

Âmbito, funções e gestão de memória

**Linguagens de Programação
2018.2019**

Teresa Gonçalves
tcg@uevora.pt

Departamento de Informática, ECT-UÉ

Sumário

Linguagens estruturadas em blocos

Blocos em linha

Funções

Funções de ordem superior

Linguagens estruturadas em blocos

Bloco

O que é?

Região de texto do programa

Como se identifica?

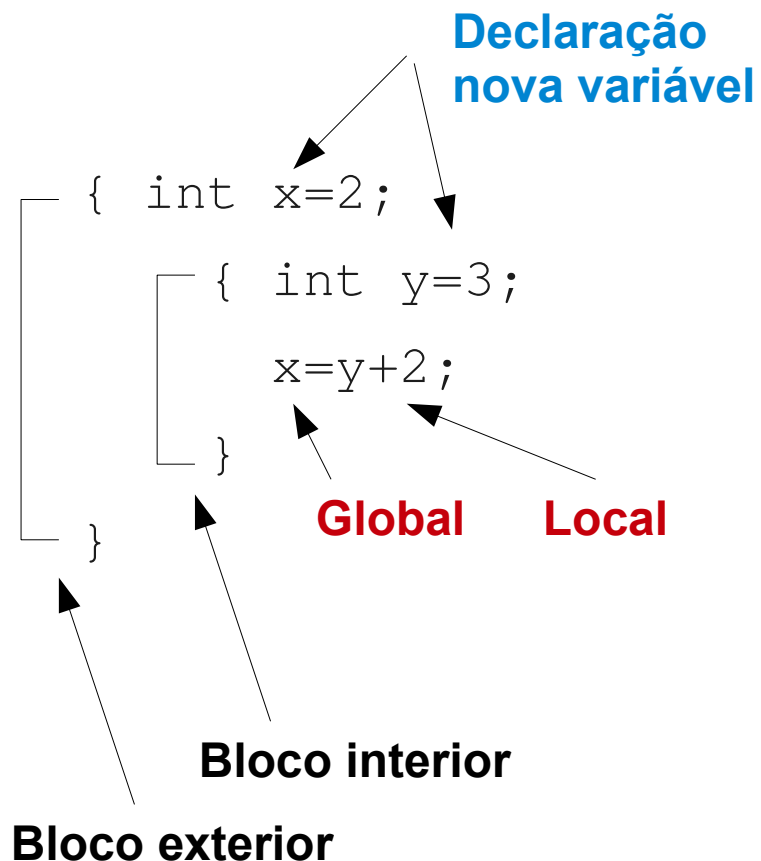
Marcadores de início e fim

O que contém?

Declarações locais à região

Instruções

Exemplo



Linguagens estruturadas em blocos

Blocos em linha

C

{ ... }

Pascal

begin ... end

ML

let ... in ... end

Funções / procedimentos

Associados à execução

Propriedades

Declaração de variáveis em diferentes pontos do programa

Declaração visível numa certa região - bloco

Os blocos podem ser aninhados, mas não se sobrepõem parcialmente

Execução das instruções dum bloco

No início é **alocada** memória para as variáveis declaradas nesse bloco

No final (alguma) essa memória pode ser libertada

Um identificador não declarado no bloco é global

refere a entidade declarada no bloco mais próximo

Conceitos básicos

Âmbito

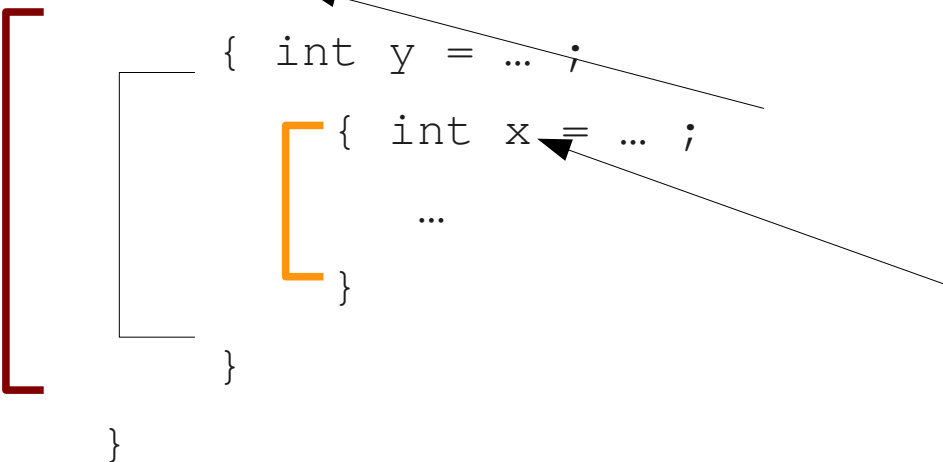
Região do texto do programa onde a declaração é visível

