

# Algoritmos de ordenação

Murilo Dantas

# Tópicos desta aula

1. Bubble Sort
2. Insertion Sort
3. Selection Sort
4. Merge Sort
5. Quick Sort

# Bubble Sort v. 0

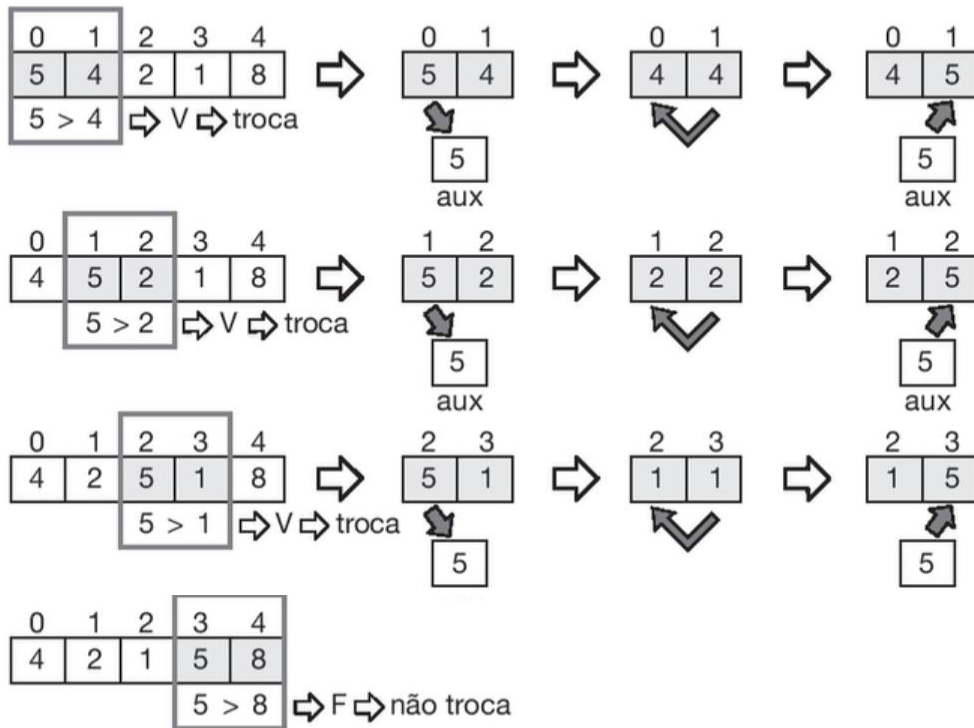
- Funcionamento
  - ▶ São efetuadas comparações em um vetor de tamanho  $n$ .
  - ▶ Cada elemento da posição  $i$  será comparado com o elemento da posição  $i+1$ .
    - ▶ Numa ordenação crescente, se o elemento  $i$  for maior que o  $i+1$ , troca-se os elementos.

# Bubble Sort v. 0

- Funcionamento (cont.)
  - ▶ Serão executados 2 laços:
    - ▶ Um laço com a quantidade de elementos.  
`for(j=1; j<=n; j++)`
    - ▶ E outro, dentro deste, que percorre da primeira à penúltima posição.  
`for(i=0; i<n-1; i++)`

