Universidade Federal de Ouro Preto – UFOP
Instituto de Ciências Exatas e Biológicas – ICEB
Departamento de Computação – DECOM
Disciplina: Algoritmos e Estrutura de Dados I – CIC102
Professor: David Menotti (menottid@gmail.com)

Lista de Exercícios 04 – Estruturas de Dados Homogêneas - Vetores

1) Escreva um algoritmo em PORTUGOL que armazene em um vetor todos os números inteiros de 0 a 50. Após isso, o algoritmo deve imprimir todos os valores armazenados.

```
algoritmo L4P01;
var
  inteiro: C, VET[0..50];
início
  para C de 0 até 50 faça
    VET[C] <- C;
  fim-para
  para C de 0 até 50 faça
    imprima(VET[C]);
  fim-para
fim
```

```
program l4p01;
var
  C: integer;
  VET: array [0..50] of integer;
begin
  for C := 0 to 50 do
    VET[C] := C;
  for C := 0 to 50 do
    write(VET[C],' ');
  writeLn('');
end.
```

```
for C = 0 : 50
  VET(C+1) = C;
end
for C = 0 : 50
  fprintf(1,'%d ',VET(C+1));
end
fprintf(1,'\n');
```

2) Escreva um algoritmo em PORTUGOL que armazene em um vetor todos os números inteiros do intervalo fechado de 1 a 100. Após isso, o algoritmo deve imprimir todos os valores armazenados.

```
algoritmo L4P02;
var
  inteiro: C, VET[1..100];
início
  para C de 1 até 100 faça
    VET[C] <- C;
  fim-para
  para C de 1 até 100 faça
    imprima(VET[C]);
  fim-para
fim
```

```
program l4p02;
var
  C: integer;
  VET: array [1..100] of integer;
begin
  for C := 1 to 100 do
    VET[C] := C;
  for C := 1 to 100 do
    write(VET[C],' ');
  writeLn('');
end.
```

```
for C = 1 : 100
  VET(C) = C;
end
for C = 1 : 100
  fprintf(1,'%d ',VET(C));
end
fprintf(1,'\n');
```

3) Escreva um algoritmo em PORTUGOL que armazene em um vetor todos os números inteiros de 100 a 1 (em ordem decrescente). Após isso, o algoritmo deve imprimir todos os valores armazenados.

```
algoritmo L4P03;
var
  inteiro: C, VET[1..100];
início
  para C de 100 até 1 passo -1 faça
    VET[100-C+1] <- C;
  fim-para
  para C de 1 até 100 faça
    imprima(VET[C]);
  fim-para
fim
```

```
program l4p03;
var
  C: integer;
  VET: array [1..100] of integer;
begin
  for C := 100 downto 1 do
    VET[100-C+1] := C;
  for C := 1 to 100 do
    write(VET[C],' ');
  writeLn('');
end.
```

```
for C = 100 : -1 : 1
  VET(100-C+1) = C;
end
for C = 1 : 100
  fprintf(1,'%d ',VET(C));
end
fprintf(1,'\n');
```

4) Escreva um algoritmo em PORTUGOL que armazene em um vetor todos os números inteiros de 100 a 200. Após isso, o algoritmo deve imprimir todos os valores armazenados.

```
algoritmo L4P04;
var
  inteiro: C,VET[100..200];
início
  para C de 100 até 200 faça
    VET[C] <- C;
  fim-para
  para C de 100 até 200 faça
    imprima(VET[C]);
  fim-para
fim
```

```
program l4p04;
var
  C: integer;
  VET: array [100..200] of integer;
begin
  for C := 100 to 200 do
    VET[C] := C;
  for C := 100 to 200 do
    write(VET[C],' ');
  writeLn('');
end.
```

```
for C = 100 : 200
  VET(C) = C;
end
for C = 100 : 200
  fprintf(1,'%d ',VET(C));
end
fprintf(1,'\n');
```

5) Escreva um algoritmo em PORTUGOL que armazene em um vetor todos os números inteiros de 200 a 100 (em ordem decrescente). Após isso, o algoritmo deve imprimir todos os valores armazenados.

```
algoritmo L4P05;
var
  inteiro: C, VET[0..100];
início
  para C de 200 até 100 passo -1 faça
    VET[200-C] <- C;
  fim-para
  para C de 0 até 100 faça
    imprima(VET[C]);
  fim-para
fim
```

```
program l4p05;
var
  C: integer;
  VET: array [0..100] of integer;
begin
  for C := 200 downto 100 do
    VET[200-C] := C;
  for C := 0 to 100 do
    write(VET[C],' ');
  writeLn('');
end.
```

```
for C = 200 : -1 : 100
  VET(200-C+1) = C;
end
for C = 0 : 100
  fprintf(1,'%d ',VET(C+1));
end
fprintf(1,'\n');
```

6) Escreva um algoritmo em PORTUGOL que armazene em um vetor todos os números múltiplos de 5, no intervalo fechado de 1 a 500. Após isso, o algoritmo deve imprimir todos os valores armazenados.

```
algoritmo L4P06;
var
  inteiro: C, VET[1..100];
início
  para C de 5 até 500 passo 5 faça
    VET[C div 5] <- C;
  fim-para
  para C de 1 até 100 faça
    imprima(VET[C]);
  fim-para
fim
```

```
program l4p06;
var
  C: integer;
  VET: array [1..100] of integer;
begin
  for C := 5 to 500 do
    if C mod 5 = 0 then
      VET[C div 5 ] := C;
  for C := 1 to 100 do
    write(VET[C],' ');
  writeLn('');
end.
```

```
for C = 5 : 5 : 500
  VET(floor(C/5)) = C;
end
for C = 1 : 100
  fprintf(1,'%d ',VET(C));
end
fprintf(1,'\n');
```

7) Escreva um algoritmo em PORTUGOL que armazene em um vetor todos os números pares do intervalo fechado de 1 a 100. Após isso, o algoritmo deve imprimir todos os valores armazenados.

```
algoritmo L4P07;
var
  inteiro: C, VET[1..50];
início
  para C de 2 até 100 passo 2 faça
    VET[C div 2] <- C;
  fim-para
  para C de 1 até 50 faça
    imprima(VET[C]);
  fim-para
fim
```

```
program l4p07;
var
  C: integer;
  VET: array [1..50] of integer;
begin
  for C := 2 to 100 do
    if C mod 2 = 0 then
      VET[C div 2] := C;
  for C := 1 to 50 do
    write(VET[C],' ');
  writeLn('');
end.
```

```
for C = 2 : 2 : 100
  VET(floor(C/2)) = C;
end
for C = 1 : 50
  fprintf(1,'%d ',VET(C));
end
fprintf(1,'\n');
```

8) Escreva um algoritmo em PORTUGOL que armazene em um vetor os 100 primeiros números ímpares. Após isso, o algoritmo deve imprimir todos os valores armazenados.

```
algoritmo L4P08;
var
  inteiro: C, VET[1..100];
início
  para C de 1 até 200 passo 2 faça
    VET[C div 2] <- C;
  fim-para
  para C de 1 até 200 faça
    imprima(VET[C]);
  fim-para
fim
```

```
program l4p08;
var
  C: integer;
  VET: array [1..100] of integer;
begin
  for C := 1 to 200 do
    if C mod 2 = 1 then
      VET[C div 2 + 1] := C;
  for C := 1 to 100 do
    write(VET[C],' ');
  writeLn('');
end.
```

```
for C = 1 : 2 : 200
  VET( floor(C/2) + 1) = C;
end
for C = 1 : 100
  fprintf(1,'%d ',VET(C));
end
fprintf(1,'\n');
```

9) Escreva um algoritmo em PORTUGOL que armazene em um vetor o quadrado dos números ímpares no intervalo fechado de 1 a 20. Após isso, o algoritmo deve imprimir todos os valores armazenados.

```
algoritmo L4P09;
var
  inteiro: C,VET[1..10];
início
  para C de 1 até 20 passo 2 faça
    VET[C div 2 + 1] <- C*C;
  fim-para
  para C de 1 até 10 faça
    imprima(VET[C]);
  fim-para
fim
```

```
program l4p09;
var
  C: integer;
  VET: array [1..10] of integer;
begin
  for C := 1 to 20 do
    if C mod 2 = 1 then
      VET[C div 2 + 1] := C*C;
  for C := 1 to 10 do
    write(VET[C],' ');
  writeLn('');
end.
```

```
for C = 1 : 2 : 20
  VET( floor(C/2) + 1 ) = C*C;
end
for C = 1 : 10
  fprintf(1,'%d ',VET(C));
end
fprintf(1,'\n');
```

10) Escreva um algoritmo em PORTUGOL que armazene em um vetor todos os números ímpares do intervalo fechado de 1 a 100. Após isso, o algoritmo deve imprimir todos os valores armazenados.

```
algoritmo L4P10;
var
  inteiro: C, VET[1..50];
início
  para C de 1 até 100 passo 2 faça
    VET[C div 2 + 1] <- C;
  fim-para
  para C de 1 até 50 faça
    imprima(VET[C]);
  fim-para
fim
```

```
program l4p10;
var
  C: integer;
  VET: array [1..50] of integer;
begin
  for C := 1 to 100 do
    if C mod 2 = 1 then
      VET[C div 2 + 1] := C;
  for C := 1 to 50 do
    write(VET[C],' ');
  writeLn('');
end.
```

```
for C = 1 : 2 : 100
  VET( floor(C/2) + 1 ) = C;
end
for C = 1 : 50
  fprintf(1,'%d ',VET(C));
end
fprintf(1,'\n');
```

11) Escreva um algoritmo em PORTUGOL que receba dez números do usuário e armazene em um vetor a metade de cada número. Após isso, o algoritmo deve imprimir todos os valores armazenados.

```
algoritmo L4P11;
var
  inteiro: C;
  real:    VALOR, METADE[1..10];
início
  para C de 1 até 10 faça
    leia(VALOR);
    METADE[C] <- VALOR / 2;
  fim-para
  para C de 1 até 10 faça
    imprima(METADE[C]);
  fim-para
fim
```

```
program l4p11;
var
  C: integer;
  VALOR: real;
  METADE: array [1..10] of real;
begin
  for C := 1 to 10 do
  begin
    write('Digite um valor: ');
    readLn(VALOR);
    METADE[C] := VALOR / 2;
  end;
  for C := 1 to 10 do
    write(METADE[C]:0,' ');
  writeLn('');
end.
```

```
for C = 1 : 10
  VALOR = input('Digite um valor: ');
  METADE(C) = VALOR / 2;
end
for C = 1 : 10
  fprintf(1,'%d ',METADE(C));
end
fprintf(1,'\n');
```

12) Escreva um algoritmo em PORTUGOL que receba dez números do usuário e armazene em um vetor o quadrado de cada número. Após isso, o algoritmo deve imprimir todos os valores armazenados.

```
algoritmo L4P12;
var
  inteiro: C;
  real:    VALOR, QUAD[1..10];
início
  para C de 1 até 10 faça;
    leia(VALOR);
    QUAD[C] <- VALOR ** 2;
  fim-para
  para C de 1 até 10 faça
    imprima(QUAD[C]);
  fim-para
fim
```

```
program l4p12;
var
  C: integer;
  VALOR: real;
  QUAD: array [1..10] of real;
begin
  for C := 1 to 10 do
  begin
    write('Digite um valor: ');
    readLn(VALOR);
    QUAD[C] := VALOR * VALOR;
  end;
  for C := 1 to 10 do
    write(QUAD[C],' ');
  writeLn('');
end.
```

```
for C = 1 : 10
  VALOR = input('Digite um valor: ');
  QUAD(C) = VALOR * VALOR;
end
for C = 1 : 10
  fprintf(1,'%d ',QUAD(C));
end
fprintf(1,'\n');
```