Tempo de vida

Período de tempo onde a localização está acessível ao programa

Exemplo

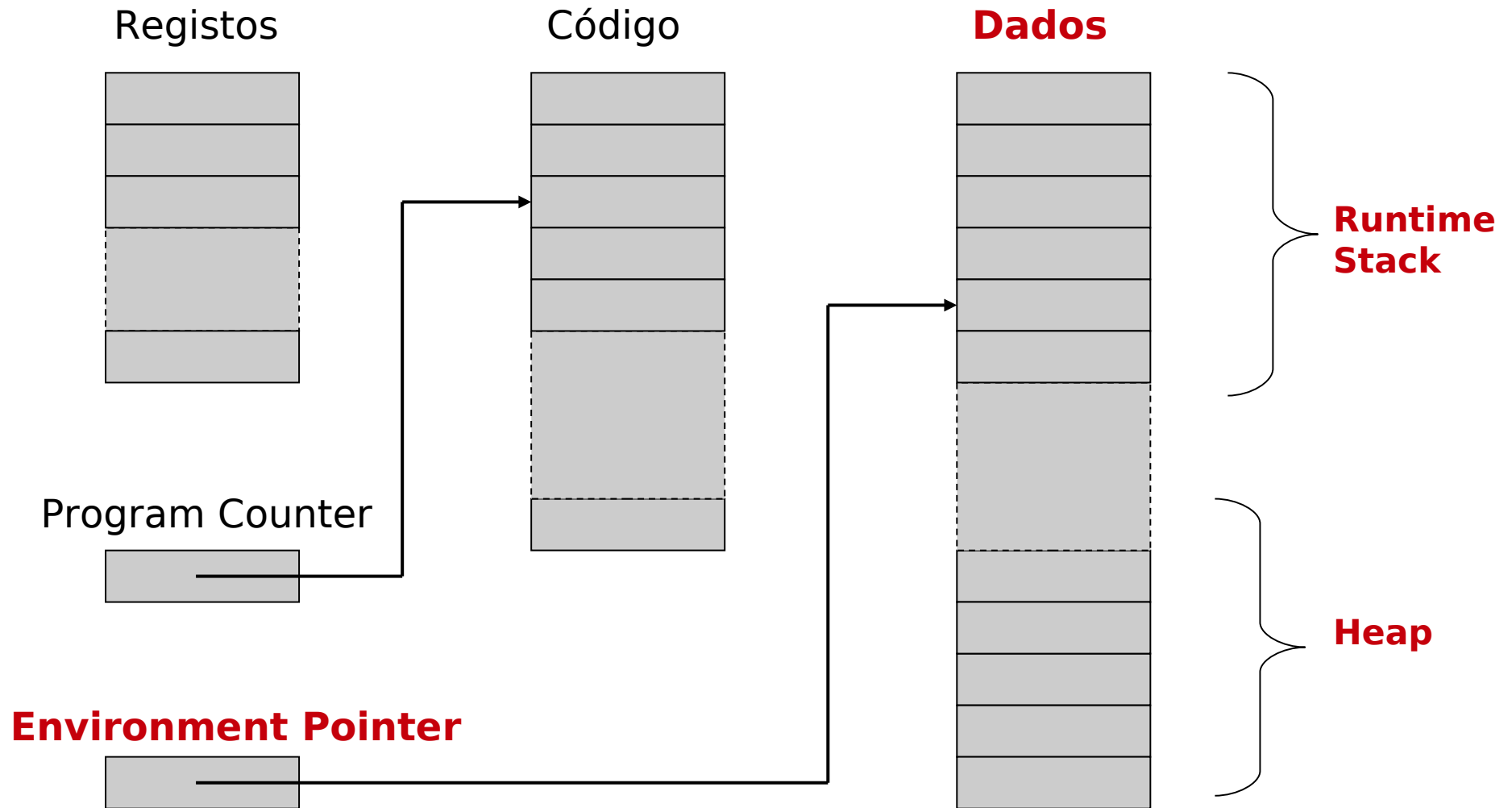
```
{ int x = ... ;  
  { int y = ... ;  
    { int x = ... ;  
      ...  
    }  
  }  
}
```



