

Alfredo's MAC0110 Journal

Alfredo Goldman

March 11, 2020

0.1 Aula 04 -[2020-03-11 qua]

Objetivo: Começar a entender como funcionam as funções

0.1.1 O uso de funções é uma abstração natural

Na aula passada já vimos umas funções e isso foi bem natural, foram elas:

- `typeof()` - Que dado um parâmetro devolve o seu tipo
- `div()` - Que dados dois parâmetros devolve a divisão inteira do primeiro pelo segundo
- `print()` e `println()` - Que dados diversos parâmetros os imprime, sem devolver nada

Inclusive, aqui vale a pena ver que podemos pedir ajuda ao Julia para saber o que fazem as funções. Para isso, se usa o `?` antes da função:

```
?typeof()  
?div()  
?print()
```

Ao fazer isso, inclusive descobrimos que o `div()` pode ser usado também como $\div$.

Uma outra função bem útil é a que permite transformar um tipo de valor em outro.

```
parse(Float64, "32")
```

Para conversão de valores em ponto flutuante para inteiros, temos a função `trunc`.

```
trunc{Int64}(2.25)
```

De forma inversa temos o `float`.

```
float(2)
```

Finalmente, podemos transformar um valor em uma string, como em:

```
string(3)
```

ou

```
string(3.57)
```

Também tem muitas funções matemáticas prontas como

- `sin(x)` - calcula seno de x em radianos
- `cos(x)`
- `tan(x)`
- `deg2rad(x)` - converte x de graus em radianos
- `rad2deg(x)`
- `log(x)` - calcula o logaritmo natural de x
- `log(x, b)` - calcula o logaritmo de x na base b
- `log2(x)` - calcula o logaritmo de x na base 2
- `log10(x)`
- `exp(x)` - calcula o expoente da base natural de x
- `abs(x)` - calcula o módulo de x

- `sqrt(x)` - calcula a raiz quadrada
- `isqrt(x)` - calcula a raiz quadrada inteira de x
- `cbrt(x)` - raiz cúbica de x
- `factorial(x)` - calcula o fatorial de x

Em julia também é possível criar funções conforme as suas necessidades, como abaixo:

```
function mensagemDeBomDia()
    println("Tenha um bom dia!")
end
```

Para usar uma função, basta chamá-la:

```
MensagemDeBomDia()
```

Funções, podem receber um ou mais parâmetros:

```
function imprime(a)
    println(" Vou imprimir ", a)
end
imprime(42)
```

Também é possível que uma função chame outra função como em:

```
function imprimeduasvezes(a)
    imprime(a)
    imprime(a)
end
imprimeduasvezes(13)
```

O número de parâmetros determina qual a função correta deve ser chamada:

```
function recebe(a)
    println("Recebi um parametro: ", a)
end
function recebe(a, b)
    println("Recebi dois parametros: ", a, " ", b)
end
```

Conforme a chamada, a função chamada será diferente:

```
recebe(1)
recebe(1, 2)
```

Também dá para chamar funções com variáveis e com operações, como em:

```
a = 10
recebe(a)
recebe(a, a + 1)
```

As funções que vimos até agora imprimem mensagens, mas não devolvem nada. O `typeof()` delas é `nothing`, ou seja, algo que não pode ser atribuído.

Mas, também é possível fazer funções que devolvem valores, como:

```
function soma1(a)
    return a + 1
end
```

Nesse caso, se for passado um parâmetro numérico, a função devolverá o valor incrementado (adicionado de 1).

Claro que isso pode ser usado com fórmulas mais complicadas como:

```
function hipotenusa(a, b)
    hip = a * a + b * b
    return hip
end
```

ou para a verificação de fórmulas, como relações trigonométricas:

```
function verificaeguacao(x)
    soma = sin(x)^2 + cos(x)^2
    return soma == 1.0
end
```