

CI1055: Algoritmos e Estruturas de Dados I

Profs. Drs. Marcos Castilho, Bruno Müller Jr, Carmem Hara

Departamento de Informática/UFPR

11 de agosto de 2020

Resumo

Passagem de parâmetros por valor (ou cópia)

Objetivos da aula

- Explicar o conceito de passagem de parâmetros por valor, também conhecido como passagem de parâmetros por cópia.

Resumo da aula anterior

```
1 program imprime_pares_final;
2 var a: integer;
3
4 (* funcao que calcula se a variavel global a eh par *)
5 function a_eh_par: boolean;
6 begin
7     a_eh_par:= true;
8     if a mod 2 <> 0 then
9         a_eh_par:= false
10 end;
11
12 begin (* programa principal *)
13     read (a);
14     while a <> 0 do
15         begin
16             if a_eh_par then (* chamada da funcao *)
17                 writeln (a);
18             read (a);
19         end;
20 end.
```

Motivação

```
1 program imprime_dois_pares_v0;
2 var a, b: integer;
3
4 (* funcao que calcula se a variavel global a eh par *)
5 function a_eh_par: boolean;
6 begin
7     a_eh_par:= true;
8     if a mod 2 <> 0 then
9         a_eh_par:= false
10 end;
11
12 (* funcao que calcula se a variavel global a eh par *)
13 function b_eh_par: boolean;
14 begin
15     b_eh_par:= true;
16     if b mod 2 <> 0 then
17         b_eh_par:= false
18 end;
19
20 begin (* programa principal *)
21     read (a, b);
22     while (a <> 0) or (b <> 0) do
23         begin
24             if a_eh_par and b_eh_par then (* chamada das funcoes *)
25                 writeln (a, ' ', b);
26             read (a, b);
27         end;
28 end.
```

Parâmetros por valor (ou por cópia)

- Um parâmetro é uma informação que a função pode receber do programa que a chamou
- Existem duas maneiras de se passar parâmetros
- No momento veremos uma delas

Parâmetros por valor (ou por cópia)

- Queremos informar à função que os cálculos devem ser feitos para um certo número inteiro
- A cada chamada da função, pode-se passar um número inteiro diferente
- Este parâmetro deve ter um identificador qualquer, não importa seu nome, importa o seu tipo!

Exemplo de protótipo da função

```
1  (*  
2     esta funcao recebe um valor inteiro n e retorna true se  
3     este inteiro n for par e false em caso contrario  
4  *)  
5  function eh_par (n: integer): boolean;
```

- No momento não importam os cálculos
- O importante agora é *como* este valor é recebido na função

Exemplo de programa principal

```
1  program imprime_dois_pares_v1;  
2  var a, b: integer;  
3  
4  (* funcao que calcula se o parametro n eh par *)  
5  function eh_par (n: integer): boolean;  
6  begin  
7      (* codigo da funcao *)  
8  end;  
9  
10 begin (* programa principal *)  
11     read (a, b);  
12     while (a <> 0) or (b <> 0) do  
13     begin  
14         if eh_par (a) and eh_par (b) then (* chamada das funcoes *)  
15             writeln (a, ' ', b);  
16             read (a, b);  
17     end;  
18 end.
```

- Na chamada `eh_par (a)`, `n` recebe uma cópia do valor da variável `a`
- Na chamada `eh_par (b)`, `n` recebe uma cópia do valor da variável `b`


```
1 program imprime_dois_pares_final;  
2 var a, b: integer;  
3  
4 (* funcao que calcula se o parametro n eh par *)  
5 function eh_par (n: integer): boolean;  
6 begin  
7     eh_par:= true;  
8     if n mod 2  $\diamond$  0 then  
9         eh_par:= false  
10 end;  
11  
12 begin (* programa principal *)  
13     read (a, b);  
14     while (a  $\diamond$  0) or (b  $\diamond$  0) do  
15         begin  
16             if eh_par (a) and eh_par (b) then (* chamada das funcoes *)  
17                 writeln (a, ' ', b);  
18             read (a, b);  
19         end;  
20 end.
```

Vejam os como isso funciona

- Será apresentada uma simulação da execução do programa.
- Ao incluir funções, o teste de mesa deve ser modificado para diagramas de execução (Tomasz Kowaltowski);
- A ideia é associar cada chamada de função (ou procedimento) a uma folha de papel;
- Esta folha armazena informações locais à função ou procedimento tais como o nome e valor das variáveis locais e local para retornar o fluxo ao terminar a execução.

Diagrama de Execução

```
program imprime_dois_pares_final;
var a, b: integer;

function eh_par (n: integer): boolean;
begin
    eh_par:= true;
    if n mod 2 <> 0 then
        eh_par:= false
    end;
end;

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

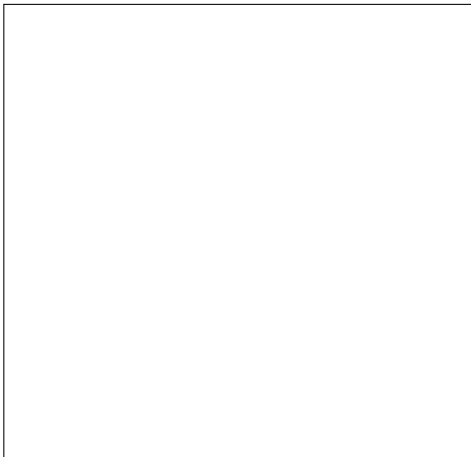
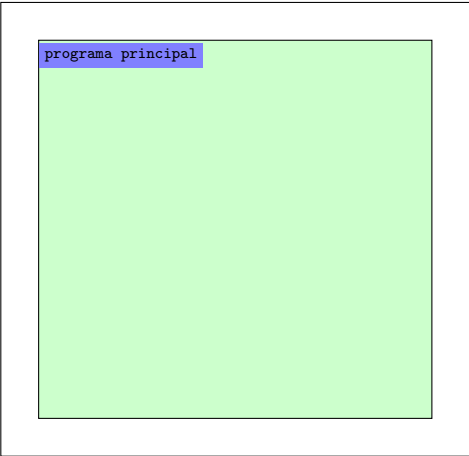


Diagrama de Execução

