Fundamentos de Programação de Computadores

Traçagens e testes
Funções

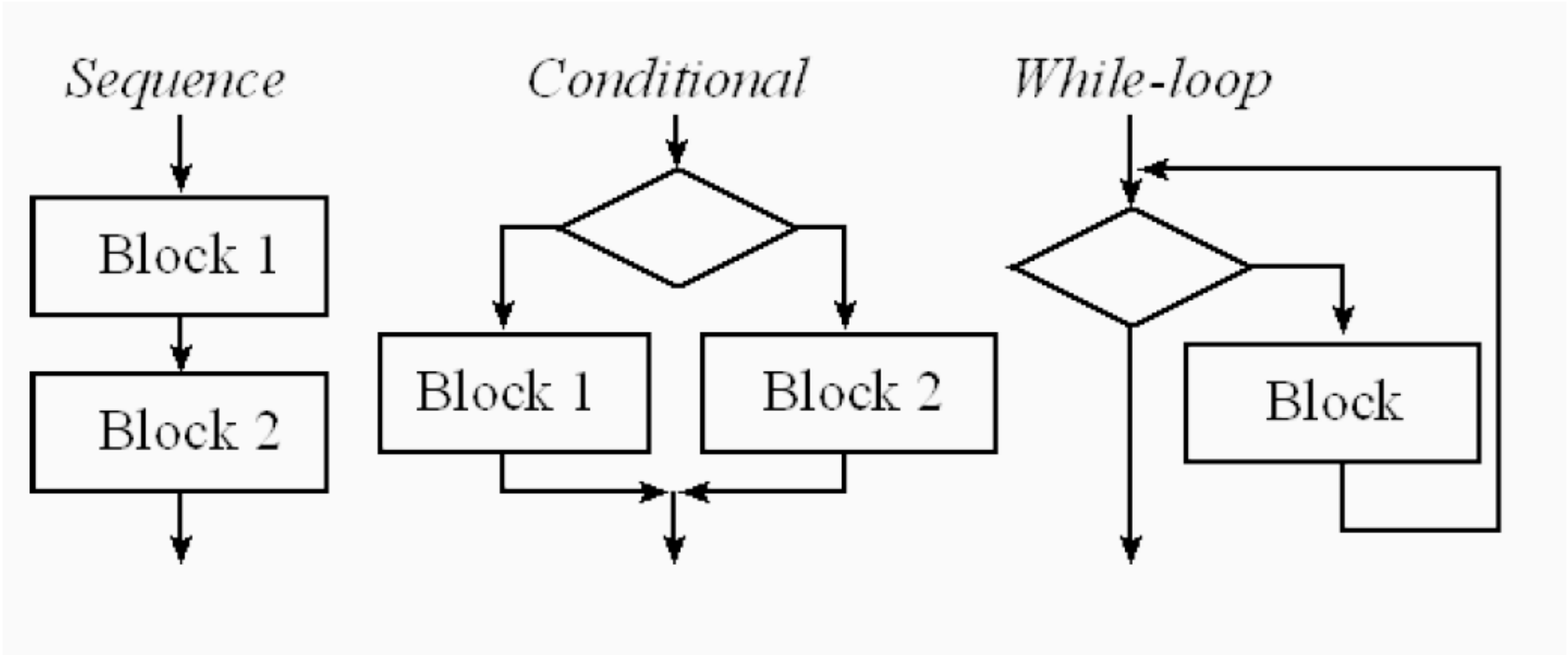
Docente: Fátima Leal

DCT DEPARTAMENTO CIÊNCIA E TECNOLOGIA

UPT UNIVERSIDADE PORTUCALENSE

Instruções Sequenciais, decisão e repetição

- **Relembrando…**



Traçagem e teste

- A **traçagem** consiste em **testar um algoritmo** para um **conjunto de valores** de entrada.
- Para isso vamos construir uma tabela que contemple:
  - Os passos do algoritmo em cada linha
  - As operações em cada coluna
  - Os resultados para cada passo utilizando um conjunto de valores
- A traçagem e teste é útil para analisar o funcionamento de um algoritmo

Traçagem e teste

- Consideremos agora o seguinte exemplo:

```
I ← 1
S ←0
enquanto I < 5  faça
        S ← S + I
        I ← I + 1
```

Como ficaria a tabela de traçagem ?

