Recursão

Prof. Rafael Guterres Jeffman
rafael.jeffman@gmail.com

Fecomércio RS | Senac

Recursão

- É o processo de repetir itens de forma auto-similar.

- É a definição de uma função baseada nela mesma.

- É a definição de uma estrutura, baseada na própria estrutura.

Representação de Algoritmos

- Todo algoritmo pode ser representado de forma recursiva.
- Cálculo Lambda
- Linguagens Funcionais

Criando um método recursivo

- Define-se o critério de parada.

- Define-se a recursão de forma que o critério de parada seja atingido.

Fatorial

- Seja a função Fatorial, f(x), a sua definição é:
  - f(x) = 1 para x = 0;
  - f(x) = f(x - 1) * x, para x > 1;

Implementação Java

```
public double fatorial (int n) {

    if (n == 0)

        return 1;

    return n * fatorial (n - 1);

}
```

Execução para N=4

- (fatorial(4))
- (4 * (fatorial(3)))
- (4 * (3 * (fatorial(2))))
- (4 * (3 * (2 * (fatorial(1)))))
- (4 * (3 * (2 * (1 * (fatorial(0))))))
- (4 * (3 * (2 * (1 * (1)))))
- (4 * (3 * (2 * (1))))
- (4 * (3 * (2)))
- (4 * (6))
- (24)

Número de Fibonacci

- Forma de calcular o n-ésimo número da série de Fibonacci.

- A função f(x) para calcular esse número poder ser definida recursivamente como:

  - f(x) = 0, para x = 0;

  - f(x) = 1, para x = 1;

  - f(x) = f(x-1) + f(x-2), para x > 1.

Implementação em Java

```
public double fibonacci (int n) {

    if (n == 0) return 0;

    if (n == 1) return 1;

    return fibonacci(n-1) + fibonacci(n-2);

}
```

Execução para N=5

- f(5)
- f(4) + f(3)
- f(3) + f(2) + f(2) + f(1)
- f(2) + f(1)+ f(1) + f(0) + f(1) + f(0) + f(1)
- f(1) + f(0) + f(1) + f(1) + f(0) + f(1) + f(0) + f(1)

Custo da Recursão

- Em linguagens imperativas, não existe otimização do processo de recursão, sendo necessária a repetição de diversas chamadas a mesma função com os mesmos parâmetros.

- Apesar da forma sucinta de descrição do algoritmo, a execução não é “sucinta”.

- Existem versões iterativas dos algoritmos recursivos, no entanto, essas não são tão “elegantes”.

Fatorial Iterativo

```
public double fatorial (int n) {

        double fat = 1;

        for (int i = 2; i < n; ++i)

                fat *= n;

        return fat;

}
```

Implementação em Java

```
public double fibonacci (int n) {

        double f1 = 0, f2 = 1, fibo = 0;

        for (int x = 1; x <= n; ++x) {

                fibo = f1 + f2;

                f2 = f1;

                f1 = fibo;

        }

        return fibo;

}
```

Exercício

- Verifique a diferença de tempo de execução dos algoritmos recursivos e iterativos para os seguintes valores de “n”:

  5 10 20 30 40 50 51 52 53 54 55 75

- Para esse exercício você vai precisar utilizar o método “System.currentTimeMillis()”