Tempo de vida de **x** **inclui** o tempo de execução do bloco interior

Esta declaração de **x** **esconde** a variável x exterior → **buraco de âmbito**

Modelo simplificado da máquina



Gestão de memória

Segmento de dados

Stack contém informação relativa à entrada/saída dos blocos

Para cada bloco existe um registo de ativação

Heap contém informação com tempo de vida variável

Apontador de ambiente

Aponta para o registo de ativação corrente

Entrada no bloco → adição de um novo registo de ativação

Saída do bloco → remoção do registo de ativação mais recente

Variáveis e parâmetros

Variáveis locais

Mantidas no registo de ativação associado ao bloco

Variáveis globais

São declaradas noutro bloco → estão num registo de ativação criado anteriormente ao bloco corrente

Parâmetros de funções e procedimentos

Mantidos no registo de ativação associado ao bloco

Registo de ativação

Estrutura de dados guardada no stack de execução

Conj. de localizações de memória para guardar informação local ao bloco

Também designado por *stack-frame*

A informação mantida depende do tipo de bloco

Blocos em linha

Funções e procedimentos

Funções de ordem superior

Variáveis locais e globais

Variável local

Variável declarada no bloco atual

Variável global

Variável declarada fora do bloco atual

O acesso envolve encontrar o RA “certo” no stack

Exemplo

x e y locais no bloco exterior

z local no bloco interior

x e y globais no bloco interior

```
{ int x=0;  
  int y=x+1;  
  { int z=(x+y) * (x-y);  
  }  
}
```

Blocos em linha

Blocos em linha

Registo de ativação

Espaço para variáveis locais

Espaço adicional para valores intermédios (se necessário)

