
 */
using System;

class Exemplo_1a
{
    //
    // parte principal
    //
    public static void Main ( )
    {
        // 1. definir dados
        int [ ] RESISTOR = new int [ 10 ];    // tabela de resistores
        int    X,                             // indice
            CONTADOR = 0;                     // para os maiores que 10 ohms
        // 2. ler dados
        for ( X = 0; X < 10; X = X+1 )
        {
            // 2.1. ler e guardar o valor de um resistor
            Console.Write ( "\nQual o valor do resistor " + X + " ? " );
            RESISTOR [ X ] = int.Parse ( Console.ReadLine ( ) );
        } // fim for
        // 3. testar valores
        for ( X = 0; X < 10; X = X+1 )
        {
            // 3.1. testar um valor
            if ( RESISTOR [ X ] > 10 )
            {
                // 3.1.1. somar se maior que 10 ohms
                CONTADOR = CONTADOR + 1;
            } // fim if
        } // fim for
        // 4. mostrar o resultado
        // 4.1. mostrar todos os valores
        for ( X = 0; X < 10; X = X+1 )
        {
            // 4.1.1. mostrar cada valor de resistor
            Console.WriteLine ( "\n" + X + " " + RESISTOR [ X ] );
        } // fim for
        // 4.2. mostrar quantos sao maiores que 10 ohms
        Console.WriteLine ( "\nMaiores que 10 = " + CONTADOR );
        // pausa para terminar
        Console.Write ( "\nPressionar ENTER para terminar." );
        Console.ReadLine ( );
    } // end Main ( )
} // fim Exemplo_1a class

```

Outra versão do programa em C#:

```

/*
 * Exemplo 1b
 * Calcular quantos resistores, em uma tabela, sao maiores que 10 ohms.
 */
using System;

class Exemplo_1b
{
    // procedimento para ler dados
    public static void LER ( int [ ] RESISTOR )
    {
        // definir dado local
        int X;           // indice
        // ler dados
        for ( X = 0; X < 10; X = X+1 )
        {
            // 2.1. ler e guardar o valor de um resistor
            Console.WriteLine ( "\nQual o valor do resistor " + X + " ? " );
            RESISTOR [ X ] = int.Parse ( Console.ReadLine ( ) );
        } // fim for
    } // fim do procedimento LER ( )
    // procedimento para mostrar dados
    public static void MOSTRAR ( int [ ] RESISTOR )
    {
        // definir dado local
        int X;           // indice
        // mostrar dados
        for ( X = 0; X < 10; X = X+1 )
        {
            // mostrar cada valor de resistor
            Console.WriteLine ( "\n" + X + " " + RESISTOR [ X ] );
        } // fim for
    } // fim do procedimento MOSTRAR ( )
    // funcao para contar os maiores que 10 ohms
    public static int CONTAR ( int [ ] RESISTOR )
    {
        // definir dados locais
        int X,           // indice
            CONTADOR = 0; // para os maiores que 10 ohms
        // testar valores
        for ( X = 0; X < 10; X = X+1 )
        {
            // testar um valor
            if ( RESISTOR [ X ] > 10 )
            {
                // 3.1.1. somar se maior que 10 ohms
                CONTADOR = CONTADOR + 1;
            } // fim if
        } // fim for
        return ( CONTADOR );
    } // fim da funcao CONTAR ( )
}

```

```
//  
// parte principal  
//  
public static void Main ( )  
{  
    // 1. definir dados  
    int [ ] RESISTOR = new int [ 10 ];    // tabela de resistores  
  
    // 2. ler dados  
    LER ( RESISTOR );  
  
    // 3. mostrar o resultado  
    // 3.1. mostrar todos os valores  
    MOSTRAR ( RESISTOR );  
    // 3.2. mostrar quantos sao maiores que 10 ohms  
    Console.WriteLine ( "\nMaiores que 10 = " + CONTAR ( RESISTOR ) );  
  
    // pausa para terminar  
    Console.Write ( "\nPressionar ENTER para terminar." );  
    Console.ReadLine ( );  
} // end Main ( )  
  
} // fim Exemplo_1b class
```

Programa em Java:

```

/**
 * Exemplo 1a
 * Calcular quantos resistores, em uma tabela, sao maiores que 10 ohms.
 */

// ----- classes necessarias

// ----- definicao de classe

class Exemplo_1a
{
//
// parte principal
//
    public static void main ( String [ ] args )
    {
        // 1. definir dados
        int [ ] RESISTOR = new int [ 10 ];    // tabela de resistores
        int    X,                             // indice
            CONTADOR = 0;                     // para os maiores que 10 ohms
        // 2. ler dados
        for ( X = 0; X < 10; X = X+1 )
        {
            // 2.1. ler e guardar o valor de um resistor
            System.out.print ( "\nQual o valor do resistor " + X + " ? " );
            RESISTOR [ X ] = Integer.parseInt ( System.console( ).readLine( ) );
        } // fim for
        // 3. testar valores
        for ( X = 0; X < 10; X = X+1 )
        {
            // 3.1. testar um valor
            if ( RESISTOR [ X ] > 10 )
            {
                // 3.1.1. somar se maior que 10 ohms
                CONTADOR = CONTADOR + 1;
            } // fim if
        } // fim for
        // 4. mostrar o resultado
        // 4.1. mostrar todos os valores
        System.out.println ( );
        for ( X = 0; X < 10; X = X+1 )
        {
            // 4.1.1. mostrar cada valor de resistor
            System.out.println ( "" + X + " " + RESISTOR [ X ] );
        } // fim for
        // 4.2. mostrar quantos sao maiores que 10 ohms
        System.out.println ( "\nMaiores que 10 = " + CONTADOR );
        // pausa para terminar
        System.out.print ( "\nPressionar ENTER para terminar." );
        System.console ( ).readLine ( );
    } // end main ( )
} // fim Exemplo_1a class

```

Outra versão do programa em Java:

```

/**
 * Exemplo 1b
 * Calcular quantos resistores, em uma tabela, sao maiores que 10 ohms.
 */

// ----- classes necessarias

// ----- definicao de classe

class Exemplo_1b
{
// procedimento para ler dados
public static void LER ( int [ ] RESISTOR )
{
// definir dado local
    int X;                // indice
// ler dados
    for ( X = 0; X < 10; X = X+1 )
    {
// 2.1. ler e guardar o valor de um resistor
        System.out.print ( "\nQual o valor do resistor " + X + " ? " );
        RESISTOR [ X ] = Integer.parseInt ( System.console( ).readLine( ) );
    } // fim for
} // fim do procedimento LER ( )

// procedimento para mostrar dados
public static void MOSTRAR ( int [ ] RESISTOR )
{
// definir dado local
    int X;                // indice
// mostrar dados
    System.out.println ( );
    for ( X = 0; X < 10; X = X+1 )
    {
// 4.1.1. mostrar cada valor de resistor
        System.out.println ( "" + X + " " + RESISTOR [ X ] );
    } // fim for
} // fim do procedimento MOSTRAR ( )

```



```

// funcao para contar os maiores que 10 ohms
public static int CONTAR ( int [ ] RESISTOR )
{
    // definir dados locais
    int X,                // indice
        CONTADOR = 0; // para os maiores que 10 ohms
    // testar valores
    for ( X = 0; X < 10; X = X+1 )
    {
        // testar um valor
        if ( RESISTOR [ X ] > 10 )
        {
            // 3.1.1. somar se maior que 10 ohms
            CONTADOR = CONTADOR + 1;
        } // fim if
    } // fim for
    return ( CONTADOR );
} // fim da funcao CONTAR ( )

//
// parte principal
//
public static void main ( String [ ] args )
{
    // 1. definir dados
    int [ ] RESISTOR = new int [ 10 ];    // tabela de resistores

    // 2. ler dados
    LER ( RESISTOR );

    // 3. mostrar o resultado
    // 3.1. mostrar todos os valores
    MOSTRAR ( RESISTOR );
    // 3.2. mostrar quantos sao maiores que 10 ohms
    System.out.println ( "\nMaiores que 10 = " + CONTAR ( RESISTOR ) );

    // pausa para terminar
    System.out.print ( "\nPressionar ENTER para terminar." );
    System.console( ).readLine( );
} // end main ( )

} // fim Exemplo_1b class

```

Programa em Python:

```
# Exemplo 1.
# Calcular quantos resistores, em uma tabela, sao maiores que 10 ohms.
#
from array import *;
# 1. definir dados
# tabela de resistores
RESISTOR = array ( 'i', [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ] );
X = 0; # indice
CONTADOR = 0; # para os maiores que 10 ohms
#
# 2. ler dados
for X in range ( 0, 10, 1 ):
# 2.1 ler e guardar o valor de um resistor
    print ( "\nQual o valor do resistor ", (X+1), " ? ", end=" " );
    RESISTOR [ X ] = int ( input ( ) );
# fim repetir

# 3. testar valores
for X in range ( 0, 10, 1 ):
# 3.1 testar um valor
    if ( RESISTOR [ X ] > 10.0 ):
# 3.1.1 somar se maior que 10 ohms
        CONTADOR = CONTADOR + 1;
    # fim se
# fim repetir

# 4. mostrar o resultado
print ( );
# 4.1 mostrar todos os valores
for X in range ( 0, 10, 1 ):
# 4.1.1 mostrar cada valor de resistor
    print ( (X+1), "\t", RESISTOR [ X ] );
# fim repetir

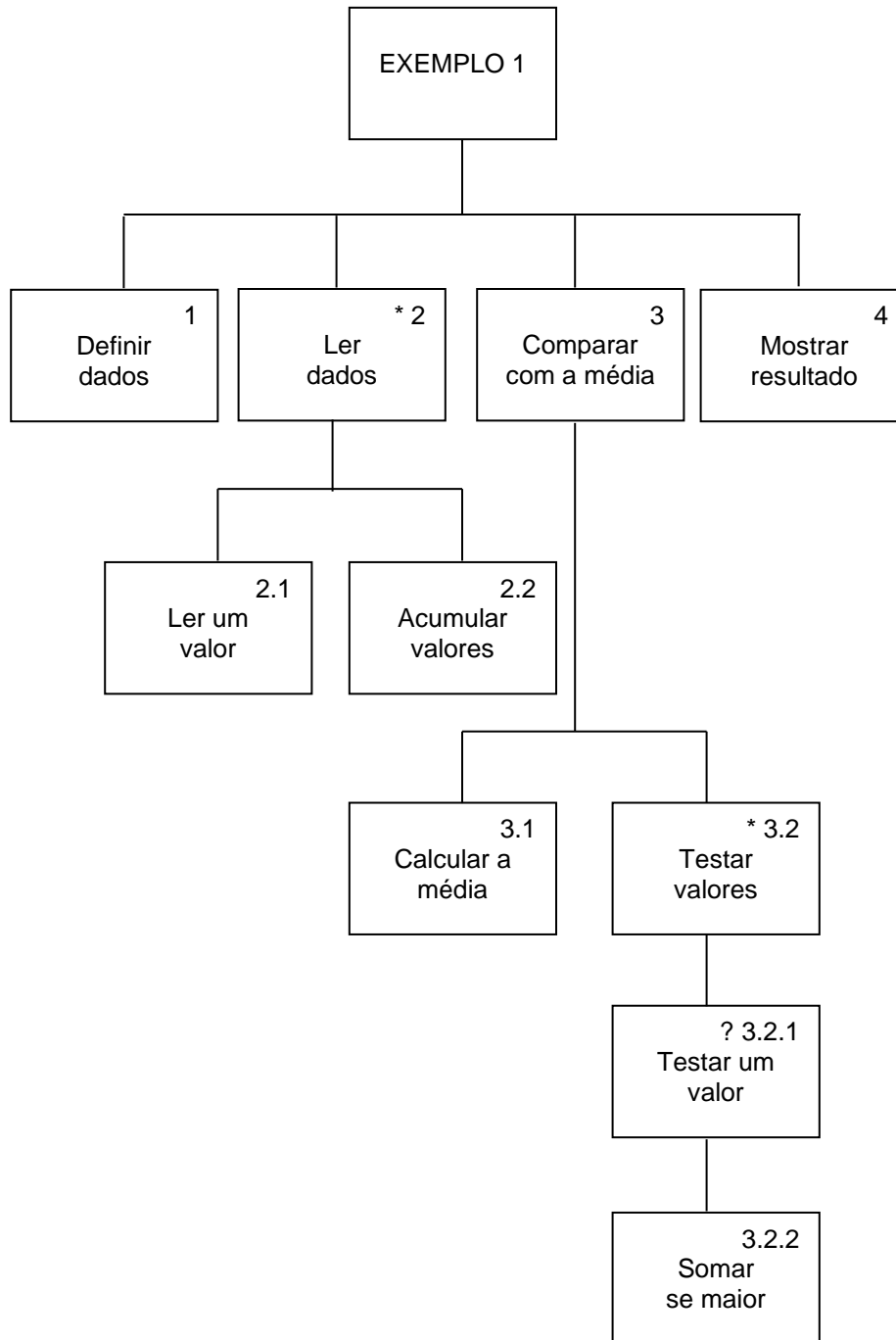
# 4.2 mostrar quantos sao maiores que 10 ohms
print ( "\n\nMaiores que 10 = ", CONTADOR );

# pausa para terminar
print ( "\n\nPressionar ENTER para terminar." );
input ( );
# fim do programa
```

Exemplo 2.

Montar uma tabela de, no máximo 10, resistores. Os valores serão lidos do teclado, e deseja-se verificar quantos valores são superiores à média.

Diagrama funcional:



Análise de dados :

- Dados do problema :

Dado	Tipo	Valor inicial	Função
RESISTOR	inteiro (10)	-	armazenar os valores de 10 resistores
I	inteiro	-	índice
SOMA	inteiro	0	somatório dos valores
MÉDIA	real	-	média dos valores

- Resultados do problema :

Dado	Tipo	Valor inicial	Função
CONTADOR	inteiro	0	contador de quantos são maiores que a média

- Avaliação da solução :

Índice	Dado	Resultado
0	10	
1	20	
2	30	√
3	50	√
4	10	
5	10	
6	15	
7	50	√
8	25	
9	30	√
	média	25
	maiores	4

Algoritmo:

Esboço:

Primeira versão, só comentários.

Exemplo 2	v.1
Ação	Bloco
! definir dados	1
! ler dados	2
! ler um valor	2.1
! acumular valores	2.2
! comparar com a média	3
! calcular a média	3.1
! testar valores	3.2
! testar um valor	3.2.1
! somar se maior que a média	3.2.2
! mostrar o resultado	4

Segunda versão, com refinamento do primeiro bloco.

Exemplo 2	v.2
Ação	Bloco
! definir dados	1
inteiro RESISTOR[10], ! tabela de resistores X, ! índice SOMA = 0, ! soma dos valores CONTADOR=0; ! para os maiores que 10 ohms float MEDIA; ! para a media	
! ler dados	2
! ler um valor	2.1
! acumular valores	2.2
! comparar com a média	3
! calcular a média	3.1
! testar valores	3.2
! testar um valor	3.2.1
! somar se maior que a média	3.2.2
! mostrar o resultado	4

Terceira versão, com refinamento do segundo bloco.

Exemplo 2	v.3
Ação	Bloco
! definir dados	1
inteiro RESISTOR[10], ! tabela de resistores X, ! índice SOMA = 0, ! soma dos valores CONTADOR=0; ! para os maiores que 10 ohms float MEDIA; ! para a media	
! ler dados	2
X = 1 : 10 : 1	
! ler e guardar o valor de um resistor tela ← ("Qual o valor do resistor ", X , " ? "); RESISTOR [ X ] ← teclado;	2.1
! acumular valores SOMA = SOMA + RESISTOR [ X ];	2.2
! comparar com a média	3
! calcular a média	3.1
! testar valores	3.2
! testar um valor	3.2.1
! somar se maior que a média	3.2.2
! mostrar o resultado	4

Quarta versão, com refinamento do terceiro bloco.

Exemplo 2	v.4
Ação	Bloco
! definir dados	1
inteiro RESISTOR[10], ! tabela de resistores X, ! índice SOMA = 0, ! soma dos valores CONTADOR=0; ! para os maiores que 10 ohms float MEDIA; ! para a media	
! ler dados	2
X = 1 : 10 : 1	
! ler e guardar o valor de um resistor tela ← ("Qual o valor do resistor ", X , " ? "); RESISTOR [ X ] ← teclado;	2.1
! acumular valores SOMA = SOMA + RESISTOR [ X ];	2.2
! comparar com a média	3
! calcular a média MÉDIA = SOMA / 10;	3.1
! testar valores	3.2
X = 1 : 10 : 1	
! testar um valor	3.2.1
RESISTOR [ X ] > MÉDIA ?	3.2.1
! somar se maior que a média CONTADOR=CONTADOR+1;	
// mostrar o resultado	4

Quinta versão, com refinamento do terceiro bloco.

Exemplo 2	v.5
Ação	Bloco
! definir dados	1
inteiro RESISTOR[10], ! tabela de resistores X, ! índice SOMA = 0, ! soma dos valores CONTADOR=0; ! para os maiores que 10 ohms float MÉDIA; ! para a media	
! ler dados	2
X = 1 : 10 : 1	
! ler e guardar o valor de um resistor tela ← ("Qual o valor do resistor ", X , " ? "); RESISTOR [ X ] ← teclado;	2.1
! acumular valores SOMA = SOMA + RESISTOR [ X ];	2.2
! comparar com a média	3
! calcular a média MÉDIA = SOMA / 10;	3.1
! testar valores	3.2
X = 1 : 10 : 1	
! testar um valor	3.2.1
RESISTOR [X] V ! somar se maior que a média > MÉDIA ? CONTADOR=CONTADOR+1;	
! mostrar o resultado	4
! mostrar todos os valores repetir para ( X = 1 : 10 : 1 ) ! ler e guardar o valor de um resistor tela ← ( "Qual o valor do resistor ", X , " ? ", RESISTOR [ X ] ); fim repetir	4.1
! mostrar quantos são maiores que a média tela ← ("Maiores que a media = ", CONTADOR);	4.2

Programa em SCILAB:

```
// Exemplo 2.
// Calcular quantos resistores, em uma tabela, sao maiores que a media.
//
// 1. definir dados
RESISTOR = zeros(10); // tabela de resistores
X = 0; // indice
SOMA = 0; // soma dos resistores
CONTADOR = 0; // para os maiores que 10 ohms
MEDIA = 0.0; // para a media
//
// 2. ler dados
for ( X = 1 : 1 : 10 )
    // 2.1 ler e guardar o valor de um resistor
    printf ( "\nQual o valor do resistor %d ? ", X );
    RESISTOR ( X ) = input ( " " );
    // 2.2 acumular valores
    SOMA = SOMA + RESISTOR ( X );
end // fim repetir

// 3. comparar com a media
// 3.1 calcular a media
MEDIA = SOMA / 10;
// 3.2 testar valores
for ( X = 1 : 1 : 10 )
    // 3.2.1 testar um valor
    if ( RESISTOR ( X ) > MEDIA )
        // somar se maior que a media
        CONTADOR=CONTADOR+1;
    end // fim if
end // fim for

// 4. mostrar o resultado
// 4.1 mostrar todos os valores
for ( X = 1 : 1 : 10 )
    // 4.1.1 mostrar cada valor de resistor
    printf ( "\n %f \t %f ", X, RESISTOR ( X ) );
end // fiim for
// 4.2 mostrar quantos sao maiores que a media
printf ( "\nMaiores que a media = %d ", CONTADOR);

// pausa para terminar
printf ( "\nPressionar ENTER para terminar." );
halt;
// fim do programa
```



Programa em C:

```
// Exemplo 2a.
// Calcular quantos resistores, em uma tabela, são maiores que a media.
//
// bibliotecas necessarias
#include <stdio.h>
#include <stdlib.h>
//
// parte principal
//
int main (void)
{
// 1. definir dados
int RESISTOR [ 10 ], // tabela de resistores
    X = 0,           // indice
    SOMA = 0,        // soma dos resistores
    CONTADOR = 0;    // para os maiores que 10 ohms
float MEDIA;        // para a media
// 2. ler dados
for ( X = 0; X < 10; X = X + 1 )
{ // 2.1 ler e guardar o valor de um resistor
    printf ( "\nQual o valor do resistor %d %s", X, " ? " );
    scanf ( "%d", &RESISTOR [ X ] );
    // 2.2 acumular valores
    SOMA = SOMA + RESISTOR [ X ];
} // fim for

// 3. comparar com a media
// 3.1 calcular a media
MEDIA = SOMA / 10;
// 3.2 testar valores
for ( X = 0; X < 10; X = X + 1 )
{ // 3.2.1 testar um valor
    if ( RESISTOR [ X ] > MEDIA )
    {
        // somar se maior que a media
        CONTADOR=CONTADOR+1;
    } // fim if
} // fim for

// 4. mostrar o resultado
// 4.1 mostrar todos os valores
for ( X = 0; X < 10; X = X + 1 )
{ // 4.1.1 mostrar cada valor de resistor
    printf ( "\n%d%s%d", X, " ", RESISTOR [ X ] );
} // fim for

// 4.2 mostrar quantos são maiores que a media
printf ( "\nMaiores que a media = %d", CONTADOR );
// pausa para terminar
printf ( "\nPressionar ENTER para terminar." );
getchar ( );
return ( 0 );
} // fim do programa
```

Programa em C++:

```
// Exemplo 2a.
// Calcular quantos resistores, em uma tabela, são maiores que a media.
//
// bibliotecas necessarias
#include <iostream>
using namespace std;
//
// parte principal
//
int main (void)
{
// 1. definir dados
int RESISTOR [ 10 ], // tabela de resistores
    X, // indice
    SOMA = 0, // soma dos resistores
    CONTADOR = 0; // para os maiores que 10 ohms
double MEDIA; // para a media
// 2. ler dados
for ( X = 0; X < 10; X = X + 1 )
{ // 2.1 ler e guardar o valor de um resistor
    cout << "\nQual o valor do resistor " << X << " ? ";
    cin >> RESISTOR [ X ];
    // 2.2 acumular valores
    SOMA = SOMA + RESISTOR [ X ];
} // fim for

// 3. comparar com a media
// 3.1 calcular a media
MEDIA = SOMA / 10;
// 3.2 testar valores
for ( X = 0; X < 10; X = X + 1 )
{ // 3.2.1 testar um valor
    if ( RESISTOR [ X ] > MEDIA )
    {
        // somar se maior que a media
        CONTADOR=CONTADOR+1;
    } // fim if
} // fim for

// 4. mostrar o resultado
// 4.1 mostrar todos os valores
for ( X = 0; X < 10; X = X + 1 )
{ // 4.1.1 mostrar cada valor de resistor
    cout << "\n" << X << " " << RESISTOR [ X ];
} // fim for

// 4.2 mostrar quantos são maiores que a media
cout << "\nMaiores que a media = " << CONTADOR;
// pausa para terminar
cout << "\nPressionar ENTER para terminar.";
cin.get ( );
return EXIT_SUCCESS;
} // fim do programa
```

Outra versão com procedimentos e funções.

Programa em C:

```
// Exemplo 2b.
// Calcular quantos resistores, em uma tabela, são maiores que a media.
//
// bibliotecas necessarias
#include <stdio.h>
#include <stdlib.h>
//

// procedimento para ler dados
void LER ( int RESISTOR [ ], int N )
{
    // definir dado local
    int X;
    // ler dados
    for ( X = 0; X < N; X = X + 1 )
    { // ler e guardar o valor de um resistor
        cout << "\nQual o valor do resistor " << X << " ? ";
        cin  >> RESISTOR [ X ];
    } // fim for
} // fim do procedimento LER ( )

// procedimento para mostrar dados
void MOSTRAR ( int RESISTOR [ ], int N )
{
    // definir dado local
    int X;
    // mostrar dados
    for ( X = 0; X < N; X = X + 1 )
    { // mostrar cada valor de resistor
        printf ( "\n%d%s%d", X, " ", RESISTOR [ X ] );
    } // fiim for
} // fim do procedimento MOSTRAR ( )

// funcao para contar os maiores que um valor de referencia
int CONTAR ( int RESISTOR [ ], int N, float REFERENCIA )
{
    // definir dados locais
    int X,
        CONTADOR = 0;
    // testar valores
    for ( X = 0; X < N; X = X + 1 )
    { // testar um valor
        if ( RESISTOR [ X ] > REFERENCIA )
        {
            // somar se maior que a referencia
            CONTADOR=CONTADOR+1;
        } // fim if
    } // fim for
    return ( CONTADOR );
} // fim da funcao CONTAR ( )
```

```

// funcao para calcular a media
float MEDIA ( int RESISTOR [ ], int N )
{
// definir dados locais
    int X,
        SOMA = 0;
    float RESULTADO = 0.0;
// acumular valores
    for ( X = 0; X < N; X = X + 1 )
    { // acumular um valor
        SOMA = SOMA + RESISTOR [ X ];
    } // fim for
// calcular a media
    if ( N > 0 )
    {
        RESULTADO = SOMA / N;
    } // fim if
    return ( RESULTADO );
} // fim da funcao MEDIA ( )

// parte principal
//
int main (void)
{
// 1. definir dados
    int RESISTOR [ 10 ];    // tabela de resistores
    float MEDIO;            // valor médio dos resistores

// 2. ler dados
    LER ( RESISTOR, 10 );

// 3. mostrar o resultado
// 3.1 mostrar todos os valores
    MOSTRAR ( RESISTOR, 10 );

// 3.2 mostrar a media
    MEDIO = MEDIA (RESISTOR, 10);
    printf ( "\nMedia = %f", MEDIO );

// 3.3 mostrar quantos sao maiores que a media
    printf ( "\nMaiores que a media = %d", CONTAR ( RESISTOR, 10, MEDIO ) );

// pausa para terminar
    printf ( "\nPressionar ENTER para terminar." );
    getchar ( );
    return ( 0 );
} // fim do programa

```

Programa em C++:

```
// Exemplo 2b.
// Calcular quantos resistores, em uma tabela, são maiores que a media.
//
// bibliotecas necessarias
#include <iostream>
using namespace std;
//

// procedimento para ler dados
void LER ( int RESISTOR [ ], int N )
{
    // definir dado local
    int X;
    // ler dados
    for ( X = 0; X < N; X = X + 1 )
    { // ler e guardar o valor de um resistor
        cout << "\nQual o valor do resistor " << X << " ? ";
        cin >> RESISTOR [ X ];
    } // fim for
} // fim do procedimento LER ( )

// procedimento para mostrar dados
void MOSTRAR ( int RESISTOR [ ], int N )
{
    // definir dado local
    int X;
    // mostrar dados
    for ( X = 0; X < N; X = X + 1 )
    { // mostrar cada valor de resistor
        cout << "\n" << X << " " << RESISTOR [ X ];
    } // fim for
} // fim do procedimento MOSTRAR ( )

// funcao para contar os maiores que um valor de referencia
int CONTAR ( int RESISTOR [ ], int N, float REFERENCIA )
{
    // definir dados locais
    int X,
        CONTADOR = 0;
    // testar valores
    for ( X = 0; X < N; X = X + 1 )
    { // testar um valor
        if ( RESISTOR [ X ] > REFERENCIA )
        {
            // somar se maior que a referencia
            CONTADOR=CONTADOR+1;
        } // fim if
    } // fim for
    return ( CONTADOR );
} // fim da funcao CONTAR ( )
```

```

// funcao para calcular a media
float MEDIA ( int RESISTOR [ ], int N )
{
// definir dados locais
    int X,
        SOMA = 0;
    float RESULTADO = 0.0;
// acumular valores
    for ( X = 0; X < N; X = X + 1 )
    { // acumular um valor
        SOMA = SOMA + RESISTOR [ X ];
    } // fim for
// calcular a media
    if ( N > 0 )
    {
        RESULTADO = SOMA / N;
    } // fim if
    return ( RESULTADO );
} // fim da funcao MEDIA ( )

// parte principal
//
int main (void)
{
// 1. definir dados
    int RESISTOR [ 10 ];    // tabela de resistores
    float MEDIO;            // valor médio dos resistores

// 2. ler dados
    LER ( RESISTOR, 10 );

// 3. mostrar o resultado
// 3.1 mostrar todos os valores
    MOSTRAR ( RESISTOR, 10 );

// 3.2 mostrar a media
    MEDIO = MEDIA (RESISTOR, 10);
    cout << "\nMedia = " << MEDIO;

// 3.3 mostrar quantos sao maiores que a media
    cout << "\nMaiores que a media = " << CONTAR ( RESISTOR, 10, MEDIO );

// pausa para terminar
    cout << "\nPressionar ENTER para terminar.";
    getchar ( );
    return EXIT_SUCCESS;
} // fim do programa

```

Programa em C#:

```

/*
 * Exemplo 2a
 * Calcular quantos resistores, em uma tabela, sao maiores que a media.
 */
using System;

class Exemplo_2a
{
    //
    // parte principal
    //
    public static void Main ( )
    {
        // 1. definir dados
        int [ ] RESISTOR = new int [ 10 ]; // tabela de resistores
        int     X = 0,                      // indice
              SOMA = 0,                    // soma dos resistores
              CONTADOR = 0;                // para os maiores que 10 ohms
        double MEDIA = 0.0;                // para a media
        // 2. ler dados
        for ( X = 0; X < 10; X = X+1 )
        {
            // 2.1. ler e guardar o valor de um resistor
            Console.Write ("Qual o valor do resistor " + X + " ? ");
            RESISTOR [ X ] = int.Parse ( Console.ReadLine ( ) );
            // 2.2. acumular valores
            SOMA = SOMA + RESISTOR [ X ];
        } // fim for
        // 3. comparar com a media
        // 3.1. calcular a media
        MEDIA = SOMA / 10.0;
        // 3.2. testar valores
        for ( X = 0; X < 10; X = X+1 )
        { // 3.2.1. testar um valor
            if ( RESISTOR [ X ] > MEDIA )
            { // 3.1.1. somar se maior que a media
                CONTADOR = CONTADOR + 1;
            } // fim if
        } // fim for
        // 4. mostrar o resultado
        // 4.1. mostrar todos os valores
        for ( X = 0; X < 10; X = X+1 )
        { // 4.1.1. mostrar cada valor de resistor
            Console.WriteLine ( "X = " + X + " RESISTOR [ X ] = " + RESISTOR [ X ] );
        } // fim for
        // 4.2. mostrar quantos sao maiores que 10 ohms
        Console.WriteLine ( "Maiores que a media = " + CONTADOR );
        // pausa para terminar
        Console.Write ( "Pressionar ENTER para terminar." );
        Console.ReadLine ( );
    } // end Main ( )
} // fim Exemplo_2a class

```

Outra versão do programa em C#:

