9) PROCEDIMENTOS e FUNÇÕES:

Procedimentos e Funções são "módulos" (ié, *subprogramas* ou *subrotinas*) que fazem parte de um programa, e que são executados quando "chamados" (ié, quando ativados).

Uma vantagem do seu uso é quando uma sequência de comandos deve ser executada repetidamente
(possivelmente com diferentes "argumentos", ié, com diferentes *parâmetros*).

Os procedimentos (e as funções) podem ser chamadas pelo "programa principal", por um outro procedimento (ou função) ou pelo próprio procedimento (ou função)!

9.1) Procedimentos:

Procedimentos têm o mesmo "formato" de um programa, ou seja, têm um cabeçalho, uma parte de declaração e uma parte executável (ié, uma sequência de comandos delimitadas por *begin* e *end*).

Como declarar:

procedure <identificador>**(**<lista de parâmetros>**)**;
  <parte de declarações>
begin
  <lista de comandos>
end;

Observações:

(1) Essa declaração deve ser feita em meio às outras declarações (de variáveis, de tipos, etc) do "programa principal" (ou de "outro lugar" onde esteja definida).

(2) Tudo o que for declarado num procedimento é considerado um *objeto local* (enquanto que as declarações no programa principal correspondem a *objetos globais*).

(3) Os parâmetros desta lista são chamados *parâmetros formais* (ou de definição), e sua especificação é composta de nome, tipo e modo de passagem (ou de substituição).

Há dois **modos de passagem** de parâmetros:
a) **por valor**
b) **por variável** (ou "por referência", ou "por nome"): neste caso, deve-se colocar a palavra reservada ***var*** em frente ao nome do parâmetro.

Como ativar:

<identificador>(<lista de parâmetros>)

Observações:

(1) A ativação de um procedimento corresponde a um *comando de chamada*, colocado na parte executável do programa principal (de outro procedimento ou do próprio procedimento).

(2) Os parâmetros desta lista são chamados *parâmetros de chamada* (ou reais, ié, "actual parameters"), e nela são colocados apenas os seus nomes.

(3) Depois de ativado, ié, depois da execução dos comandos do procedimento, a execução do programa volta ao comando seguinte (a este comando de chamada).

Exemplos:

(1)
```
procedure TROCA(var A,B: real);
  var AUX: real;
begin
  AUX:=A;
  A:=B;
  B:=AUX
end;
```

==========

(2)
```
program MANIPULA_VETOR_E_PROC;
  type  VETOR=array[1..20] of real;
  var V: VETOR;
      N, I: integer;
```

```
procedure LE(var A:VETOR; TAM:integer);
    var I: integer;
begin
    for I:=1 to TAM do
        begin
            write(I,‘ºelemento:’);
            readln(A[I])
        end;
end;

procedure IMPR(A:VETOR; TAM:integer);
    var I: integer;
begin
    for I:=1 to TAM do write(A[I],‘ ’)
end;

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

procedure INV(var A:VETOR; TAM:integer);
    var I: integer;
begin
    for I:=1 to TAM div 2 do
        TROCA(A[I],A[TAM-I+1])
end;

begin (*Programa Principal*)
    repeat
        write(‘N= ’);
        readln(N)
    until (N>0) and (N<=20);
    LE(V,N);
    IMPR(V,N);
    INV(V,N);
    IMPR(V,N);
end.
```

==========

(3)
```
procedure LINHA(CARAC:char);
   var I: integer;
begin
    for I:=1 to 60 do write(CARAC);
    writeln
end;
```

==========

(4)
```
procedure MENU(var OPCAO: char);
begin
   writeln('OPÇÕES:');
   writeln('1 -  matriz transposta');
   writeln('2 -  imprime diagonais');
   writeln('3 -  soma outra matriz');
   writeln('4 -  produto por escalar');
   writeln('5 -  fim');
   write('digite opção: ');
   readln(OPCAO);
end;
```

Observações:

(1) Deve haver uma correspondência 1-1 entre os parâmetros *de chamada* e os *formais*, que devem ser do mesmo tipo.

(2) Na **passagem por valor**:

a) o parâmetro de chamada é uma **expressão**.

b) o parâmetro formal é como se fosse uma variável local, cujo valor inicial é o valor corrente do parâmetro de chamada (ié, da expressão correspondente).

c) O procedimento pode modificar esse valor do parâmetro formal (recebido do parâmetro de chamada), não afetando o valor do parâmetro de chamada correspondente (no caso em que aquela expressão seja uma variável(!), no programa principal ou em outro lugar de onde foi feita a chamada).

d) O parâmetro formal não pode representar o resultado de uma computação feita no procedimento.

(3) Na **passagem por variável** (ié, quando o parâmetro formal é

precedido pela palavra ***var***), o parâmetro de chamada correspondente deve ser uma **variável**, que será representada pelo parâmetro formal durante toda a execução do procedimento.
Neste caso, qualquer alteração feita no procedimento formal provoca idêntica alteração no parâmetro de chamada correspondente.