Exemplo

```
{ int x=0;  
  int y=x+1;  
  { int z=(x+y) * (x-y) ;  
  }  
}
```

Push registo com espaço para x, y
Atribui valores a x e y

Push registo para bloco interior
Atribui valor a z

Pop registo do bloco interior

Pop registo do bloco exterior

RA para blocos em linha

Control link

Apontador para o RA anterior no stack

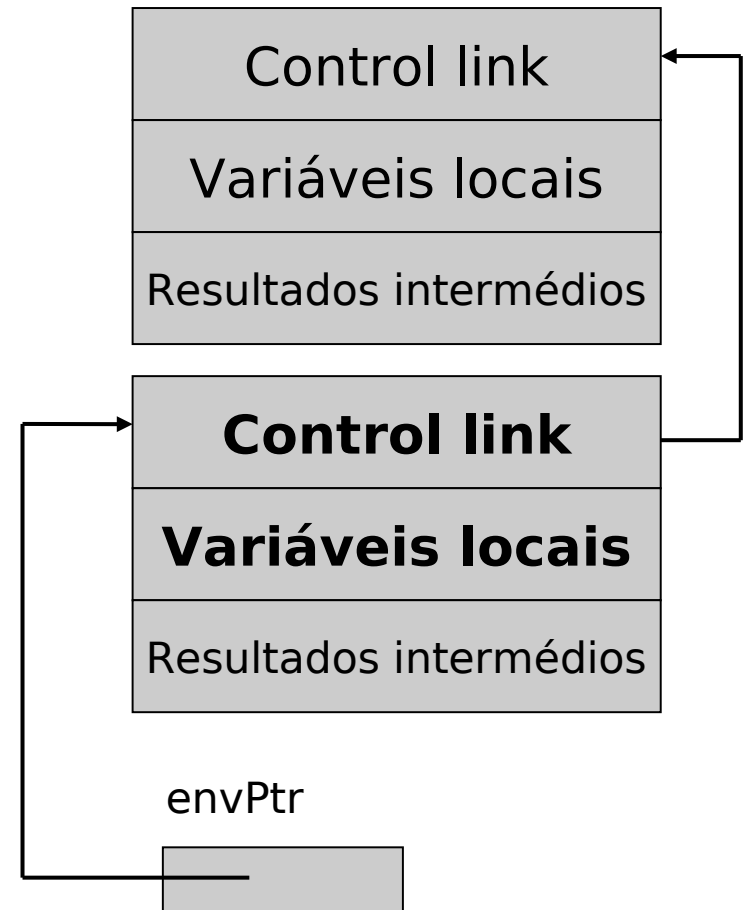
Push

Control link corrente = EnvPtr

EnvPtr = novo registo

Pop

EnvPtr é restabelecido utilizando o control link corrente



Exemplo

```
{ int x=0;  
  int y=x+1;  
  { int z=(x+y) * (x-y);  
  }  
}
```

Push registo com espaço para x, y

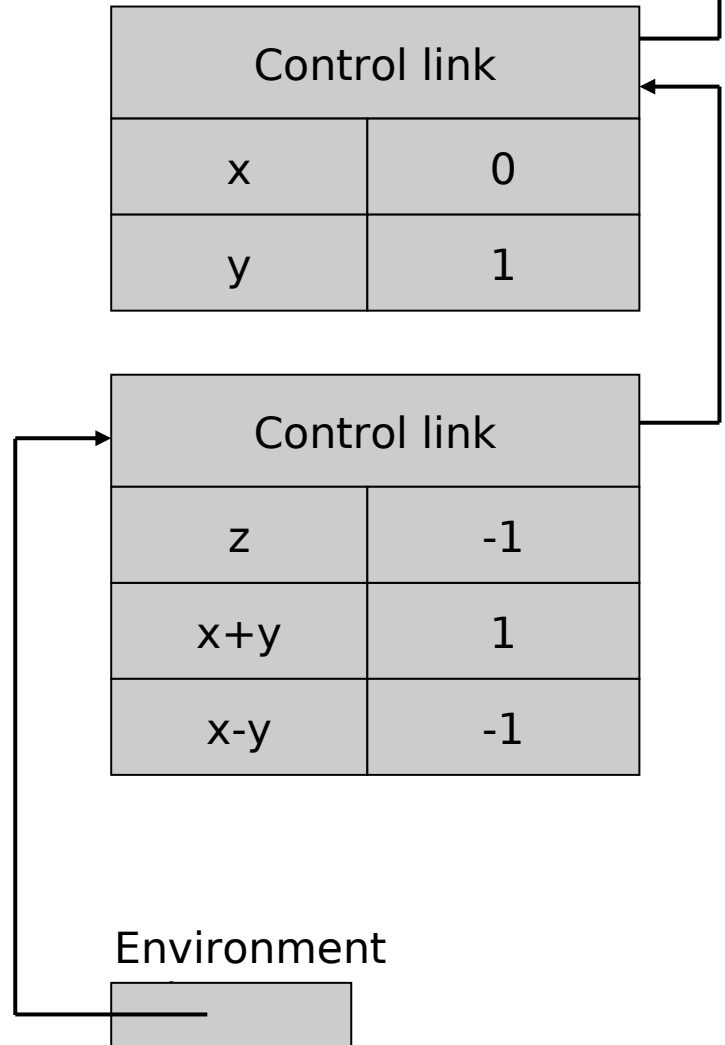
Atribui valores a x e y

Push registo para bloco interior

Atribui valor a z

Pop registo do bloco interior

Pop registo do bloco exterior



Funções e procedimentos

Funções e procedimentos

Sintaxe

```
<tipo> function f(<pars>)  
{  
    <vars locais>;  
    <corpo função>;  
}
```

Registro de ativação

Endereço de retorno

Endereço para valor de retorno da função

Parâmetros

Variáveis locais

Resultados intermédios (+ valor de retorno)