```

/*
 * Exemplo 2b
 * Calcular quantos resistores, em uma tabela, sao maiores que a media.
 */
using System;

class Exemplo_2b
{
    // procedimento para ler dados
    public static void LER ( int [ ] RESISTOR, int N )
    {
        // definir dado local
        int X;           // indice
        // ler dados
        for ( X = 0; X < N; X = X+1 )
        {
            // 2.1. ler e guardar o valor de um resistor
            Console.WriteLine ( "\nQual o valor do resistor " + X + " ? " );
            RESISTOR [ X ] = int.Parse ( Console.ReadLine ( ) );
        } // fim for
    } // fim do procedimento LER ( )

    // procedimento para mostrar dados
    public static void MOSTRAR ( int [ ] RESISTOR, int N )
    {
        // definir dado local
        int X;           // indice
        // mostrar dados
        for ( X = 0; X < N; X = X+1 )
        { // mostrar cada valor de resistor
            Console.WriteLine ( "\n" + X + " " + RESISTOR [ X ] );
        } // fim for
    } // fim do procedimento MOSTRAR ( )

    // funcao para contar os maiores que 10 ohms
    public static int CONTAR ( int [ ] RESISTOR, int N, double REFERENCIA )
    {
        // definir dados locais
        int X,           // indice
            CONTADOR = 0; // para os maiores que 10 ohms
        // testar valores
        for ( X = 0; X < N; X = X+1 )
        { // testar um valor
            if ( RESISTOR [ X ] > REFERENCIA )
            { // 3.1.1. somar se maior que 10 ohms
                CONTADOR = CONTADOR + 1;
            } // fim if
        } // fim for
        return ( CONTADOR );
    } // fim da funcao CONTAR ( )
}

```



```

// funcao para calcular a media
public static double MEDIA ( int [ ] RESISTOR, int N )
{
    // definir dados locais
    int    X = 0,                // indice
          SOMA = 0;
    double RESULTADO = 0.0; // para a media
    // acumular valores
    for ( X = 0; X < N; X = X+1 )
    {
        // acumular um valor
        SOMA = SOMA + RESISTOR [ X ];
    } // fim for
    // calcular a media
    if ( N > 0 )
    {
        RESULTADO = SOMA / N;
    } // fim if
    return ( RESULTADO );
} // fim da funcao MEDIA ( )

//
// parte principal
//
public static void Main ( )
{
    // 1. definir dados
    int [ ] RESISTOR = new int [ 10 ];    // tabela de resistores
    double MEDIO;                        // valor medio dos resistores

    // 2. ler dados
    LER ( RESISTOR, 10 );

    // 3. mostrar o resultado
    // 3.1. mostrar todos os valores
    MOSTRAR ( RESISTOR, 10 );
    // 3.2. mostrar a media
    MEDIO = MEDIA ( RESISTOR , 10 );
    Console.WriteLine ( "\nMedia = " + MEDIO );
    // 3.3. mostrar quantos sao maiores que a media
    Console.WriteLine ( "\nMaiores que a media = " +
        CONTAR ( RESISTOR, 10, MEDIO ) );

    // pausa para terminar
    Console.Write ( "\nPressionar ENTER para terminar." );
    Console.ReadLine ( );
} // end Main ( )

} // fim Exemplo_2b class

```

Programa em Java:

```

/**
 * Exemplo 2a
 * Calcular quantos resistores, em uma tabela, sao maiores que a media.
 */

// ----- classes necessarias

// ----- definicao de classe

class Exemplo_2a
{
//
// parte principal
//
    public static void main ( String [ ] args )
    {
        // 1. definir dados
        int [ ] RESISTOR = new int [ 10 ];    // tabela de resistores
        int    X          = 0,                // indice
              SOMA         = 0,                // soma dos resistores
              CONTADOR     = 0;                // para os maiores que 10 ohms
        double MEDIA      = 0.0;              // para a media

        // 2. ler dados
        for ( X = 0; X < 10; X = X+1 )
        {
            // 2.1. ler e guardar o valor de um resistor
            System.out.print ( "\nQual o valor do resistor " + X + " ? " );
            RESISTOR [ X ] = Integer.parseInt ( System.console( ).readLine( ) );
            // 2.2. acumular valores
            SOMA = SOMA + RESISTOR [ X ];
        } // fim for

        // 3. comparar com a media
        // 3.1. calcular a media
        MEDIA = SOMA / 10.0;
        // 3.2. testar valores
        for ( X = 0; X < 10; X = X+1 )
        { // 3.2.1. testar um valor
            if ( RESISTOR [ X ] > MEDIA )
            { // 3.1.1. somar se maior que a media
                CONTADOR = CONTADOR + 1;
            } // fim if
        } // fim for
    }
}

```

```
// 4. mostrar o resultado
// 4.1. mostrar todos os valores
for ( X = 0; X < 10; X = X+1 )
{ // 4.1.1. mostrar cada valor de resistor
  System.out.println ( "" + X + " " + RESISTOR [ X ] );
} // fim for
// 4.2. mostrar quantos sao maiores que 10 ohms
System.out.println ( "\nMaiores que a media = " + CONTADOR );

// pausa para terminar
System.out.print ( "\nPressionar ENTER para terminar." );
System.console( ).readLine( );
} // end main ( )

} // fim Exemplo_2a class
```

Outra versão do programa em Java:

```

/**
 * Exemplo 2b
 * Calcular quantos resistores, em uma tabela, sao maiores que a media.
 */

// ----- classes necessarias

// ----- definicao de classe

class Exemplo_2b
{
// procedimento para ler dados
public static void LER ( int [ ] RESISTOR, int N )
{
// definir dado local
    int X;           // indice
// ler dados
    for ( X = 0; X < N; X = X+1 )
    {
// 2.1. ler e guardar o valor de um resistor
        System.out.print ( "\nQual o valor do resistor " + X + " ? " );
        RESISTOR [ X ] = Integer.parseInt ( System.console( ).readLine( ) );
    } // fim for
} // fim do procedimento LER ( )

// procedimento para mostrar dados
public static void MOSTRAR ( int [ ] RESISTOR, int N )
{
// definir dado local
    int X;           // indice
// mostrar dados
    for ( X = 0; X < N; X = X+1 )
    { // mostrar cada valor de resistor
        System.out.println ( "" + X + " " + RESISTOR [ X ] );
    } // fim for
} // fim do procedimento MOSTRAR ( )

// funcao para contar os maiores que 10 ohms
public static int CONTAR ( int [ ] RESISTOR, int N, double REFERENCIA )
{
// definir dados locais
    int X,           // indice
        CONTADOR = 0; // para os maiores que 10 ohms
// testar valores
    for ( X = 0; X < N; X = X+1 )
    { // testar um valor
        if ( RESISTOR [ X ] > REFERENCIA )
        { // 3.1.1. somar se maior que 10 ohms
            CONTADOR = CONTADOR + 1;
        } // fim if
    } // fim for
    return ( CONTADOR );
} // fim da funcao CONTAR ( )

```

```

// funcao para calcular a media
public static double MEDIA ( int [ ] RESISTOR, int N )
{
    // definir dados locais
    int X, // indice
        SOMA = 0;
    double RESULTADO = 0.0; // para a media
    // acumular valores
    for ( X = 0; X < N; X = X+1 )
    {
        // acumular um valor
        SOMA = SOMA + RESISTOR [ X ];
    } // fim for
    // calcular a media
    if ( N > 0 )
    {
        RESULTADO = SOMA / N;
    } // fim if
    return ( RESULTADO );
} // fim da funcao MEDIA ( )

//
// parte principal
//
public static void main ( String [ ] args )
{
    // 1. definir dados
    int [ ] RESISTOR = new int [ 10 ]; // tabela de resistores
    double MEDIO; // valor medio dos resistores

    // 2. ler dados
    LER ( RESISTOR, 10 );

    // 3. mostrar o resultado
    // 3.1. mostrar todos os valores
    MOSTRAR ( RESISTOR, 10 );
    // 3.2. mostrar a media
    MEDIO = MEDIA ( RESISTOR , 10 );
    System.out.println ( "\nMedia = " + MEDIO );
    // 3.3. mostrar quantos sao maiores que a media
    System.out.println ( "\nMaiores que a media = " + CONTAR ( RESISTOR, 10, MEDIO ) );

    // pausa para terminar
    System.out.print ( "\nPressionar ENTER para terminar." );
    System.console ( ).readLine ( );
} // end main ( )

} // fim Exemplo_2b class

```

Programa em Python:

```
# Exemplo 2.
# Calcular quantos resistores, em uma tabela, sao maiores que a media.
#
from array import *;
# 1. definir dados
# tabela de resistores
RESISTOR = array ( 'i', [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ] );
X = 0; # indice
SOMA = 0; # soma dos resistores
CONTADOR = 0; # para os maiores que 10 ohms
MEDIA = 0.0; # para a media
#
# 2. ler dados
for X in range ( 0, 10, 1 ):
# 2.1 ler e guardar o valor de um resistor
    print ( "\nQual o valor do resistor ", (X+1), " ? ", end=" " );
    RESISTOR [ X ] = int ( input ( " " ) );
# 2.2 acumular valores
    SOMA = SOMA + RESISTOR [ X ];
# fim repetir

# 3. comparar com a media
# 3.1 calcular a media
MEDIA = SOMA / 10.0;
# 3.2 testar valores
for X in range ( 0, 10, 1 ):
# 3.2.1 testar um valor
    if ( RESISTOR [ X ] > MEDIA ):
        # somar se maior que a media
        CONTADOR = CONTADOR + 1;
    # fim se
# fim repetir

# 4. mostrar o resultado
print ( );
# 4.1 mostrar todos os valores
for X in range ( 0, 10, 1 ):
# 4.1.1 mostrar cada valor de resistor
    print ( (X+1), "\t", RESISTOR [ X ] );
# fiim repetir
# 4.2 mostrar quantos sao maiores que a media
print ( "\nMaiores que a media = ", CONTADOR );

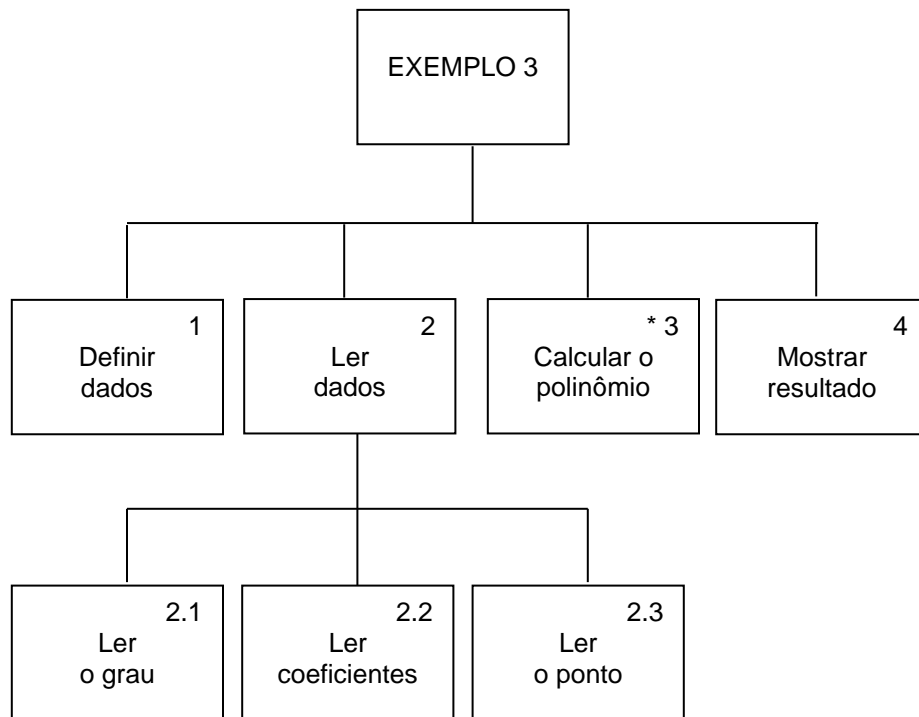
# pausa para terminar
print ( "\nPressionar ENTER para terminar." );
input ( );
# fim do programa
```

## Exemplo 3.

Montar uma tabela de, no máximo 10, valores reais, coeficientes de um polinômio. Os valores de N e dos coeficientes serão lidos do teclado, e deseja-se calcular o valor do polinômio no ponto X, que também será lido do teclado:

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + a_{n-2} x^{n-2} + \cdots + a_1 x^1 + a_0$$

Diagrama funcional:



Análise de dados :

- Dados do problema :

Dado	Tipo	Valor inicial	Função
A	real (10)	-	armazenar até 10 coeficientes
N	inteiro	-	grau do polinômio
PX	real	-	valor do ponto
X	inteiro	-	índice

- Resultados do problema :

Dado	Tipo	Valor inicial	Função
P	real	0	valor do polinômio

- Avaliação da solução :

Índice	Dado	Resultado
0	1	
1	-2	
2	1	
X	2	1

Algoritmo:

Esboço:

Primeira versão, só comentários.

Exemplo 3	v.1
Ação	Bloco
! definir dados	1
! ler dados	2
! ler o grau do polinômio (N)	2.1
! ler os coeficientes do polinômio (A's)	2.2
! ler o valor do ponto (PX)	2.3
! calcular o polinômio	3
! mostrar o resultado	4



Segunda versão, com refinamento do primeiro bloco.

Exemplo 3	v.2
Ação	Bloco
! definir dados	1
inteiro N,       ! grau do polinômio X;               ! índice real A [ 10 ]; ! tabela de coeficientes real PX,       ! valor do ponto P = 0.0; ! valor do polinômio	
! ler dados	2
! ler o grau do polinômio (N)	2.1
! ler os coeficientes do polinômio (A's)	2.2
! ler o valor do ponto (X)	2.3
! calcular o polinômio	3
! mostrar o resultado	4

Terceira versão, com refinamento do segundo e do quarto bloco.

Exemplo 3	v.3
Ação	Bloco
! definir dados	1
inteiro N,       ! grau do polinômio X;               ! índice real A [ 10 ]; ! tabela de coeficientes real PX,       ! valor do ponto P = 0.0; ! valor do polinômio	
! ler dados	2
! ler o grau do polinômio (N)	2.1
tela ← "Qual o grau do polinômio (N) ? "; N ← teclado;	
! ler os coeficientes do polinômio (A's)	2.2
X = 1 : (N+1) : 1	2.2.1
! ler um coeficiente	
tela ← ( "A[", X , "]= " ); A [ X ] ← teclado;	2.2.2
! ler o valor do ponto (X)	2.3
tela ← "Em que ponto (X) ? "; PX ← teclado;	
! calcular o polinômio	3
! mostrar o resultado	4
tela ← ( "P(", PX , ")= ", P );	

Quarta versão, com refinamento do terceiro bloco.

Exemplo 3	v.4
Ação	Bloco
! definir dados	1
inteiro N,       ! grau do polinômio X;       ! índice real   A [ 10 ]; ! tabela de coeficientes real   PX,       ! valor do ponto P = 0.0; ! valor do polinômio	
! ler dados	2
! ler o grau do polinômio (N)	2.1
tela ← “\nQual o grau do polinômio (N) ? “; N ← teclado;	
! ler os coeficientes do polinômio (A's)	2.2
X = 1 : (N+1) : 1	2.2.1
! ler um coeficiente	2.2.2
tela ← ( “\nA[ , X , “]= “ ); A [ X ] ← teclado;	
! ler o valor do ponto (X)	2.3
tela ← “\nEm que ponto (X) ? “; PX ← teclado;	
! calcular o polinômio	3
P = A [ 1 ]; ! valor inicial	3.1
X = 2 : (N+1) : 1	3.2
! somar um termo do polinômio	3.2.1
P = P + A [ X ] * ( PX ^ (X-1) );	
! mostrar o resultado	4
tela ← ( “\nP( , X , “)= “ , P );	

Programa em SCILAB:

```
// Exemplo 3.
// Calcular o valor do polinomio em um determinado ponto.
//
// 1. definir dados
N = 0;           // grau do polinomio
X = 0;           // indice
A ( 1:10 ) = 0;  // tabela de coeficientes
PX = 0;          // valor do ponto
P = 0.0;         // valor do polinomio

// 2. ler dados
// ler o grau do polinomio (N)
N = input ( "\nQual o grau do polinomio ? " );
// ler os coeficientes do polinomio (A's)
for ( X = 1 : 1 : (N+1) )
    // ler um coeficiente
    printf ( "\nA [ %d ] = ", X );
    A ( X ) = input ( " " );
end // for
// ler o valor do ponto (X)
PX = input ( "\nEm que ponto ? " );

// 3. calcular o polinomio
P = A ( 1 );     // valor inicial
for ( X = 2 : 1 : (N+1) )
    // somar um termo do polinomio
    P = P + A ( X ) * PX ^ ( X-1 );
end // for

// 4. mostrar o resultado
printf ( "\nP( %f )= %f ", PX, P );

// pausa para terminar
printf ( "\nPressionar ENTER para terminar." );
halt;
// fim do programa
```

Programa em C:

```
// Exemplo 3a.
// Calcular o valor do polinomio em um determinado ponto.
//
// bibliotecas necessarias
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
//
// parte principal
//
int main (void)
{
// 1. definir dados
    int    N,           // grau do polinomio
           X;           // indice
    float A [ 10 ];     // tabela de coeficientes
    float PX,           // valor do ponto
           P = 0.0;     // valor do polinomio

// 2. ler dados
    // ler o grau do polinomio (N)
    printf ( "\nQual o grau do polinomio (N) ? " );
    scanf ( "%d", &N );
    // ler os coeficientes do polinomio (A's)
    for ( X = 0; X <= N; X = X + 1 )
    {
        // ler um coeficiente
        printf ( "\nA[%d] = %d", X;
        scanf ( "%f", &A [ X ] );
    } // fim for
    // ler o valor do ponto (X)
    printf ( "\nEm que ponto (X) ? " );
    scanf ( "%f", &PX );

// 3. calcular o polinomio
    P = A [ 0 ];        // valor inicial
    for ( X = 1; X <= N; X = X + 1 )
    {
        // somar um termo do polinomio
        P = P + A [ X ] * pow ( PX, X );
    } // fim for

// 4. mostrar o resultado
    printf ( "\nP(%f) = %f", PX, P );

// pausa para terminar
    printf ( "\nPressionar ENTER para terminar." );
    getchar ( );
    return ( 0 );
} // fim do programa
```

Programa em C++:

```
// Exemplo 3a.
// Calcular o valor do polinomio em um determinado ponto.
//
// bibliotecas necessarias
#include <iostream>
using namespace std;
#include <math.h>
//
// parte principal
//
int main (void)
{
// 1. definir dados
int    N,          // grau do polinomio
      X;          // indice
double A [ 10 ];  // tabela de coeficientes
double PX,        // valor do ponto
      P = 0.0;    // valor do polinomio

// 2. ler dados
// ler o grau do polinomio (N)
cout << "\nQual o grau do polinomio (N) ? ";
cin  >> N;
// ler os coeficientes do polinomio (A's)
for ( X = 0; X <= N; X = X + 1 )
{
// ler um coeficiente
cout << "\nA[" << X << "] = ";
cin  >> A [ X ];
} // fim for
// ler o valor do ponto (X)
cout << "\nEm que ponto (X) ? ";
cin  >> PX;

// 3. calcular o polinomio
P = A [ 0 ];        // valor inicial
for ( X = 1; X <= N; X = X + 1 )
{
// somar um termo do polinomio
P = P + A [ X ] * pow ( PX, X );
} // fim for

// 4. mostrar o resultado
cout << "\nP(" << PX << ") = " << P;

// pausa para terminar
cout << "\nPressionar ENTER para terminar.";
cin.get ( );
return EXIT_SUCCESS;
} // fim do programa
```

Outra versão com procedimentos e função.

Programa em C:

```
// Exemplo 3b.
// Calcular o valor do polinomio em um determinado ponto.
//
// bibliotecas necessarias
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
//

// procedimento para ler dados
void LER ( float COEFICIENTE [ ], int & N )
{
// definir dado local
int X;
// ler dados
printf ( "\nQual o grau do polinomio ? " );
scanf ( "%d", &N );
for ( X = 0; X <= N; X = X + 1 )
{ // ler e guardar o valor de um coeficiente
printf ( "\nQual o valor do coeficiente \"%d ? ", X );
scanf ( "%f", &COEFICIENTE [ X ] );
} // fim for
} // fim do procedimento LER ( )

// procedimento para mostrar dados
void MOSTRAR ( float COEFICIENTE [ ], int N )
{
// definir dado local
int X;
// mostrar dados
for ( X = 0; X <= N; X = X + 1 )
{ // mostrar cada valor de coeficiente
printf ( "\n%d %f", X, COEFICIENTE [ X ] );
} // fiim for
} // fim do procedimento MOSTRAR ( )

// funcao para avaliar o polinomio
float AVALIAR ( float COEFICIENTE [ ], int N, float PX )
{
// definir dados locais
int X;
float P = COEFICIENTE [ 0 ];
// calcular o somatorio
for ( X = 1; X <= N; X = X + 1 )
{ // somar cada termo
P = P + COEFICIENTE [ X ] * pow ( PX, X );
} // fim for
return ( P );
} // fim da funcao AVALIAR ( )
```

```
// parte principal
//
// 1. definir dados
int N;           // grau do polinomio
float A [ 10 ];  // tabela de coeficientes
float PX;        // valor do ponto

// 2. ler dados
// 2.1 ler o grau e os coeficientes do polinomio
LER ( A, N );
// 2.2 ler o valor do ponto (X)
printf ( "\nEm que ponto ? " );
scanf ( "%f", &PX );

// 3. calcular e mostrar o resultado
printf ( "\nP(%f) = %f", PX, AVALIAR ( A, N, PX ) );

// pausa para terminar
printf ( "\nPressionar ENTER para terminar." );
getchar ( );
} // fim do programa
```

Programa em C++:

```
// Exemplo 3b.
// Calcular o valor do polinomio em um determinado ponto.
//
// bibliotecas necessarias
#include <iostream>
using namespace std;
#include <math.h>
//

// procedimento para ler dados
void LER ( float COEFICIENTE [ ], int & N )
{
// definir dado local
int X;
// ler dados
cout << "\nQual o grau do polinomio ? ";
cin >> N;
for ( X = 0; X <= N; X = X + 1 )
{ // ler e guardar o valor de um coeficiente
cout << "\nQual o valor do coeficiente " << X << " ? ";
cin >> COEFICIENTE [ X ];
} // fim for
} // fim do procedimento LER ( )

// procedimento para mostrar dados
void MOSTRAR ( float COEFICIENTE [ ], int N )
{
// definir dado local
int X;
// mostrar dados
for ( X = 0; X <= N; X = X + 1 )
{ // mostrar cada valor de coeficiente
cout << "\n" << X << " " << COEFICIENTE [ X ];
} // fim for
} // fim do procedimento MOSTRAR ( )

// funcao para avaliar o polinomio
float AVALIAR ( float COEFICIENTE [ ], int N, float PX )
{
// definir dados locais
int X;
float P = COEFICIENTE [ 0 ];
// calcular o somatorio
for ( X = 1; X <= N; X = X + 1 )
{ // somar cada termo
P = P + COEFICIENTE [ X ] * pow ( PX, X );
} // fim for
return ( P );
} // fim da funcao AVALIAR ( )
```



```
// parte principal
//
// 1. definir dados
int N;           // grau do polinomio
float A [ 10 ];  // tabela de coeficientes
float PX;        // valor do ponto

// 2. ler dados
// 2.1 ler o grau e os coeficientes do polinomio
LER ( A, N );
// 2.2 ler o valor do ponto (X)
cout << "\nEm que ponto ? ";
cin  >> PX;

// 3. calcular e mostrar o resultado
cout << "\nP(" << PX << ")= " << AVALIAR ( A, N, PX );

// pausa para terminar
cout << "\nPressionar ENTER para terminar.";
getchar ( );
} // fim do programa
```

Programa em C#:

```

/*
 * Exemplo 3a
 * Calcular o valor do polinomio em um determinado ponto.
 */
using System;

class Exemplo_3a
{
    //
    // parte principal
    //
    public static void Main ( )
    {
        // 1. definir dados
        int      N,                // grau do polinomio
            X;                    // indice
        double [ ] A = new double [ 10 ]; // tabela de coeficientes
        double    PX,              // valor do ponto
            P = 0.0;              // valor do polinomio

        // 2. ler dados
        // 2.1. ler o grau do polinomio
        Console.Write ( "\nQual o grau do polinomio ? " );
        N = int.Parse ( Console.ReadLine ( ) );
        // 2.2. ler os coeficientes do polinomio (A's)
        for ( X = 0; X <= N; X = X + 1 )
        {
            // 2.2.1. ler um coeficiente
            Console.Write ( "\nA[" + X + "]= " );
            A [ X ] = double.Parse ( Console.ReadLine ( ) );
        } // fim for
        // 2.3. ler o valor do ponto
        Console.Write ( "\nEm que ponto ? " );
        PX = double.Parse ( Console.ReadLine ( ) );

        // 3. calcular o polinomio
        P = A [ 0 ]; // valor inicial
        for ( X = 1; X <= N; X = X + 1 )
        {
            // somar um termo do polinomio
            P = P + A [ X ] * Math.Pow ( PX, X );
        } // fim for

        // 4. mostrar resultado
        Console.WriteLine ( "\nP(" + PX + ")= " + P );

        // pausa para terminar
        Console.Write ( "\nPressionar ENTER para terminar." );
        Console.ReadLine ( );
    } // end Main ( )
} // fim Exemplo_3a class

```

Outra versão do programa em C#:

```

/*
 * Exemplo 3b
 * Calcular o valor do polinomio em um determinado ponto.
 */
using System;

class Exemplo_3b
{
    // definir dado global
    static int N = 0;          // grau do polinomio

    // procedimento para ler dados
    static void LER ( double [ ] COEFICIENTE )
    {
        // definir dado local
        int X;                 // indice
        // ler dados
        Console.Write ( "\nQual o grau do polinomio ? " );
        N = int.Parse ( Console.ReadLine ( ) );
        for ( X = 0; X <= N; X = X + 1 )
        { // ler e guardar o valor de um coeficiente
            Console.Write ( "\nA[" + X + "]= " );
            COEFICIENTE [ X ] = double.Parse ( Console.ReadLine ( ) );
        } // fim for
    } // fim do procedimento LER ( )

    // procedimento para mostrar dados
    static void MOSTRAR ( double [ ] COEFICIENTE, int N )
    {
        // definir dado local
        int X;                 // indice
        // mostrar dados
        for ( X = 0; X <= N; X = X+1 )
        { // mostrar cada valor de coeficiente
            Console.WriteLine ( "\n" + X + " " + COEFICIENTE [ X ] );
        } // fim for
    } // fim do procedimento MOSTRAR ( )

    // funcao para avaliar o polinomio
    static double AVALIAR ( double [ ] COEFICIENTE, int N, double PX )
    {
        // definir dados locais
        int X;                 // indice
        double P = COEFICIENTE [ 0 ];
        // calcular somatorio
        for ( X = 1; X <= N; X = X+1 )
        { // somar cada termo
            P = P + COEFICIENTE [ X ] * Math.Pow ( PX, X );
        } // fim for
        return ( P );
    } // fim da funcao AVALIAR ( )
}

```

```

//
// parte principal
//
public static void main ( String [ ] args )
{
    // 1. definir dados
    double [ ] A = new double [ 10 ];    // tabela de coeficientes
    double    PX,                        // valor do ponto
              P = 0.0;                  // valor do polinomio

    // 2. ler dados
    // 2.1. ler o grau e os coeficientes do polinomio
        LER ( A );
    // 2.2. ler o valor do ponto
        Console.Write ( "\nEm que ponto ? " );
        PX = double.Parse ( Console.ReadLine ( ) );

    // 3. calcular e mostrar o resultado
        Console.WriteLine ( "\nP(" + PX + ")= " + AVALIAR ( A, N, PX ) );

    // pausa para terminar
        Console.Write ( "\nPressionar ENTER para terminar." );
        Console.ReadLine ( );
    } // end Main ( )
} // fim Exemplo_3b class

```

Programa em Java:

```

/**
 * Exemplo 3a
 * Calcular o valor do polinomio em um determinado ponto.
 */

// ----- classes necessarias

// ----- definicao de classe

class Exemplo_3a
{
//
// parte principal
//
    public static void main ( String [ ] args )
    {
        // 1. definir dados
        int      N,                // grau do polinomio
            X;                    // indice
        double [ ] A = new double [ 10 ]; // tabela de coeficientes
        double    PX,              // valor do ponto
            P = 0.0;              // valor do polinomio