```
program imprime_dois_pares_final;  
var a, b: integer;  
  
function eh_par (n: integer): boolean;  
begin  
  eh_par:= true;  
  if n mod 2 <> 0 then  
    eh_par:= false  
end;  
  
begin (* programa principal *)  
  read (a, b);  
  while (a <> 0) or (b <> 0) do  
  begin  
    if eh_par (a) and eh_par (b) then  
      writeln (a, ' ', b);  
    read (a, b);  
  end;  
end.
```

Inicia folha de papel do
programa principal



programa principal

Diagrama de Execução

```
program imprime_dois_pares_final;  
var a, b: integer;  
  
function eh_par (n: integer): boolean;  
begin  
  eh_par:= true;  
  if n mod 2 <> 0 then  
    eh_par:= false  
end;  
  
begin (* programa principal *)  
  read (a, b);  
  while (a <> 0) or (b <> 0) do  
  begin  
    if eh_par (a) and eh_par (b) then  
      writeln (a, ' ', b);  
    read (a, b);  
  end;  
end.
```

Cria espaço para as variáveis
do programa principal

programa principal

a: ? b: ?

Diagrama de Execução

```
program imprime_dois_pares_final;  
var a, b: integer;  
  
function eh_par (n: integer): boolean;  
begin  
  eh_par:= true;  
  if n mod 2 <> 0 then  
    eh_par:= false  
end;  
  
begin (* programa principal *)  
  read (a, b);  
  while (a <> 0) or (b <> 0) do  
  begin  
    if eh_par (a) and eh_par (b) then  
      writeln (a, ' ', b);  
    read (a, b);  
  end;  
end.
```

programa principal

a: ? b: ?

Diagrama de Execução

```
program imprime_dois_pares_final;  
var a, b: integer;  
  
function eh_par (n: integer): boolean;  
begin  
  eh_par:= true;  
  if n mod 2 <> 0 then  
    eh_par:= false  
end;  
  
begin (* programa principal *)  
  read (a, b);  
  while (a <> 0) or (b <> 0) do  
  begin  
    if eh_par (a) and eh_par (b) then  
      writeln (a, ' ', b);  
    read (a, b);  
  end;  
end.  
  
usuário digita 2,3
```

programa principal

a: 2 b: 3

Diagrama de Execução

```
program imprime_dois_pares_final;  
var a, b: integer;  
  
function eh_par (n: integer): boolean;  
begin  
  eh_par:= true;  
  if n mod 2 <> 0 then  
    eh_par:= false  
end;  
  
begin (* programa principal *)  
  read (a, b);  
  while (a <> 0) or (b <> 0) do  
  begin  
    if eh_par (a) and eh_par (b) then  
      writeln (a, ' ', b);  
    read (a, b);  
  end;  
end.
```

programa principal

a: 2 b: 3

Diagrama de Execução

```
program imprime_dois_pares_final;  
var a, b: integer;  
  
function eh_par (n: integer): boolean;  
begin  
  eh_par:= true;  
  if n mod 2 <> 0 then  
    eh_par:= false  
end;  
  
begin (* programa principal *)  
  read (a, b);  
  while (a <> 0) or (b <> 0) do  
  begin  
    if eh_par (a) and eh_par (b) then  
      writeln (a, ' ', b);  
    read (a, b);  
  end;  
end.
```

1. Desvia fluxo
2. Cria nova folha
3. Aloca variáveis: eh_par, n:=cópia de a
4. Indica onde retornar ao finalizar

programa principal

a: 2 b: 3

Diagrama de Execução

```
program imprime_dois_pares_final;  
var a, b: integer;  
  
function eh_par (n: integer): boolean;  
begin  
  eh_par:= true;  
  if n mod 2 <> 0 then  
    eh_par:= false  
end;  
  
begin (* programa principal *)  
  read (a, b);  
  while (a <> 0) or (b <> 0) do  
  begin  
    if eh_par (a) ⌚ and eh_par (b) then  
      writeln (a, ' ', b);  
    read (a, b);  
  end;  
end.
```

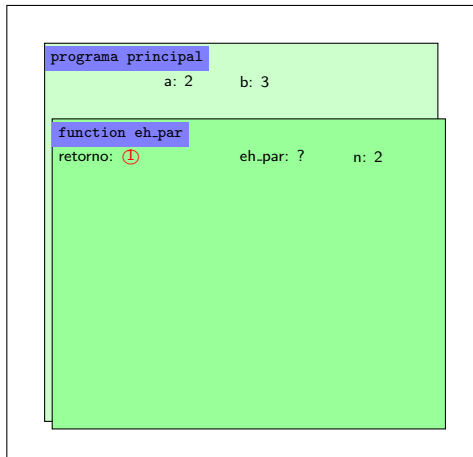


Diagrama de Execução

```
program imprime_dois_pares_final;  
var a, b: integer;  
  
function eh_par (n: integer): boolean;  
begin  
    eh_par:= true;  
    if n mod 2 <> 0 then  
        eh_par:= false  
    end;  
end;  
  
begin (* programa principal *)  
    read (a, b);  
    while (a <> 0) or (b <> 0) do  
        begin  
            if eh_par (a) ⌚ and eh_par (b) then  
                writeln (a, ' ', b);  
            read (a, b);  
        end;  
    end;  
end.
```

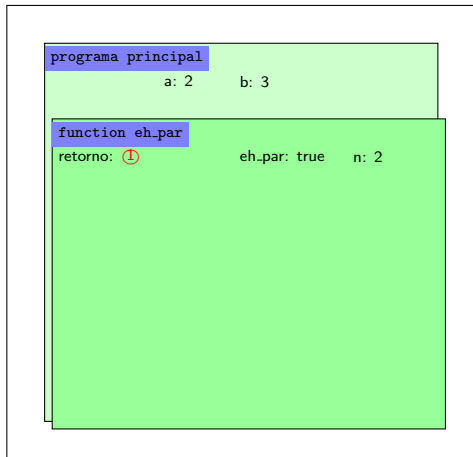


Diagrama de Execução