13) Escreva um algoritmo em PORTUGOL que receba dez números do usuário e armazene em um vetor o cubo de cada número. Após isso, o algoritmo deve imprimir todos os valores armazenados.

```
algoritmo L4P13;
var
  inteiro: C;
  real:    VALOR, CUB[1..10];
início
  para C de 1 até 10 faça
    leia(VALOR);
    CUB[C] <- VALOR ** 3;
  fim-para
  para C de 1 até 10 faça
    imprima(CUB[C]);
  fim-para
fim
```

```
program l4p13;
var
  C: integer;
  VALOR: real;
  CUB: array [1..10] of real;
begin
  for C := 1 to 10 do
  begin
    write('Digite um valor: ');
    readLn(VALOR);
    CUB[C] := VALOR * VALOR * VALOR;
  end;
  for C := 1 to 10 do
    write(CUB[C],' ');
  writeLn('');
end.
```

```
for C = 1 : 10
  VALOR = input('Digite um valor: ');
  CUB(C) = VALOR * VALOR * VALOR;
end
for C = 1 : 10
  fprintf(1,'%d ',CUB(C));
end
fprintf(1,'\n');
```

14) Escreva um algoritmo em PORTUGOL que receba quinze números do usuário e armazene em um vetor a raiz quadrada de cada número. Caso o valor digitado seja menor que zero o número –1 deve ser atribuído ao elemento do vetor. Após isso, o algoritmo deve imprimir todos os valores armazenados.

```
algoritmo L4P14;
var
  inteiro: C;
  real:    VALOR, RAIZ[1..15];
início
  para C de 1 até 15 faça
    leia(VALOR);
    se ( VALOR >= 0 ) então
      RAIZ[C] <- raiz(VALOR);
    senão
      RAIZ[C] <- -1;
      imprima("Não é possível calcular a raiz quadrada! Número negativo!");
    fim-se
  fim-para
  para C de 1 até 10 faça
    imprima(RAIZ[C]);
  fim-para
fim
```

```
program l4p14;
var
  C: integer;
  VALOR: real;
  RAIZ: array [1..15] of real;
begin
  for C := 1 to 15 do
  begin
    write('Digite um valor: ');
    readLn(VALOR);
    if ( VALOR >= 0 ) then
      RAIZ[C] := SqRt(VALOR)
    else begin
      RAIZ[C] := -1;
      write('Nao e possivel calcular a raiz quadrada! Numero negativo!'); end;
  end;
  for C := 1 to 15 do
    write(RAIZ[C],' ');
  writeLn('');
end.
```

```
for C = 1 : 15
  VALOR = input('Digite um valor: ');
  if ( VALOR >= 0 )
    RAIZ(C) = sqrt(VALOR);
  else
    RAIZ(C) = -1;
    disp('Nao e possivel calcular a raiz quadrada! Numero negativo!');
  end
end
for C = 1 : 15
  fprintf(1,'%d ',RAIZ(C));
end
fprintf(1,'\n');
```

15) Escreva um algoritmo em PORTUGOL que receba oito números do usuário e armazene em um vetor o logaritmo de cada um deles na base 10. Caso não seja possível calcular o valor para o número digitado, o número –1 deve ser atribuído ao elemento do vetor. Após isso, o algoritmo deve imprimir todos os valores armazenados.

```
algoritmo L4P15;
var
  inteiro: C;
  real:    VALOR, LG[1..10];
início
  para C de 1 até 10 faça
    leia(VALOR);
    se ( VALOR > 0 ) então
      LG[C] <- log(VALOR) / log(10);
    senão
      LG[C] <- -1;
      imprima("Não é possível calcular o logaritmo! Número negativo ou zero!");
    fim-se
  fim-para
  para C de 1 até 10 faça
    imprima(LG[C]);
  fim-para
fim
```

```
program l4p15;
var
  C: integer;
  VALOR: real;
  LG: array [1..10] of real;
begin
  for C := 1 to 10 do
  begin
    write('Digite um valor: ');
    readLn(VALOR);
    if ( VALOR > 0 ) then
      LG[C] := Ln(VALOR) / Ln(10)
    else begin
      LG[C] := -1;
      write('Nao eh possivel calcular o logartimo! Numero negativo ou zero!'); end;
  end;
  for C := 1 to 10 do
    write(LG[C],' ');
  writeLn('');
end.
```

```
for C = 1 : 8
  VALOR = input('Digite um valor: ');
  if ( VALOR > 0 )
    LG(C) = log10(VALOR);  % ln(VALOR) / ln(10);
  else
    LG(C) = -1;
    disp('Nao eh possivel calcular o logartimo! Numero negativo ou zero!');
  end
end
for C = 1 : 8
  fprintf(1,'%f ',LG(C));
end
fprintf(1,'\n');
```

16) Escreva um algoritmo em PORTUGOL que receba a altura de 10 atletas. Esse algoritmo deve imprimir a altura daqueles atletas que tem altura maior que a média.

```
algoritmo L4P16;
var
  inteiro: C;
  real:    ALTURA, ,SOMA, MEDIA, VETALT[1..10];
início
  para C de 1 até 10 faça
    leia(ALTURA);
    VETALT[C] <- ALTURA;
  fim-para
  SOMA <- 0;
  para C de 1 até 10 faça
    SOMA <- SOMA + VETALT[C];
  fim-para
  MEDIA <- SOMA / 10;
  para C de 1 até 10 faça
    se ( VETALT[C] > MEDIA ) então
      imprima(VETALT[C]);
    fim-se
  fim-para
fim
```

```
algoritmo L4P16B;
var
  inteiro: C;
  real:    ALTURA, ,SOMA, MEDIA, VETALT[1..10];
início
  SOMA <- 0;
  para C de 1 até 10 faça
    leia(ALTURA);
    VETALT[C] <- ALTURA;
    SOMA <- SOMA + VETALT [C];
  fim-para
  MEDIA <- SOMA / 10;
  para C de 1 até 10 faça
    se ( VETALT [C] > MEDIA ) então
     imprima(VETALT [C]);
    fim-se
  fim-para
fim
```

```
program l4p16;
var
  C: integer;
  ALTURA, SOMA, MEDIA: real;
  VETALT: array [1..10] of real;
begin
  for C := 1 to 10 do
  begin
    write('Digite a altura: ');
    readLn(ALTURA);
    VETALT[C] := ALTURA;
  end;
  SOMA := 0;
  for C := 1 to 10 do
    SOMA := SOMA + VETALT[C];
  MEDIA := SOMA / 5;
  for C := 1 to 10 do
    if ( VETALT[C] > MEDIA ) then
      writeLn(VETALT[C]:3:2);
end.
```

```
program l4p16b;
var
  C: integer;
  ALTURA, SOMA, MEDIA: real;
  VETALT: array [1..10] of real;
begin
  SOMA := 0;
  for C := 1 to 10 do
  begin
    write('Digite a altura: ');
    read(ALTURA);
    VETALT[C] := ALTURA;
    SOMA := SOMA + VETALT [C];
  end;
  MEDIA := SOMA / 10;
  for C := 1 to 10 do
  begin
    if ( VETALT [C] > MEDIA ) then
     writeLn(VETALT[C]:3:2);
  end;
end.
```

```
for C = 1 : 10
  ALTURA = input('Digite a altura: ');
  VETALT(C) = ALTURA;
end
SOMA = 0;
for C = 1 : 10
  SOMA = SOMA + VETALT(C);
end
MEDIA = SOMA / 10;
for C = 1 : 10
  if ( VETALT(C) > MEDIA )
    fprintf(1,'%d ',VETALT(C));
  end
end
fprintf(1,'\n');
```

```
SOMA = 0;
for C = 1 : 10
  ALTURA = input('Digite a altura: ');
  VETALT(C) = ALTURA;
  SOMA = SOMA + VETALT(C);
end
MEDIA = SOMA / 10;
for C = 1 : 10
  if ( VETALT(C) > MEDIA )
    fprintf(1,'%d ',VETALT(C));
  end
end
fprintf(1,'\n');
```

17) A série de Fibonacci é formada pela seqüência:

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ...

Escreva um algoritmo em PORTUGOL que armazene em um vetor os 50 primeiros termos da série de FIBONACCI. Após isso, o algoritmo deve imprimir todos os valores armazenados.

```
algoritmo L4P17;
var
  inteiro: C, ATUAL, ANT1, ANT2, VET[1..50];
início
  ANT2 <- 1;
  ANT1 <- 1;
  VET[1] <- 1;
  VET[2] <- 1;
  para C de 3 até 50 faça
    ATUAL <- ANT1 + ANT2;
    VET[C] <- ATUAL;
    ANT2 <- ANT1;
    ANT1 <- ATUAL;
  fim-para
  para C de 1 até 50 faça
   imprima(VET[C]);
  fim-para
fim
```

```
program l4p17;
var
  C: integer;
  ATUAL, ANT1, ANT2: real;
  VET: array [1..50] of real;
begin
  ANT2 := 1;
  ANT1 := 1;
  VET[1] := 1;
  VET[2] := 1;
  for C := 3 to 50 do
  begin
    ATUAL := ANT1 + ANT2;
    VET[C] := ATUAL;
    ANT2 := ANT1;
    ANT1 := ATUAL;
  end;
  for C := 1 to 50 do
    write(VET[C]:0:0,' ');
  writeLn('');
end.
```

```
ANT2 = 1;
ANT1 = 1;
VET(1) = 1;
VET(2) = 1;
for C = 3 : 50
  ATUAL = ANT1 + ANT2;
  VET(C) = ATUAL;
  ANT2 = ANT1;
  ANT1 = ATUAL;
end
for C = 1 : 50
  fprintf(1,'%d ',VET(C));
end
fprintf(1,'\n');
```

18) Implementar um algoritmo em PORTUGOL para calcular o sen(X). O valor de X deverá ser digitado em graus. O valor do seno de X será calculado pela soma dos 15 primeiros termos da série a seguir:

$$\text{sen}(X) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \frac{x^{11}}{11!} + \ldots$$

Esses termos devem ser armazenados em um vetor de reais.

```
algoritmo L4P18;
var
  inteiro: I, J, FAT;
  real:    X, SN, VET[1..15];
início
  SN <- 0;
  leia(X);
  X <- X * pi/180;
  para I de 1 até 15 faça
    FAT <- 1;
    para J de 2 até 2*I - 1 faça
      FAT <- FAT * J;
    fim-para
    se ( I mod 2 = 0 ) então
      SN <- SN - ( X ** ( 2 * I - 1 ) ) / FAT; { termo par }
      VET[I] <- - ( X ** ( 2 * I - 1 ) ) / FAT;
    senão
      SN <- SN + ( X ** ( 2 * I - 1 ) ) / FAT; { termo ¡mpar }
      VET[I] <- + ( X ** ( 2 * I - 1 ) ) / FAT;
    fim-se
  fim-para
  imprima("SEN(",X * 180/pi,") = ",SN);
fim
```

```
program l4p18;
var
  I, J: integer;
  FAT, X, XE, SN: real;
  VET: array [1..15] of real;
begin
  SN := 0;
  writeLn('Sen(x)');
  write('Digite o valor de x: ');
  readLn(X);
  X := 0.01745329252 * X;  {X * pi/180}
  XE := X;
  for I := 1 to 15 do
  begin
    FAT := 1;
    for J := 2 to 2*I - 1 do
      FAT := FAT * J;
    if ( I mod 2 = 0 ) then begin
      SN := SN - XE / FAT; { termo par }
      VET[I] := - XE / FAT; end
    else begin
      SN := SN + XE / FAT; { termo ¡mpar }
      VET[I] := + XE / FAT; end;
    XE := XE * X * X;
  end;
  writeLn('Sen(',X / 0.01745329252:5:4,') = ',SN:5:4);
end.
```

```
SN = 0;
disp('Sen(x)');
X = input('Digite o valor de x: ');
XE = X;
X = X * pi/180;
for I = 1 : 15
  FAT = 1;
  for J = 2 : 2*I - 1
    FAT = FAT * J;
  end
  if ( mod(I,2) == 0 )
    SN = SN - ( X ^ (2*I-1) ) / FAT; % termo par
    VET(I) = - ( X ^ (2*I-1) ) / FAT;
  else
    SN = SN + ( X ^ (2*I-1) ) / FAT; % termo impar
    VET(I) = + ( X ^ (2*I-1) ) / FAT;
  end
end
fprintf(1,'Sen(%f) = %f\n',XE,SN);
```

19) Escreva um algoritmo em PORTUGOL, que leia um conjunto de 50 fichas correspondente à alunos e armazene-as em vetores, cada uma contendo, a altura e o código do sexo de uma pessoa (código = 1 se for masculino e 2 se for feminino), e calcule e imprima:

- A maior e a menor altura da turma;
- As mulheres com altura acima da média da altura das mulheres;
- As pessoas com altura abaixo da média da turma.

```
algoritmo L4P19;
var
  inteiro: C, CODSEXO, NMULHER;
  real:    ALTURA, MAIOR, MENOR;
  real:    VETALT[1..50], VETSEX[1..50];
  real:    SOMAMULHER, MEDIAMULHER;
  real:    SOMATURMA, MEDIATURMA;
início
  para C de 1 até 50 faça
    leia(ALTURA);
    leia(CODSEXO);
    VETALT[C] <- ALTURA;
    VETSEX[C] <- CODSEXO;
  fim-para
  NMULHER    <- 0;
  SOMAMULHER <- 0;
  SOMATURMA  <- 0;
  MAIOR <- VETALT[1];
  MENOR <- VETALT[1];
  para C de 1 até 50 faça
    se ( VETALT[C] > MAIOR ) então
      MAIOR <- VETALT[C];
    senão
      se ( VETALT[C] < MENOR ) então
        MENOR <- VETALT[C];
      fim-se
    fim-se
    se ( VETSEX[C] = 2 ) então
      NMULHER <- NMULHER + 1;
      SOMAMULHER <- SOMAMULHER + VETALT[C];
    fim-se
    SOMATURMA <- SOMATURMA + VETALT[C];
  fim-para
  MEDIAMULHER <- SOMAMULHER / NMULHER;
  MEDIATURMA  <- SOMATURMA  / 50;
  imprima("Maior altura da turma: ",MAIOR);
  imprima("Menor altura da turma: ",MENOR);
  imprima("Mulheres com altura acima da media das mulheres");
  para C de 1 até 50 faça
    se ( VETSEX[C] = 2 ) e ( VETALT[C] > MEDIAMULHER ) então
      imprima(VETALT[C]);
    fim-se
  fim-para
  imprima("Pessoas com altura abaixo da media");
  para C de 1 até 50 faça
    se ( VETALT[C] < MEDIATURMA ) então
      imprima(VETALT[C]);
    fim-se
  fim-para
fim
```

```
algoritmo L4P19B;
var
  inteiro: C, CODSEXO,NMULHER;
  real:    ALTURA,MAIOR,MENOR;
  real:    VETALT[1..50],VETSEX[1..50];
  real:    SOMAMULHER,MEDIAMULHER;
  real:    SOMATURMA,MEDIATURMA;
início
  para C de 1 até 50 faça
    leia(ALTURA);
    leia(CODSEXO);
    VETALT[C] <- ALTURA;
    VETSEX[C] <- CODSEXO;
  fim-para
  NMULHER    <- 0;
  SOMAMULHER <- 0;
  SOMATURMA  <- 0;
  MAIOR <- VETALT[1];
  MENOR <- VETALT[1];
  para C de 1 até 50 faça
    se ( VETALT[C] > MAIOR ) então
      MAIOR <- VETALT[C];
    senão
      se ( VETALT[C-1] < MENOR ) então
        MENOR <- VETALT[C];
      fim-se
    fim-se
    se ( VETSEX[C] = 2 ) então
      NMULHER <- NMULHER + 1;
      SOMAMULHER <- SOMAMULHER + VETALT[C];
    fim-se
    SOMATURMA <- SOMATURMA + VETALT[C];
  fim-para
  MEDIAMULHER <- SOMAMULHER / NMULHER;
  MEDIATURMA  <- SOMATURMA  / 50;
  imprima("Maior altura da turma: ",MAIOR);
  imprima("Menor altura da turma: ",MENOR);
  imprima("Mulheres com altura acima da media das mulheres");
  para C de 1 até 50 faça
    se ( VETSEX[C] = 2 ) e ( VETALT[C] > MEDIAMULHER ) então
      imprima(VETALT[C]);
    fim-se
  fim-para
  imprima("Pessoas com altura abaixo da media");
  para C de 1 até 50 faça
    se ( VETALT[C] < MEDIATURMA ) então
      imprima(VETALT[C]);
    fim-se
  fim-para
fim
```

```
program l4p19;
var
  C, CODSEXO, NMULHER: integer;
  ALTURA, MAIOR, MENOR: real;
  VETALT,VETSEX: array [1..50] of real;
  SOMAMULHER, MEDIAMULHER: real;
  SOMATURMA, MEDIATURMA: real;
begin
  for C := 1 to 50 do
  begin
    write('Altura: ');
    readLn(ALTURA);
    write('Sexo (1=M/2=F): ');
    readLn(CODSEXO);
    VETALT[C] := ALTURA;
    VETSEX[C] := CODSEXO;
  end;
  NMULHER    := 0;
  SOMAMULHER := 0;
  SOMATURMA  := 0;
  MAIOR := VETALT[1];
  MENOR := VETALT[1];
  for C := 1 to 50 do
  begin
    if ( VETALT[C] > MAIOR ) then
      MAIOR := VETALT[C]
    else
      if ( VETALT[C] < MENOR ) then
        MENOR := VETALT[C];
    if ( VETSEX[C] = 2 ) then begin
      NMULHER := NMULHER + 1;
      SOMAMULHER := SOMAMULHER + VETALT[C]; end;
    SOMATURMA := SOMATURMA + VETALT[C];
  end;
  MEDIAMULHER := SOMAMULHER / NMULHER;
  MEDIATURMA  := SOMATURMA  / 50;
  writeln('Maior altura da turma: ',MAIOR);
  writeln('Menor altura da turma: ',MENOR);
  writeLn('Mulheres com altura acima da media das mulheres');
  for C := 1 to 50 do
    if ( VETSEX[C] = 2 ) and ( VETALT[C] > MEDIAMULHER ) then
      write(VETALT[C],' ');
  writeLn('');
  writeLn('Pessoas com altura abaixo da media');
  for C := 1 to 50 do
    if ( VETALT[C] < MEDIATURMA ) then
      write(VETALT[C],' ');
  writeLn('');
end.
```

```
program l4p19b;
var
  C, CODSEXO, NMULHER: integer;
  ALTURA, MAIOR, MENOR: real;
  VETALT,VETSEX: array [1..50] of real;
  SOMAMULHER, MEDIAMULHER: real;
  SOMATURMA, MEDIATURMA: real;
begin
  for C := 1 to 50 do
  begin
    write('Altura: ');
    readLn(ALTURA);
    write('Sexo (1=M/2=F): ');
    readLn(CODSEXO);
    VETALT[C] := ALTURA;
    VETSEX[C] := CODSEXO;
  end;
  NMULHER    := 0;
  SOMAMULHER := 0;
  SOMATURMA  := 0;
  MAIOR := VETALT[1];
  MENOR := VETALT[1];
  for C := 1 to 50 do
  begin
    if ( VETALT[C] > MAIOR ) then
      MAIOR := VETALT[C]
    else
      if ( VETALT[C] < MENOR ) then
        MENOR := VETALT[C];
    if ( VETSEX[C] = 2 ) then begin
      NMULHER := NMULHER + 1;
      SOMAMULHER := SOMAMULHER + VETALT[C]; end;
    SOMATURMA := SOMATURMA + VETALT[C];
  end;
  MEDIAMULHER := SOMAMULHER / NMULHER;
  MEDIATURMA  := SOMATURMA  / 50;
  writeln('Maior altura da turma: ',MAIOR);
  writeln('Menor altura da turma: ',MENOR);
  writeLn('Mulheres com altura acima da media das mulheres');
  for C := 1 to 50 do
    if ( VETSEX[C] = 2 ) and ( VETALT[C] > MEDIAMULHER ) then
      write(VETALT[C],' ');
  writeLn('');
  writeLn('Pessoas com altura abaixo da media');
  for C := 1 to 50 do
    if ( VETALT[C] < MEDIATURMA ) then
      write(VETALT[C],' ');
  writeLn('');
end.
```

```
for C = 1 : 50
  ALTURA = input('Altura: ');
  CODSEXO = input('Sexo (1=M/2=F): ');
  VETALT(C) = ALTURA;
  VETSEX(C) = CODSEXO;
end
NMULHER    = 0;
SOMAMULHER = 0;
SOMATURMA  = 0;
MAIOR = VETALT(1);
MENOR = VETALT(1);
for C = 1 : 50
  if ( VETALT(C) > MAIOR )
    MAIOR = VETALT(C);
  else
    if ( VETALT(C) < MENOR )
      MENOR = VETALT(C);
    end
  end
  if ( VETSEX(C) == 2 )
    NMULHER = NMULHER + 1;
    SOMAMULHER = SOMAMULHER + VETALT(C);
  end
  SOMATURMA = SOMATURMA + VETALT(C);
end
MEDIAMULHER = SOMAMULHER / NMULHER;
MEDIATURMA  = SOMATURMA  / 50;
fprintf(1,'Maior altura da turma: %.2f\n',MAIOR);
fprintf(1,'Menor altura da turma: %.2f\n',MENOR);
disp('Mulheres com altura acima da media das mulheres');
for C = 1 : 50
  if ( VETSEX(C) == 2 ) & ( VETALT(C) > MEDIAMULHER )
    fprintf(1,'%d ',VETALT(C));
  end
end
fprintf(1,'\n');
disp('Pessoas com altura abaixo da media');
for C = 1 : 50
  if ( VETALT(C) < MEDIATURMA )
   fprintf(1,'%d ',VETALT(C));
  end
end
fprintf(1,'\n');
```

```
for C = 1 : 50
  ALTURA = input('Altura: ');
  CODSEXO = input('Sexo (1=M/2=F): ');
  VETALT(C) = ALTURA;
  VETSEX(C) = CODSEXO;
end
NMULHER    = 0;
SOMAMULHER = 0;
SOMATURMA  = 0;
MAIOR = VETALT(1);
MENOR = VETALT(1);
for C = 1 : 50
  if ( VETALT(C) > MAIOR )
    MAIOR = VETALT(C);
  else
    if ( VETALT(C) < MENOR )
      MENOR = VETALT(C);
    end
  end
  if ( VETSEX(C) == 2 )
    NMULHER = NMULHER + 1;
    SOMAMULHER = SOMAMULHER + VETALT(C);
  end
  SOMATURMA = SOMATURMA + VETALT(C);
end
MEDIAMULHER = SOMAMULHER / NMULHER;
MEDIATURMA  = SOMATURMA  / 50;
fprintf(1,'Maior altura da turma: %.2f\n',MAIOR);
fprintf(1,'Menor altura da turma: %.2f\n',MENOR);
disp('Mulheres com altura acima da media das mulheres');
for C = 1 : 50
  if ( VETSEX(C) == 2 ) & ( VETALT(C) > MEDIAMULHER )
    fprintf(1,'%d ',VETALT(C));
  end
end
fprintf(1,'\n');
disp('Pessoas com altura abaixo da media');
for C = 1 : 50
  if ( VETALT(C) < MEDIATURMA )
   fprintf(1,'%d ',VETALT(C));
  end
end
fprintf(1,'\n');
```

