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 06 – Modularização (Procedimentos e Funções)

- ***Procedimentos: Passagem de parâmetros.***

1) Escreva um procedimento que receba um número inteiro e imprima o mês correspondente ao número. Por exemplo, 2 corresponde à "fevereiro". O procedimento deve mostrar uma mensagem de erro caso o número recebido não faça sentido. Gere também um algoritmo que leia um valor e chame o procedimento criado.

```
algoritmo L6P01;
var
  inteiro : X;

procedimento MES(inteiro:N);
constante
  M[1..12,10] = ("Janeiro","Fevereiro","Março","Abril","Maio","Junho",
                 "Julho","Agosto","Setembro","Outubro","Novembro","Dezembro");
início
  se ( ( N < 1 ) ou ( N > 12 ) ) então
    imprima("O numero não corresponde a um mes. ");
  senão
    imprima(N, "corresponde a ",M[N]);
  fim-se
fim

início
  leia(X);
  MES(X);
fim
```

```
program L6P01;
var
  X : integer;

procedure MES(N : integer);
const
  M : array[1..12] of string[10] =
    ('Janeiro','Fevereiro','Marco','Abril','Maio','Junho',
     'Julho','Agosto','Setembro','Outubro','Novembro','Dezembro');
begin
  if ((N<1) or (N>12)) then
    writeLn('O numero nao corresponde a um mes!')
  else
    writeLn('O numero corresponde ao mes ',M[N]);
end;

begin
  write('Entre com um numero: ');
  readLn(X);
  MES(X);
end.
```

```
function L6P01;
X = input('Entre com um numero: ');
MES(X);

function MES(NUM);
SM = {'Janeiro','Fevereiro','Março','Abril','Maio','Junho', ...
    'Julho','Agosto','Setembro','Outubro','Novembro','Dezembro'};
if ( (NUM < 1 ) | ( NUM > 12 ) )
  disp('O numero nao corresponde a um mes.');
else
  fprintf(1,'%d corresponde a %s\n',NUM,SM{NUM});
end
```

2) Escreva um procedimento que receba um número inteiro e o imprima na forma extensa. Por exemplo, para 1 a saída desejada é "Um". A função deve ser capaz de gerar o extenso dos números de 0 até 10, inclusive. Caso um número não compatível seja recebido o procedimento deve mostrar uma mensagem de erro. Crie também um algoritmo que leia um valor inteiro e chame o procedimento criado acima para a impressão do número extenso.

```
algoritmo L6P02;
var
  inteiro : N;

procedimento NE(inteiro : X);
constante
  NUM[0..10,7] =
   ("Zero","Um","Dois","Tres","Quatro","Cinco","Seis","Sete","Oito","Nove","Dez");
início
  se ( X < 0 ) ou ( X > 10) então
    imprima("Erro! Número inválido!");
  senão
    imprima(NUM[X]);
  fim-se
fim

início
  leia(N);
  NE(N);
fim
```

```
program L6P02 ;
var
  N : integer ;

procedure NE ( X : integer ) ;
  const NUM: array [0..10] of string[7] =
    ('Zero','Um','Dois','Tres','Quatro','Cinco','Seis','Sete','Oito','Nove','Dez');
begin
  if ( X < 0 ) or ( X > 10 ) then
    writeLn('Erro! Numero invalido!')
  else
    writeLn(NUM[X]);
end;

begin
  write('Entre com um numero: ');
  readLn( N );
  NE ( N );
end.
```

```
function L6P02;
N = input('Entre com um número: ');
NE(N);

function NE ( X );
NUM = {'Zero';'Um';'Dois';'Tres';'Quatro';'Cinco';'Seis';'Sete';'Oito';'Nove';'Dez'};
if ( X < 0 ) | ( X > 10 )
  disp('Erro! Número Invalido!');
else
  disp(NUM{X+1});
end
```

3) Escreva um procedimento que gere um cabeçalho para um relatório. Esse procedimento deve receber um literal (*string*, ou cadeia de caracteres) como parâmetro. O cabeçalho tem a seguinte forma:

```
============================================
 UFMG - Universidade Federal de Minas Gerais
 ICEx - Instituto de Ciências Exatas
 Disciplina de Programação de Computadores
 Nome: Fulano de Tal
============================================
```

onde **Fulano de Tal**, corresponde ao parâmetro passado.

```
algoritmo L6P03;
var
  literal : X;

procedimento CAB( literal : NOME );
início
  imprima("============================================");
  imprima(" UFMG - Universidade Federal de Minas Gerais");
  imprima(" ICEx - Instituto de Ciencias Exatas");
  imprima(" Disciplina de Programacao de Computadores");
  imprima(" NOME: ",NOME);
  imprima("============================================");
fim

início
  leia(X);
  CAB(X);
fim
```

```
program L6P03;
var
  X : string;

procedure CAB(NOME : string);
begin
  writeLn('============================================');
  writeLn(' UFMG - Universidade Federal de Minas Gerais');
  writeLn(' ICEx - Instituto de Ciencias Exatas');
  writeLn(' Disciplina de Programacao de Computadores');
  writeLn(' NOME: ',NOME);
  writeLn('============================================');
end;

begin
  write('Informe o nome: ');
  readLn(X);
  CAB(X);
end.
```

```
function L6P03;
X = input('Entre com um nome: ','s');
CAB(X);

function CAB(NOME);
fprintf(1,'============================================\n');
fprintf(1,' UFMG - Universidade Federal de Minas Gerais\n');
fprintf(1,' ICEx - Instituto de Ciencias Exatas\n');
fprintf(1,' Disciplina de Programacao de Computadores\n');
fprintf(1,' Nome: %s\n',NOME);
fprintf(1,'============================================\n');
```

4) Escreva um procedimento que receba um número arábico inteiro e imprima o corresponde número em romano. Por exemplo, para 5 a saída desejada é "V". A função deve ser capaz de gerar o número romano para os 50 primeiros inteiros. Uma mensagem de erro deve ser mostrada caso um número fora dessa faixa seja recebido. Crie também um algoritmo que leia um valor inteiro e chame o procedimento criado acima para a impressão do número romano.

```
algoritmo L6P04;
var
  inteiro : X , I , C;

procedimento ROM(inteiro : N);
início
  se ( N < 1 ) ou ( N > 50 ) então
    imprima("erro!");
  senão
    se ( N = 50 ) então
      imprima("L");
    senão  {N<50}
      se ( N >= 40 ) então
        imprima("XL");
      senão  {N<40}
        para I de 1 até (N div 10) faça
          imprima("X");
        fim-para
      fim-se
    fim-se
    C <- N mod 10;
    enquanto (C =< 3) e (C > 0) faça
      imprima("I");
      C <- C - 1;
    fim-enquanto
    se ( C = 4 ) então
      imprima("IV");
    fim-se
    se ( C = 9 ) então
      imprima("IX");
    senão
      se ( C >= 5 ) então
        imprima("V");
        C <- C - 5;
        se ( C > 0 ) e ( C =< 3 ) então
          para I de 1 até C faça
            imprima("I");
          fim-para
        fim-se
      fim-se
    fim-se
  fim-se
fim

início
  leia(X);
  ROM(X);
fim
```

```
program L6P04;
var
  X , I , C : integer;

procedure ROM( N : integer);
begin
  if ( ( N < 1 ) or ( N > 50 ) ) then
    writeLn('Erro!')
  else
  begin
    if ( N = 50) then
      write('L')
    else
      if (N >= 40) then
        write('XL')
      else
        for I := 1 to (N div 10) do
          write('X');
    C := N mod 10;
    while ( ( C <= 3 ) and ( C > 0 ) ) do
    begin
      write('I');
      C := C - 1;
    end;
    if ( C = 4 ) then
      write('IV');
    if ( C = 9 ) then
      write('IX')
    else
      if ( C >= 5) then
      begin
        write('V');
        C := C - 5;
        if ( ( C > 0 ) and ( C <= 3 ) ) then
          for I := 1 to C do
            write('I');
        end;
      end;
    writeLn('');
  end;

begin
  write('Entre com um numero: ');
  readLn(X);
  ROM(X);
end.
```

```
function L6P04;
X = input('Entre com um número: ');
ROM(X);

function ROM(N);
if ( ( N < 1 ) | ( N > 50 ) )
  disp('Erro!');
else
  if ( N == 50 )
    fprintf(1,'L');
  else
    if ( N >= 40 )
      fprintf(1,'XL');
    else
      A = floor( N / 10 );
      for I = 1 : A
        fprintf(1,'X');
      end
    end
  end
  C = mod( N , 10 );
  while ( ( C <= 3 ) & ( C > 0 ) )
    fprintf(1,'I');
    C = C - 1;
  end
  if ( C == 4 )
    fprintf(1,'IV');
  end
  if ( C == 4 )
    fprintf(1,'IX');
  else
    if ( C >= 5 )
      fprintf(1,'V');
      C = C - 5;
      if ( ( C > 0 ) & ( C <= 3 ) )
        fprintf(1,'I');
      end
    end
  end
  fprintf(1,'\n');
end
```

