



# **ALGORITMOS E COMPLEXIDADE (ARA0174)**

## **AULA 6**

**DATA: 24/03/2022**



**AULA 6 - ALGORITMOS E COMPLEXIDADE**

**TEMA 2 - RECURSIVIDADE**

**CONTINUAÇÃO...**



**Estácio**

# Aula 6- ALGORITMOS E COMPLEXIDADE

## TEMA 2 - RECURSIVIDADE

### Exercício 4

Construa um programa em C++ que inverta uma “string” (conjunto de caracteres) utilizando uma função recursiva. (Use um vetor de 10 posições chamado “palavra” e uma variável tamanho “Para guardar o tamanho da palavra”).

**Exemplo:**

**Observação:**

**>> Digite uma palavra:** PROGRAMA

**>> Palavra invertida:** A M A R G O R P

# Aula 6- ALGORITMOS E COMPLEXIDADE

## TEMA 2 - RECURSIVIDADE

### Exercício 4

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

Strlen = Retorna o número de caracteres de uma string, não contando o caractere NULL.

```
void Inverter(char palavra[], int tamanho);
```

```
int main(){
```

```
    char palavra[10];
```

```
    int tamanho;
```

```
    cout << "\n Digite uma palavra: ";
```

```
    cin >> palavra;
```

```
    tamanho=strlen(palavra)-1;
```

```
    cout << "\n\n Palavra invertida: ";
```

```
    Inverter(palavra, tamanho);
```

```
    return 0;
```

```
}
```

```
void Inverter(char palavra[], int tamanho) {
```

```
    if (tamanho >= 0) {
```

```
        cout << " " << palavra[tamanho];
```

```
        Inverter(palavra, tamanho-1);
```

```
    }
```

```
    return;
```

```
}
```

**POR EXEMPLO:**

Palavra: LEO -> L E O \0  
          0 1 2 3

tamanho = strlen(palavra) - 1;  
tamanho = 3 - 1 = 2

palavra[2] = O  
Inverter(Recursivo, 2-1)

palavra[1] = E  
Inverter(Recursivo, 1-1)

palavra[0] = L  
Inverter(Recursivo, 0-1)

Tamanho = - 1  
(sai da recursividade devido tamanho >=0)



# Aula 6- ALGORITMOS E COMPLEXIDADE

## TEMA 2 - RECURSIVIDADE

### Exercício 4

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

Strlen = Retorna o número de caracteres de uma string, não contando o caractere NULL.

```
void Inverter(char palavra[], int tamanho);
```

```
int main(){
```

```
    char palavra[10];
    int tamanho;
    cout << "\n Digite uma palavra: ";
    cin >> palavra;
    tamanho=strlen(palavra)-1;
    cout << "\n\n Palavra invertida: ";
    Inverter(palavra, tamanho);
    return 0;
```

```
}
```

```
void Inverter(char palavra[], int tamanho) {
    if (tamanho >= 0) {
        cout << " " << palavra[tamanho];
        Inverter(palavra, tamanho-1);
    }
    return;
```

```
}
```

#### OUTRO EXEMPLO:

Palavra: CARLA -> C A R L A \0  
          0 1 2 3 4 5

tamanho = strlen(palavra) - 1;  
tamanho = 5 - 1 = 4

palavra[4] = A  
Inverter(Recursivo, 4-1)

palavra[3] = L  
Inverter(Recursivo, 3-1)

palavra[2] = R  
Inverter(Recursivo, 2-1)

palavra[1] = A  
Inverter(Recursivo, 1-1)

palavra[0] = C  
Inverter(Recursivo, 0-1)

Tamanho = - 1  
(sai da recursividade devido tamanho >=0)

# Aula 6- ALGORITMOS E COMPLEXIDADE

## Exercício 4

### TEMA 2 - RECURSIVIDADE

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

void Inverter(char palavra[], int tamanho);

int main(){

    char palavra[10];
    int tamanho;
    cout << "\n Digite uma palavra: ";
    cin >> palavra;
    tamanho=strlen(palavra)-1;
    cout << "\n\n Palavra invertida: ";
    Inverter(palavra, tamanho);
    return 0;
}

void Inverter(char palavra[], int tamanho) {
    if (tamanho >= 0) {
        cout << " " << palavra[tamanho];
        Inverter(palavra, tamanho-1);
    }
    return;
}
```

#### OUTRO EXEMPLO:

Palavra: RENATO -> R E N A T O \0  
                  0 1 2 3 4 5 6

tamanho = strlen(palavra) - 1;  
tamanho = 6 - 1 = 5

palavra[5] = O  
Inverter(Recursivo, 5-1)

palavra[4] = T  
Inverter(Recursivo, 4-1)

palavra[3] = A  
Inverter(Recursivo, 3-1)

palavra[2] = N  
Inverter(Recursivo, 2-1)

palavra[1] = E  
Inverter(Recursivo, 1-1)

palavra[0] = R  
Inverter(Recursivo, 0-1)

Tamanho = - 1  
(sai da recursividade devido tamanho >=0)

Strlen =  
Retorna o  
número de  
caracteres  
de uma  
string, não  
contando o  
caractere  
NULL.

# Aula 6- ALGORITMOS E COMPLEXIDADE

## TEMA 2 - RECURSIVIDADE

### Exercício 4

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

```
void Inverter(char palavra[], int tamanho);
```

```
int main(){
```

*AO COMPILAR E EXECUTAR O CÓDIGO APARECERÁ:*

0 1 2 3

Digite uma palavra: Recursivo

Palavra invertida: o v i s r u c e R

POR EXEMPLO:

Inverter(Recursivo, 1-1)

palavra[0] = L

Inverter(Recursivo, 0-1)

Tamanho = - 1

(sai da recursividade devido tamanho >=0)

```
}

void Inverter(char palavra[], int tamanho) {
    if (tamanho >= 0) {
        cout << " " << palavra[tamanho];
        Inverter(palavra, tamanho-1);
    }
    return;
}
```

# Aula 6- ALGORITMOS E COMPLEXIDADE

## TEMA 2 - RECURSIVIDADE

### Exercício 5

Construa um programa em C++ que, através de uma função recursiva, realize um cálculo exponencial de  $x$  elevado a  $y$ .

#### Exemplo:

**Digite o número inteiro x: 2**

**Digite o número inteiro y: 3**

**Resposta: x elevado a y é: 8**



# Aula 6- ALGORITMOS E COMPLEXIDADE

## TEMA 2 - RECURSIVIDADE

### Exercício 5

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

int expo (int x, **AO COMPILAR E EXECUTAR O CÓDIGO APARECERÁ:**