20) Construa um algoritmo em PORTUGOL para calcular a média de valores PARES e ÍMPARES, de 50 números que serão digitados pelo usuário. Ao final o algoritmo deve mostrar estas duas médias. O algoritmo deve mostrar também o maior número PAR digitado e o menor número ÍMPAR digitado. Esses dados devem ser armazenados em um vetor. Além disso, devem ser impressos os valores PARES maiores que a média PAR, bem como os valores ÍMPARES menor que a média ÍMPAR.

```
algoritmo L4P20;
var
  inteiro: VALOR, VETVAL[1..50], SOMAPAR, SOMAIMP, MAIORPAR, MENORIMP, C, CPAR, CIMP;
  real:    MEDIAPAR, MEDIAIMP;
início
  MAIORPAR <- 0;
  MENORIMP <- 0;
  SOMAPAR <- 0;
  SOMAIMP <- 0;
  CPAR <- 0;
  CIMP <- 0;
  para C de 1 até 50 faça
    leia(VALOR);
    VETVAL[C] <- VALOR;
    se ( VETVAL[C] mod 2 = 0 ) { é par } então
      se ( ( VETVAL[C] > MAIORPAR ) ou ( CPAR = 0) ) então
        MAIORPAR <- VETVAL[C];
      fim-se
      SOMAPAR <- SOMAPAR + VETVAL[C];
      CPAR <- CPAR + 1;
    senão
      se ( ( VETVAL[C] < MENORIMP ) ou ( CIMP = 0 ) ) então
        MENORIMP <- VETVAL[C];
      fim-se
      SOMAIMP <- SOMAIMP + VETVAL[C];
      CIMP <- CIMP + 1;
    fim-se
  fim-para
  se ( CPAR <> 0 ) então
    imprima("Maior par: ",MAIORPAR);
    MEDIAPAR <- SOMAPAR / CPAR;
    imprima("A media dos valores pares digitados eh: "  ,MEDIAPAR);
    imprima("Valores PARES maiores que a media PAR");
    para C de 1 até 50 faça
      se ( VETVAL[C] mod 2 = 0 ) e ( VETVAL[C] > MEDIAPAR ) então
        imprima(VETVAL[C]);
      fim-se
    fim-para
  senão
    imprima("Não foi digitado valor par!")
  fim-se
  se ( CIMP <> 0 ) então
    imprima("Menor impar: ",MENORIMP);
    MEDIAIMP <- SOMAIMP / CIMP;
    imprima("A media dos valores impares digitados eh: ",MEDIAIMP);
    imprima("Valores IMPARES menores que a media IMPAR");
    para C de 1 até 50 faça
      se ( VETVAL[C] mod 2 = 1 ) e ( VETVAL[C] < MEDIAIMP ) então
        imprima(VETVAL[C]);
      fim-se
    fim-para
  senão
    imprima("Não foi digitado valor impar!")
  fim-se
fim
```

```
program l4p20b;
var
  VALOR, SOMAPAR, SOMAIMP, MAIORPAR, MENORIMP, C, CPAR, CIMP: integer;
  VETVAL: array [1..50] of integer;
  MEDIAPAR, MEDIAIMP: real;
begin
  MAIORPAR := 0;
  MENORIMP := 0;
  SOMAPAR := 0;
  SOMAIMP := 0;
  CPAR := 0;
  CIMP := 0;
  for C := 1 to 50 do
  begin
    write('Digite um valor: ');
    readLn(VALOR);
    VETVAL[C] := VALOR;
    if ( VETVAL[C] mod 2 = 0 ) { é par } then begin
      if ( ( VETVAL[C] < MAIORPAR ) or ( CPAR = 0) ) then
        MAIORPAR := VETVAL[C];
      SOMAPAR := SOMAPAR + VETVAL[C];
      CPAR := CPAR + 1; end
    else begin
      if ( ( VETVAL[C] > MENORIMP ) or ( CIMP = 0 ) ) then
        MENORIMP := VETVAL[C];
      SOMAIMP := SOMAIMP + VETVAL[C];
      CIMP := CIMP + 1; end;
  end;
  if ( CPAR <> 0 ) then begin
    writeLn('Maior par: ',MAIORPAR);
    MEDIAPAR := SOMAPAR / CPAR;
    writeln('A media dos valores pares digitados eh: '  ,MEDIAPAR);
    writeLn('Valores PARES maiores que a media PAR');
    for C := 1 to 50 do
      if ( VETVAL[C] mod 2 = 0 ) and ( VETVAL[C] > MEDIAPAR ) then
        write(VETVAL[C],' ');
    writeLn(''); end
  else
    writeLn('Não foi digitado valor impar!');
  if ( CIMP <> 0 ) then begin
    writeLn('Menor impar: ',MENORIMP);
    MEDIAIMP := SOMAIMP / CIMP;
    writeln('A media dos valores impares digitados eh: ',MEDIAIMP);
    writeLn('Valores IMPARES menores que a media IMPAR');
    for C := 1 to 50 do
      if ( VETVAL[C] mod 2 = 1 ) and ( VETVAL[C] < MEDIAIMP ) then
        write(VETVAL[C],' ');
    writeLn(''); end
  else
    writeLn('Não foi digitado valor impar!');
end.
```

```
SOMAPAR = 0;
SOMAIMP = 0;
CPAR = 0;
CIMP = 0;
for C = 1 : 50
  VALOR = input('Digite um valor: ');
  VETVAL(C) = VALOR;
  if ( mod(VETVAL(C),2) == 0 ) % é par
    if ( ( VETVAL(C) < MAIORPAR ) | ( CPAR == 0) )
      MAIORPAR = VETVAL(C);
    end
    SOMAPAR = SOMAPAR + VETVAL(C);
    CPAR = CPAR + 1;
  else
    if ( ( VETVAL(C) > MENORIMP ) | ( CIMP == 0 ) )
      MENORIMP = VETVAL(C);
    end
    SOMAIMP = SOMAIMP + VETVAL(C);
    CIMP = CIMP + 1;
  end
end
if ( CPAR ~= 0 )
  fprintf(1,'Maior par: %d\n',MAIORPAR);
  MEDIAIMP = SOMAIMP / CIMP;
  fprintf(1,'A media dos valores pares digitados eh: %f\n',MEDIAPAR);
  disp('Valores PARES maiores que a media PAR');
  for C = 1 : 50
    if ( mod(VETVAL(C),2) == 0 ) & ( VETVAL(C) > MEDIAPAR )
      fprintf(1,'%d ',VETVAL(C));
    end
  end
  fprintf(1,'\n');
else
  disp('Não foi digitado valor par!');
end
if ( CIMP ~= 0 )
  fprintf(1,'Menor impar: %d\n',MENORIMP);
  MEDIAPAR = SOMAPAR / CPAR;
  fprintf(1,'A media dos valores impares digitados eh: %f\n',MEDIAIMP);
  disp('Valores IMPARES menores que a media IMPAR');
  for C = 1 : 50
    if ( mod(VETVAL(C),2) == 1 ) & ( VETVAL(C) < MEDIAIMP )
      fprintf(1,'%d ',VETVAL(C));
    end
  end
  fprintf(1,'\n');
else
  disp('Não foi digitado valor impar!');
end
```

21) Em uma cidade do interior, sabe-se que, de janeiro a abril de 1976 (121 dias), não ocorreu temperatura inferior a 15ºC nem superior a 40ºC. As temperaturas verificadas em cada dia estão disponíveis em uma unidade de entrada de dados.
Fazer um algoritmo em PORTUGOL que calcule e imprima:

- A menor temperatura ocorrida;
- A maior temperatura ocorrida;
- A temperatura média;
- O número de dias nos quais a temperatura foi inferior à temperatura média.

```
algoritmo L4P21;
var
  inteiro: NDIAS, C;
  real:    MENORTEMP, MAIORTEMP, MEDIA, SOMA, VETEMP[1..121];
início
  MENORTEMP <- 0;
  MAIORTEMP <- 0;
  MEDIA <- 0;
  NDIAS <- 0;
  SOMA <- 0;
  para C de 1 até 121 faça
    imprima("Digite a temperatura: ");
    leia(VETEMP[C]);
    se ((VETEMP[C] < MENORTEMP) ou (C = 1)) então
      MENORTEMP <- VETEMP[C];
    fim-se
    se ((VETEMP[C] > MAIORTEMP) ou (C = 1)) então
      MAIORTEMP <- VETEMP[C];
    fim-se
    SOMA <- SOMA + VETEMP[C];
  fim-para
  MEDIA <- SOMA / 121;
  para C de 1 até 121 faça
    se  (VETEMP[C] < MEDIA) então
      NDIAS <- NDIAS + 1;
    fim-se
  fim-para
  imprima("A menor temperatura ocorrida eh: ",MENORTEMP);
  imprima("A maior temperatura ocorrida eh: ",MAIORTEMP);
  imprima("A temperatura media eh: ", MEDIA);
  imprima("Numero de dias nos quais a temperatura foi inferior a media: ",NDIAS);
fim
```

```
program l4p21;
var
  NDIAS, C : integer;
  MENORTEMP, MAIORTEMP, MEDIA, SOMA: real;
  VETEMP: array [1..121] of real;
begin
  MENORTEMP := 0;
  MAIORTEMP := 0;
  MEDIA := 0;
  NDIAS := 0;
  SOMA := 0;
  for C := 1 to 121 do
  begin
    write('Digite a temperatura: ');
    readln(VETEMP[C]);
    if ((VETEMP[C] < MENORTEMP) or (C = 1)) then
      MENORTEMP := VETEMP[C];
    if ((VETEMP[C] > MAIORTEMP) or (C = 1)) then
      MAIORTEMP := VETEMP[C];
    SOMA := SOMA + VETEMP[C];
  end;
  MEDIA := SOMA / 121;
  for C := 1 to 121 do
    if  (VETEMP[C] < MEDIA) then
      NDIAS := NDIAS + 1;
  writeln('A menor temperatura ocorrida eh: ',MENORTEMP:2:2);
  writeln('A maior temperatura ocorrida eh: ',MAIORTEMP:2:2);
  writeln('A temperatura media eh: ', MEDIA:2:2);
  writeln('Numero de dias nos quais a temperatura foi inferior a media: ',NDIAS);
end.
```

```
MENORTEMP = 0;
MAIORTEMP = 0;
MEDIA = 0;
NDIAS = 0;
SOMA = 0;
for C = 1 : 121
  VETEMP(C) = input('Digite a temperatura: ');
  if ((VETEMP(C) < MENORTEMP) | (C == 1))
    MENORTEMP = VETEMP(C);
  end
  if ((VETEMP(C) > MAIORTEMP) | (C == 1))
    MAIORTEMP = VETEMP(C);
  end
  SOMA = SOMA + VETEMP(C);
end
MEDIA = SOMA / 121;
for C = 1 : 121
  if  (VETEMP(C) < MEDIA)
    NDIAS = NDIAS + 1;
  end
end
fprintf(1,'A menor temperatura ocorrida eh: %f\n',MENORTEMP);
fprintf(1,'A maior temperatura ocorrida eh: %f\n',MAIORTEMP);
fprintf(1,'A temperatura media eh: %f\n', MEDIA);
fprintf(1,'Numero de dias nos quais a temperatura foi inferior a media: %d',NDIAS);
```

22) Faça um algoritmo em PORTUGOL que:
a) Leia uma frase de 80 caracteres, incluindo brancos;
b) Conte quantos brancos existem na frase;
c) Conte quantas vezes a letra A aparece;
d) Imprima o que foi calculado nos itens *b* e *c*.

```
algoritmo L4P22;
var
  literal: ENTRADA;
  inteiro: NUMBRANCO, NUMA, C;
início
  NUMA <- 0;
  NUMBRANCO <- 0;
  imprima("Digite uma frase contendo 80 caracteres: ");
  leia(ENTRADA);
  para  C de 1 até 80 faça
    se (ENTRADA[C] = " ") então
      NUMBRANCO <- NUMBRANCO + 1;
    fim-se
    se ((ENTRADA[C] = "A") ou (ENTRADA[C] = "a")) então
      NUMA <- NUMA + 1;
    fim-se
  fim-para
  imprima("Existem ",NUMBRANCO," espacos em branco na frase.");
  imprima("Existem ",NUMA," letras A na frase.");
fim
```

```
program l4p22;
var
  entrada: String[80];
  numBranco, numA, C : integer;
begin
  numA := 0;
  numBranco := 0;
  writeln('Digite uma frase contendo 80 caracteres: ');
  readln(entrada);
  for  C := 1 to 80 do
  begin
    if (entrada[C] = ' ') then
      numBranco := numBranco + 1;
    if ((entrada[C] = 'A') or (entrada[C] = 'a')) then
      numA := numA + 1;
  end;
  writeln('Existem ',numBranco,' espacos em branco na frase.');
  writeln('Existem ',numA,' letras A na frase.');
end.
```

```
numA = 0;
numBranco = 0;
entrada = input('Digite uma frase contendo 80 caracteres: ');
for  C = 1 : 80
  if (entrada(C) == ' ')
    numBranco = numBranco + 1;
  end
  if ((entrada(C) == 'A') | (entrada(C) == 'a'))
    numA = numA + 1;
  end
end
fprintf(1,'Existem %d espacos em branco na frase.\n',numBranco);
fprintf(1,'Existem %d letras A na frase.\n',numA);
```

