Capítulo VI : Subprogramas

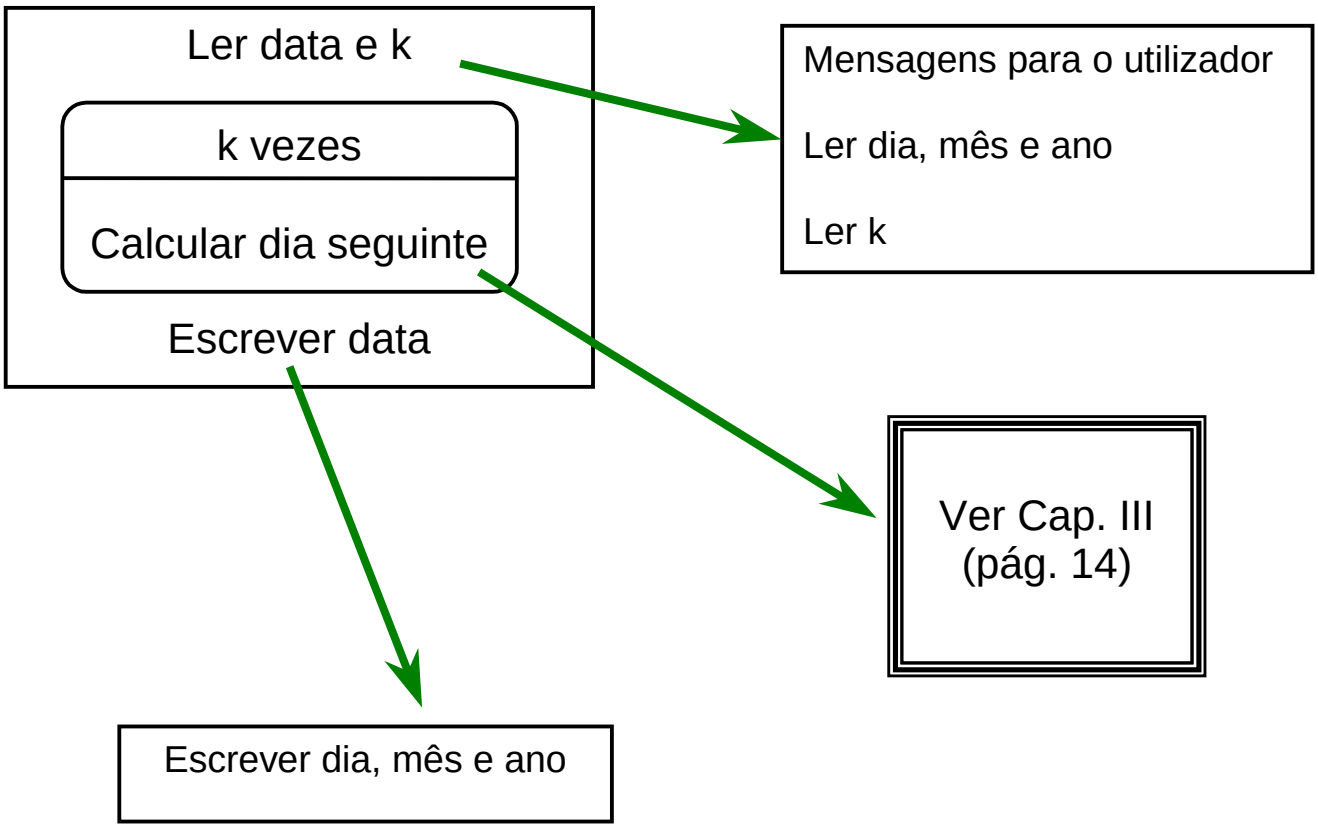
Subdivisão dos programas em unidades funcionais.

- **Procedimento** (**procedure**): **realiza uma acção**
- **Função** (**function**) : **calcula e fornece um valor**

5.1. Procedimentos

Problema: É dada uma data, pelos valores de **dia**, **mês** e **ano**, e também um valor inteiro e positivo **k**.
Qual a data de **k** dias depois da data dada?