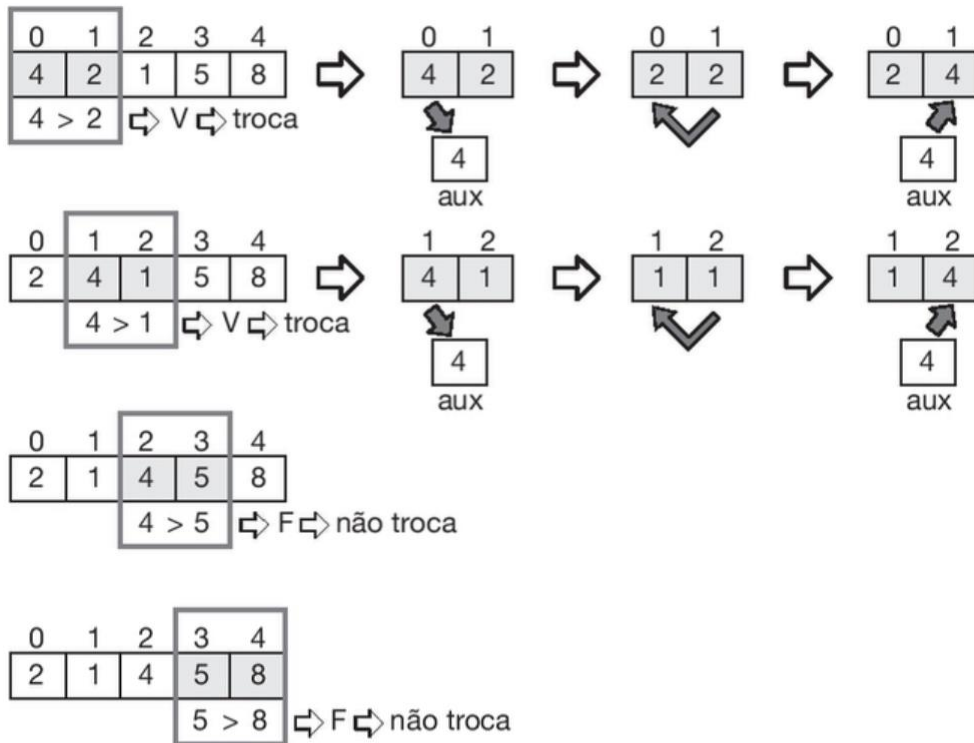
# Bubble Sort v. 0: exemplo

1ª execução do laço



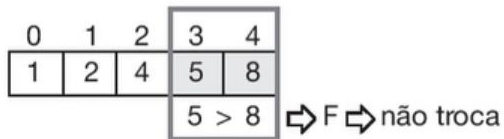
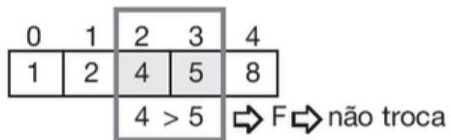
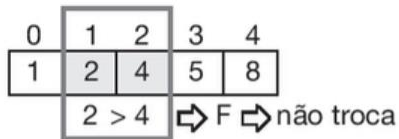
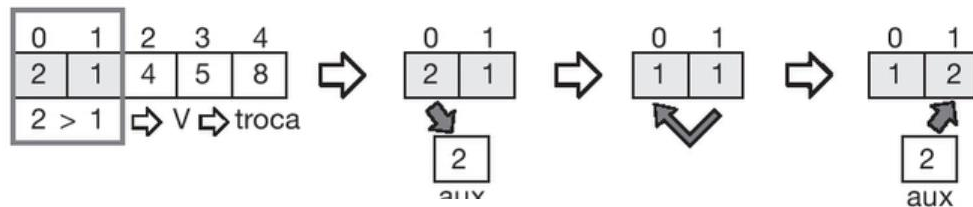
# Bubble Sort v. 0: exemplo

2ª execução do laço



# Bubble Sort v. 0: exemplo

3ª execução do laço



# Bubble Sort v. 0: exemplo

4ª execução do laço

|       |   |                 |   |   |
|-------|---|-----------------|---|---|
| 0     | 1 | 2               | 3 | 4 |
| 1     | 2 | 4               | 5 | 8 |
| 1 > 2 |   | ⇒ F ⇒ não troca |   |   |

|       |   |                 |   |   |
|-------|---|-----------------|---|---|
| 0     | 1 | 2               | 3 | 4 |
| 1     | 2 | 4               | 5 | 8 |
| 2 > 4 |   | ⇒ F ⇒ não troca |   |   |

|       |   |                 |   |   |
|-------|---|-----------------|---|---|
| 0     | 1 | 2               | 3 | 4 |
| 1     | 2 | 4               | 5 | 8 |
| 4 > 5 |   | ⇒ F ⇒ não troca |   |   |

|       |   |                 |   |   |
|-------|---|-----------------|---|---|
| 0     | 1 | 2               | 3 | 4 |
| 1     | 2 | 4               | 5 | 8 |
| 5 > 8 |   | ⇒ F ⇒ não troca |   |   |



# Bubble Sort v. 0: exemplo

## 5ª execução do laço

Apesar de o vetor já estar ordenado,  
mais uma execução do laço será realizada.

|       |   |                 |   |   |
|-------|---|-----------------|---|---|
| 0     | 1 | 2               | 3 | 4 |
| 1     | 2 | 4               | 5 | 8 |
| 1 > 2 |   | ⇒ F ⇒ não troca |   |   |

|       |   |                 |   |   |
|-------|---|-----------------|---|---|
| 0     | 1 | 2               | 3 | 4 |
| 1     | 2 | 4               | 5 | 8 |
| 2 > 4 |   | ⇒ F ⇒ não troca |   |   |

