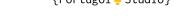
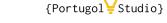
## Portugol Studio - Dominando a Lógica de Programação

Rodrigo R. Silva





- Recursividade é a definição de uma sub-rotina (função ou método) que pode invocar a si mesma.
- Em geral, uma definição recursiva é definida por casos: um número limitado de casos base e um caso recursivo.
- Os casos base são geralmente situações triviais e não envolvem recursão.
- Um exemplo comum usando recursão é a função para calcular o fatorial de um natural N. Nesse caso, no caso base o valor de 0! é 1.
- No caso recursivo, dado um N > 0, o valor de N! é calculado multiplicando por N o valor de (N-1)!, e assim por diante, de tal forma que N! tem como valor N \* (N-1) \* (N-2) \* ... \* (N-N)!, onde (N-N)! representa obviamente o caso base.

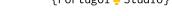


2 / 5

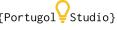
## Em termos recursivos:

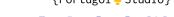
```
função fatorial(x: inteiro): inteiro
inicio
    se x = 0 então
        fatorial <- 1
    senão
        fatorial <- x * fatorial(x - 1)
    fim_se
fim</pre>
```





Funções cujos domínios podem ser definidos recursivamente (tal como o domínio dos números naturais) possuem frequentemente definições recursivas que seguem a definição recursiva do domínio (no caso dos naturais, definimos o comportamento da função com entrada 0, e para cada entrada positiva sucessor(n) definimos o comportamento da função recursiva a partir de seu comportamento com entrada n).

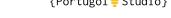




A partir desta definição, também chamada relação de recorrência, calculamos fatorial(3) da seguinte forma:

```
fatorial(3) = 3 * fatorial(3 - 1)
= 3 * fatorial(2)
= 3 * 2 * fatorial(2 - 1)
= 3 * 2 * fatorial(1)
= 3 * 2 * 1 * fatorial(1 - 1)
= 3 * 2 * 1 * fatorial(0)
= 3 * 2 * 1 * 1
= 6
```





## **OBRIGADO!**



github.com/Prof-Rodrigo-Silva



@rodrigo-rosa-da-silva-472928138