RA para funções

End. retorno

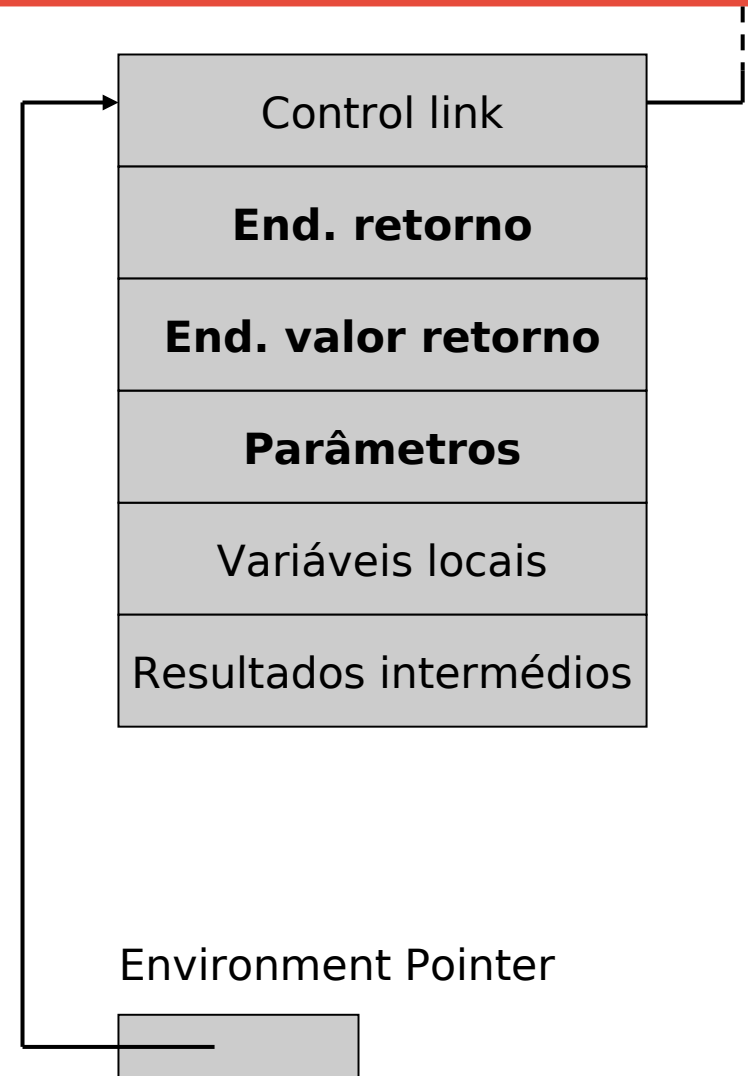
Endereço do código a executar depois do retorno da função