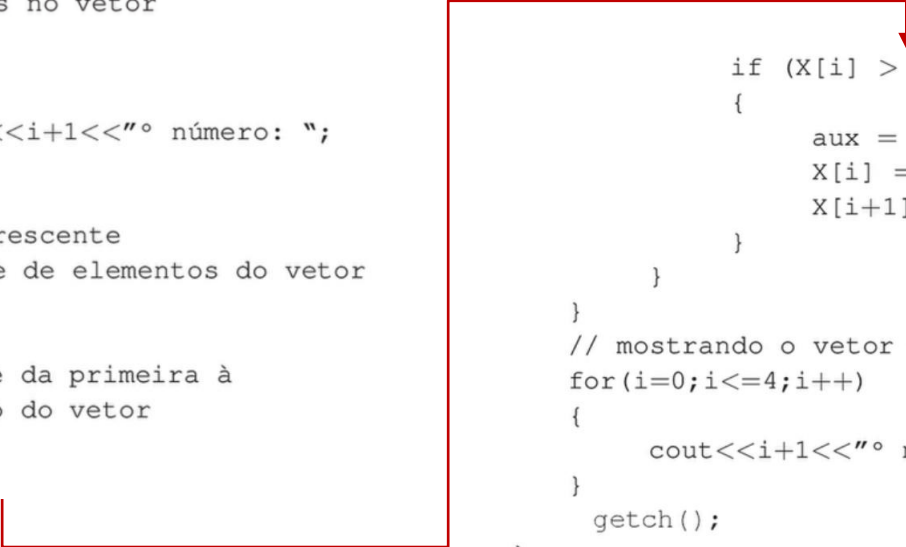
|       |   |                 |   |   |
|-------|---|-----------------|---|---|
| 0     | 1 | 2               | 3 | 4 |
| 1     | 2 | 4               | 5 | 8 |
| 4 > 5 |   | ⇒ F ⇒ não troca |   |   |

|       |   |                 |   |   |
|-------|---|-----------------|---|---|
| 0     | 1 | 2               | 3 | 4 |
| 1     | 2 | 4               | 5 | 8 |
| 5 > 8 |   | ⇒ F ⇒ não troca |   |   |

# Bubble Sort v. 0: código

```
#include <iostream.h>
#include <conio.h>
void main()
{
    int X[5], n, i, aux;
    clrscr();
    // carregando os números no vetor

    for(i=0;i<=4;i++)
    {
        cout<<"Digite o "<<i+1<<"º número: ";
        cin>>X[i];
    }
    // ordenando de forma crescente
    // laço com a quantidade de elementos do vetor
    for(n=1;n<=5;n++)
    {
        // laço que percorre da primeira à
        // penúltima posição do vetor
        for(i=0;i<=3;i++)
        {
            if (X[i] > X[i+1])
            {
                aux = X[i];
                X[i] = X[i+1];
                X[i+1] = aux;
            }
        }
        // mostrando o vetor ordenado
        for(i=0;i<=4;i++)
        {
            cout<<i+1<<"º número: "<<X[i]<<"\n";
        }
        getch();
    }
}
```



# Bubble Sort v. 1

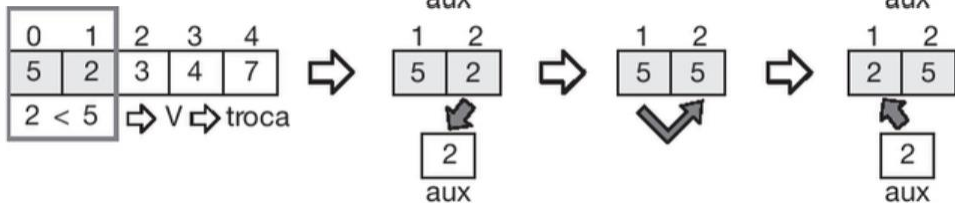
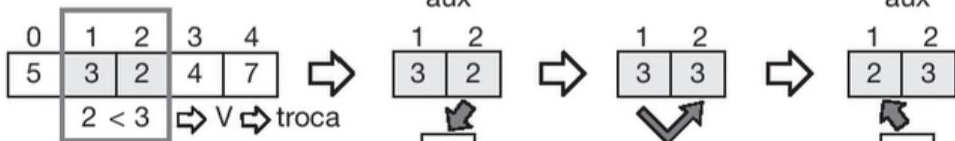
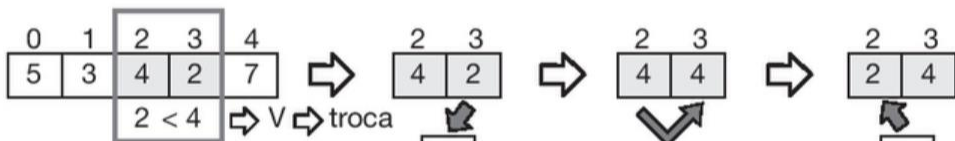
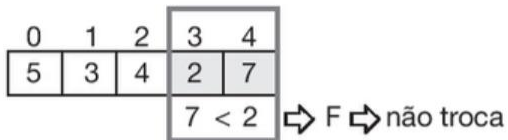
- Funcionamento
  - ▶ São efetuadas comparações em um vetor de tamanho  $n$ .
  - ▶ Cada elemento da posição  $i$  será comparado com o elemento da posição  $i-1$ .
    - ▶ Quando a ordenação procurada é encontrada, a troca será efetuada.