23) Fazer um algoritmo em PORTUGOL que:

a) Leia o valor inteiro de n (n ≤ 1000) e os n valores de uma variável composta A de valores numéricos, ordenados de forma crescente;
b) Determine e imprima, para cada número que se repete no conjunto, a quantidade de vezes em que ele aparece repetido;
c) Elimine os elementos repetidos, formando um novo conjunto;
d) Imprima o conjunto obtido no item *c*.

```
algoritmo L4P23;
var
  inteiro:  C, J, N, NVEZES;
  real:     ANTERIOR, VETORIG[1..1000], VETSRP[1..1000];
início
  imprima("Digite o valor de n: ");
  leia(N);
  imprima("Digite os numeros em ordem crescente: ");
  para C de 1 até N faça
    leia(VETORIG[C]);
  fim-para
  ANTERIOR <- VETORIG[1];
  NVEZES <- 1;
  VETSRP[1] <- VETORIG[1];
  J <- 1;
  para C de 2 até N faça
    se (VETORIG[C] = ANTERIOR) então
      NVEZES <- NVEZES + 1
    senão
      imprima("O numero ",ANTERIOR," se repete ",NVEZES," vezes");
      J <- J + 1;
      VETSRP[J] <- VETORIG[C];
      ANTERIOR <- VETORIG[C];
      NVEZES <- 1;
    fim-se
 fim-para
 imprima("O numero ",ANTERIOR," se repete ",NVEZES," vezes");
 imprima("O vetor sem numeros repetido eh: ");
 para C de 1 até J faça
   imprima(VETSRP[C], " ");
 fim-para
fim
```

```
program l4p23;
var
  C, J, n, nvezes  : integer;
  anterior : real;
  vetORIG, vetSRP : array [1..1000] of real;
begin
  write('Digite o valor de n: ');
  readln(n);
  writeln('Digite os numeros em ordem crescente: ');
  for C := 1 to n do
    readln(vetORIG[C]);
  anterior := vetORIG[1];
  nvezes := 1;
  vetSRP[1] := vetORIG[1];
  J := 1;
  for C := 2 to n do
  begin
    if (vetORIG[C] = anterior) then
      nvezes := nvezes + 1
    else
    begin
      writeln('O numero ',anterior:6:4,' se repete ',nvezes,' vezes');
      J := J + 1;
      vetSRP[J] := vetORIG[C];
      anterior := vetORIG[C];
      nvezes := 1;
    end;
 end;
 writeln('O numero ',anterior:6:4,' se repete ',nvezes,' vezes');
 writeln('O vetor sem numeros repetido eh: ');
 for C :=1 to J do
   write(vetSRP[C]:6:4, ' ');
end.
```

```
n = input('Digite o valor de n: ');
fprintf(1,'Digite os numeros em ordem crescente: \n');
for C = 1 : n
  vetORIG(C) = input('Digite: ');
end
anterior = vetORIG(1);
nvezes = 1;
vetSRP(1) = vetORIG(1);
J = 1;
for C = 2 : n
  if (vetORIG(C) == anterior)
    nvezes = nvezes + 1;
  else
    fprintf(1,'O numero %f se repete %d vezes \n',anterior,nvezes);
    J = J + 1;
    vetSRP(J) = vetORIG(C);
    anterior = vetORIG(C);
    nvezes = 1;
  end
end
fprintf(1,'O numero %f se repete %d vezes \n',anterior,nvezes);
fprintf(1,'O vetor sem numeros repetido eh: ');
for C = 1 : J
  fprintf(1,'%f ',vetSRP(C));
end
```

24) Dado um conjunto de 100 valores numéricos disponíveis num meio de entrada qualquer, fazer um algoritmo em PORTUGOL para armazená-los numa variável composta B, e calcular e imprimir o valor do somatório dado a seguir:

$$S = (b_1 - b_{100})^3 + (b_2 - b_{99})^3 + (b_3 - b_{98})^3 + ... + (b_{50} - b_{51})^3$$

```
algoritmo L4P24;
var
  inteiro: C;
  real:    SOMA, VARB[1..100];
início
  SOMA <- 0;
  para C <- 1 até 100 faça
    imprima("Digite o ",C,"o valor numerico: ");
    leia(VARB[C]);
  fim-para
  para C de 1 até 50 faça
    SOMA <- SOMA + (VARB[C]-VARB[101-C])*(VARB[C]-VARB[101-C])*(VARB[C]-VARB[101-C]);
  fim-para
  imprima("O valor do somatorio eh: ", SOMA);
fim
```

```
program l4p24;
var
  C : integer;
  SOMA: real;
  varB : array [1..100] of real;
begin
  SOMA := 0;
  for C := 1 to 100 do
  begin
    write('Digite o ',C,'o valor numerico: ');
    readln(varB[C]);
  end;
  for C := 1 to 50 do
    SOMA := SOMA + (varB[C]-varB[101-C])*(varB[C]-varB[101-C])*(varB[C]-varB[101-C]);
  write('O valor do somatorio eh: ', SOMA);
end.
```

```
SOMA = 0;
for C = 1 : 100
  fprintf(1,'Digite o %d',C);
  varB(C) = input('o valor numerico: \n');
end
for C = 1 : 50
  SOMA = SOMA + (varB(C)-varB(101-C))*(varB(C)-varB(101-C))*(varB(C)-varB(101-C));
end
fprintf(1,'O valor do somatorio eh: %f\n',SOMA);
```

25) Fazer um algoritmo em PORTUGOL que:

a) Leia um conjunto de valores inteiros correspondentes a 80 notas dos alunos de uma turma, notas estas que variam de 0 a 10;
b) Calcule a freqüência absoluta e a freqüência relativa de cada nota;
c) Imprima uma tabela contendo os valores das notas (de 0 a 10) e suas respectivas freqüências absoluta e relativa.

Observações:

1. Freqüência absoluta de uma nota é o número de vezes em que aparece no conjunto de dados;
2. Freqüência relativa é a freqüência absoluta divida pelo número total de dados;
3. Utilizar como variável composta somente aquelas que forem necessárias.

```
algoritmo L4P25;
var
  inteiro: C, NOTA, FREQAB[0..10];
início
  para C de 0 até 10 faça
    FREQAB[C] <- 0;
  fim-para
  para C de 1 até 80 faça
    imprima("Digite a nota: ");
    leia(NOTA);
    se (NOTA = 0) então
      FREQAB[0] <- FREQAB[0] + 1;
    fim-se
    se (NOTA = 1) então
      FREQAB[1] <- FREQAB[1] + 1;
    fim-se
    se (NOTA = 2) então
      FREQAB[2] <- FREQAB[2] + 1;
    fim-se
    se (NOTA = 3) então
      FREQAB[3] <- FREQAB[3] + 1;
    fim-se
    se (NOTA = 4) então
      FREQAB[4] <- FREQAB[4] + 1;
    fim-se
    se (NOTA = 5) então
      FREQAB[5] <- FREQAB[5] + 1;
    fim-se
    se (NOTA = 6) então
      FREQAB[6] <- FREQAB[6] + 1;
    fim-se
    se (NOTA = 7) então
      FREQAB[7] <- FREQAB[7] + 1;
    fim-se
    se (NOTA = 8) então
      FREQAB[8] <- FREQAB[8] + 1;
    fim-se
    se (NOTA = 9) então
      FREQAB[9] <- FREQAB[9] + 1;
    fim-se
    se (NOTA = 10) então
      FREQAB[10] <- FREQAB[10] + 1;
    fim-se
  fim-para
  para C de 0 até 10 faça
    imprima("A frequencia absoluta da nota ",C," eh ",FREQAB[C]);
    imprima("A frequencia relativa da nota ",C," eh ",FREQAB[C] / 80);
  fim-para
fim
```

```
algoritmo L4P25B;
var
  inteiro: C, NOTA, FREQAB[0..10];
início
  para C de 0 até 10 faça
    FREQAB[C] <- 0;
  fim-para
  para C de 1 até 80 faça
    imprima("Digite a nota: ");
    leia(NOTA);
    se (NOTA >= 0) e (NOTA <= 10) então
      FREQAB[NOTA] <- FREQAB[NOTA] + 1;
    senão
      imprima("Nota inválida!");
    fim-se
  fim-para
  para C de 0 até 10 faça
    imprima("A frequencia absoluta da nota ",C," eh ",FREQAB[C]);
    imprima("A frequencia relativa da nota ",C," eh ",100 * FREQAB[C] / 80);
  fim-para
fim
```

```
program l4p25;
var
  C , NOTA: integer;
  FREQAB : array [0..10] of integer;
begin
  for C := 0 to 10 do    {inicializa o vetor com zeros}
    FREQAB[C] := 0;
  for C := 1 to 80 do
    begin
      write('Digite a nota: ');
      readln(NOTA);
      if (NOTA = 0) then
        FREQAB[0] := FREQAB[0] + 1;
      if (NOTA = 1) then
        FREQAB[1] := FREQAB[1] + 1;
      if (NOTA = 2) then
        freqab[2] := FREQAB[2] + 1;
      if (NOTA = 3) then
        FREQAB[3] := FREQAB[3] + 1;
      if (NOTA = 4) then
        FREQAB[4] := FREQAB[4] + 1;
      if (NOTA = 5) then
        FREQAB[5] := FREQAB[5] + 1;
      if (NOTA = 6) then
        FREQAB[6] := FREQAB[6] + 1;
      if (NOTA = 7) then
        FREQAB[7] := FREQAB[7] + 1;
      if (NOTA = 8) then
        FREQAB[8] := FREQAB[8] + 1;
      if (NOTA = 9) then
        FREQAB[9] := FREQAB[9] + 1;
      if (NOTA = 10) then
        FREQAB[10] := FREQAB[10] + 1;
    end;
  for C := 0 to 10 do
    begin
    writeln('A frequencia absoluta da nota ',C,' eh ',FREQAB[C]);
    writeln('A frequencia relativa da nota ',C,' eh ',FREQAB[C] / 80);
    end;
end.
```

```
program l4p25b;
var
  C, NOTA: integer;
  FREQAB: array [0..10] of integer;
begin
  for C := 0 to 10 do
    FREQAB[C] := 0;
  for C := 1 to 80 do
  begin
    write('Digite a nota: ');
    readLn(NOTA);
    if (NOTA >= 0) and (NOTA <= 10) then
      FREQAB[NOTA] := FREQAB[NOTA] + 1
    else
      writeLn('Nota inválida!');
  end;
  for C := 0 to 10 do
  begin
    writeLn('A frequencia absoluta da nota ',C,' eh ',FREQAB[C]);
    writeLn('A frequencia relativa da nota ',C,' eh ',100 * FREQAB[C] / 80 : 6:2);
  end;
end.
```

```
for C = 1 : 11     %Inicializa o vetor com zeros
  freqab(C) = 0;
end
for C = 1 : 80
  nota = input('Digite a nota: ');
  if (nota == 0)
    freqab(1) = freqab(1) + 1;
  elseif (nota == 1)
    freqab(2) = freqab(2) + 1;
  elseif (nota == 2)
    freqab(3) = freqab(3) + 1;
  elseif (nota == 3)
    freqab(4) = freqab(4) + 1;
  elseif (nota == 4)
    freqab(5) = freqab(5) + 1;
  elseif (nota == 5)
    freqab(6) = freqab(6) + 1;
  elseif (nota == 6)
    freqab(7) = freqab(7) + 1;
  elseif (nota == 7)
    freqab(8) = freqab(8) + 1;
  elseif (nota == 8)
    freqab(9) = freqab(9) + 1;
  elseif (nota == 9)
    freqab(10) = freqab(10) + 1;
  elseif (nota == 10)
    freqab(11) = freqab(11) + 1;
  end
end
for C = 1 : 11
  fprintf(1,'A frequencia absoluta da nota %d eh %d\n',C-1,freqab(C));
  fprintf(1,'A frequencia relativa da nota %d eh %6.2f\n',C-1,100 * freqab(C) / 80);
end
```

```
for C = 1 : 11
  FREQAB[C] = 0;
end
for C = 1 : 80
  NOTA = input('write('Digite a nota: ');
  if ( (NOTA >= 0) & (NOTA <= 10) )
    FREQAB[NOTA+1] := FREQAB[NOTA+1] + 1;
  else
    disp('Nota inválida!');
  end
end
for C := 0 to 10 do
  fprintf(1,'A frequencia absoluta da nota %d eh %d',C,FREQAB[C+1]);
  fprintf(1,'A frequencia relativa da nota %d eh %6.2f',C,100 * FREQAB[C+1] / 80);
end
```

