Disciplina BCM0505-15
Processamento da Informação

Funções

Profa. Carla Negri Lintzmayer

carla.negri@ufabc.edu.br

http://professor.ufabc.edu.br/~carla.negri

Centro de Matemática, Computação e Cognição – Universidade Federal do ABC

Agenda

Funções ou módulos

Definição

Chamada

Pratique!

Funções ou módulos

Funções

Função

É uma estrutura que agrupa um conjunto de instruções/comandos, que são executados quando a função é **chamada** e possui um objetivo bem particular.

São importantes pois:

- nos permitem abstrair partes relevantes e/ou que se repetem mais de uma vez, evitando inconsistências e facilitando alterações;
- dividem a solução do problema em problemas menores;
  - cada um deles pode ser resolvido de forma independente e em momentos distintos;
  - cada um deles pode ser testado individualmente, facilitando a identificação e correção de erros;
- nos permitem **reaproveitar** algoritmos (seus ou de outros).

Funções

Já vimos duas funções!

- Leia() e Escreva() são funções: não sabemos como elas funcionam exatamente, mas podemos utilizá-las para receber e imprimir valores.
- Elas têm objetivos bem definidos:
  - Para Leia, nós colocamos dentro dos parênteses uma sequência de variáveis e o que ela faz é atribuir valores a essas variáveis.
  - Para Escreva, nós colocamos dentro dos parênteses uma sequência de strings e/ou variáveis e o que ela faz é imprimir esses valores na saída.
- Para nós, essas duas funções funcionam como **caixa preta**: suas estruturas são desconhecidas, mas sabemos quais são as entradas e as saídas esperadas.

Exemplo

1: INTEIRO: $a,\ b,\ x$
2: LEIA$(a,\ b)$
3: $x \leftarrow (a+b)/2 + |a-b|/2$
4: ESCREVA(x)

Exemplo

```
1:  Função Maior(Inteiro: x, Inteiro: y)
2:      Inteiro: z
3:      z ← (x + y)/2 + |y − x|/2
4:      Devolve z
5:
6:  Função Principal( )
7:      Inteiro: a, b, x
8:      Leia(a, b)
9:      x ← Maior(a, b)
10:     Escreva(x)
11:
12: Principal()
```

Funções

- Uma função têm dois "momentos" importantes: definição e chamada.
  - Com LEIA e ESCREVA, por exemplo, nós só fizemos *chamadas*.
- Na definição, precisamos indicar qual é o nome da função, quais dados ela deve receber e quais comandos ela faz para atingir seus objetivos.
- Na chamada, indicamos apenas o nome da função e passamos os dados requeridos de acordo com a definição.
- A definição, portanto, não tem *efeito*. É apenas na hora da chamada que os comandos são de fatos executados.

Definição

Definição de funções

Toda definição função deve obedecer a seguinte sintaxe:

```
1: Função NOME(TIPO_1: param_1, ..., TIPO_n: param_n)
2:     comando1
3:     comando2
4:     ...
5:     Devolve
```

onde:

- A linha 1 contém o **cabeçalho** da definição.
- As linhas abaixo do cabeçalho fazem parte do **corpo** da função e estão **indentadas** com relação ao cabeçalho: elas começam a ser escritas após alguns espaços.

Definição de funções - Cabeçalho

Função NOME(TIPO_1: $param_1$, ..., TIPO_n: $param_n$)

- **Função** é o indicador de que o que segue é a definição de uma função.
- NOME é o *identificador* da função, cujas regras de sintaxe seguem as mesmas regras dos identificadores de variáveis.
- Após o nome da função, deve haver (e) e, dentro deles, uma lista de **parâmetros**:
  - Cada $param_i$ é o identificador de uma variável que poderá ser utilizada no corpo da função, e deve ser precedida do seu tipo.
  - Uma função pode ter zero ou mais parâmetros.

Definição de funções - Corpo

- Os comandos do corpo da função podem ser quaisquer um dos que já vimos e ainda vamos ver.
- O comando **Devolve** indica o fim da função:
  - Muitas vezes as funções produzem resultados que precisam ser devolvidos a quem as chamou, caso em que **Devolve** é normalmente seguido de algum valor ou expressão.
  - Se a função não devolver nenhum valor, podemos omitir o **Devolve** ou deixá-lo sem nada à sua direita.

Devolver?

- Para de fato executar os comandos que escrevemos no corpo de uma função, precisamos **chamá-la**.
  - Uma chamada a uma função só pode ocorrer se ela for definida primeiro.
- Uma chamada de função tem a seguinte sintaxe:

```
1: NOME(arg_1, arg_2, ..., arg_n)
```

- Se a função NOME possui algum comando **Devolve** em seu corpo, ela irá devolver um valor, o qual substituirá o comando de chamada.
- Cada arg_i é um **argumento**, que é um valor ou expressão ou variável que tem o mesmo tipo que $param_i$, de acordo com a definição da função (a ordem é muito importante).

