



FUNÇÕES E PROCEDIMENTOS

Prof. André Vignatti

PORQUE ESTUDAR FUNÇÕES/PROCEDIMENTOS?

- modularidade
- reaproveitamento de código
- legibilidade do código

O QUE O ALUNO DEVE ENTENDER?

1. quando usar **função** e quando usar **procedimento**?
2. quando usar **variáveis locais** ou **variáveis globais**?
3. quando usar **passagem de parâmetros por valor** ou por **referência**?

EXEMPLO BÁSICO

Problema: ler uma sequência de valores inteiros terminada por zero, imprima somente os que são pares

```
program imprime_pares;  
var a: integer;  
begin  
    read (a);  
    while a <> 0 do  
    begin  
        if a mod 2 = 0 then  
            writeln (a);  
        read (a);  
    end;  
end.
```

variável global



VARIÁVEIS GLOBAIS

- declaradas antes do *begin* do programa principal
- visíveis em todo o programa (mesmo nos subprogramas)

```
program imprime_pares;  
var a: integer;  
begin  
    read (a);  
    while a <> 0 do  
    begin  
        if a mod 2 = 0 then  
            writeln (a);  
        read (a);  
    end;  
end.
```

→ **variável global**

FUNÇÕES

Função: subprograma que pode ser usado pelo programa principal

```
program imprime_pares;  
var a: integer;
```

```
(* funcao que calcula se a variavel global a eh par *)
```

```
function a_eh_par: boolean;  
begin
```

```
(* codigo da funcao, ainda nao escrito por razoes didaticas *)
```

```
end;
```

```
begin (* programa principal *)
```

```
  read (a);
```

```
  while a <> 0 do
```

```
  begin
```

```
    if a_eh_par then
```

```
      writeln (a);
```

```
      read (a);
```

```
    end;
```

```
end.
```

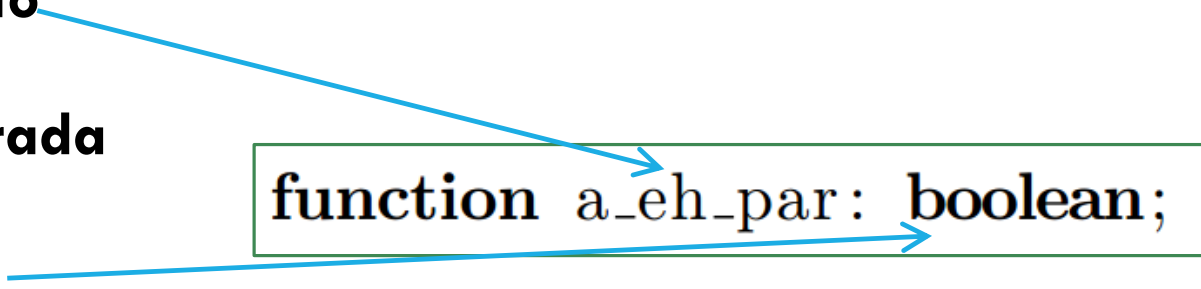
função

chamada da função

PROTÓTIPO DA FUNÇÃO

Protótipo ou **interface** da função é composto de:

1. **nome da função**
2. **valores de entrada**
3. **valor de saída**



```
function a_eh_par: boolean;
```

Obs:

- **valores de entrada:** 0 ou mais parâmetros (variáveis), de diversos tipos
- **valor de saída:** somente 1 único

IMPLEMENTANDO A FUNÇÃO

- várias formas de implementar
- a função exige saída, então usa-se o nome dela para atribuir um valor

```
(* primeira versao da funcao *)  
if a mod 2 = 0 then  
    a_eh_par:= true  
else  
    a_eh_par:= false;
```

```
(* segunda versao da funcao *)  
if a mod 2 <> 0 then  
    a_eh_par:= false  
else  
    a_eh_par:= true;
```

```
(* terceira versao da funcao *)  
if a mod 2 < 1 then  
    a_eh_par:= true  
else  
    a_eh_par:= false;
```

```
(* quarta versao da funcao *)  
if a mod 2 = 1 then  
    a_eh_par:= false  
else  
    a_eh_par:= true;
```


USANDO PARÂMETROS

até agora: modularidade de código

○ que aconteceria se tivéssemos que testar se DUAS variáveis diferentes são par?

- **Solução 1:** escrever a função “b_eh_par”
- **Solução 2:** reaproveitar código → usar **parâmetros** na função

```
function eh_par (n: integer): boolean;
```

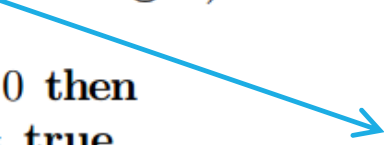


o parâmetro da função

```
program imprime_pares;
var a: integer;

(* funcao que calcula se a variavel global a eh par *)
function eh_par (n: integer): boolean;
begin
    if n mod 2 = 0 then
        eh_par:= true
    else
        eh_par:= false;
end;

begin (* programa principal *)
    read (a);
    while a <> 0 do
    begin
        if eh_par (a) then
            writeln (a);
        read (a);
    end;
end.
```



só existe durante a execução da função

PROCEDIMENTOS

Procedimento é uma função sem valor de saída

Exemplo: faça um procedimento que troca o valor de duas variáveis real

1ª tentativa:
(**não funciona!**)

```
procedure troca (a, b : real);  
begin  
    temp:= a;  
    a:= b;  
    b:= temp;  
end;
```

2ª tentativa:
(**funciona!**)

```
procedure troca (var a, b : real);  
begin  
    temp:= a;  
    a:= b;  
    b:= temp;  
end;
```

PASSAGEM DE PARÂMETROS POR VALOR E REFERÊNCIA

Diferença das tentativas é a forma de passagem de parâmetros:

- ✓ a 1ª é por **valor**
- ✓ a 2ª é por **referência**

Passagem de parâmetro por valor: ao iniciar a função/procedimento, é usada uma **cópia da variável**

Passagem de parâmetro por referência: ao iniciar a função/procedimento, é usada **variável verdadeira**

```
program imprimetrocado;  
var x,y,temp: real; (* variaveis globais *)  
  
(* procedimento que troca os conteudos da variaveis *)  
procedure troca (var a, b: real);  
begin  
    temp:= a;  
    a:= b;  
    b:= temp;  
end;  
  
begin (* programa principal *)  
    read (x,y);  
    troca (x,y);  
    writeln (x,y);  
end.
```

VARIÁVEIS LOCAIS

Variáveis locais: só existem dentro do escopo da função/procedimento

```
program imprimetrocado;
var x,y: real; (* variaveis globais *)

(* procedimento que troca os conteudos da variaveis *)
procedure troca (var a, b: real);
var temp: real; (* variavel local, temporaria para uso exclusivo neste procedimento *)
begin
    temp:= a;
    a:= b;
    b:= temp;
end;

begin (* programa principal *)
    read (x,y);
    troca (x,y);
    writeln (x,y);
end.
```

Vantagem: modularidade do código