26) Um armazém trabalha com 100 mercadorias diferentes identificadas pelos números inteiros de 1 a 100. O dono do armazém anota a quantidade de cada mercadoria vendida durante o mês. Ele tem uma tabela que indica, para cada mercadoria, o preço de venda. Escreva um algoritmo em PORTUGOL para calcular o faturamento mensal do armazém. A tabela de preços é fornecida seguida pelos números das mercadorias e as quantidades vendidas. Quando uma mercadoria não tiver nenhuma venda, é informado o valor zero no lugar da quantidade.

```
algoritmo L4P26;
var
  inteiro: QUANT, C;
  real:    FATURAMENTO, PRECO[1..100];
início
  FATURAMENTO <- 0;
  para C de 1 até 100 faça
    imprima("Digite o preco de venda da mercadoria n.",C," :");
    leia(PRECO[C]);
  fim-para
  para C de 1 até 100 faça
    imprima("Digite a quantidade vendida da mercadoria",C," :");
    leia(QUANT);
    FATURAMENTO <- FATURAMENTO + QUANT * PRECO[C];
  fim-para
  imprima("O faturamento eh: ",FATURAMENTO);
fim
```

```
program l4p26;
var
  QUANT, C : integer;
  FATURAMENTO : real;
  PRECO: array [1..100] of real;
begin
  FATURAMENTO := 0;
  for C := 1 to 100 do
  begin
    write('Digite o preco de venda da mercadoria n.',C,' :');
    readln(PRECO[C]);
  end;
  for C := 1 to 100 do
  begin
    write('Digite a quantidade vendida da mercadoria',C,' :');
    readln(QUANT);
    FATURAMENTO := FATURAMENTO + QUANT * PRECO[C];
  end;
  writeln('O faturamento eh: ',FATURAMENTO:10:2);
end.
```

```
FATURAMENTO = 0;
for C = 1 : 100
  fprintf(1,'Digite o preco de venda da mercadoria n. %d',C);
  PRECO(C) = input(' :');
end
for C = 1 : 100
  fprintf(1,'Digite a quantidade vendida da mercadoria n.%d',C)
  QUANT = input(' :');
  FATURAMENTO = FATURAMENTO + QUANT * PRECO(C);
end
fprintf(1,'O faturamento eh: %9.2f',FATURAMENTO);
```

27) Uma grande firma deseja saber quais os três empregados mais recentes. Fazer um algoritmo em PORTUGOL para ler um número indeterminado de informações (máximo de 300) contendo o número do empregado e o número de meses de trabalho deste empregado e imprimir os três mais recentes.
Observações: A última informação contém os dois números iguais a zero. Não existem dois empregados admitidos no mesmo mês.

```
algoritmo L4P27;
var
  inteiro: RECENTE1, RECENTE2, RECENTE3, C, NEMP[1..300], NMESES[1..300];
início
  imprima("Digite o numero do empregado: ");
  leia(NEMP[1]);
  imprima("Digite o numero de meses de trabalho: ");
  leia(NMESES[1]);
  RECENTE1 <- 1;
  RECENTE2 <- 2;
  RECENTE3 <- 3;
  C <- 1;
  enquanto ((NEMP[C] <> 0) ou (NMESES[C] <> 0)) faça
    se (NMESES[C] < NMESES[RECENTE1]) então
      RECENTE3 := RECENTE2;
      RECENTE2 := RECENTE1;
      RECENTE1 := C;
    senão
      se (NMESES[C] < NMESES[RECENTE2]) então
        RECENTE3 := RECENTE2;
        RECENTE2 := C;
      senão
        se (NMESES[C] < NMESES[RECENTE3]) então
          RECENTE3 := C;
        fim-se
      fim-se
    fim-se
    C := C + 1;
    imprima("Digite o numero do empregado: ");
    leia(NEMP[C]);
    imprima("Digite o numero de meses de trabalho: ");
    leia(NMESES[C]);
  fim-enquanto
  se (C = 1) então {O usuario digitou zero na primeira vez}
    imprima("Nao foi digitado nenhum empregado!")
  senão
    imprima("O 1o empregado mais recente eh o de numero: ",NEMP[RECENTE1]);
    imprima("O 2o empregado mais recente eh o de numero: ",NEMP[RECENTE2]);
    imprima("O 3o empregado mais recente eh o de numero: ",NEMP[RECENTE3]);
  fim-se
fim
```

```
program  l4p27;
var
  RECENTE1, RECENTE2, RECENTE3, C : integer;
  NEMP, NMESES: array [1..300] of integer;
begin
  for C := 2 to 3 do      {Inicializa o 2. e 3. valores}
    NMESES[C] := 0;       {do vetor para a comparacao no if}
  write('Digite o numero do empregado: ');
  readln(NEMP[1]);
  write('Digite o numero de meses de trabalho: ');
  readln(NMESES[1]);
  RECENTE1 := 1;
  RECENTE2 := 2;
  RECENTE3 := 3;
  C := 1;
  while ((NEMP[C] <> 0) or (NMESES[C] <> 0)) do
  begin
    if (NMESES[C] < NMESES[RECENTE1]) then
    begin
      RECENTE3 := RECENTE2;
      RECENTE2 := RECENTE1;
      RECENTE1 := C;
    end
    else if (NMESES[C] < NMESES[RECENTE2]) then
    begin
      RECENTE3 := RECENTE2;
      RECENTE2 := C;
    end
    else if (NMESES[C] < NMESES[RECENTE3]) then
      RECENTE3 := C;
    C := C + 1;
    write('Digite o numero do empregado: ');
    readln(NEMP[C]);
    write('Digite o numero de meses de trabalho: ');
    readln(NMESES[C]);
  end;
  if (C = 1) then {O usuario digitou zero na primeira vez}
    writeln('Nao foi digitado nenhum empregado!')
  else
  begin
    writeln('O 1o empregado mais recente eh o de numero: ',NEMP[RECENTE1]);
    writeln('O 2o empregado mais recente eh o de numero: ',NEMP[RECENTE2]);
    writeln('O 3o empregado mais recente eh o de numero: ',NEMP[RECENTE3]);
  end;
end.
```

```
for C = 2 : 3       %Inicializa o 2. e 3. valores para
  NMESES(C) = 0;    %a operacao logica no if
end;
NEMP(1) = input('Digite o numero do empregado: ');
NMESES(1) = input('Digite o numero de meses de trabalho: ');
RECENTE1 = 1;
RECENTE2 = 2;
RECENTE3 = 3;
C = 1;
while ((NEMP(C) ~= 0) | (NMESES(C) ~= 0))
  if (NMESES(C) < NMESES(RECENTE1))
    RECENTE3 = RECENTE2;
    RECENTE2 = RECENTE1;
    RECENTE1 = C;
  elseif (NMESES(C) < NMESES(RECENTE2))
    RECENTE3 = RECENTE2;
    RECENTE2 = C;
  elseif (NMESES(C) < NMESES(RECENTE3))
    RECENTE3 = C;
  end
  C = C + 1;
  NEMP(C) = input('Digite o numero do empregado: ');
  NMESES(C) = input('Digite o numero de meses de trabalho: ');
end
if (C == 1)     {O usuario digitou zero na primeira vez}
  fprintf(1,'Nao foi digitado nenhum empregado!');
else
  fprintf(1,'O 1o empregado mais recente eh o de numero: %f\n',NEMP(RECENTE1));
  fprintf(1,'O 2o empregado mais recente eh o de numero: %f\n',NEMP(RECENTE2));
  fprintf(1,'O 3o empregado mais recente eh o de numero: %f\n',NEMP(RECENTE3));
end
```

28) Fazer um algoritmo em PORTUGOL que:

a) Leia uma variável composta A com 30 valores numéricos distintos;
b) Leia outra variável composta B com 30 valores numéricos;
c) Leia o valor de uma variável X;
d) Verifique qual o elemento de A que é igual a X;
e) Imprima o elemento de B de posição correspondente à do elemento de A igual a X..

```
algoritmo L4P28;
var
  inteiro: C;
  real:    X, VARA[1..30], VARB[1..30];
início
  imprima("Digite os valores da variavel composta A: ");
  para C de 1 até 30 faça
    leia(VARA[C]);
  fim-para
  imprima("Digite os valores da variavel composta B: ");
  para C de 1 até 30 faça
    leia(VARB[C]);
  fim-para
  imprima("Digite o valor de X: ");
  leia(X);
  para C de 1 até 30 faça
    se (VARA[C] = X) então
      imprima("O elemento de B eh: ",VARB[C]);
      C <- 30;    {finaliza o loop}
    senão
      se (C = 30) então
        imprima("Nao ha elemento de A igual a X.");
      fim-se
    fim-se
  fim-para
fim
```

```
algoritmo L4P28B;
var
  inteiro: C;
  real:    X, VARA[1..30], VARB[1..30];
início
  imprima("Digite os valores da variavel composta A: ");
  para C de 1 até 30 faça
    leia(VARA[C]);
  fim-para
  imprima("Digite os valores da variavel composta B: ");
  para C de 1 até 30 faça
    leia(VARB[C]);
  fim-para
  imprima("Digite o valor de X: ");
  leia(X);
  C <- 0;
  repita
    C <- C + 1;
    se (VARA[C] = X) então
      imprima("O elemento de B eh: ",VARB[C]);
    senão
      se (C = 30) então
        imprima("Nao ha elemento de A igual a X.");
      fim-se
    fim-se
  até ((VARA[C] = X) ou (C > 30))
fim
```

```
program  l4p28;
var
  C : integer;
  X: real;
  varA, varB: array [1..30] of real;
begin
  write('Digite os valores da variavel composta A: ');
  for C := 1 to 30 do
    readln(varA[C]);
  write('Digite os valores da variavel composta B: ');
  for C := 1 to 30 do
    readln(varB[C]);
  write('Digite o valor de X: ');
  readln(X);
  for C := 1 to 30 do
  begin
    if (varA[C] = X) then
    begin
      writeln('O elemento de B eh: ',varB[C]:6:3);
      C := 30;    {finaliza o loop}
    end
    else if (C = 30) then
      writeln('Nao ha elemento de A igual a X.');
  end;
end.
```

```
program  l4p28b;
var C : integer;
    X: real;
    varA, varB: array [1..30] of real;
begin
  write('Digite os valores da variavel composta A: ');
  for C := 1 to 30 do
    readln(varA[C]);
  write('Digite os valores da variavel composta B: ');
  for C := 1 to 30 do
    readln(varB[C]);
  write('Digite o valor de X: ');
  readln(X);
  C := 0;
  repeat
  begin
    C := C + 1;
    if (varA[C] = X) then
      writeln('O elemento de B eh: ',varB[C]:6:3)
    else if (C = 30) then
      writeln('Nao ha elemento de A igual a X.');
  end;
  until ((varA[C] = X) or (C > 30));
end.
```

```
fprintf(1,'Digite os valores da variavel composta A: \n');
for C = 1 : 30
  varA(C) = input('');
end
fprintf(1,'Digite os valores da variavel composta B: \n');
for C = 1 : 30
  varB(C) = input('');
end
X = input('Digite o valor de X: ');
sinal = 0;
C = 1;
while (sinal == 0 & C <= 30)
  if (varA(C) == X)
    fprintf(1,'O elemento de B eh: %f\n',varB(C));
    sinal = 1;    %finaliza o loop
  elseif (C == 30)
    fprintf(1,'Nao ha elemento de A igual a X.');
  end
  C = C + 1;
end
```