```
int main(void) {
```

**POR EXEMPLO:**

```
    *** Exponencial de x elevado a y ***
```

```
    Digite o numero inteiro x:2
```

```
    Digite o numero inteiro y:3
```

```
    Resposta: x elevado a y é: 8
```

```
int expo
if (y == 0) {
    return 1; }

if (y == 1) {
    return x; }

return x*expo(x,y-1);
}
```

**return 2 \* 2 \* 2 \* 1**

**return 8**

# Aula 6- ALGORITMOS E COMPLEXIDADE

## TEMA 2 - RECURSIVIDADE

### Exercício 5

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

int expo (int x, int y);

int main(void) {
    setlocale(LC_ALL,"portuguese");
    int x, y, e;
    cout << "\n *** Exponencial de x elevado a y ***";
    cout << "\n\n Digite o numero inteiro x:";
    cin >> x;
    cout << "\n Digite o numero inteiro y:";
    cin >> y;
    if (y < 0) {
        cout << "y tem que ser maior ou igual a zero!!";
        getch(); }
    else {
        e=expo(x,y);
        cout << "\n\n Resposta: x elevado a y é: " << e;
        getch();}
}

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

    if (y == 1) {
        return x; }

    return x*expo(x,y-1);
}
```

#### POR EXEMPLO:

Digite o número inteiro x: 2

Digite o número inteiro y: 3

Iniciando:

Se y = 0 então return 1;

Se y = 1 então return x;

1ª. passagem  
return x \* expo(x, y - 1)

return 2 \* expo(2, 3 - 1)  
2

# Aula 6- ALGORITMOS E COMPLEXIDADE

## TEMA 2 - RECURSIVIDADE

### Exercício 5

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

int expo (int x, int y);

int main(void) {
    setlocale(LC_ALL, "portuguese");
    int x, y, e;
    cout << "\n *** Exponencial de x elevado a y ***";
    cout << "\n\n Digite o numero inteiro x:";
    cin >> x;
    cout << "\n Digite o numero inteiro y:";
    cin >> y;
    if (y < 0) {
        cout << "y tem que ser maior ou igual a zero!!";
        getch(); }
    else {
        e=expo(x,y);
        cout << "\n\n Resposta: x elevado a y é: " << e;
        getch();}
}

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

    if (y == 1) {
        return x; }

    return x*expo(x,y-1);
}
```

#### POR EXEMPLO:

Digite o número inteiro x: 2

Digite o número inteiro y: 2

Iniciando:

Se y = 0 então return 1;

Se y = 1 então return x;

2ª. passagem

return x \* expo(x, y - 1)

return 2 \* 2 \* expo(2, 2 - 1)

1

# Aula 6- ALGORITMOS E COMPLEXIDADE

## TEMA 2 - RECURSIVIDADE

### Exercício 5

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

int expo (int x, int y);

int main(void) {
    setlocale(LC_ALL, "portuguese");
    int x, y, e;
    cout << "\n *** Exponencial de x elevado a y ***";
    cout << "\n\n Digite o numero inteiro x:";
    cin >> x;
    cout << "\n Digite o numero inteiro y:";
    cin >> y;
    if (y < 0) {
        cout << "y tem que ser maior ou igual a zero!!";
        getch(); }
    else {
        e=expo(x,y);
        cout << "\n\n Resposta: x elevado a y é: " << e;
        getch();}
}

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

    if (y == 1) {
        return x; }

    return x*expo(x,y-1);
}
```

#### POR EXEMPLO:

Digite o número inteiro x: 2

Digite o número inteiro y: 1

Iniciando:

**Se y = 1 então return x;**  
**(ENTRA NESSA CONDIÇÃO).**

3ª. passagem  
return x \* expo(x, y - 1)

return 2 \* 2 \* 2



# Aula 6- ALGORITMOS E COMPLEXIDADE

## TEMA 2 - RECURSIVIDADE

### Exercício 5

Construa um programa em C++ que, através de uma função recursiva, realize um cálculo exponencial de  $x$  elevado a  $y$ .

#### Exemplo:

**Digite o número inteiro x: 2**

**Digite o número inteiro y: 3**

**Resposta: x elevado a y é: 8**

# Aula 6- ALGORITMOS E COMPLEXIDADE

## TEMA 2 - RECURSIVIDADE

### Exercício 5

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

```
int expo (int x,
```

*AO COMPILAR E EXECUTAR O CÓDIGO APARECERÁ:*

```
int main(void) {
```

**POR EXEMPLO:**

```
*** Exponencial de x elevado a y ***
```

```
Digite o numero inteiro x:2
```

```
Digite o numero inteiro y:3
```

```
Resposta: x elevado a y é: 8
```

**return 2 \* 2 \* 2 \* 1**

**return 8**

```
int expo
if (y == 0) {
    return 1; }

if (y == 1) {
    return x; }

return x*expo(x,y-1);
}
```

# Aula 6- ALGORITMOS E COMPLEXIDADE

## TEMA 2 - RECURSIVIDADE

### Exercício 5

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

int expo (int x, int y);

int main(void) {
    setlocale(LC_ALL,"portuguese");
    int x, y, e;
    cout << "\n *** Exponencial de x elevado a y ***";
    cout << "\n\n Digite o numero inteiro x:";
    cin >> x;
    cout << "\n Digite o numero inteiro y:";
    cin >> y;
    if (y < 0) {
        cout << "y tem que ser maior ou igual a zero!!";
        getch(); }
    else {
        e=expo(x,y);
        cout << "\n\n Resposta: x elevado a y é: " << e;
        getch();}
}

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

    if (y == 1) {
        return x; }

    return x*expo(x,y-1);
}
```

#### POR EXEMPLO:

Digite o número inteiro x: 2

Digite o número inteiro y: 3

Iniciando:

Se y = 0 então return 1;

Se y = 1 então return x;

1ª. passagem  
return x \* expo(x, y - 1)

return 2 \* expo(2, 3 - 1)  
2

# Aula 6- ALGORITMOS E COMPLEXIDADE

## TEMA 2 - RECURSIVIDADE

### Exercício 5

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

int expo (int x, int y);

int main(void) {
    setlocale(LC_ALL, "portuguese");
    int x, y, e;
    cout << "\n *** Exponencial de x elevado a y ***";
    cout << "\n\n Digite o numero inteiro x:";
    cin >> x;
    cout << "\n Digite o numero inteiro y:";
    cin >> y;
    if (y < 0) {
        cout << "y tem que ser maior ou igual a zero!!";
        getch(); }
    else {
        e=expo(x,y);
        cout << "\n\n Resposta: x elevado a y é: " << e;
        getch();}
}

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

    if (y == 1) {
        return x; }

    return x*expo(x,y-1);
}
```

