# **Funções**

### **Objetivos**

Definição

Exemplos

Necessidade e benefícios na utilização de funções

Passagem de parâmetros para funções (esquema de alocação de memória)

Vetores e funções

## **Funções**

Funções são blocos de programas que retornam um valor.

```
x := sqrt(y);
linha := FloatToStr(x);
y := 4 * abs(x);
```

## **Funções**

O bloco de comandos deve conter pelo menos uma instrução que atribui o valor de retorno da função.

#### Sintaxe:



## Funções – Exemplo

```
função noPatas (entradas: N1, N2)
inicio
  resultado \leftarrow 4*(N1 + N2)
fim
programa teste
inicio
 caes \leftarrow 4
 gatos \leftarrow 3
 imprima 'Total de patas: ', noPatas(caes, gatos)
fim
                                                cederj
```

#### A função noPatas em Pascal

```
function noPatas(N1, N2 : integer): integer;
begin
   result := 4*(N1+N2);
end;
var
   caes, gatos : integer;
begin
   caes := 4;
   gatos := 3;
   writeln('Total de patas: ',
            noPatas(caes, gatos));
end.
```

#### A função noPatas em Java

```
public class Patas {
  public static int noPatas(int N1, int N2) {
    return 4*(N1+N2);
  public static void main(String args[]) {
     int caes = 4;
     int gatos = 3;
     System.out.println("Total de patas: " +
      noPatas (caes, gatos));
```

## **Funções**

Mas é necessário que eu use funções em meus programas?



Sim e Não. ?!?!?!?!?!?!?

# **Funções**

É necessário que eu use funções em meus programas?

Sim, se as funções foram escritas por algum outro programador!

Se você está escrevendo seus próprios módulos de software (procedimentos ou funções), o uso de funções não é obrigatório, ainda que ele apresente algumas vantagens.

<u>cederj</u>

#### noPatas como um Procedimento

```
procedimento noPatas (entradas: N1, N2
                         saidas: patas)
inicio
  patas \leftarrow 4*(N1 + N2)
fim
programa teste
inicio
  caes \leftarrow 4
  gatos \leftarrow 3
  noPatas (caes, gatos, pes)
  imprima 'Total de patas: ', pes
fim
```



## **Funções**

O uso de funções traz, no entanto, algumas vantagens:

É possível utilizar a chamada da função diretamente em expressões:

```
imprima 'Total de patas: ', noPatas (caes, gatos)
```

#### ao invés de:

```
noPatas(caes, gatos, pes)
imprima 'Total de patas: ', pes
```



## Vantagens na utilização de funções

Além disso, em construções do tipo:

$$y \leftarrow raiz (x)$$

é mais fácil perceber que y é a raiz de x do que na construção:

onde é impossível dizer, sem conhecer a função, se y é raiz de x ou x é raiz de y cederj

#### Outro exemplo: A função converte

```
função Converte (entradas: Fer)
inicio
  resultado \leftarrow (Fer - 32) * 5 / 9
fim
programa teste
inicio
  imprima 'Fahrenheit: '
  leia Far
  Cel ← Converte (Far)
  imprima 'Celsius: ', Cel
fim
```



# A função converte

#### Saida:

Fahrenheit: 212

Celsius: 100.00



### A função Converte em Pascal

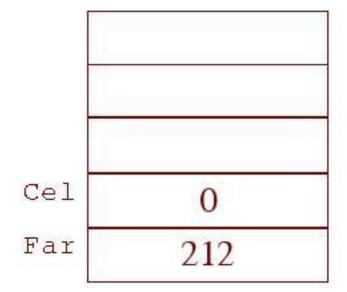
```
function converte (Fer : real): real;
begin
   result := (Fer - 32) * 5 / 9;
end;
var
   Far, Cel : real;
begin
   write ('Fahrenheit: ');
   readln(Far);
   Cel := converte(Far);
   writeln('Celsius: ', Cel:0:2);
end.
```

