INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
BAHIA

Modularização

Modularização

- A **modularização** consiste num **método** utilizado para **facilitar** a **construção** de **grandes programas**;
- A idéia é **dividir grandes programas** em **pequenas etapas**, que são os ***módulos*** ou ***subprogramas*****.**
- A **primeira etapta** delas, por onde começa a execução do trabalho, recebe o nome de ***programa principal*****,** e as outras são os ***subprogramas*** propriamente ditos;
- Os **subprogramas** são **executados** sempre que ocorre uma **chamada** dos mesmos;

Vantagens

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA BAHIA

- **Vantagens da utilização de subprogramas:**
  - Economia de código: escreve-se menos;
  - Desenvolvimento modularizado: pensa-se no algoritmo por partes;
  - Facilidade de depuração (correção/acompanhamento): é mais fácil corrigir/detectar um erro apenas uma
  - vez do que dez vezes;
  - Facilidade de alteração do código: se é preciso alterar, altera-se apenas uma vez;
  - Generalidade de código com o uso de parâmetros: escreve-se algoritmos para situações genéricas.
- Há duas espécies de subprogramas: **PROCEDIMENTO** e **FUNÇÃO**.

Procedimento

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA BAHIA

- Um **subprograma** do tipo **PROCEDIMENTO** é, na realidade, um **programa** com **vida própria**;

- Para ser **processado** ele tem que ser **solicitado** pelo **programa principal** que o contém, ou por outro **subprograma**, ou por ele **mesmo**.

Declaração Procedimento

Declaração:

```
PROCEDURE nome;
   declaração dos objetos locais ao Procedimento
   BEGIN
      comandos do Procedimento
   END;
```

onde: ***nome*** é o identificador associado ao procedimento.

- Exemplo sem procedimento

```
Program CALCULA_MÉDIA; {sem o uso de procedimentos}

var
   NOTA1,NOTA2,MEDIA : real;

begin
{lê as notas}
   write('Digite a primeira nota: ');
   readln(NOTA1);
   write('Digite a segunda nota: ');
   readln(NOTA2);
{calcula a media}
   MEDIA := (NOTA1 + NOTA2) / 2;
{escreve o resultado}
   writeln('Media = ',MEDIA,4:1)
end.
```

Registro em Pascal

- Mostraremos agora o mesmo programa, utilizando um procedimento.

```
Program CALCULA_MÉDIA; {usando procedimento}
var
   NOTA1,NOTA2,MEDIA : real;

{declaração do procedimento}
procedure LER_NOTAS;
   begin
   write('Digite a primeira nota: ');
   readln(NOTA1);
   write('Digite a segunda nota: ');
   readln(NOTA2);
   end;

{Programa Principal}
begin
   LER_NOTAS; {ativação do procedimento LER_NOTAS}
   MEDIA := (NOTA1 + NOTA2) / 2; {calcula a media}
   Writeln('Media = ',MEDIA,4:1) {escreve o resultado}
end.
```

Função

- As **funções**, embora bastante **semelhantes** aos **procedimentos**, têm a característica especial de **retornar** ao **programa** que as **chamou** um valor **associado** ao nome da função.

- Esta **característica** permite uma **analogia** com o conceito de **função** da **Matemática**.

Função

- Declaração:

```
FUNCTION nome : tipo;
   declaração dos objetos locais à Função
   BEGIN
      Comandos da Função
   END;
```

onde: ***nome*** é o identificador associado à função.
tipo é o tipo da função, ou seja, o tipo do valor de retorno.