5) Escreva um procedimento que receba um número natural e imprima os três primeiros caracteres do dia da semana correspondente ao número. Por exemplo, 7 corresponde à "SAB". O procedimento deve mostrar uma mensagem de erro caso o número recebido não corresponda à um dia da semana. Gere também um algoritmo que utilize esse procedimento, chamando-o, mas antes lendo um valor para passagem de parâmetro.

```
algoritmo L6P05;
var
  inteiro : X;

procedimento DIASEMANA(inteiro : NUM);
constante
  SEM[1..7,3] = ("DOM","SEG","TER","QUA","QUI","SEX","SAB");
início
  se ( NUM >= 1 ) e ( NUM <= 7 ) então
    imprima(NUM," corresponde a: ",SEM[NUM])
  senão
    imprima("parametro recebido (",NUM,") nao correspondente a um dia da semana!");
  fim-se
fim

início
  leia(X);
  DIASEMANA(X);
fim
```

```
program L6P05;
var
  X: integer;

procedure DIASEMANA(NUM: integer);
const SEM: array [1..7] of string[3] =
  ('DOM','SEG','TER','QUA','QUI','SEX','SAB');
begin
  if ( NUM >= 1 ) and ( NUM <= 7 ) then
    writeLn(NUM,' corresponde a: ',SEM[NUM])
  else
    writeLn('parametro recebido (',NUM,') nao correspondente a um dia da semana!');
end;

begin
  write('Entre com um numero: ');
  readLn(X);
  DIASEMANA(X);
end.
```

```
function L6P05;
X = input('Entre com um numero: ');
DIASEMANA(X);

function DIASEMANA(NUM);
SEM = {'DOM';'SEG';'TER';'QUA';'QUI';'SEX';'SAB'};
if ( NUM >= 1 ) & ( NUM <= 7 )
  fprintf(1,'%d corresponde a: %s\n',NUM,SEM{NUM});
else
  fprintf(1,'parametro recebido (%d) nao correspondente a um dia da semana!\n',NUM);
end
```

- ***Funções que verificam uma situação, retorno booleano (verdadeiro, falso)***

6) Escreva uma função que receba um número inteiro. Esta função deve verificar se tal número é primo. No caso positivo, a função deve retornar 1, caso contrário zero. Escreva também um algoritmo para testar tal função.

```
algoritmo L6P06;
var
  inteiro : X;

função PRIMO(inteiro : N) : inteiro;
var
  inteiro : C;
início
  se ( N < 2 ) então
    PRIMO <- 0;
  senão
    PRIMO <- 1;
    para C de 2 até N-1 faça
      se ( (N mod C) = 0 ) então
        PRIMO <- 0;
      fim-se
    fim-para
  fim-se
fim

início
  leia(X);
  se ( PRIMO(X) = 1 ) então
    imprima("É primo!");
  senão
    imprima("Não é primo!");
  fim-se
fim
```

```
program L6P06;
var
  X : integer;

function PRIMO (N : integer) : integer;
var
  C : integer;
begin
  if ( N < 2 ) then
    PRIMO := 0
  else
  begin
    PRIMO := 1;
    for C := 2 to (N-1) do
      if ( (N mod C) = 0 ) then
        PRIMO := 0;
  end;
end;

begin
  write('Entre com um numero: ');
  readLn(X);
  if ( PRIMO(X) = 1 ) then
    writeLn('Eh primo!')
  else
    writeLn('Nao eh primo!');
end.
```

```
function L6P06;
X = input('Entre com um numero: ');
if ( PRIMO(X) == 1 )
  disp('Eh primo!');
else
  disp('Nao eh primo!');
end

function RET = PRIMO(N);
if ( N < 2 )
  RET = 0;
else
  RET = 1;
  for C = 2 : (N-1)
    if ( mod(N,C) == 0 )
      RET = 0;
    end
  end
end
```

7) Escreva uma função que receba dois números inteiros ***x*** e ***y***. Essa função deve verificar se ***x*** é divisível por ***y***. No caso positivo, a função deve retornar 1, caso contrário zero. Escreva também um algoritmo para testar tal **função**.

```
algoritmo L6P07;
var
  inteiro : A, B;

função DVS(inteiro : X, Y): inteiro;
início
  se ( Y = 0 ) então
    DVS <- 0;
  senão
    se ( (X mod Y) = 0 ) então
      DVS <- 1;
    senão
      DVS <- 0;
    fim-se
  fim-se
fim

início
  leia(A,B);
  se ( DVS(A,B) = 1 ) então
    imprima(A," eh divisivel por ",B);
  senão
    imprima(A," não eh divisivel por ",B);
  fim-se
fim
```

```
program L6P07;
var
  A, B : integer;

function DVS(X, Y : integer) : integer;
begin
  if ( Y = 0 ) then
    DVS := 0
  else
    if ( X mod Y = 0 ) then
      DVS := 1
    else
      DVS := 0;
end;

begin
  write('Entre com um valor: ');
  readLn(A);
  write('Entre com um valor: ');
  readLn(B);
  if ( DVS(A,B) = 1 ) then
    writeLn(A,' eh divisivel por ',B)
  else
    writeLn(A,' eh divisivel por ',B);
end.
```

```
function L6P07;
A = input('Entre com um valor: ');
B = input('Entre com um valor: ');
if ( DVS(A,B) == 1 )
  fprintf(1,'%d eh divisivel por %d\n',A,B);
else
  fprintf(1,'%d nao eh divisivel por %d\n',A,B);
end

function RET = DVS(X,Y);
if ( Y == 0 )
  RET = 0;
else
  if ( mod(X,Y) == 0 )
    RET = 1;
  else
    RET = 0;
  end
end
```

8) Um número é dito ser regular caso sua decomposição em fatores primos apresenta apenas potências de 2, 3 e 5. Faça uma função que verifique se um número é (retorne 1) ou não (retorne 0) regular. Escreva também um algoritmo para testar tal função.

```
algoritmo L6P08;
var
  inteiro : X;

função REGULAR(inteiro : N ) : inteiro;
início
  enquanto ( ( N mod 2 ) = 0 ) faça
    N <- N/2;
  fim-enquanto
  enquanto ( ( N mod 3 ) = 0 ) faça
    N <- N/3;
  fim-enquanto
  enquanto ( ( N mod 5 ) = 0 ) faça
    N <- N/5;
  fim-enquanto
  se ( N = 1 ) então
    REGULAR <- 1;
  senão
    REGULAR <- 0;
  fim-se
fim

início
  leia(X);
  se ( REGULAR(X) = 1 ) então
    imprima("Eh regular!");
  senão
    imprima("Não eh regular!");
  fim-se
fim
```

```
program L6P08;
var
  X : integer;

function REGULAR(N : integer) : integer;
begin
  while ( ( N mod 2 ) = 0 ) do
    N := N div 2;
  while ( ( N mod 3 ) = 0 ) do
    N := N div 3;
  while ( ( N mod 5 ) = 0 ) do
    N := N div 5;
  if (N=1) then
    REGULAR := 1
  else
    REGULAR := 0;
end;

begin
  write('Entre com um numero: ');
  readLn(X);
  if ( REGULAR(X) = 1 ) then
    writeLn('Eh regular!')
  else
    writeLn('Nao eh regular!');
end.
```

```
function L6P08;
X = input('Entre com um numero: ');
if (REGULAR(X) == 1)
  disp('Eh regular!');
else
  disp('Nao eh regular!');
end

function RET = REGULAR(N);
while ( mod( N , 2 ) == 0)
  N = N / 2;
end
while ( mod( N , 3 ) == 0 )
  N = N / 3;
end
while ( mod( N , 5 ) == 0 )
  N = N / 5;
end
if ( N == 1 )
  RET = 1;
else
  RET = 0;
end
```