End. valor retorno

Endereço no registo ativação do bloco chamador para receber valor de retorno

Parâmetros

Posições para dados do bloco chamador



Exemplo

Função

```
fact(n) = if n<=1  
         then 1  
         else n*fact(n-1)
```

End. valor retorno

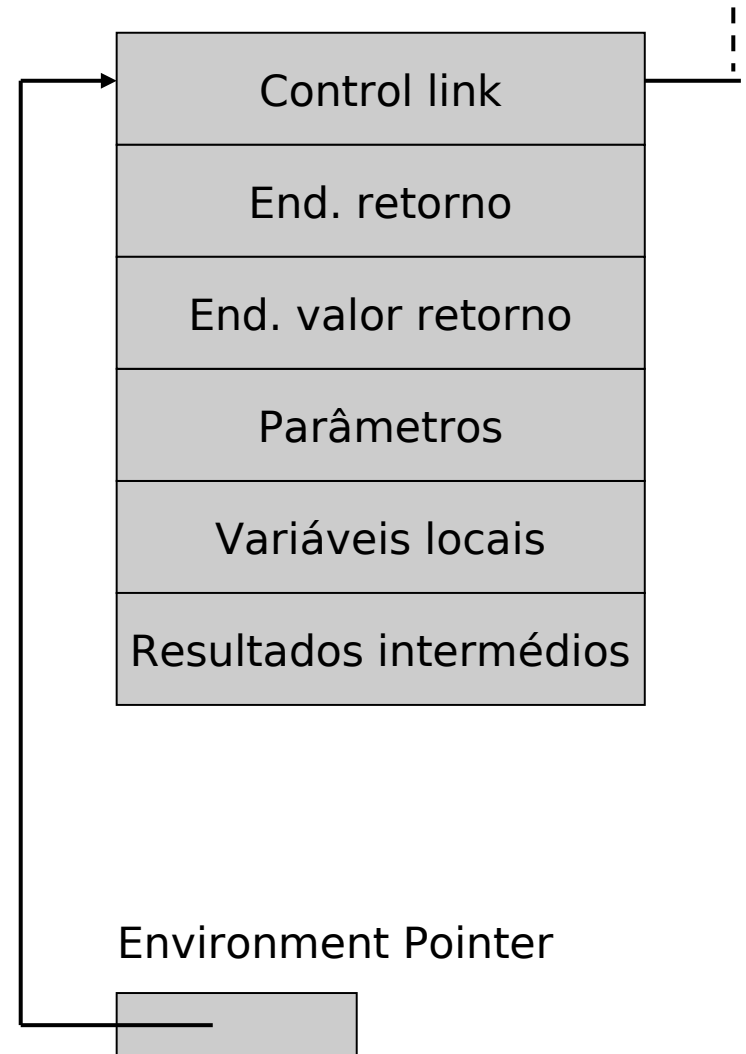
Localização para colocar fact(n)

Parâmetros

Valor de n

Resultado intermédio

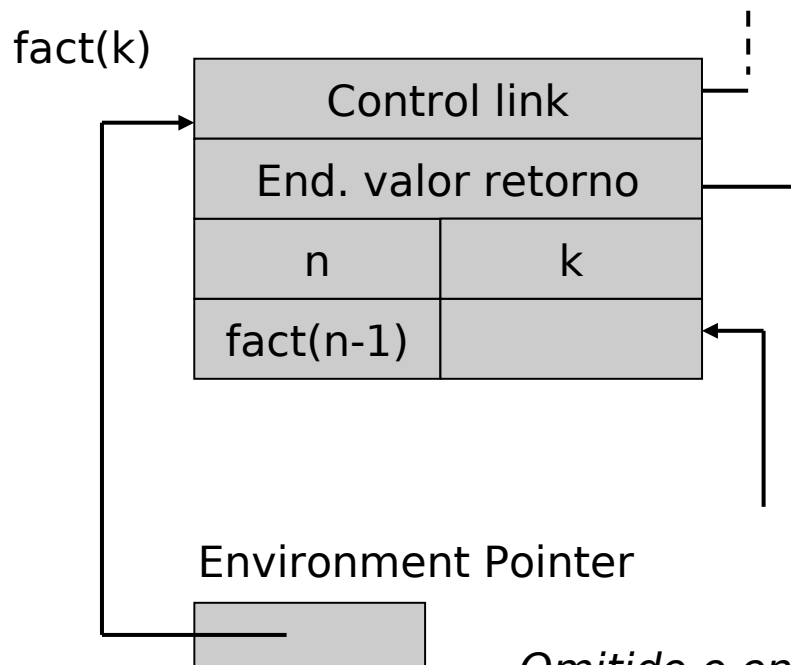
Local para fact(n-1)



Chamada de função

1 + fact(3)

```
fact(n) = if n<=1  
then 1  
else n*fact(n-1)
```



*Omitido o endereço de retorno
Aponta para segmento de código!*

fact(3)

Control link	
End. valor retorno	
n	3
fact(n-1)	