```
program imprime_dois_pares_final;  
var a, b: integer;  
  
function eh_par (n: integer): boolean;  
begin  
  eh_par:= true;  
  if n mod 2 <> 0 then  
    eh_par:= false  
end;  
  
begin (* programa principal *)  
  read (a, b);  
  while (a <> 0) or (b <> 0) do  
  begin  
    if eh_par (a) ⌚ and eh_par (b) then  
      writeln (a, ' ', b);  
    read (a, b);  
  end;  
end.
```

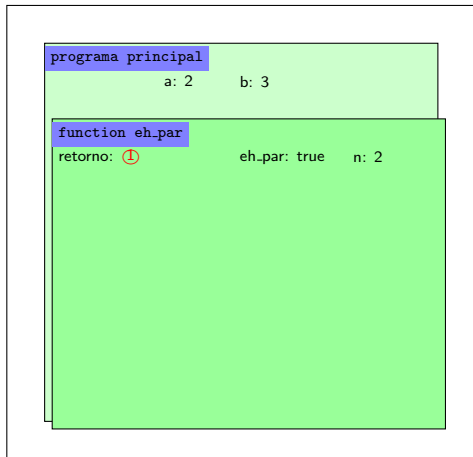


Diagrama de Execução

```
program imprime_dois_pares_final;  
var a, b: integer;  
  
function eh_par (n: integer): boolean;  
begin  
  eh_par:= true;  
  if n mod 2 <> 0 then  
    eh_par:= false  
end;  
  
begin (* programa principal *)  
  read (a, b);  
  while (a <> 0) or (b <> 0) do  
  begin  
    if eh_par (a) Ⓛ and eh_par (b) then  
      writeln (a, ' ', b);  
    read (a, b);  
  end;  
end.
```

1. coloca o valor de eh_par no chamador
2. desvia fluxo para o local indicado em retorno
3. retira folha

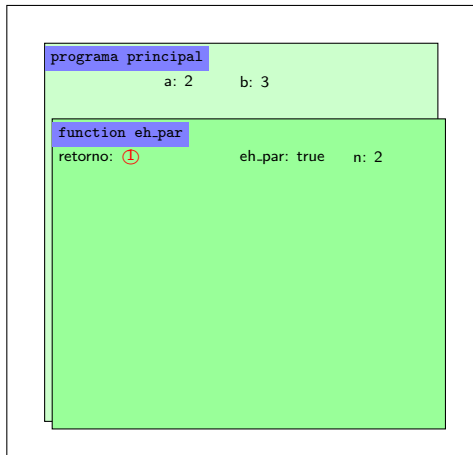


Diagrama de Execução

```
program imprime_dois_pares_final;  
var a, b: integer;  
  
function eh_par (n: integer): boolean;  
begin  
  eh_par:= true;  
  if n mod 2 <> 0 then  
    eh_par:= false  
end;  
  
begin (* programa principal *)  
  read (a, b);  
  while (a <> 0) or (b <> 0) do  
    begin (*true*)  
      if eh_par (a) and eh_par (b) then  
        writeln (a, ' ', b);  
      read (a, b);  
    end;  
end.
```

1. Desvia fluxo
2. Cria nova folha
3. Aloca variáveis: eh_par, n:=cópia de b
4. Indica onde retornar ao finalizar

programa principal

a: 2 b: 3

Diagrama de Execução

```
program imprime_dois_pares_final;  
var a, b: integer;  
  
function eh_par (n: integer): boolean;  
begin  
  eh_par:= true;  
  if n mod 2 <> 0 then  
    eh_par:= false  
end;  
  
begin (* programa principal *)  
  read (a, b);  
  while (a <> 0) or (b <> 0) do  
    begin (*true*)  
      if eh_par (a) and eh_par (b) then  
        writeln (a, ' ', b);  
      read (a, b);  
    end;  
end.
```

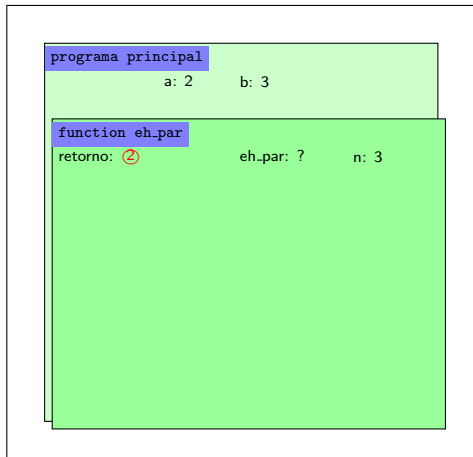


Diagrama de Execução

```
program imprime_dois_pares_final;  
var a, b: integer;  
  
function eh_par (n: integer): boolean;  
begin  
    eh_par:= true;  
    if n mod 2 <> 0 then  
        eh_par:= false  
    end;  
end;  
  
begin (* programa principal *)  
    read (a, b);  
    while (a <> 0) or (b <> 0) do  
        begin (*true*)  
            if eh_par (a) and eh_par (b) then  
                writeln (a, ' ', b);  
            read (a, b);  
        end;  
    end;  
end.
```

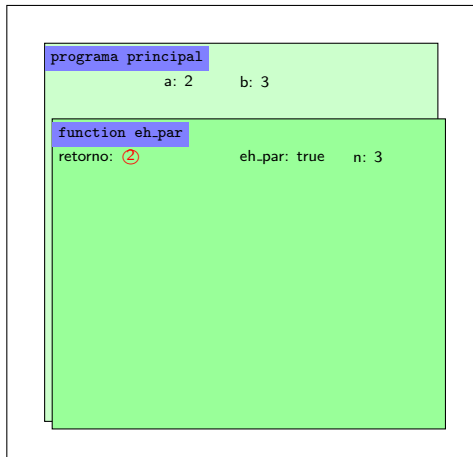


Diagrama de Execução