Abordagem:



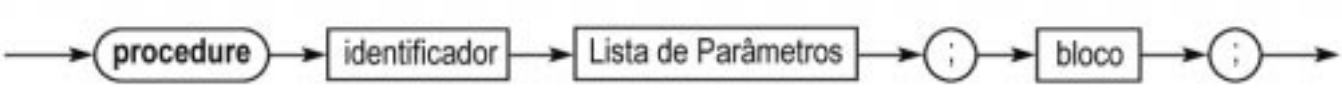
Esquema do Programa:

```
program IncrementarData0(input, output);

(* Declarações das Variáveis *)
(* Declarações dos Procedimentos *)

begin  LerDados;              (* Chamada de Procedimento *)
       for  i:= 1 to k do
            DiaSeguinte;      (* Chamada de Procedimento *)
       EscreverData           (* Chamada de Procedimento *)
end.
```

Diagramas de Sintaxe:

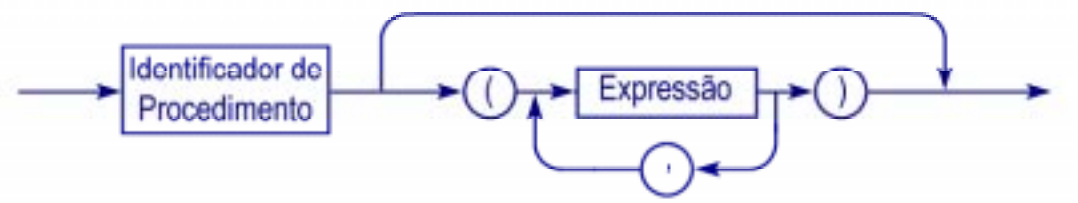
Declaração de Procedimento

(Bloco)



Chamada de Procedimento

(Instrução)



```
program IncrementarData1(input, output);
var   dia, mes, ano, ultimo, i, k : integer;
      bissexto: boolean;
(* Declarações dos Procedimentos *)
procedure  DiaSeguinte;
           begin  bissexto:= (ano mod 4 = 0) and
                             (ano mod 100 <> 0) or (ano mod 400 = 0);
                  case mes of
                        1, 3, 5, 7, 8, 10, 12 : ultimo := 31;
                        4, 6, 9, 11 : ultimo := 30;
                        2 : if  bissexto
                            then ultimo := 29
                            else  ultimo := 28
                  end;
                  if  dia = ultimo
                  then   begin   dia := 1;
                                 if mes = 12
                                 then begin   mes := 1;
                                              ano := ano+1
                                      end
                                 else  mes := mes+1
                         end
                  else  dia := dia+1
           end (* DiaSeguinte *);

procedure  LerDados;
           begin  writeln('Indique o dia, mes e ano da presente data');
                  readln(dia, mes, ano);
                  writeln('Quantos dias após a presente data?');
                  readln(k)
           end (* LerDados *);

procedure  EscreverData;
           begin  write(k, ' Dias após a presente data são: ');
                  writeln(dia:3, '/', mes:2, '/', ano:4)
           end (* EscreverData *);

begin (* Programa Principal *)
      LerDados;
      for  i:= 1 to k do
           DiaSeguinte;
      EscreverData
end (* IncrementarData1 *).
```

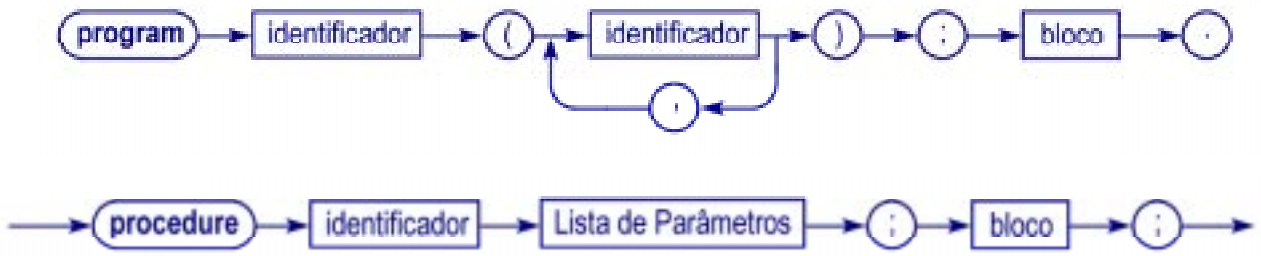
Notas gerais sobre Subprogramas:

- Cada Subprograma pode ser chamado várias vezes no mesmo Programa.
- Os Subprogramas podem (em certos casos) ser testados de forma independente.
- O Subprograma, já testado, pode ser reutilizado noutros Programas.
- Quanto mais complexo (ou mais extenso) o Programa, maior a necessidade de subdivisão em Subprogramas.
- Os Subprogramas aumentam a clareza de um Programa, tornando mais simples a sua compreensão, modificação ou extensão.