        // 2. ler dados
        // 2.1. ler o grau do polinomio
        System.out.print ( "\nQual o grau do polinomio ? " );
        N = Integer.parseInt ( System.console( ).readLine( ) );
        // 2.2. ler os coeficientes do polinomio (A's)
        for ( X = 0; X <= N; X = X + 1 )
        {
            // 2.2.1. ler um coeficiente
            System.out.print ( "\nA[" + X + "]= " );
            A [ X ] = Double.parseDouble ( System.console( ).readLine( ) );
        } // fim for
        // 2.3. ler o valor do ponto
        System.out.print ( "\nEm que ponto ? " );
        PX = Double.parseDouble ( System.console( ).readLine( ) );

        // 3. calcular o polinomio
        P = A [ 0 ]; // valor inicial
        for ( X = 1; X <= N; X = X + 1 )
        {
            // somar um termo do polinomio
            P = P + A [ X ] * Math.pow ( PX, X );
        } // fim for

        // 4. mostrar resultado
        System.out.println ( "\nP(" + PX + ")= " + P );
        // pausa para terminar
        System.out.print ( "\nPressionar ENTER para terminar." );
        System.console( ).readLine( );
    } // end main ( )
} // fim Exemplo_3a class

```

Outra versão do programa em Java:

```

/**
 * Exemplo 3b
 * Calcular o valor do polinomio em um determinado ponto.
 */

// ----- classes necessarias

// ----- definicao de classe

class Exemplo_3b
{
// definir dado global
    public static int N = 0;                // grau do polinomio

// procedimento para ler dados
    public static void LER ( double [ ] COEFICIENTE )
    {
        // definir dado local
        int X;                            // indice
        // ler dados
        System.out.print ( "\nQual o grau do polinomio ? " );
        N = Integer.parseInt ( System.console( ).readLine( ) );
        for ( X = 0; X <= N; X = X + 1 )
        { // ler e guardar o valor de um coeficiente
            System.out.print ( "\nA[" + X + "] = " );
            COEFICIENTE [ X ] = Double.parseDouble ( System.console( ).readLine( ) );
        } // fim for
    } // fim do procedimento LER ( )

// procedimento para mostrar dados
    public static void MOSTRAR ( double [ ] COEFICIENTE, int N )
    {
        // definir dado local
        int X;                            // indice
        // mostrar dados
        for ( X = 0; X <= N; X = X+1 )
        { // mostrar cada valor de coeficiente
            System.out.println ( "" + X + " " + COEFICIENTE [ X ] );
        } // fim for
    } // fim do procedimento MOSTRAR ( )
}

```

```

// funcao para avaliar o polinomio
public static double AVALIAR ( double [ ] COEFICIENTE, int N, double PX )
{
    // definir dados locais
    int X; // indice
    double P = COEFICIENTE [ 0 ];
    // calcular somatorio
    for ( X = 1; X <= N; X = X+1 )
    { // somar cada termo
        P = P + COEFICIENTE [ X ] * Math.pow ( PX, X );
    } // fim for
    return ( P );
} // fim da funcao AVALIAR ( )

//
// parte principal
//
public static void main ( String [ ] args )
{
    // 1. definir dados
    double [ ] A = new double [ 10 ]; // tabela de coeficientes
    double PX, // valor do ponto
        P = 0.0; // valor do polinomio

    // 2. ler dados
    // 2.1. ler o grau e os coeficientes do polinomio
    LER ( A );
    // 2.2. ler o valor do ponto
    System.out.print ( "\nEm que ponto ? " );
    PX = Double.parseDouble ( System.console( ).readLine( ) );

    // 3. calcular e mostrar o resultado
    System.out.println ( "\nP(" + PX + ")= " + AVALIAR ( A, N, PX ) );

    // pausa para terminar
    System.out.print ( "\nPressionar ENTER para terminar." );
    System.console( ).readLine( );
} // end main ( )

} // fim Exemplo_3b class

```

Programa em Python:

```
# Exemplo 3.
# Calcular o valor do polinomio em um determinado ponto.
#
from array import *;
# 1. definir dados
N = 0;          # grau do polinomio
X = 0;          # indice
               # tabela de coeficientes
A = array ( 'f', [ 0.0, 0.0, 0.0, 0.0, 0.0,
                  0.0, 0.0, 0.0, 0.0, 0.0 ] );
PX = 0;         # valor do ponto
P = 0.0;        # valor do polinomio

# 2. ler dados
# ler o grau do polinomio (N)
N = int ( input ( "\nQual o grau do polinomio ? " ) );
# ler os coeficientes do polinomio (A's)
for X in range ( 0, (N+1), 1 ):
# ler um coeficiente
    print ( "\nA [ ", X, " ] = ", end=" " );
    A [ X ] = float ( input ( " " ) );
# fim repetir
# ler o valor do ponto (X)
PX = float ( input ( " \nEm que ponto ? " ) );

# 3. calcular o polinomio
P = 0.0;        # valor inicial
for X in range ( 0, (N+1), 1 ):
# somar um termo do polinomio
    P = P + A [ X ] * PX ** ( X );
# fim repetir

# 4. mostrar o resultado
print ( "\nP ( ", PX, " ) = ", P );

# pausa para terminar
print ( "\nPressionar ENTER para terminar." );
input ( );
# fim do programa
```

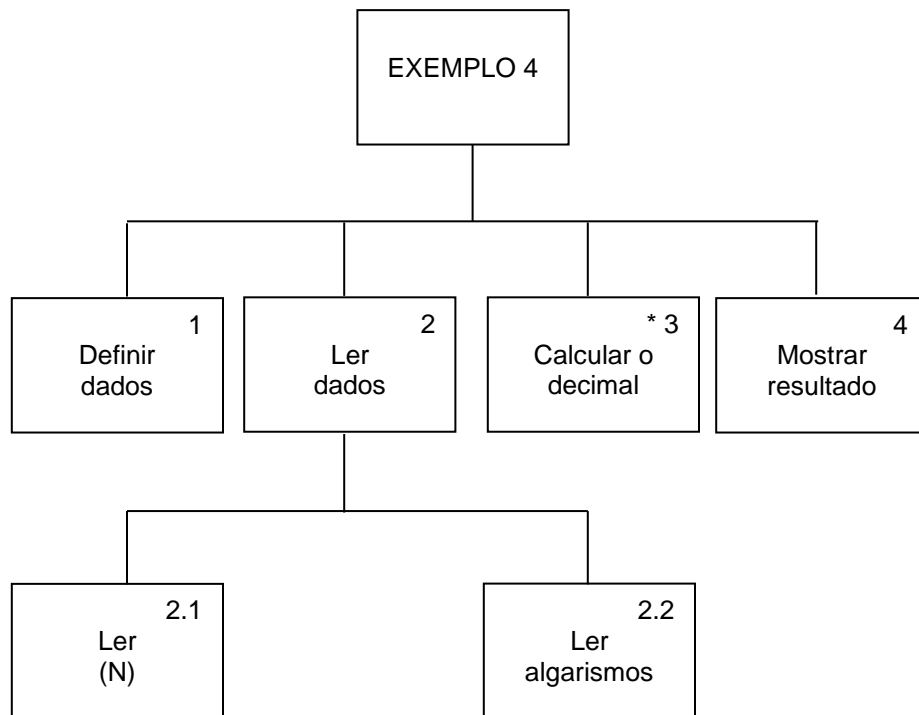


## Exemplo 4.

Montar uma tabela de, no máximo 10, valores inteiros, com os algarismos de um número inteiro expresso na base 2. Os valores de N e dos algarismos serão lidos do teclado, e deseja-se calcular o valor decimal equivalente.

$$DECIMAL = a_n 2^n + a_{n-1} 2^{n-1} + a_{n-2} 2^{n-2} + \dots + a_1 2^1 + a_0 2^0$$

Diagrama funcional:



Análise de dados :

- Dados do problema :

Dado	Tipo	Valor inicial	Função
A	inteiro (10)	-	armazenar até 10 algarismos
N	inteiro	-	número de algarismos
I	inteiro	-	índice

- Resultados do problema :

Dado	Tipo	Valor inicial	Função
DECIMAL	inteiro	0	valor decimal equivalente

- Avaliação da solução :

Índice	Dado	Resultado
0	1	
1	0	
2	1	
3	1	
	DECIMAL	13

Algoritmo:

Esboço:

Primeira versão, só comentários.

Exemplo 4	v.1
Ação	Bloco
! definir dados	1
! ler dados	2
! ler o número de algarismos (N)	2.1
! ler cada algarismo (A's)	2.2
! calcular o decimal equivalente	3
! mostrar o resultado	4

Segunda versão, com refinamento do primeiro bloco.

Exemplo 4	v.2
Ação	Bloco
! definir dados	1
inteiro DECIMAL = 0, ! valor equivalente N, ! número de algarismos X, ! índice A [ 10 ]; ! tabela de algarismos	
! ler dados	2
! ler o número de algarismos (N)	2.1
! ler cada algarismo (A's)	2.2
! calcular o decimal equivalente	3
! mostrar o resultado	4

Terceira versão, com refinamento do segundo e do quarto bloco.

Exemplo 4	v.3
Ação	Bloco
! definir dados	1
inteiro DECIMAL = 0, ! valor equivalente N, ! número de algarismos X, ! índice A [ 10 ]; ! tabela de algarismos	
! ler dados	2
! ler o número de algarismos (N)	2.1
tela ← "Quantos algarismos (N) ? "; N ← teclado;	
! ler cada algarismo (A's)	2.2
X = 1 : N : 1	2.2.1
! ler um coeficiente	
tela ← ("A[", X, "] = "); A [ X ] ← teclado;	2.2.2
! calcular o decimal equivalente	3
! mostrar o resultado	4
tela ← ( "DECIMAL = ", DECIMAL );	

Quarta versão, com refinamento do terceiro bloco.

Exemplo 4	v.3
Ação	Bloco
! definir dados	1
inteiro DECIMAL = 0, ! valor equivalente N, ! número de algarismos X, ! índice P = 1, ! potência de 2 A [ 10 ]; ! tabela de algarismos	
! ler dados	2
! ler o número de algarismos (N)	2.1
tela ← "Quantos algarismos (N) ? "; N ← teclado;	
! ler cada algarismo (A's)	2.2
X = 1 : N : 1	2.2.1
! ler um coeficiente	
tela ← ("A[", X, "] = "); A [ X ] ← teclado;	2.2.2
! calcular o decimal equivalente	3
X = 1 : N : 1	3.2
! somar cada termo	3.2.1
DECIMAL = DECIMAL + A [ X ] * P;	
! calcular a próxima potência	3.2.2
P = P * 2;	
! calcular o decimal equivalente	4
! mostrar o resultado	
tela ← ( "DECIMAL = ", DECIMAL );	

Programa em SCILAB:

```
// Exemplo 4.
// Calcular o valor decimal equivalente de um binario.
//
// 1. definir dados
DECIMAL = 0;           // valor equivalente
N = 0;                 // numero de algarismos
X = 0;                 // indice
P = 1;                 // potencia de 2
A ( 1:10 ) = 0;        // tabela de algarismos

// 2. ler dados
// 2.1 ler o numero de algarismos
N = input ( "\nQuantos algarismos ? " );
// 2.2 ler cada algarismo (A's)
for ( X = 1 : 1 : N )
    // ler um algarismo
    printf ( "\nA( %d ) = ", X );
    A ( X ) = input ( " " );
end // for

// 3. calcular o decimal equivalente
for ( X = 1 : 1 : N )
// 3.1 somar cada termo
    DECIMAL = DECIMAL + A ( X ) * P;
// 3.2 calcular a proxima potencia
    P = P * 2;
end // for

// 4. mostrar o resultado
printf ( "\nDECIMAL = %d ", DECIMAL );

// pausa para terminar
printf ( "\nPressionar ENTER para terminar." );
halt;
// fim do programa
```

Programa em C:

```
// Exemplo 4a.
// Calcular o valor decimal equivalente a um binario.
//
// bibliotecas necessarias
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
//
// parte principal
//
int main (void)
{
// 1. definir dados
    int DECIMAL = 0,           // valor equivalente
        N,                    // numero de algarismos
        X,                    // indice
        P = 1,                // potencia de 2
        A [ 10 ];             // tabela de algarismos

// 2. ler dados
// 2.1 ler o número de algarismos
    printf ( "\nQuantos algarismos ? " );
    scanf ( "%d", &N );
// 2.2 ler cada algarismo (A's)
    for ( X = 0; X <= N; X = X + 1 )
    {
        // ler um algarismo
        printf ( "\nA[%d] = ", X );
        scanf ( "%d", &A [ X ] );
    } // fim for

// 3. calcular o decimal equivalente
    for ( X = 0; X < N; X = X + 1 )
    {
        // 3.1 somar cada termo
        DECIMAL = DECIMAL + A [ X ] * P;
        // 3.2 calcular a proxima potencia
        P = P * 2;
    }

// 4. mostrar o resultado
    printf ( "\nDECIMAL = %d", DECIMAL );

// pausa para terminar
    printf ( "\nPressionar ENTER para terminar." );
    getchar ( );
    return ( 0 );
} // fim do programa
```

Programa em C++:

```
// Exemplo 4a.
// Calcular o valor decimal equivalente a um binario.
//
// bibliotecas necessarias
#include <iostream>
using namespace std;
#include <math.h>
//
// parte principal
//
int main (void)
{
// 1. definir dados
    int DECIMAL = 0,           // valor equivalente
        N,                     // numero de algarismos
        X,                     // indice
        P = 1,                 // potencia de 2
        A [ 10 ];              // tabela de algarismos

// 2. ler dados
// 2.1 ler o número de algarismos
    cout << "\nQuantos algarismos ? ";
    cin  >> N;
// 2.2 ler cada algarismo (A's)
    for ( X = 0; X <= N; X = X + 1 )
    {
        // ler um algarismo
        cout << "\nA[" << X << "] = ";
        cin  >> A [ X ];
    } // fim for

// 3. calcular o decimal equivalente
    for ( X = 0; X < N; X = X + 1 )
    {
        // 3.1 somar cada termo
        DECIMAL = DECIMAL + A [ X ] * P;
        // 3.2 calcular a proxima potencia
        P = P * 2;
    }

// 4. mostrar o resultado
    cout << "\nDECIMAL = " << DECIMAL;

// pausa para terminar
    cout << "\nPressionar ENTER para terminar.";
    getchar ( );
    return EXIT_SUCCESS;
} // fim do programa
```

Programa em C#:

```

/*
 * Exemplo 4a
 * Calcular o valor decimal equivalente a um binario.
 */
using System;

class Exemplo_4a
{
    //
    // parte principal
    //
    public static void Main ( )
    {
        // 1. definir dados
        int [ ] A = new int [ 10 ];    // tabela de Algarismos
        int DECIMAL = 0,              // valor equivalente
            N,                          // numero de Algarismos
            X,                          // indice
            P = 1;                     // potencia de 2

        // 2. ler dados
        // 2.1. ler o numero de Algarismos
        Console.Write ( "\nQuantos Algarismos ? " );
        N = int.Parse ( Console.ReadLine ( ) );
        // 2.2. ler cada Algarismo (A's)
        for ( X = 0; X < N; X = X + 1 )
        {
            // 2.2.1. ler um coeficiente
            Console.Write ( "\nA[" + X + "] = " );
            A [ X ] = int.Parse ( Console.ReadLine ( ) );
        } // fim for

        // 3. calcular o decimal equivalente
        for ( X = 0; X < N; X = X + 1 )
        {
            // 3.1. somar cada termo
            DECIMAL = DECIMAL + A [ X ] * P;
            // 3.2. calcular a proxima potencia
            P = P * 2;
        } // fim for

        // 4. mostrar o resultado
        Console.WriteLine ( "\nDECIMAL = " + DECIMAL );

        // pausa para terminar
        Console.Write ( "\nPressionar ENTER para terminar." );
        Console.ReadLine ( );
    } // end Main ( )
} // fim Exemplo_4a class

```



Programa em Java:

```

/**
 * Exemplo 4a
 * Calcular o valor decimal equivalente a um binario.
 */

// ----- classes necessarias

// ----- definicao de classe

class Exemplo_4a
{
//
// parte principal
//
    public static void main ( String [ ] args )
    {
        // 1. definir dados
        int [ ] A = new int [ 10 ];    // tabela de algarismos
        int DECIMAL = 0,              // valor equivalente
            N,                          // numero de algarismos
            X,                          // indice
            P = 1;                     // potencia de 2

        // 2. ler dados
        // 2.1. ler o numero de algarismos
        System.out.print ( "\nQuantos algarismos ? " );
        N = Integer.parseInt ( System.console( ).readLine( ) );
        // 2.2. ler cada algarismo (A's)
        for ( X = 0; X < N; X = X + 1 )
        {
            // 2.2.1. ler um coeficiente
            System.out.print ( "\nA[" + X + "]= " );
            A [ X ] = Integer.parseInt ( System.console( ).readLine( ) );
        } // fim for

        // 3. calcular o decimal equivalente
        for ( X = 0; X < N; X = X+1 )
        {
            // 3.1. somar cada termo
            DECIMAL = DECIMAL + A [ X ] * P;
            // 3.2. calcular a proxima potencia
            P = P * 2;
        } // fim for

        // 4. mostrar o resultado
        System.out.println ( "\nDECIMAL = " + DECIMAL );

        // pausa para terminar
        System.out.print ( "\nPressionar ENTER para terminar." );
        System.console( ).readLine( );
    } // end main ( )
} // fim Exemplo_4a class

```

Outra versão com procedimentos e função.

Programa em C:

```
// Exemplo 4b.
// Calcular o valor decimal equivalente a um binario.
//
// bibliotecas necessarias
#include <stdio.h>
#include <stdlib.h>
//

// procedimento para ler dados
void LER_BINARIO ( int ALGARISMO [ ], int & N )
{
// definir dado local
int X;
// ler dados
printf ( "\nQuantos algarismos ? " );
scanf ( "%d", &N );
for ( X = 0; X <= N; X = X + 1 )
{ // ler e guardar cada algarismo
do
{
printf ( "\nQual o algarismo %d ? ", X );
scanf ( "%d", &ALGARISMO [ X ] );
}
while ( ALGARISMO [ X ] != 0 && ALGARISMO [ X ] != 1 );
} // fim for
} // fim do procedimento LER_BINARIO ( )

// procedimento para mostrar dados
void MOSTRAR_BINARIO ( float ALGARISMO [ ], int N )
{
// definir dado local
int X;
// mostrar dados
for ( X = N; X >= 0; X = X - 1 )
{ // mostrar cada valor de coeficiente
printf ( "\n%d  %f", X, ALGARISMO [ X ] );
} // fim for
} // fim do procedimento MOSTRAR_BINARIO ( )
```

```

// funcao para converter para decimal
int DECIMAL ( int ALGARISMO [ ], int N, int PX )
{
// definir dados locais
    int X,           // indice
        P = 1 ;      // potencia de X
    int D = 0;        // valor decimal
// calcular o somatorio
    for ( X = 0; X <= N; X = X + 1 )
    { // somar cada termo
        D = D + ALGARISMO [ X ] * P;
        // calcular a proxima potencia
        P = P * PX;
    } // fim for
    return ( P );
} // fim da funcao DECIMAL ( )

// parte principal
//
//
int main (void)
{
// 1. definir dados
    int N,           // numero de algarismos
        A [ 10 ];    // tabela de algarismos

// 2. ler os algarismos do número binario
    LER_BINARIO ( A, N );

// 3. calcular e mostrar o resultado
    MOSTRAR_BINARIO ( A, N );
    printf ( "\nDECIMAL equivalente = %d", DECIMAL ( A, N, 2 ) );

// pausa para terminar
    printf ( "\nPressionar ENTER para terminar." );
    getchar ( );
    return ( 0 );
} // fim do programa

```

Programa em C++:

```
// Exemplo 4b.
// Calcular o valor decimal equivalente a um binario.
//
// bibliotecas necessarias
#include <iostream>
using namespace std;
//

// procedimento para ler dados
void LER_BINARIO ( int ALGARISMO [ ], int & N )
{
    // definir dado local
    int X;
    // ler dados
    cout << "\nQuantos algarismos ? ";
    cin >> N;
    for ( X = 0; X <= N; X = X + 1 )
    { // ler e guardar cada algarismo
        do
        {
            cout << "\nQual o algarismo " << X << " ? ";
            cin >> ALGARISMO [ X ];
        }
        while ( ALGARISMO [ X ] != 0 && ALGARISMO [ X ] != 1 );
    } // fim for
} // fim do procedimento LER_BINARIO ( )

// procedimento para mostrar dados
void MOSTRAR_BINARIO ( int ALGARISMO [ ], int N )
{
    // definir dado local
    int X;
    // mostrar dados
    for ( X = N; X >= 0; X = X - 1 )
    { // mostrar cada valor de coeficiente
        cout << "\n" << X << " " << ALGARISMO [ X ];
    } // fiim for
} // fim do procedimento MOSTRAR_BINARIO ( )
```

```

// funcao para converter para decimal
int DECIMAL ( int ALGARISMO [ ], int N, int PX )
{
// definir dados locais
    int X,           // indice
        P = 1 ;      // potencia de X
    int D = 0;        // valor decimal
// calcular o somatorio
    for ( X = 0; X <= N; X = X + 1 )
    { // somar cada termo
        D = D + ALGARISMO [ X ] * P;
        // calcular a proxima potencia
        P = P * PX;
    } // fim for
    return ( P );
} // fim da funcao DECIMAL ( )


// parte principal
//
//
int main (void)
{
// 1. definir dados
    int N,           // numero de algarismos
        A [ 10 ];    // tabela de algarismos

// 2. ler os algarismos do número binario
    LER_BINARIO ( A, N );

// 3. calcular e mostrar o resultado
    MOSTRAR_BINARIO ( A, N );
    cout << "\nDECIMAL equivalente = " << DECIMAL ( A, N, 2 );

// pausa para terminar
    cout << "\nPressionar ENTER para terminar.";
    getchar ( );
    return EXIT_SUCCESS;
} // fim do programa

```

Outra versão do programa em C#:

```

/*
 * Exemplo 4b
 * Calcular o valor decimal equivalente a um binario.
 */
using System;

class Exemplo_4b
{
    // definir dado global
    public static int N = 0; // numero de algarismos

    // procedimento para ler dados
    public static void LER_BINARIO ( int [ ] ALGARISMO )
    {
        // definir dado local
        int X; // indice
        // ler dados
        Console.Write ( "\nQuantos algarismos ? " );
        N = int.Parse ( Console.ReadLine ( ) );
        for ( X = 0; X < N; X = X + 1 )
        {
            do
            {
                // ler e guardar cada algarismo
                Console.Write ( "\nQual o algarismo " + X + " ? " );
                ALGARISMO [ X ] = int.Parse ( Console.ReadLine ( ) );
            }
            while ( ALGARISMO [ X ] != 0 && ALGARISMO [ X ] != 1 );
        } // fim for
    } // fim do procedimento LER_BINARIO ( )

    // procedimento para mostrar dados
    public static void MOSTRAR_BINARIO ( int [ ] ALGARISMO, int N )
    {
        // definir dado local
        int X; // indice
        // mostrar dados
        for ( X = 0; X <= N; X = X+1 )
        {
            // mostrar cada valor de algarismo
            Console.WriteLine ( "\n" + X + " " + ALGARISMO [ X ] );
        } // fim for
    } // fim do procedimento MOSTRAR_BINARIO ( )
}

```

```

// funcao para converter para decimal
public static int DECIMAL ( int [ ] ALGARISMO, int N, int PX )
{
    // definir dados locais
    int X,           // indice
        P = 1;       // potencia de X
    int D = 0;       // valor decimal
    // calcular somatorio
    for ( X = 0; X < N; X = X+1 )
    {
        // somar cada termo
        D = D + ALGARISMO [ X ] * P;
        P = P * PX;
    } // fim for
    return ( D );
} // fim da funcao DECIMAL ( )

//
// parte principal
//
public static void Main ( )
{
    // 1. definir dados
    int [ ] A = new int [ 10 ];    // tabela de algarismos

    // 2. ler os algarismos do numero binario
    LER_BINARIO ( A );

    // 3. calcular e mostrar o resultado
    Console.WriteLine ( "\nDECIMAL equivalente = " + DECIMAL ( A, N, 2 ) );

    // pausa para terminar
    Console.Write ( "\nPressionar ENTER para terminar." );
    Console.ReadLine ( );
} // end Main ( )

} // fim Exemplo_4b class

```

Outra versão do programa em Java:

```

/**
 * Exemplo 4b
 * Calcular o valor decimal equivalente a um binario.
 */

// ----- classes necessarias

// ----- definicao de classe

class Exemplo_4b
{
// definir dado global
    public static int N = 0; // numero de algarismos

// procedimento para ler dados
    public static void LER_BINARIO ( int [ ] ALGARISMO )
    {
        // definir dado local
        int X;           // indice
        // ler dados
        System.out.print ( "\nQuantos algarismos ? " );
        N = Integer.parseInt ( System.console( ).readLine( ) );
        for ( X = 0; X < N; X = X + 1 )
        {
            do
            {
                // ler e guardar cada algarismo
                System.out.print ( "\nQual o algarismo " + X + " ? " );
                ALGARISMO [ X ] = Integer.parseInt ( System.console( ).readLine( ) );
            }
            while ( ALGARISMO [ X ] != 0 && ALGARISMO [ X ] != 1 );
        } // fim for
    } // fim do procedimento LER_BINARIO ( )

// procedimento para mostrar dados
    public static void MOSTRAR_BINARIO ( int [ ] ALGARISMO, int N )
    {
        // definir dado local
        int X;           // indice
        // mostrar dados
        for ( X = 0; X <= N; X = X+1 )
        {
            // mostrar cada valor de algarismo
            System.out.println ( "\n" + X + " " + ALGARISMO [ X ] );
        } // fim for
    } // fim do procedimento MOSTRAR_BINARIO ( )
}

```



```

// funcao para converter para decimal
public static int DECIMAL ( int [ ] ALGARISMO, int N, int PX )
{
    // definir dados locais
    int X,           // indice
        P = 1;       // potencia de X
    int D = 0;       // valor decimal
    // calcular somatorio
    for ( X = 0; X < N; X = X+1 )
    {
        // somar cada termo
        D = D + ALGARISMO [ X ] * P;
        P = P * PX;
    } // fim for
    return ( D );
} // fim da funcao DECIMAL ( )

//
// parte principal
//
public static void main ( String [ ] args )
{
    // 1. definir dados
    int [ ] A = new int [ 10 ];    // tabela de algarismos

    // 2. ler os algarismos do numero binario
    LER_BINARIO ( A );

    // 3. calcular e mostrar o resultado
    System.out.println ( "\nDECIMAL equivalente = " + DECIMAL ( A, N, 2 ) );

    // pausa para terminar
    System.out.print ( "\nPressionar ENTER para terminar." );
    System.console ( ).readLine ( );
} // end main ( )

} // fim Exemplo_4b class

```

Programa em Python:

```
# Exemplo 4.
# Calcular o valor decimal equivalente de um binario.
#
from array import *;
# 1. definir dados
DECIMAL = 0; # valor equivalente
N = 0;      # numero de algarismos
X = 0;      # indice
P = 1;      # potencia de 2
            # tabela de algarismos
A = array ( 'i', [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ] );

# 2. ler dados
# 2.1 ler o numero de algarismos
N = int ( input ( " \nQuantos algarismos ? " ) );
# 2.2 ler cada algarismo (A's)
for X in range ( 0, N, 1 ):
# ler um algarismo
    print ( " \nA [ ", X, " ] = ", end=" " );
    A [ X ] = int ( input ( " " ) );
# fim repetir

# 3. calcular o decimal equivalente
for X in range ( 0, N, 1 ):
# 3.1 somar cada termo
    DECIMAL = DECIMAL + A [ X ] * P;
# 3.2 calcular a proxima potencia
    P = P * 2;
# fim repetir

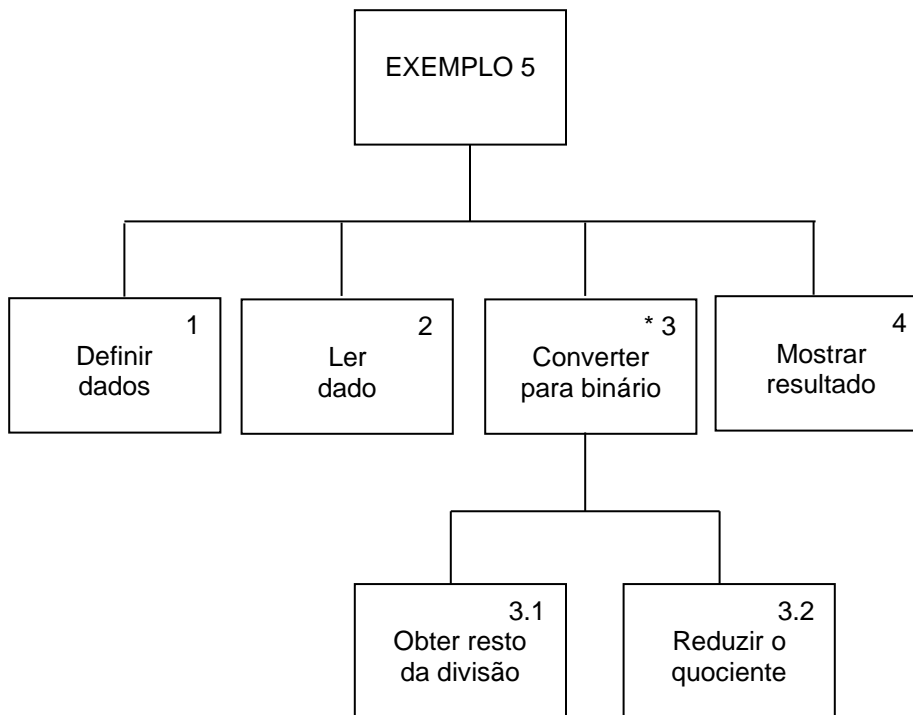
# 4. mostrar o resultado
print ( " \nDECIMAL = ", DECIMAL );

