

# Funções recursivas

Roberto Rocha



**E se uma função chamar a si mesma?**

# Funções recursivas

- Funções podem ser chamadas recursivamente, dentro do corpo de uma função podemos chamar novamente a própria função.
- As implementações recursivas devem ser pensadas conforme a definição recursiva do problema que se deseja resolver.
- Exemplo: Valor do fatorial de um número definido de forma recursiva:

$$n! = \begin{cases} 1, & \text{se } n = 0 \\ n \times (n-1)!, & \text{se } n > 0 \end{cases}$$

# Funções recursivas

$$n! = \begin{cases} 1, & \text{se } n = 0 \\ n \times (n-1)!, & \text{se } n > 0 \end{cases}$$

função fat ( n:inteiro):inteiro

início

se (n=0) então

retorne 1

senão

retorne n\*fat(n-1)

fimfunção

início

escreva(fat(5))

fimalgoritmo

fat(5)

fat(5)

fat(4)

fat(3)

fat(2)

fat(1)

fat(0)

retorne 1\*fat(0)

retorne 2\*fat(1)

retorne 3\*fat(2)

retorne 4\*fat(3)

retorne 5\*fat(4)

# Funções recursivas

$$n! = \begin{cases} 1, & \text{se } n = 0 \\ n \times (n-1)!, & \text{se } n > 0 \end{cases}$$

função fat ( n:inteiro):inteiro

inicio

se (n=0) então

retorne 1

senão

retorne n\*fat(n-1)

fimfunção

inicio

escreva(fat(5))

fimalgoritmo

120

fat(5)

fat(5)

fat(4)

fat(3)

fat(2)

fat(1)

fat(0)

1

1 \* 1

2 \* 1

3 \* 2

4 \* 6

5 \* 24

120

# Funções recursivas

função fat ( n:inteiro):inteiro

inicio

se (n=0) então

retorne 1

senão

retorne n\*fat(n-1)

fimfunção

inicio

escreva(fat(5))

fimalgoritmo

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  int fat(int n);
4  int main()
5  {
6      printf("fatorial de 5=%d\n", fat(5));
7      return 0;
8  }
9
10 int fat(int n){
11     if (n==0)
12         return 1;
13     else
14         return n*fat(n-1);
15 }
```

fatorial de 5=120

Process returned 0 (0x0) execution time : 0.016 s  
Press any key to continue.

# Funções recursivas

Exercício de fixação:

Faça um programa que peça um número inteiro ao usuário e retorne a soma de todos os números de 1 até o número que o usuário introduziu ou seja:  $1 + 2 + 3 + \dots + n$

- a) faça sem utilizar recursividade
- b) utilize agora recursividade.

# Funções recursivas

Exercício de fixação:

Faça um programa que peça um número inteiro ao usuário e retorne a soma de todos os números de 1 até o número que o usuário introduziu ou seja:  $1 + 2 + 3 + \dots + n$

a) faça sem utilizar recursividade

b) utilize agora recursividade.

Algoritmo "Soma1aN"

var

soma,i,n:inteiro

inicio

leia(n)

$s \leftarrow 0$

para i de 1 ate n passo 1 faça

$s \leftarrow s+i$

fimpara

escreva(s)

fimalgoritmo

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main()
5  {
6      int soma,i,n;
7      printf("Digite um valor:");
8      scanf("%d",&n);
9      soma=0;
10     for(i=1;i<=n;i=i+1){
11         soma=soma+i;
12     }
13     printf("A soma dos numero de 1 a %d = %d\n",n,soma);
14     return 0;
15 }
```

```
Digite um valor:5
A soma dos numero de 1 a 5 = 15
Process returned 0 (0x0)   execution time : 20.843 s
Press any key to continue.
```



# Funções recursivas

Exercício de fixação:

Faça um programa que peça um número inteiro ao usuário e retorne a soma de todos os números de 1 até o número que o usuário introduziu ou seja:  $1 + 2 + 3 + \dots + n$

a) faça sem utilizar recursividade

b) utilize agora recursividade.

Vamos criar uma definição para a solução

$$\sum_1^n \begin{cases} 1, & \text{se } n = 1 \\ n + \sum_1^{n-1} & \text{se } n > 1 \end{cases}$$

função soma (n:inteiro):inteiro

início

se (n=1) então

retorne 1

senão

retorne n+soma(n-1)

fimfunção

início

escreva(soma(5))

fimalgoritmo

# Funções recursivas

Exercício de fixação:

Faça um programa que peça um número inteiro ao usuário e retorne a soma de todos os números de 1 até o número que o usuário introduziu ou seja:  $1 + 2 + 3 + \dots + n$

a) faça sem utilizar recursividade

b) utilize agora recursividade.