#### POR EXEMPLO:

Digite o número inteiro x: 2

Digite o número inteiro y: 2

Iniciando:

Se y = 0 então return 1;

Se y = 1 então return x;

2ª. passagem

return x \* expo(x, y - 1)

return 2 \* 2 \* expo(2, 2 - 1)

1



# Aula 6- ALGORITMOS E COMPLEXIDADE

## TEMA 2 - RECURSIVIDADE

### Exercício 5

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

int expo (int x, int y);

int main(void) {
    setlocale(LC_ALL,"portuguese");
    int x, y, e;
    cout << "\n *** Exponencial de x elevado a y ***";
    cout << "\n\n Digite o numero inteiro x:";
    cin >> x;
    cout << "\n Digite o numero inteiro y:";
    cin >> y;
    if (y < 0) {
        cout << "y tem que ser maior ou igual a zero!!";
        getch(); }
    else {
        e=expo(x,y);
        cout << "\n\n Resposta: x elevado a y é: " << e;
        getch();}
}

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

    if (y == 1) {
        return x; }

    return x*expo(x,y-1);
}
```

#### POR EXEMPLO:

Digite o número inteiro x: 2

Digite o número inteiro y: 1

Iniciando:

Se y = 0 então return 1;

**Se y = 1 então return x;**  
(ENTRA NESSA CONDIÇÃO).

3ª. passagem

return x \* expo(x, y - 1)

return 2 \* 2 \* 2 \* expo(2, 1 - 1)

0

# Aula 6- ALGORITMOS E COMPLEXIDADE

## TEMA 2 - RECURSIVIDADE

### Exercício 5

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

int expo (int x, int y);

int main(void) {
    setlocale(LC_ALL, "portuguese");
    int x, y, e;
    cout << "\n *** Exponencial de x elevado a y ***";
    cout << "\n\n Digite o numero inteiro x:";
    cin >> x;
    cout << "\n Digite o numero inteiro y:";
    cin >> y;
    if (y < 0) {
        cout << "y tem que ser maior ou igual a zero!!";
        getch(); }
    else {
        e=expo(x,y);
        cout << "\n\n Resposta: x elevado a y é: " << e;
        getch();}
}

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

    if (y == 1) {
        return x; }

    return x*expo(x,y-1);
}
```

#### POR EXEMPLO:

Digite o número inteiro x: 2

Digite o número inteiro y: 1

Iniciando:

Se y = 0 então return 1;

**Se y = 1 então return x;**  
(ENTRA NESSA CONDIÇÃO).

3ª. passagem  
return x \* expo(x, y - 1)

return 2 \* 2 \* 2 \* expo(2, **1 - 1**)  
0

# Aula 6- ALGORITMOS E COMPLEXIDADE

## TEMA 2 - RECURSIVIDADE

### Exercício 5

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

int expo (int x, int y);

int main(void) {
    setlocale(LC_ALL, "portuguese");
    int x, y, e;
    cout << "\n *** Exponencial de x elevado a y ***";
    cout << "\n\n Digite o numero inteiro x:";
    cin >> x;
    cout << "\n Digite o numero inteiro y:";
    cin >> y;
    if (y < 0) {
        cout << "y tem que ser maior ou igual a zero!!";
        getch(); }
    else {
        e=expo(x,y);
        cout << "\n\n Resposta: x elevado a y e: " << e;
        getch();}
}

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

    if (y == 1) {
        return x; }

    return x*expo(x,y-1);
}
```

#### POR EXEMPLO:

Digite o número inteiro x: 2

Digite o número inteiro y: 0

Iniciando:

**Se y = 0 então return 1;**  
*(ENTRA NESTA CONDIÇÃO).*

Se y = 1 então return x;

**4ª. passagem**  
**return x \* expo(x, y - 1)**

**return 2 \* 2 \* 2 \* 1**

# Aula 6- ALGORITMOS E COMPLEXIDADE

## TEMA 2 - RECURSIVIDADE

### Exercício 5

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

int expo (int x, int y);

int main(void) {
    setlocale(LC_ALL, "portuguese");
    int x, y, e;
    cout << "\n *** Exponencial de x elevado a y ***";
    cout << "\n\n Digite o numero inteiro x:";
    cin >> x;
    cout << "\n Digite o numero inteiro y:";
    cin >> y;
    if (y < 0) {
        cout << "y tem que ser maior ou igual a zero!!";
        getch(); }
    else {
        e=expo(x,y);
        cout << "\n\n Resposta: x elevado a y é: " << e;
        getch();}
}

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

    if (y == 1) {
        return x; }

    return x*expo(x,y-1);
}
```

#### POR EXEMPLO:

Digite o número inteiro x: 2

Digite o número inteiro y: 0

Iniciando:

**Se y = 0 então return 1;**  
**(ENTRA NESSA CONDIÇÃO).**

Se y = 1 então return x;

**4ª. passagem**  
**return x \* expo(x, y - 1)**

**return 2 \* 2 \* 2 \* 1**



# Aula 6- ALGORITMOS E COMPLEXIDADE

## TEMA 2 - RECURSIVIDADE

### Exercício 6

Construa um programa em C++ que, através de uma função recursiva, realize a soma de todos os números ímpares inteiros a partir do número digitado pelo teclado pelo usuário até 1.

**DICA:**

**Na análise condicional utilize a seguinte condição (valor % 2 == 0).**

# Aula 6- ALGORITMOS E COMPLEXIDADE

## TEMA 2 - RECURSIVIDADE

### Exercício 6

#### POR EXEMPLO:

Digite o número inteiro valor: 3

Se  $(3 \geq 1)$  então

Se o resto da divisão for 0 ( $3 \% 2 == 0$ ) é PAR. **(NÃO É).**

Como 3 não é PAR prossigo com o código:  
 $\text{return } 3 + s\_impar(3-1)$

2

Se  $(2 \geq 1)$  então

Se o resto da divisão for 0 ( $2 \% 2 == 0$ ) é PAR. **(SIM É).**

Como 2 É PAR então faço  $s\_impar(2-1)$ .

