



# **ALGORITMOS E COMPLEXIDADE (ARA0174)**

## **AULA 5**

**DATA: 17/03/2022**



# Aula 5- ALGORITMOS E COMPLEXIDADE

## TEMA 2 - RECURSIVIDADE

CONTINUAÇÃO...



**Estácio**

# Aula 5- ALGORITMOS E COMPLEXIDADE

## TEMA 2 - RECURSIVIDADE

### Exercício 1

Construa um programa em C++ que faça uma rotina recursiva para realizar a soma de todos os números compreendidos de 0 a n (n é o valor digitado pelo usuário).

**Exemplo:**

Se  $n = 3$ , então a soma de todos os números compreendidos de 0 a 3 será:

$$3 + 2 + 1 + 0 = 6.$$

# Aula 5- ALGORITMOS E COMPLEXIDADE

## TEMA 2 - RECURSIVIDADE

### Exercício 1

```
#include<iostream>
#include<locale.h>
using namespace std;

int soma_n(int num);

int main()
{
    int n, soma_final;

    setlocale(LC_ALL, "portuguese");

    cout << "\n\tPrograma (SOMA TODOS OS NÚMEROS DE 0 a n:)\n";

    cout << "\n Digite o número: ";
    cin >> n; /*o número digitado vai ser guardado na memória*/
    soma_final = soma_n(n); /*A variável soma_final está chamando a função soma_n*/
    cout << "\n\n 0 soma de todos os números de 0 até " << n << " = " << soma_final << "\n\n";

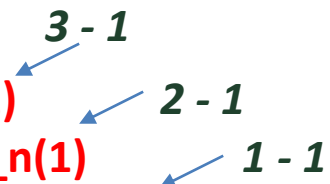
    return 0;
}

int soma_n(int num)
{
    int resultado;
    if (num == 0)
    {
        return (0);
    }
    else
    {
        resultado = num + soma_n(num - 1);
    }
    return (resultado);
}
```

#### POR EXEMPLO:

**resultado = num + soma\_n(num-1);**

*Digitando o número 3 o processo recursivo ocorre em linhas gerais conforme apresentado abaixo:*

  
3 - 1  
**resultado = 3 + soma\_n(2)**  
2 - 1  
**resultado = 3 + 2 + soma\_n(1)**  
1 - 1  
**resultado = 3 + 2 + 1 + soma\_n(0)**  
**resultado = 3 + 2 + 1 + 0**  
**resultado = 6**

# Aula 5- ALGORITMOS E COMPLEXIDADE

## TEMA 2 - RECURSIVIDADE

### Exercício 1

```
#include<iostream>
#include<locale.h>
using namespace std;
```

```
int soma_n(int num);
```

```
int main()
```

```
{
```

```
    int n, soma;
```

```
    setlocale(LC_ALL, "portuguese");
```

*AO COMPILAR E EXECUTAR O CÓDIGO APARECERÁ:*

Programa (SOMA TODOS OS NÚMEROS DE 0 a n:)

Digite o número: 3

O soma de todos os números de 0 até 3 = 6

```
{
    int resultado;
    if (num == 0)
    {
        return (0);
    }
    else
    {
        resultado = num + soma_n(num - 1);
    }
    return (resultado);
}
```



# Aula 5- ALGORITMOS E COMPLEXIDADE

## TEMA 2 - RECURSIVIDADE

### Exercício 2

Escreva um código em C++ utilizando uma função recursiva para a multiplicação de um dado inteiro “x” por um inteiro “y”, usando somas sucessivas conforme mostrado no exemplo a seguir.