função soma (n:inteiro):inteiro

início

se (n=1) então

retorne 1

senão

retorne n+soma(n-1)

fimfunção

início

escreva(soma(5))

fimalgoritmo

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  int soma(int n);
4  int main()
5  {
6      printf("A soma de 1 ate %d = %d\n",5,soma(5));
7      return 0;
8  }
9
10 int soma(int n)
11 {
12     if (n==1)
13         return 1;
14     else
15         return n+soma(n-1);
16 }
```

A soma de 1 ate 5 = 15

Process returned 0 (0x0) execution time : 0.016 s

Press any key to continue.

# Funções recursivas

Exercício de fixação:

Crie uma função que receba 2 parâmetros(x,y) e devolva x elevado a y.

**a) faça sem utilizar recursividade**

**b) utilize agora recursividade.**

# Potenciação

Potencia entre dois valores:

$$2^3 = 2 \times 2 \times 2 = 8$$

A potência de um número (X) por outro (Y) pode ser calculado como a multiplicação sucessiva do primeiro valor y vezes

Exemplo: potencia de 2 e 3

$$2^3 = 2 \times 2 \times 2 = 8$$

# Potenciação

Programa para pedir ao usuário dois valores e imprimir o primeiro elevado ao segundo valor.

```
#include <stdio.h>
#include <stdlib.h>

/*
Programa para pedir ao usuário dois valores
e imprimir o primeiro elevado ao segundo valor.
*/

int main()
{
    int a,b,pot,i;
    printf("Digite o valor de a:");
    scanf("%i",&a);
    printf("Digite o valor de b:");
    scanf("%i",&b);
    pot=1;
    for (i=1;i<=b;i++)
    {
        pot=pot*a;
    }
    printf("%d elevado a %d = %d\n",a,b,pot);
    return 0;
}
```

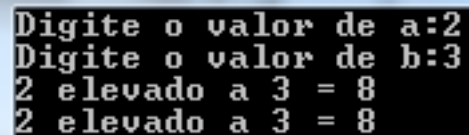
```
Digite o valor de a:2
Digite o valor de b:3
2 elevado a 3 = 8
```

# Potenciação

Faça uma função que receba dois parâmetros (passados por valor) e retorne a potencia do primeiro parâmetro elevado ao segundo parâmetro.

```
int potencia(int a, int b);
int main()
{
    int a,b,pot,i;
    printf("Digite o valor de a:");
    scanf("%i",&a);
    printf("Digite o valor de b:");
    scanf("%i",&b);
    pot=1;
    for (i=1;i<=b;i++)
    {
        pot=pot*a;
    }
    printf("%d elevado a %d = %d\n",a,b,pot);
    printf("%d elevado a %d = %d\n",a,b,potencia(a,b));
    return 0;
}

int potencia(int a, int b)
{
    int pot;
    pot=1;
    while (b>0)
    {
        pot=pot*a;
        b--;
    }
    return pot;
}
```

A screenshot of a terminal window showing the execution of the program. It displays the prompts for input and the resulting output for two test cases: 2 to the power of 3 and 2 to the power of 3.

```
Digite o valor de a:2
Digite o valor de b:3
2 elevado a 3 = 8
2 elevado a 3 = 8
```

# Funções recursivas - Potenciação

Exercício de fixação:

Crie uma função que receba 2 parâmetros(x,y) e devolva x elevado a y.

a) faça sem utilizar recursividade

b) utilize agora recursividade.

$$x^y = \begin{cases} 1, & \text{se } y = 0 \\ x * x^{y-1} & \end{cases}$$

```
função potencia (x,y:inteiro):inteiro
inicio
    se (y=0) então
        retorne 1
    senão
        retorne x * potencia(x,y-1)
    fimse
fimfunção
```

```
inicio
    escreva(pot(2,3))
finalgoritmo
```

# Funções recursivas

Exercício de fixação:

Crie uma função que receba 2 parâmetros(x,y) e devolva x elevado a y.

a) faça sem utilizar recursividade

b) utilize agora recursividade.

função potencia (x,y:inteiro):inteiro

inicio

se (y=0) então

retorne 1

senão

retorne x \* potencia(x,y-1)

fimse

fimfunção

inicio

escreva(pot(2,3))

fimalgoritmo

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  int potencia(int x, int y);
4  int main()
5  {
6      printf("A resposta de 2 elevado a 3 = %d!\n", potencia(2,3));
7      return 0;
8  }
9
10 int potencia(int x, int y)
11 {
12     if (y==0)
13     {
14         return 1;
15     }
16     else{
17         return x * potencia(x,y-1);
18     }
19 }
```

```
A resposta de 2 elevado a 3 = 8!
Process returned 0 (0x0)   execution time : 3.397 s
Press any key to continue.
```



# Funções recursivas

Exercício de fixação:

Crie uma função que imprima o  $n$ -ésimo termo da série de Fibonacci

- a) faça sem utilizar recursividade**
- b) utilize agora recursividade.**

# Série de Fibonacci

Entrada

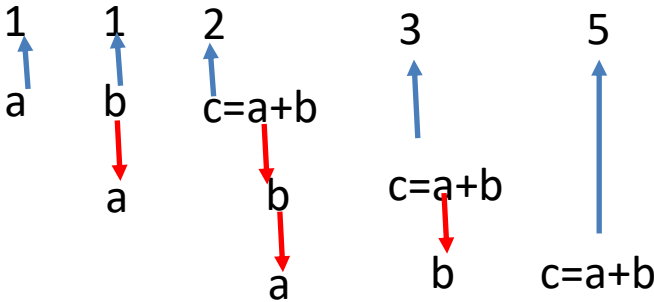
Número

Processamento

Imprimir N números da série de Fibonacci

Saída

Imprimir o enésimo termo da série de Fibonacci



$a = b$   
 $b = c$

# Imprimir os N primeiros termos da Série de Fibonacci

## Algoritmo "Fibonacci"

// Imprimir os N primeiros termos da Série de Fibonacci

var

numero, a,b,c,cont:inteiro

inicio

leia(numero)

a ← 1

b ← 1

para cont de 2 ate numero-1) faca

c ← a+b

a ← b

b ← c

fimpara

escreva(c)

fimalgoritmo

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main()
5  {
6      int a,b,c,cont,numero;
7      printf("Digite o termo desejado:");
8      scanf("%d",&numero);
9      a=1;
10     b=1;
11     for (cont=2;cont<numero;cont=cont+1)
12     {
13         c=a+b;
14         a=b;
15         b=c;
16     }
17     printf("O %d termo da serie = %d\n",numero,c);
18     return 0;
19 }
```

```
Digite o termo desejado:6
0 6 termo da serie = 8
```

```
Process returned 0 (0x0)    execution time : 2.159 s
Press any key to continue.
```

# Funções recursivas

Exercício de fixação:

Crie uma função que imprima o enésimo termo da serie de Fibonacci

a) faça sem utilizar recursividade

b) utilize agora recursividade.

$$F(n) \begin{cases} 1, & \text{se } n = 1 \\ 1, & \text{se } n = 2 \\ f(n-1)+f(n-2), & \text{se } n > 2 \end{cases}$$

```
função fibonacci(n:inteiro):inteiro
inicio
    se (n=1 ou n = 2) então
        retorne 1
    senão
        retorne fibonacci(n-1) + fibonacci(n-2)
    fimse
fimfunção
```

```
inicio
    escreva(fibonacci(6))
finalgoritmo
```

# Funções recursivas

Exercício de fixação:

Crie uma função que imprima o enésimo termo da serie de Fibonacci

a) faça sem utilizar recursividade

b) utilize agora recursividade.

função fibonacci(n:inteiro):inteiro

inicio

se (n=1 ou n = 2) então

retorne 1

senão

retorne fibonacci(n-1) + fibonacci(n-2)

fimse

fimfunção

inicio

escreva(fibonacci(6))

fimalgoritmo

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  int fibonacci(int n);
4  int main()
5  {
6      printf("O %d termo da serie = %d\n",6,fibonacci(6));
7      return 0;
8  }
9  int fibonacci(int n)
10 {
11     if (n==1 || n==2)
12     {
13         return 1;
14     }
15     else
16     {
17         return fibonacci(n-1)+fibonacci(n-2);
18     }
19 }
```

O 6 termo da serie = 8

Process returned 0 (0x0) execution time : 3.355 s  
Press any key to continue.

# Resto da divisão

resto inteiro da divisão de X por Y

O resto inteiro de um número (X) por outro (Y) pode ser calculado como o que sobra após sucessivas subtrações do segundo valor em relação ao primeiro

Exemplo: resto da divisão de 7 por 2

$$7 - 2 = 5$$

$$5 - 2 = 3$$

$$3 - 2 = 1$$

Como 1 é menor do que 2 o resto será esse valor!

# Resto da divisão

resto inteiro da divisão de X por Y

O resto inteiro de um número (X) por outro (Y) pode ser calculado como o que sobra após sucessivas subtrações do segundo valor em relação ao primeiro

Exemplo: resto da divisão de 8 por 2

$$8 - 2 = 6$$

$$6 - 2 = 4$$

$$4 - 2 = 2$$

$$2 - 2 = 0$$

Como 0 é menor do que 2 o resto será esse valor!

# Resto da divisão

resto inteiro da divisão de X por Y

O resto inteiro de um número (X) por outro (Y) pode ser calculado como o que sobra após sucessivas subtrações do segundo valor em relação ao primeiro

Exemplo: resto da divisão de 5 por 7

Como 5 é menor do que 7 o resto será esse valor!

$$\text{resto}(x, y) \begin{cases} x, & \text{se } x < y \\ \text{resto}(x-y, y), & \text{se } x > y \end{cases}$$



# Exercícios:

**Para essa prática não se pode utilizar o operador %**

- 1 - Faça um programa que peça ao usuários dois valores e imprima o resto inteiro da divisão de X por Y –você não pode utilizar o operador %
- 2 – Faça uma função que receba dois parâmetros (passados por valor) e retorne o resto inteiro da divisão de X por Y – você não pode utilizar o operador %
- 3 – Faça uma função **RECURSIVA** que receba dois parâmetros (passados por valor) e retorne o resto inteiro da divisão de X por Y



**PUC Minas**  
**Virtual**