29) Fazer um algoritmo em PORTUGOL que:

a) Leia o valor inteiro de M (M ≤ 30) e os M valores de uma variável composta A;
b) Leia o valor inteiro de N (N ≤ 20) e os N valores de um variável composta B;
c) Determine o conjunto C = A ∪ B (união de A com B), onde C não deverá conter elementos repetidos (A e B não contêm elementos repetidos);
d) Imprima os elementos contidos em A, B e C.

```
algoritmo L4P29B;
var
  inteiro: M, N, C, I, OFFSET;
  real:  VARA[1..30], VARB[1..20], VARC[1..50];
início
  imprima("Digite o valor de M: ");
  leia(M);
  para C de 1 até M faça
    imprima("Digite o ",C,". valor da variavel A: ");
    leia(VARA[C]);
    VARC[C] <- VARA[C];
  fim-para
  OFFSET <- M;
  imprima("Digite o valor de N: ");
  leia(N);
  para C de 1 até N faça
    imprima("Digite o ",C,". valor da variavel B: ");
    leia(VARB[C]);
    para I de 1 até M faça
      se (VARB[C] = VARA[I]) então
        I <- M + 1 ;  {para o loop}
      fim-se
    fim-para
    se (I = M) então   {O loop anterior terminou sem encontrar B = A}
      OFFSET <- OFFSET + 1;
      VARC[OFFSET] <- VARB[C];
    fim-se
  fim-para
  imprima("Os valores da variavel composta A sao:");
  para C de 1 até M faça
    imprima(VARA[C]," ");
  fim-para
  imprima("Os valores da variavel composta B sao:");
  para C de 1 até N faça
    imprima(VARB[C]," ");
  fim-para
  imprima("Os valores de A U B sem elementos repetidos sao:");
  para C de 1 até OFFSET faça
    imprima(VARC[C]," ");
  fim-para
fim
```

```
program  l4p29;
var
  M, N, C, I, offset : integer;
  varA: array [1..30] of real;
  varB: array [1..20] of real;
  varC: array [1..50] of real;
begin
  write('Digite o valor de M: ');
  readln(M);
  for C := 1 to M do
  begin
    write('Digite o ',C,'. valor da variavel A: ');
    readln(varA[C]);
    varC[C] := varA[C];
  end;
  offset := M;
  write('Digite o valor de N: ');
  readln(N);
  for C := 1 to N do
  begin
    write('Digite o ',C,'. valor da variavel B: ');
    readln(varB[C]);
    for I := 1 to M do
      if (varB[C] = varA[I]) then
        I := M + 1 ;  {para o loop}
    if (I = M) then   {O loop anterior terminou sem encontrar B = A}
    begin
      offset := offset + 1;
      varC[offset] := varB[C];
    end;
  end;
  writeln('Os valores da variavel composta A sao:');
  for C := 1 to M do
    write(varA[C]:6:3,' ');
  writeln('');
  writeln('Os valores da variavel composta B sao:');
  for C := 1 to N do
    write(varB[C]:6:3,' ');
  writeln('');
  writeln('Os valores de A U B sem elementos repetidos sao:');
  for C := 1 to offset do
    write(varC[C]:6:3,' ');
end.
```

```
M =input('Digite o valor de M: ');
for C = 1 : M
  fprintf(1,'Digite o %d',C);
  varA(C) = input('. valor da variavel A: ');
  varC(C) = varA(C);
end
offset = M;
N = input('Digite o valor de N: ');
for C = 1 : N
  fprintf(1,'Digite o %d',C);
  varB(C) = input('. valor da variavel B: ');
  sinal = 0;
  for I = 1 : M
    if (varB(C) == varA(I))
      sinal = 1;    %o loop encontrou A = B
    end
  end
  if (sinal == 0)  %O loop anterior terminou sem encontrar B = A
    offset = offset + 1;
    varC(offset) = varB(C);
  end
end
fprintf(1,'Os valores da variavel composta A sao:\n');
for C = 1 : M
  fprintf(1,'%f ',varA(C));
end
fprintf(1,'\n');
fprintf(1,'Os valores da variavel composta B sao:\n');
for C = 1 : N
  fprintf(1,'%f ',varB(C));
end
fprintf(1,'\n');
fprintf(1,'Os valores de A U B sem elementos repetidos sao:\n');
for C = 1 : offset
  fprintf(1,'%f ',varC(C));
end
```

30) Seja

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

Escrever um algoritmo em PORTUGOL que:

a) Leia o valor de n, sendo n ≤ 20;
b) Leia os coeficientes a_i, $i = 0, 1, 2, ..., n;$;
c) Calcule o valor de P para 10 valores lidos para x;
d) Imprima o valor de x e o valor de P correspondente.

```
algoritmo L4P30;
var
  inteiro:  C, I, J, N;
  real:     P, XEXP, X[1..10], COEFA[0..20];
início
  imprima("Digite o valor de n: ");
  leia(N);
  para C de 0 até n faça
    imprima("Digite o coeficiente a",C, ": ");
    leia(COEFA[C]);
  fim-para
  para C de 1 até 10 faça
    imprima("Digite o ",C, ". valor de x: ");
    leia(X[C]);
  fim-para
  para C de 1 até 10 faça
    P <- COEFA[0];
    para I de 1 até N faça
      XEXP <- 1;
      para J de 1 até I faça
        XEXP := XEXP * X[C];
      fim-para
      P <- P + COEFA[I] * XEXP;
    fim-para;
    imprima("A soma P de x",C, " eh: ",P);
  fim-para
fim
```

```
program l4p30;
var C, I, J, N: integer;
    P, Xexp: real;
    X: array [1..10] of real;
    coefA: array [0..20] of real;
begin
  write('Digite o valor de n: ');
  readln(N);
  for C := 0 to N do
   begin
     write('Digite o coeficiente a',C,': ');
     readln(coefA[C]);
   end;
  for C := 1 to 10 do
   begin
     write('Digite o ',C,'. valor de x: ');
     readln(X[C]);
   end;
  for C := 1 to 10 do
   begin
     P := coefA[0];
     for I := 1 to N do
      begin
        Xexp := 1;
        for J := 1 to I do
          Xexp := Xexp * X[C];
        P := P + coefA[I] * Xexp;
      end;
     writeln('A soma P de x',C,' eh: ',P:10:4);
   end;
end.
```

```
N = input('Digite o valor de n: ');
if ( ( N < 1 ) | ( N > 20 ) )
  fprintf(1,'erro!\n');
else
  for C = 0 : N
    fprintf(1,'Digite o coeficiente a%d',C);
    coefA(C+1) = input(': ');
  end
  for C = 1 : 10
    fprintf(1,'Digite o %d',C);
    X(C) = input('. valor de x: ');
  end
  for C = 1 : 10
    P = coefA(1);
    for I = 1 : N
      Xexp = 1;
      for J = 1 : I
        Xexp = Xexp * X(C);
      end
      P = P + coefA(I+1) * Xexp;
    end
    fprintf(1,'A soma P de x %d eh: %f\n',C,P);
  end
end
```

31) Faça um algoritmo em PORTUGOL que leia um valor N (N ≤ 20) e os N valores de uma variável composta. Ordene os valores recebidos em forma crescente e imprima a variável composta ordenada.

```
algoritmo L4P31;
var
  inteiro: C, I, N, MENOR;
  real:    AUXILIAR, VETOR[1..20];
início
  imprima("Digite o valor N: ");
  leia(N);
  imprima("Digite os ",N," valores da variavel composta:");
  para C de 1 até N faça
    leia(VETOR[C]);
  fim-para
  para C de 1 até N - 1 faça
    MENOR <- C;
    para I de C + 1 até N faça
      se (VETOR[I] < VETOR[MENOR]) então
        MENOR <- I;
      fim-se
    fim-para
    AUXILIAR <- VETOR[MENOR];
    VETOR[MENOR] <- VETOR[C];
    VETOR[C] <- AUXILIAR;
  fim-para
  imprima("A variavel composta ordenada em ordem crescente eh:");
  para C de 1 até N faça
    imprima(VETOR[C]," ");
  fim-para
fim
```

```
algoritmo L4P31B;
var
  inteiro: C, I, N;
  real:    AUXILIAR, VETOR[1..20];
início
  imprima("Digite o valor N: ");
  leia(N);
  imprima("Digite os ",N," valores da variavel composta:");
  para C de 1 até N faça
    leia(VETOR[C]);
  fim-para
  para C de 1 até N - 1 faça
    para I de 1 até N - C faça
      se (VETOR[I] > VETOR[I + 1]) então
        AUXILIAR <- VETOR[I];
        VETOR[I] <- VETOR[I + 1];
        VETOR[I + 1] <- AUXILIAR;
      fim-se
    fim-para
  fim-para
  imprima("A variavel composta ordenada em ordem crescente eh:");
  para C de 1 até N faça
    imprima(VETOR[C]," ");
  fim-para
fim
```

```
program l4p31;
var
  C, I, N, menor: integer;
  auxiliar: real;
  vetor: array [1..20] of real;
begin
  write('Digite o valor N: ');
  readln(N);
  writeln('Digite os ',N,' valores da variavel composta:');
  for C := 1 to N do
    readln(vetor[C]);
  for C := 1 to N - 1 do
  begin
    menor := C;
    for I := C + 1 to N do
      if (vetor[I] < vetor[menor]) then
        menor := I;
    auxiliar := vetor[menor];
    vetor[menor] := vetor[C];
    vetor[C] := auxiliar;
  end;
  writeln('A variavel composta ordenada em ordem crescente eh:');
  for C := 1 to N do
    write(vetor[C]:6:3,' ');
end.
```

```
program l4p31b;
var
  C, I, N: integer;
  auxiliar: real;
  vetor: array [1..20] of real;
begin
  write('Digite o valor N: ');
  readln(N);
  writeln('Digite os ',N,' valores da variavel composta:');
  for C := 1 to N do
    readln(vetor[C]);
  for C := 1 to N - 1 do
    for I := 1 to N - C do
      if (vetor[I] > vetor[I + 1]) then
      begin
        auxiliar := vetor[I];
        vetor[I] := vetor[I + 1];
        vetor[I + 1] := auxiliar;
      end;
  writeln('A variavel composta ordenada em ordem crescente eh:');
  for C := 1 to N do
    write(vetor[C]:6:3,' ');
end.
```

```
N = input('Digite o valor N: ');
fprintf(1,'Digite os %d valores da variavel composta:',N);
for C = 1 : N
  vetor(C) = input('');
end
for C = 1 : N - 1
  menor = C;
  for I = C + 1 : N
    if (vetor(I) < vetor(menor))
      menor = I;
    end
  end
  auxiliar = vetor(menor);
  vetor(menor) = vetor(C);
  vetor(C) = auxiliar;
end
fprintf(1,'A variavel composta ordenada em ordem crescente eh:\n');
for C = 1 : N
  fprintf(1,'%f ',vetor(C));
end
```

```
N = input('Digite o valor N: ');
fprintf(1,'Digite os %d valores da variavel composta:',N);
for C = 1 : N
  vetor(C) = input('');
end
for C = 1 : N - 1
  for I = 1 : N - C
    if (vetor(I) > vetor(I + 1))
      auxiliar = vetor(I);
      vetor(I) = vetor(I + 1);
      vetor(I + 1) = auxiliar;
    end
  end
end
fprintf(1,'A variavel composta ordenada em ordem crescente eh:\n');
for C = 1 : N
  fprintf(1,'%f ',vetor(C));
end
```