```
program imprime_dois_pares_final;  
var a, b: integer;  
  
function eh_par (n: integer): boolean;  
begin  
  eh_par:= true;  
  if n mod 2 <> 0 then  
    eh_par:= false  
end;  
  
begin (* programa principal *)  
  read (a, b);  
  while (a <> 0) or (b <> 0) do  
    begin (*true*)  
      if eh_par (a) and eh_par (b) then  
        writeln (a, ' ', b);  
      read (a, b);  
    end;  
  end.  
end.
```

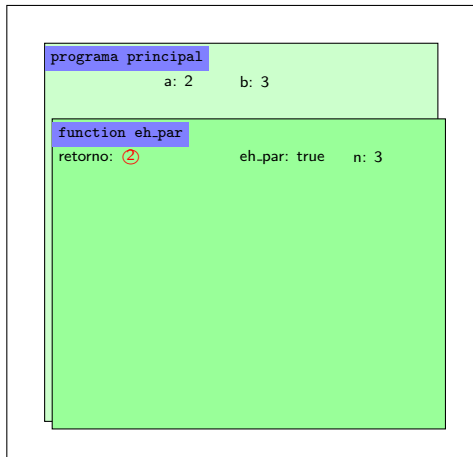


Diagrama de Execução

```
program imprime_dois_pares_final;  
var a, b: integer;  
  
function eh_par (n: integer): boolean;  
begin  
  eh_par:= true;  
  if n mod 2 <> 0 then  
    eh_par:= false  
end;  
  
begin (* programa principal *)  
  read (a, b);  
  while (a <> 0) or (b <> 0) do  
    begin (*true*)  
      if eh_par (a) and eh_par (b) then  
        writeln (a, ' ', b);  
      read (a, b);  
    end;  
end.
```

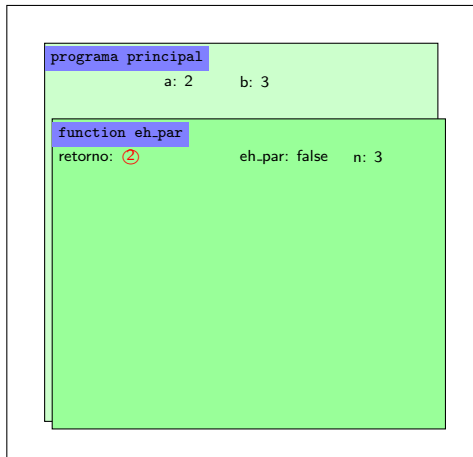


Diagrama de Execução

```
program imprime_dois_pares_final;  
var a, b: integer;  
  
function eh_par (n: integer): boolean;  
begin  
  eh_par:= true;  
  if n mod 2 <> 0 then  
    eh_par:= false  
end;  
  
begin (* programa principal *)  
  read (a, b);  
  while (a <> 0) or (b <> 0) do  
    begin (*true*)  
      if eh_par (a) and eh_par (b) then  
        writeln (a, ' ', b);  
        read (a, b);  
      end;  
    end.  
1. coloca o valor de eh_par no chamador  
2. desvia fluxo para o local indicado em retorno  
3. retira folha
```

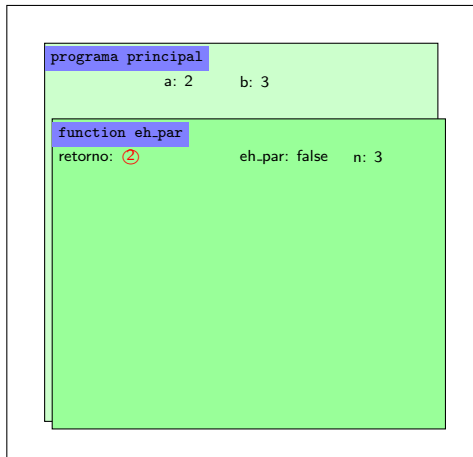


Diagrama de Execução

```
program imprime_dois_pares_final;  
var a, b: integer;  
  
function eh_par (n: integer): boolean;  
begin  
  eh_par:= true;  
  if n mod 2 <> 0 then  
    eh_par:= false  
end;  
  
begin (* programa principal *)  
  read (a, b);  
  while (a <> 0) or (b <> 0) do  
    begin (*true*      (*false*)  
      if eh_par (a) and eh_par (b) then  
        writeln (a, ' ', b);  
      read (a, b);  
    end;  
  end.  
1. coloca o valor de eh_par no chamador  
2. desvia fluxo para o local indicado em retorno  
3. retira folha
```

programa principal

a: 2 b: 3

Diagrama de Execução

```
program imprime_dois_pares_final;
var a, b: integer;

function eh_par (n: integer): boolean;
begin
  eh_par:= true;
  if n mod 2 <> 0 then
    eh_par:= false
  end;
end;

begin (* programa principal *)
  read (a, b);
  while (a <> 0) or (b <> 0) do
  begin
    if eh_par (a) and eh_par (b) then
      writeln (a, ' ', b);
      read (a, b);
    end;
  end.

usuário digita 0,0
```

programa principal

a: 0 b: 0

Diagrama de Execução

```
program imprime_dois_pares_final;  
var a, b: integer;  
  
function eh_par (n: integer): boolean;  
begin  
  eh_par:= true;  
  if n mod 2 <> 0 then  
    eh_par:= false  
end;  
  
begin (* programa principal *)  
  read (a, b);  
  while (a <> 0) or (b <> 0) do  
  begin  
    if eh_par (a) and eh_par (b) then  
      writeln (a, ' ', b);  
    read (a, b);  
  end;  
end.
```

programa principal

a: 0 b: 0

Diagrama de Execução

