Funções (modularização)

Patrícia de Siqueira Ramos

UNIFAL-MG, *campus* Varginha

29 de Setembro de 2020

Funções

- As funções e procedimentos são blocos de instruções que realizam ações específicas (mostrar resultados ou calcular algo)
- As variáveis declaradas dentro de uma função só existirão enquanto a função estiver sendo executada, sendo chamadas **variáveis locais**
- As variáveis usadas normalmente são chamadas **variáveis globais** e existirão enquanto o programa estiver sendo executado
- Um exemplo de função muito utilizada é a `Escreva()`
  ela recebe como parâmetros o que desejamos imprimir na tela

Exemplo 1 - função que soma dois números

```
Função soma(a, b)
    Real: a, b, s
    s = a + b
    Retorne(s)
Fim
```

- variáveis locais: a, b e s

Exemplo 1 - função que soma dois números

```
Função soma(a, b)
    Real: a, b, s
    s = a + b
    Retorne(s)
Fim
```

- variáveis locais: a, b e s

Algoritmo que lê dois números e utiliza a função soma:

```
Início
    Real: x, y, w
    Leia(x, y)
    w = soma(x, y)
    Escreva('Soma:', w)
Fim
```

- variáveis globais: x, y e w

Exemplo 1 - função que soma dois números - Python

```python
def soma(a, b):
    s = a + b
    return s
```

Exemplo 1 - função que soma dois números - Python

```python
def soma(a, b):
    s = a + b
    return s
```

Código em Python que lê dois números e utiliza a função soma:

```python
x = float(input('x: '))
y = float(input('y: '))
w = soma(x, y)
print('Soma:', w)
```

Observações

- Uma função pode servir apenas para mostrar algo na tela
- Exemplo:

```
Função mostra(nome)
    String: nome
    Escreva('Oi,', nome)
Fim
```

Observações

- Uma função pode servir apenas para mostrar algo na tela
- Exemplo:

```
Função mostra(nome)
    String: nome
    Escreva('Oi,', nome)
Fim
```

Uso:

```
Início
    String: aluno
    Escreva('Insira seu nome:')
    Leia(aluno)
    mostra(aluno)
Fim
```

Diferença entre `return` e `print` no python

Observações sobre o uso de funções no `python`:

- Usamos `return` geralmente quando o resultado de uma função é numérico e queremos usar seu resultado em outra função:

```
def soma(a, b):
    s = a + b
    return s
```

- Usamos `print` geralmente quando queremos mostrar apenas valores na tela, sem usar em outra função:

```
def mostra(n):
    for i in range(n):
        print(i)
```

Observações

- Uma função pode não receber nenhum parâmetro
- Exemplo:

```
Função nada()
   Inteiro: x
   x = 3 * 4
   Retorne(x)
Fim
```

Essa função não tem graça nenhuma, ela sempre retorna o valor 12 porque ela foi definida assim.

Observações

- Uma função pode não receber nenhum parâmetro
- Exemplo:

```
Função nada()
   Inteiro: x
   x = 3 * 4
   Retorne(x)
Fim
```

Essa função não tem graça nenhuma, ela sempre retorna o valor 12 porque ela foi definida assim.
Uso:

```
Início
   Inteiro: num
   num = nada()
   Escreva(num)
Fim
```

Exemplo 2 - função que retorna a raiz cúbica de um número

```
Função raiz3(num)
    Real: num, r
    r = num ** (1 / 3)
    Retorne(r)
Fim
```

Exemplo 2 - função que retorna a raiz cúbica de um número

```
Função raiz3(num)
    Real: num, r
    r = num ** (1 / 3)
    Retorne(r)
Fim
```

Algoritmo que utiliza a função raiz cúbica:

```
Início
    Real: z, raiz
    Leia(z)
    raiz = raiz3(z)
    Escreva('Raiz cúbica é', raiz)
Fim
```

Exercício 1

Elabore uma função que recebe um número `num` e o grau `n` da raiz. Tal função deve se chamar `raizn(num, n)`.
Por exemplo:

- `raizn(9, 2)` deve retornar 3
- `raizn(27, 3)` deve retornar 3
- `raizn(256, 4)` deve retornar 4

Também faça um algoritmo que use tal função.

Exercício 1 - raizn(num, n)

```
Função raizn(num, n)
    Real: num, r
    Inteiro: n
    r = num ** (1 / n)
    Retorne(r)
Fim
```

Algoritmo que utiliza a função:

```
Início
    Real: a, raiz
    Inteiro: b
    Escreva('Número:')
    Leia(a)
    Escreva('Grau da raiz:')
    Leia(b)
    raiz = raizn(a, b)
    Escreva('Resultado da raiz', raiz)
Fim
```

Exemplo 3 - função que retorna o maior de dois números

```
Função maior2(a, b)
    Real: a, b
    Se a > b Então
        Retorne(a)
    Senão
        Retorne(b)
    FimSe
Fim
```

Algoritmo que utiliza a função maior2:

```
Início
    Real: n1, n2, m
    Leia(n1, n2)
    m = maior2(n1, n2)
    Escreva('Maior:', m)
Fim
```

Exercício 2

Faça uma função que retorna o maior de 3 números passados como parâmetros.

Exercício 2 - maior3(a, b, c)

```
Função maior3(a, b, c)
    Real: a, b, c
    Se a > b Então
        Se a > c Então
            Retorne(a)
        Senão
            Retorne(c)
        FimSe
    Senão
        Se b > c Então
            Retorne(b)
        Senão
            Retorne(c)
        FimSe
    FimSe
Fim
```

(continua)

Exercício 2 - algoritmo que usa `maior3(a, b, c)`

Algoritmo que utiliza a função:

```
Início
    Real: x, y, z, m
    Escreva('Insira três números:')
    Leia(x, y, z)
    m = maior3(x, y, z)
    Escreva('Maior:', m)
Fim
```

Exercício 3

Faça uma função chamada `divi` que receba um número e retorne quantos divisores esse número possui. Exemplo:
`divi(5)` deve retornar 2
`divi(10)` deve retornar 4

Exercício 3 - divi(n)

```
Função divi(n)
    Inteiro: n, div, i
    div = 0
    Para i de 1 até n faça
        Se n % i == 0 Então
            div = div + 1
        FimSe
    FimPara
    Retorne(div)
Fim
```

Testar no Python:
divi(7) deve retornar 2
divi(8) deve retornar 4

Exercício 4

Faça uma função chamada `primo` que receba um número e retorne se o número é primo ou não. Utilize a função `divi` para calcular o número de divisores do número. Exemplo:
`primo(5)` deve retornar `primo`
`primo(10)` deve retornar `não primo`

Exercício 4 - primo(x)

```
Função primo(x)
    Inteiro: x, d
    d = divi(x)
    Se d == 2 Então
        Escreva('primo')
    Senão
        Escreva('não primo')
    FimSe
Fim
```

Uso da função:

```
Início
    Inteiro: y
    Leia(y)
    primo(y)
Fim
```