Linguagem de Programação

ADS – Análise e Desenvolvimento de Sistemas
Prof. Vagner Macedo

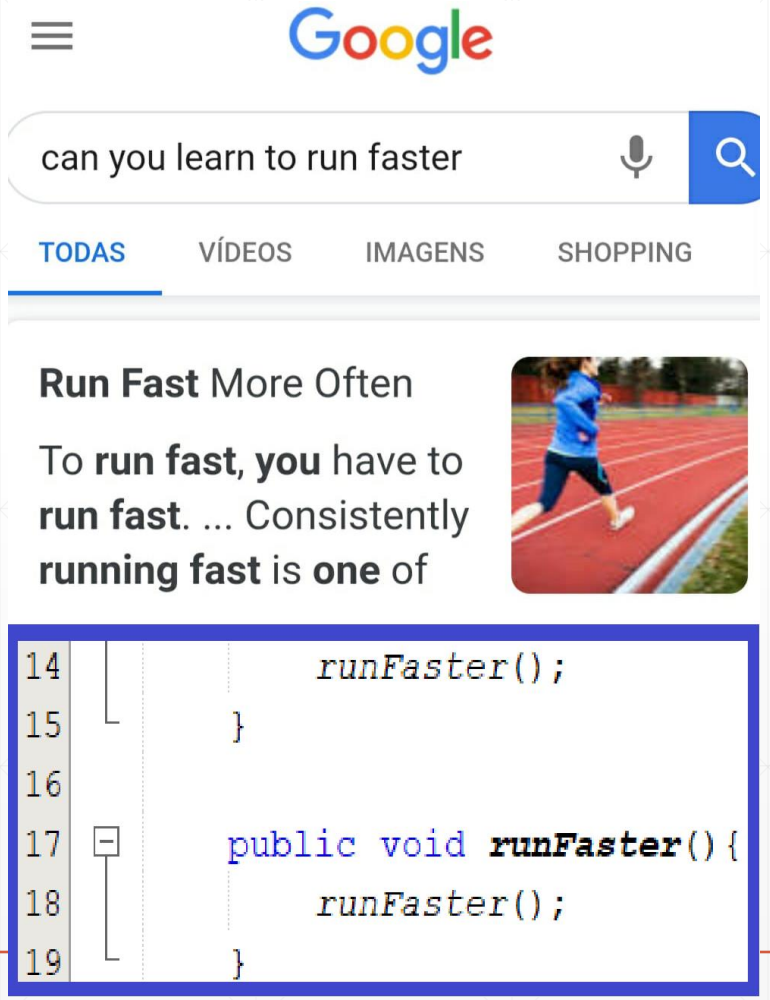
Aula 13
Funções recursivas

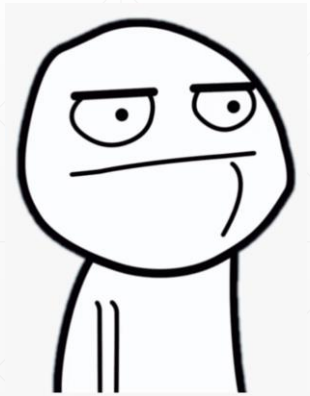
Funções Recursivas

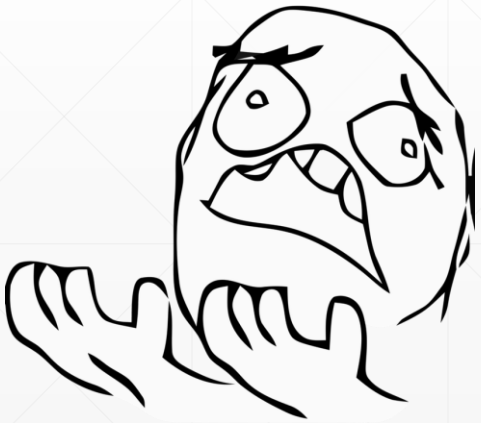
- Dizemos que uma **função recursiva é definida em termos dela mesma** (ou seja, ela chama a si própria e tem os próprios meios para deixar de se chamar).

- É um recurso de programação que pode ser usado em diversas linguagens, dentre as quais, Java.

Funções Recursivas







Funções Recursivas

- Assim, uma **função recursiva** é aquela que **chama a si própria** por um número limitado de vezes.
  - Quando a própria função chama a si própria diretamente, temos uma **recursão direta**.

- Isso também pode ser considerado quando uma função invoca uma segunda função que, em algum momento, volte a chamar a primeira função, tornando esse conjunto de funções um **processo recursivo**.
  - Nesse caso, temos uma **recursão indireta**.