9) Criar uma função que determine se um caractere, recebido como parâmetro, é ou não uma letra do alfabeto. A função deve retornar 1 caso positivo e 0 em caso contrário. Escreva também um algoritmo para testar tal função.

```
algoritmo L6P09;
var
  caractere : X;

função ALF(caractere : N) : inteiro;
constante
  A[1..26,1] = ("A","B","C","D","E","F","G","H","I","J","K","L","M",
                "N","O","P","Q","R","S","T","U","V","W","X","Y","Z");
var
  inteiro : C;
início
  ALF <- 0;
  para C de 1 até 26 faça
    se (N = A[C]) então
      ALF <- 1;
    fim-se
  fim-para
fim

início
  leia(X);
  se ( ALF(X) = 1 ) então
    imprima("Pertence ao alfabeto");
  senão
    imprima("Não pertence ao alfabeto");
  fim-se
fim
```

```
program L6p09;
var
  X : char;

function ALF(N : char) : integer;
const
  A : array[1..26] of char =
    ('A','B','C','D','E','F','G','H','I','J','K','L','M',
     'N','O','P','Q','R','S','T','U','V','W','X','Y','Z');
var
  C : integer;
begin
  ALF := 0;
  for C := 1 to 26 do
    if ( N = A[C] ) then
      ALF := 1;
end;

begin
  write('Entre com um caractere: ');
  readLn(X);
  if ( ALF(X) = 1 ) then
    writeLn('Pertence ao alfabeto')
  else
    writeLn('Nao pertence ao alfabeto');
end.
```

```
function L6P09;
X = input('Entre com um caractere: ','s');
if ( ALF(X) == 1 )
  disp('Pertence ao alfabeto');
else
  disp('Nao pertence ao alfabeto');
end

function RET = ALF(N);
A = {'A','B','C','D','E','F','G','H','I','J','K','L','M', ...
     'N','O','P','Q','R','S','T','U','V','W','X','Y','Z'};
RET = 0;
for C = 1 : 26
  if (N == A{C})
    RET = 1;
  end
end
```

10) Um número é dito ser **capicua** quando lido da esquerda para a direita é o mesmo que quando lido da direita para a esquerda. O ano 2002, por exemplo, é **capicua**. Então, elabore uma função para verificar se um número possui essa característica. Caso o número seja **capicua**, a função deve retornar 1 e 0 em caso contrário. Escreva também um algoritmo para testar tal função.

```
algoritmo L6P10;
var
  inteiro : X;

função REVERSO(inteiro : NUM) :inteiro;
var
  inteiro: RET, MUL, REV;
início
  REV <- 0;
  enquanto ( NUM <> 0 ) faça
    RET <- NUM mod 10;
    NUM <- NUM div 10;
    REV <- REV * 10 + RET;
  fim-enquanto
  REVERSO <- REV;
fim

função CAPICUA(inteiro : NUM) : lógico;
início
  se (REVERSO(NUM) = NUM) então
    CAPICUA <- verdadeiro;
  senão
    CAPICUA <- falso;
  fim-se
fim

início
  leia(X);
  se ( CAPICUA(X) ) então
    imprima(X," eh capicua!");
  senão
    imprima(X," nao eh capicua!");
  fim-se
fim
```

```
program L6P10;
var
  X : integer;

function REVERSO(NUM : integer) : integer;
var
  RET, MUL, REV: integer;
begin
  REV := 0;
  while ( NUM <> 0 ) do
  begin
    RET := NUM mod 10;
    NUM := NUM div 10;
    REV := REV * 10 + RET;
  end;
  REVERSO := REV;
end;

function CAPICUA(NUM : integer) : boolean;
begin
  if (REVERSO(NUM) = NUM) then
    CAPICUA := true
  else
    CAPICUA := false;
end;

begin
  write('Entre com um numero: ');
  readLn(X);
  if ( CAPICUA(X) ) then
    writeLn(X,' eh capicua!')
  else
    writeLn(X,' nao eh capicua!');
end.


function L6P10;
X = input('Entre com um numero: ');
if ( CAPICUA(X) == 1 )
  fprintf(1,'%d eh capicua!\n',X);
else
  fprintf(1,'%d nao eh capicua!\n',X);
end

function REV = REVERSO(NUM);
REV = 0;
while ( NUM ~= 0 )
  RET = mod(NUM,10);
  NUM = floor(NUM / 10);
  REV = REV * 10 + RET;
end

function RET = CAPICUA(NUM);
if (REVERSO(NUM) == NUM)
  RET = 1;
else
  RET = 0;
end
```

- ***Funções que retornam um valor calculado***

11) Criar uma função (não recursiva) que calcule e retorne o valor do fatorial de um número natural. A função deve retornar -1 caso não seja possível calcular o valor do fatorial. Escreva também um algoritmo para testar tal função.

```
algoritmo L6P11;
var
  inteiro : X, VAL;

função FAT(inteiro : N) : inteiro;
var
  inteiro : C , F;
início
  se ( N > 0 ) então
    F <- 1;
    para C de 2 até N faça
      F <- F * C;
    fim-para
  senão
    se ( N < 0 ) então
      F <- -1;
    senão
      F <- 1;
    fim-se
  fim-se
  FAT <- F;
fim

início
  leia(X);
  VAL <- FAT(X);
  se ( VAL = -1 ) então
    imprima('Não existe fatorial de ',X);
  senão
    imprima('O fatorial do numero ',X,' eh ',VAL);
  fim-se
fim
```

```
program L6P11;
var
  X , VAL : integer;

function FAT(N : integer) : integer;
var
  C , F : integer;
begin
  if ( N >= 0) then
  begin
    F := 1;
    for C := 2 to N do
      F := F * C;
  end
  else
    if ( N < 0 ) then
      F := -1;
  FAT := F;
end;

begin
  write('Entre com um numero: ');
  readLn(X);
  VAL := FAT(X);
  if ( VAL = -1 ) then
    writeLn('Nao existe fatorial para ',X)
  else
    writeLn('O fatorial do numero ',X,' eh ',VAL);
end.
```

```
function L6P11;
N = input('Entre com um numero: ');
VAL = FAT(N);
if ( N == -1 )
  fprintf(1,'Nao existe fatorial de %d\n',N);
else
  fprintf(1,'O fatorial do numero %d eh: %d\n',N,VAL);
end

function RET = FAT(N);
if ( N >= 0 )
  RET = 1;
  for C = 1 : N
    RET = RET * C;
  end
else
  RET = -1;
end
```

12) Criar uma função que calcule e retorne o número de arranjos de *n* elementos *p* a *p*. A fórmula do arranjo é a seguinte:

$$A_p^n = \frac{n!}{(n-p)!}$$

Caso não seja capaz de calcular tal arranjo a função deve retornar -1. Um algoritmo para testar tal função também deve ser escrito.

```
algoritmo L6P12;
var
  inteiro : NL, PL, VAL;

função FAT(inteiro : M) inteiro;
var
  inteiro : C;
início
  FAT <- 1;
  para C de 2 até M faça
    FAT <- FAT * C;
  fim-para
fim

função ARRNJ(inteiro : N, P) : inteiro;
var
  inteiro : C, FATN, FATNP;
início
  se ( N > 0 ) e ( P > 0 ) e ( N > P ) então
    FATN  <- FAT(N);
    FATNP <- FAT(N-P);
    ARRNJ <- FATN div FATNP;
  senão
    ARRNJ <- -1;
  fim-se
fim

início
  leia(NL,PL);
  VAL <- ARRNJ(NL,PL);
  se ( VAL = -1 ) então
    imprima("Impossivel calcular o arranjo de ",NL,", ",PL," a ",PL," elementos");
  senão
    imprima("Combinacao: ",VAL);
  fim-se
fim
```

```
program L6P12;
var
  NL , PL, VAL : integer;

function FAT( M : integer) : integer;
var
  C, F : integer;
begin
  F := 1;
  for C := 2 to M do
    F := F * C;
  FAT := F;
end;

function ARRNJ(N , P : integer) : integer;
var
  C, FATN , FATNP : integer;
begin
  if ( N > 0 ) and ( P > 0 ) and ( N > P ) then
  begin
    FATN  := FAT(N);
    FATNP := FAT(N-P);
    ARRNJ := FATN div FATNP;
  end
  else
    ARRNJ := -1;
end;

begin
  write('Entre com um valor: ');
  readLn(NL);
  write('Entre com um valor: ');
  readLn(PL);
  VAL := ARRNJ(NL,PL);
  if ( VAL = -1 ) then
    writeLn('Impossivel calcular o arranjo de ',NL,', ',PL,' a ',PL,' elementos')
  else
    writeLn('Arranjo: ',VAL);
end.
```

```
function L6P12;
NL = input('Informe o valor de N: ');
PL = input('Informe o valor de P: ');
VAL = ARRNJ(NL,PL);
if ( VAL == -1 )
  fprintf(1,'Impossivel calcular o arranjo de %d, %d a %d elementos',NL,PL,PL);
else
  fprintf(1,'Arranjo: %.0f\n',VAL);
end

function F = FAT(M);
F = 1;
for C = 2 : M
  F = F * C;
end

function RET = ARRNJ(N,P);
if ( N > 0 ) & ( P > 0 ) & ( N > P )
  FATN  = FAT(N);
  FATNP = FAT(N-P);
  RET = FATN/FATNP;
else
  RET = -1;
end
```