Função

- Exemplo: O programa calcula a média dos elementos de um vetor, **sem uso** de **Procedimentos** ou **Funções**.

```
Program SOMA_VETOR; {sem o uso de procedimentos ou funções}
const N = 30;
var  VETOR : array[1..N] of integer;
     I,SOMA,MEDIA : integer;
begin
{lê os valores do vetor}
    for I:=1 to N do
        readln(VETOR[I]);
{calcula a media}
    SOMA := 0;
    for I:=1 to N do
        SOMA := SOMA + VETOR[I];
    MEDIA := SOMA div N;
{escreve o resultado}
    writeln(MEDIA)
end.
```

Função

- Mesmo programa, **utilizando** um **procedimento** para **ler** os valores do vetor e uma função para efetuar o cálculo da média.

```
Program SOMA_VETOR; {usando uma função e um procedimento}

const
   N = 30;

var
   VETOR : array[1..N] of integer;

{declaração do procedimento}
procedure LER_DADOS;
   var I : integer;
   begin
      for I:=1 to N do
          readln(VETOR[I]);
   end;

{declaração da função}
function MEDIA : integer;
   var I,SOMA : integer;
   begin
      SOMA := 0;
      for I:=1 to N do
         SOMA := SOMA + VETOR[I];
      MEDIA := SOMA div N;
   end;
```

```
{Programa Principal}
begin
{ativa o procedimento LER_DADOS}
    LER_DADOS;
{escreve o resultado, chamando a função MEDIA}
    writeln(MEDIA)
end.
```

Variáveis Globais e Locais

- Observe que, no exemplo anterior, **declaramos** uma **variável** no **programa principal** e **outras** nos **subprogramas**;

- Podemos dizer que a variável **VETOR**, que foi **declarada** no **programa principal** é uma ***variável global*** aos subprogramas;

- A variável **I** é dita ***variável local*** ao procedimento **LER_DADOS** e as variáveis **I** e **SOMA** são **locais** à função **MEDIA**;

```
var
   VETOR : array[1..N] of integer;

{declaração do procedimento}
procedure LER_DADOS;
   var I : integer;
   begin
      for I:=1 to N do
          readln(VETOR[I]);
   end;

{declaração da função}
function MEDIA : integer;
   var I,SOMA : integer;
   begin
      SOMA := 0;
      for I:=1 to N do
         SOMA := SOMA + VETOR[I];
      MEDIA := SOMA div N;
   end;
```

Variáveis Globais e Locais

- O uso de ***variáveis globais*** dentro de **procedimentos** e **funções** serve para implementar um mecanismo de **transmissão** de **informações** de um **nível** mais **externo** para um mais **interno**.

- As ***variáveis locais*** dos **procedimentos** e **funções** são criadas e alocadas quando da sua **ativação** e automaticamente **liberadas** quando do seu **término**.

```
var
   VETOR : array[1..N] of integer;

{declaração do procedimento}
procedure LER_DADOS;
   var I : integer;
   begin
      for I:=1 to N do
          readln(VETOR[I]);
   end;

{declaração da função}
function MEDIA : integer;
   var I,SOMA : integer;
   begin
      SOMA := 0;
      for I:=1 to N do
         SOMA := SOMA + VETOR[I];
      MEDIA := SOMA div N;
   end;
```

- A utilização de ***variáveis globais*** não constitui, no entanto, uma boa **prática** de **programação**.

Variáveis Globais e Locais

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA BAHIA

- Recomendações
  - Todo **sub-programas** devem **apenas** utilizar as ***variáveis locais***, conhecidas dentro dos **mesmos**;
  - A **transmissão** de **informações** para dentro e fora dos **subprogramas** deve ser feita através dos ***parâmetros*** de **transmissão**

Parâmetros

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA BAHIA

- **Utilizado** quando se deseja **escrever** um **subprograma** que seja o mais **genérico possível**, deve-se usar a **passagem** de **parâmetros**;

- Dá-se a designação de ***parâmetro real*** ou ***de chamada*** ao objeto utilizado na unidade chamadora/ativadora e de ***parâmetro formal*** ou ***de definição*** ao recebido como parâmetro no **subprograma**.

- Dentre os modos de passagem de parâmetros, podemos destacar a ***passagem por valor*** e a ***passagem por referência*.**

Parâmetros

- Na **linguagem Pascal**, a **declaração** dos **procedimentos** e **funções** com **parâmetros** se **diferencia** da forma já **apresentada** apenas pela inclusão da **lista de parâmetros formais** no cabeçalho.