Traçagem e teste

Passo	I	S	enquanto I< 5	S+I	I+1
1	1	0			
2	1	0	1 < 5 ✓	1=0+1	2=1+1
3	2	1	2 < 5 ✓	3=1+2	3=2+1
4	3	3	3 < 5 ✓	6=3+3	4=3+1
5	4	6	4 < 5 ✓	10=6+4	5=4+1
6	5	10	5 < 5 ×		

Funções e Procedimentos

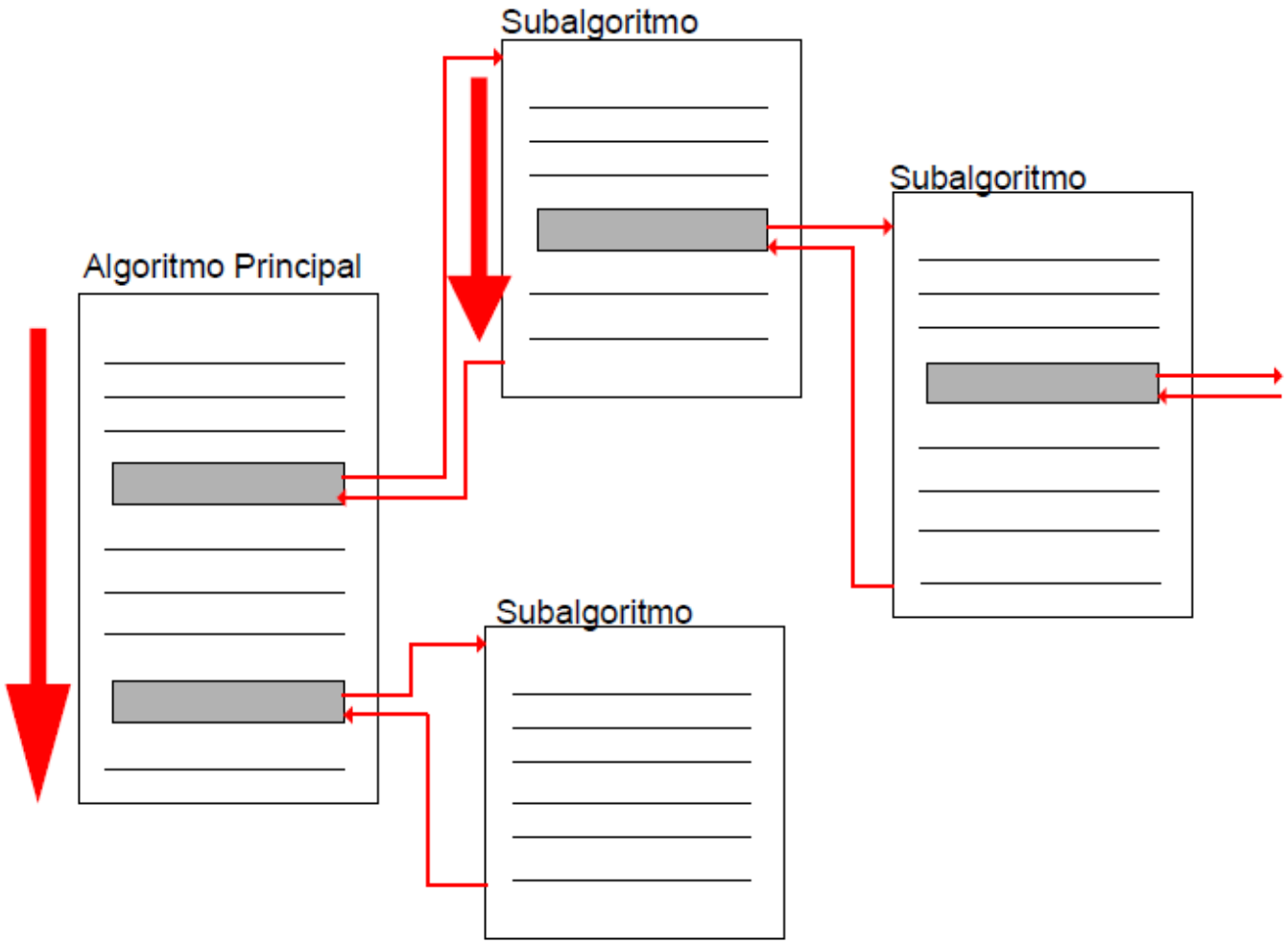
- Até agora, temos utilizado sequências lógicas de instruções para a construção de algoritmos
- Para melhorar a **eficiência dos algoritmos**, por vezes é necessário criar uma **divisão do algoritmo** principal **em sub-tarefas** que possam ser utilizadas sempre que seja necessário
- Estas **sub-tarefas** designam-se neste contexto de **sub-algoritmos ou módulos** que serão traduzidos em **funções e procedimentos**
- As **funções e procedimentos** são uma boa prática de programação
- **Facilita a leitura e manutenção** das aplicações