13) Criar uma função que calcule e retorne o número de combinações de *n* elementos *p* a *p*. A fórmula de combinação é a seguinte:

$$C_p^n = \frac{n!}{p!(n-p)!}$$

Caso não seja capaz de calcular tal combinação a função deve retornar -1. Um algoritmo para testar tal função também deve ser escrito.

```
algoritmo L6P13;
var
  inteiro : NL, PL, VAL;

função FAT(inteiro : M) inteiro;
var
  inteiro : C;
início
  FAT <- 1;
  para C de 2 até M faça
    FAT <- FAT * C;
  fim-para
fim

função COMB(inteiro : N, P) : inteiro;
var
  inteiro : C, FATN, FATP, FATNP;
início
  se ( N > 0 ) e ( P > 0 ) e ( N > P ) então
    FATN  <- FAT(N);
    FATP  <- FAT(P);
    FATNP <- FAT(N-P);
    COMB <- FATN/(FATP*FATNP);
  senão
    COMB <- -1;
  fim-se
fim

início
  leia(NL,PL);
  VAL <- COMB(NL,PL);
  se ( VAL = -1 ) então
    imprima("Impossivel calcular o arranjo de ",NL,", ",PL," a ",PL," elementos");
  senão
    imprima("Combinacao: ",VAL);
  fim-se
fim
```

```
program L6P13;
var
  NL , PL, VAL : integer;

function FAT( M : integer) : integer;
var
  C, F : integer;
begin
  F := 1;
  for C := 2 to M do
    F := F * C;
  FAT := F;
end;

function COMB(N , P : integer) : integer;
var
  C, FATN , FATP, FATNP : integer;
begin
  if ( N > 0 ) and ( P > 0 ) and ( N > P ) then
  begin
    FATN  := FAT(N);
    FATP  := FAT(P);
    FATNP := FAT(N-P);
    COMB := FATN div (FATP * FATNP);
  end
  else
    COMB := -1;
end;

begin
  write('Entre com um valor: ');
  readLn(NL);
  write('Entre com um valor: ');
  readLn(PL);
  VAL := COMB(NL,PL);
  if ( VAL = -1 ) then
    writeLn('Impossivel calcular a combinação de ',NL,', ',PL,' a ',PL,' elementos')
  else
    writeLn('Combinacao: ',VAL);
end.


function L6P13;
NL = input('Entre com um valor: ');
PL = input('Entre com um valor: ');
VAL = COMB(NL,PL);
if ( VAL == -1 )
  fprintf(1,'Impossivel calcular a combinação de %d, %d a %d elementos',N,P,P);
else
  fprintf(1,'Combinacao: %.0f\n',VAL);
end

function F = FAT(M);
F = 1;
for C = 2 : M
  F = F * C;
end

function RET = COMB(N,P);
if ( N > 0 ) & ( P > 0 ) & ( N > P )
  FATN  = FAT(N);
  FATP  = FAT(P);
  FATNP = FAT(N-P);
  RET = FATN / ( FATP * FATNP );
else
  RET = -1;
end
```

14) Criar uma função que calcule e retorne o ***MAIOR*** entre dois valores recebidos como parâmetro. Um algoritmo para testar tal função deve ser criado.

```
algoritmo L6P14;
var
  real : A , B, VAL;

função MAIOR(real: X, Y): real;
início
  se ( X > Y ) então
    MAIOR <- X;
  senão
    MAIOR <- Y;
  fim-se
fim

início
  leia(A,B);
  VAL <- MAIOR(A,B)
  imprima("Maior: ",VAL);
fim
```

```
program L6p14;
var
  A, B, VAL: real;

function MAIOR( X , Y : real ) : real;
begin
  if ( X > Y) then
    MAIOR := X
  else
    MAIOR := Y;
end;

begin
  write('Entre com um valor: ');
  readLn(A);
  write('Entre com um valor: ');
  readLn(B);
  VAL := MAIOR(A,B);
  writeLn('Maior: ', VAL );
end.
```

```
function L6P14;
A = input('Entre com um valor: ');
B = input('Entre com um valor: ');
fprintf(1,'Maior: %f\n', MAIOR(A,B) );

function RET = MAIOR( X , Y );
if ( X > Y )
  RET = X;
else
  RET = Y;
end
```

15) Criar uma função que verifique quantas vezes um número inteiro ***x*** é divisível por um número inteiro ***y***. A função deve retornar -1 caso não seja possível calcular. Escreva também um algoritmo para testar tal função.

```
algoritmo L6P15;
var
  inteiro : A, B, VAL;

função VEZDIV(inteiro: X, Y) : inteiro;
var
  inteiro: VEZ, RET;
início
  se ( Y = 0 ) então
    VEZDIV <- -1; {divisao por zero}
  senão
    VEZ <- 0;
    RET <- X mod Y;
    enquanto ( RET = 0 ) faça
      VEZ <- VEZ + 1;
      X   <- X div Y;
      RET <- X mod Y;
    fim-enquanto
    VEZDIV <- VEZ;
  fim-se
fim

início
  leia(A,B);
  VAL <- VEZDIV(A,B);
  se ( VAL = -1 ) então
    imprima("Impossivel calcular, divisao por zero!");
  senão
    imprima(A," eh divisivel por ",B," - ",RET," vez(es)");
  fim-se
fim
```

```
program L6P15;
var
  A, B, VAL : integer;

function VEZDIV(X,Y: integer) :integer;
var
  VEZ, RET: integer;
begin
  if ( Y = 0 ) then
    VEZDIV := -1 {divisao por zero}
  else
  begin
    VEZ := 0;
    RET := X mod Y;
    while ( RET = 0 ) do
    begin
      VEZ := VEZ + 1;
      X := X div Y;
      RET := X mod Y;
    end;
    VEZDIV := VEZ;
  end;
end;

begin
  write('Entre com um valor: ');
  readLn(A);
  write('Entre com um valor: ');
  readLn(B);
  VAL := VEZDIV(X,Y);
  if ( VAL = -1 ) then
    writeLn('Impossivel calcular, divisao por zero!')
  else
    writeLn(X,' eh divisivel por ',Y,' - ',VAL,' vez(es)');
end.
```

```
function L6P15;
A = input('Digite um valor para x: ');
B = input('Digite um valor para y: ');
VAL = VEZDIV(A,B);
if ( VAL == -1 )
  disp('Impossivel calcular, divisao por zero!');
else
  fprintf(1,'%d eh divisivel por %d - %d vez(es)',A,B,VAL);
end

function VEZ = VEZDIV(X,Y);
if ( Y == 0 )
  VEZ = -1; % divisao por zero
else
  VEZ = 0;
  RET = mod(X,Y);
  while ( RET == 0 )
    VEZ = VEZ + 1;
    X = floor(X / Y);
    RET = mod(X,Y);
  end
end
```

- ***Funções retornando mais de um parâmetro***

16) Construa uma função que efetue a ***TROCA*** dos valores de ***a*** por ***b,*** recebidos como parâmetro. Ou seja, essa função deve substituir o valor de ***a*** pelo de ***b***, e reciprocamente. Crie também um algoritmo que leia dois valores quaisquer, e imprima os valores após a chamada da função ***TROCA***.

```
algoritmo L6P16;
var
  real : X, Y;

função TROCA(var real : A, B) : real;
var
  real: AUX;
início
  AUX <- A;
  A <- B;
  B <- AUX;
fim

início
  leia(X,Y);
  TROCA(X,Y);
  imprima('O valor de X passa a ser : ',X);
  imprima('O valor de Y passa a ser : ',Y);
fim
```

```
program L6P16;
var
  X, Y : real;

function TROCA(var A,B : real) : real;
var
  AUX : real;
begin
  AUX := A;
  A   := B;
  B   := AUX;
end;

begin
  write('Entre com um valor: ');
  readLn(X);
  write('Entre com um valor: ');
  readLn(Y);
  TROCA(X,Y);
  writeLn('O valor de X passa a ser: ',X);
  writeLn('O valor de Y passa a ser: ',Y);
end.
```

```
function L6P16;
X = input('Entre com um valor: ');
Y = input('Entre com um valor: ');
[X,Y] = TROCA(X,Y);
fprintf(1,'O valor de X passa a ser: %d\n',X);
fprintf(1,'O valor de Y passa a ser: %d\n',Y);

function [A,B] = TROCA(A,B);
AUX = A;
A = B;
A = AUX;
```