Uma correcta utilização de Subprogramas reduz:

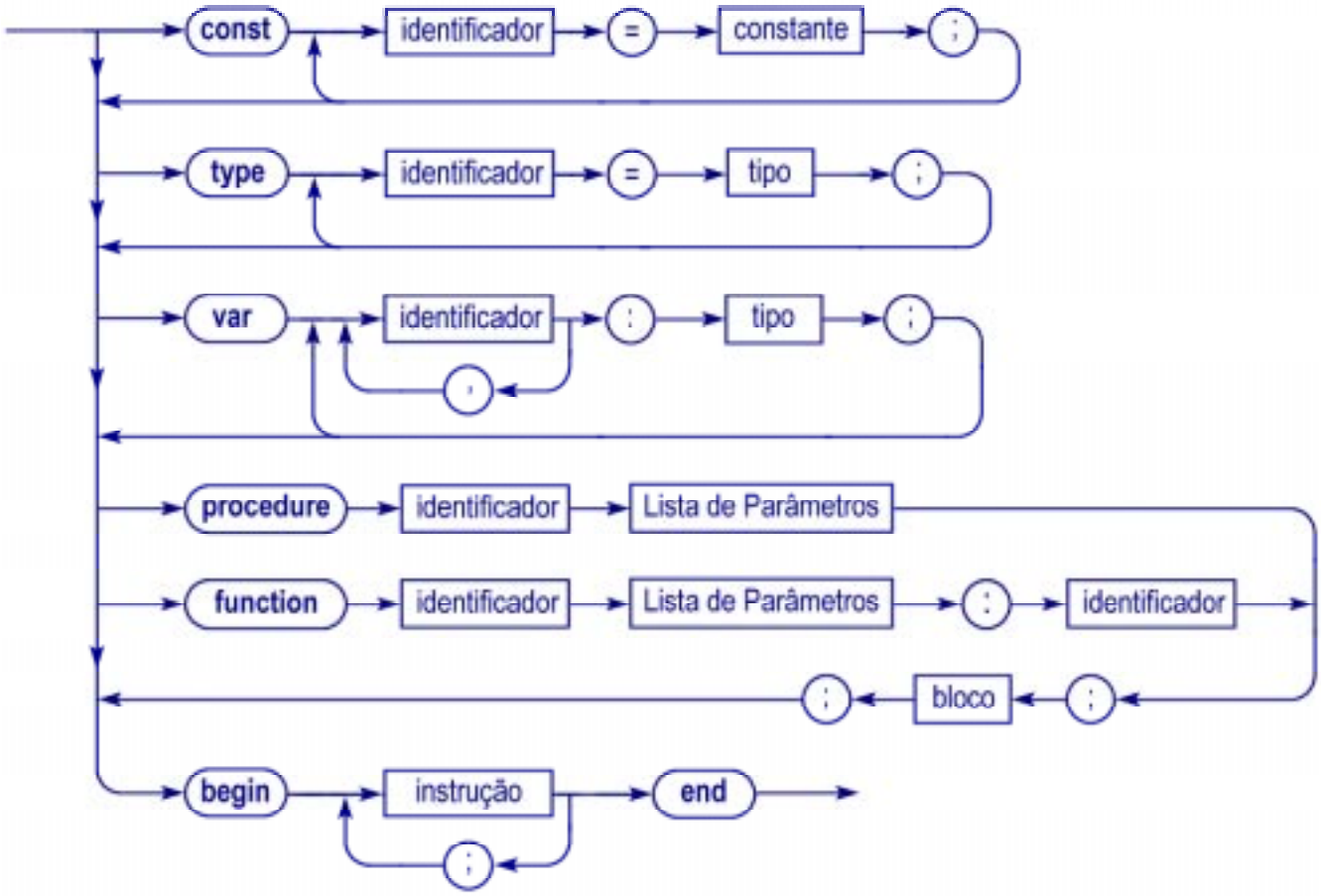
- O comprimento total do Programa.
- As áreas de potenciais erros.
- O esforço de programação.

5.2. Domínio dos identificadores



- O Bloco de um Subprograma é equivalente ao Bloco de um Programa.
- É portanto sintáticamente possível declarar Constantes, Tipos, Variáveis e outros Subprogramas, no Bloco de um Subprograma.

