

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
RIO GRANDE DO NORTE
Campus Currais Novos

ALGORITMOS

FUNÇÕES – AULA III – RECURSÃO

Prof. Bruno E. G. Gomes

Currais Novos,
2011

INTRODUÇÃO

- Uma função, a grosso modo, recebe valores e retorna uma resposta para quem a chamou.
- No corpo da função, podem ser chamadas outras funções
 - Fazem parte do trabalho requerido pela função que as chama.
- Uma função também pode chamar a si própria
 - É o que chamamos de *recursão* ou *recursividade*



FUNÇÕES RECURSIVAS

- Uma função recursiva resolve parte do problema a cada chamada a ela mesma
- Deve existir um caso mais simples (caso base) que provoque a parada das chamadas recursivas
- Após atingir o caso base, os demais casos na “pilha de execução” são calculados até ser devolvido o resultado final da função

- **Forma geral:**

```
f1 (p1, p2, ..., pn) : tipo
  inicio
    se <caso_base> entao
      retorne <valor>
    senao
      retorne f1(p1, p2, ..., pn)
    fimse
  fimfuncao
```



DEFINIÇÃO DE FUNÇÃO RECURSIVA

Duas partes fundamentais:

- **Passo básico** (ou ponto de parada, ou caso base)
 - Resultado é imediatamente conhecido
 - Resolvido sem recursão
 - Geralmente é um valor limite inferior ou superior da regra geral
 - Possibilita parar a chamada recursiva

- **Passo recursivo** (ou regra geral)
 - Reduz a resolução do problema através da invocação recursiva de casos menores
 - Até a parada no passo básico (resultado conhecido, utilizado para resolver as outras funções)



FUNÇÕES RECURSIVAS

- Pode ser uma forma mais simples de descrever determinados problemas
- Execução normalmente é menos eficiente que algoritmos não-recursivos (denominados *iterativos*)
 - Algoritmos iterativos são aqueles que utilizam repetição (*for*, *while*, etc.)
- Em algumas linguagens de programação, tais como as linguagens funcionais (ML, p.ex), não existe repetição, apenas recursão



EXEMPLO CLÁSSICO – FATORIAL

$$\text{fatorial}(0) = 1$$

$$\text{fatorial}(n) = n * (n - 1) * (n - 2) * \dots * 1$$



EXEMPLO CLÁSSICO – FATORIAL

$$\text{fatorial}(0) = 1$$

$$\text{fatorial}(n) = n * (n - 1) * (n - 2) * \dots * 1$$

Caso base

Caso geral



EXEMPLO CLÁSSICO – FATORIAL

$$\text{fatorial}(0) = 1$$

Caso base

$$\text{fatorial}(n) = n * (n - 1) * (n - 2) * \dots * 1$$

ou, de outra forma:

$$\text{fatorial}(n) = n * \text{fatorial}(n - 1)$$

Caso geral



EXEMPLO CLÁSSICO – FATORIAL

$$\textit{fatorial}(0) = 1$$

$$\textit{fatorial}(n) = n * \textit{fatorial}(n - 1)$$


```
int fatorial (int n) {  
    if (n == 0) {  
        return 1;  
    } else {  
        return n * fatorial (n - 1);  
    }  
}
```



EXECUÇÃO DA FUNÇÃO FATORIAL RECURSIVA

- Suponha que se queira calcular o fatorial de 3, então, a chamada à função seria: **fatorial (3)**
- Pilha de execução para fatorial (3):
 1. **fatorial (3) = 3 * fatorial (2)**
 2. **fatorial (2) = 2 * fatorial (1)**
 3. **fatorial (1) = 1 * fatorial (0)**
 4. **fatorial (0) = 1**

Neste ponto, atingimos o caso base ($n == 0$). Nesse caso, a função começa a ser calculada a partir do valor conhecido (quanto $n == 0$, o fatorial é 1)

3. **fatorial (1) = 1 * fatorial (0) = 1 * 1 = 1**
 2. **fatorial (2) = 2 * fatorial (1) = 2 * 1 = 2**
 1. **fatorial (3) = 3 * fatorial (2) = 3 * 2 = 6 (resultado final)**
- 

DESENVOLVIMENTO DE ALGORITMOS RECURSIVOS

1. Certificar-se de que problema pode ser dividido em sub-passos da solução geral;
1. Definir um ponto de parada (caso base);
1. Definir uma regra geral;
2. Verificar se o algoritmo termina – se converge para o caso base.



USAR OU NÃO A RECURSÃO?

- Todo algoritmo recursivo, de forma mais fácil ou mais difícil, sempre pode ser resolvido de forma iterativa (e *vice-versa*)
- Em C++, soluções iterativas normalmente possuem melhor desempenho que recursivas
- Algoritmos recursivos geralmente produzem soluções mais elegantes que os iterativos
 - Use-a quando a solução for simples, sem comprometer o desempenho



EXERCÍCIOS

1. Faça uma função recursiva que resolva a seguinte equação:

$$r(x) = 2 * r(x - 1) - 4 \quad //\text{passo recursivo}$$

$$r(0) = 2 \quad //\text{caso base}$$

2. Defina uma função recursiva que receba um número inteiro x ($x \geq 0$) e retorne o termo correspondente da série de *Fibonnaci*.

o Fibonnaci = 1 1 2 3 5 8 13 21 ...

3. Faça uma função recursiva que calcule a potência de um número x (base) por y (expoente)

`int pot (int x, int y)`

Assuma que $y \geq 0$