1

Se  $(1 \geq 1)$  então

Se o resto da divisão for 0 ( $1 \% 2 == 0$ ) é PAR. **(NÃO É).**

Como 1 não é PAR prossigo com o código:  
 $\text{return } 3 + 1 + s\_impar(1-1)$

0

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

int s_impar(int valor);

int main () {
    setlocale(LC_ALL, "portuguese");
    int valor;
    cout << "\n\n Digite um número inteiro: ";
    cin >> valor;
    cout << "\n Números ímpares:";
    cout << "\n\n Resultado: " << s_impar(valor);
    getch();
}

int s_impar(int valor) {
    if (valor >=1) {
        if(valor % 2 == 0) {
            return s_impar(valor-1);
        }
        cout << " " << valor;
        return valor + s_impar(valor-1);
    }
}
```

# Aula 6- ALGORITMOS E COMPLEXIDADE

## TEMA 2 - RECURSIVIDADE

### Exercício 6

POR EXEMPLO:

Digite o número inteiro valor: 3

AO COMPILAR E EXECUTAR O CÓDIGO APARECERÁ:

Digite um número inteiro: 5

Números ímpares: 5 3 1

Resultado: 9

PAR. (NAO E).

Como 1 não é PAR prossigo com o código:  
return 3 + 1 + s\_impar( 1-1 )

0

# Aula 6- ALGORITMOS E COMPLEXIDADE

## TEMA 2 - RECURSIVIDADE

### Exercício 6

#### POR EXEMPLO:

Digite o número inteiro valor: 3

Se  $(3 \geq 1)$  então

Se o resto da divisão for 0 ( $3 \% 2 == 0$ ) é PAR. **(NÃO É).**

Como 3 não é PAR prossigo com o código:  
 $\text{return } 3 + \text{s\_impar}(3-1)$

2

Se  $(2 \geq 1)$  então

Se o resto da divisão for 0 ( $2 \% 2 == 0$ ) é PAR. **(SIM É).**

Como 2 É PAR então faço  $\text{s\_impar}(2-1)$ .

1

Se  $(1 \geq 1)$  então

Se o resto da divisão for 0 ( $1 \% 2 == 0$ ) é PAR. **(NÃO É).**

Como 1 não é PAR prossigo com o código:  
 $\text{return } 3 + 1 + \text{s\_impar}(1-1)$

0

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

int s_impar(int valor);

int main () {
    setlocale(LC_ALL, "portuguese");
    int valor;
    cout << "\n\n Digite um número inteiro: ";
    cin >> valor;
    cout << "\n Números ímpares:";
    cout << "\n\n Resultado: " << s_impar(valor);
    getch();
}

int s_impar(int valor) {
    if (valor >=1) {
        if(valor % 2 == 0) {
            return s_impar(valor-1);
        }
        cout << " " << valor;
        return valor + s_impar(valor-1);
    }
}
```



# Aula 6- ALGORITMOS E COMPLEXIDADE

## TEMA 2 - RECURSIVIDADE

### Exercício 6

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

int s_impar(int valor);

int main () {
    setlocale(LC_ALL, "portuguese");
    int valor;
    cout << "\n\n Digite um número inteiro: ";
    cin >> valor;
    cout << "\n Números ímpares:";
    cout << "\n\n Resultado: " << s_impar(valor);
    getch();
}

int s_impar(int valor) {
    if (valor >=1) {
        if(valor % 2 == 0) {
            return s_impar(valor-1);
        }
        cout << " " << valor;
        return valor + s_impar(valor-1);
    }
}
```

#### POR EXEMPLO:

Digite o número inteiro valor: 3

Se  $(3 \geq 1)$  então

Se o resto da divisão for 0 ( $3 \% 2 == 0$ ) é PAR. **(NÃO É).**

Como 3 não é PAR prossigo com o código:  
return 3 + s\_impar( 3-1 )

2

Se  $(2 \geq 1)$  então

Se o resto da divisão for 0 ( $2 \% 2 == 0$ ) é PAR. **(SIM É).**

Como 2 É PAR então faço s\_impar( 2-1 ).

1

Se  $(1 \geq 1)$  então

Se o resto da divisão for 0 ( $1 \% 2 == 0$ ) é PAR. **(NÃO É).**

Como 1 não é PAR prossigo com o código:  
return 3 + 1 + s\_impar( 1-1 )

0

# Aula 6- ALGORITMOS E COMPLEXIDADE

## TEMA 2 - RECURSIVIDADE

### Exercício 6

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

int s_impar(int valor);

int main () {
    setlocale(LC_ALL, "portuguese");
    int valor;
    cout << "\n\n Digite um número inteiro: ";
    cin >> valor;
    cout << "\n Números ímpares: ";
    cout << "\n\n Resultado: " << s_impar(valor);
    getch();
}

int s_impar(int valor) {
    if (valor >=1) {
        if(valor % 2 == 0) {
            return s_impar(valor-1);
        }
        cout << " " << valor;
        return valor + s_impar(valor-1);
    }
}
```

#### POR EXEMPLO:

Digite o número inteiro valor: 3

Se  $(3 \geq 1)$  então

Se o resto da divisão for 0 ( $3 \% 2 == 0$ ) é PAR. **(NÃO É).**

Como 3 não é PAR prossigo com o código:  
return 3 + s\_impar( 3-1 )

2

Se  $(2 \geq 1)$  então

Se o resto da divisão for 0 ( $2 \% 2 == 0$ ) é PAR. **(SIM É).**

Como 2 É PAR então faço s\_impar( 2-1 ).

1

Se  $(1 \geq 1)$  então

Se o resto da divisão for 0 ( $1 \% 2 == 0$ ) é PAR. **(NÃO É).**

Como 1 não é PAR prossigo com o código:  
return 3 + 1 + s\_impar( 1-1 )

0

# Aula 6- ALGORITMOS E COMPLEXIDADE

## TEMA 2 - RECURSIVIDADE

### Exercício 6

#### POR EXEMPLO:

Digite o número inteiro valor: 3

Se  $(3 \geq 1)$  então

Se o resto da divisão for 0 ( $3 \% 2 == 0$ ) é PAR. **(NÃO É).**

Como 3 não é PAR prossigo com o código:

return 3 + s\_impar( 3-1 )

2

Se  $(2 \geq 1)$  então

Se o resto da divisão for 0 ( $2 \% 2 == 0$ ) é PAR. **(SIM É).**

Como 2 É PAR então faço s\_impar( 2-1 ).