```
program imprime_dois_pares_final;  
var a, b: integer;  
  
function eh_par (n: integer): boolean;  
begin  
  eh_par:= true;  
  if n mod 2 <> 0 then  
    eh_par:= false  
end;  
  
begin (* programa principal *)  
  read (a, b);  
  while (a <> 0) or (b <> 0) do  
    begin  
      if eh_par (a) and eh_par (b) then  
        writeln (a, ' ', b);  
      read (a, b);  
    end;  
  end;  
end.
```

programa principal

a: 0 b: 0

Diagrama de Execução

```
program imprime_dois_pares_final;
var a, b: integer;

function eh_par (n: integer): boolean;
begin
  eh_par:= true;
  if n mod 2 <> 0 then
    eh_par:= false
  end;
end;

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

programa principal

a: 0 b: 0

- Por estes motivos, o nome do identificador do parâmetro na função não importa
- O que importa é o tipo do parâmetro, que deve ser o mesmo do tipo da variável do programa que chamou a função.

Diagrama de Execução

```
program imprime_dois_pares_final;
var a, b: integer;

function eh_par (a: integer): boolean;
begin
    eh_par:= true;
    if a mod 2 <> 0 then
        eh_par:= false
    end;
end;

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

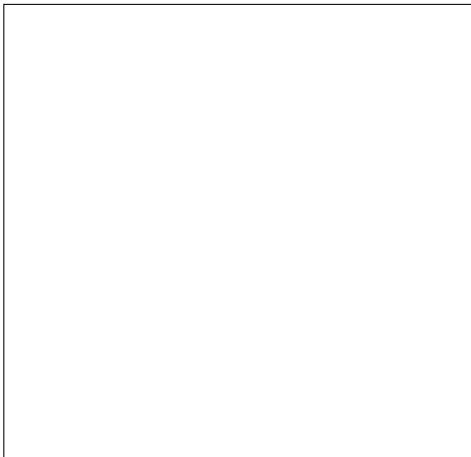
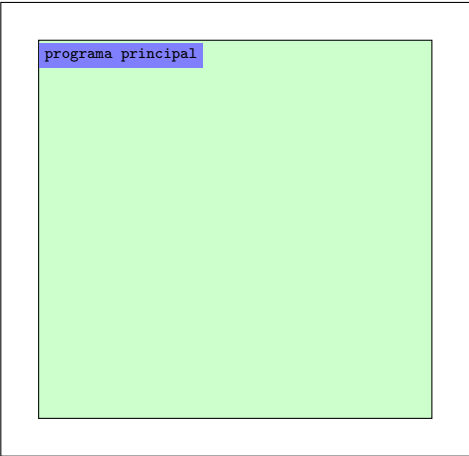


Diagrama de Execução

```
program imprime_dois_pares_final;  
var a, b: integer;  
  
function eh_par (a: integer): boolean;  
begin  
  eh_par:= true;  
  if a mod 2 <> 0 then  
    eh_par:= false  
end;  
  
begin (* programa principal *)  
  read (a, b);  
  while (a <> 0) or (b <> 0) do  
    begin  
      if eh_par (a) and eh_par (b) then  
        writeln (a, ' ', b);  
      read (a, b);  
    end;  
end.
```

Inicia folha de papel do
programa principal



programa principal

Diagrama de Execução

```
program imprime_dois_pares_final;  
var a, b: integer;  
  
function eh_par (a: integer): boolean;  
begin  
    eh_par:= true;  
    if a mod 2 <> 0 then  
        eh_par:= false  
    end;  
end;  
  
begin (* programa principal *)  
    read (a, b);  
    while (a <> 0) or (b <> 0) do  
        begin  
            if eh_par (a) and eh_par (b) then  
                writeln (a, ' ', b);  
            read (a, b);  
        end;  
    end;  
end.
```

Cria espaço para as variáveis
do programa principal

programa principal

a: ? b: ?

Diagrama de Execução

```
program imprime_dois_pares_final;  
var a, b: integer;  
  
function eh_par (a: integer): boolean;  
begin  
  eh_par:= true;  
  if a mod 2 <> 0 then  
    eh_par:= false  
end;  
  
begin (* programa principal *)  
  read (a, b);  
  while (a <> 0) or (b <> 0) do  
  begin  
    if eh_par (a) and eh_par (b) then  
      writeln (a, ' ', b);  
    read (a, b);  
  end;  
end.
```

programa principal

a: ? b: ?

Diagrama de Execução

```
program imprime_dois_pares_final;  
var a, b: integer;  
  
function eh_par (a: integer): boolean;  
begin  
  eh_par:= true;  
  if a mod 2 <> 0 then  
    eh_par:= false  
end;  
  
begin (* programa principal *)  
  read (a, b);  
  while (a <> 0) or (b <> 0) do  
  begin  
    if eh_par (a) and eh_par (b) then  
      writeln (a, ' ', b);  
    read (a, b);  
  end;  
end.  
  
usuário digita 2,3
```

programa principal

a: 2 b: 3

Diagrama de Execução

```
program imprime_dois_pares_final;  
var a, b: integer;  
  
function eh_par (a: integer): boolean;  
begin  
  eh_par:= true;  
  if a mod 2 <> 0 then  
    eh_par:= false  
end;  
  
begin (* programa principal *)  
  read (a, b);  
  while (a <> 0) or (b <> 0) do  
  begin  
    if eh_par (a) and eh_par (b) then  
      writeln (a, ' ', b);  
    read (a, b);  
  end;  
end.
```

programa principal

a: 2 b: 3

Diagrama de Execução