#### A função converte em Java

```
public class IO extends EasyIn {
  public static float converte(float Fer) {
      return (Fer - 32) * 5 / 9;
  public static void main(String args[]) {
      float Far, Cel;
      System.out.print("Fahrenheit: ");
      Far = getFloat();
      Cel = converte(Far);
      System.out.println("Celsius: " + Cel);
```

```
var
  Far, Cel: real;
                 Cel
                 Far
                                     cederj
```

```
readln(Far);
```



```
// chamada da função
Cel := converte(Far);
```

result:Converte	?
Fer:Converte	212
Cel	0
Far	212

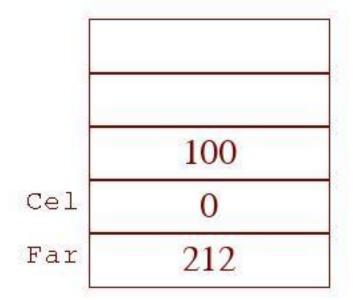


```
result := (Fer - 32) * 5 / 9;
```

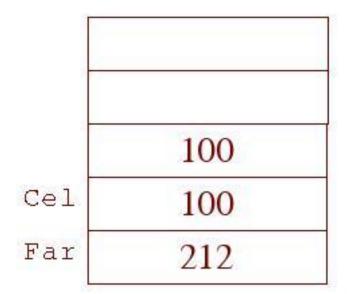
result:Converte	100
Fer:Converte	212
Cel	0
Far	212



```
// término da função
  result := (Fer - 32) * 5 / 9;
end;
```



```
// retorno da função
Cel := converte(Far);
```



#### Outro exemplo: A função area

```
função CalculaArea (entradas: 1, c)
inicio
  resultado \leftarrow 1 * c
fim
programa teste
inicio
  imprima 'Largura do jardim? '
  leia larg
  imprima 'Comprimento do jardim?'
  leia comp
  imprima 'Area: ', CalculaArea(larg, comp)
fim
                                         cederj
```

#### A função area

#### Saida:

```
Largura do jardim? 100
Comprimento do jardim? 50
Area: 5000
```



### A função area em Pascal

```
function CalculaArea(l, c: integer) : integer;
begin
  result := 1 * c;
end;
var
   larg, comp: integer;
begin
   write('Largura do jardim? ');
   readln(larg);
   write('Comprimento do jardim?');
   readln(comp);
  writeln('Area: ', CalculaArea(larg, comp));
end.
                                             cederi
```

#### A funcao area em Java

```
public class IO extends EasyIn {
  public static int CalculaArea(int 1, int c) {
   return 1*c;
  public static void main(String args[]) {
   int larg, comp;
   System.out.print("Largura do jardim? ");
   larg = getInt();
   System.out.print("Comprimento do jardim? ");
   comp = getInt();
   System.out.println("Area: " +
                       CalculaArea(larg, comp));
                                             cederi
```

Um vetor pode ser passado como parâmetro us ando-se apenas o nome do mesmo na chamada à função ou procedimento

```
inicio
  // inicializa o vetor
  imprima 'Maior: ', achaMaior(vetor, pos)
  imprima 'posicao: ', pos
fim
```



```
inicio
  // inicializa o vetor
  imprima 'Maior: ', achaMaior(vetor, pos)
  imprima 'posicao: ', pos
fim
```

vetor: vetor de números inteiros achaMaior: função que devolve o maior elemento de um vetor e a posição onde ele foi encontrado

pos: posição do maior elemento do vetor



```
função achaMaior (entradas: tab
                    saidas: pos)
inicio
  resultado \leftarrow tab[0]
  pos \leftarrow 0
  para i ← 1 até tamanho (tab) faça
      se tab[i] > resultado então
         resultado ←tab[i]
         pos ← i
     fim se
  fim para
fim
```

# Vetores e funções: Alocação em memória

Uma cópia de cada um dos elementos do vetor é criada na pilha.

	6
	3
	6
	2
	6
vetor[3]	3
vetor[2]	6
vetor[1]	2
boa	1

resultado:achaMaior
tab[3]:achaMaior
tab[2]:achaMaior
tab[1]:achaMaior

pos:achaMaior



A passagem de um vetor por valor pode ser uma operação lenta, se o vetor tiver muitos elementos!



# Vetores e funções: passagem por referência

```
função achaMaior (saídas: tab
                  saídas: pos)
inicio
   resultado <- tab[0]
   pos <- 0
   para i <- 1 até tamanho (tab) faça
      se tab[i] > resultado então
         resultado <- tab[i]
         pos <- i
      fim se
   fim para
fim
```

# Vetores e funções: passagem por referência

#### Alocação em memória

6	resultado:achaMaior
6	
3	tab[3]:achaMaior
6	tab[2]:achaMaior
2	tab[1]:achaMaior
1	pos:achaMaior
	6



# Vetores e funções: passagem por referência

A passagem de vetores por referência pode ser vantajos a, se os vetores forem grandes, e se o tempo de processamento for um dado crítico no programa.

Evita-se assim a cópia dos valores dos vetores na pilha.