Bloco



- **É conveniente declarar no Bloco de um Subprograma todos os identificadores que são utilizados unicamente nesse Subprograma.**

- **Os identificadores declarados no Bloco de um Subprograma chamam-se locais a esse Subprograma, não podendo ser utilizados fora dele.**
- **Os identificadores declarados no Bloco do Programa Principal são globais, podendo ser utilizados no Programa Principal e em todos os seus Subprogramas.**

p. ex.:

```
var   ultimo : integer;
      bissexto: boolean;     (* São locais ao Procedimento DiaSeguinte *)

var   dia, mes, ano, i, k : integer;      (* São Variáveis globais *)
```

```
program IncrementarData2(input, output);
var   dia, mes, ano, i, k : integer;
procedure  DiaSeguinte;
    var  ultimo : integer;
         bissexto: boolean;
    begin   bissexto:= (ano mod 4 = 0) and
                       (ano mod 100 <> 0) or (ano mod 400 = 0);
            case mes of
                 1, 3, 5, 7, 8, 10, 12 : ultimo := 31;
                 4, 6, 9, 11 : ultimo := 30;
                 2 :  if  bissexto
                      then ultimo := 29
                      else  ultimo := 28
            end;
            if  dia = ultimo
            then   begin   dia := 1;
                           if mes = 12
                           then  begin   mes := 1;
                                         ano := ano+1
                                 end
                           else  mes := mes+1
                   end
            else  dia := dia+1
    end (* DiaSeguinte *);
procedure  LerDados;
           begin   writeln('Indique o dia, mes e ano da presente data');
                   readln(dia, mes, ano);
                   writeln('Quantos dias após a presente data?');
                   readln(k)
           end (* LerDados *);
procedure  EscreverData;
           begin   write(k, ' Dias após a presente data são: ');
                   writeln(dia:3, '/', mes:2, '/', ano:4)
           end (* EscreverData *);
begin (* Programa Principal *)
      LerDados;
      for  i:= 1 to k do
           DiaSeguinte;
      EscreverData
end (* IncrementarData2 *).
```

Bloco do Programa Principal

Bloco do Procedimento DiaSeguinte

De um modo geral, a utilização de Declarações Locais:

- Torna claro que as Variáveis Locais só têm significado dentro dos Subprogramas em que foram declaradas, tornando o programa **mais legível**.
- Assegura que a utilização dessas Variáveis, fora dos Subprogramas onde foram declaradas, provoque um erro imediatamente detectado pelo Compilador, **facilitando a detecção e correcção dos erros**.
- Facilita a minimização do espaço de Memória usado por essas Variáveis, assim tornando o programa **mais eficiente**.

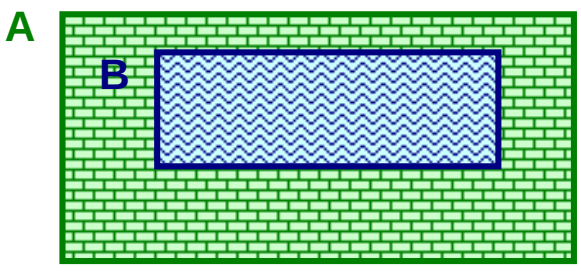
Procedimentos englobados noutros Procedimentos:

- No Bloco de um Subprograma podem ser declarados: Constantes, Tipos, Variáveis e, em particular, outros Subprogramas.

Domínio dos Identificadores:

Regras:

1. O Domínio de um Identificador é o Bloco no qual está declarado e todos os Blocos englobados nesse Bloco, exceptuando a Regra 2.
2. Quando um Identificador é redeclarado num Bloco B, englobado por A, a nova declaração sobrepõe-se à anterior, no Bloco B e em todos os Blocos englobados por B.



Exemplo de Procedimentos Englobados:

```
procedure  DiaSeguinte;
           var  ultimo : integer;

           procedure CalcularUltimo;
                     var bissexto: boolean;
                     begin (* CalcularUltimo *)
                           bissexto:= (ano mod 4 = 0) and
                              (ano mod 100 <> 0) or (ano mod 400 = 0);
                           case mes of
                                 1, 3, 5, 7, 8, 10, 12 : ultimo := 31;
                                 4, 6, 9, 11 : ultimo := 30;
                                 2 :  if  bissexto
                                      then ultimo := 29
                                      else  ultimo := 28
                           end;
                     end (* CalcularUltimo *);

           procedure Actualizar;
                     begin (* Actualizar *)
                           if  dia = ultimo
                           then begin dia := 1;
                                      if mes = 12
                                      then begin mes := 1;
                                                 ano := ano+1
                                           end
                                      else  mes := mes+1
                                end
                           else  dia := dia+1
                     end (* Actualizar *);

           begin (* DiaSeguinte *)
                 CalcularUltimo;
                 Actualizar
           end (* DiaSeguinte *);
```

- **A Variável** ultimo **é local a** DiaSeguinte **e global a** CalcularUltimo **e** Actualizar**.**
- **A Variável** bissexto **é local a** CalcularUltimo**.**
- **Os Procedimentos** CalcularUltimo **e** Actualizar **são locais ao Procedimento** DiaSeguinte.