**Exemplo:**

$$4 * 3 = 3 + 3 + 3 + 3 = 12.$$

# Aula 5- ALGORITMOS E COMPLEXIDADE

## TEMA 2 - RECURSIVIDADE

### Exercício 2 (ANALISANDO)

```
int mult (int x, int y) {
```

```
}
```

# Aula 5- ALGORITMOS E COMPLEXIDADE

## TEMA 2 - RECURSIVIDADE

### Exercício 2 (ANALISANDO)

```
int mult (int x, int y) {  
    if (x == 0)  
        return  
  
}
```



# Aula 5- ALGORITMOS E COMPLEXIDADE

## TEMA 2 - RECURSIVIDADE

### Exercício 2 (ANALISANDO)

```
int mult (int x, int y) {  
    if (x == 0)  
        return 0;  
  
}
```

# Aula 5- ALGORITMOS E COMPLEXIDADE

## TEMA 2 - RECURSIVIDADE

### Exercício 2 (ANALISANDO)

```
int mult (int x, int y) {  
    if (x == 0)  
        return 0;  
  
    if (x == 1)  
        return  
  
}
```

# Aula 5- ALGORITMOS E COMPLEXIDADE

## TEMA 2 - RECURSIVIDADE

### Exercício 2 (ANALISANDO)

```
int mult (int x, int y) {  
    if (x == 0)  
        return 0;  
  
    if (x == 1)  
        return y;  
  
}
```

# Aula 5- ALGORITMOS E COMPLEXIDADE

## TEMA 2 - RECURSIVIDADE

### Exercício 2 (ANALISANDO)

```
int mult (int x, int y) {  
    if (x == 0)  
        return 0;  
  
    if (x == 1)  
        return y;  
  
    return y + mult (x-1, y);  
}
```

# Aula 5- ALGORITMOS E COMPLEXIDADE

## TEMA 2 - RECURSIVIDADE

**Vamos verificar....**



# Aula 5- ALGORITMOS E COMPLEXIDADE

## TEMA 2 - RECURSIVIDADE

### Exercício 2 (ANALISANDO)

*Utilizando como exemplo  $x=3$  e  $y=5$*

```
mult (3, 5)
```

# Aula 5- ALGORITMOS E COMPLEXIDADE

## TEMA 2 - RECURSIVIDADE

### Exercício 2 (ANALISANDO)

*Utilizando como exemplo  $x=3$  e  $y=5$*

$3 - 1$

`mult(3, 5) = 5 + mult(2, 5)`

# Aula 5- ALGORITMOS E COMPLEXIDADE

## TEMA 2 - RECURSIVIDADE

### Exercício 2 (ANALISANDO)

*Utilizando como exemplo  $x=3$  e  $y=5$*

$$\begin{aligned} \text{mult}(3, 5) &= 5 + \text{mult}(2, 5) \\ &= 5 + 5 + \text{mult}(1, 5) \end{aligned}$$

Diagram illustrating the recursive steps for  $\text{mult}(3, 5)$ :

- Step 1:  $\text{mult}(3, 5) = 5 + \text{mult}(2, 5)$  (labeled  $3 - 1$ )
- Step 2:  $\text{mult}(2, 5) = 5 + \text{mult}(1, 5)$  (labeled  $2 - 1$ )

# Aula 5- ALGORITMOS E COMPLEXIDADE

## TEMA 2 - RECURSIVIDADE

### Exercício 2 (ANALISANDO)

*Utilizando como exemplo  $x=3$  e  $y=5$*

```
mult(3, 5) = 5 + mult(2, 5)
            = 5 + 5 + mult(1, 5)
            = 5 + 5 + 5
```

$3 - 1$

$2 - 1$

*Caso base: Se  $x == 1$  então retorno  $y$ .*

# Aula 5- ALGORITMOS E COMPLEXIDADE

## TEMA 2 - RECURSIVIDADE

### Exercício 2 (ANALISANDO)

*Utilizando como exemplo  $x=3$  e  $y=5$*

```
mult(3, 5) = 5 + mult(2, 5)
            = 5 + 5 + mult(1, 5)
            = 5 + 5 + 5
            = 15
```