1

Se  $(1 \geq 1)$  então

Se o resto da divisão for 0 ( $1 \% 2 == 0$ ) é PAR. **(NÃO É).**

Como 1 não é PAR prossigo com o código:

return 3 + 1 + s\_impar( 1-1 )

0

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

int s_impar(int valor);

int main () {
    setlocale(LC_ALL, "portuguese");
    int valor;
    cout << "\n\n Digite um número inteiro: ";
    cin >> valor;
    cout << "\n Números ímpares:";
    cout << "\n\n Resultado: " << s_impar(valor);
    getch();
}

int s_impar(int valor) {
    if (valor >=1) {
        if(valor % 2 == 0) {
            return s_impar(valor-1);
        }
        cout << " " << valor;
        return valor + s_impar(valor-1);
    }
}
```



# Aula 6- ALGORITMOS E COMPLEXIDADE

## TEMA 2 - RECURSIVIDADE

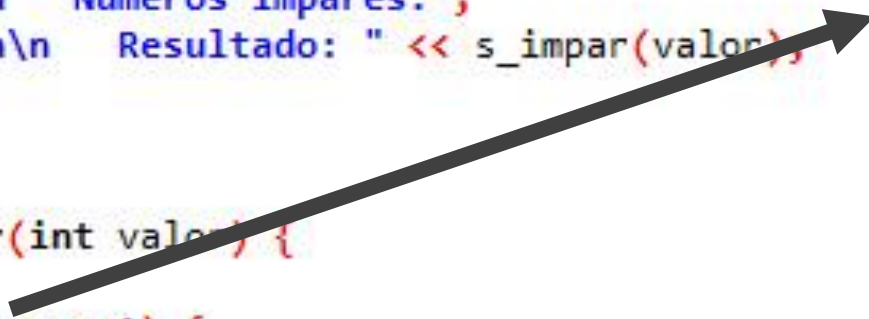
### Exercício 6

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

int s_impar(int valor);

int main () {
    setlocale(LC_ALL, "portuguese");
    int valor;
    cout << "\n\n Digite um número inteiro: ";
    cin >> valor;
    cout << "\n Números ímpares:";
    cout << "\n\n Resultado: " << s_impar(valor);
    getch();
}

int s_impar(int valor) {
    if (valor >=1) {
        if(valor % 2 == 0) {
            return s_impar(valor-1);
        }
        cout << " " << valor;
        return valor + s_impar(valor-1);
    }
}
```



#### POR EXEMPLO:

Digite o número inteiro valor: 3

Se  $(3 \geq 1)$  então

Se o resto da divisão for 0 ( $3 \% 2 == 0$ ) é PAR. **(NÃO É).**

Como 3 não é PAR prossigo com o código:  
 $\text{return } 3 + \text{s\_impar}(3-1)$

2

Se  $(2 \geq 1)$  então

Se o resto da divisão for 0 ( $2 \% 2 == 0$ ) é PAR. **(SIM É).**

Como 2 É PAR então faço  $\text{s\_impar}(2-1)$ .

1

Se  $(1 \geq 1)$  então

Se o resto da divisão for 0 ( $1 \% 2 == 0$ ) é PAR. **(NÃO É).**

Como 1 não é PAR prossigo com o código:  
 $\text{return } 3 + 1 + \text{s\_impar}(1-1)$

0



# Aula 6- ALGORITMOS E COMPLEXIDADE

## TEMA 2 - RECURSIVIDADE

### Exercício 6

#### POR EXEMPLO:

Digite o número inteiro valor: 3

Se  $(3 \geq 1)$  então

Se o resto da divisão for 0 ( $3 \% 2 == 0$ ) é PAR. **(NÃO É).**

Como 3 não é PAR prossigo com o código:  
 $\text{return } 3 + \text{s\_impar}(3-1)$

2

Se  $(2 \geq 1)$  então

Se o resto da divisão for 0 ( $2 \% 2 == 0$ ) é PAR. **(SIM É).**

Como 2 É PAR então faço  $\text{s\_impar}(2-1)$ .

1

Se  $(1 \geq 1)$  então

Se o resto da divisão for 0 ( $1 \% 2 == 0$ ) é PAR. **(NÃO É).**

Como 1 não é PAR prossigo com o código:  
 $\text{return } 3 + 1 + \text{s\_impar}(1-1)$

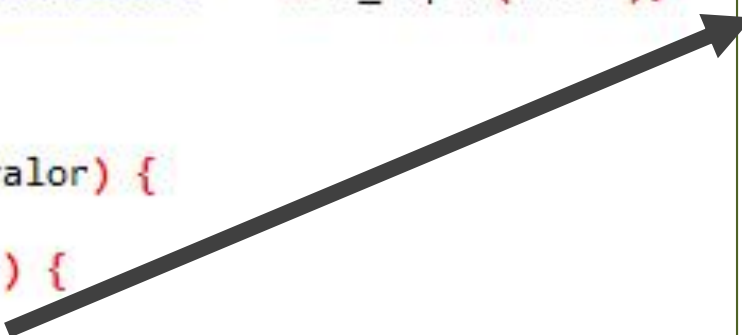
0

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

int s_impar(int valor);

int main () {
    setlocale(LC_ALL, "portuguese");
    int valor;
    cout << "\n\n Digite um número inteiro: ";
    cin >> valor;
    cout << "\n Números ímpares:";
    cout << "\n\n Resultado: " << s_impar(valor);
    getch();
}

int s_impar(int valor) {
    if (valor >=1) {
        if(valor % 2 == 0) {
            return s_impar(valor-1);
        }
        cout << " " << valor;
        return valor + s_impar(valor-1);
    }
}
```



# Aula 6- ALGORITMOS E COMPLEXIDADE

## TEMA 2 - RECURSIVIDADE

### Exercício 6

#### POR EXEMPLO:

Digite o número inteiro valor: 3

Se  $(3 \geq 1)$  então

Se o resto da divisão for 0 ( $3 \% 2 == 0$ ) é PAR. **(NÃO É).**

Como 3 não é PAR prossigo com o código:  
 $\text{return } 3 + \text{s\_impar}(3-1)$

2

Se  $(2 \geq 1)$  então

Se o resto da divisão for 0 ( $2 \% 2 == 0$ ) é PAR. **(SIM É).**

Como 2 É PAR então faço  $\text{s\_impar}(2-1)$ .

1

Se  $(1 \geq 1)$  então

Se o resto da divisão for 0 ( $1 \% 2 == 0$ ) é PAR. **(NÃO É).**

Como 1 não é PAR prossigo com o código:  
 $\text{return } 3 + 1 + \text{s\_impar}(1-1)$

0

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

int s_impar(int valor);

int main () {
    setlocale(LC_ALL, "portuguese");
    int valor;
    cout << "\n\n Digite um número inteiro: ";
    cin >> valor;
    cout << "\n Números ímpares:";
    cout << "\n\n Resultado: " << s_impar(valor);
    getch();
}

int s_impar(int valor) {
    if (valor >=1) {
        if(valor % 2 == 0) {
            return s_impar(valor-1);
        }
        cout << " " << valor;
        return valor + s_impar(valor-1);
    }
}
```