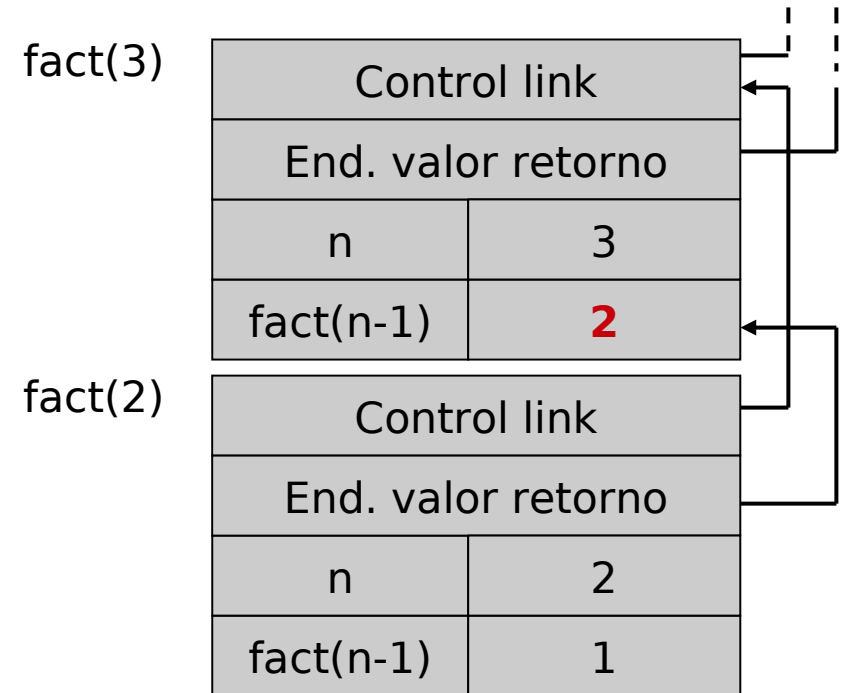
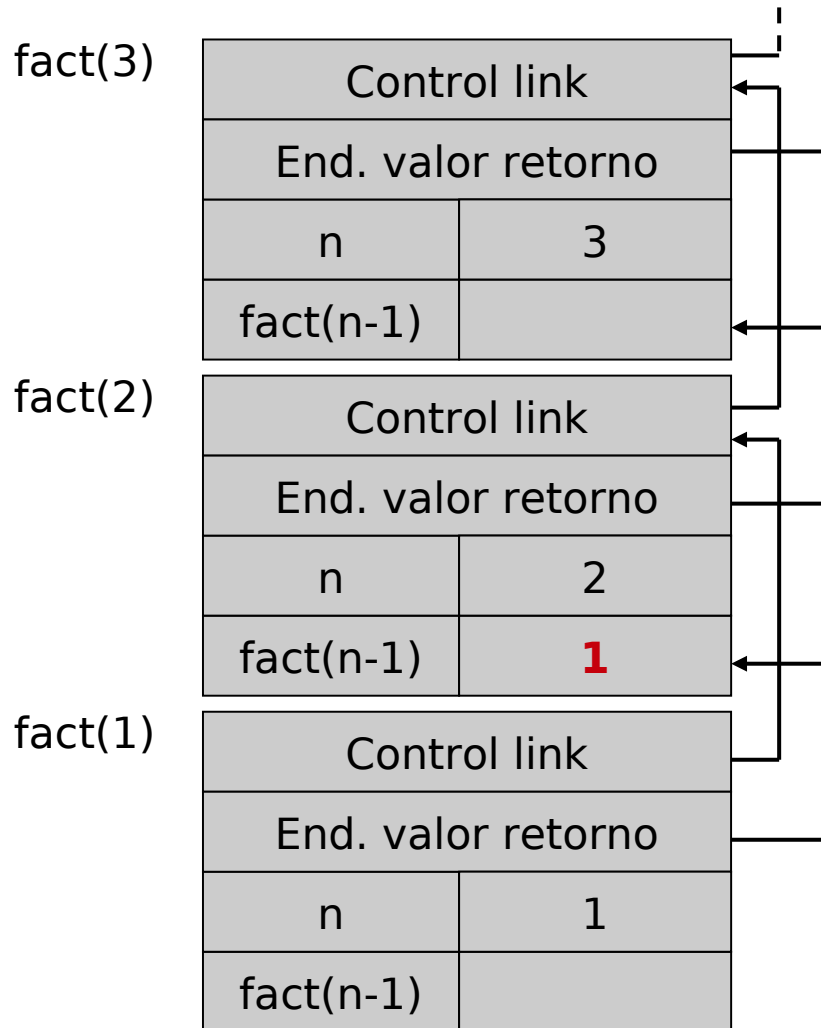
fact(2)