Funções e Procedimentos

- As funções e procedimentos:
  - Podem ser utilizadas dentro de expressões
  - Possuem um ou mais argumentos
  - Podem ter argumentos simples, expressões ou mesmo invocações a funções
  - **Alteram o fluxo do algoritmo**
  - Quando a **função é invocada** o controlo de execução passa para as **instruções que se encontram dentro da função**

  - Analisemos a seguinte imagem:

Funções e Procedimentos



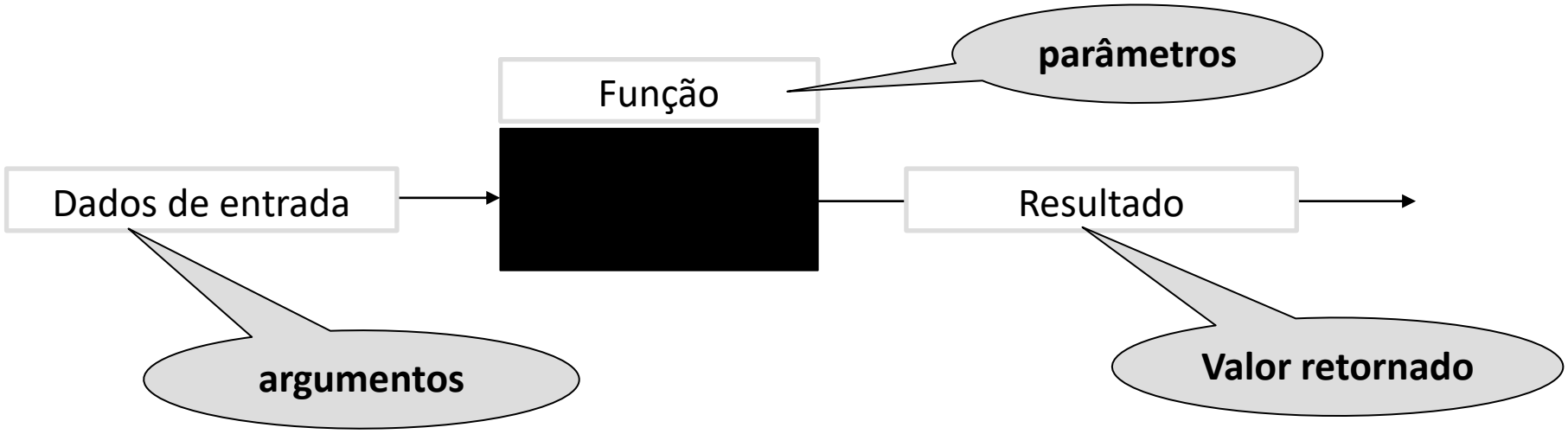
Funções

- **Aspetos importantes de funções**:
- **Correspondência** entre **argumentos** e os **parâmetros** da função
- As **variáveis** definidas na própria **função** são **locais** à função
- Quando a **função é invocada** deve carregar nos seus **argumentos** a mesma **quantidade e o mesmo tipo** de elementos que estão declarados nos **parâmetros da função**