```
program imprime_dois_pares_final;  
var a, b: integer;  
  
function eh_par (a: integer): boolean;  
begin  
  eh_par:= true;  
  if a mod 2 <> 0 then  
    eh_par:= false  
end;  
  
begin (* programa principal *)  
  read (a, b);  
  while (a <> 0) or (b <> 0) do  
    begin  
      if eh_par (a) and eh_par (b) then  
        writeln (a, ' ', b);  
      read (a, b);  
    end;  
end.
```

1. Desvia fluxo
2. Cria nova folha
3. Aloca variáveis: eh_par, a:=cópia de a
4. Indica onde retornar ao finalizar

programa principal

a: 2 b: 3

Diagrama de Execução

```
program imprime_dois_pares_final;  
var a, b: integer;  
  
function eh_par (a: integer): boolean;  
begin  
  eh_par:= true;  
  if a mod 2 <> 0 then  
    eh_par:= false  
end;  
  
begin (* programa principal *)  
  read (a, b);  
  while (a <> 0) or (b <> 0) do  
  begin  
    if eh_par (a) ⌚ and eh_par (b) then  
      writeln (a, ' ', b);  
    read (a, b);  
  end;  
end.
```

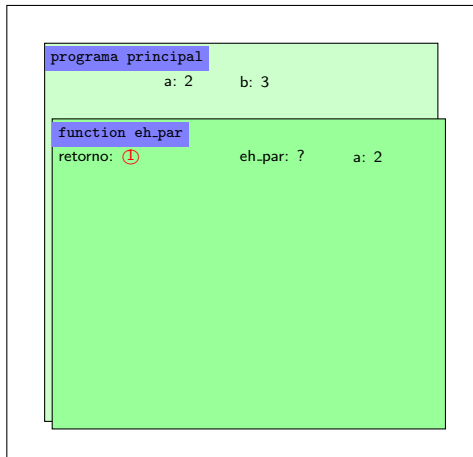


Diagrama de Execução

```
program imprime_dois_pares_final;  
var a, b: integer;  
  
function eh_par (a: integer): boolean;  
begin  
    eh_par:= true;  
    if a mod 2 <> 0 then  
        eh_par:= false  
    end;  
end;  
  
begin (* programa principal *)  
    read (a, b);  
    while (a <> 0) or (b <> 0) do  
        begin  
            if eh_par (a) ⌚ and eh_par (b) then  
                writeln (a, ' ', b);  
            read (a, b);  
        end;  
    end;  
end.
```

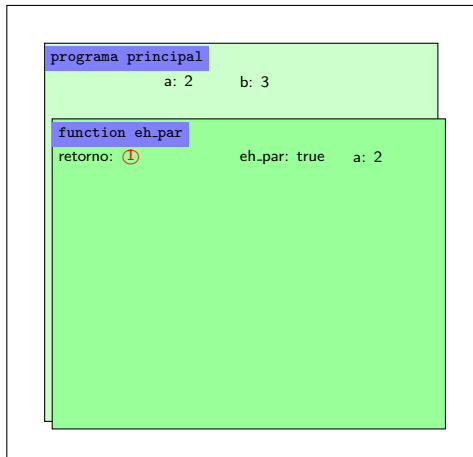


Diagrama de Execução

```
program imprime_dois_pares_final;  
var a, b: integer;  
  
function eh_par (a: integer): boolean;  
begin  
  eh_par:= true;  
  if a mod 2 <> 0 then  
    eh_par:= false  
end;  
  
begin (* programa principal *)  
  read (a, b);  
  while (a <> 0) or (b <> 0) do  
  begin  
    if eh_par (a) ⌚ and eh_par (b) then  
      writeln (a, ' ', b);  
    read (a, b);  
  end;  
end.
```

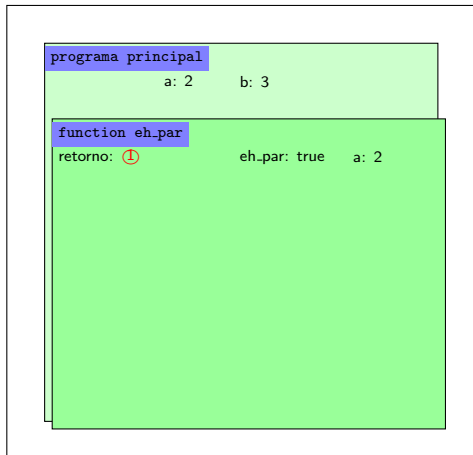


Diagrama de Execução

```
program imprime_dois_pares_final;  
var a, b: integer;  
  
function eh_par (a: integer): boolean;  
begin  
  eh_par:= true;  
  if a mod 2 <> 0 then  
    eh_par:= false  
end;  
  
begin (* programa principal *)  
  read (a, b);  
  while (a <> 0) or (b <> 0) do  
  begin  
    if eh_par (a) ⌚ and eh_par (b) then  
      writeln (a, ' ', b);  
    read (a, b);  
  end;  
end.
```

1. coloca o valor de eh_par no chamador
2. desvia fluxo para o local indicado em retorno
3. retira folha

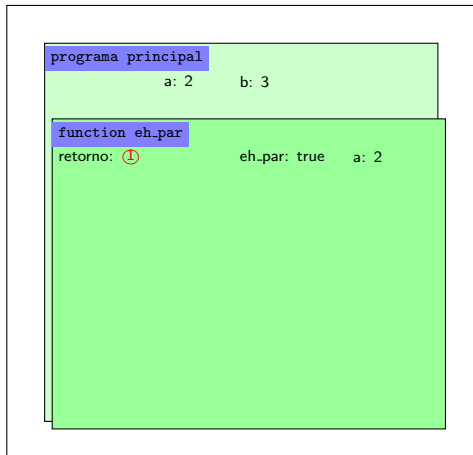


Diagrama de Execução