# pausa para terminar
print ( " \nPressionar ENTER para terminar." );
input ( );
# fim do programa
```

Exemplo 5.

Ler um número inteiro (X) do teclado e convertê-lo para o equivalente em binário, através de divisões sucessivas.

Diagrama funcional:



Análise de dados :

- Dados do problema :

Dado	Tipo	Valor inicial	Função
DECIMAL	inteiro	-	valor decimal
I	inteiro	-	índice
QUOCIENTE	inteiro	-	quociente da divisão por 2

- Resultados do problema :

Dado	Tipo	Valor inicial	Função
A	inteiro (10)	-	armazenar até 10 algarismos
N	inteiro	-	número de algarismos

- Avaliação da solução :

Índice	Dado DECIMAL=13	Resultado
0		1
1		0
2		1
3		1

Algoritmo:

Esboço:

Primeira versão, só comentários.

Exemplo 5	v.1
Ação	Bloco
! definir dados	1
! ler dado	2
! calcular o binário equivalente	3
! guardar o resto de divisão por 2	3.1
! reduzir o quociente	3.2
! mostrar o resultado	4

Segunda versão, com refinamento do primeiro bloco.

Exemplo 5	v.2
Ação	Bloco
! definir dados	1
inteiro DECIMAL, ! valor decimal N, ! número de algarismos X, ! índice QUOCIENTE, ! quociente A [ 10 ]; ! tabela de algarismos	
! ler dado	2
! calcular o binário equivalente	3
! guardar os restos de divisão por 2	3.1
! reduzir o quociente a cada divisão	3.2
! mostrar o resultado	4

Terceira versão, com refinamento do segundo e do quarto bloco.

Exemplo 5	v.3
Ação	Bloco
! definir dados	1
inteiro DECIMAL, ! valor decimal N, ! número de algarismos X, ! índice QUOCIENTE, ! quociente A [ 10 ]; ! tabela de algarismos	
! ler dado	2
tela ← “\nQual o valor decimal ? “; DECIMAL ← teclado;	
! calcular o binário equivalente	3
! guardar os restos de divisão por 2	3.1
! reduzir o quociente a cada divisão	3.2
! mostrar o resultado	4
tela ← ( “\nDECIMAL = ”, DECIMAL ); tela ← “\nBINÁRIO = “;	4.1
X = (N+1) : 1 : (-1)	4.2
! mostrar um algarismo	
tela ← A [ X ];	4.2.1

Quarta versão, com refinamento do terceiro bloco.

Exemplo 5	v.3
Ação	Bloco
! definir dados	1
inteiro DECIMAL, ! valor decimal N, ! número de algarismos X, ! índice QUOCIENTE, ! quociente A [ 10 ]; ! tabela de algarismos	
! ler dado	2
tela ← “\nQual o valor decimal ? “; DECIMAL ← teclado;	
! calcular o binário equivalente	3
QUOCIENTE = DECIMAL;	3.1
! guardar os restos da divisão por 2	3.2
QUOCIENTE > 0 ?	
! guardar resto de divisão por 2	3.2.1
A [ X ] = QUOCIENTE % 2;	
! reduzir o quociente a cada divisão	3.2.2
QUOCIENTE = QUOCIENTE / 2;	
! mostrar o resultado	4
tela ← ( “\nDECIMAL = ”, DECIMAL ); tela ← “\nBINÁRIO = “;	4.1
X = (N+1) : 1 : (-1)	4.2
! mostrar um algarismo	4.2.1
tela ← A [ X ];	

Programa em SCILAB:

```
// Exemplo 5.
// Calcular o valor binario equivalente a um decimal.
//
// 1. definir dados
DECIMAL = 0;           // valor decimal
N = 0;                 // numero de Algarismos
X = 0;                 // indice
QUOCIENTE = 0;         // quociente
A ( 1:10 ) = 0;        // tabela de Algarismos

// 2. ler dado
DECIMAL = input ( "Qual o valor decimal ? " );

// 3. calcular o binario equivalente
// 3.1 copiar o valor decimal, para nao perder
QUOCIENTE = DECIMAL;
// 3.2 obter os restos da divisao por 2
while ( QUOCIENTE > 0 )
// 3.2.1 guardar um resto de divisao por 2
N = N + 1;
A ( N ) = mod ( QUOCIENTE, 2 );
// 3.2.2 reduzir o quociente
QUOCIENTE = floor (QUOCIENTE / 2);
end // while

// 4. mostrar o resultado
// 4.1 mostrar o valor decimal
printf ( "\nDECIMAL= %d ", DECIMAL );
// 4.2 mostrar o equivalente binario
printf ( "\nBINARIO = " );
for ( X = N : (-1) : 1 )
// mostrar um Algarismo
printf ( " %d ", A ( X ) );
end // for

// pausa para terminar
printf ( "\nPressionar ENTER para terminar." );
halt;
// fim do programa
```

Programa em C:

```
// Exemplo 5a.
// Calcular o valor binario equivalente a um decimal.
//
// bibliotecas necessarias
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
//
// parte principal
//
int main (void)
{
// 1. definir dados
int DECIMAL, // valor decimal
    N = 0, // número de algarismos
    X, // indice
    QUOCIENTE, // quociente
    A [ 10 ]; // tabela de algarismos

// 2. ler dado
printf ( "\nQual o valor decimal ? " );
scanf ( "%d", &DECIMAL );

// 3. calcular o binario equivalente
// 3.1 copiar o valor decimal, para nao perder
QUOCIENTE = DECIMAL;
// 3.2 obter os restos da divisao por 2
while ( QUOCIENTE > 0 )
{
// 3.2.1 guardar um resto de divisao por 2
N = N + 1;
A [ N ] = QUOCIENTE % 2;
// 3.2.2 reduzir o quociente
QUOCIENTE = QUOCIENTE / 2;
} // fim while

// 4. mostrar o resultado
// 4.1 mostrar o valor decimal
printf ( "\nDECIMAL= %d", DECIMAL );
// 4.2 mostrar o equivalente binario
printf ( "\nBINARIO = " );
for ( X = N; X >= 1; X = X - 1 )
{
// mostrar um algarismo
printf ( "%d", A [ X ] );
} // fim for

// pausa para terminar
printf ( "Pressionar ENTER para terminar." );
getchar ( );
return ( 0 );
} // fim do programa
```



Programa em C++:

```
// Exemplo 5a.
// Calcular o valor binario equivalente a um decimal.
//
// bibliotecas necessarias
#include <iostream>
using namespace std;
#include <math.h>
//
// parte principal
//
int main (void)
{
// 1. definir dados
int DECIMAL, // valor decimal
    N = 0,    // número de algarismos
    X,        // indice
    QUOCIENTE, // quociente
    A [ 10 ]; // tabela de algarismos

// 2. ler dado
cout << "\nQual o valor decimal ? ";
cin  >> DECIMAL;

// 3. calcular o binario equivalente
// 3.1 copiar o valor decimal, para nao perder
QUOCIENTE = DECIMAL;
// 3.2 obter os restos da divisao por 2
while ( QUOCIENTE > 0 )
{
// 3.2.1 guardar um resto de divisao por 2
N = N + 1;
A [ N ] = QUOCIENTE % 2;
// 3.2.2 reduzir o quociente
QUOCIENTE = QUOCIENTE / 2;
} // fim while

// 4. mostrar o resultado
// 4.1 mostrar o valor decimal
cout << "\nDECIMAL=" << DECIMAL;
// 4.2 mostrar o equivalente binario
cout << "\nBINARIO = ";
for ( X = N; X >= 1; X = X - 1 )
{
// mostrar um algarismo
cout << A [ X ];
} // fim for

// pausa para terminar
cout << "Pressionar ENTER para terminar.";
cin.get ( );
return EXIT_SUCCESS;
} // fim do programa
```

Programa em C#:

```

/*
 * Exemplo 5a
 * Calcular o valor binario equivalente a um decimal.
 */
using System;

class Exemplo_5a
{
    //
    // parte principal
    //
    public static void Main ( )
    {
        // 1. definir dados
        int [ ] A = new int [ 10 ]      // tabela de algarismos
        int DECIMAL,                    // valor decimal
            N = 0,                       // numero de algarismos
            X,                            // indice
            QUOCIENTE;

        // 2. ler dado
        Console.Write ( "\nQual o valor decimal ? " );
        DECIMAL = int.Parse ( Console.ReadLine ( ) );

        // 3. calcular o binario equivalente
        // 3.1. copiar o valor decimal, para nao perder
        QUOCIENTE = DECIMAL;
        // 3.2. obter os restos da divisao por 2
        while ( QUOCIENTE > 0 )
        {
            // 3.2.1. guardar um resto de divisao por 2
            N = N + 1;
            A [ N ] = QUOCIENTE % 2;
            // 3.2.2. reduzir o quociente
            QUOCIENTE = QUOCIENTE / 2;
        } // fim while

        // 4. mostrar o resultado
        // 4.1. mostrar o valor decimal
        Console.WriteLine ( "\nDECIMAL = " + DECIMAL );
        // 4.2. mostrar o equivalente binario
        Console.Write ( "\nBINARIO = " );
        for ( X = N; X >= 1; X = X-1 )
        {
            // mostrar um algarismo
            Console.Write ( A [ X ] );
        } // fim for

        // pausa para terminar
        Console.Write ( "\nPressionar ENTER para terminar." );
        Console.ReadLine ( );
    } // end Main ( )
} // fim Exemplo_5a class

```

Programa em Java:

```

/**
 * Exemplo 5a
 * Calcular o valor binario equivalente a um decimal.
 */

// ----- classes necessarias

// ----- definicao de classe

class Exemplo_5a
{
//
// parte principal
//
    public static void main ( String [ ] args )
    {
        // 1. definir dados
        int [ ] A = new int [ 10 ];    // tabela de algarismos
        int DECIMAL,                  // valor decimal
            N = 0,                     // numero de algarismos
            X,                          // indice
            QUOCIENTE;

        // 2. ler dado
        System.out.print ( "\nQual o valor decimal ? " );
        DECIMAL = Integer.parseInt ( System.console( ).readLine( ) );

        // 3. calcular o binario equivalente
        // 3.1. copiar o valor decimal, para nao perder
        QUOCIENTE = DECIMAL;
        // 3.2. obter os restos da divisao por 2
        while ( QUOCIENTE > 0 )
        {
            // 3.2.1. guardar um resto de divisao por 2
            N = N + 1;
            A [ N ] = QUOCIENTE % 2;
            // 3.2.2. reduzir o quociente
            QUOCIENTE = QUOCIENTE / 2;
        } // fim while

        // 4. mostrar o resultado
        // 4.1. mostrar o valor decimal
        System.out.println ( "\nDECIMAL = " + DECIMAL );
        // 4.2. mostrar o equivalente binario
        System.out.print ( "\nBINARIO = " );
        for ( X = N; X >= 1; X = X-1 )
        {
            // mostrar um algarismo
            System.out.print ( A [ X ] );
        } // fim for

        // pausa para terminar
        System.out.print ( "\n\nPressionar ENTER para terminar." );
        System.console( ).readLine( );
    } // end main ( )

} // fim Exemplo_5a class

```

Outra versão com procedimentos e função.

Programa em C:

```
// Exemplo 5b.
// Calcular o valor binario equivalente de um decimal.
//
// bibliotecas necessarias
#include <stdio.h>
#include <stdlib.h>
//

// procedimento para mostrar dados
void MOSTRAR_BINARIO ( int ALGARISMO [ ], int N )
{
    // definir dado local
    int X;
    // mostrar dados
    for ( X = N; X >= 1; X = X - 1 )
    { // mostrar cada algarismo
        printf ( "\n%d  %d", X, ALGARISMO [ X ] );
    } // fiim for
} // fim do procedimento MOSTRAR_BINARIO ( )

// funcao para converter para BINARIO
void BINARIO ( int ALGARISMO [ ], int & N, int X )
{
    if ( X == 0 )
    {
        N = 1;
        ALGARISMO [ 1 ] = 0;
    }
    else
    {
        N = 0;
        while ( X > 0 )
        {
            // guardar um resto de divisao por 2
            N = N + 1;
            ALGARISMO [ N ] = X % 2;
            // reduzir o quociente
            X = X / 2;
        } // fim while
    } // fim if
} // fim do procedimento BINARIO ( )
```

```
//  
// parte principal  
//  
int main (void)  
{  
    // 1. definir dados  
    int DECIMAL,           // valor decimal  
        N,                 // número de algarismos  
        A [ 10 ];         // tabela de algarismos  
  
    // 2. ler dado  
    printf ( "\nQual o valor decimal ? " );  
    scanf ( "%d", &DECIMAL );  
  
    // 3. converter decimal para binario  
    BINARIO ( A, N, DECIMAL );  
  
    // 4. calcular e mostrar o resultado  
    printf ( "\nDECIMAL= %d", DECIMAL );  
    printf ( "\nBINÁRIO = " );  
    MOSTRAR_BINARIO ( A, N );  
  
    // pausa para terminar  
    printf ( "\nPressionar ENTER para terminar." );  
    getchar ( );  
    return ( 0 );  
} // fim do programa
```

Programa em C++:

```
// Exemplo 5b.
// Calcular o valor binario equivalente de um decimal.
//
// bibliotecas necessarias
#include <iostream>
using namespace std;
//

// procedimento para mostrar dados
void MOSTRAR_BINARIO ( int ALGARISMO [ ], int N )
{
    // definir dado local
    int X;
    // mostrar dados
    for ( X = N; X >= 1; X = X - 1 )
    { // mostrar cada algarismo
        cout << "\n" << X << " " << ALGARISMO [ X ];
    } // fiim for
} // fim do procedimento MOSTRAR_BINARIO ( )

// funcao para converter para BINARIO
void BINARIO ( int ALGARISMO [ ], int & N, int X )
{
    if ( X == 0 )
    {
        N = 1;
        ALGARISMO [ 1 ] = 0;
    }
    else
    {
        N = 0;
        while ( X > 0 )
        {
            // guardar um resto de divisao por 2
            N = N + 1;
            ALGARISMO [ N ] = X % 2;
            // reduzir o quociente
            X = X / 2;
        } // fim while
    } // fim if
} // fim do procedimento BINARIO ( )
```

```

//
// parte principal
//
int main (void)
{
// 1. definir dados
    int DECIMAL,           // valor decimal
        N,                 // número de algarismos
        A [ 10 ];          // tabela de algarismos

// 2. ler dado
    cout << "\nQual o valor decimal ? ";
    cin  >> DECIMAL;

// 3. converter decimal para binario
    BINARIO ( A, N, DECIMAL );

// 4. calcular e mostrar o resultado
    cout << "\nDECIMAL=" << DECIMAL;
    cout << "\nBINÁRIO = ";
    MOSTRAR_BINARIO ( A, N );

// pausa para terminar
    cout << "\nPressionar ENTER para terminar.";
    getchar ( );
    return EXIT_SUCCESS;
} // fim do programa

```

Outra versão do programa em C#:

```
/**
 * Exemplo 5b.
 * Calcular o valor binario equivalente a um decimal.
 */
using System;

class Exemplo_5b
{
    // definir dado global
    public static int N = 0; // numero de algarismos

    // procedimento para mostrar dados
    public static void MOSTRAR_BINARIO ( int [ ] ALGARISMO, int N )
    {
        // definir dado local
        int X;           // indice
        // mostrar dados
        for ( X = N; X >= 1; X = X-1 )
        {
            // mostrar cada algarismo
            Console.Write ( ALGARISMO [ X ] );
        } // fim for
    } // fim do procedimento MOSTRAR_BINARIO ( )

    // funcao para converter para binario
    public static void BINARIO ( int [ ] ALGARISMO, int X )
    {
        // definir dados locais
        int P = 1,           // potencia de X
            D = 0;           // valor decimal

        if ( X == 0 )
        {
            N = 1;
            ALGARISMO [ 1 ] = 0;
        }
        else
        {
            N = 0;
            while ( X > 0 )
            {
                // guardar um resto de divisao por 2
                N = N + 1;
                ALGARISMO [ N ] = X % 2;
                // reduzir o quociente
                X = X / 2;
            } // fim while
        } // fim if
    } // fim da funcao BINARIO ( )
}
```



```
//
// parte principal
//
public static void Main ( )
{
    // 1. definir dados
    int [ ] A = new int [ 10 ];    // tabela de algarismos
    int DECIMAL;                  // valor decimal

    // 2. ler dado
    DECIMAL = IO.readint ( "\nQual o valor decimal ? " );

    // 3. converter decimal para binario
    BINARIO ( A, DECIMAL );

    // 4. calcular e mostrar o resultado
    Console.WriteLine ( "\nDECIMAL = " + DECIMAL );
    Console.Write ( "\nBINARIO = " );
    MOSTRAR_BINARIO ( A, N );

    // pausa para terminar
    Console.Write ( "\nPressionar ENTER para terminar." );
    Console.ReadLine ( );
} // end Main ( )

} // fim Exemplo_5b class
```

Outra versão do programa em Java:

```

/**
 * Exemplo 5b.
 * Calcular o valor binario equivalente a um decimal.
 */

// ----- classes necessarias

// ----- definicao de classe

class Exemplo_5b
{
// definir dado global
    public static int N = 0;          // numero de algarismos

// procedimento para mostrar dados
    public static void MOSTRAR_BINARIO ( int [ ] ALGARISMO, int N )
    {
        // definir dado local
        int X;                        // indice
        // mostrar dados
        for ( X = N; X >= 1; X = X-1 )
        {
            // mostrar cada algarismo
            System.out.print ( ALGARISMO [ X ] );
        } // fim for
    } // fim do procedimento MOSTRAR_BINARIO ( )

// funcao para converter para binario
    public static void BINARIO ( int [ ] ALGARISMO, int X )
    {
        // definir dados locais
        int P = 1,                    // potencia de X
            D = 0;                     // valor decimal

        if ( X == 0 )
        {
            N = 1;
            ALGARISMO [ 1 ] = 0;
        }
        else
        {
            N = 0;
            while ( X > 0 )
            {
                // guardar um resto de divisao por 2
                N = N + 1;
                ALGARISMO [ N ] = X % 2;
                // reduzir o quociente
                X = X / 2;
            } // fim while
        } // fim if
    } // fim da funcao BINARIO ( )
}

```

```

//
// parte principal
//
public static void main ( String [ ] args )
{
// 1. definir dados
    int [ ] A = new int [ 10 ];      // tabela de algarismos
    int DECIMAL;                     // valor decimal

// 2. ler dado
    System.out.print ( "\nQual o valor decimal ? " );
    DECIMAL = Integer.parseInt ( System.console( ).readLine( ) );

// 3. converter decimal para binario
    BINARIO ( A, DECIMAL );

// 4. calcular e mostrar o resultado
    System.out.println ( "\nDECIMAL = " + DECIMAL );
    System.out.print ( "\nBINARIO = " );
    MOSTRAR_BINARIO ( A, N );

// pausa para terminar
    System.out.print ( "\n\nPressionar ENTER para terminar." );
    System.console( ).readLine( );
} // end main ( )

} // fim Exemplo_5b class

```

Programa em Python:

```
# Exemplo 5.
# Calcular o valor binario equivalente a um decimal.
#
from array import *;

# 1. definir dados
DECIMAL = 0; # valor decimal
N = 0;       # numero de algarismos
X = 0;       # indice
QUOCIENTE = 0; # quociente
            # tabela de algarismos
A = array ( 'i', [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ] );

# 2. ler dado
DECIMAL = int ( input ( "\nQual o valor decimal ? " ) );

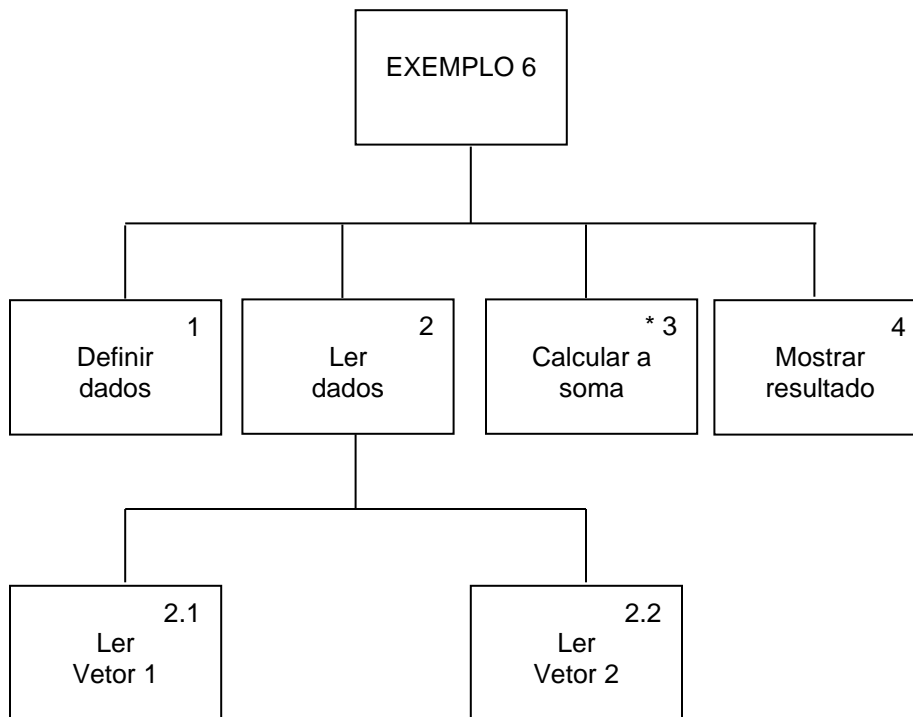
# 3. calcular o binario equivalente
# 3.1 copiar o valor decimal, para nao perder
QUOCIENTE = DECIMAL;
# 3.2 obter os restos da divisao por 2
while ( QUOCIENTE > 0 ):
# 3.2.1 guardar um resto de divisao por 2
    A [ N ] = QUOCIENTE % 2;
    N = N + 1;
# 3.2.2 reduzir o quociente
    QUOCIENTE = QUOCIENTE // 2;
# fim repetir

# 4. mostrar o resultado
# 4.1 mostrar o valor decimal
print ( "\nDECIMAL = ", DECIMAL, end=" " );
# 4.2 mostrar o equivalente binario
print ( "\nBINARIO = ", end=" " );
for X in range ( N, 0-1, -1 ):
# mostrar um algarismo
    print ( " ", A [ X ], " ", end=" " );
# fim repetir
print ( );
# pausa para terminar
print ( "\nPressionar ENTER para terminar." );
input ( );
# fim do programa
```

## Exemplo 6.

Dadas as coordenadas de dois vetores tridimensionais, calcular o vetor resultante da soma dos dois primeiros.

Diagrama funcional:



Análise de dados :

- Dados do problema :

Dado	Tipo	Valor inicial	Função
V1	real (10)	-	armazenar até 10 coeficientes
V2	real (10)	-	armazenar até 10 coeficientes
I	inteiro	-	índice

- Resultados do problema :

Dado	Tipo	Valor inicial	Função
V3	real (10)	-	armazenar até 10 coeficientes

- Avaliação da solução :

$$V1 = 1x - 2y + 1z = [1 \ -2 \ 1]$$

$$V2 = 0x + 2y + 1z = [0 \ 2 \ 1]$$

$$V3 = 1x + 0y + 2z = [1 \ 0 \ 2]$$

Índice	Vetor 1	Vetor 2	Resultado
0	1	1	2
1	-2	2	0
2	1	0	1

Algoritmo:

Esboço:

Primeira versão, só comentários.

Exemplo 6	v.1
Ação	Bloco
! definir dados	1
! ler dados	2
! ler o primeiro vetor	2.1
! ler o segundo vetor	2.2
! calcular a soma dos dois vetores	3
! mostrar o resultado	4

Segunda versão, com refinamento do primeiro bloco.

Exemplo 6	v.2
Ação	Bloco
! definir dados	1
inteiro X;           ! índice real V1 [ 10 ], ! primeiro vetor V2 [ 10 ], ! segundo vetor V3 [ 10 ]; ! soma de dois vetores	
! ler o primeiro vetor	2.1
! ler o segundo vetor	2.2
! calcular a soma dos dois vetores	3
! mostrar o resultado	4

Terceira versão, com refinamento do segundo e do quarto bloco.

Exemplo 6	v.3
Ação	Bloco
! definir dados	1
inteiro X;           ! índice real V1 [ 10 ], ! primeiro vetor V2 [ 10 ], ! segundo vetor V3 [ 10 ]; ! soma de dois vetores	
! ler dados	2
! ler o primeiro vetor	2.1
X = 1 : 3 : 1	2.1.1
! ler uma coordenada	
tela ← ("V1(", X, ")= ");	
V1 [ X ] ← teclado;	
! ler o segundo vetor	2.2
X = 1 : 3 : 1	2.2.1
! ler uma coordenada	
tela ← ("V2(", X, ")= ");	
V2 [ X ] ← teclado;	
! calcular a soma dos dois vetores	3
! mostrar o resultado	4
tela ← "V3= [ ";	
X = 1 : 3 : 1	4.1
! ler uma coordenada	
tela ← (V3 [ X ], " ");	
tela ← "]"	

Quarta versão, com refinamento do terceiro bloco.

Exemplo 6	v.4
Ação	Bloco
! definir dados	1
inteiro X; ! índice real V1 [ 10 ], ! primeiro vetor V2 [ 10 ], ! segundo vetor V3 [ 10 ]; ! soma de dois vetores	
! ler dados	2
! ler o primeiro vetor	2.1
X = 1 : 3 : 1	2.1.1
! ler uma coordenada	
tela ← (“\nV1(“ , X , “)= “); V1 [ X ] ← teclado;	
! ler o segundo vetor	2.2
X = 1 : 3 : 1	2.2.1
! ler uma coordenada	
tela ← (“\nV2(“ , X , “)= “); V2 [ X ] ← teclado;	
! calcular a soma dos dois vetores	3
X = 1 : 3 : 1	3.1.1
! somar termo a termo	
V3 [ X ] = V1 [ X ] + V2 [ X ];	
! mostrar o resultado	4
tela ← “\nV3= [ “;	
X = 1 : 3 : 1	4.1
! ler uma coordenada	
tela ← (V3 [ X ], “ “);	
tela ← “]”;	



Programa em SCILAB:

```
// Exemplo 6.
// Calcular a soma de dois vetores tridimensionais.
//
// 1. definir dados
X = 0;           // indice
V1 ( 1:10 ) = 0; // primeiro vetor
V2 ( 1:10 ) = 0; // segundo vetor
V3 ( 1:10 ) = 0; // soma de dois vetores

// 2. ler dados
// 2.1 ler o primeiro vetor
for ( X = 1 : 1 : 3 )
// 2.1.1 ler uma coordenada
    printf ( "\nV1 ( %d ) = ", X );
    V1 ( X ) = input ( ' ' );
end // for
// 2.2 ler o segundo vetor
for ( X = 1 : 1 : 3 )
// 2.2.1 ler uma coordenada
    printf ( "\nV2 ( %d ) = ", X );
    V2 ( X ) = input ( ' ' );
end // for

// 3. calcular a soma dos dois vetores
for ( X = 1 : 1 : 3 )
// 3.1somar termo a termo
    V3 ( X ) = V1 ( X ) + V2 ( X );
end // for

// 4. mostrar o resultado
printf ( "\nV3 = [ " );
for ( X = 1 : 1 : 3 )
// 4.1 ler uma coordenada
    printf ( " %d ", V3 ( X ) );
end // for
printf ( " ]\n" );

// pausa para terminar
printf ( "\nPressionar ENTER para terminar." );
halt;
// fim do programa
```

Programa em C:

```
// Exemplo 6a.
// Calcular a soma de dois vetores tridimensionais.
//
// bibliotecas necessarias
#include <stdio.h>
#include <stdlib.h>
//
// parte principal
//
int main (void)
{
    // 1. definir dados
    int   X;           // indice
    float V1 [ 10 ],   // primeiro vetor
          V2 [ 10 ],   // segundo vetor
          V3 [ 10 ];   // soma de dois vetores

    // 2. ler dados
    // 2.1. ler o primeiro vetor
    for ( X = 0; X < 3; X = X + 1 )
    {
        // 2.1.1. ler uma coordenada
        printf ( "\nV1[%d] = ", X );
        scanf ( "%f", &V1 [ X ] );
    } // fim for
    // 2.2. ler o segundo vetor
    for ( X = 0; X < 3; X = X + 1 )
    {
        // 2.2.1. ler uma coordenada
        printf ( "\nV2[%d] = ", X );
        scanf ( "%f", &V2 [ X ] );
    } // fim for

    // 3. calcular a soma dos dois vetores
    for ( X = 0; X < 3; X = X + 1 )
    {
        // 3.1. somar termo a termo
        V3 [ X ] = V1 [ X ] + V2 [ X ];
    } // fim for

    // 4. mostrar o resultado
    printf ( "\nV3= [ ";
    for ( X = 0; X < 3; X = X + 1 )
    {
        // 4.1. ler uma coordenada
        printf ( "%f ", V3 [ X ] );
    } // fim for
    printf ( "]" );

    // pausa para terminar
    printf ( "Pressionar ENTER para terminar." );
    getchar ( );
    return ( 0 );
} // fim do programa
```

Programa em C++:

```
// Exemplo 6a.
// Calcular a soma de dois vetores tridimensionais.
//
// bibliotecas necessarias
#include <iostream>
using namespace std;
//
// parte principal
//
int main (void)
{
    // 1. definir dados
    int   X;           // indice
    float V1 [ 10 ],   // primeiro vetor
          V2 [ 10 ],   // segundo vetor
          V3 [ 10 ];   // soma de dois vetores