# Bubble Sort v. 1

- Funcionamento (cont.)
  - ▶ Serão executados 2 laços:
    - ▶ Um laço com a quantidade de elementos – 1.  
`for(j=1; j<n; j++)`
    - ▶ E outro, dentro deste, que percorre da última posição à posição  $j$ , fazendo com que as posições já ordenadas não sejam percorridas.  
`for(i=n-1; i>=j; i--)`

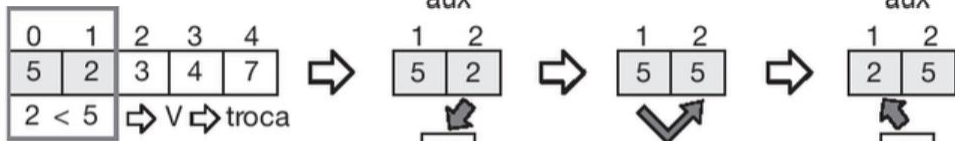
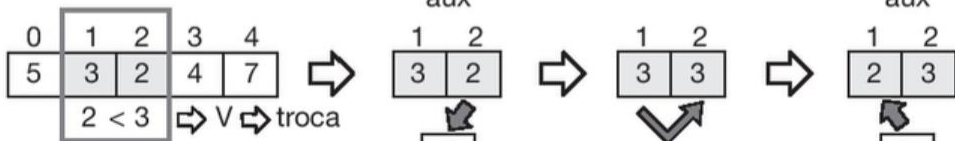
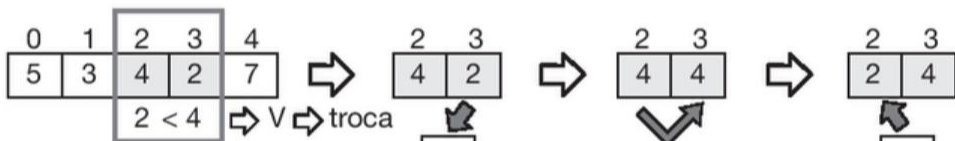
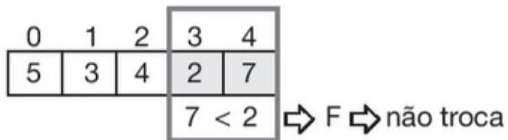
# Bubble Sort v. 1: exemplo

1ª execução do laço



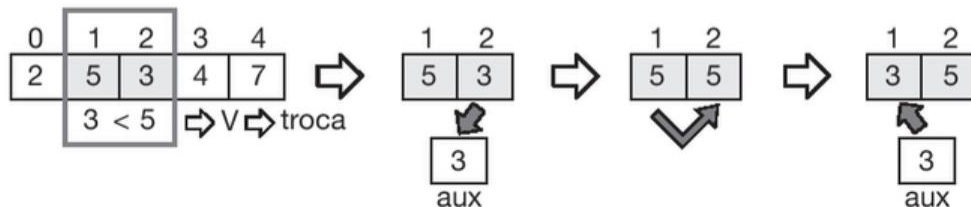
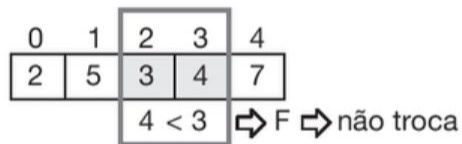
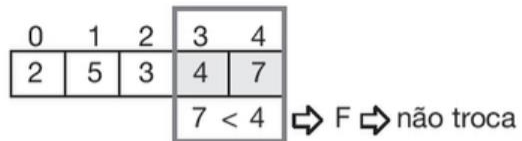
# Bubble Sort v. 1: exemplo