17) Construa uma função que receba três valores, ***a***, ***b*** e ***c***, retorne (passagem por referência) o ***MAIOR*** e o ***MENOR*** valor desses três. Deve ser criado um algoritmo para utilizar tal função lendo os três valores e imprimindo o maior e o menor valor computado.

```
algoritmo L6P17;
var
  inteiro : C;
  real : N[1..3], MA, ME;

função MAIORMENOR(real : A, B, C; var real : MAIOR, MENOR ) : inteiro;
início
  se ( A > B ) e ( A > C ) então
    MAIOR <- A;
  fim-se
  se ( B > A ) e ( B > C ) então
    MAIOR <- B;
  fim-se
  se ( C > A ) e ( C > B ) então
    MAIOR <- C;
  fim-se
  se ( A < B ) e ( A < C ) então
    MENOR <- A;
  fim-se
  se ( B < A ) e ( B < C ) então
    MENOR <- B;
  fim-se
  se ( C < A ) e ( C < B ) então
    MENOR <- C;
  fim-se
fim

início
  para C de 1 até 3 faça
    leia(N[C]);
  fim-para
  MAIORMENOR(N[1],N[2],N[3],MA,ME);
  imprima("Maior: ",MA);
  imprima("Menor: ",ME);
fim
```

```
program L6P17;
var
  C: integer;
  MA, ME : real;
  N : array [1..3] of real;

function MAIORMENOR( A, B, C : real; var MAIOR, MENOR : real ) : integer;
begin
  if ( A > B ) and ( A > C ) then
    MAIOR := A;
  if ( B > A ) and ( B > C ) then
    MAIOR := B;
  if ( C > B ) and ( C > A ) then
    MAIOR := C;
  if ( A < B ) and ( A < C ) then
    MENOR := A;
  if ( B < A ) and ( B < C ) then
    MENOR := B;
  if ( C < B ) and ( C < A ) then
    MENOR := C;
end;

begin
  for C := 1 to 3 do
  begin
    write('Informe um Numero: ');
    readLn(N[C]);
  end;
  MAIORMENOR(N[1],N[2],N[3],MA,ME);
  writeLn('Maior: ',MA);
  writeLn('Menor: ',ME);
end.
```

```
function L6P17;
for C = 1 : 3
   N(C) = input('Informe um número: ');
end
[MA,ME] = MAIORMENOR(N(1),N(2),N(3));
fprintf(1,'Maior: %f\n',MA);
fprintf(1,'Menor: %f\n',ME);

function [MAIOR,MENOR] = MAIORMENOR(A,B,C);
if ( A > B ) & ( A > C )
  MAIOR = A;
end
if ( B > A ) & ( B > C )
  MAIOR = B;
end
if ( C > B ) & ( C > A )
  MAIOR = C;
end
if ( A < B ) & ( A < C )
  MENOR = A;
end
if ( B < A ) & ( B < C )
  MENOR = B;
end
if ( C < B ) & ( C < A )
  MENOR = C;
end
```

18) Construa uma função que receba dois valores inteiros ***a*** e ***b***, retorne (passagem por referência) o quociente, **div**, e o resto divisão, **mod**, de ***a*** por ***b***. A função deve retornar -1 caso não seja possível realizar as operações e 0 caso seja possível. Um algoritmo para utilizar tal função deve ser criado, tratando o retorno da função.

```
algoritmo L6P18;
var
  inteiro : X, Y, QUOC, REST, VAL;

função DIVMOD(inteiro : A, B; var inteiro: DV, MD ) : integer;
início
  se ( B = 0 ) então
    DIVMOD <- -1
  senão
    DIVMOD <- 0;
    DV <- A div B;
    MD <- A mod B;
  fim-se
fim

início
  leia(X,Y);
  VAL <- DIVMOD(X,Y,QUOC,REST)
  se ( VAL = -1 ) então
    imprima("Impossivel realizar o calculo (divisao por zero)!");
  senão
    imprima(QUOC,REST);
  fim-se
fim
```

```
program L6P18;
var
  X, Y, QUOC, REST , VAL : integer;

function DIVMOD(A, B: integer; var DV, MD : integer) : integer;
begin
  if ( B = 0 ) then
    DIVMOD := -1
  else
  begin
    DIVMOD := 0;
    DV := A div B;
    MD := A mod B;
  end;
end;

begin
  write('Entre com um valor: ');
  readLn(X);
  write('Entre com um valor: ');
  readLn(Y);
  VAL := DIVMOD(X,Y,QUOC,REST);
  if ( VAL = -1 ) then
    writeLn('Impossivel realizar o calculo (divisao por zero)!')
  else
    writeLn('Quociente: ',QUOC,' e Resto da Divisao: ',REST);
end.
```

```
function L6P18;
X = input('Entre com um valor: ');
Y = input('Entre com um valor: ');
[VAL,QUOC,REST] = DIVMOD(X,Y);
if ( VAL == -1 )
  fprintf(1,'Impossivel realizar o calculo (divisao por zero)!');
else
  fprintf(1,'Quociente: %d e Resto da Divisao: %d\n',QUOC,REST);
end

function [RET,DV,MD] = DIVMOD(A,B);
if ( B == 0 )
  RET = -1;
  DV = 0;
  MD = 0;
else
  RET = 0;
  DV = floor(A/B);
  MD = mod(A,B );
end
```

19) Construa uma função que receba cinco valores e determine (retorne por passagem pro referência) o 2º e o 4º maior valores dentre eles. Construa também um algoritmo para ler tais valores, e imprimir o resultado obtido com a chamada da função.

```
algoritmo L6P19;
var
  inteiro : C;
  real : V[1..5], MM2, MM4;

função MAIPAR(real : A, B, C, D, E; var real : M2, M4): inteiro;
var
  real : V[1..5] , AUX;
  inteiro : I, J;
início
  V[1] <- A; V[2] <- B; V[3] <- C;V[4] <- D; V[5] <- E;
  para I de 1 até 4 faça
    para J de 1 até ( 5 - I ) faça
      se ( V[J] > V[J+1] ) então
        AUX    <- V[J];
        V[J]   <- V[J+1];
        V[J+1] <- AUX;
      fim-se
    fim-para
  fim-para
  M2 <- V[4];
  M4 <- V[2];
fim

início
  para C de 1 até 5 faça
    leia(V[C]);
  fim-para
  MAIPAR(V[1],V[2],V[3],V[4],V[5],MM2,MM4);
  imprima("Segundo maior ",MM2);
  imprima("Quarto maior " ,MM4);
fim
```

```
program L6p19;
var
  C : integer;
  MM2,MM4 : real;
  V : array[1..5] of real;

function MAIPAR( A, B, C, D, E : real; var M2, M4 : real) : integer;
var
  AUX : real;
  I , J : integer;
begin
  V[1] := A; V[2] := B; V[3] := C; V[4] := D; V[5] := E;
  for I := 1 to 4 do
    for J := 1 to (5-I) do
      if ( V[J] > V[J+1] ) then
      begin
        AUX    := V[J];
        V[J]   := V[J+1];
        V[J+1] := AUX;
      end;
  M2 := V[4];
  M4 := V[2];
end;

begin
  for C := 1 to 5 do
  begin
    write('Entre com um numero: ');
    readLn(V[C]);
  end;
  MAIPAR(V[1],V[2],V[3],V[4],V[5],MM2,MM4);
  writeLn('Segundo maior ',MM2);
  writeLn('Quarto maior ' ,MM4);
end.


function L6P19;
for C = 1 : 5
  V(C) = input('Entre com um numero: ');
end
[MM2,MM4] = VAL(V(1),V(2),V(3),V(4),V(5));
fprintf(1,'Segundo maior: %f\n',MM2);
fprintf(1,'Quarto maior: %f\n' ,MM4);

function [M2,M4] = VAL(A,B,C,D,E);
V(1) = A;  V(2) = B;  V(3) = C;  V(4) = D;  V(5) = E;
for I = 1 : 4
  for J = 1 : (5-I)
    if ( V(J) > V(J+1) )
      AUX    = V(J);
      V(J)   = V(J+1);
      V(J+1) = AUX;
    end
  end
end
M2 = V(4);
M4 = V(2);
```