    // 2. ler dados
    // 2.1. ler o primeiro vetor
    for ( X = 0; X < 3; X = X + 1 )
    {
        // 2.1.1. ler uma coordenada
        cout << "\nV1[" << X << "] = ";
        cin  >> V1 [ X ];
    } // fim for
    // 2.2. ler o segundo vetor
    for ( X = 0; X < 3; X = X + 1 )
    {
        // 2.2.1. ler uma coordenada
        cout << "\nV2[" << X << "] = ";
        cin  >> V2 [ X ];
    } // fim for

    // 3. calcular a soma dos dois vetores
    for ( X = 0; X < 3; X = X + 1 )
    {
        // 3.1. somar termo a termo
        V3 [ X ] = V1 [ X ] + V2 [ X ];
    } // fim for

    // 4. mostrar o resultado
    cout << "\nV3 = [ ";
    for ( X = 0; X < 3; X = X + 1 )
    {
        // 4.1. ler uma coordenada
        cout << V3 [ X ] << " ";
    } // fim for
    cout << "]" ;

    // pausa para terminar
    cout << "Pressionar ENTER para terminar.";
    getchar ( );
    return EXIT_SUCCESS;
} // fim do programa
```

Programa em C#:

```

/*
 * Exemplo 6a
 * Calcular a soma de dois vetores tridimensionais.
 */
using System;

class Exemplo_6a
{
    //
    // parte principal
    //
    public static void Main ( )
    {
        // 1. definir dados
        int    X;           // indice
        double [ ] V1 = new double [ 10 ], // primeiro vetor
              V2 = new double [ 10 ], // segundo vetor
              V3 = new double [ 10 ]; // soma de dois vetores
        // 2. ler dados
        // 2.1. ler o primeiro vetor
        for ( X = 0; X < 3; X = X+1 )
        {
            // 2.1.1. ler uma coordenada
            Console.Write ( "\nV1[" + X + "] = " );
            V1 [ X ] = double.Parse ( Console.ReadLine ( ) );
        } // fim for
        // 2.2. ler o segundo vetor
        for ( X = 0; X < 3; X = X+1 )
        {
            // 2.1.2. ler uma coordenada
            Console.Write ( "\nV2[" + X + "] = " );
            V2 [ X ] = double.Parse ( Console.ReadLine ( ) );
        } // fim for
        // 3. calcular a soma de dois vetores
        for ( X = 0; X < 3; X = X+1 )
        {
            // 3.1. somar termo a termo
            V3 [ X ] = V1 [ X ] + V2 [ X ];
        } // fim for
        // 4. mostrar resultado
        Console.Write ( "\nV3[ " );
        for ( X = 0; X < 3; X = X+1 )
        {
            // 4.1. mostrar uma coordenada
            Console.Write ( V3 [ X ] + " " );
        } // fim for
        Console.Write ( "]" );
        // pausa para terminar
        Console.Write ( "\nPressionar ENTER para terminar." );
        Console.ReadLine ( );
    } // end Main ( )
} // fim Exemplo_6a class

```

Programa em Java:

```

/**
 * Exemplo 6a
 * Calcular a soma de dois vetores tridimensionais.
 */

// ----- classes necessarias

// ----- definicao de classe

class Exemplo_6a
{
//
// parte principal
//
    public static void main ( String [ ] args )
    {
        // 1. definir dados
        int    X;           // indice
        double [ ] V1 = new double [ 10 ], // primeiro vetor
               V2 = new double [ 10 ],    // segundo  vetor
               V3 = new double [ 10 ];    // soma de dois vetores

        // 2. ler dados
        // 2.1. ler o primeiro vetor
        for ( X = 0; X < 3; X = X+1 )
        {
            // 2.1.1. ler uma coordenada
            System.out.print ( "\nV1[" + X + "]= " );
            V1 [ X ] = Double.parseDouble ( System.console( ).readLine( ) );
        } // fim for
        // 2.2. ler o segundo  vetor
        for ( X = 0; X < 3; X = X+1 )
        {
            // 2.1.2. ler uma coordenada
            System.out.print ( "\nV2[" + X + "]= " );
            V2 [ X ] = Double.parseDouble ( System.console( ).readLine( ) );
        } // fim for

        // 3. calcular a soma de dois vetores
        for ( X = 0; X < 3; X = X+1 )
        {
            // 3.1. somar termo a termo
            V3 [ X ] = V1 [ X ] + V2 [ X ];
        } // fim for
    }
}

```

```
// 4. mostrar resultado
System.out.print ( "\nV3[ " );
for ( X = 0; X < 3; X = X+1 )
{
    // 4.1. mostrar uma coordenada
    System.out.print ( V3 [ X ] + " " );
} // fim for
System.out.print ( "]" );

// pausa para terminar
System.out.print ( "\n\nPressionar ENTER para terminar." );
System.console( ).readLine( );
} // end main ( )
} // fim Exemplo_6a class
```

Outra versão com procedimentos.

Programa em C:

```
// Exemplo 6b.
// Calcular a soma de dois vetores tridimensionais.
//
// bibliotecas necessarias
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
//
// definir tipo
typedef float VETOR [ 10 ];

// procedimento para ler dados
void LER ( VETOR V, int & N )
{
    // definir dado local
    int X;
    // ler dados
    printf ( "\nQual o numero de elementos ? " );
    scanf ( "%d", &N );
    for ( X = 0; X < N; X = X + 1 )
    { // ler e guardar o valor de um elemento
        printf ( "\nQual o valor do elemento %d ? ", X );
        scanf ( "%f", &V [ X ] );
    } // fim for
} // fim do procedimento LER ( )

// procedimento para mostrar dados
void MOSTRAR ( VETOR V, int N )
{
    // definir dado local
    int X;
    // mostrar dados
    printf ( " [ " );
    for ( X = 0; X < N; X = X + 1 )
    {
        printf ( "%f ", V [ X ] );
    } // fim for
    printf ( " ] " );
} // fim do procedimento MOSTRAR ( )

// procedimento para somar vetores
void SOMAR ( VETOR V3, VETOR V1, VETOR V2, int N )
{
    // definir dado local
    int X;
    // somar termos
    for ( X = 0; X <= N; X = X + 1 )
    { // somar termo a termo
        V3 [ X ] = V1 [ X ] + V2 [ X ];
    } // fiim for
} // fim do procedimento SOMAR ( )
```

```
//
// parte principal
//
int main (void)
{
    // 1. definir dados
    int      X,      // indice
            N;      // tamanho do vetor
    VETOR V1,      // primeiro vetor
            V2,      // segundo vetor
            V3;      // soma de dois vetores

    // 2. ler dados
    // 2.1 ler o primeiro vetor
    LER ( V1, N );
    // 2.2 ler o segundo vetor
    LER ( V2, N );

    // 3. calcular a soma dos dois vetores
    SOMAR ( V3, V1, V2, N );

    // 4. mostrar o resultado
    printf ( "\nV3 = " );
    MOSTRAR ( V3, N );

    // pausa para terminar
    printf ( "Pressionar ENTER para terminar." );
    getchar ( );
    return ( 0 );
} // fim do programa
```



Programa em C++:

```
// Exemplo 6b.
// Calcular a soma de dois vetores tridimensionais.
//
// bibliotecas necessarias
#include <iostream>
using namespace std;
#include <math.h>
//
// definir tipo
typedef float VETOR [ 10 ];

// procedimento para ler dados
void LER ( VETOR V, int & N )
{
    // definir dado local
    int X;
    // ler dados
    cout << "\nQual o numero de elementos ? ";
    cin  >> N;
    for ( X = 0; X < N; X = X + 1 )
    { // ler e guardar o valor de um elemento
        cout << "\nQual o valor do elemento " << X << " ? ";
        cin  >> V [ X ];
    } // fim for
} // fim do procedimento LER ( )

// procedimento para mostrar dados
void MOSTRAR ( VETOR V, int N )
{
    // definir dado local
    int X;
    // mostrar dados
    cout << " [ ";
    for ( X = 0; X < N; X = X + 1 )
    {
        cout << V [ X ] << " ";
    } // fim for
    cout << "] ";
} // fim do procedimento MOSTRAR ( )

// procedimento para somar vetores
void SOMAR ( VETOR V3, VETOR V1, VETOR V2, int N )
{
    // definir dado local
    int X;
    // somar termos
    for ( X = 0; X <= N; X = X + 1 )
    { // somar termo a termo
        V3 [ X ] = V1 [ X ] + V2 [ X ];
    } // fiim for
} // fim do procedimento SOMAR ( )
```

```
//
// parte principal
//
int main (void)
{
    // 1. definir dados
    int      X,      // indice
            N;      // tamanho do vetor
    VETOR V1,      // primeiro vetor
            V2,      // segundo vetor
            V3;      // soma de dois vetores

    // 2. ler dados
    // 2.1 ler o primeiro vetor
    LER ( V1, N );
    // 2.2 ler o segundo vetor
    LER ( V2, N );

    // 3. calcular a soma dos dois vetores
    SOMAR ( V3, V1, V2, N );

    // 4. mostrar o resultado
    cout << "\nV3 = " ;
    MOSTRAR ( V3, N );

    // pausa para terminar
    cout << "Pressionar ENTER para terminar.";
    cin.get ( );
    return EXIT_SUCCESS;
} // fim do programa
```

Outra versão do programa em C#:

```

/*
 * Exemplo 6b
 * Calcular a soma de dois vetores tridimensionais.
 */
using System;

class Exemplo_6b
{
    // definir dado global
    static int N = 0;          // numero de Algarismos

    // procedimento para ler dados
    public static void LER ( double [ ] V )
    {
        // definir dado local
        int X;                 // indice
        // ler dados
        Console.Write ( "\nQual o numero de elementos ? " );
        N = int.Parse ( Console.ReadLine ( ) );
        for ( X = 0; X < N; X = X+1 )
        { // ler e guardar o valor de um elemento
            Console.Write ( "\nQual o valor do elemento " + X + " ? " );
            V [ X ] = double.Parse ( Console.ReadLine ( ) );
        } // fim for
    } // fim do procedimento LER ( )

    // procedimento para mostrar dados
    public static void MOSTRAR ( double [ ] V, int N )
    {
        // definir dado local
        int X;                 // indice
        // mostrar dados
        Console.Write ( " [ " );
        for ( X = 0; X < N; X = X+1 )
        { // mostrar uma coordenada
            Console.Write ( V [ X ] + " " );
        } // fim for
        Console.Write ( "]" );
    } // fim do procedimento MOSTRAR ( )

    // procedimento para somar vetores
    public static void SOMAR ( double [ ] V3, double [ ] V1, double [ ] V2, int N )
    {
        // definir dado local
        int X;                 // indice
        // 3. calcular a soma dos dois vetores
        for ( X = 0; X < N; X = X+1 )
        { // 3.1. somar termo a termo
            V3 [ X ] = V1 [ X ] + V2 [ X ];
        } // fim for
    } // fim do procedimento MOSTRAR ( )
}

```

```

//
// parte principal
//
public static void Main ( )
{
    // 1. definir dados
    int      X;                // indice
    double [ ] V1 = new double [ 10 ], // primeiro vetor
              V2 = new double [ 10 ], // segundo  vetor
              V3 = new double [ 10 ]; // soma de dois vetores

    // 2. ler dados
    // 2.1. ler o primeiro vetor
    LER ( V1 );
    // 2.2. ler o segundo  vetor
    LER ( V2 );

    // 3. calcular a soma de dois vetores
    SOMAR ( V3, V1, V2, N );

    // 4. mostrar resultado
    Console.Write ( "\nV3 = " );
    MOSTRAR ( V3, N );

    // pausa para terminar
    Console.Write ( "\nPressionar ENTER para terminar." );
    Console.ReadLine ( );
} // end Main ( )

} // fim Exemplo_6b class

```

Outra versão do programa em Java:

```

/**
 * Exemplo 6b
 * Calcular a soma de dois vetores tridimensionais.
 */

// ----- classes necessarias

// ----- definicao de classe

class Exemplo_6b
{
// definir dado global
    public static int N = 0;          // numero de Algarismos

// procedimento para ler dados
    public static void LER ( double [ ] V )
    {
        // definir dado local
        int X;                        // indice
        // ler dados
        System.out.print ( "\nQual o numero de elementos ? " );
        N = Integer.parseInt ( System.console( ).readLine( ) );
        for ( X = 0; X < N; X = X+1 )
        { // ler e guardar o valor de um elemento
            System.out.print ( "\nQual o valor do elemento " + X + " ? " );
            V [ X ] = Double.parseDouble ( System.console( ).readLine( ) );
        } // fim for
    } // fim do procedimento LER ( )

// procedimento para mostrar dados
    public static void MOSTRAR ( double [ ] V, int N )
    {
        // definir dado local
        int X;                        // indice
        // mostrar dados
        System.out.print ( " [ " );
        for ( X = 0; X < N; X = X+1 )
        { // mostrar uma coordenada
            System.out.print ( V [ X ] + " " );
        } // fim for
        System.out.print ( "]" );
    } // fim do procedimento MOSTRAR ( )
}

```

```

// procedimento para somar vetores
public static void SOMAR ( double [ ] V3, double [ ] V1, double [ ] V2, int N )
{
    // definir dado local
    int X;                                // indice
    // 3. calcular a soma dos dois vetores
    for ( X = 0; X < N; X = X+1 )
    { // 3.1. somar termo a termo
        V3 [ X ] = V1 [ X ] + V2 [ X ];
    } // fim for
} // fim do procedimento MOSTRAR ( )

//
// parte principal
//
public static void main ( String [ ] args )
{
    // 1. definir dados
    int X;                                // indice
    double [ ] V1 = new double [ 10 ],    // primeiro vetor
    V2 = new double [ 10 ],              // segundo vetor
    V3 = new double [ 10 ];              // soma de dois vetores

    // 2. ler dados
    // 2.1. ler o primeiro vetor
    LER ( V1 );
    // 2.2. ler o segundo vetor
    LER ( V2 );

    // 3. calcular a soma de dois vetores
    SOMAR ( V3, V1, V2, N );

    // 4. mostrar resultado
    System.out.print ( "\nV3 = " );
    MOSTRAR ( V3, N );

    // pausa para terminar
    System.out.print ( "\n\nPressionar ENTER para terminar." );
    System.console( ).readLine( );
} // end main ( )

} // fim Exemplo_6b class

```

Programa em Python:

```
# Exemplo 6.
# Calcular a soma de dois vetores tridimensionais.
#
from array import *;

# 1. definir dados
X = 0;      # indice
            # primeiro vetor
V1 = array ( 'f', [ 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 ] );
            # segundo vetor
V2 = array ( 'f', [ 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 ] );
            # soma de dois vetores
V3 = array ( 'f', [ 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 ] );

# 2. ler dados
# 2.1 ler o primeiro vetor
for X in range ( 0, 3, 1 ):
# 2.1.1 ler uma coordenada
    print ( "\nV1 [ ", X, " ] = ", end=" " );
    V1 [ X ] = float ( input ( " " ) );
# fim repetir
# 2.2 ler o segundo vetor
for X in range ( 0, 3, 1 ):
# 2.2.1 ler uma coordenada
    print ( "\nV2 [ ", X, " ] = ", end=" " );
    V2 [ X ] = float ( input ( " " ) );
# fim repetir

# 3. calcular a soma dos dois vetores
for X in range ( 0, 3, 1 ):
# 3.1 somar termo a termo
    V3 [ X ] = V1 [ X ] + V2 [ X ];
# fim repetir

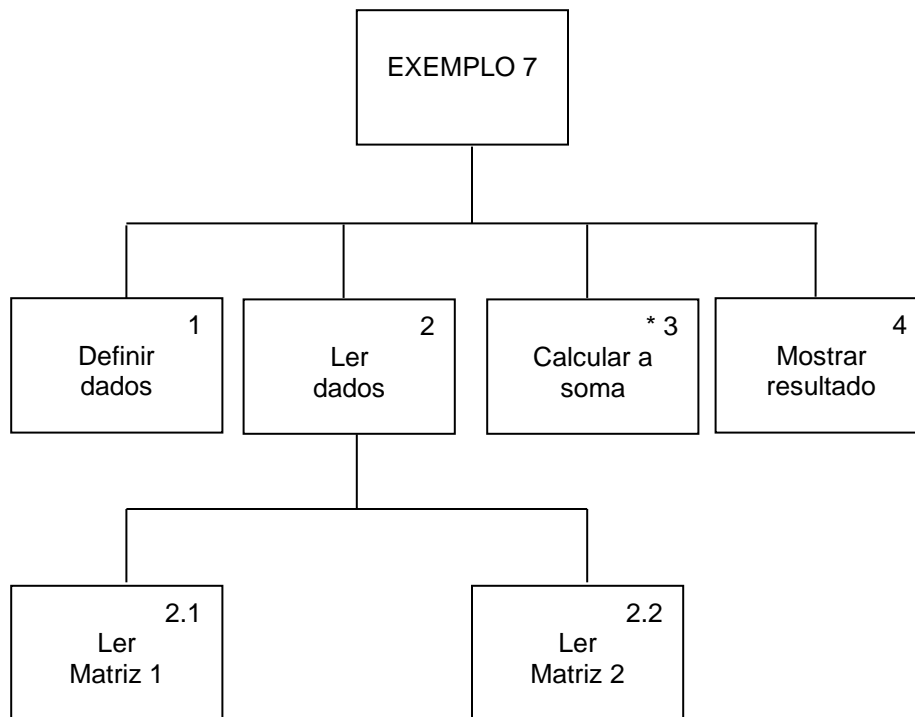
# 4. mostrar o resultado
print ( "\nV3 = [", end=" " );
for X in range ( 0, 3, 1 ):
# 4.1 ler uma coordenada
    print ( " ", V3 [ X ], end=" " );
# fim repetir
print ( " ]\n" );

# pausa para terminar
print ( "\nPressionar ENTER para terminar." );
input ( );
# fim do programa
```

## Exemplo 7.

Dados os elementos de duas matrizes de dimensões  $M \times N$ , calcular a matriz resultante da soma das duas primeiras.

Diagrama funcional:





Análise de dados :

- Dados do problema :

Dado	Tipo	Valor inicial	Função
M1	real (10,10)	-	armazenar até 10x10 elementos
M2	real (10,10)	-	armazenar até 10x10 elementos
M	inteiro	-	número de linhas
N	inteiro	-	número de colunas
X	inteiro	-	índice para linha
Y	inteiro	-	índice para coluna

- Resultados do problema :

Dado	Tipo	Valor inicial	Função
M3	real (10,10)	-	armazenar até 10x10 elementos

- Avaliação da solução :

$$M1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad M2 = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \quad M3 = M1 + M2 = \begin{bmatrix} 2 & 2 & 3 \\ 4 & 6 & 6 \\ 7 & 8 & 10 \end{bmatrix}$$

Algoritmo:

Esboço:

Primeira versão, só comentários.

Exemplo 7	v.1
Ação	Bloco
! definir dados	1
! ler dados	2
! ler o número de linhas	2.1
! ler o número de colunas	2.2
! ler a primeira matriz	2.1
! ler a segunda matriz	2.2
! calcular a soma de duas matrizes	3
! mostrar o resultado	4

Segunda versão, com refinamento do primeiro bloco.

Exemplo 7	v.2
Ação	Bloco
! definir dados	1
inteiro M, N;                   ! número de linhas e colunas inteiro X, Y;                   ! índices real   M1 [ 10 ] [ 10 ], ! primeira matriz M2 [ 10 ] [ 10 ], ! segunda matriz M3 [ 10 ] [ 10 ]; ! soma de duas matrizes	
! ler dados	
! ler o número de linhas	2.1
! ler o número de colunas	2.2
! ler a primeira matriz	2.3
! ler a segunda matriz	2.4
! calcular a soma de duas matrizes	3
! mostrar o resultado	4

Terceira versão, com refinamento do segundo e do quarto bloco.

Exemplo 7	v.3
Ação	Bloco
! definir dados	1
inteiro M, N;                   ! número de linhas e colunas inteiro X, Y;                   ! índices real   M1 [ 10 ] [ 10 ], ! primeira matriz M2 [ 10 ] [ 10 ], ! segunda matriz M3 [ 10 ] [ 10 ]; ! soma de duas matrizes	
! ler dados	2
! ler o número de linhas	2.1
tela ← “\nNúmero de linhas (M) ? “; M ← teclado;	
! ler o número de colunas	2.2
tela ← “\nNúmero de colunas (N) ? “; N ← teclado;	
! ler a primeira matriz	2.3
X = 1 : M : 1	2.3.1
Y = 1 : N : 1	
! ler um elemento	
tela ← ( “\nM1[“, X , “,”, Y , “]= “ );	
M1 [ X ] [ Y ] ← teclado;	
! ler a segunda matriz	2.4
X = 1 : M : 1	2.4.1
Y = 1 : N : 1	
! ler um elemento	
tela ← ( “\nM2[“, X , “,”, Y , “]= “ );	
M2 [ X ] [ Y ] ← teclado;	
! calcular a soma dos dois vetores	3
! mostrar o resultado	4
X = 1 : M : 1	4.1
Y = 1 : N : 1	
! mostrar um elemento	
tela ← ( M3 [ X ] [ Y ], “ “ );	
tela ← “\n”;	

Quarta versão, com refinamento do terceiro e quarto blocos.

Exemplo 7	v.3
Ação	Bloco
! definir dados	1
inteiro M, N;                   ! número de linhas e colunas inteiro X, Y;                   ! índices real M1 [ 10 ] [ 10 ], ! primeira matriz M2 [ 10 ] [ 10 ], ! segunda matriz M3 [ 10 ] [ 10 ]; ! soma de duas matrizes	
! ler dados	2
! ler o número de linhas	2.1
tela ← “\nNúmero de linhas (M) ? “; M ← teclado;	
! ler o número de colunas	2.2
tela ← “\nNúmero de colunas (N) ? “; N ← teclado;	
! ler a primeira matriz	2.3
X = 1 : M : 1	2.3.1
Y = 1 : N : 1	
! ler um elemento	
tela ← (“\nM1[“, X, “”, Y, “]= “); M1 [ X ] [ Y ] ← teclado;	
! ler a segunda matriz	2.4
X = 1 : M : 1	2.4.1
Y = 1 : N : 1	
! ler um elemento	
tela ← (“\nM2[“, X, “”, Y, “]= “); M2 [ X ] [ Y ] ← teclado;	
! calcular a soma dos dois vetores	3
X = 1 : M : 1	
Y = 1 : N : 1	
! somar cada elemento	3.1
M3 [ X ] [ Y ] = M1 [ X ] [ Y ] + M2 [ X ] [ Y ];	
! mostrar o resultado	4
tela ← “\nMatriz resultante:\n”;	
X = 1 : M : 1	
Y = 1 : N : 1	
! mostrar um elemento de cada vez	4.1
tela ← M3 [ X ] [ Y ] << “ “;	
tela ← “\n”;	

Programa em SCILAB:

```
// Exemplo 7.
// Calcular a soma de duas matrizes MxN.
//
// 1. definir dados
M = 0;           // numero de linhas
N = 0;           // numero de colunas
X = 0;           // indice
Y = 0;           // indice
M1 = zeros ( 10, 10 ); // primeira matriz
M2 ( 1:10, 1:10 ) = 0; // segunda matriz
M3 ( 1:10, 1:10 ) = 0; // soma de duas matrizes
// 2. ler dados
// 2.1 ler o numero de linhas
M = input ( "\nNumero de linhas ? " );
// 2.2 ler o numero de colunas
N = input ( "\nNumero de colunas ? " );
// 2.3 ler a primeira matriz
for ( X = 1 : 1 : M )
    for ( Y = 1 : 1 : N )
        // 2.3.1 ler um elemento
        printf ( "\nM1[ %d , %d ] = %f ", X, Y );
        M1 ( X, Y ) = input ( " " );
    end // for
end // for
// 2.4 ler a segunda matriz
for ( X = 1 : 1 : M )
    for ( Y = 1 : 1 : N )
        // 2.3.1 ler um elemento
        printf ( "\nM2[ %d , %d ] = %f ", X, Y );
        M2 ( X, Y ) = input ( " " );
    end // for
end // for
// 3. calcular a soma de duas matrizes
for ( X = 1 : 1 : M )
    for ( Y = 1 : 1 : N )
        // 3.1 somar cada elemento
        M3 ( X, Y ) = M1 ( X, Y ) + M2 ( X, Y );
    end // for
end // for
// 4. mostrar o resultado
printf ( "\nMatriz resultante:\n" );
for ( X = 1 : 1 : M )
    for ( Y = 1 : 1 : N )
        // 4.1 mostrar um elemento de cada vez
        printf ( " %f ", M3 ( X, Y ) );
    end // for
    printf ( " \n " );
end // for

// pausa para terminar
printf ( "\nPressionar ENTER para terminar." );
halt;
// fim do programa
```

Programa em C:

```
// Exemplo 7a.
// Calcular a soma de duas matrizes MxN.
//
// bibliotecas necessarias
#include <stdio.h>
#include <stdlib.h>
//
// parte principal
//
int main (void)
{
// 1. definir dados
    int  M, N;           // número de linhas e colunas
    int  X, Y;           // indices
    float M1 [ 10 ] [ 10 ], // primeira matriz
          M2 [ 10 ] [ 10 ], // segunda matriz
          M3 [ 10 ] [ 10 ]; // soma de duas matrizes
// 2. ler dados
// 2.1 ler o numero de linhas
    printf ( "\nNumero de linhas ? " ); scanf ( "%d", &M );
// 2.2 ler o numero de colunas
    printf ( "\nNumero de colunas ? " ); scanf ( "%d", &N );
// 2.3 ler a primeira matriz
    for ( X = 0; X < M; X = X + 1 )
        for ( Y = 0; Y < N; Y = Y + 1 )
        { // 2.3 1 ler um elemento
            printf ( "\nM1[%d,%d] = ", X, Y ); scanf ( "%f", &M1 [ X ] [ Y ] );
        } // fim for
// 2.4 ler a segunda matriz
    for ( X = 0; X < M; X = X + 1 )
        for ( Y = 0; Y < N; Y = Y + 1 )
        { // 2.4 1 ler um elemento
            printf ( "\nM2[%d,%d] = ", X, Y ); scanf ( "%f", &M2 [ X ] [ Y ] );
        } // fim for
// 3. calcular a soma de duas matrizes
    for ( X = 0; X < M; X = X + 1 )
        for ( Y = 0; Y < N; Y = Y + 1 )
        { // 3.1 somar cada elemento
            M3 [ X ] [ Y ] = M1 [ X ] [ Y ] + M2 [ X ] [ Y ];
// 4. mostrar o resultado
            printf ( "\nMatriz resultante:\n" );
            for ( X = 0; X < M; X = X + 1 )
            {
                for ( Y = 0; Y < N; Y = Y + 1 )
                { // 4.1 mostrar um elemento de cada vez
                    printf ( "%ft", M3 [ X ] [ Y ] );
                } // fim for
                printf ( "\n" );
            } // fim for
// pausa para terminar
            printf ( "Pressionar ENTER para terminar." );
            getchar ( );
            return ( 0 );
        } // fim do programa
}
```

Programa em C++:

```
// Exemplo 7a.
// Calcular a soma de duas matrizes MxN.
//
// bibliotecas necessarias
#include <iostream>
using namespace std;
//
// parte principal
//
int main (void)
{
// 1. definir dados
    int  M, N;           // número de linhas e colunas
    int  X, Y;           // indices
    float M1 [ 10 ] [ 10 ], // primeira matriz
          M2 [ 10 ] [ 10 ], // segunda matriz
          M3 [ 10 ] [ 10 ]; // soma de duas matrizes
// 2. ler dados
// 2.1 ler o numero de linhas
    cout << "\nNumero de linhas ? "; cin >> M;
// 2.2 ler o numero de colunas
    cout << "\nNumero de colunas ? "; cin >> N;
// 2.3 ler a primeira matriz
    for ( X = 0; X < M; X = X + 1 )
        for ( Y = 0; Y < N; Y = Y + 1 )
        { // 2.3 1 ler um elemento
            cout << "\nM1[" << X << ", " << Y << "] = "; cin >> M1 [ X ] [ Y ];
        } // fim for
// 2.4 ler a segunda matriz
    for ( X = 0; X < M; X = X + 1 )
        for ( Y = 0; Y < N; Y = Y + 1 )
        { // 2.4 1 ler um elemento
            cout << "\nM2[" << X << ", " << Y << "] = "; cin >> M2 [ X ] [ Y ];
        } // fim for
// 3. calcular a soma de duas matrizes
    for ( X = 0; X < M; X = X + 1 )
        for ( Y = 0; Y < N; Y = Y + 1 )
        { // 3.1 somar cada elemento
            M3 [ X ] [ Y ] = M1 [ X ] [ Y ] + M2 [ X ] [ Y ];
// 4. mostrar o resultado
            cout << "\nMatriz resultante:\n";
            for ( X = 0; X < M; X = X + 1 )
            {
                for ( Y = 0; Y < N; Y = Y + 1 )
                { // 4.1 mostrar um elemento de cada vez
                    cout << M3 [ X ] [ Y ] << "\t";
                } // fim for
                cout << "\n";
            } // fim for
// pausa para terminar
            cout << "Pressionar ENTER para terminar.";
            getch ( );
            return EXIT_SUCCESS;
        } // fim do programa
}
```

Programa em C#:

```

/*
 * Exemplo 7a
 * Calcular a soma de duas matrizes MxN.
 */
using System

class Exemplo_7a
{
//
// parte principal
//
public static void Main ( )
{
// 1. definir dados
int      M, N,                // numero de linhas e colunas
          X, Y;               // indices

double [ , ] M1 = new double [ 10, 10 ], // primeiro matriz
           M2 = new double [ 10, 10 ],    // segundo  matriz
           M3 = new double [ 10, 10 ];    // soma de duas matrizes