# Aula 6- ALGORITMOS E COMPLEXIDADE

## TEMA 2 - RECURSIVIDADE

### Exercício 6

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

int s_impar(int valor);

int main () {
    setlocale(LC_ALL, "portuguese");
    int valor;
    cout << "\n\n Digite um número inteiro: ";
    cin >> valor;
    cout << "\n Números ímpares:";
    cout << "\n\n Resultado: " << s_impar(valor);
    getch();
}

int s_impar(int valor) {
    if (valor >=1) {
        if(valor % 2 == 0) {
            return s_impar(valor-1);
        }
        cout << " " << valor;
        return valor + s_impar(valor-1);
    }
}
```

#### POR EXEMPLO:

Digite o número inteiro valor: 3

Se  $(3 \geq 1)$  então

Se o resto da divisão for 0 ( $3 \% 2 == 0$ ) é PAR. **(NÃO É).**

Como 3 não é PAR prossigo com o código:  
return 3 + s\_impar( 3-1 )

2

Se  $(2 \geq 1)$  então

Se o resto da divisão for 0 ( $2 \% 2 == 0$ ) é PAR. **(SIM É).**

Como 2 É PAR então faço s\_impar( 2-1 ).

1

Se  $(1 \geq 1)$  então

Se o resto da divisão for 0 ( $1 \% 2 == 0$ ) é PAR. **(NÃO É).**

Como 1 não é PAR prossigo com o código:  
return 3 + 1 + s\_impar( 1-1 )

0



# Aula 6- ALGORITMOS E COMPLEXIDADE

## TEMA 2 - RECURSIVIDADE

### Exercício 6

#### POR EXEMPLO:

Digite o número inteiro valor: 3

Se  $(3 \geq 1)$  então

Se o resto da divisão for 0 ( $3 \% 2 == 0$ ) é PAR. **(NÃO É).**

Como 3 não é PAR prossigo com o código:  
 $\text{return } 3 + \text{s\_impar}(3-1)$

2

Se  $(2 \geq 1)$  então

Se o resto da divisão for 0 ( $2 \% 2 == 0$ ) é PAR. **(SIM É).**

Como 2 É PAR então faço  $\text{s\_impar}(2-1)$ .

1

Se  $(1 \geq 1)$  então

Se o resto da divisão for 0 ( $1 \% 2 == 0$ ) é PAR. **(NÃO É).**

Como 1 não é PAR prossigo com o código:  
 $\text{return } 3 + 1 + \text{s\_impar}(1-1)$

0

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

int s_impar(int valor);

int main () {
    setlocale(LC_ALL, "portuguese");
    int valor;
    cout << "\n\n Digite um número inteiro: ";
    cin >> valor;
    cout << "\n Números ímpares:";
    cout << "\n\n Resultado: " << s_impar(valor);
    getch();
}

int s_impar(int valor) {
    if (valor >=1) {
        if(valor % 2 == 0) {
            return s_impar(valor-1);
        }
        cout << " " << valor;
        return valor + s_impar(valor-1);
    }
}
```



# Aula 6- ALGORITMOS E COMPLEXIDADE

## TEMA 2 - RECURSIVIDADE

### Exercício 6

#### POR EXEMPLO:

Digite o número inteiro valor: 3

Se  $(3 \geq 1)$  então

Se o resto da divisão for 0 ( $3 \% 2 == 0$ ) é PAR. **(NÃO É).**

Como 3 não é PAR prossigo com o código:  
 $\text{return } 3 + \text{s\_impar}(3-1)$

2

Se  $(2 \geq 1)$  então

Se o resto da divisão for 0 ( $2 \% 2 == 0$ ) é PAR. **(SIM É).**

Como 2 É PAR então faço  $\text{s\_impar}(2-1)$ .

1

Se  $(1 \geq 1)$  então

Se o resto da divisão for 0 ( $1 \% 2 == 0$ ) é PAR. **(NÃO É).**

Como 1 não é PAR prossigo com o código:  
 $\text{return } 3 + 1 + \text{s\_impar}(1-1)$

0

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

int s_impar(int valor);

int main () {
    setlocale(LC_ALL, "portuguese");
    int valor;
    cout << "\n\n Digite um número inteiro: ";
    cin >> valor;
    cout << "\n Números ímpares:";
    cout << "\n\n Resultado: " << s_impar(valor);
    getch();
}

int s_impar(int valor) {
    if (valor >=1) {
        if(valor % 2 == 0) {
            return s_impar(valor-1);
        }
        cout << " " << valor;
        return valor + s_impar(valor-1);
    }
}
```

# Aula 6- ALGORITMOS E COMPLEXIDADE

## TEMA 2 - RECURSIVIDADE

### **Exercício 7** (COMO FAZ A SOMA DOS PARES?)

Construa um programa em C++ que, através de uma função recursiva, realize a soma de todos os números pares inteiros a partir do número digitado pelo teclado pelo usuário até 1.

**PERGUNTA:**

**Na análise condicional qual condição será utilizada?**

# Aula 6- ALGORITMOS E COMPLEXIDADE

## TEMA 2 - RECURSIVIDADE

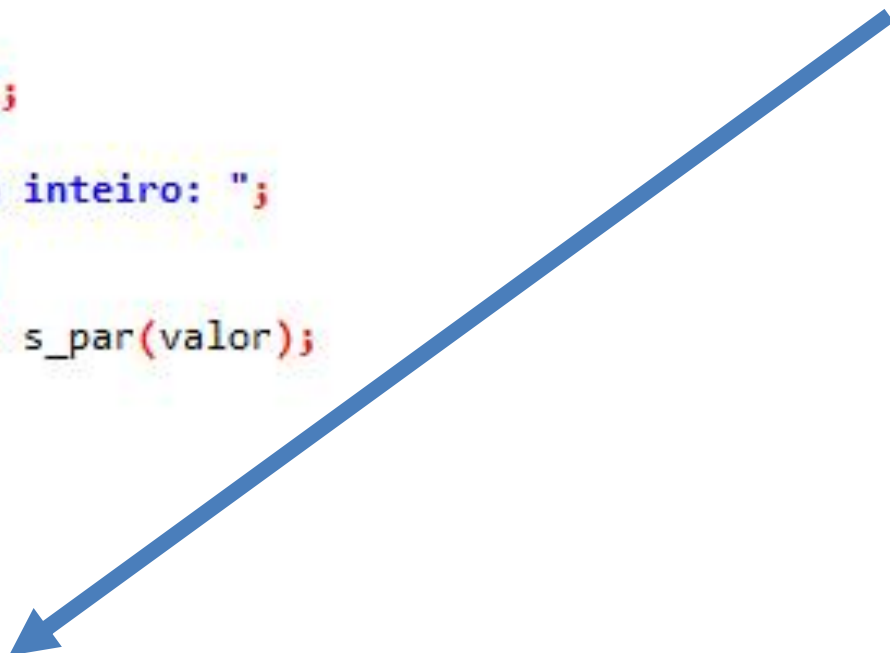
### Exercício 7

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

int s_par(int valor);

int main () {
    setlocale(LC_ALL, "portuguese");
    int valor;
    cout << "\n\n    Digite um número inteiro: ";
    cin >> valor;
    cout << "\n    Números pares:";
    cout << "\n\n    Resultado: " << s_par(valor);
    getch();
}

int s_par(int valor) {
    if (valor >= 1) {
        if(valor % 2 != 0) {
            return s_par(valor-1);
        }
        cout << " " << valor;
        return valor + s_par(valor-1);
    }
}
```