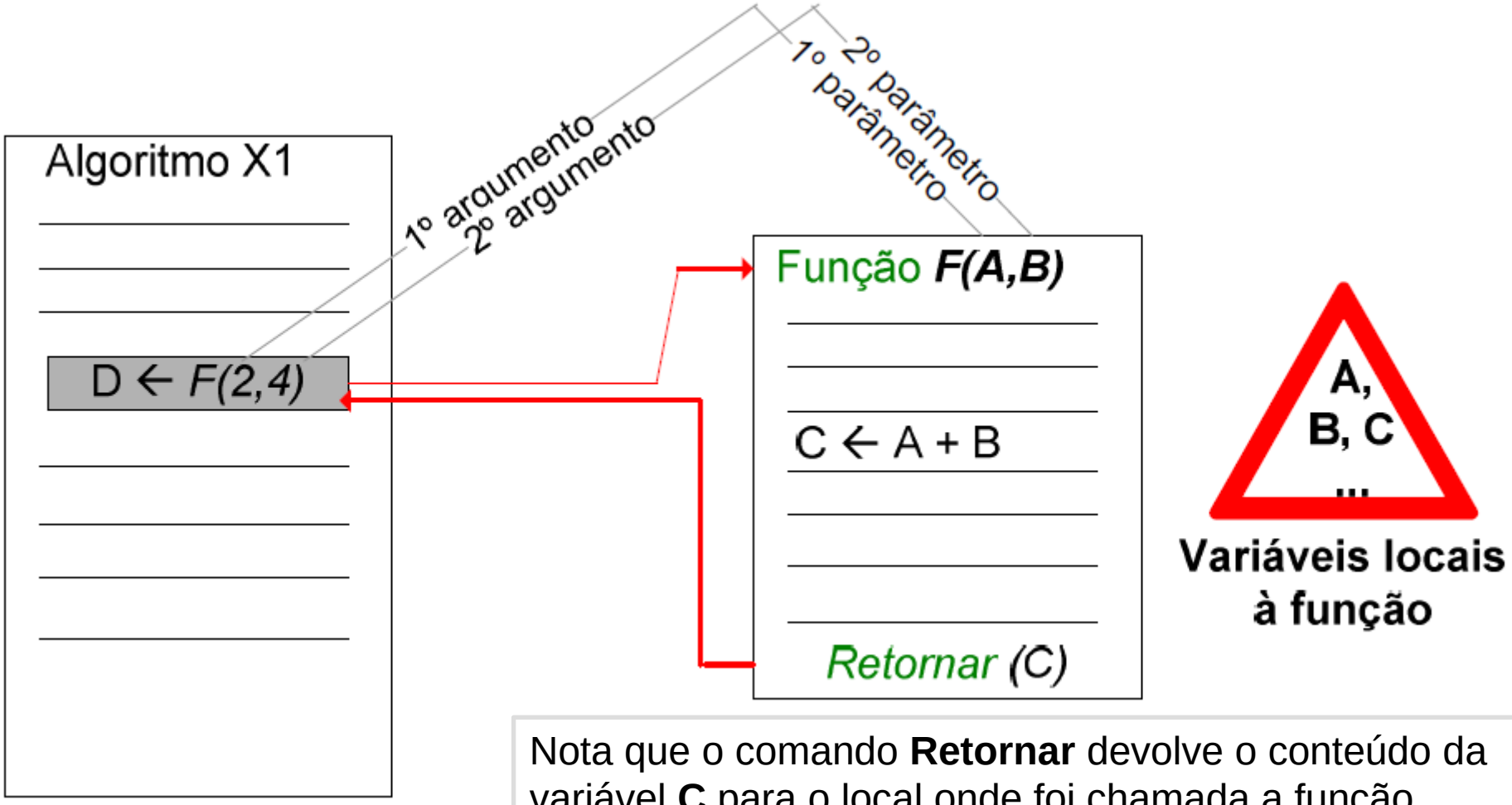
Funções

- Na programação, uma função tem um funcionamento similar a uma função matemática

