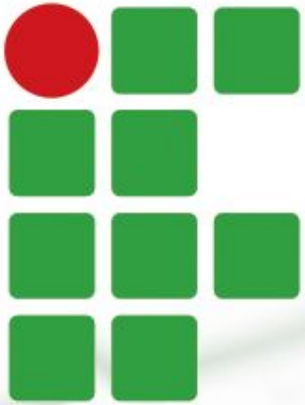

INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Estruturas de Dados 2

- Recursividade -

Prof. MSc. Adriano Antunes Prates

INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Recursividade

"Para aprender Recursividade, antes você precisa aprender Recursividade" - *Bob Esponja*

Recursividade

- O objetivo principal da recursão é dividir um problema em **instâncias menores**, até que possa ser resolvido, retornando posteriormente às **instâncias originais**.
- Trata-se de um recurso bastante aplicada na técnica de programação **Divisão e Conquista**.
- O projeto de muitos algoritmos eficientes é baseado no método da **Divisão e Conquista**.

Recursividade

- Exemplo clássico…

- **Faça um algoritmo que calcule o fatorial de 6.**

Recursividade

- Exemplo clássico…

- **Faça um algoritmo que calcule o fatorial de 6.**

Você poderia logo pensar em um **algoritmo iterativo** (*resolução usando laços de repetição*)

6! == 1 * 2 * 3 * 4 * 5 * 6

6! == 720

Recursividade

- Exemplo clássico…
- **Faça um algoritmo que calcule o fatorial de 6.**

Entretanto, poderia elaborar um **algoritmo recursivo**

6! == 6 * 5!

Recursividade

- Exemplo clássico…
- **Faça um algoritmo que calcule o fatorial de 6.**

Entretanto, poderia elaborar um **algoritmo recursivo**

6! == 6 * 5!

5! == 5 * 4!

Recursividade

- Exemplo clássico…
- **Faça um algoritmo que calcule o fatorial de 6.**

Entretanto, poderia elaborar um **algoritmo recursivo**

6! == 6 * 5!

5! == 5 * 4!

4! == 4 * 3!

Recursividade

- Exemplo clássico…
- **Faça um algoritmo que calcule o fatorial de 6.**

Entretanto, poderia elaborar um **algoritmo recursivo**

6! == 6 * 5!

5! == 5 * 4!

4! == 4 * 3!

3! == 3 * 2!

Recursividade

- Exemplo clássico…
- **Faça um algoritmo que calcule o fatorial de 6.**

Entretanto, poderia elaborar um **algoritmo recursivo**

6! == 6 * 5!

5! == 5 * 4!

4! == 4 * 3!

3! == 3 * 2!

2! == 2 * 1!

Recursividade

- Exemplo clássico…
- **Faça um algoritmo que calcule o fatorial de 6.**

Entretanto, poderia elaborar um **algoritmo recursivo**

6! == 6 * 5!

5! == 5 * 4!

4! == 4 * 3!

3! == 3 * 2!

2! == 2 * 1!

1! == 1 (essa é fácil)

Recursividade

- Exemplo clássico…
- **Faça um algoritmo que calcule o fatorial de 6.**

Entretanto, poderia elaborar um **algoritmo recursivo**

6! == 6 * 5!

5! == 5 * 4!

4! == 4 * 3!

3! == 3 * 2!

2! == 2 * 1

1! == 1 (essa é fácil)

Recursividade

- Exemplo clássico…
- **Faça um algoritmo que calcule o fatorial de 6.**

Entretanto, poderia elaborar um **algoritmo recursivo**

6! == 6 * 5!

5! == 5 * 4!

4! == 4 * 3!

3! == 3 * 2

2! == 2 * 1

1! == 1 (essa é fácil)

Recursividade

- Exemplo clássico…
- **Faça um algoritmo que calcule o fatorial de 6.**

Entretanto, poderia elaborar um **algoritmo recursivo**

6! == 6 * 5!

5! == 5 * 4!

4! == 4 * 6

3! == 3 * 2

2! == 2 * 1

1! == 1 (essa é fácil)

Recursividade

- Exemplo clássico…
- **Faça um algoritmo que calcule o fatorial de 6.**

Entretanto, poderia elaborar um **algoritmo recursivo**

6! == 6 * 5!

5! == 5 * 24

4! == 4 * 6

3! == 3 * 2

2! == 2 * 1

1! == 1 (essa é fácil)

Recursividade

- Exemplo clássico…
- **Faça um algoritmo que calcule o fatorial de 6.**

Entretanto, poderia elaborar um **algoritmo recursivo**

6! == 6 * 120

5! == 5 * 24

4! == 4 * 6

3! == 3 * 2

2! == 2 * 1

1! == 1 (essa é fácil)

Recursividade

- Exemplo clássico…
- **Faça um algoritmo que calcule o fatorial de 6.**

Entretanto, poderia elaborar um **algoritmo recursivo**

6! == 720

5! == 5 * 24

4! == 4 * 6

3! == 3 * 2

2! == 2 * 1

1! == 1 (essa é fácil)

INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Recursividade

- Exemplo clássico
-

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

factorial(n) =

En ivo

1! == 1 (essa é fácil)

Recursividade

- A estrutura básica de uma função recursiva se resume a:
  - Se a instância **é grande**, **reduza-a a uma instância menor do mesmo problema**;
  - Se a instância **é pequena**, **resolva-o diretamente, como puder**.
  - **Depois de resolvida** a instância menor, **volte à instância original**.

Recursão vs. Iteração

- Tanto **iteração** quanto **recursão** usam um mecanismo de repetição.
- A **iteração** usa repetição em forma de estruturas de repetição (p.ex.: for, while, do-while).
- Já a **recursão** usa a repetição na forma de **chamadas sucessivas a uma rotina**.
- **Mas ambas precisam de um teste de terminação.**
- A iteração termina quando uma condição de teste falha, e a recursão termina quando se atinge o caso trivial de solução.

Recursão vs. Iteração

- Tanto **iteração** quanto **recursão** podem, se não definidas corretamente, entrar em loop infinito de execução…
- No caso da iteração, se o teste nunca se tornar falso.
- **No caso da recursão, se o problema não for reduzido gradativamente, de forma que convirja para a solução do caso trivial.**

INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Bora Codar (1)

- Tente elaborar **SEM CONSULTAS EXTERNAS**, os seguintes algoritmos recursivos…
- **Função Recursiva para retornar:**

1. Resultado de X^Y
2. A soma dos valores de um Array.
3. O maior valor de um Array.
4. Índice (Busca sequencial) de um valor em Array.
5. Índice (Busca binária) de um valor em Array.

INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Bora Codar (2)

- A **sequência Fibonacci**, amplamente encontrada na natureza, é aquela em que o valor do N-ésimo termo da série é a soma dos dois valores anteriores
  - **1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...**
- **Faça em um mesmo código-fonte, duas funções para encontrar o N-ésimo termo da série:**
  - Função Iterativa
  - Função Recursiva
- **Invoque ambas funções para encontrar o 45º termo da série e analise o que acontece...**