# Aula 6- ALGORITMOS E COMPLEXIDADE

## TEMA 2 - RECURSIVIDADE

### Exercício 7

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

```
int s_par(int valor);
```

```
int main () {  
    setlocale(LC_ALL, "portuguese");
```

*AO COMPILAR E EXECUTAR O CÓDIGO APARECERÁ:*

Digite um número inteiro: 8

Números pares: 8 6 4 2

Resultado: 20

```
if (valor >= 1) {
```

```
    if(valor % 2 != 0) {  
        return s_par(valor-1);  
    }
```

```
    cout << " " << valor;  
    return valor + s_par(valor-1);  
}
```

```
}
```



# Aula 6- ALGORITMOS E COMPLEXIDADE

## TEMA 3 - ALGORITMOS DE ORDENAÇÃO AVANÇADOS

## TEMA 3 - ALGORITMOS DE ORDENAÇÃO AVANÇADOS

# Aula 6- ALGORITMOS E COMPLEXIDADE

## TEMA 3 - ALGORITMOS DE ORDENAÇÃO AVANÇADOS



Ordenação

# Aula 6- ALGORITMOS E COMPLEXIDADE

## TEMA 3 - ALGORITMOS DE ORDENAÇÃO AVANÇADOS

### Métodos Simples

- mais eficientes para **pequenos conjuntos**;
- usam **muitas comparações**;
- códigos **pequenos**;
- códigos de **fácil entendimento**;

Exemplos:

1. *Bubble Sort (Ordenação Bolha ou por troca);*
2. *Insertion Sort(Ordenação por inserção);*
3. *Selection Sort(Ordenação por seleção).*

### ATENÇÃO:

O bubble sort, o insert sort e o selection sort tem a mesma complexidade computacional, porém, isto não quer dizer que todos executem ao mesmo tempo para a mesma instância.

# Aula 6- ALGORITMOS E COMPLEXIDADE

## TEMA 3 - ALGORITMOS DE ORDENAÇÃO AVANÇADOS

### Bubble Sort (Ordenação Bolha)

O princípio básico desse método é trocar de posições **todas as vezes** que forem encontrados **valores de posições adjacentes fora de ordem**.



# Aula 6- ALGORITMOS E COMPLEXIDADE

## TEMA 3 - ALGORITMOS DE ORDENAÇÃO AVANÇADOS

### EXEMPLO DE COMO O PROCESSO É REALIZADO NA ORDENAÇÃO BOLHA (PASSO A PASSO)

1. Neste exemplo é utilizado o método de ordenação bolha, para colocar em ordem em crescente os conjuntos de valores [7, 1, 3, 2, 6, 4], sendo mostrado o números de COMPARAÇÕES E TROCAS necessárias.

[7, 1, 3, 2, 6, 4] = COMPARA 1 E [1, 7, 3, 2, 6, 4] TROCA 1

[1, 7, 3, 2, 6, 4] = COMPARA 2 E [1, 3, 7, 2, 6, 4] TROCA 2

[1, 3, 7, 2, 6, 4] = COMPARA 3 E [1, 3, 2, 7, 6, 4] TROCA 3

[1, 3, 2, 7, 6, 4] = COMPARA 4 E [1, 3, 2, 6, 7, 4] TROCA 4

[1, 3, 2, 6, 7, 4] = COMPARA 5 E [1, 3, 2, 6, 4, 7] TROCA 5

[1, 3, 2, 6, 4, 7] = COMPARA 6 E [1, 3, 2, 6, 4, 7] NÃO TROCA

# Aula 6- ALGORITMOS E COMPLEXIDADE

## TEMA 3 - ALGORITMOS DE ORDENAÇÃO AVANÇADOS

### EXEMPLO DE COMO O PROCESSO É REALIZADO NA ORDENAÇÃO BOLHA (PASSO A PASSO)

1. Neste exemplo é utilizado o método de ordenação bolha, para colocar em ordem em crescente os conjuntos de valores [7, 1, 3, 2, 6, 4], sendo mostrado o números de COMPARAÇÕES E TROCAS necessárias.

CONTINUANDO...

[1, 3, 2, 6, 4, 7] = COMPARA 7 E [1, 2, 3, 6, 4, 7] TROCA 6

[1, 2, 3, 6, 4, 7] = COMPARA 8 E [1, 2, 3, 6, 4, 7] NÃO TROCA

[1, 2, 3, 6, 4, 7] = COMPARA 9 E [1, 2, 3, 4, 6, 7] TROCA 7

[1, 2, 3, 4, 6, 7]

**FINAL:** COMPARAÇÕES: 9 E TROCAS: 7.

# Aula 6- ALGORITMOS E COMPLEXIDADE

## TEMA 3 - ALGORITMOS DE ORDENAÇÃO AVANÇADOS

### Bubble Sort (Ordenação Bolha - CÓDIGO)

```
void bolha(int vet[ ], int tam)
{
    int j,i, aux;
    for (i=0; i<tam -1; i++) {

        for(j=tam-1; j>i; j--) {

            if( vet[j] < vet[j-1] )
            {
                aux=vet[j];
                vet[j]= vet[j-1];
                vet[j-1]=aux;
            } // Fecha o IF.

        } // Fecha o FOR INTERNO.
    } // Fecha o FOR EXTERNO.
} // Fecha o void bolha.
```

# Aula 6- ALGORITMOS E COMPLEXIDADE

## TEMA 3 - ALGORITMOS DE ORDENAÇÃO AVANÇADOS

### Insertion Sort (Ordenação por Inserção)

O princípio básico desse método é considerar o vetor como dois subvetores, um ordenado e o outro, desordenado, buscando posicionar o elemento que se encontra no subvetor desordenado, no vetor ordenado.