// 2. ler dados
// 2.1 ler o numero de linhas
Console.Write ( "\nNumero de linhas ? " );
M = int.Parse ( Console.ReadLine ( ) );
// 2.2 ler o numero de colunas
Console.Write ( "\nNumero de colunas ? " );
N = int.Parse ( Console.ReadLine ( ) );
// 2.3. ler a primeira matriz
for ( X = 0; X < M; X = X+1 )
for ( Y = 0; Y < N; Y = Y+1 )
{
// 2.3.1. ler um elemento
Console.Write ( "\nM1[" + X + "," + Y + "]= " );
M1 [ X, Y ] = double.Parse ( Console.ReadLine ( ) );
} // fim for
// 2.4. ler a segunda  matriz
for ( X = 0; X < M; X = X+1 )
for ( Y = 0; Y < N; Y = Y+1 )
{
// 2.4.1. ler um elemento
Console.Write ( "\nM2[" + X + "," + Y + "]= " );
M2 [ X, Y ] = double.Parse ( Console.ReadLine ( ) );
} // fim for

// 3. calcular a soma de duas matrizes
for ( X = 0; X < M; X = X+1 )
for ( Y = 0; Y < N; Y = Y+1 )
{
// 3.1. somar cada elemento
M3 [ X, Y ] = M1 [ X, Y ] + M2 [ X, Y ];
} // fim for

```

```
// 4. mostrar resultado
IO.println ( "\nMatriz resultante: " );
for ( X = 0; X < M; X = X+1 )
{
    for ( Y = 0; Y < N; Y = Y+1 )
    {
        // 4.1. mostrar um elemento de cada vez
        Console.Write ( "" + M3 [ X, Y ] + "\t" );
    } // fim for
    Console.WriteLine ( );
} // fim for

// pausa para terminar
Console.Write ( "\nPressionar ENTER para terminar." );
Console.ReadLine ( );
} // end Main ( )

} // fim Exemplo_7a class
```



Programa em Java:

```

/**
 * Exemplo 7a
 * Calcular a soma de duas matrizes MxN.
 */

// ----- classes necessarias

// ----- definicao de classe

class Exemplo_7a
{
//
// parte principal
//
    public static void main ( String [ ] args )
    {
        // 1. definir dados
        int          M, N,                // numero de linhas e colunas
                  X, Y;                  // indices

        double [ ] [ ] M1 = new double [ 10 ][ 10 ],    // primeiro matriz
                  M2 = new double [ 10 ][ 10 ],    // segundo matriz
                  M3 = new double [ 10 ][ 10 ];        // soma de duas matrizes

        // 2. ler dados
        // 2.1 ler o numero de linhas
        System.out.print ( "\nNumero de linhas ? " );
        M = Integer.parseInt ( System.console( ).readLine( ) );
        // 2.2 ler o numero de colunas
        System.out.print ( "\nNumero de colunas ? " );
        N = Integer.parseInt ( System.console( ).readLine( ) );
        // 2.3. ler a primeira matriz
        for ( X = 0; X < M; X = X+1 )
            for ( Y = 0; Y < N; Y = Y+1 )
            {
                // 2.3.1. ler um elemento
                System.out.print ( "\nM1[" + X + "," + Y + "]= " );
                M1 [ X ][ Y ] = Double.parseDouble ( System.console( ).readLine( ) );
            } // fim for
        // 2.4. ler a segunda matriz
        for ( X = 0; X < M; X = X+1 )
            for ( Y = 0; Y < N; Y = Y+1 )
            {
                // 2.4.1. ler um elemento
                System.out.print ( "\nM2[" + X + "," + Y + "]= " );
                M2 [ X ][ Y ] = Double.parseDouble ( System.console( ).readLine( ) );
            } // fim for
    }
}

```

```

// 3. calcular a soma de duas matrizes
for ( X = 0; X < M; X = X+1 )
    for ( Y = 0; Y < N; Y = Y+1 )
    {
        // 3.1. somar cada elemento
        M3 [ X ][ Y ] = M1 [ X ][ Y ] + M2 [ X ][ Y ];
    } // fim for

// 4. mostrar resultado
System.out.println ( "\nMatriz resultante: " );
for ( X = 0; X < M; X = X+1 )
{
    for ( Y = 0; Y < N; Y = Y+1 )
    {
        // 4.1. mostrar um elemento de cada vez
        System.out.print ( "" + M3 [ X ][ Y ] + "\t" );
    } // fim for
    System.out.println ( );
} // fim for

// pausa para terminar
System.out.print ( "\n\nPressionar ENTER para terminar." );
System.console( ).readLine( );
} // end main ( )

} // fim Exemplo_7a class

```

Outra versão com procedimentos.

Programa em C:

```
// Exemplo 7b.
// Calcular a soma de duas matrizes MxN.
//
// bibliotecas necessarias
#include <stdio.h>
#include <stdlib.h>
//

// definir constante e tipo
const int MAXTAM = 10;
typedef float MATRIZ [ MAXTAM ] [ MAXTAM ];

// procedimento para ler dados
void LER ( MATRIZ A, int & M, int & N )
{
    // definir dados locais
    int X, Y;
    // ler o numero de linhas
    printf ( "\nNumero de linhas ? " ); scanf ( "%d", &M );
    // ler o numero de colunas
    printf ( "\nNumero de colunas ? " ); scanf ( "%d", &N );
    // ler os elementos da matriz
    for ( X = 0; X < M; X = X + 1 )
        for ( Y = 0; Y < N; Y = Y + 1 )
        { // ler um elemento
            printf ( "\nElemento [%d, %d] = ", X, Y ); scanf ( "%f", &A [ X ] [ Y ] );
        } // fim for
} // fim do procedimento LER ( )

// procedimento para mostrar dados
void MOSTRAR ( MATRIZ A, int M, int N )
{
    // definir dados locais
    int X, Y;
    // mostrar os elementos da matriz
    for ( X = 0; X < M; X = X + 1 )
    {
        for ( Y = 0; Y < N; Y = Y + 1 )
        { // mostrar um elemento de cada vez
            printf ( "%f\t", A [ X ] [ Y ] );
        } // fim for
        printf ( "\n" );
    } // fim for
} // fim do procedimento MOSTRAR ( )
```

```

// procedimento para somar matrizes
void SOMAR ( MATRIZ C, MATRIZ A, MATRIZ B, int M, int N )
{
    // definir dado local
    int X, Y;
    // somar os elementos da matriz
    for ( X = 0; X < M; X = X + 1 )
        for ( Y = 0; Y < N; Y = Y + 1 )
        { // somar cada elemento
            C [ X ] [ Y ] = A [ X ] [ Y ] + B [ X ] [ Y ];
        } // fin for
} // fim do procedimento SOMAR ( )

// parte principal
//
// 1. definir dados
int  X, Y,                // indices
    L1, C1,              // linhas e colunas da primeira  matriz
    L2, C2,              // linhas e colunas da segunda matriz
    L3, C3;              // linhas e colunas da terceira  matriz
float M1 [ 10 ] [ 10 ],  // primeira  matriz
      M2 [ 10 ] [ 10 ],  // segunda matriz
      M3 [ 10 ] [ 10 ];  // soma de duas matrizes

// 2. ler dados
// 2.1 ler a primeira matriz
LER ( M1, L1, C1 );
// 2.2 ler a segunda matriz
LER ( M2, L2, C2 );

// 3. calcular a soma de duas matrizes
if ( L1 == L2 && C1 == C2 )
{
    L3 = L1;
    C3 = C1;
    SOMAR ( M3, M1, M2, L3, C3 );
} // fim if

// 4. mostrar o resultado
printf ( "\nMatriz resultante:\n" );
MOSTRAR ( M3, L3, C3 );

// pausa para terminar
printf ( "\nPressionar ENTER para terminar." );
getchar ( );
return ( 0 );
} // fim do programa

```

Programa em C++:

```
// Exemplo 7b.
// Calcular a soma de duas matrizes MxN.
//
// bibliotecas necessarias
#include <iostream>
using namespace std;
//

// definir constante e tipo
const int MAXTAM = 10;
typedef float MATRIZ [ MAXTAM ] [ MAXTAM ];

// procedimento para ler dados
void LER ( MATRIZ A, int & M, int & N )
{
    // definir dados locais
    int X, Y;
    // ler o numero de linhas
    cout << "\nNumero de linhas ? "; cin >> M;
    // ler o numero de colunas
    cout << "\nNumero de colunas ? "; cin >> N;
    // ler os elementos da matriz
    for ( X = 0; X < M; X = X + 1 )
        for ( Y = 0; Y < N; Y = Y + 1 )
        { // ler um elemento
            cout << "\nElemento [" << X << ", " << Y << "] = "; cin >> A [ X ] [ Y ];
        } // fim for
} // fim do procedimento LER ( )

// procedimento para mostrar dados
void MOSTRAR ( MATRIZ A, int M, int N )
{
    // definir dados locais
    int X, Y;
    // mostrar os elementos da matriz
    for ( X = 0; X < M; X = X + 1 )
    {
        for ( Y = 0; Y < N; Y = Y + 1 )
        { // mostrar um elemento de cada vez
            cout << A [ X ] [ Y ] << " ";
        } // fim for
        cout << "\n";
    } // fim for
} // fim do procedimento MOSTRAR ( )
```

```

// procedimento para somar matrizes
void SOMAR ( MATRIZ C, MATRIZ A, MATRIZ B, int M, int N )
{
    // definir dado local
    int X, Y;
    // somar os elementos da matriz
    for ( X = 0; X < M; X = X + 1 )
        for ( Y = 0; Y < N; Y = Y + 1 )
        { // somar cada elemento
            C [ X ] [ Y ] = A [ X ] [ Y ] + B [ X ] [ Y ];
        } // fin for
} // fim do procedimento SOMAR ( )

// parte principal
//
// 1. definir dados
int    X, Y,                // indices
      L1, C1,              // linhas e colunas da primeira  matriz
      L2, C2,              // linhas e colunas da segunda matriz
      L3, C3;              // linhas e colunas da terceira  matriz
float M1 [ 10 ] [ 10 ],    // primeira  matriz
      M2 [ 10 ] [ 10 ],    // segunda matriz
      M3 [ 10 ] [ 10 ];    // soma de duas matrizes

// 2. ler dados
// 2.1 ler a primeira matriz
LER ( M1, L1, C1 );
// 2.2 ler a segunda matriz
LER ( M2, L2, C2 );

// 3. calcular a soma de duas matrizes
if ( L1 == L2 && C1 == C2 )
{
    L3 = L1;
    C3 = C1;
    SOMAR ( M3, M1, M2, L3, C3 );
} // fim if

// 4. mostrar o resultado
cout << "\nMatriz resultante:\n";
MOSTRAR ( M3, L3, C3 );

// pausa para terminar
cout << "\nPressionar ENTER para terminar.";
getchar ( );
return EXIT_SUCCESS;
} // fim do programa

```

Outra versão do programa em C#:

```

/*
 * Exemplo 7b
 * Calcular a soma de duas matrizes MxN.
 */
using System;

class Exemplo_7b
{
    // definir constante global
    public static final int MAXTAM = 10;

    // definir dado global
    public static int M = 0, // numero de linhas
        N = 0; // numero de colunas

    // procedimento para ler dados
    public static void LER ( double [ , ] A )
    {
        // definir dados locais
        int X, Y;
        // ler o numero de linhas
        Console.Write ( "\nNumero de linhas ? " );
        M = int.Parse ( Console.ReadLine ( ) );
        // ler o numero de colunas
        Console.Write ( "\nNumero de colunas ? " );
        N = int.Parse ( Console.ReadLine ( ) );
        // ler os elementos da matriz
        for ( X = 0; X < M; X = X+1 )
            for ( Y = 0; Y < N; Y = Y+1 )
            {
                // ler um elemento
                Console.Write ( " \nElemento[" + X + "," + Y + "]= " );
                A [ X, Y ] = double.Parse ( Console.ReadLine ( ) );
            } // fim for
    } // fim do procedimento LER ( )

    // procedimento para mostrar dados
    public static void MOSTRAR ( double [ , ] A, int M, int N )
    {
        // definir dados locais
        int X, Y;
        // mostrar os elementos da matriz
        for ( X = 0; X < M; X = X+1 )
        {
            for ( Y = 0; Y < N; Y = Y+1 )
            {
                // mostrar um elemento de cada vez
                Console.Write ( "" + A [ X , Y ] + "\t" );
            } // fim for
            Console.WriteLine ( );
        } // fim for
    } // fim do procedimento MOSTRAR ( )
}

```

```

// procedimento para somar matrizes
public static void SOMAR ( double [ , ] C,
                           double [ , ] A,
                           double [ , ] B,
                           int M, int N )
{
    // definir dados locais
    int X, Y;
    // somar os elementos da matriz
    for ( X = 0; X < M; X = X+1 )
        for ( Y = 0; Y < N; Y = Y+1 )
        {
            // somar cada elemento
            C [ X , Y ] = A [ X, Y ] + B [ X, Y ];
        } // fim for
    } // fim do procedimento SOMAR ( )

//
// parte principal
//
public static void Main ( )
{
    // 1. definir dados
    int    X, Y,                                // indices
          L1, C1,                               // linhas e colunas da primeira matriz
          L2, C2,                               // linhas e colunas da segunda matriz
          L3 = 0, C3 = 0;                       // linhas e colunas da terceira matriz
    double [ , ] M1 = new double [ 10, 10 ],   // primeiro matriz
              M2 = new double [ 10, 10 ],       // segundo matriz
              M3 = new double [ 10, 10 ];       // soma de duas matrizes

    // 2. ler dados
    // 2.1. ler a primeira matriz
    LER ( M1 ); L1 = M; C1 = N;
    // 2.4. ler a segunda matriz
    LER ( M2 ); L2 = M; C2 = N;

    // 3. calcular a soma de duas matrizes
    if ( L1 == L2 && C1 == C2 )
    {
        L3 = L1; C3 = C1;
        SOMAR ( M3, M1, M2, L3, C3 );
    } // fim if

    // 4. mostrar resultado
    Console.WriteLine ( "\nMatriz resultante: " );
    MOSTRAR ( M3, L3, C3 );

    // pausa para terminar
    Console.Write ( "\nPressionar ENTER para terminar." );
    Console.ReadLine ( );
} // end Main ( )

} // fim Exemplo_7b class

```



Outra versão do programa em Java:

```

/**
 * Exemplo 7b
 * Calcular a soma de duas matrizes MxN.
 */

// ----- classes necessarias

// ----- definicao de classe

class Exemplo_7b
{
    // definir constante global
    public static final int MAXTAM = 10;

    // definir dado global
    public static int M = 0, // numero de linhas
                    N = 0; // numero de colunas

    // procedimento para ler dados
    public static void LER ( double [ ][ ] A )
    {
        // definir dados locais
        int X, Y;

        // ler o numero de linhas
        System.out.print ( "\nNumero de linhas ? " );
        M = Integer.parseInt ( System.console( ).readLine( ) );
        // ler o numero de colunas
        System.out.print ( "\nNumero de colunas ? " );
        N = Integer.parseInt ( System.console( ).readLine( ) );
        // ler os elementos da matriz
        for ( X = 0; X < M; X = X+1 )
            for ( Y = 0; Y < N; Y = Y+1 )
            {
                // ler um elemento
                System.out.print ( "\nElemento [" + X + "," + Y + "] = " );
                A [ X ][ Y ] = Double.parseDouble ( System.console( ).readLine( ) );
            } // fim for
    } // fim do procedimento LER ( )
}

```

```

// procedimento para mostrar dados
public static void MOSTRAR ( double [ ][ ] A, int M, int N )
{
    // definir dados locais
    int X, Y;
    // mostrar os elementos da matriz
    for ( X = 0; X < M; X = X+1 )
    {
        for ( Y = 0; Y < N; Y = Y+1 )
        {
            // mostrar um elemento de cada vez
            System.out.print ( "" + A [ X ][ Y ] + "\t" );
        } // fim for
        System.out.println ( );
    } // fim for
} // fim do procedimento MOSTRAR ( )

// procedimento para somar matrizes
public static void SOMAR ( double [ ][ ] C,
                           double [ ][ ] A,
                           double [ ][ ] B,
                           int M, int N )
{
    // definir dados locais
    int X, Y;
    // somar os elementos da matriz
    for ( X = 0; X < M; X = X+1 )
        for ( Y = 0; Y < N; Y = Y+1 )
        {
            // somar cada elemento
            C [ X ][ Y ] = A [ X ][ Y ] + B [ X ][ Y ];
        } // fim for
} // fim do procedimento SOMAR ( )

//
// parte principal
//
public static void main ( String [ ] args )
{
    // 1. definir dados
    int X, Y, // indices
        L1, C1, // linhas e colunas da primeira matriz
        L2, C2, // linhas e colunas da segunda matriz
        L3 = 0, C3 = 0; // linhas e colunas da terceira matriz
    double [ ][ ] M1 = new double [ 10 ][ 10 ], // primeiro matriz
        M2 = new double [ 10 ][ 10 ], // segundo matriz
        M3 = new double [ 10 ][ 10 ]; // soma de duas matrizes

    // 2. ler dados
    // 2.1. ler a primeira matriz
    LER ( M1 ); L1 = M; C1 = N;
    // 2.4. ler a segunda matriz
    LER ( M2 ); L2 = M; C2 = N;

```

```
// 3. calcular a soma de duas matrizes
if ( L1 == L2 && C1 == C2 )
{
    L3 = L1; C3 = C1;
    SOMAR ( M3, M1, M2, L3, C3 );
} // fim if

// 4. mostrar resultado
System.out.println ( "\nMatriz resultante: " );
MOSTRAR ( M3, L3, C3 );

// pausa para terminar
System.out.print ( "\n\nPressionar ENTER para terminar." );
System.console ( ).readLine ( );
} // end main ( )

} // fim Exemplo_7b class
```

Programa em Python:

```
# Exemplo 7.
# Calcular a soma de duas matrizes MxN.
#
def create_int_matrix ( lines, columns ):
    matrix = [ ];
    for x in range ( lines ):
        row = [ ];
        for y in range ( columns ):
            row.append ( 0 );
        matrix.append ( row );
    return ( matrix );
#

# 1. definir dados
M = 0;                                # numero de linhas
N = 0;                                # numero de colunas
X = 0;                                # indice
Y = 0;                                # indice
M1 = create_int_matrix ( 10, 10 );    # primeira matriz
M2 = create_int_matrix ( 10, 10 );    # segunda matriz
M3 = create_int_matrix ( 10, 10 );    # soma de duas matrizes

# 2. ler dados
# 2.1 ler o numero de linhas
M = int ( input ( "\nNumero de linhas ? " ) );
# 2.2 ler o numero de colunas
N = int ( input ( "\nNumero de colunas ? " ) );

# 2.3 ler a primeira matriz
for X in range ( 0, M, 1 ):
    for Y in range ( 0, N, 1 ):
        # 2.3 1 ler um elemento
        print ( "\nM1 [ ", X, ", ", Y, " ] = ", end=" " );
        M1 [ X ][ Y ] = int ( input ( " " ) );
        # fim repetir
# fim repetir

# 2.4 ler a segunda matriz
for X in range ( 0, M, 1 ):
    for Y in range ( 0, N, 1 ):
        # 2.3 1 ler um elemento
        print ( "\nM2 [ ", X, ", ", Y, " ] = ", end=" " );
        M2 [ X ][ Y ] = int ( input ( " " ) );
        # fim repetir
# fim repetir
```

```

# 3. calcular a soma de duas matrizes
for X in range ( 0, M, 1 ):
    for Y in range ( 0, N, 1 ):
        # 3.1 somar cada elemento
        M3 [ X ][ Y ] = M1 [ X ][ Y ] + M2 [ X ][ Y ];
        # fim repetir
    # fim repetir

# 4. mostrar o resultado
print ( " \nMatriz resultante:\n " );
for X in range ( 0, M, 1 ):
    for Y in range ( 0, N, 1 ):
        # 4.1 mostrar um elemento de cada vez
        print ( M3 [ X ][ Y ], "\t", end=" " );
        # fim repetir
    print ( "\n" );
# fim repetir

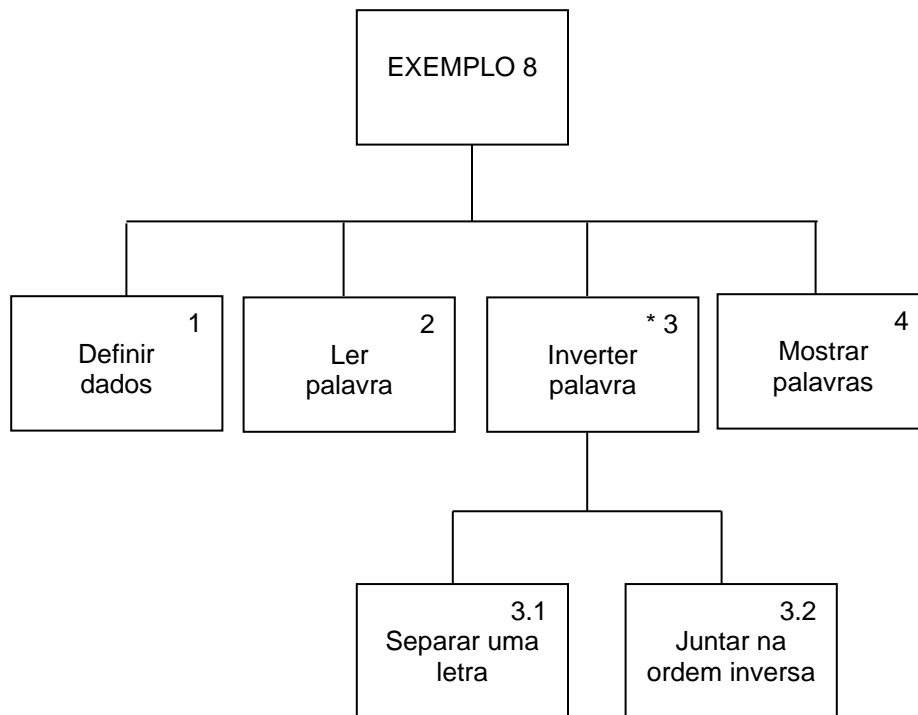
# pausa para terminar
print ( "\nPressionar ENTER para terminar." );
input ( );
# fim do programa

```

## Exemplo 8.

Ler uma palavra do teclado, inverter a ordem das letras e mostrar a palavra original e a invertida.

Diagrama funcional:



Análise de dados :

- Dados do problema :

Dado	Tipo	Valor inicial	Função
PALAVRA	caractere(20)	-	armazenar palavra original
I	inteiro	0	índice
TAMANHO	inteiro	0	tamanho da palavra original
LETRA	caractere	-	armazenar uma letra

- Resultados do problema :

Dado	Tipo	Valor inicial	Função
INVERTIDA	caractere(20)	vazia	armazenar palavra invertida

- Avaliação da solução :

palavra original = ROMA

palavra invertida = AMOR

Algoritmo:

Esboço:

Primeira versão, só comentários.

Exemplo 8	v.1
Ação	Bloco
! definir dados	1
! ler palavra	2
! inverter palavra	3
! separar uma letra	3.1
! juntar na ordem inversa	3.2
! mostrar palavras	4

Segunda versão, com refinamento do primeiro bloco.

Exemplo 8	v.2
Ação	Bloco
! definir dados	1
caractere PALAVRA [ 20 ], ! palavra original INVERTIDA [ 20 ]; ! palavra invertida inteiro X = 0, ! índice TAMANHO = 0; ! tamanho da palavra caractere LETRA; ! uma letra ! situação inicial INVERTIDA [ 0 ] = ε ; ! vazia	
! ler palavra	2
! inverter palavra	3
! separar uma letra	3.1
! juntar na ordem inversa	3.2
! mostrar palavras	4

Terceira versão, com refinamento do segundo e do quarto bloco.

Exemplo 8	v.3
Ação	Bloco
! definir dados	1
caractere PALAVRA [ 20 ], ! palavra original INVERTIDA [ 20 ]; ! palavra invertida inteiro X = 0, ! índice TAMANHO = 0; ! tamanho da palavra caractere LETRA; ! uma letra ! situação inicial INVERTIDA [ 0 ] = ε ; ! vazia	
! ler palavra	2
tela ← "Qual a palavra ?"; PALAVRA ← teclado;	
! inverter palavra	3
! separar uma letra	3.1
! juntar na ordem inversa	3.2
! mostrar palavras	4
tela ← ( "Palavra original =", PALAVRA ); tela ← ( "Palavra invertida =", INVERTIDA );	



Quarta versão, com refinamento do terceiro bloco.

Exemplo 8	v.4
Ação	Bloco
! definir dados	1
caractere PALAVRA [ 20 ], ! palavra original INVERTIDA [ 20 ]; ! palavra invertida inteiro X = 0, ! índice TAMANHO = 0; ! tamanho da palavra caractere LETRA; ! uma letra ! situação inicial INVERTIDA [ 0 ] = ε ; ! vazia	
! ler palavra	2
tela ← "Qual a palavra ?"; PALAVRA ← teclado;	
! inverter palavra	3
! determinar o tamanho da palavra original	3.1
X = 1 : TAMANHO : 1	3.2
! separar uma letra	3.2.1
! juntar na ordem inversa	3.2.2
! mostrar palavras	4
tela ← ("Palavra original =", PALAVRA ); tela ← ("Palavra invertida =", INVERTIDA);	

Quinta versão, com novo refinamento do terceiro bloco.

Exemplo 8	v.5
Ação	Bloco
! definir dados	1
caractere PALAVRA [ 20 ], ! palavra original INVERTIDA [ 20 ]; ! palavra invertida inteiro X = 0, ! índice TAMANHO = 0; ! tamanho da palavra caractere LETRA; ! uma letra ! situação inicial INVERTIDA [ 0 ] = ε ; ! vazia	
! ler palavra	2
tela ← "Qual a palavra ?"; PALAVRA ← teclado;	
! inverter palavra	3
! determinar o tamanho da palavra original	3.1
PALAVRA [ TAMANHO ] ≠ ε ?	
TAMANHO = TAMANHO + 1;	3.1.1
X = 1 : (TAMANHO-1) : 1	3.2
! separar uma letra	3.2.1
! juntar na ordem inversa	3.2.2
! mostrar palavras	4
tela ← ("Palavra original =", PALAVRA ); tela ← ("Palavra invertida =", INVERTIDA );	

Sexta versão, com mais um refinamento do terceiro bloco.

Exemplo 8	v.6
Ação	Bloco
! definir dados	1
caractere PALAVRA [ 20 ], ! palavra original INVERTIDA [ 20 ]; ! palavra invertida inteiro X = 0, ! índice TAMANHO = 0; ! tamanho da palavra caractere LETRA; ! uma letra ! situação inicial INVERTIDA [ 0 ] = ε ; ! vazia	
! ler palavra	2
tela ← "Qual a palavra ?"; PALAVRA ← teclado;	
! inverter palavra	3
! determinar o tamanho da palavra original	3.1
PALAVRA [ TAMANHO ] ≠ ε ?	
TAMANHO = TAMANHO + 1;	3.1.1
X = 1 : (TAMANHO-1) : 1	3.2
! separar uma letra LETRA = PALAVRA [ X ];	3.2.1
! juntar na ordem inversa	3.2.2
INVERTIDA [ TAMANHO - X ] = LETRA;	
INVERTIDA [ TAMANHO ] = ε ; ! fechar a palavra	
! mostrar palavras	4
tela ← ( "Palavra original =" , PALAVRA ); tela ← ( "Palavra invertida =" , INVERTIDA );	

Programa em SCILAB:

```
// Exemplo 8.
// Ler e inverter uma palavra.
//
// 1. definir dados
PALAVRA (1:20) = " ", // palavra original
INVERTIDA (1:20) = " " // palavra invertida
X = 0; // indice
TAMANHO = 0; // tamanho da palavra
LETRA = ' '; // uma letra
// situacao inicial
INVERTIDA = " "; // vazia
// ler palavra
PALAVRA = input ( "\nQual a palavra ? " );
// inverter palavra
// determinar o tamanho da palavra original
TAMANHO = length ( PALAVRA );
//
for ( X = 1 : 1: TAMANHO )
// separar uma letra
LETRA = PALAVRA ( X );
// juntar na ordem inversa
INVERTIDA ( TAMANHO - X + 1 ) = LETRA;
end // for
// mostrar palavras
printf ( "\nPalavra original = %s", PALAVRA );
printf ( "\nPalavra invertida = %s", INVERTIDA );

// pausa para terminar
printf ( "\nPressionar ENTER para terminar." );
halt;
// fim do programa
```

OBS.:

Ao entrar com a palavra usar apóstrofos: 'abc' .

Programa em C:

```
// Exemplo 8a.
// Ler e inverter uma palavra.
//
// bibliotecas necessarias
#include <stdio.h>
#include <stdlib.h>
//
// parte principal
//
int main (void)
{
// 1. definir dados
char PALAVRA [ 20 ], // palavra original
    INVERTIDA [ 20 ]; // palavra invertida
int   X = 0,         // indice
    TAMANHO = 0;     // tamanho da palavra
char LETRA;          // uma letra
// situacao inicial
INVERTIDA [ 0 ] = '\0' ; // vazia
// 2. ler palavra
printf ( "\nQual a palavra ? " );
scanf ( "%s", PALAVRA );
// 3. inverter palavra
// 3.1. determinar o tamanho da palavra original
while ( PALAVRA [ TAMANHO ] != '\0' )
{ TAMANHO = TAMANHO + 1; }
// 3.2. para cada letra
for ( X = 0; X <= (TAMANHO-1); X = X + 1)
{
// 3.2.1. separar uma letra
LETRA = PALAVRA [ X ];
// 3.2.2. juntar na ordem inversa
INVERTIDA [ TAMANHO - X - 1 ] = LETRA;
} // fim for
INVERTIDA [ TAMANHO ] = '\0'; // fechar a palavra
// 4. mostrar palavras
printf ( "\nPalavra original  = %s", PALAVRA );
printf ( "\nPalavra invertida = %s", INVERTIDA );