- Esta deve vir entre **parênteses** e **cada parâmetro** deve ter o seu **tipo** especificado. A forma geral é:

```
PROCEDURE nome (lista de parâmetros formais)

FUNCTION nome (lista de parâmetros formais) : tipo
```

A ***lista de parâmetros formais*** tem a seguinte forma:

```
parâmetro1 : tipo; parâmetro2 : tipo; ...; parâmetro n : tipo
```

Exemplos da lista de parâmetros:

```
procedure PROC (X,Y,Z:integer; K:real)

function FUNC (A,B:real; C:string) : integer
```

Passagem por Valor

- Na passagem de parâmetros ***por valor***, as **alterações** feitas nos **parâmetros formais**, dentro do **subprograma**, **não** se **refletem** nos **parâmetros reais**.

- O valor do **parâmetro real** é **copiado** no **parâmetro formal**, na chamada do subprograma. Assim, quando a passagem é por **valor**, isto significa que o parâmetro é de ***entrada*****.**

```
Program EXEMPLO_PASSAGEM_PARÂMETROS;

var N1,N2 : integer;

Procedure PROC(X:integer; var Y:integer)
   begin
      X:=1;
      Y:=1;
   end;

begin
   N1:=0; N2:=0;
   PROC(N1,N2);
   writeln(N1); {será exibido o valor 0}
   writeln(N2); (será exibido o valor 1}
end.
```

Passagem por Rererência

- Na passagem de parâmetros ***por referência***, toda **alteração** feita num **parâmetro formal** corresponde a **mesma alteração** feita no seu **parâmetro** real **associado**.

- Assim, quando a passagem é por **referência**, isto significa que o parâmetro é de ***entrada-saída***.

```
Program EXEMPLO_PASSAGEM_PARÂMETROS;

var N1,N2 : integer;

Procedure PROC(X:integer; var Y:integer)
   begin
      X:=1;
      Y:=1;
   end;

begin
   N1:=0; N2:=0;
   PROC(N1,N2);
   writeln(N1); {será exibido o valor 0}
   writeln(N2); (será exibido o valor 1}
end.
```

- Para se utilizar a **passagem** por **referência**, deve-se acrescentar a palavra **VAR** antes do **nome** do **parâmetro**.

Passagem de Parâmetros

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA BAHIA

- OBS: Na **chamada** de **procedimentos** ou **funções** utilizando **parâmetros**, devemos **acrescentar** após o **nome** do procedimento ou função uma **lista de parâmetros reais** (de chamada), os quais devem ser do **mesmo tipo** e **quantidade** dos **parâmetros formais declarados.**

```
Program EXEMPLO_PASSAGEM_PARÂMETROS;

var N1,N2 : integer;

Procedure PROC(X:integer; var Y:integer);
   begin
      X:=1;
      Y:=1;
   end;

begin
   N1:=0; N2:=0;
   PROC(N1,N2);
   writeln(N1); {será exibido o valor 0}
   writeln(N2); (será exibido o valor 1}
end.
```

Exemplo

- 1 - Escrever uma função chamada MAIOR que receba dois números inteiros e retorne o maior deles. Escrever um programa para ler dois números inteiros e, utilizando a função MAIOR, calcular e exibir o maior valor entre os números lidos.

```
Program CALCULA_MAIOR;

var X,Y,M : integer;

function MAIOR (NUM1,NUM2:integer) : integer;
   begin
      If NUM1 > NUM2 then
         MAIOR := NUM1
      else
         MAIOR := NUM2;
   end;

begin
   readln(X,Y);
   M := MAIOR(X,Y);
   writeln(M);
end.
```

Exemplos

- 2 - Escrever um procedimento chamado DOBRA que multiplique um número inteiro (recebido como parâmetro) por 2. Escrever um programa para ler um valor inteiro e , utilizando o procedimento DOBRA, calcular e exibir o dobro do mesmo.

```
Program CALCULA_DOBRO;

var X : integer;

procedure DOBRA (var NUM:integer);
   begin
      NUM := NUM * 2
   end;

begin
   readln(X);
   DOBRA(X);
   writeln(X);
end.
```

Exercício