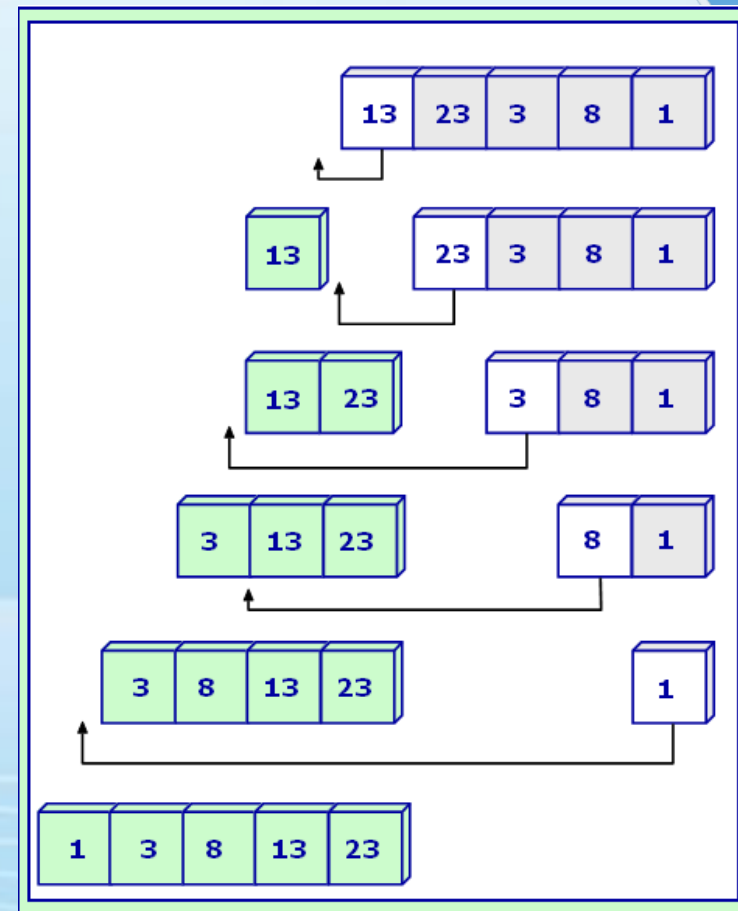


# Aula 6- ALGORITMOS E COMPLEXIDADE

## TEMA 3 - ALGORITMOS DE ORDENAÇÃO AVANÇADOS

### Insertion Sort (Ordenação por Inserção)

O princípio básico desse método é considerar o vetor como dois subvetores, um ordenado e o outro, desordenado, buscando posicionar o elemento que se encontra no subvetor desordenado, no vetor ordenado.

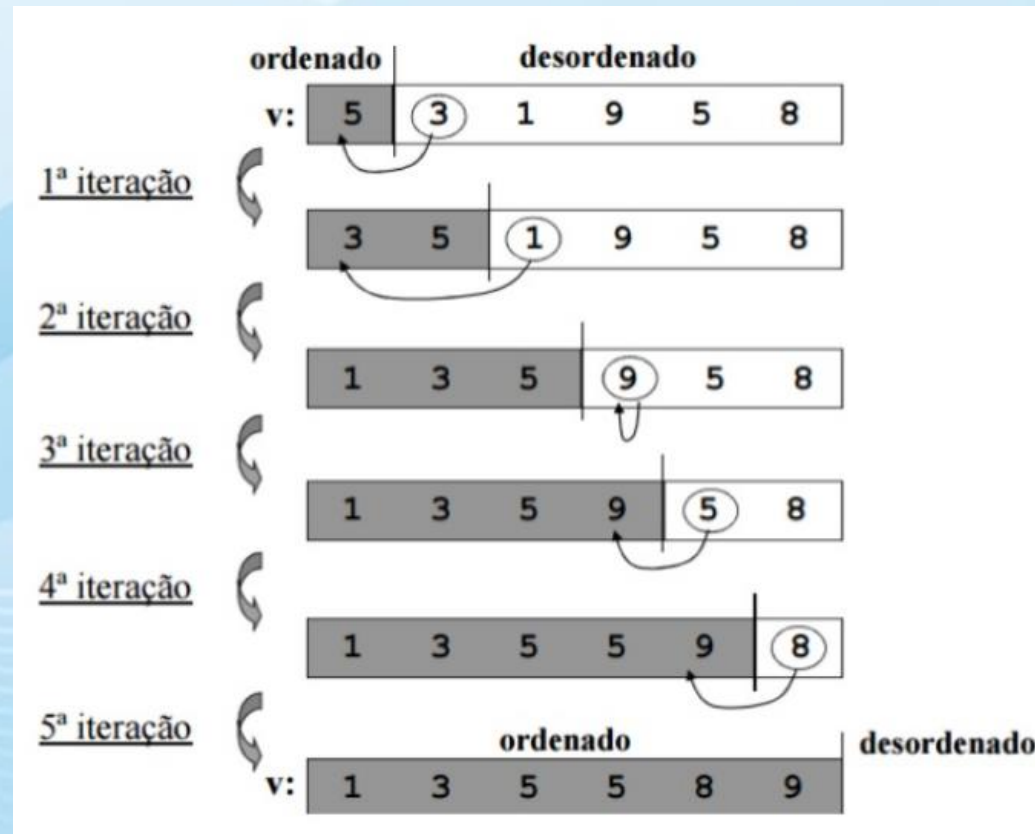


# Aula 6- ALGORITMOS E COMPLEXIDADE

## TEMA 3 - ALGORITMOS DE ORDENAÇÃO AVANÇADOS

### EXEMPLO DE COMO O PROCESSO É REALIZADO NA ORDENAÇÃO POR INSERÇÃO (PASSO A PASSO)

2. Neste exemplo o objetivo é colocar em *ordem crescente*, através do método de **ORDENAÇÃO POR INSERÇÃO**, o conjunto de valores [5, 3, 1, 9, 5, 8]". Acompanhe o **PASSO A PASSO**.



# Aula 6- ALGORITMOS E COMPLEXIDADE

## TEMA 3 - ALGORITMOS DE ORDENAÇÃO AVANÇADOS

### Insertion Sort (Ordenação por inserção - CÓDIGO)

```
void insercao(int vet[], int tam)
{
    int j, i, aux;

    for (i=1;i<tam;i++)
    {
        aux = vet[i];

        for(j=i; j>0 && aux <vet[j-1]; j--) {
            vet[j]=vet[j-1];
        }
        vet[j]=aux;
    }
}
```