// pausa para terminar
printf ( "Pressionar ENTER para terminar." );
getchar ( );
return ( 0 );
} // fim do programa
```

Outra versão, com uso de biblioteca:

```
// Exemplo 8b.
// Ler e inverter uma palavra.
//
// bibliotecas necessarias
#include <stdio.h>
#include <stdlib.h>
#include <string.h>      // para strlen( )
//
typedef char STRING [ 20 ];

// parte principal
//
int main (void)
{
// 1. definir dados
    STRING PALAVRA ,      // palavra original
          INVERTIDA;      // palavra invertida
    int   X = 0;          // indice
// 2. ler palavra
    printf ( "\nQual a palavra ? " );
    scanf ( "%s", PALAVRA );
// 3. inverter palavra
    for ( X = 0; X < strlen (PALAVRA); X = X + 1)
    { // 3.2. separar e juntar uma letra na ordem inversa
        INVERTIDA [ strlen (PALAVRA) - X - 1 ] = PALAVRA [ X ];
    } // fim for
    INVERTIDA [ strlen (PALAVRA) ] = '\0'; // fechar a palavra
// 4. mostrar palavras
    printf ( "\nPalavra original  = %s", PALAVRA );
    printf ( "\nPalavra invertida = %s", INVERTIDA );

// pausa para terminar
    printf ( "Pressionar ENTER para terminar." );
    getchar ( );
    return ( 0 );
} // fim do programa
```

Programa em C++:

```
// Exemplo 8a.
// Ler e inverter uma palavra.
//
// bibliotecas necessarias
#include <iostream>
#include <string>
using namespace std;
//
// parte principal
//
int main (void)
{
// 1. definir dados
    string PALAVRA,          // palavra original
           INVERTIDA];      // palavra invertida
    int    X = 0,            // indice
           TAMANHO = 0;      // tamanho da palavra
    char LETRA;              // uma letra
// 2. ler palavra
    cout << "\nQual a palavra ? ";
    cin  >> PALAVRA;
// situacao inicial
    INVERTIDA = PALAVRA; // vazia
// 3. inverter palavra
    // 3.1. determinar o tamanho da palavra original
    TAMANHO ] = PALAVRA.length ( )
    // 3.2. para cada letra
    for ( X = 0; X <= (TAMANHO-1); X = X + 1)
    {
        // 3.2.1. separar uma letra
        LETRA = PALAVRA [ X ];
        // 3.2.2. juntar na ordem inversa
        INVERTIDA [ TAMANHO - X - 1 ] = LETRA;
    } // fim for
// 4. mostrar palavras
    cout << "\nPalavra original =" << PALAVRA;
    cout << "\nPalavra invertida=" << INVERTIDA;

// pausa para terminar
    cout << "Pressionar ENTER para terminar.";
    cin.get ( );
    return EXIT_SUCCESS;
} // fim do programa
```

Programa em C#:

```

/*
 * Exemplo 8
 * Ler e inverter uma palavra.
 */
using System;

class Exemplo_8
{
    //
    // parte principal
    //
    public static void Main ( )
    {
        // 1. definir dados
        string PALAVRA,           // palavra original
            INVERTIDA;           // palavra invertida
        int    X = 0,             // indice
            TAMANHO = 0;         // tamanho da palavra
        char   LETRA;             // uma letra da palavra

        // 2. ler palavra
        Console.Write ( "\nQual a palavra ? " );
        PALAVRA = Console.ReadLine ( );

        // 3. inverter palavra
        // 3.1. determinar o tamanho da palavra original
        TAMANHO = PALAVRA.Length;
        // 3.2. para cada letra
        for ( X = 0; X < TAMANHO; X = X+1 )
        {
            // 3.2.1. separar uma letra
            LETRA = PALAVRA [ X ];
            // 3.2.2. juntar na ordem inversa
            INVERTIDA = LETRA + INVERTIDA;
        } // fim for

        // 4. mostrar palavras
        Console.WriteLine ( "\nPalavra original = " + PALAVRA );
        Console.WriteLine ( "\nPalavra invertida = " + INVERTIDA );

        // pausa para terminar
        Console.Write ( "\nPressionar ENTER para terminar." );
        Console.ReadLine ( );
    } // end Main ( )
} // fim Exemplo_8 class

```

Programa em Java:

```

/**
 * Exemplo 8
 * Ler e inverter uma palavra.
 */

// ----- classes necessarias

// ----- definicao de classe

class Exemplo_8
{
//
// parte principal
//
    public static void main ( String [ ] args )
    {
        // 1. definir dados
        String PALAVRA,           // palavra original
            INVERTIDA = "";       // palavra invertida
        int   X = 0,              // indice
            TAMANHO = 0;          // tamanho da palavra
        char  LETRA;              // uma letra da palavra

        // 2. ler palavra
        System.out.print ( "\nQual a palavra ? " );
        PALAVRA = System.console( ).readLine( );

        // 3. inverter palavra
        // 3.1. determinar o tamanho da palavra original
        TAMANHO = PALAVRA.length ( );
        // 3.2. para cada letra
        for ( X = 0; X < TAMANHO; X = X+1 )
        {
            // 3.2.1. separar uma letra
            LETRA = PALAVRA.charAt ( X );
            // 3.2.2. juntar na ordem inversa
            INVERTIDA = LETRA + INVERTIDA;
        } // fim for

        // 4. mostrar palavras
        System.out.println ( "\nPalavra original = " + PALAVRA );
        System.out.println ( "\nPalavra invertida = " + INVERTIDA );

        // pausa para terminar
        System.out.print ( "\n\nPressionar ENTER para terminar." );
        System.console( ).readLine( );
    } // end main ( )
} // fim Exemplo_8 class

```



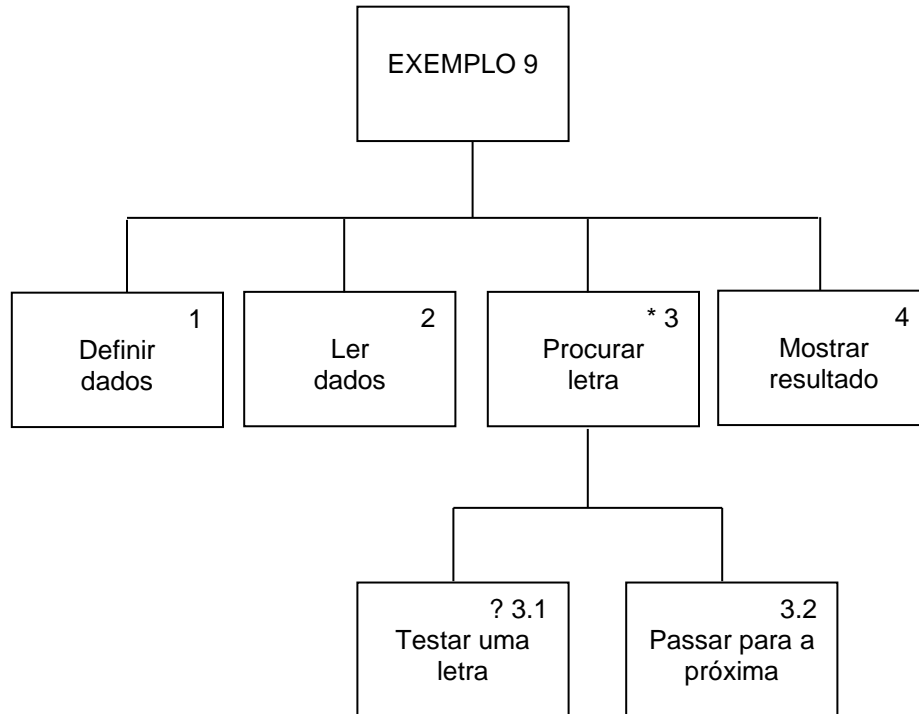
Programa em Python:

```
# Exemplo 8.
# Ler e inverter uma palavra.
#
# 1. definir dados
PALAVRA = "",          # palavra original
INVERTIDA = "",        # palavra invertida
X = 0;                # indice
TAMANHO = 0;          # tamanho da palavra
LETRA = " ";          # uma letra
# situacao inicial
INVERTIDA = " ";      # vazia
# ler palavra
PALAVRA = input ( "\nQual a palavra ? " );
# inverter palavra
# determinar o tamanho da palavra original
TAMANHO = len ( PALAVRA );
#
for X in range ( 0, TAMANHO, 1 ):
# separar uma letra
    LETRA = PALAVRA [ X ];
# juntar na ordem inversa
    INVERTIDA = LETRA + INVERTIDA;
# fim repetir
# mostrar palavras
print ( "\nPalavra original = ", PALAVRA );
print ( "\nPalavra invertida = ", INVERTIDA );
print ( );
# pausa para terminar
print ( "\nPressionar ENTER para terminar." );
input ( );
# fim do programa
```

## Exemplo 9.

Ler uma palavra e uma letra do teclado, e verificar se a letra existe na palavra.

Diagrama funcional:



Análise de dados :

- Dados do problema :

Dado	Tipo	Valor inicial	Função
PALAVRA	caractere(20)	-	armazenar palavra original
LETRA	caractere	-	armazenar uma letra
I	inteiro	0	índice
TAMANHO	inteiro	0	tamanho da palavra original

- Resultados do problema :

Dado	Tipo	Valor inicial	Função
RESPOSTA	inteiro	0	indicar se achar a letra, ou não

- Avaliação da solução :

palavra original = ROMA

letra = A      EXISTE

letra = E      NÃO EXISTE

- Considerações:

É interessante considerar o número de testes necessários para se determinar a pertinência de um determinado elemento :

Pior caso      : N  
 Caso médio    : (N+1)/2  
 Melhor caso    : 1

A não-pertinência é determinada por (N+1) comparações.

Algoritmo:

Esboço:

Primeira versão, só comentários.

Exemplo 9	v.1
Ação	Bloco
! definir dados	1
! ler dados	2
! procurar letra	3
! testar uma letra	3.1
! passar para a próxima	3.2
! mostrar resultado	4

Segunda versão, com refinamento do primeiro e do segundo bloco.

Exemplo 9	v.2
Ação	Bloco
! definir dados	1
caractere PALAVRA [ 20 ]; ! palavra original caractere LETRA;            ! uma letra inteiro    X = 0,                ! índice TAMANHO = 0; ! tamanho da palavra RESPOSTA = 0; ! resultado da procura	
! ler dados	2
tela ← "Qual a palavra ?"; PALAVRA ← teclado; tela ← "Qual a letra ?";    LETRA ← teclado;	
! procurar letra	3
! testar uma letra	3.1
! passar para a próxima	3.2
! mostrar resultado	4

Terceira versão, com refinamento do quarto bloco.

Exemplo 9	v.3
Ação	Bloco
! definir dados	1
caractere PALAVRA [ 20 ]; ! palavra original caractere LETRA; ! uma letra inteiro X = 0, ! índice TAMANHO = 0; ! tamanho da palavra RESPOSTA = 0; ! resultado da procura	
! ler dados	2
tela ← "Qual a palavra ?"; PALAVRA ← teclado; tela ← "Qual a letra ?"; LETRA ← teclado;	
! procurar letra	3
! testar uma letra	3.1
! passar para a próxima	3.2
! mostrar resultado	4
tela ← ("A letra ", LETRA); se ( RESPOSTA = 1 ) tela ← " EXISTE "; senão tela ← " NAO EXISTE "; tela ← (" na palavra ", PALAVRA);	

Quarta versão, com refinamento do terceiro bloco.

Exemplo 8	v.4
Ação	Bloco
! definir dados	1
caractere PALAVRA [ 20 ]; ! palavra original caractere LETRA; ! uma letra inteiro X = 0, ! índice TAMANHO = 0; ! tamanho da palavra RESPOSTA = 0; ! resultado da procura	
! ler dados	2
tela ← "Qual a palavra ?"; PALAVRA ← teclado; tela ← "Qual a letra ?"; LETRA ← teclado;	
! procurar letra	3
! determinar o tamanho da palavra original	3.1
PALAVRA [ TAMANHO ] ≠ ε ?	
TAMANHO = TAMANHO + 1;	3.1.1
X < TAMANHO & RESPOSTA = 0 ?	3.2
LETRA	! testar uma letra
=	RESPOSTA = 1;
PALAVRA [ X ]	! passar para a próxima
?	X = X + 1;
! mostrar resultado	4
tela ← ("A letra ", LETRA); se ( RESPOSTA = 1 ) tela ← " EXISTE "; senão tela ← " NAO EXISTE "; tela ← (" na palavra ", PALAVRA);	

Programa em SCILAB:

```
// Exemplo 9.
// Ler e procurar uma letra em uma palavra.
//
// 1. definir dados
PALAVRA (1:20) = " ", // palavra original
LETRA = ' ';          // uma letra
X      = 1;           // indice
TAMANHO = 0;          // tamanho da palavra
RESPOSTA = 0;         // resultado da procura
// 2. ler dados
PALAVRA = input ( "\nQual a palavra ? " );
LETRA    = input ( "\nQual a letra ? " );
// 3. procurar letra
// determinar o tamanho da palavra original
TAMANHO = length ( PALAVRA );
while ( X < TAMANHO & RESPOSTA == 0 )
// 3.1 testar uma letra
if ( LETRA == PALAVRA ( X ) )
    RESPOSTA = 1; // encontrou
else
// 3.2 passar para a proxima
    X = X + 1;
end // fim se
end // while
// 4. mostrar resultado
printf ( "\nA letra %c", LETRA );
if ( RESPOSTA == 1 )
    printf ( " EXISTE " );
else
    printf ( " NAO EXISTE " );
end // if
printf ( " na palavra %s ", PALAVRA );
//
// pausa para terminar
printf ( "\n\nPressionar ENTER para terminar." );
halt;
// fim do programa
```

OBS.:

Ao entrar com a palavra e a letra usar apóstrofes: 'abc' e 'b'.

Programa em C:

```
// Exemplo 9a.
// Ler e procurar uma letra em uma palavra.
//
// bibliotecas necessarias
#include <stdio.h>
#include <stdlib.h>
//
// parte principal
//
int main (void)
{
// 1. definir dados
char PALAVRA [ 20 ]; // palavra original
char LETRA;          // uma letra
int   X = 0,          // índice
      TAMANHO = 0;    // tamanho da palavra
      RESPOSTA = 0;   // resultado da procura
// 2. ler dados
printf ( "\nQual a palavra ? " );
scanf ( "%s", PALAVRA );
fflush ( stdin );
printf ( "\nQual a letra ? " );
scanf ( "%c", &LETRA );
// 3. procurar letra
// 3.1. determinar o tamanho da palavra original
while ( PALAVRA [ TAMANHO ] != '\0' )
{ TAMANHO = TAMANHO + 1; }
// 3.2. testar uma letra
while ( X < TAMANHO && RESPOSTA == 0 )
{ // 3.2.1. testar uma letra
  if ( LETRA == PALAVRA [ X ] )
    RESPOSTA = 1; // encontrou
  else
  { // 3.2.2. passar para a proxima
    X = X + 1;
  } // fim se
} // fim while
// 4. mostrar resultado
printf ( "\nA letra %c", LETRA );
if ( RESPOSTA == 1 )
  printf ( " EXISTE " );
else
  printf ( " NAO EXISTE " );
printf ( " na palavra %s", PALAVRA );
//
// pausa para terminar
printf ( "Pressionar ENTER para terminar." );
getchar ( );
return ( 0 );
} // fim do programa
```

Outra versão com biblioteca:

Programa em C:

```
// Exemplo 9b.
// Ler e procurar uma letra em uma palavra.
//
// bibliotecas necessarias
#include <stdio.h>
#include <stdlib.h>
#include <string.h>      // para strlen ( )
//
// parte principal
//
int main (void)
{
// 1. definir dados
char PALAVRA [ 20 ] ; // palavra original
char LETRA;           // uma letra
int   X = 0,           // indice
      RESPOSTA = 0;    // resultado da procura
// 2. ler dados
printf ( "\nQual a palavra ? " );
scanf ( "%s", PALAVRA );
fflush ( stdin );
printf ( "\nQual a letra ? " );
scanf ( "%c", &LETRA );
// 3. procurar letra
while ( X < strlen ( PALAVRA ) && RESPOSTA == 0 )
{ // 3.2.1. testar uma letra
  if ( LETRA == PALAVRA [ X ] )
    RESPOSTA = 1;      // encontrou
  else
  { // 3.2.2. passar para a proxima
    X = X + 1;
  } // fim se
} // fim while
// 4. mostrar resultado
printf ( "\nA letra %c", LETRA );
if ( RESPOSTA == 1 )
  printf ( " EXISTE " );
else
  printf ( " NAO EXISTE " );
printf ( " na palavra %s", PALAVRA );
//
// pausa para terminar
printf ( "Pressionar ENTER para terminar." );
getchar ( );
return ( 0 );
} // fim do programa
```

Programa em C++:

```
// Exemplo 9a.
// Ler e procurar uma letra em uma palavra.
//
// bibliotecas necessarias
#include <iostream>
#include <string>
using namespace std;
//
// parte principal
//
int main (void)
{
// 1. definir dados
    string PALAVRA;          // palavra original
    char  LETRA;             // uma letra
    int    X = 0,             // índice
           TAMANHO = 0;      // tamanho da palavra
           RESPOSTA = 0;     // resultado da procura
// 2. ler dados
    cout << "\nQual a palavra ? ";
    cin  >> PALAVRA;
    cout << "\nQual a letra ? ";
    cin  >> LETRA;
// 3. procurar letra
// 3.1. determinar o comprimento da palavra
    TAMANHO = PALAVRA.length ( );
// 3.2. testar uma letra
    while ( X < TAMANHO && RESPOSTA == 0 )
    { // 3.2.1. testar uma letra
        if ( LETRA == PALAVRA [ X ] )
            RESPOSTA = 1;      // encontrou
        else
        { // 3.2.2. passar para a proxima
            X = X + 1;
        } // fim se
    } // fim se
// 4. mostrar resultado
    cout << "\nA letra " << LETRA;
    if ( RESPOSTA == 1 )
        cout << " EXISTE ";
    else
        cout << " NAO EXISTE ";
    cout << " na palavra " << PALAVRA;
//
// pausa para terminar
    cout << "Pressionar ENTER para terminar.";
    cin.get ( );
    return EXIT_SUCCESS;
} // fim do programa
```



Programa em C#:

```

/*
 * Exemplo 9
 * Ler e procurar uma letra em uma palavra.
 */
using System;

class Exemplo_9
{
    //
    // parte principal
    //
    public static void Main ( )
    {
        // 1. definir dados
        string PALAVRA,           // palavra original
            INVERTIDA;           // palavra invertida
        char  LETRA    = ' ';    // uma letra da palavra
        int   X        = 0,      // indice
            TAMANHO    = 0,      // tamanho da palavra
            RESPOSTA   = 0;      // resultado da procura

        // 2. ler palavra
        Console.Write ( "\nQual a palavra ? " );
        PALAVRA = Console.ReadLine ( );
        Console.Write ( "\nQual a letra ? " );
        LETRA = (char) Console.Read ( );
        Console.ReadLine ( );           // limpar o buffer

        // 3. procurar letra
        // 3.1. determinar o tamanho da palavra original
        TAMANHO = PALAVRA.Length;
        while ( X < TAMANHO && RESPOSTA == 0 )
        {
            // 3.2.1 testar uma letra
            if ( LETRA == PALAVRA [ X ] )
                RESPOSTA = 1;           // encontrou
            else
            {
                // 3.2.2. passar para proxima
                X = X + 1;
            } // fim if
        } // fim while

        // 4. mostrar resultado
        Console.Write ( "\nA letra " + LETRA );
        if ( RESPOSTA == 1 )
            Console.Write ( " EXISTE " );
        else
            Console.Write ( " NAO EXISTE " );
        Console.Write ( " na palavra " + PALAVRA );

        // pausa para terminar
        Console.Write ( "\nPressionar ENTER para terminar." );
        Console.ReadLine ( );
    } // end Main ( )
} // fim Exemplo_9 class

```

Programa em Java:

```

/**
 * Exemplo 9
 * Ler e procurar uma letra em uma palavra.
 */

// ----- classes necessarias

// ----- definicao de classe

class Exemplo_9
{
//
// parte principal
//
    public static void main ( String [ ] args )
    {
        // 1. definir dados
        String PALAVRA,           // palavra original
            INVERTIDA;           // palavra invertida
        char  LETRA    = ' ';    // uma letra da palavra
        int   X        = 0,      // indice
            TAMANHO    = 0,      // tamanho da palavra
            RESPOSTA   = 0;      // resultado da procura

        // 2. ler palavra
        System.out.print ( "\nQual a palavra ? " );
        PALAVRA = System.console( ).readLine( );
        System.out.print ( "\nQual a letra  ? " );
        LETRA  = (System.console( ).readLine( )).charAt(0);

        // 3. procurar letra
        // 3.1. determinar o tamanho da palavra original
        TAMANHO = PALAVRA.length ( );
        while ( X < TAMANHO && RESPOSTA == 0 )
        {
            // 3.2.1 testar uma letra
            if ( LETRA == PALAVRA.charAt ( X ) )
                RESPOSTA = 1;           // encontrou
            else
                // 3.2.2. passar para proxima
                X = X + 1;
        } // fim while

        // 4. mostrar resultado
        System.out.print ( "\nA letra \" + LETRA + "\"" );
        if ( RESPOSTA == 1 )
            System.out.print ( " EXISTE " );
        else
            System.out.print ( " NAO EXISTE " );
        System.out.print ( " na palavra \" + PALAVRA + "\"" );

        // pausa para terminar
        System.out.print ( "\n\nPressionar ENTER para terminar." );
        System.console( ).readLine( );
    } // end main ( )
} // fim Exemplo_9 class

```

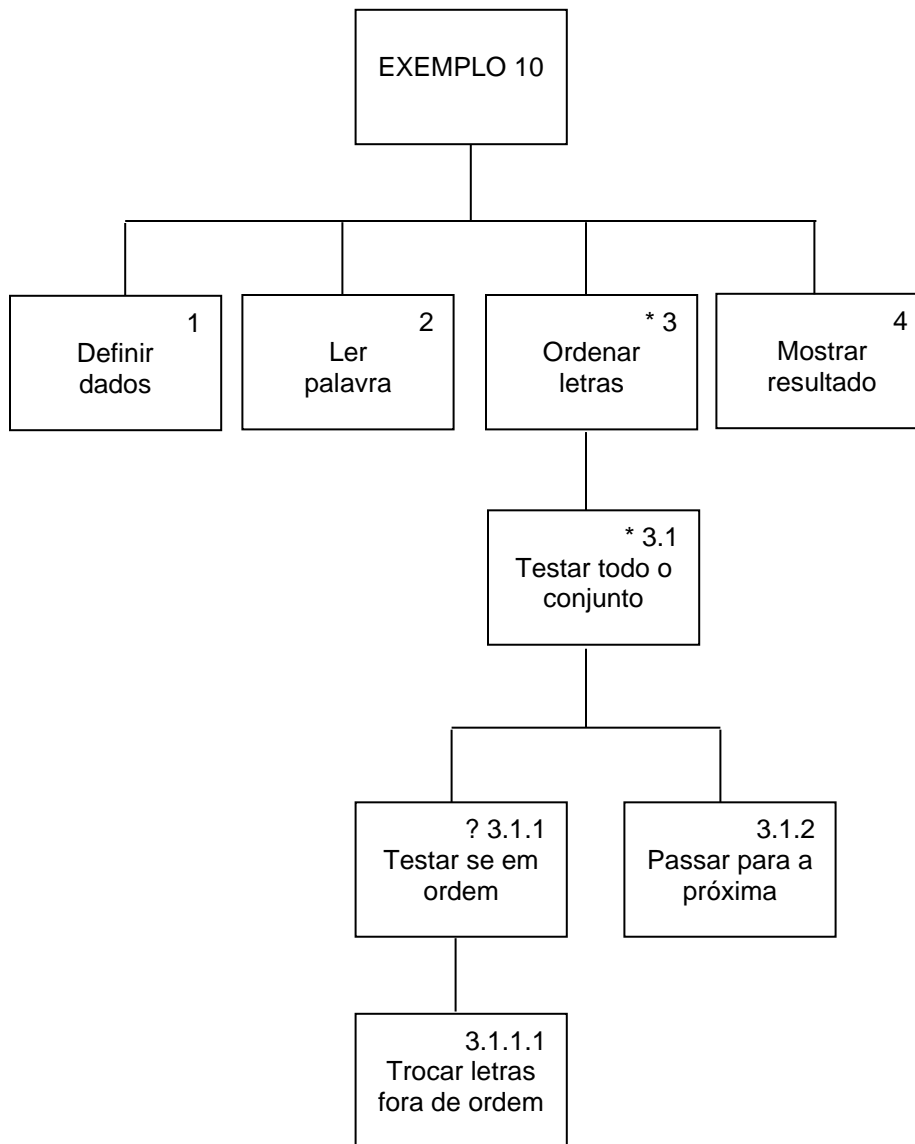
Programa em Python:

```
# Exemplo 9.
# Ler e procurar uma letra em uma palavra.
#
# 1. definir dados
PALAVRA = "", # palavra original
LETRA = '_'; # uma letra
X = 1; # indice
TAMANHO = 0; # tamanho da palavra
RESPOSTA = 0; # resultado da procura
# 2. ler dados
PALAVRA = input ( " \nQual a palavra ? " );
LETRA = chr ( ord ( input ( " \nQual a letra ? " )[0] ) );
# 3. procurar letra
# determinar o tamanho da palavra original
TAMANHO = len ( PALAVRA );
while ( X < TAMANHO and RESPOSTA == 0 ):
# 3.1 testar uma letra
    if ( LETRA == PALAVRA [ X ] ):
        RESPOSTA = 1; # encontrou
    else:
# 3.2 passar para a proxima
        X = X + 1;
# fim se
# fim repetir
# 4. mostrar resultado
print ( "\nA letra \\", LETRA, "\\", end=" " );
if ( RESPOSTA == 1 ):
    print ( " EXISTE " , end=" " );
else:
    print ( " NAO EXISTE " , end=" " );
# fim se
print ( " na palavra " , PALAVRA );
#
print ( );
# pausa para terminar
print ( "\nPressionar ENTER para terminar." );
input ( );
# fim do programa
```

Exemplo 10.

Ler uma palavra do teclado, e colocar suas letras em ordem crescente.

Diagrama funcional:



Análise de dados :

- Dados do problema :

Dado	Tipo	Valor inicial	Função
PALAVRA	caractere(20)	-	armazenar palavra original
I	inteiro	0	índice
TAMANHO	inteiro	0	tamanho da palavra original
LETRA	caractere	-	auxiliar para troca

- Resultados do problema :

Dado	Tipo	Valor inicial	Função
ORDENADA	caractere(20)	-	armazenar letras ordenadas

- Avaliação da solução :

Dados	Resultado
0 9 8 7 6 5 4 3 2 1	0 1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9 0	0 1 2 3 4 5 6 7 8 9

Uso da indicação de ordenação por uma chave:

- inicialmente, supor o conjunto não ordenado;
- testar um para de letras, trocar a ordem (se preciso) e verificar :