32) Faça um algoritmo em PORTUGOL que leia um valor N (N ≤ 20) e os N valores de uma variável composta. Ordene os valores recebidos em forma decrescente e imprima a variável composta ordenada.

```
algoritmo L4P32;
var
  inteiro: C, I, N, MAIOR: integer;
  real:    AUXILIAR, VETOR[1..20];
início
  imprima("Digite o valor N: ");
  leia(N);
  imprima("Digite os ",N," valores da variavel composta:");
  para C de 1 até N faça
    leia(VETOR[C]);
  fim-para
  para C de 1 até N - 1 faça
    MAIOR <- C;
    para I de C + 1 até N faça
      se (VETOR[I] > VETOR[MAIOR]) então
        MAIOR <- I;
      fim-se
    fim-para
    AUXILIAR <- VETOR[MAIOR];
    VETOR[MAIOR] <- VETOR[C];
    VETOR[C] <- AUXILIAR;
  fim-para
  imprima("A variavel composta ordenada em ordem decrescente eh:");
  para C de 1 até N faça
    imprima(VETOR[C]," ");
  fim-para
fim
```

```
algoritmo L4P32B;
var
  inteiro: C, I, N;
  real:    AUXILIAR, VETOR[1..20];
início
  imprima("Digite o valor N: ");
  leia(N);
  imprima("Digite os ",N," valores da variavel composta:");
  para C de 1 até N faça
    leia(VETOR[C]);
  fim-para
  para C de 1 até N - 1 faça
    para I de 1 até N - C faça
      se (VETOR[I] < VETOR[I + 1]) então
        AUXILIAR <- VETOR[I];
        VETOR[I] <- VETOR[I + 1];
        VETOR[I + 1] <- AUXILIAR;
      fim-se
    fim-para
  fim-para
  imprima("A variavel composta ordenada em ordem decrescente eh:");
  para C de 1 até N faça
    imprima(VETOR[C]," ");
  fim-para
fim
```

```
program l4p32;
var
  C, I, N, maior: integer;
  auxiliar: real;
  vetor: array [1..20] of real;
begin
  write('Digite o valor N: ');
  readln(N);
  writeln('Digite os ',N,' valores da variavel composta:');
  for C := 1 to N do
    readln(vetor[C]);
  for C := 1 to N - 1 do
  begin
    maior := C;
    for I := C + 1 to N do
      if (vetor[I] > vetor[maior]) then
        maior := I;
    auxiliar := vetor[maior];
    vetor[maior] := vetor[C];
    vetor[C] := auxiliar;
  end;
  writeln('A variavel composta ordenada em ordem decrescente eh:');
  for C := 1 to N do
    write(vetor[C]:6:3,' ');
end.
```

```
program l4p32b;
var
  C, I, N: integer;
  auxiliar: real;
  vetor: array [1..20] of real;
begin
  write('Digite o valor N: ');
  readln(N);
  writeln('Digite os ',N,' valores da variavel composta:');
  for C := 1 to N do
    readln(vetor[C]);
  for C := 1 to N - 1 do
    for I := 1 to N - C do
      if (vetor[I] < vetor[I + 1]) then
      begin
        auxiliar := vetor[I];
        vetor[I] := vetor[I + 1];
        vetor[I + 1] := auxiliar;
      end;
  writeln('A variavel composta ordenada em ordem decrescente eh:');
  for C := 1 to N do
    write(vetor[C]:6:3,' ');
end.
```

```
N = input('Digite o valor N: ');
fprintf(1,'Digite os %d valores da variavel composta:',N);
for C = 1 : N
  vetor(C) = input('');
end
for C = 1 : N - 1
  maior = C;
  for I = C + 1 : N
    if (vetor(I) > vetor(maior))
      maior = I;
    end
  end
  auxiliar = vetor(maior);
  vetor(maior) = vetor(C);
  vetor(C) = auxiliar;
end
fprintf(1,'A variavel composta ordenada em ordem decrescente eh:\n');
for C = 1 : N
  fprintf(1,'%f ',vetor(C));
end
```

```
N = input('Digite o valor N: ');
fprintf(1,'Digite os %d valores da variavel composta:',N);
for C = 1 : N
  vetor(C) = input('');
end
for C = 1 : N - 1
  for I = 1 : N - C
    if (vetor(I) < vetor(I + 1))
      auxiliar = vetor(I);
      vetor(I) = vetor(I + 1);
      vetor(I + 1) = auxiliar;
    end
  end
end
fprintf(1,'A variavel composta ordenada em ordem decrescente eh:\n');
for C = 1 : N
  fprintf(1,'%f ',vetor(C));
end
```

33) Fazer algoritmo em PORTUGOL que:

a) Leia o valor inteiro de n (n ≤ 1000) e os n valores de uma variável composta de valores numéricos;
b) Ordenar a variável composta e imprimi-la ordenada.
c) Determine e imprima, para cada número que se repete no conjunto, a quantidade de vezes em que ele aparece repetido;

```
algoritmo L4P33;
var
  inteiro: C, I, n, NVEZES;
  real:    AUXILIAR, ANTERIOR, VETOR,[1..1000];
início
  imprima("Digite o valor de n: ");
  leia(n);
  imprima("Digite os numeros da variavel composta: ");
  para C de 1 até n faça
    leia(VETOR[C]);
  fim-para
  para C de 1 até n - 1 faça
    para I de 1 até n - C faça
      se (VETOR[I] > VETOR[I + 1]) então
        AUXILIAR <- VETOR[I];
        VETOR[I] <- VETOR[I + 1];
        VETOR[I + 1] <- AUXILIAR;
      fim-se
    fim-para
  fim-para
  imprima("O vetor ordenado eh: ");
  para C de 1 até n faça
    imprima(VETOR[C]);
  fim-para
  ANTERIOR <- VETOR[1];
  NVEZES <- 1;
  para C de 2 até n faça
    se (VETOR[C] = ANTERIOR) então
      NVEZES = NVEZES + 1;
    senão
      imprima("O numero ",ANTERIOR," se repete ",NVEZES," vezes");
      ANTERIOR <- VETOR[C];
      NVEZES <- 1;
    fim-se
  fim-para
  imprima("O numero ",ANTERIOR," se repete ",NVEZES," vezes");
fim
```

```
program l4p33;
var
  C, I, n, nvezes  : integer;
  auxiliar, anterior : real;
  vetor: array [1..1000] of real;
begin
  write('Digite o valor de n: ');
  readln(n);
  writeln('Digite os numeros da variavel composta: ');
  for C := 1 to n do
    readln(vetor[C]);
  for C := 1 to n - 1 do
    for I := 1 to n - C do
      if (vetor[I] > vetor[I + 1]) then
      begin
        auxiliar := vetor[I];
        vetor[I] := vetor[I + 1];
        vetor[I + 1] := auxiliar;
      end;
  writeln('O vetor ordenado eh: ');
  for C := 1 to n do
    write(vetor[C]:6:4, ' ');
  writeln('');
  anterior := vetor[1];
  nvezes := 1;
  for C := 2 to n do
  begin
    if (vetor[C] = anterior) then
      nvezes := nvezes + 1
    else
    begin
      writeln('O numero ',anterior:6:4,' se repete ',nvezes,' vezes');
      anterior := vetor[C];
      nvezes := 1;
    end;
  end;
  writeln('O numero ',anterior:6:4,' se repete ',nvezes,' vezes');
end.
```

```
n = input('Digite o valor de n: ');
fprintf(1,'Digite os numeros da variavel composta: \n');
for C = 1 : n
  vetor(C) = input('');
end
for C = 1 : N - 1
  for I = 1 : N - C
    if (vetor(I) > vetor(I + 1))
      auxiliar = vetor(I);
      vetor(I) = vetor(I + 1);
      vetor(I + 1) = auxiliar;
    end
  end
end
fprintf(1,'O vetor ordenado eh: \n');
for C = 1 : n
  fprintf(1,'%f ',vetor(C));
end
fprintf(1,'\n');
anterior = vetor(1);
nvezes = 1;
for C = 2 : n
  if (vetor(C) == anterior)
    nvezes = nvezes + 1;
  else
    fprintf(1,'O numero %f se repete %d vezes\n',anterior,nvezes);
    anterior = vetor(C);
    nvezes = 1;
  end
end
fprintf(1,'O numero %f se repete %d vezes\n',anterior,nvezes);
```

34) Numa corrida há 10 corredores, de número de inscrição de 1 a 10. Faça um algoritmo em PORTUGOL que leia os valores do número do corredor e o seu respectivo tempo na corrida. Além disso, o programa deve imprimir a qualificação e o tempo de corrida, do primeiro ao décimo colocado, identificando o número de inscrição do corredor referente àquela colocação. Suponha que não há tempos iguais.

```
algoritmo L4P34;
var
  inteiro: C, I, AUX, NUMERO[1..10];
  real:    AUXILIAR, TEMPO[1..10];
início
  para C de 1 até 10 faça
    imprima("Digite o numero de inscricao do corredor: ");
    leia(NUMERO[C]);
    imprima("Digite o tempo de corrida deste corredor: ");
    leia(TEMPO[C]);
  fim-para
  para C de 1 até 9 faça
    para I de 1 até 10 - C faça
      se (TEMPO[I] > TEMPO[I + 1]) então
        AUXILIAR <- TEMPO[I];
        TEMPO[I] <- TEMPO[I + 1];
        TEMPO[I + 1] <- AUXILIAR;
        AUX <- NUMERO[I];
        NUMERO[I] <- NUMERO[I + 1];
        NUMERO[I + 1] <- AUX;
      fim-se
    fim-para
  fim-para
  para C de 1 até 10 faça
    imprima("O ",C,". foi o de numero ",NUMERO[C]," com o tempo de ",TEMPO[C]);
  fim-para
fim
```

```
program  l4p34;
var
  C, I, aux: integer;
  auxiliar: real;
  numero: array [1..10] of integer;
  tempo: array [1..10] of real;
begin
  for C := 1 to 10 do
  begin
    write('Digite o numero de inscricao do corredor: ');
    readln(numero[C]);
    write('Digite o tempo de corrida deste corredor: ');
    readln(tempo[C]);
  end;
  for C := 1 to 9 do
    for I := 1 to 10 - C do
      if (tempo[I] > tempo[I + 1]) then
      begin
        auxiliar := tempo[I];
        tempo[I] := tempo[I + 1];
        tempo[I + 1] := auxiliar;
        aux := numero[I];
        numero[I] := numero[I + 1];
        numero[I + 1] := aux;
      end;
  for C := 1 to 10 do
     writeln('O ',C,'. foi o de numero ',numero[C],' com o tempo de ',tempo[C]:3:3);
end.
```

```
for C = 1 : 10
  numero(C) = input('Digite o numero de inscricao do corredor: ');
  tempo(C) = input('Digite o tempo de corrida deste corredor: ');
end
for C = 1 : 9
  for I = 1 : 10 - C
    if (tempo(I) > tempo(I + 1))
      auxiliar = tempo(I);
      tempo(I) = tempo(I + 1);
      tempo(I + 1) = auxiliar;
      aux = numero(I);
      numero(I) = numero(I + 1);
      numero(I + 1) = aux;
    end
  end
end
for C = 1 : 10
  fprintf(1,'O %d. foi o de numero %d com o tempo de %f\n',C,numero(C),tempo(C));
end
```

35) Faça um algoritmo em PORTUGOL que leia uma variável composta de N valores numéricos (N ≤ 20) e ordene essa variável em ordem crescente. O programa também deve ler um número k e imprimir, antes e depois da ordenação, o k-ésimo termo da variável composta.

```
algoritmo L4P35;
var
  inteiro: C, I, K, N;
  real:    AUX, VET[1..20];
início
  imprima("Digite o valor de N: ");
  leia(N);
  imprima("Digite os valores numericos da variavel composta: ");
  para C de 1 até N faça
    leia(VET[C]);
  fim-para
  imprima("Digite o valor de K: ");
  leia(K);
  imprima("O k-esimo termo antes da ordenacao eh: ",vetor[K]);
  para C de 1 até N - 1 faça
    para I de 1 até N - C faça
      se (VET [I] > VET[I + 1]) então
        AUX <- VET[I];
        VET[I] <- VET[I + 1];
        VET[I + 1] <- AUX;
      fim-se
    fim-para
  fim-para
  imprima("O k-esimo termo antes da ordenacao eh: ",VET[K]);
fim
```

```
program l4p35;
var C, I, K, N : integer;
    AUX : real;
    VETOR: array [1..20] of real;
begin
  write('Digite o valor de N: ');
  readln(N);
  writeln('Digite os valores numericos da variavel composta: ');
  for C := 1 to N do
    readln(VET[C]);
  write('Digite o valor de K: ');
  readln(K);
  writeln('O k-esimo termo antes da ordenacao eh: ',VET[K]:6:3);
  for C := 1 to N - 1 do
   for I := 1 to N - C do
     if (VET[I] > VET[I + 1]) then
      begin
        AUX := VET[I];
        VET[I] := VET[I + 1];
        VET[I + 1] := AUX;
      end;
  writeln('O k-esimo termo depois da ordenacao eh: ',VET[K]:6:3);
end.
```

```
N = input('Digite o valor de N: ');
fprintf(1,'Digite os valores numericos da variavel composta: \n');
for C = 1 : N
  VET(C) = input('');
end
K = input('Digite o valor de K: ');
fprintf(1,'O k-esimo termo antes da ordenacao eh: %f\n',VET(K));
for C = 1 : N - 1
  for I = 1 : N - C
    if (VET(I) > VET(I + 1))
      AUX = VET(I);
      VET(I) = VET(I + 1);
      VET(I + 1) = AUX;
    end
  end
end
fprintf(1,'O k-esimo termo depois da ordenacao eh: %f',VET(K));
```