```
program imprime_dois_pares_final;  
var a, b: integer;  
  
function eh_par (a: integer): boolean;  
begin  
  eh_par:= true;  
  if a mod 2 <> 0 then  
    eh_par:= false  
end;  
  
begin (* programa principal *)  
  read (a, b);  
  while (a <> 0) or (b <> 0) do  
    begin (*true*)  
      if eh_par (a) and eh_par (b) then  
        writeln (a, ' ', b);  
      read (a, b);  
    end;  
end.
```

1. Desvia fluxo
2. Cria nova folha
3. Aloca variáveis: eh_par, a:=cópia de b
4. Indica onde retornar ao finalizar

programa principal

a: 2 b: 3

Diagrama de Execução

```
program imprime_dois_pares_final;  
var a, b: integer;  
  
function eh_par (a: integer): boolean;  
begin  
    eh_par:= true;  
    if a mod 2 <> 0 then  
        eh_par:= false  
    end;  
end;  
  
begin (* programa principal *)  
    read (a, b);  
    while (a <> 0) or (b <> 0) do  
        begin (*true*)  
            if eh_par (a) and eh_par (b) then  
                writeln (a, ' ', b);  
            read (a, b);  
        end;  
    end;  
end.
```

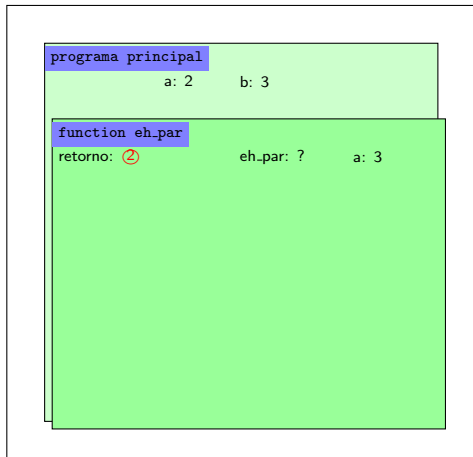


Diagrama de Execução

```
program imprime_dois_pares_final;  
var a, b: integer;  
  
function eh_par (a: integer): boolean;  
begin  
    eh_par:= true;  
    if a mod 2 <> 0 then  
        eh_par:= false  
    end;  
end;  
  
begin (* programa principal *)  
    read (a, b);  
    while (a <> 0) or (b <> 0) do  
        begin (*true*)  
            if eh_par (a) and eh_par (b) then  
                writeln (a, ' ', b);  
            read (a, b);  
        end;  
    end;  
end.
```

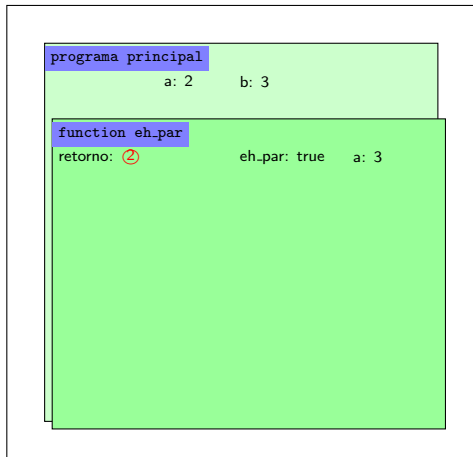


Diagrama de Execução

```
program imprime_dois_pares_final;  
var a, b: integer;  
  
function eh_par (a: integer): boolean;  
begin  
  eh_par:= true;  
  if a mod 2 <> 0 then  
    eh_par:= false  
end;  
  
begin (* programa principal *)  
  read (a, b);  
  while (a <> 0) or (b <> 0) do  
    begin (*true*)  
      if eh_par (a) and eh_par (b) then  
        writeln (a, ' ', b);  
      read (a, b);  
    end;  
  end.  
end.
```

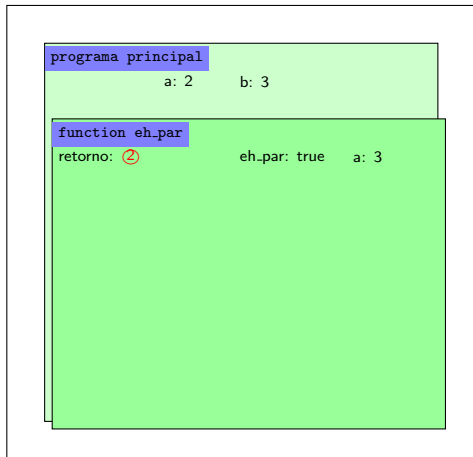


Diagrama de Execução

```
program imprime_dois_pares_final;  
var a, b: integer;  
  
function eh_par (a: integer): boolean;  
begin  
  eh_par:= true;  
  if a mod 2 <> 0 then  
    eh_par:= false  
end;  
  
begin (* programa principal *)  
  read (a, b);  
  while (a <> 0) or (b <> 0) do  
    begin (*true*)  
      if eh_par (a) and eh_par (b) then  
        writeln (a, ' ', b);  
      read (a, b);  
    end;  
  end.  
end.
```

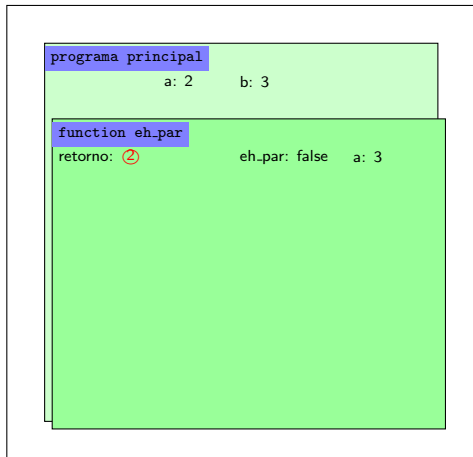


Diagrama de Execução