```

se não ORDENADO então
|  "houve troca, o conjunto não estava ordenado"
senão
|  "não houve troca, o conjunto está ordenado"
fim se

```

- repetir o teste de ordenação enquanto houver troca.

Uso do limite de ordenação (K) :

- inicialmente, considerar todo o tamanho do conjunto;
- a cada passo da ordenação, se houver alguma troca, é certo que o maior elemento desta vez será colocado em ordem, restando (K-1) elementos para ordenar.

- Considerações :

É interessante considerar o número de testes necessários para se ordenar  $N$  elementos.

Genericamente :  $(N-1) (N-1)$

Com os melhoramentos :  $(N-1) + (N-2) + \dots + (N-k) \quad k \geq 2$

Algoritmo:

Esboço:

Primeira versão, só comentários.

Exemplo 10	v.1
Ação	Bloco
! definir dados	1
! ler palavra	2
! ordenar letras	3
! testar todo o conjunto	3.1
! testar um par de letras	3.1.1
! trocar se fora de ordem	
! passar para a próxima	3.1.2
! mostrar resultado	4

Segunda versão, com refinamento do primeiro e do segundo bloco.

Exemplo 10	v.2
Ação	Bloco
! definir dados	1
caractere PALAVRA [ 20 ] , ! palavra original ORDENADA [ 20 ] ; ! letras ordenadas inteiro X = 0, ! índice TAMANHO = 0; ! tamanho da palavra caractere LETRA; ! auxiliar para a troca	
! ler palavra	2
tela ← "Qual a palavra ?"; PALAVRA ← teclado;	
! ordenar letras	3
! testar todo o conjunto	3.1
! testar um par de letras	3.1.1
! trocar se fora de ordem	
! passar para a próxima	3.1.2
! mostrar resultado	4

Terceira versão, com refinamento do terceiro e do quarto bloco.

Exemplo 10	v.3
Ação	Bloco
! definir dados	1
caractere PALAVRA [ 20 ], ! palavra original ORDENADA [ 20 ]; ! letras ordenadas inteiro X = 0, ! índice TAMANHO = 0; ! tamanho da palavra caractere LETRA; ! auxiliar para a troca	
! ler palavra	2
tela ← "\nQual a palavra ?"; PALAVRA ← teclado;	
! ordenar letras	3
! determinar o tamanho da palavra original	3.1
PALAVRA [ TAMANHO ] ≠ ε ?	
! aproveitar e copiar ORDENADA[TAMANHO] = PALAVRA[TAMANHO]; TAMANHO = TAMANHO + 1;	3.1.1
ORDENADA [ TAMANHO ] = ε ;	
! repetir enquanto NÃO ORDENADO	
! testar todo o conjunto	3.2
! testar um par de letras	3.2.1
! trocar se fora de ordem	
! passar para a próxima	3.2.2
! mostrar resultado	4
X = 0 : (TAMANHO-1) : 1	4.1
! mostrar uma letra por vez tela ← (ORDENADA [ X ], " ");	4.1.1

Quarta versão, com novo refinamento do terceiro bloco.

Exemplo 10		v.4
Ação		Bloco
! definir dados		1
caractere PALAVRA [ 20 ], ! palavra original ORDENADA [ 20 ]; ! letras ordenadas inteiro X = 0, ! índice TAMANHO = 0; ! tamanho da palavra caractere LETRA; ! auxiliar para a troca		
! ler palavra		2
tela ← "Qual a palavra ?"; PALAVRA ← teclado;		
! ordenar letras		3
! determinar o tamanho da palavra original		3.1
PALAVRA [ TAMANHO ] ≠ ε ?		
! aproveitar e copiar ORDENADA[TAMANHO] = PALAVRA[TAMANHO]; TAMANHO = TAMANHO + 1;		3.1.1
ORDENADA [ TAMANHO ] = ε ;		
! repetir enquanto NÃO ORDENADO		
! testar todo o conjunto		3.2
X = 2;		
X < TAMANHO ?		
! testar um par de letras		3.2.1
ORDENADA[ X ] < ORDENADA[X-1] ?	V	! trocar se fora de ordem LETRA = PALAVRA[ X ]; ORDENADA[X]=ORDENADA[X-1]; ORDENADA[ X-1 ]= LETRA;
! passar para a próxima		3.2.2
X = X + 1;		
! mostrar resultado		4
X = 0 : (TAMANHO-1) : 1		4.1
! mostrar uma letra por vez		4.1.1
tela ← (ORDENADA [ X ], " ");		



Quinta versão, com mais um refinamento do terceiro bloco.

Exemplo 10	v.5
Ação	Bloco
! definir dados	1
caractere PALAVRA [ 20 ] , ! palavra original ORDENADA [ 20 ] ; ! letras ordenadas inteiro X = 0, ! índice K = 0, ! tamanho restante ORDENADO = 0, ! indicador de repetição TAMANHO = 0; ! tamanho da palavra caractere LETRA; ! auxiliar para a troca	
! ler palavra	2
tela ← "Qual a palavra ?"; PALAVRA ← teclado;	
! ordenar letras	3
! determinar o tamanho da palavra original	3.1
PALAVRA [ TAMANHO ] ≠ ε ?	
! aproveitar e copiar ORDENADA[TAMANHO] = PALAVRA[TAMANHO]; TAMANHO = TAMANHO + 1;	3.1.1
ORDENADA [ TAMANHO ] = ε ;	
K = TAMANHO;	
! repetir enquanto NÃO ORDENADO	
K > 1 & ORDENADO = 0 ?	
ORDENADO = 1; ! testar todo o conjunto X = 2;	3.2
X < K ?	
! testar um par de letras	3.2.1
ORDENADA[ X ] < ORDENADA[X-1] ?	V ! trocar se fora de ordem LETRA = PALAVRA[ X ]; ORDENADA[X]=ORDENADA[X-1]; ORDENADA[ X-1 ] = LETRA; ORDENADO = 0;
! passar para a próxima	3.2.2
X = X + 1;	
! reduzir o tamanho para testar	3.3
K = K - 1;	
! mostrar resultado	4
X = 0 : (TAMANHO-1) : 1	4.1
! mostrar uma letra por vez tela ← (ORDENADA [ X ], " ");	4.1.1

Programa em SCILAB:

```
// Exemplo 10.
// Ler e ordenar as letras de uma palavra.
//
// 1. definir dados
PALAVRA (1:20) = " "; // palavra original
ORDENADA (1:20) = " "; // letras ordenadas
X = 1; // indice
K = 0; // tamanho restante
ORDENADO = 0; // indicador de repetição
TAMANHO = 0; // tamanho da palavra
LETRA = ' '; // uma letra
//
// 2. ler palavra
PALAVRA = input ( "Qual a palavra ? " );
// 3. ordenar letras
// 3.1. determinar o tamanho da palavra original
TAMANHO = length ( PALAVRA );
// aproveitar e copiar
ORDENADA = PALAVRA;
K = TAMANHO;
// 3.2. repetir enquanto NAO ORDENADO
while ( K > 1 & ORDENADO == 0 )
// 3.2.1. testar conjunto
ORDENADO = 1;
X = 2;
while ( X <= K )
// testar um par de letras
if ( ORDENADA ( X ) < ORDENADA ( X-1 ) )
// trocar se fora de ordem
LETRA = ORDENADA ( X );
ORDENADA ( X ) = ORDENADA ( X-1 );
ORDENADA ( X-1 ) = LETRA;
ORDENADO = 0;
end // if
// 3.2.2 passar para a proxima
X = X + 1;
end // while
// 3.3. reduzir o tamanho para testar
K = K - 1;
end // while
//
// 4. mostrar resultado
for ( X = 1 : 1 : TAMANHO )
// 4.1 mostrar uma letra por vez
printf ( " %c ", ORDENADA ( X ) );
end // for
// pausa para terminar
printf ( "\n\nPressionar ENTER para terminar." );
halt;
// fim do programa
```

OBS.:

Ao entrar com a palavra usar apóstrofes: 'abc'.

Programa em C:

```
// Exemplo 10a.
// Ler e ordenar as letras de uma palavra.
//
// bibliotecas necessárias
#include <stdio.h>
#include <stdlib.h>
//
// parte principal
//
int main (void)
{ // 1. definir dados
  char PALAVRA [ 20 ]; // palavra original
    ORDENADA [ 20 ]; // letras ordenadas
  int  X = 0,           // índice
        K = 0,          // tamanho restante
        ORDENADO = 0,   // indicador de repetição
        TAMANHO = 0;    // tamanho da palavra
  char LETRA;           // auxiliar para a troca
//
// 2. ler palavra
  printf ( "\nQual a palavra ? " );
  scanf ( "%s", PALAVRA );
//
// 3. ordenar letras
// 3.1 determinar o tamanho da palavra original
  while ( PALAVRA [ TAMANHO ] != '\0' )
  { // aproveitar e copiar
    ORDENADA [ TAMANHO ] = PALAVRA [ TAMANHO ];
    TAMANHO = TAMANHO + 1;
  } // fim while
  ORDENADA [ TAMANHO ] = '\0';
```

```

K = TAMANHO;
// repetir enquanto NAO ORDENADO
while ( K > 1 && ORDENADO == 0 )
{ // 3.2 testar conjunto
  ORDENADO = 1;
  X = 2;
  while ( X <= K )
  { // 3.2.1. testar um par de letras
    if ( ORDENADA [ X ] < ORDENADA [ X-1 ] )
    { // trocar se fora de ordem
      LETRA = ORDENADA [ X ];
      ORDENADA [ X ] = ORDENADA [ X-1 ];
      ORDENADA [ X-1 ] = LETRA;
      ORDENADO = 0;
    } // fim if
    // 3.2.2 passar para a proxima
    X = X + 1;
  } // fim while
  // reduzir o tamanho para testar
  K = K - 1;
} // fim while
//
// 4. mostrar resultado
for ( X = 0; X < (TAMANHO-1); X = X + 1 )
{ // 4.1 mostrar uma letra por vez
  printf ( "%c ", ORDENADA [ X ] );
} // fim for

// pausa para terminar
printf ( "Pressionar ENTER para terminar." );
getchar ( );
return ( 0 );
} // fim do programa

```

Outra versão com otimizações:

```
// Exemplo 10b.
// Ler e ordenar as letras de uma palavra.
//
// bibliotecas necessárias
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
//
// parte principal
//
int main (void)
{ // 1. definir dados
  char PALAVRA [ 20 ]; // palavra original
    ORDENADA [ 20 ]; // letras ordenadas
  int   X = 0,          // indice
        K = 0,          // tamanho restante
        ORDENADO = 0,   // indicador de repetição
        TAMANHO = 0;    // tamanho da palavra
  char LETRA;           // auxiliar para a troca
//
// 2. ler palavra
  printf ( "\nQual a palavra ? " );
  scanf ( "%s", PALAVRA );
//
// 3. preparar dados para ordenar letras
  // 3.1.1 determinar o tamanho da palavra original
  TAMANHO = strlen ( PALAVRA );
  // 3.1.2 copiar a palavra
  strcpy, ( ORDENADA, PALAVRA );
```

```

K = TAMANHO;
// 3.2 repetir enquanto NAO ORDENADO
while ( K > 0 && ORDENADO == 0 )
{ // 3.2.1. testar conjunto
  ORDENADO = 1;
  X = 1;
  while ( X <= K )
  { // testar um par de letras
    if ( ORDENADA [ X ] < ORDENADA [ X-1 ] )
    { // 3.2.1.1 trocar se fora de ordem
      LETRA = ORDENADA [ X ];
      ORDENADA [ X ] = ORDENADA [ X-1 ];
      ORDENADA [ X-1 ] = LETRA;
      ORDENADO = 0;
    } // fim if
    // 3.2.2 passar para a proxima
    X = X + 1;
  } // fim while
  // 3.3 reduzir o tamanho para testar
  K = K - 1;
} // fim while
//
// 4. mostrar resultado
for ( X = 0; X < (TAMANHO-1); X = X + 1 )
{ // 4.1 mostrar uma letra por vez
  printf ( "%c ", ORDENADA [ X ] );
} // fim for

// pausa para terminar
printf ( "Pressionar ENTER para terminar." );
getchar ( );
return ( 0 );
} // fim do programa

```

Programa em C++:

```
// Exemplo 10a.
// Ler e ordenar as letras de uma palavra.
//
// bibliotecas necessárias
#include <iostream>
#include <string>
using namespace std;
//
// parte principal
//
int main (void)
{ // 1. definir dados
    string PALAVRA;      // palavra original
    string ORDENADA;     // letras ordenadas
    int    X = 0,        // índice
          K = 0,        // tamanho restante
          ORDENADO = 0, // indicador de repetição
          TAMANHO = 0;  // tamanho da palavra
    char  LETRA;        // auxiliar para a troca
//
// 2. ler palavra
    cout << "\nQual a palavra ? ";
    cin  >> PALAVRA;
//
// 3. ordenar letras
    // 3.1.1 determinar o tamanho da palavra original
    TAMANHO = PALAVRA.length ( );
    // 3.1.2 copiar a palavra original
    ORDENADA = PALAVRA;
```

```

K = TAMANHO;
// repetir enquanto NAO ORDENADO
while ( K > 1 && ORDENADO == 0 )
{ // 3.2 testar conjunto
  ORDENADO = 1;
  X = 2;
  while ( X <= K )
  { // 3.2.1. testar um par de letras
    if ( ORDENADA [ X ] < ORDENADA [ X-1 ] )
    { // trocar se fora de ordem
      LETRA = ORDENADA [ X ];
      ORDENADA [ X ] = ORDENADA [ X-1 ];
      ORDENADA [ X-1 ] = LETRA;
      ORDENADO = 0;
    } // fim if
    // 3.2.2 passar para a proxima
    X = X + 1;
  } // fim while
  // reduzir o tamanho para testar
  K = K - 1;
} // fim while
//
// 4. mostrar resultado
for ( X = 0; X < (TAMANHO-1); X = X + 1 )
{ // 4.1 mostrar uma letra por vez
  cout << ORDENADA [ X ] << " ";
} // fim for

// pausa para terminar
cout << "Pressionar ENTER para terminar.";
cin.get ( );
return EXIT_SUCCESS;
} // fim do programa

```



Programa em C#:

```

/**
 * Exemplo 10
 * Ler e ordenar as letras em uma palavra.
 */
using System;

class Exemplo_10
{
    //
    // parte principal
    //
    public static void Main ( )
    {
        // 1. definir dados
        string  PALAVRA;                // palavra original
        char [ ] ORDENADA  = new char [ 10 ];    // letras ordenadas
        char    LETRA      = ' ';           // uma letra da palavra
        int     X           = 0,             // indice
                K           = 0,             // tamanho restante
                TAMANHO     = 0;             // tamanho da palavra
        bool    ORDENADO   = false;          // indicador de repeticao

        // 2. ler palavra
        Console.Write ( "\nQual a palavra ? " );
        PALAVRA = Console.ReadLine ( );
        // copiar a palavra para uma tabela de letras
        X = 0;
        while ( X < PALAVRA.Length )
        {
            ORDENADA [ X ] = PALAVRA [ X ];
            X = X + 1;
        } // fim while
    }
}

```

```

// 3. ordenar letras
// 3.1. determinar o tamanho da palavra original
TAMANHO = PALAVRA.Length;
K = TAMANHO;
// 3.2. repetir enquanto NAO ORDENADO
while ( K > 0 && ! ORDENADO )
{
    // 3.2.1. testar conjunto
    ORDENADO = true;
    X = 1;
    while ( X < K )
    {
        // 3.2.1.1. testar um par de letras
        if ( ORDENADA [ X ] < ORDENADA [ X-1 ] )
        {
            // 3.2.1.1. trocar se fora de ordem
            LETRA = ORDENADA [ X ];
            ORDENADA [ X ] = ORDENADA [ X-1 ];
            ORDENADA [ X-1 ] = LETRA;
            ORDENADO = false;
        } // fim se
        // 3.2.1.2. passar para proxima
        X = X + 1;
    } // fim while
    // 3.2.3. reduzir o tamanho para testar
    K = K - 1;
} // fim while

// 4. mostrar resultado
Console.WriteLine ( "\nPALAVRA = " + PALAVRA );
Console.Write ( "\nORDENADA = " );
for ( X = 0; X < TAMANHO; X = X+1 )
{
    // 4.1. mostrar uma letra por vez
    Console.Write ( "" + ORDENADA [ X ] );
    //
} // fim for
// pausa para terminar
Console.Write ( "\nPressionar ENTER para terminar." );
Console.ReadLine ( );
} // end Main ( )

} // fim Exemplo_10 class

```

Programa em Java:

```

/**
 * Exemplo 10
 * Ler e ordenar as letras em uma palavra.
 */

// ----- classes necessarias

// ----- definicao de classe

class Exemplo_10
{
//
// parte principal
//
    public static void main ( String [ ] args )
    {
        // 1. definir dados
        String  PALAVRA;                // palavra original
        char [ ] ORDENADA  = new char [ 10 ]; // letras ordenadas
        char    LETRA      = ' ';        // uma letra da palavra
        int     X           = 0;          // indice
                K           = 0;          // tamanho restante
                TAMANHO     = 0;          // tamanho da palavra
        boolean ORDENADO   = false;       // indicador de repeticao

        // 2. ler palavra
        System.out.print ( "\nQual a palavra ? " );
        PALAVRA = System.console( ).readLine( );
        // copiar a palavra para uma tabela de letras
        X = 0;
        while ( X < PALAVRA.length ( ) )
        {
            ORDENADA [ X ] = PALAVRA.charAt ( X );
            X = X + 1;
        } // fim while
    }
}

```

```

// 3. ordenar letras
// 3.1. determinar o tamanho da palavra original
TAMANHO = PALAVRA.length ( );
K = TAMANHO;
// 3.2. repetir enquanto NAO ORDENADO
while ( K > 0 && ! ORDENADO )
{
    // 3.2.1. testar conjunto
    ORDENADO = true;
    X = 1;
    while ( X < K )
    {
        // 3.2.1.1. testar um par de letras
        if ( ORDENADA [ X ] < ORDENADA [ X-1 ] )
        {
            // 3.2.1.1. trocar se fora de ordem
            LETRA      = ORDENADA [ X ];
            ORDENADA [ X ] = ORDENADA [ X-1 ];
            ORDENADA [ X-1 ] = LETRA;
            ORDENADO = false;
        } // fim se
        // 3.2.1.2. passar para proxima
        X = X + 1;
    } // fim while
    // 3.2.3. reduzir o tamanho para testar
    K = K - 1;
} // fim while

// 4. mostrar resultado
System.out.println ( "\nPALAVRA = " + PALAVRA );
System.out.print ( "\nORDENADA = " );
for ( X = 0; X < TAMANHO; X = X+1 )
{
    // 4.1. mostrar uma letra por vez
    System.out.print ( "" + ORDENADA [ X ] );
    //
} // fim for

// pausa para terminar
System.out.print ( "\n\nPressionar ENTER para terminar." );
System.console( ).readLine( );
} // end main ( )

} // fim Exemplo_10 class

```

Programa em Python:

```
# Exemplo 10.
# Ler e ordenar as letras de uma palavra.
#
# 1. definir dados
PALAVRA = "";          # palavra original
ORDENADA = "";          # letras ordenadas
X = 1;                  # indice
K = 0;                  # tamanho restante
ORDENADO = 0;           # indicador de repeticao
TAMANHO = 0;           # tamanho da palavra
LETRA1 = '_';           # uma letra
LETRA2 = '_';           # outra letra
#
# 2. ler palavra
PALAVRA = input ( " \nQual a palavra ? " );
# 3. ordenar letras
# 3.1. determinar o tamanho da palavra original
TAMANHO = len ( PALAVRA );
# aproveitar e copiar
ORDENADA = PALAVRA;
K = TAMANHO;
# 3.2. repetir enquanto NAO ORDENADO
while ( K > 1 & ORDENADO == 0 ):
# 3.2.1. testar conjunto
    ORDENADO = 1;
    X = 1;
    while ( X < K ):
# testar um par de letras
        if ( ORDENADA [ X ] < ORDENADA [ X-1 ] ):
# trocar se fora de ordem
            LETRA1 = ORDENADA [ X ];
            LETRA2 = ORDENADA [ X-1 ];
            ORDENADA = ORDENADA [:(X-1)] + LETRA1 + LETRA2 + ORDENADA [(X+1):];
            ORDENADO = 0;
        # fim se
    # 3.2.2 passar para a proxima
    X = X + 1;
# fim repetir
# 3.3. reduzir o tamanho para testar
K = K - 1;
# fim repetir
#
# 4. mostrar resultado
for X in range ( 0, TAMANHO, 1 ):
# 4.1 mostrar uma letra por vez
    print ( " ", ORDENADA [ X ], " ", end=" " );
# fim repetir
# pausa para terminar
print ( "\nPressionar ENTER para terminar." );
input ( );
# fim do programa
```

### Exercícios propostos.

1. Fazer um algoritmo para :

- ler uma matriz real A, de dimensões M x N, (M < 20, N < 50);
- ler os valores de m e n do teclado;
- ler os elementos de cada linha da matriz;
- mostrar a matriz A e a sua transposta.

2. Fazer um algoritmo para :

- ler uma matriz inteira (10 x 10);
- imprimir a matriz B;
- calcular e mostrar a soma dos elementos situados abaixo da diagonal principal, incluindo os elementos da própria diagonal.

3. Fazer um algoritmo para :

- ler um valor inteiro (N < 20);
- ler os coeficientes reais do vetor (A) com (N) elementos;
- ler valores reais para (X), o último será zero;
- para cada (X), calcular e mostrar o valor de P(x) :

$$P(x) = a_1x^{n-1} + a_2x^{n-2} + \dots + a_{n-1}x^1 + a_n$$

4. Fazer um algoritmo para :

- ler um conjunto de 100 notas, com valor de 1 a 100;
- calcular e mostrar uma tabela com os valores das notas e suas respectivas frequência absoluta (quantas vezes aparece) e relativa (absoluta dividida pelo total de notas).

5. Fazer um algoritmo para :

- corrigir 500 provas de múltipla escolha com 30 questões, cada questão tem um peso diferente;
- os dados estão dispostos da seguinte forma :
  - os primeiros contém o gabarito da prova;
  - a seguir, estão os valores de cada questão;
  - os demais têm o número do aluno e suas respostas;
- calcular a nota de cada aluno e mostrar, para cada aluno, seu número e sua nota.

6. O desvio padrão é uma medida estatística que permite verificar a distribuição de um conjunto de dados. Pode-se afirmar, por exemplo, que 67% dos valores de um conjunto estão no intervalo "média-dp" e "média+dp", sendo "med" igual à média aritmética dos dados e "dp" igual ao desvio padrão.

Fazer um algoritmo para :

- ler um conjunto (X) com 1000 números inteiros;
- calcular e mostrar a média e o desvio padrão desses números;
- calcular e mostrar quantos elementos estão no intervalo "média-dp" e "média+dp".

$$média = \frac{1}{n} \sum_{i=1}^n x_i \quad \text{e} \quad dp = \frac{1}{n+1} \left[ \sum_{i=1}^n x_i^2 - \frac{1}{n} \left( \sum_{i=1}^n x_i \right)^2 \right]$$

7. Fazer um algoritmo para :

- ler um conjunto (A) de 40 elementos e outro conjunto (B) de 50 elementos;
- calcular e mostrar todos os elementos que estiverem na interseção de A e B.

8. Fazer um algoritmo para :
- ler dois arranjos de 25 elementos inteiros cada um;
  - intercalar os elementos desses dois arranjos formando, assim, um novo arranjo com 50 elementos;
  - mostrar esse novo arranjo.
9. Dada uma matriz (10x10) lida, por linha, fazer um algoritmo para verificar se há proporcionalidade entre duas linhas consecutivas da matriz, ou seja :

$$\frac{M_{m,i}}{M_{m+1,i}} \quad \text{para } X = 1 \text{ até } 10 \text{ e } m \text{ é uma linha da matriz.}$$

Mostrar as seguintes mensagens :

- "Não existe proporcionalidade entre linhas da matriz.", ou
  - "Existe proporcionalidade entre linhas da matriz".
10. Uma certa firma fez uma pesquisa de mercado. Para isto, confeccionou-se um questionário composto de 10 perguntas a serem respondidas com um sim ou não, cada uma. Após coletadas as respostas, foram fornecidos o número de pessoas entrevistadas, e as 10 respostas de cada pessoa (S=sim, N=não).
- Fazer um algoritmo para mostrar em uma linha, com base nos dados coletados, para cada pergunta: o número da pergunta, quantas pessoas responderam "sim", a porcentagem de respostas afirmativas em relação ao total de respostas, a preferência ou se houve empate.
11. Fazer um algoritmo para :
- ler um número na base 10 e convertê-lo para a base 16.
12. Fazer um algoritmo para :
- ler uma cadeia de caracteres representando um número hexadecimal e convertê-lo para a base 10.
13. Fazer um algoritmo para :
- ler duas matrizes reais (10 x 10);
  - definir um procedimento para multiplicar as duas matrizes.
14. Fazer um algoritmo para :
- ler um conjunto de 10 elementos;
  - definir uma função para retornar o maior elemento.
15. Fazer um algoritmo para :
- ler uma matriz real (10 x 10);
  - definir uma função lógica capaz de dizer se a matriz é simétrica (elementos iguais em posições opostas, em relação à diagonal), ou não.

16. Fazer um algoritmo para :
- ler um conjunto de 20 elementos reais;
  - calcular e mostrar o valor de S, dado por :

$$S = (a_1 - a_{20}^2) + (a_2 - a_{19}^2) + \cdots + (a_{10} - a_{11}^2) = \sum_{k=1}^{10} (a_k - a_{21-k}^2)$$

17. Escrever uma função para verificar se um caractere é uma letra maiúscula.  
Usar a função para testar (N) símbolos lidos do teclado, um por vez.
18. Escrever uma função para verificar se um caractere é uma letra minúscula.  
Usar a função para testar (N) símbolos lidos do teclado, um por vez.
19. Escrever uma função para verificar se um caractere é uma letra (maiúscula ou minúscula).  
Usar a função para testar (N) símbolos lidos do teclado, um por vez.
20. Escrever uma função para verificar se um caractere é um algarismo entre 0 e 9.  
Usar a função para testar (N) símbolos lidos do teclado, um por vez.
21. Escrever uma função para verificar se um caractere é um espaço em branco.  
Usar a função para testar (N) símbolos lidos do teclado, um por vez.
22. Escrever uma função para verificar se um caractere é um sinal de pontuação { . , ; : }  
Usar a função para testar (N) símbolos lidos do teclado, um por vez.
23. Escrever uma função para verificar se um caractere é um operador aritmético { - + \* / % }  
Usar a função para testar (N) símbolos lidos do teclado, um por vez.
24. Escrever uma função para verificar se um caractere é um operador lógico { & | ! ^ }  
Usar a função para testar (N) símbolos lidos do teclado, um por vez.
25. Escrever uma função para calcular o tamanho de uma cadeia de caracteres.
26. Escrever uma função para verificar se existe uma determinada letra em uma cadeia de caracteres.
27. Escrever uma função para verificar se existe uma determinada letra em uma cadeia de caracteres e informar em qual posição aparece pela primeira vez; se não existir, informar posição igual a zero.



28. Escrever uma função para verificar se existe uma determinada letra em uma cadeia de caracteres e informar em qual posição aparece pela última vez; se não existir, informar posição igual a zero.
29. Escrever um procedimento para unir duas cadeias de caracteres formando uma terceira.
30. Escrever uma função para tirar a primeira letra de uma cadeia de caracteres.
31. Escrever uma função para tirar a última letra de uma cadeia de caracteres.
32. Escrever uma função para contar o número de espaços em branco de uma palavra.
33. Escrever uma função para contar o número de vogais de uma palavra.
34. Escrever um procedimento para eliminar espaços em branco no início de uma palavra.
35. Escrever um procedimento para eliminar espaços em branco no final de uma palavra.
36. Escrever um procedimento para eliminar espaços em branco duplicados no meio de uma palavra.
37. Escrever um procedimento para eliminar qualquer letra duplicada no meio de uma palavra.
38. Escrever um procedimento para apagar última letra de uma cadeia de caracteres.
39. Escrever um procedimento para apagar (n) letras de uma cadeia de caracteres, a partir de uma determinada posição.
40. Escrever um procedimento para inserir (n) letras de uma cadeia de caracteres, a partir de uma determinada posição.
41. Escrever uma função lógica para receber como parâmetro uma cadeia de caracteres representando uma data, e dizer se é válida, ou não.
42. Escrever uma função inteira para receber como parâmetro uma cadeia de caracteres representando uma data, e retornar o número de dias desde o início do ano, se for uma data válida.
43. Escrever uma função inteira para receber duas datas como parâmetros e retornar a diferença, em número de dias, entre elas.

44. Escrever uma função inteira para receber uma seqüência de caracteres com Algarismos e convertê-los para o decimal equivalente.
45. Escrever uma função inteira para receber uma seqüência de caracteres com Algarismos e convertê-los para o binário equivalente.
46. Escrever uma função inteira para receber uma seqüência de caracteres com Algarismos e convertê-los para uma base qualquer, fornecida como parâmetro.
47. Escrever uma função real para receber uma seqüência de caracteres com Algarismos e sinal, e convertê-los para o decimal equivalente.
48. Escrever uma função real para receber uma seqüência de caracteres com Algarismos e sinal, e convertê-los para o binário equivalente.
49. Pode-se calcular a raiz quadrada de um número positivo através do método de aproximação sucessivas de Newton, descrito a seguir :
- seja "a" o número do qual deseja-se obter a raiz quadrada;
  - a primeira aproximação para a raiz quadrada será dada por :

$$x_1 = a / 2$$

- a próxima ou sucessiva aproximação é dada por :

$$x_{n+1} = \frac{(x_n^2 + a)}{2x_n}$$

- fazer um algoritmo para :
- ler o valor de "a" do teclado e calcular e mostrar a 25ª. aproximação.

50. A conversão de graus Fahrenheit para Centígrados é obtida por :

$$C = 5 (F - 32) / 9$$

- fazer uma função para calcular e mostrar uma tabela de graus Centígrados em função de graus Fahrenheit, que variem de 50 a 150 de 1 em 1.

51. Fazer um procedimento para gerar e mostrar a seguinte seqüência e armazenar em uma tabela:

$$289 - 256 - 225 - 196 - \dots - 9 - 4 - 1$$

52. Fazer um algoritmo para :
- ler dois números escritos na base (4);
  - calcular e mostrar a soma destes números, usando um procedimento de conversão.

53. Fazer um algoritmo para:
- ler um número inteiro, na base 10, de até 5 dígitos;
  - transformar esse número da base 10 para a base 2, usando um procedimento que converta a saída em uma cadeia de caracteres;
  - mostrar o número na base 10 e na base 2.
54. Fazer um algoritmo para :
- ler uma cadeia de caracteres expressando um número inteiro, na base 3;
  - transformar esse número da base 3 para a base 10; usando um procedimento que converta a entrada em um número inteiro;
  - mostrar o número na base 3 e na base 10.
55. Fazer um algoritmo para :
- ler do teclado um conjunto de palavras, uma por vez;
  - a última palavra será "FIM";
  - contar o número de vezes em que as palavras "E", "OU" e "NÃO" aparecem neste conjunto, usando uma função para verificar a existência.
56. Fazer um algoritmo para :
- ler do teclado duas cadeias de caracteres;
  - intercalar os símbolos de cada cadeia.
57. Fazer um algoritmo para :
- montar uma matriz de translação para os eixos cartesianos x-y-z;
  - ler um vetor de posição [ x y z ] e multiplicá-lo pela matriz de translação.
58. Fazer um algoritmo para :
- montar uma matriz de rotação em relação aos eixos cartesianos x-y;
  - ler um vetor de posição [ x y z ] e multiplicá-lo pela matriz de rotação.
59. Fazer um algoritmo para :
- montar uma matriz de rotação em relação aos eixos cartesianos y-z;
  - ler um vetor de posição [ x y z ] e multiplicá-lo pela matriz de rotação.
60. Fazer um algoritmo para :
- montar uma matriz de rotação em relação aos eixos cartesianos x-z;
  - ler um vetor de posição [ x y z ] e multiplicá-lo pela matriz de rotação.