Control link	
End. valor retorno	
n	2
fact(n-1)	

fact(1)

Control link	
End. valor retorno	
n	1
fact(n-1)	

Retorno da função



```
fact(n) = if n <= 1
          then 1
          else n * fact(n-1)
```

Parâmetro formal e atual

Parâmetro formal

Nome utilizado na declaração da função

Parâmetro atual

Valor do parâmetro numa chamada de função

Exemplo

```
proc p ( int x, int y )  
  if (x>y) then ... else ...;  
  ...  
  x := y*2+3;  
  ...  
}  
p ( z, 4*z+1 ) ;
```

The diagram illustrates the relationship between formal parameters and actual parameters. In the function declaration, the formal parameters are `x` and `y`, which are circled in red. In the function call, the actual parameters are `z` and `4*z+1`, which are also circled in red. Arrows point from the actual parameters to the formal parameters: one arrow from `z` to `x`, and another from `4*z+1` to `y`.

Parâmetros formais

Parâmetros atuais

R-value e L-value

Atribuição

$x := y + 3$

L-value

Refere **localização** da variável

R-value

Refere **conteúdo** da variável

Passagem por referência e por valor

Passagem por valor

Coloca o R-value do parâmetro atual no RA

Características

- Função não pode alterar valor da variável passada

- Reduz a criação de pseudónimos

- Pode ser menos eficiente para grandes estruturas

Passagem por referência

Coloca o L-value do parâmetro atual no RA

Características

- Função pode alterar valor à variável que é passada

- Podem existir vários nomes para a mesma localização de memória

- Pode ser menos eficiente para pequenas estruturas

Exemplo

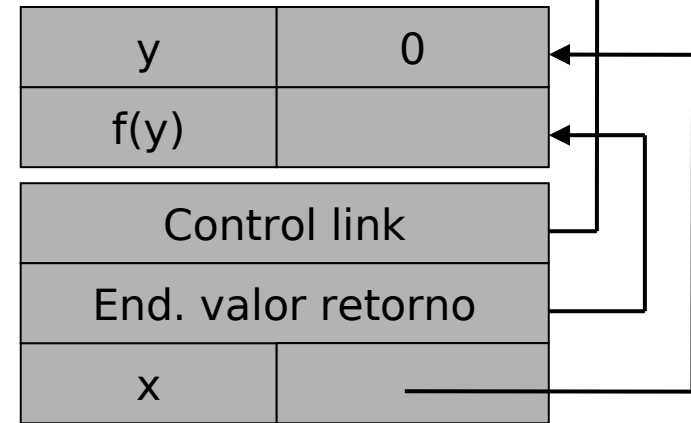
```
function f (x) =  
  { x = x+1;  
    return x;  
  }
```

```
var y = 0;
```

```
print (f(y)+y);
```

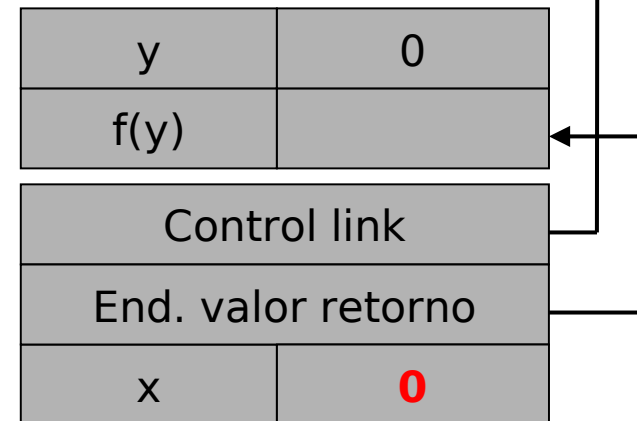
Passagem por
referência

f(y)



Passagem por
valor

f(y)



Outro exemplo

```
function f (pass-by-ref x, pass-by-value y)
begin
  x:=2;
  y:=1;
  if x=1 then return 1 else return 2;
end;
var z: int;
z:=0;
print f(z, z);
```

Qual o output?