(4) Sempre que um parâmetro representar um "resultado" do procedimento, ele deverá ser definido como um parâmetro com **passagem por variável**.

(5) Recomenda-se que **passagem por valor** seja utilizada sempre que o parâmetro servir apenas como "entrada" (**Obs:** há exceções; ver mais adiante!).

(6) As variáveis declaradas nos procedimentos são ditas **variáveis locais**, e só podem ser usadas naqueles procedimentos em que estão definidas. Variáveis declaradas no programa principal são ditas **variáveis globais**, e podem se usadas dentro dos procedimentos.

(7) A lista de parâmetros de um procedimento pode não existir!
Neste caso as referências serão apenas a objetos globais.

Exemplo:

```
program PARM;
    var A,B: integer;
    procedure ESCREVE(X:integer;var Y:integer);
    begin
      X:=X+1; Y:=Y+1; writeln(X,Y)
    end;
begin (*PP*)
    A:=0; B:=0;
    ESCREVE(A,B);
    writeln(A,B)
end.
```

9.2) Funções:

Funções são procedimentos que resultam (ié, retornam) necessariamente pelo menos um valor (do tipo simples ou *pointer*), e são ativadas no meio

de uma expressão.

Como declarar:

function <identificador>(<lista de parâmetros>)**:** <tipo do resultado>;
    <parte de declarações>
begin
    <lista de comandos>
end;

Obs: Na "parte executável" da declaração da função deve haver pelo menos um comando que atribua um valor ao nome da função.

Como ativar:

Uma função é ativada quando seu nome (acompanhado dos parâmetros de chamada correspondentes) é usado numa expressão no programa principal (ou em outro procedimento/função, ou na própria função).

Observações:

(1) Tudo o que foi mencionado para parâmetros de procedimentos também é válido para funções.

(2) Na declaração da função, o aparecimento do seu nome dentro da parte executável (no meio de uma expressão!) implica em uma nova chamada (dita **chamada recursiva**).

(3) Uma função pode se substituída por um procedimento em que o resultado seja passado por um parâmetro com **passagem por variável.**

Exemplos:

(1)
```
program EX_FUNC_01;
    var N: integer;

    function FAT(NUM:integer):integer;
        var I, F: integer;
    begin
        F:=1;
        for I:=1 to NUM do
            F:=F*I;
        FAT:=F
```

```
        end;

    begin (*PP*)
        repeat
            write('N= '); readln(N);
            writeln('.....fatorial= ', FAT(N))
        until KEYPRESSED
    end.
```

==========

(2)
```
function SOMA(A:VETOR;N:integer): real;
    var I: integer;
    S: real;
begin
    S:=0;
    for I:=1 to N do
        S:=S+A[I];
    SOMA:=S
end;
```

==========

(3)
```
program EX_FUNC_04:
    var NUM: integer;

    function PRIMO(N:integer):boolean;
        var I: integer;
    begin
        I:=2;
        while (N mod I <> 0) and (I < sqrt(N))
            do I:=I+1;
        if  N mod I = 0 then PRIMO:=false
                        else  PRIMO:=true
    end;

begin (*PP*)
    repeat
```

```
        write('N = '); readln(NUM);
        if PRIMO(NUM) {=true}
              then writeln('.....é primo')
             else   writeln('.....não é primo')
    end.
```

==========

(4)
```
    function ACHOU(X:real;V:VETOR;N:integer): boolean;
        var I: integer;
    begin
        ACHOU:=false;
         for I:=1 to N do
             if V[I]=X then ACHOU:=true
    end;
```

==========

(5)
```
    program EX_FUNC_05;
        var N, P: integer;

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

        function COMBINA(N,P:integer):integer;
        begin
            COMBINA:=FAT(N)/(FAT(N-P)*FAT(P))
        end;

    begin (*PP*)
        write('N = '); readln(N);
        write('P = ');  readln(P);
        write('A combinação de ', N,'  ');
        write(P, ' a ',P, ' é ',COMBINA(N,P))
```

```
end.
```

==========

(6)
```
procedure FATORIAL(N:integer; var FAT: integer;);
   var I: integer;
begin
    FAT:=1;
    for I:=1 to N do
        FAT:=FAT*I
end;
```

Mais Observações sobre Procedimentos e Funções:

(1) Se o parâmetro é estruturado (ié, um *array*, um *record*, etc), a "cópia" (na **passagem por valor**) pode ser cara. Portanto, é preferível (excepcionalmente!) defini-lo com **passagem por variável**.

(2) *Procedures* e *Functions* também podem ser passadas como parâmetros. Neste caso, somente com **passagem por valor**.

(3) É também permitido o uso de arquivos como parâmetros. Apenas, com **passagem por variável**.

(4) Um procedimento (ou uma função) pode ser referenciado (por outro procedimento) antes mesmo da sua definição, desde que seja incluída (antes da definição de todos os procedimentos) uma cópia de seu cabeçalho seguido da palavra reservada *forward*. Veja, como exemplos, os **prog87** e **prog88**.