20) Construa uma função, que receba três coeficientes relativos à uma equação de segundo grau ($a.x^2 + b.x + c = 0$) e calcule suas raízes através da fórmula de báscara:

$$x = \frac{-b \pm \sqrt{\Delta}}{2a} \qquad \Delta = b^2 - 4ac$$

A função deve levar em conta a possibilidade da existência de nenhuma, uma ou duas raízes. A função deve retornar o número de raízes ou -1 em caso de inconsistência. Os valores das raízes devem ser retornados. Construa também um algoritmo para utilizar a função construída.

```
algoritmo L6P20;
var
  real: A0, A1, A2, X1L, X2L;
  inteiro: VAL;

função EQ2(real: A, B, C; var real: X1, X2) :inteiro;
var
  real: DELTA;
início
  se ( A = 0 ) então
    EQ2 <- -1;
  senão
    DELTA <- B * B - 4 * A * C;
    se ( DELTA < 0 ) então
      EQ2 <- 0;
    senão
      se ( DELTA = 0 ) então
        EQ2 <- 1;
        X1 <- -B / ( 2 * A );
        X2 <- X1;
      senão { DELTA > 0 }
        EQ2 <- 2;
        X1 <- ( -B + raiz(DELTA) ) / ( 2 * A );
        X2 <- ( -B - raiz(DELTA) ) / ( 2 * A );
      fim-se
    fim-se
  fim-se
fim

início
  leia(A0,A1,A2);
  VAL <- EQ2(A0,A1,A2,X1L,X2L);
  se ( VAL = -1 ) então
    imprima("Incosistencia, possivel a0 = 0!");
  senão
    se ( VAL = 0 ) então
      imprima("Nao existe raiz real para tal equacao!");
    senão
      se ( VAL = 1 ) então
        imprima("Uma unica raiz real, x1 = x2 = ",X1L);
      senão
        se ( VAL = 2 ) então
          imprima("Duas raizes reais diferentes");
          imprima("x1 = ",X1L);
          imprima("x2 = ",X2L);
        fim-se
      fim-se
    fim-se
  fim-se
fim
```

```
program L6P20;
var
  A0, A1, A2, X1L, X2L : real;
  VAL : integer;

function EQ2(A, B, C : real; var X1, X2 : real) :integer;
var
  DELTA : real;
begin
  if ( A = 0 ) then
    EQ2 := -1
  else
  begin
    DELTA := B * B - 4 * A * C;
    if ( DELTA < 0 ) then
      EQ2 := 0
    else if ( DELTA = 0 ) then
    begin
      EQ2 := 1;
      X1 := -B / ( 2 * A );
      X2 := X1;
    end
    else {DELTA > 0 }
    begin
      EQ2 := 2;
      X1 := ( -B + SqRt(DELTA) ) / ( 2 * A );
      X2 := ( -B - SqRt(DELTA) ) / ( 2 * A );
    end;
  end;
end;

begin
  writeLn('Equacao do 2o. grau - a0.x^2 + a1.x + a2 = 0');
  write('Digite o coeficiente a0: ');
  readLn(A0);
  write('Digite o coeficiente a1: ');
  readLn(A1);
  write('Digite o coeficiente a2: ');
  readLn(A2);
  VAL := EQ2(A0,A1,A2,X1L,X2L);
  if ( VAL = -1 ) then
    writeLn('Incosistencia, possivel a0 = 0!')
  else if ( VAL = 0 ) then
    writeLn('Nao existe raiz real para tal equacao!')
  else if ( VAL = 1 ) then
    writeLn('Uma unica raiz real, x1 = x2 = ',X1L:5:4)
  else if ( VAL = 2 ) then
  begin
    writeLn('Duas raizes reais diferentes');
    writeLn('x1 = ',X1L:5:4);
    writeLn('x2 = ',X2L:5:4);
  end;
end.
```

```
function L6P20;
disp('Equacao do 2o. grau - a0.x^2 + a1.x + a2 = 0');
AZ = input('Digite o coeficiente a0: ');
AU = input('Digite o coeficiente a1: ');
AD = input('Digite o coeficiente a2: ');
[VAL,X1L,X2L] = EQ2(AZ,AU,AD);
if ( VAL == -1 )
    disp('Incosistencia, possivel a0 = 0!');
elseif ( VAL == 0 )
    disp('Nao existe raiz real para tal equacao!');
elseif ( VAL == 1 )
    fprintf(1,'Uma unica raiz real, x1 = x2 = %f\n',X1L);
elseif ( VAL == 2 )
  disp('Duas raizes reais diferentes!');
  fprintf(1,'x1 = %f\n',X1L);
  fprintf(1,'x2 = %f\n',X2L);
end

function [EQ2,X1,X2] = EQ2(A,B,C);
if ( A == 0 )
  EQ2 = -1;
  X1 = -1;
  X2 = -1;
else
  DELTA = B * B - 4 * A * C;
  if ( DELTA < 0 )
    EQ2 = 0;
    X1 = -1;
    X2 = -1;
  elseif ( DELTA == 0 )
    EQ2 = 1;
    X1 = -B / ( 2 * A );
    X2 = X1;
  else % DELTA > 0
    EQ2 = 2;
    X1 = ( -B + sqrt(DELTA) ) / ( 2 * A );
    X2 = ( -B - sqrt(DELTA) ) / ( 2 * A );
  end
end
```

- ***Transformações***

21) Crie uma função que realize a conversão para Radianos (***rad***) a partir de Graus (***grad***), onde ***grad*** é passado como parâmetro e ***rad*** é retornado. Sabe-se que 180º (graus) está para π radianos. Crie também um algoritmo para testar tal função.

```
algoritmo L6P21;
var
  real : GRA, RAD;

função CONV(real : GRAD) : real;
constante
  PI = 3.1415;
início
  CONV <- (GRAD*PI/180);
fim

início
  leia(GRA);
  RAD <- CONV(GRA);
  imprima(GRA," graus é equivalente a ",RAD," radianos");
fim
```

```
program L6P21;
var
  GRA, RAD : real;

function CONV(GRAD : real) : real;
begin
  CONV := (GRAD * PI/180);
end;

begin
  write('Entre com uma angulo em graus: ');
  readLn(GRA);
  RAD := CONV(X);
  writeLn(GRA:2:2,' graus eh equivalente a ',RAD:2:8,' radianos');
end.
```

```
function L6P21;
GRA = input('Entre com um angulo em graus: ');
RAD = CONV(GRA);
fprintf(1,'%.2f graus eh equivalente a %.8f radianos\n',GRA,RAD);

function RAD = CONV(GRAD);
RAD = GRAD*pi/180;
```

22) Crie uma função que realize a conversão de Fahrenheit (***F***) para graus Celsius (***C***), onde ***F*** é passado como parâmetro e ***C*** é retornado. Sabe-se que os pontos de fusão e ebulição nas escalas Celsius e Fahrenheit são: 0ºC e 100ºC, e 32ºF e 212ºF, respectivamente. Crie também um algoritmo para testar tal função.

```
algoritmo L6P22;
var
  real: C, F;
  inteiro :  RET;

função CONVFC(real : F, var real : C) : real;
início
  se (F < 459.4) então { -273oC = 0K}
    CONVFC <- 0;
  senão
    CONVFC <- 1;
    TEMP <- ( ( F - 32 ) * 9) / 5;
  fim-se
fim

início
  leia(F);
  RET <- TEMP(F,C);
  se (RET = 1) então
    imprima("A temperatura de ",F,"° é, em °C: ",C);
  senão
    imprima("Impossivel calcular, fisicamente temperatura não existe! ");
  fim-se
fim
```

```
program L6P22;
var
  F, C : real;
  RET : integer;

function CONVFC ( F : real; var C : real) : integer;
begin
  if ( F < 459.4 ) then
    CONVFC := 0
  else
  begin
    CONVFC := 1;
    TEMP := ( ( F - 32 ) * 5) / 9;
  end;
end;

begin
  write('Entre com uma temperatura na escala Fahrenheit: ');
  readLn(F);
  RET := TEMP(F,C);
  if ( RET = 1 ) then
    writeLn('A temperatura de ',F,' graus F em graus Celsius: ',C:5:2)
  else
    writeLn('Impossivel calcular, fisicamente temperatura não existe!');
end.
```

```
function L6P22;
F = input('Entre com uma temperatura em Fahrenheit: ');
[RET,C] = TEMP(F);
if (RET == 1)
  fprintf(1,'A temperatura de %d °F em °C é: %d\n',F,C);
else
  fprintf(1,' Impossivel calcular, fisicamente temperatura não existe!');
end

function [RET,C] = CONVFC(F);
if ( F < 459.4)
  RET = 0;
else
  RET = 1;
  C = ( ( F - 32 ) * 5 ) / 9;
end
```