Funções Recursivas

- Lembrando da definição do slide anterior: “**função recursiva** é aquela que **chama a si própria** por um número limitado de vezes”.

- Esse limite é dado pelo tamanho da pilha. Se o valor correspondente ao tamanho máximo da pilha for atingido, haverá um estouro da pilha (“*stack overflow”).*

- Cada vez que uma função é chamada de forma recursiva, são alojados e armazenados uma cópia dos seus parâmetros, de modo a não perder os valores dos parâmetros das chamadas anteriores.

Funções Recursivas

- Basicamente, uma **função recursiva** tem duas partes fundamentais:
  - **Ponto de Parada ou Condição de Parada**: é o ponto onde a função será encerrada.
  - **Regra Geral**: é o método que reduz a resolução do problema através da invocação recursiva de casos menores, que por sua vez são resolvidos pela resolução de casos ainda menores pela própria função, assim sucessivamente até atingir o "*ponto de parada*" que finaliza a função.

Funções Recursivas

- Em geral, uma **função recursiva** é uma solução mais simples e elegante quando comparada a funções tradicionais (iterativas), já que executa tarefas repetitivas **sem utilizar nenhuma estrutura de repetição**, como **for, while** ou **do..while**.
- Entretanto....
  - Essa elegância e simplicidade têm um preço, pois requerem muita atenção em sua implementação.
  - Há de se considerar, também, o custo computacional de sua execução (para soluções simples, a recursividade pode não ser a melhor saída).

Funções Recursivas

- Por outro lado, muitos algoritmos são natural e inerentemente recursivos, de modo que somente com certa dificuldade podem ser programados de forma iterativa (não recursiva). Nesses casos, a recursividade é imprescindível.
- A recursividade é muito útil para implementar definições indutivas, em geral. Por exemplo:
  - fatorial, série de Fibonacci, ordenação etc.
- Também é útil para manipular estruturas de dados recursivas:
  - listas ligadas, pilhas, árvores etc.

Funções Recursivas

- **Recursão de cauda (*Tail recursion*):**
  - Ocorre quando a recursão é a última coisa a ser executada na sub-rotina. Qualquer lógica que a sub-rotina precise fazer ocorrem antes da recursão.

```
public static void main(String[] args) {
    imprimirNumerosRecursivamente(1);
}
private static void imprimirNumerosRecursivamente (int i){
    System.out.println("Entrando na função recursiva: " + i);
    if (i < 10)
        imprimirNumerosRecursivamente (i + 1);
}
```

Funções Recursivas

- **Recursão de cabeça (Head recursion):**
  - Ocorre quando a recursão é a primeira coisa a ser executada na sub-rotina. Qualquer outro processo a ser executado ocorre após a recursão.

```
public static void main(String[] args) {
    imprimirNumerosRecursivamente(1);
}
private static void imprimirNumerosRecursivamente (int i){
    if (i < 10)
        imprimirNumerosRecursivamente (i + 1);
    System.out.println("Entrando na função recursiva: " + i);
}
```

Funções Recursivas

- Exemplo 1 (impressão de números de 1 a ~~10~~ 5 – *que poderia ser facilmente resolvida com algoritmos iterativos*):

```
public static void main(String[] args) {
    int x = 1;
    System.out.println("Valor de 'x' antes da função: " + x);
    imprimirNumerosRecursivamente(x);
    System.out.println("Valor de 'x' após a função: " + x);
}
private static void imprimirNumerosRecursivamente (int i){
    System.out.println("Entrando na função recursiva: " + i);
    if (i < 5)
        imprimirNumerosRecursivamente (i + 1);
    System.out.println("Saindo da função recursiva: " + i);
}
```

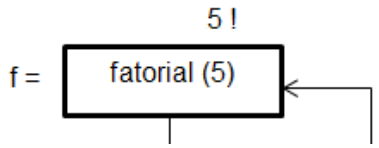
Funções Recursivas

- Exemplo 2 (cálculo de fatorial – *função tipicamente recursiva*):

```
public static void main(String[] args) {
    int f = 0;
    f = fatorialRecursivo(5);
    System.out.println(f);
}
private static int fatorialRecursivo (int n) {
    if (n <= 0)
        return 1;
    else
        return n * fatorialRecursivo(n-1);
}
```

Funções Recursivas

- Cálculo de fatorial:




```
factorial( n ):
if n == 1:
    return 1
else:
    return n * factorial(n-1):
```

factorial(n) =

www.penjee.com

Obrigado!