$3 - 1$

$2 - 1$

*Caso base: Se  $x == 1$  então retorno  $y$*



# Aula 5- ALGORITMOS E COMPLEXIDADE

## TEMA 2 - RECURSIVIDADE

### Exercício 2 (C++)

```
#include<iostream>
#include<conio.h>
#include<locale>
using namespace std;

int mult(int x, int y);
int main() {
    setlocale(LC_ALL, "portuguese");
    int x, y;
    cout << "\n\n MULTIPLICAÇÃO (UTILIZANDO RECURSIVIDADE)";
    cout << "\n\n Digite um número inteiro x: ";
    cin >> x;
    cout << "\n Digite um número inteiro y: ";
    cin >> y;
    cout << "\n\n Resposta: " << mult(x,y);
    cout << "\n\n PRESSIONE QUALQUER <T E C L A> PARA FINALIZAR !!!";
    getch();
    return 0;
}

int mult (int x, int y) {
    if (x == 0) { return 0; }
    if (x == 1) { return y; }

    return y + mult (x-1,y);
}
```

# Aula 5- ALGORITMOS E COMPLEXIDADE

## TEMA 2 - RECURSIVIDADE

### Exercício 2 (C++)

```
#include<iostream>
```

```
#include<conio.h>
```

```
#include<locale.h>
```

```
using namespace std;
```

AO COMPILAR E EXECUTAR O CÓDIGO APARECERÁ:

```
in MULTIPLICAÇÃO (UTILIZANDO RECURSIVIDADE)
```

```
in Digite um número inteiro x: 4
```

```
Digite um número inteiro y: 5
```

```
Resposta: 20
```

```
} PRESSIONE QUALQUER <T E C L A> PARA FINALIZAR !!!
```

```
in -----  
Process exited after 6.624 seconds with return value 0
```

```
Pressione qualquer tecla para continuar. . . _
```

```
return y + mult (x-1,y);
```

```
}
```

# Aula 5- ALGORITMOS E COMPLEXIDADE

## TEMA 2 - RECURSIVIDADE

### Exercício 3

Construa um programa em C++ que faça a contagem dos “dígitos” de um determinado número digitado pelo usuário utilizando recursão.

#### Exemplo:

Se o usuário inserir o número 149, deve se ter a seguinte exibição na tela:

**O número digitado tem 3 dígitos.**

# Aula 5- ALGORITMOS E COMPLEXIDADE

## TEMA 2 - RECURSIVIDADE

### Exercício 3

```
#include<iostream>
#include<locale.h>
using namespace std;

int n_digitos(int n1);

int main()
{
    setlocale(LC_ALL,"portuguese");
    int n,contador;
    cout << "\n\t Programa (CONTA O NÚMEROS DE DIGITOS) :\n";
    cout << "\n Digite o numero: ";
    cin >> n;

    contador = n_digitos(n);

    cout << "\n\n Quantidade de digitos = " << contador;
    return 0;
}

int n_digitos(int n1)
{
    static int cont=0; // tem como função indicar
                       // que tal variável é permanente.

    if(n1>0)
    {
        cont++;
        cout << "\n Divide por 10 = " << n1;
        n_digitos(n1/10);
    }

    return cont;
}
```

**POR EXEMPLO:**

**cont++**

**n\_dígitos(45/10)**

*Digitando o número 45 o processo recursivo ocorre em linhas gerais conforme apresentado abaixo:*

**cont++**

**cont = 1**

**n\_dígitos(45/10)**

**cont++**

**cont = 2**

**n\_dígitos(4/10)**



# Aula 5- ALGORITMOS E COMPLEXIDADE

## TEMA 2 - RECURSIVIDADE

### Exercício 3

```
#include<iostream>
#include<locale.h>
using namespace std;

int n_digitos(int n1);
```

```
int main()
{
```

*AO COMPILAR E EXECUTAR O CÓDIGO APARECERÁ:*

```
se
in
    Programa (CONTA O NÚMEROS DE DIGITOS) :

    Digite o numero: 5600

    Divide por 10 = 5600
    Divide por 10 = 560
    Divide por 10 = 56
    Divide por 10 = 5

    Quantidade de digitos = 4

    -----
    Process exited after 2.557 seconds with return value 0
    Pressione qualquer tecla para continuar. . . _

}
return cont;
}
```