23) Crie uma função que realize a conversão de Polegadas (***pol***) para Centímetros (***cm***), onde ***pol*** é passado como parâmetro e ***cm*** é retornado. Sabe-se que 1 polegada está para 2,54 centímetros. Crie também um algoritmo para testar tal função.

```
algoritmo L6P23;
var
  real : CM, POL;

função CONV( real : POLEG ) : real;
início
  CONV <- POLEG * 2.54;
fim

início
  leia(POL);
  CM <- CONV(POL);
  imprima(POL," polegadas corresponde a ",CM," centimetros");
fim
```

```
program L6P23;
var
  POL, CM : real;

function CONV( POLEG : real ) : real;
begin
  CONV := CM / 2.54;
end;

begin
  write('Entre com um de polegadas: ');
  readLn(POL);
  CM := CONV(POL);
  writeLn(POL:3:2,' polegadas corresponde a ',CM:3:2,' centimetros');
end.
```

```
function L6P23;
POL = input('Entre com uma medida em polegadas: ');
CM = CONV(POL);
fprintf(1,'%.2f polegadas corresponde a %.2f centimetros\n',POL,CM);

function RET = CONV(POLEG);
RET = POLEG / 2.54;
```

24) Crie uma função que realize a conversão de pés (***feet***) para metros (***m***), onde ***feet*** é passado como parâmetro e ***m*** é retornado. Sabe-se que 1 metro está para 3,281 pés. Crie também um algoritmo para testar tal função.

```
algoritmo L6P24;
var
  real : M, F;

função METROS(real: F) : real;
início
  METROS <- F / 3.281;
fim

início
  leia(F);
  M <- METROS(F);
  imprima(F," pés = ",M," metros");
fim
```

```
program L6p24;
var
  M, F : real;

function METROS( F : real ) : real;
begin
  METROS := F / 3.281;
end;

begin
  write('Entre com uma medida em pes: ');
  readLn(F);
  M := METROS(F);
  writeLn(F:5:3,' pes = ',M:5:3,' metros');
end.
```

```
function L6p24;
F = input('Entre com a medida em pés: ');
M = METROS(F);
fprintf(1,'%f pés = %f\n metros',F,M);

function RET = METROS(F);
RET = F/3.281;
```

25) Crie uma função que realize a conversão da escala Kelvin (***K*** - escala absoluta) para a escala Fahrenheit (***F***). Sabe-se que 273K equivale a 32ºF e a cada variação de 10 unidades na escala Kelvin equivale a 18 na escala Fahrenheit. A função deve retornar zero caso não seja possível realizar a conversão e um em caso contrário. Crie também um algoritmo para testar tal função.

```
algoritmo L6P25;
var
  real : FL, KL;

função CONVKF(real: K; var real : F) : inteiro;
início
  { 273K - 32F, 373K - 212F}
  se ( K < 0 ) então
    CONVKF <- 0;
  senão
    CONVKF <- 1;
    F <- ( 5 * 212 - 9 * (373-K) ) / 5;
  fim-se
fim

início
  leia(KL);
  se ( CONVKF(KL,FL) = 0 ) então
    imprima("Impossivel calcular, temperatura Kelvin negativa!");
  senão
    imprima("A correspondente temperatura na escala Fahrenheit eh ",FL);
  fim-se
fim
```

```
program L6P25;
var
  FL, KL : real;

function CONVKF(K : real; var F : real) : integer;
begin
  if ( K < 0 ) then
    CONVKF := 0
  else
  begin
    { 273K - 32F, 373K - 212F}
    CONVKF := 1;
    F := ( 5 * 212 - 9 * (373-K) ) / 5;
  end;
end;

begin
  write('Entre com a temperatura na escala Kelvin: ');
  readLn(KL);
  if ( CONVKF(KL,FL) = 0 ) then
    writeLn('Impossivel calcular, temperatura Kelvin negativa!')
  else
    writeLn('A correspondente temperatura na escala Fahrenheit eh ',FL:3:2);
end.
```

```
function L6P25;
KL = input('Entre com a temperatura na escala Kelvin: ');
[VAL,FL] = CONVKF(KL);
if ( VAL == 0 )
  fprintf(1,'Impossivel calcular, temperatura Kelvin negativa!\n');
else
  fprintf(1,'A correspondente temperatura na escala Fahrenheit eh %.2f\n',FL);
end

function [RET,F] = CONVKF(K);
% 273K - 32F, 373K - 212F
if ( K < 0 )
  RET = 0;
  F = 0;
else
  RET = 1;
  F = ( 5 * 212 - 9 * (373-K) ) / 5;
end
```

- ***Funções recursivas***

26) Seja a série de Fibonacci:

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

que pode ser definida recursivamente por:

$$Fib(n) = \begin{cases} 1 & \text{se} \quad n = 1 \vee n = 2 \\ Fib(n-1) + Fib(n-2) & \text{se} \quad n > 2 \end{cases}$$

Então escreva:

- Uma função recursiva que gere o termo de ordem n da série de Fibonacci.
- Um algoritmo que, utilizando a função definida acima gere a série de Fibonacci até o termo de ordem 20.

```
algoritmo L6P26;
var
  inteiro : C;

função FIB(inteiro : N) : inteiro;
var
  inteiro : F;
início
  se ( ( N = 1 ) ou ( N = 2 ) ) então
    F <- 1;
  senão-se ( N > 2 ) então
    F <- FIB(N-2) + FIB(N-1);
  fim-se
  FIB(N) <- F;
fim

início
  para C de 1 até 20 faça
    imprima(FIB(C));
  fim-para
fim
```

```
program L6P26;
var
  C : integer;

function FIB ( N : integer) : integer;
var
  F : integer;
begin
  if ( ( N = 1 ) or ( N = 2 ) ) then
    F := 1
  else if ( N > 2 ) then
    F := FIB(N-2) + FIB(N-1);
  FIB := F;
end;

begin
  for C := 1 to 20 do
   write(FIB(C),' ');
  writeLn('');
end.
```

```
function L6P26;
for C = 1 : 20
 fprintf(1,'%d ',FIB(C));
end
fprintf(1,'\n');

function F = FIB(N);
if ( ( N == 1 ) | ( N == 2 ) )
  F = 1;
elseif ( N > 2 )
  F = FIB(N-2) + FIB(N-1);
end
```

27) Pode-se calcular o quociente da divisão, ***DIV***, de ***x*** por ***y***, dois números inteiros, usando-se a seguinte definição:

$$DIV(x,y) = \begin{cases} 1 + DIV(|x| - |y|, |y|), & \text{se } |x| > |y| \\ 0 & \text{se } |x| < |y| \\ 1 & \text{se } |x| = |y| \end{cases}$$

Então, pede-se que seja criada uma função recursiva para descrever tal definição. A função deve retornar -1 caso não seja possível realizar o cálculo. Além disso, crie um algoritmo que leia os dois valores inteiros e utilize a função criada para calcular o quociente de ***x*** por ***y***, e imprima o valor computado.

```
algoritmo L6P27;
var
  inteiro : A , B , VAL;

função DVS(inteiro: X , Y) : inteiro;
início
  se ( Y = 0 ) então
    DVS <- -1;
  senão-se ( abs(X) > abs(Y) ) então
    DVS <- 1 + DVS(abs(X)-abs(Y),abs(Y));
  senão-se ( abs(X) < abs(Y) ) então
    DVS <- 0;
  senão-se ( abs(X) = abs(Y) ) então
    DVS <- 1;
  fim-se
fim

início
  leia(A,B);
  VAL <- DVS(A,B);
  se ( VAL = -1 ) então
    imprima("Impossivel realizar o calculo! (divisao por zero)");
  senão
    imprima("O quociente da divisao de ",A," por ",B," eh ",VAL);
  fim-se
fim
```

```
program L6P27;
var
  A , B , VAL: integer;

function DVS(X , Y : integer) : integer;
begin
  if ( Y = 0 ) then
    DVS := -1
  else if ( abs(X) > abs(Y) ) then
    DVS := 1 + DVS(abs(X)-abs(Y),abs(Y))
  else if ( abs(X) < abs(Y) ) then
    DVS := 0
  else if ( abs(X) = abs(Y) ) then
    DVS := 1;
end;

begin
  write('Entre com um valor: ');
  readLn(A);
  write('Entre com um valor: ');
  readLn(B);
  VAL := DVS(A,B);
  if ( VAL = -1 ) then
    writeLn('Impossivel realizar o calculo! (divisao por zero)')
  else
    writeLn('O quociente da divisao de ',A,' por ',B,' eh ',VAL);
end.
```

```
function L6P27;
A = input('Informe um valor: ');
B = input('Informe um valor: ');
VAL = DVS(A,B);
if ( VAL == -1 )
  disp('Impossivel realizar calculo! (divisao por zero)');
else
  fprintf(1,'O quociente da divisao de %d por %d eh %d\n',A,B,VAL);
end

function RET = DVS(X,Y);
if Y == 0
  RET = -1;
elseif ( abs(X) > abs(Y) )
  RET = 1 + DVS(abs(X)-abs(Y),abs(Y));
elseif ( abs(X) < abs(Y) )
  RET = 0;
elseif ( abs(X) == abs(Y) )
  RET = 1;
end
```