Exemplo de aplicação das Regra de Domínio dos Identificadores:

```
program P;

var i, j : integer;

procedure Q;

    const i = 16;
    var k : char;

    procedure R;

        var j : real;
        begin  (* Aqui: variável local j : real;
                        constante não-local i = 16;
                        variável não-local k : char;
                        procs. não-locais R e Q *)
        end;

    begin   (* Aqui:  constante local i = 16;
                       variável local k : char;
                       procedimento local R;
                       variável não-local j : integer;
                       procedimento não-local Q *)
    end;

begin  (* Aqui: variáveis locais i, j : integer;
                 procedimento local Q *)
end.
```

- **A "Indentação" deve reflectir a estrutura dos Blocos.**
- **Assume-se que os Identificadores "Standard"** (integer, real, input, maxint, sqrt, abs, read, writeln, ...) **estão declarados num bloco imaginário que engloba todos os Programas Pascal, podendo assim ser sempre utilizados.**

Que fazer?

- **Evitar declarações globais!**
- **Declarar cada Identificador no Bloco em que é usado.**
- **Quando um Identificador tem de ser usado em dois ou mais Procedimentos para representar o mesmo objecto, deve ser declarado no Bloco que imediatamente os engloba.**
- **Localizar sempre todos contadores de ciclos.**

Não esquecendo que:

- **As declarações locais aumentam a liberdade de escolha dos Indentificadores a usar.**
- **Identificadores locais a Procedimentos "disjuntos" são totalmente independentes.**

Exercício:
Qual o resultado de cada um destes programas?

```
program simples (output);
var x : integer;

procedure  alterar;
           begin x:= 1;
           end;

begin  x:= 0;
       alterar;
       writeln(x)
end.
```

Resultado: **1**
(A variável x é global.)

```
program facil (output);
var x : integer;

procedure  mudar;
           var x : integer;
           begin x:= 1;
           end;

begin  x:= 0;
       mudar;
       writeln(x)
end.
```

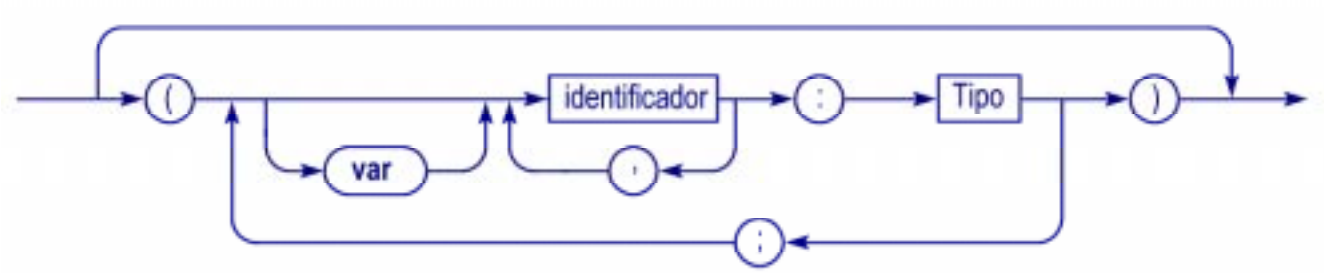
Resultado: **0**
(Duas variáveis x, uma global outra local ao procedimento.)

5.3. Parâmetros

Diagramas de Sintaxe: Declaração de Procedimento:

(Bloco)

Lista de Parâmetros (Formais)



Exemplo:

Parâmetros Formais

```
program MiasDatas (input, output);
var dia, mes, ano, ... : integer;

procedure DiaSeguinte(var dia, mes, ano : integer);

            var  ultimo : integer;
                 bissexto: boolean;
            begin ...
                  ...
            end (* DiaSeguinte *);

begin ...
      (* Aqui 29/2/2000 *)
      DiaSeguinte(dia, mes, ano);
      (* Aqui 1/3/2000 *)
      ...
end (* MaisDatas*).
```