```
program imprime_dois_pares_final;
var a, b: integer;

function eh_par (a: integer): boolean;
begin
  eh_par:= true;
  if a mod 2 <> 0 then
    eh_par:= false
end;

begin (* programa principal *)
  read (a, b);
  while (a <> 0) or (b <> 0) do
    begin (*true*)
      if eh_par (a) and eh_par (b) then
        writeln (a, ' ', b);
      read (a, b);
    end;
  end.
1. coloca o valor de eh_par no chamador
2. desvia fluxo para o local indicado em retorno
3. retira folha
```

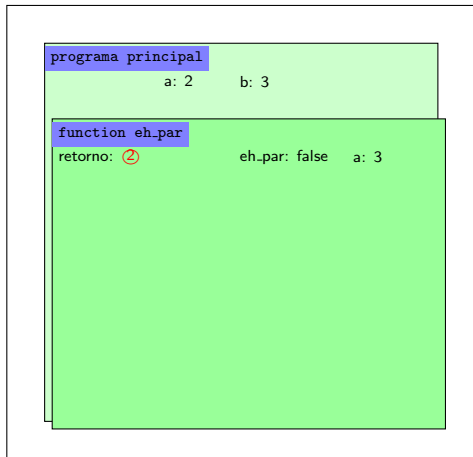


Diagrama de Execução

```
program imprime_dois_pares_final;
var a, b: integer;

function eh_par (a: integer): boolean;
begin
  eh_par:= true;
  if a mod 2 <> 0 then
    eh_par:= false
end;

begin (* programa principal *)
  read (a, b);
  while (a <> 0) or (b <> 0) do
    begin (*true*)      (*false*)
      if eh_par (a) and eh_par (b) then
        writeln (a, ' ', b);
      read (a, b);
    end;
  end.
1. coloca o valor de eh_par no chamador
2. desvia fluxo para o local indicado em retorno
3. retira folha
```

programa principal

a: 2 b: 3

Diagrama de Execução

```
program imprime_dois_pares_final;  
var a, b: integer;  
  
function eh_par (a: integer): boolean;  
begin  
  eh_par:= true;  
  if a mod 2 <> 0 then  
    eh_par:= false  
end;  
  
begin (* programa principal *)  
  read (a, b);  
  while (a <> 0) or (b <> 0) do  
  begin  
    if eh_par (a) and eh_par (b) then  
      writeln (a, ' ', b);  
      read (a, b);  
    end;  
  end.  
  
usuário digita 0,0
```

programa principal

a: 0 b: 0

Diagrama de Execução

```
program imprime_dois_pares_final;  
var a, b: integer;  
  
function eh_par (a: integer): boolean;  
begin  
  eh_par:= true;  
  if a mod 2 <> 0 then  
    eh_par:= false  
end;  
  
begin (* programa principal *)  
  read (a, b);  
  while (a <> 0) or (b <> 0) do  
  begin  
    if eh_par (a) and eh_par (b) then  
      writeln (a, ' ', b);  
    read (a, b);  
  end;  
end.
```

programa principal

a: 0 b: 0

Diagrama de Execução

```
program imprime_dois_pares_final;
var a, b: integer;

function eh_par (a: integer): boolean;
begin
  eh_par:= true;
  if a mod 2 <> 0 then
    eh_par:= false
  end;
end;

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

programa principal

a: 0 b: 0

Diagrama de Execução

```
program imprime_dois_pares_final;
var a, b: integer;

function eh_par (a: integer): boolean;
begin
  eh_par:= true;
  if a mod 2 <> 0 then
    eh_par:= false
  end;
end;

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

programa principal

a: 0 b: 0

- É possível definir quantas funções quisermos
- Por exemplo:
 - `function eh_par (n: integer): boolean;`
retorna true se n é par
 - `function media (a, b: integer): integer;`
retorna a média aritmética entre a e b

Exemplo

```
1 program imprime_dois_pares_final;  
2 var a, b: integer;  
3  
4 (* funcao que calcula se o parametro n eh par *)  
5 function eh_par (n: integer): boolean;  
6 begin  
7     eh_par:= true;  
8     if n mod 2  $\diamond$  0 then  
9         eh_par:= false  
10 end;  
11  
12 (* funcao que retorna a media aritmetica de dois inteiros *)  
13 function media (n, m: integer): integer;  
14 begin  
15     media:= (n + m) div 2;  
16 end;  
17  
18 begin (* programa principal *)  
19     read (a, b);  
20     if eh_par (a) and eh_par (b) then  
21         writeln ('media= ', media(a, b));  
22 end.
```

Os cálculos são feitos em cópias!

- Como os cálculos são feitos em cópias, pode-se alterar o valor sem que haja reflexo no programa que ativou a função

Exemplo

```
1  program contando_digitos;  
2  var a, cont: longint;  
3  
4  (* funcao que retorna quantos digitos n possui *)  
5  function num_digitos (n: longint): longint;  
6  begin  
7      cont:= 0;          (* variavel global *)  
8      while n > 0 do  
9          begin  
10         n:= n div 10;  
11         cont:= cont + 1;  
12     end;  
13     num_digitos:= cont;  
14 end;  
15  
16 begin (* programa principal *)  
17     read (a);  
18     writeln (a, ' possui ', num_digitos (a), ' digitos');  
19     writeln (a); (* a nao teve seu valor alterado *)  
20 end.
```

- Normalmente, as funções devem ter poucas linhas de código
- É desejável que caiba em uma tela, sem precisar rolar a tela para ver todo o código
- Assim, cada pedaço fica bem definido e o código se torna de fácil compreensão
- Se o código é muito longo, pense em quebrá-lo em outras funções pequenas

- este material está no livro no capítulo 8, seção 8.2.5

- Slides feitos em \LaTeX usando beamer
- Licença

Creative Commons Atribuição-Uso Não-Comercial-Vedada a Criação de Obras Derivadas 2.5 Brasil License.<http://creativecommons.org/licenses/by-nc-nd/2.5/br/>

Creative Commons Atribuição-Uso Não-Comercial-Vedada a Criação de Obras Derivadas 2.5 Brasil License.<http://creativecommons.org/licenses/by-nc-nd/2.5/br/>