28) Pode-se calcular o resto da divisão, ***MOD***, de ***x*** por ***y***, dois números inteiros, usando-se a seguinte definição:

$$MOD(x,y) = \begin{cases} MOD(|x|-|y|,|y|), & \text{se } |x| > |y| \\ |x| & \text{se } |x| < |y| \\ 0 & \text{se } |x| = |y| \end{cases}$$

Então, pede-se que seja criada uma função recursiva para descrever tal definição. A função deve retornar -1 caso não seja possível realizar o cálculo. Além disso, crie um algoritmo que leia os dois valores inteiros e utilize a função criada para calcular o resto da divisão de ***x*** por ***y***, e imprima o valor computado.

```
algoritmo L6P28;
var
  inteiro: A , B , VAL;

função MDS(inteiro: X , Y) : inteiro;
início
  se ( Y = 0 ) então
    MDS <- -1;
  senão-se ( abs(X) > abs(Y) ) então
    MDS <- MDS(abs(X)-abs(Y),abs(Y));
  senão-se ( abs(X) < abs(Y) ) então
    MDS <- abs(X);
  senão-se ( abs(X) = abs(Y) ) então
    MDS <- 0;
  fim-se
fim

início
  leia(A,B);
  VAL <- MDS(A,B);
  se ( VAL = -1 ) então
    imprima("Impossivel realizar o calculo! (divisao por zero)");
  senão
    imprima("O resto da divisao de ",A," por ",B," eh ",VAL);
  fim-se
fim
```

```
program L6P28;
var
  A , B , VAL: integer;

function MDS(X , Y : integer) : integer;
begin
  if ( Y = 0 ) then
    MDS := -1
  else
    if ( abs(X) > abs(Y) ) then
      MDS := MDS(abs(X)-abs(Y),abs(Y))
    else if ( abs(X) < abs(Y) ) then
      MDS := abs(X)
    else if ( abs(X) = abs(Y) ) then
      MDS := 0;
end;

begin
  write('Informe um valor: ');
  readLn(A);
  write('Informe um valor: ');
  readLn(B);
  VAL := MDS(A,B);
  if ( VAL = -1 ) then
    writeLn('Impossivel realizar o calculo! (divisao por zero)')
  else
    writeLn('O resto da divisao de ',A,' por ',B,' eh ',VAL);
end.
```

```
function L6P28;
A = input('Informe um valor: ');
B = input('Informe um valor: ');
VAL = MDS(A,B);
if ( VAL == -1 )
  disp('Impossivel realizar calculo! (divisao por zero)');
else
  fprintf(1,'O resto da divisao de %d por %d eh %d\n',A,B,VAL);
end

function RET = MDS(X,Y);
if Y == 0
  RET = -1;
elseif ( abs(X) > abs(Y) )
  RET = MDS(abs(X)-abs(Y),abs(Y));
elseif ( abs(X) < abs(Y) )
  RET = abs(X);
elseif ( abs(X) == abs(Y) )
  RET = 0;
end
```

29) O máximo divisor comum (***MDC***) de dois números inteiros ***x*** e ***y*** pode ser calculado usando-se uma definição recursiva:

$$MDC(x, y) = MDC(x - y, y), \text{ se } x > y .$$

Além disso, sabe-se que:

$$MDC(x, y) = MDC(y, x)$$

$$MDC(x, x) = x$$

Exemplo:

$$MDC(10,6) = MDC(4,6) = MDC(6,4) = MDC(2,4) = MDC(4,2) = MDC(2,2) = 2$$

Então, pede-se que seja criada uma função recursiva para descrever tal definição. Crie, também, um algoritmo que leia os dois valores inteiros e utilize a função criada para calcular o ***MDC*** de ***x*** e ***y***, e imprima o valor computado.

```
algoritmo L6P29;
var
  inteiro : A, B, VAL;

função MDC(inteiro : X, Y) : inteiro;
início
  se ( X = 0 ) ou ( Y = 0 ) então
    MDC <- 0;
  senão-se X > Y então
    MDC <- MDC(X-Y,Y);
  senão-se ( X < Y ) então { MDC(x,y) = MDC(y,x) }
    MDC <- MDC(Y-X,X);
  senão
    MDC <- X;
  fim-se
fim

início
  leia(A,B);
  VAL <- MDC(A,B);
  imprima("MDC(",A,",",B,") = ",VAL);
fim
```

```
program L6p29;
var
  A, B, VAL : integer;

function MDC(X,Y: integer): integer;
begin
  if ( X = 0 ) or ( Y = 0 ) then
    MDC := 0
  else if ( X > Y ) then
    MDC := MDC(X-Y,Y)
  else if ( X < Y ) then { mdc(x,y) = mdc(y,x) }
    MDC := MDC(Y-X,X)
  else { X = Y}
    MDC := X;
end;

begin
  write('A: ');
  readLn(A);
  write('B: ');
  readLn(B);
  VAL := MDC(A,B);
  writeLn('MDC(',A,',',B,') = ',VAL);
end.
```

```
function L6P29;
A = input('Entre com um valor: ');
B = input('Entre com um valor: ');
VAL = MDC(A,B);
fprintf(1,'MDC(%d,%d) = %d\n',A,B,VAL);

function RET = MDC(X,Y);
if ( X == 0 ) | ( Y == 0 )
  RET = 0;
elseif ( X > Y )
  RET = MDC(X-Y,Y);
elseif ( X < Y )
  RET = MDC(Y-X,X);
else
  RET = X;
end
```

30) O fatorial de um número ***n***, inteiro e positivo, pode ser definido recursivamente, ou seja:

$$n! = \begin{cases} 1 & se \quad n = 0 \\ n.(n-1)! & se \quad n \geq 1 \end{cases}$$

Então, pede-se que seja criada uma função recursiva que calcule o fatorial de um número n. A função deve retornar -1 caso não seja possível calcular o fatorial Além disso, crie um algoritmo que leia um valor, utilize a função criada para calcular o fatorial e imprima o valor computado.

```
algoritmo L6P30;
var
  inteiro : I, VAL;

função FAT(inteiro : N) : inteiro;
início
  se ( N < 0 ) então
    FAT <- -1;
  senão-se (N = 0) então
    FAT <- 1;
  senão
    FAT <- N * FAT(N-1);
  fim-se
fim

início
  leia(I);
  VAL <- FAT(I);
  se ( VAL = -1 ) então
    imprima("Impossivel calcular o fatorial de ",I);
  senão
    imprima("O Fatorial de ",I," eh ",VAL);
  fim-se
fim
```

```
program L6P30;
var
  I : integer;
  VAL : real;

function FAT(N : integer) : real;
begin
  if ( N < 0 ) then
    FAT := -1
  else if (N = 0) then
    FAT := 1
  else
    FAT := N * FAT(N-1);
end;

begin
  write('Entre com um numero: ');
  readLn(I);
  VAL := FAT(I);
  if ( VAL = -1 ) then
    writeLn('Impossivel calcular o fatorial de ',I)
  else
    writeLn('O Fatorial de ',I,' eh ',VAL:1:0);
end.
```

```
function L6P30;
I = input('Digite um numero: ');
VAL = FAT(I);
if ( VAL == -1 )
  fprintf(1,'Impossivel calcular o fatorial de %d\n',I);
else
  fprintf(1,'O Fatorial de %d eh %d\n',I,VAL);
end

function F = FAT(N);
if ( N < 0 )
  F = -1;
elseif (N == 0)
  F = 1;
else
  F = N * FAT(N-1);
end
```