Parâmetros Actuais

Exemplo: Variações sobre o BubbleSort

1ª Variação

```
...
begin   (* Programa Principal *)
        ...
        for  limite:= n downto 2 do
             for  i:= 1 to limite-1 do
                  if x[i] > x[i+1]
                  then begin  aux:= x[i];
                              x[i]:= x[i+1];
                              x[i+1]:= aux
                       end;
        ...
end.
```

2ª Variação

```
...
procedure  trocar(var a, b : real);
           var aux : real;
           begin  aux:= a;
                  a:= b;
                  b:= aux
           end;

begin   (* Programa Principal *)
        ...
        for  limite:= n downto 2 do
             for  i:= 1 to limite-1 do
                  if x[i] > x[i+1]
                  then trocar(x[i], x[i+1]);
        ...
end.
```

3ª Variação

```
procedure ordenar(var a, b : real);
          var aux : real;
          begin if a > b
                then begin aux:= a;
                           a:= b;
                           b:= aux
                     end
          end;

begin   (* Programa Principal *)
        ...
        for limite:= n downto 2 do
            for i:= 1 to limite-1 do
                ordenar(x[i], x[i+1]);
        ...
end.
```

4ª Variação

```
procedure BubbleSort(var x : vector; n : integer);
          var i, limite : integer;
          procedure trocar(var a, b : real);
                    var aux : real;
                    begin   aux:= a;
                            a:= b;
                            b:= aux
                    end;

begin   (* BubbleSort *)

        for limite:= n downto 2 do
            for i:= 1 to limite-1 do
                if x[i] > x[i+1]
                then trocar(x[i], x[i+1])

end (* BubbleSort *);
```

- **Na declaração de um Procedimento é definida uma Lista de Parâmetros Formais;**
- **Na chamada do Procedimento, estes são substituídos pelos correspondentes Parâmetros Actuais (ou Reais).**
- **O número de Parâmetros nas duas listas é necessariamente igual.**

Mais um exemplo com o BubbleSort:

```
program Ordenamento (input, output);
const max = 1000;
type vector = array [1..max] of real;
var  v : vector;
     dim : integer;

procedure LerVector(var x : vector; var n : integer);
          var i : integer;

          begin writeln('Qual a dimensão do vector?');
                readln(n);
                writeln('Escreva os', n, 'elementos, um por linha:');
                for i:= 1 to n do
                    readln(x[i])
          end (* LerVector *);

procedure EscreverVector(x : vector; n : integer);
          var i : integer;

          begin writeln('Vector Final:');
                for i:= 1 to n do
                    writeln(x[i])
          end (* EscreverVector *);
```

```
procedure BubbleSort(var x : vector; n : integer);
          var i, limite : integer;

          procedure trocar(var a, b : real);
                    var aux : real;
                    begin   aux:= a;
                            a:= b;
                            b:= aux
                    end;

          begin for limite:= n downto 2 do
                    for i:= 1 to limite-1 do
                        if x[i] > x[i+1]
                        then trocar(x[i], x[i+1])
          end (* BubbleSort *);

begin (* Programa Principal *)
      LerVector(v, dim);
      BubbleSort(v, dim);
      EscreverVector(v, dim)
end (* Ordenamento *).
```

- **No Procedimento** LerVector **os valores dos dois Parâmetros** v **e** dim **foram alterados;**
- **No Procedimento** BubbleSort **o vector** v **foi alterado (ordenado), mas não o valor da sua dimesão** dim**;**
- **No Procedimento** EscreverVector **nenhum dos valores dos seus Parâmetros** v **ou** dim **foi alterado.**

Qual é a diferença?

Classes de Parâmetros (Modos de Ligação):

- **Classe Variável - Ligação por Referência**
- **Classe Valor - Ligação por Valor**

Exemplo:

```
program simples (output);
var x : integer;

procedure  alterar(var a : integer);
           begin a:= 1;
           end;

begin  x:= 0;
       alterar(x);
       writeln(x)
end.
```

Resultado: **1**

O Parâmetro Formal a **foi declarado da Classe Variável.**

A chamada do Procedimento alterou o Valor do Parâmetro Actual x **no Programa Principal.**

```
program facil (output);
var x : integer;

procedure  mudar(a : integer);
           begin a:= 1;
           end;

begin  x:= 0;
       mudar(x);
       writeln(x)
end.
```

Resultado: **0**

O Parâmetro Formal a **foi declarado da Classe Valor.**

A chamada do Procedimento não mudou o Valor do Parâmetro Actual x **no Programa Principal.**