Parâmetros e argumentos

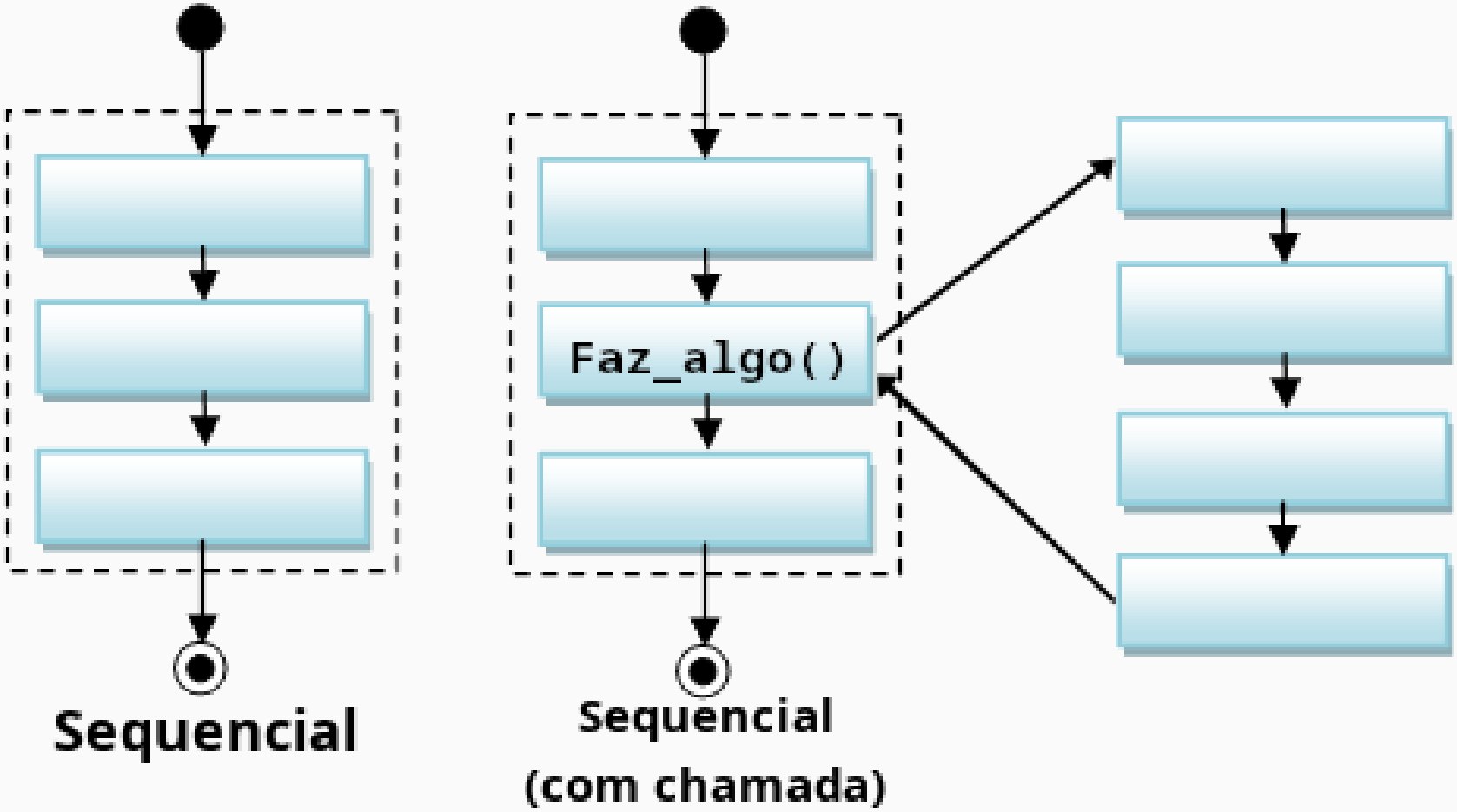
- Esses termos são usados para a mesma coisa: informação que é passada para a função.
- **Parâmetro** é a variável listada dentro dos parênteses na definição da função.
- **Argumento** é o valor que é enviado para a função quando ela é chamada.

Chamada

Chamadas a funções

- O **fluxo de execução** de um algoritmo, como vimos, é a ordem em que os comandos são executados: linha após linha, de cima para baixo.
- Se uma chamada a uma função é encontrada, então o fluxo de execução é desviado para a primeira linha do corpo da função:
  - Copia-se o valor de cada argumento, em ordem, para cada parâmetro.
  - Executa-se cada linha do corpo da função: de cima para baixo, uma após a outra.
  - Ao final da execução, o fluxo de execução retorna ao ponto exato onde a função foi chamada.
  - Se a função tinha um comando **Devolve** , então o valor que aparece em frente ao **Devolve** substitui o comando da chamada da função.

Fluxo de execução



Chamadas a funções

- É importante observar também que uma função pode chamar outra.
- Por isso, a partir de agora não faz mais sentido ler um algoritmo de cima para baixo.
- Deve-se seguir o fluxo de execução a partir do primeiro comando que efetivamente é executado.

Exemplo

```
1:  Função Maior(Inteiro: x, Inteiro: y)
2:      Inteiro: z
3:      z ← (x + y)/2 + |y − x|/2
4:      Devolve z
5:
6:  Função Principal( )
7:      Inteiro: a, b, x
8:      Leia(a, b)
9:      x ← Maior(a, b)
10:     Escreva(x)
11:
12: Principal()
```

comando						
1						
2						
3						
4						
5						
6						
7						
8						
9						
10						

Pratique!

Exercício 1

Faça um algoritmo que recebe 4 inteiros e apresenta o maior deles. Use funções para estruturar seu algoritmo.

Exercício 2

Neste problema, queremos resolver expressões da forma

$$\sum_{k=i}^{f} k\,.$$

Por exemplo, $\sum_{k=3}^{234} k = 27492$, $\sum_{k=2342}^{340928} k = 58113379745$.

Faça um algoritmo que recebe o valor de i e f, sendo $i \leq f$, e que apresenta o valor final da expressão.

Exercício 3

O problema de compressão de dados tem como objetivo reduzir o espaço ocupado por dados em algum dispositivo. Faça um algoritmo que recebe um número de 5 dígitos e o reduz em um número de 2 dígitos da seguinte forma: o primeiro e o último dígitos são somados, o segundo e o quarto dígitos são somados, esses dois são somados entre si e com o dígito central.