- É semelhante a uma caixa preta que recebe valores e retorna uma resultado



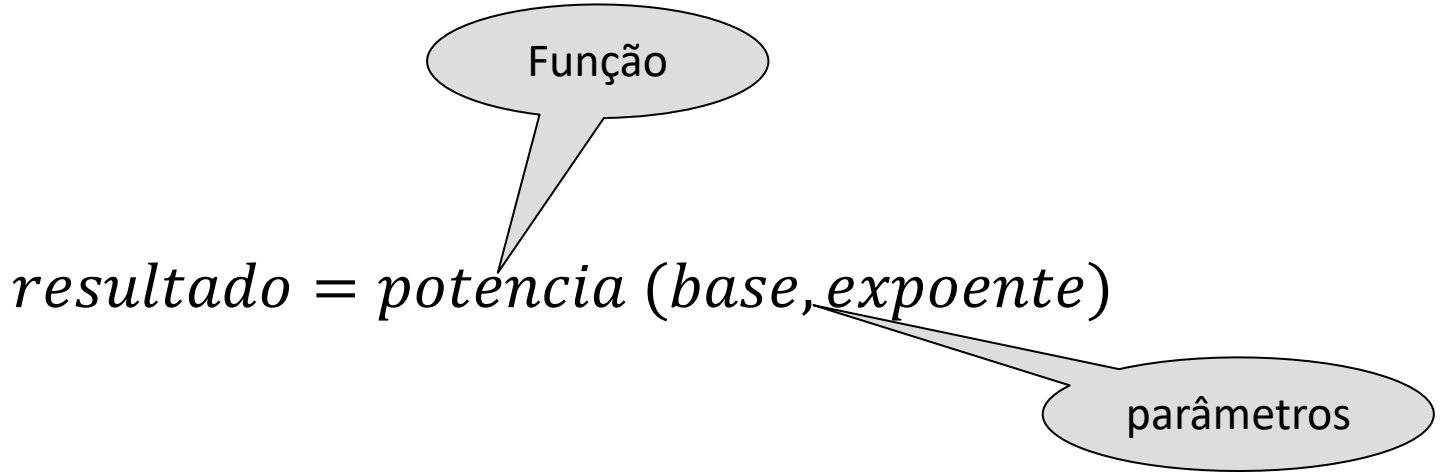
Funções



Nota que o comando **Retornar** devolve o conteúdo da variável **C** para o local onde foi chamada a função.

Funções

- Exemplo…



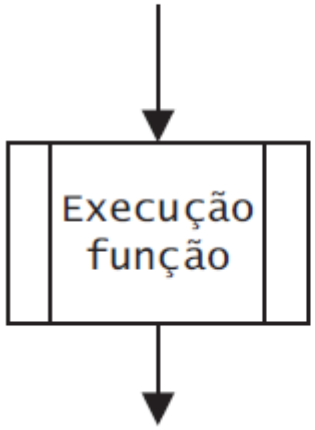
- A **função** é identificada por um **nome**, sendo a **lista de parâmetros** constituída por **zero ou mais variáveis** passadas à função.

- As **variáveis definidas nas funções** serão **variáveis locais**. As **variáveis do algoritmo principal** são chamadas de **globais**

- A função irá executar uma lista particular de comandos retornando no final um resultado

Funções

- No fluxograma e em pseudocódigo:




```
Função nomeFuncao(listaParametros)
inicio
...
Lista de comandos
Retornar
fim-função
```

Funções

```
Algoritmo função.
Var base, expoente, resultado: inteiro
Início
        Escrever (“Introduza a base:”)
        Ler (base)
        Escrever (“Introduza o expoente:”)
        Ler (expoente)

        resultado = potencia (base, expoente)

        Escreve (“O resultado é:”, resultado)

Fim
```

Valor a receber da função

Variáveis a enviar para a função

Parâmetros que a função recebe

```
Função potencia (base, expoente)
Var calculo: inteiro
Início
        calculo = base ^ expoente
        Retornar (calculo)
Fim_função
```

Valor que se irá retornar

Funções

Versão com ciclo **para**

```
Para i de 1 até expoente faça
        calculo = calculo * base
fim_para
```

```
Algoritmo função.
Var base, expoente, resultado: inteiro
Início
        Escrever (“Introduza a base:”)
        Ler (base)
        Escrever (“Introduza o expoente:”)
        Ler (expoente)

        resultado = potencia (base, expoente)

        Escrever (“O resultado é:”, resultado)

Fim
```

```
Função potencia (base, expoente)
Var calculo, i: inteiro
Início
        i = 1
        calculo = 1
        Enquanto i<=expoente faça
                calculo = calculo * base
                i = i+1
        fim_enquanto
         Retornar (calculo)
Fim_função
```

Funções

- Vamos proceder à **traçagem** da função do exemplo anterior com o **ciclo enquanto** utilizando como teste a **base 2** e **expoente 3**

	i	**base**	**expoente**	**calculo**	**i ≤ expoente**	**calculo = calculo * base**
Passo 1	1	2	3	1		calculo = 1*2
Passo 2	2	2	3	2		calculo = 2 * 2
Passo 3	3	2	3	4		calculo = 4 * 2
Passo 4	4	2	3	8		
A função vai retornar: calculo = 8						

Exercício

O algoritmo seguinte utiliza uma função? Escreva o pseudocódigo correspondente.

```
Algoritmo Principal.
Var x, y, maior: inteiro
Início
      Escrever (“Introduza o valor 1:”)
      Ler (x)
      Escrever (“Introduza o valor 2:” )
      Ler (y)
      Se x != y então
            maior = Maior (x, y)
            Escrever (“O valor maior é:”, maior )
      Senão
            Escrever (“Valores iguais.” )
      fim_se
Fim
```

```
Função Maior(x, y)
Var maior: inteiro
Início
      Se x > y então
         maior = x
      Senão
         maior = y

      Retornar (maior)
Fim_função
```

UNIVERSIDADE
PORTUCALENSE

Do conhecimento à prática.