1ª execução do laço



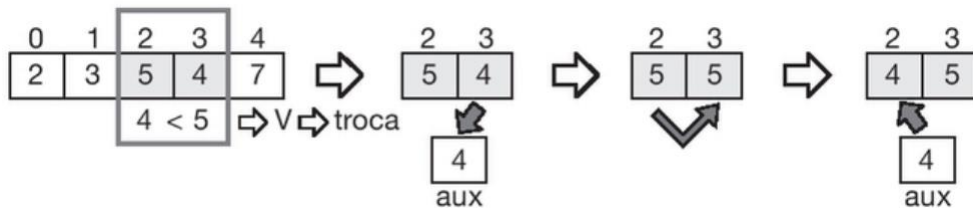
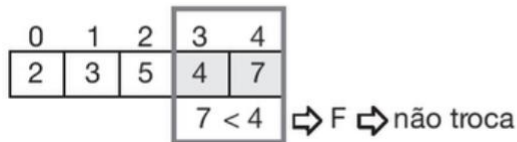
# Bubble Sort v. 1: exemplo

2ª execução do laço



# Bubble Sort v. 1: exemplo

## 3ª execução do laço



## 4ª execução do laço

Apesar de o vetor já estar ordenado, mais uma execução do laço será realizada.






# Bubble Sort v. 1: código

```
#include <iostream.h>
#include <conio.h>
void main()
{
    int X[5], j, i, aux;
    clrscr();
    // carregando os números no vetor
    for(i=0;i<=4;i++)
    {
        cout<<"Digite o "<<i+1<<"º número: ";
        cin>>X[i];
    }
    // ordenando de forma crescente
    // laço com a quantidade de elementos do vetor - 1
    for(j=1;j<=4;j++)
    {
        // laço que percorre da última posição à
        // posição j do vetor
        for(i=4;i>=j;i--)
        {
            if (X[i] < X[i-1])
            {
                aux = X[i];
                X[i] = X[i-1];
                X[i-1] = aux;
            }
        }
    }

    // mostrando o vetor ordenado
    for(i=0;i<=4;i++)
    {
        cout<<i+1<<"º número: "<<X[i]<<"\n";
    }
    getch();
}
```



# Bubble Sort v. 2

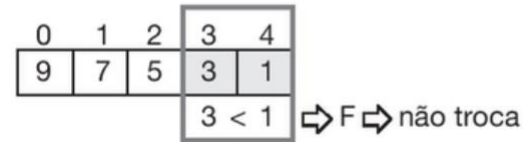
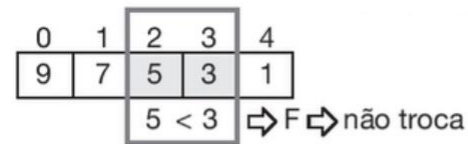
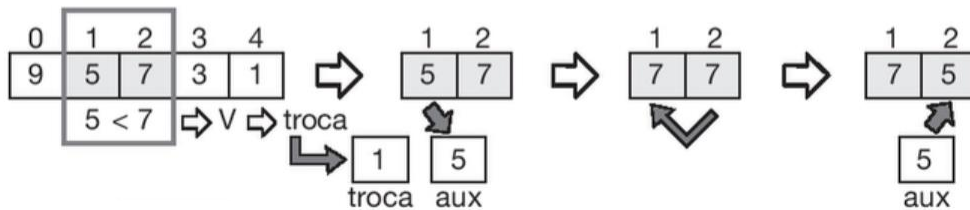
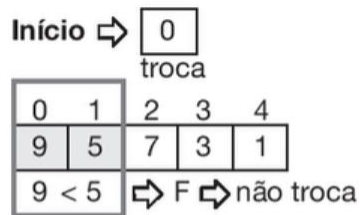
- Funcionamento
  - ▶ São efetuadas comparações em um vetor de tamanho  $n$ .
  - ▶ Cada elemento da posição  $i$  será comparado com o elemento da posição  $i+1$ .
    - ▶ Quando a ordenação procurada é encontrada, a troca será efetuada.

# Bubble Sort v. 2

- Funcionamento (cont.)
  - ▶ Serão executados 2 laços:
    - ▶ Um laço com a quantidade de elementos e enquanto houver trocas.  
`(j=1) e while(j <= n && troca == 1)`
    - ▶ E outro, dentro deste, que percorre da primeira à penúltima posição.  
`for(i=0; i<n-1; i++)`

# Bubble Sort v. 2: exemplo

1ª execução do laço  
 $n = 1$  e troca = 1



Fim =  $\Rightarrow$ 

|   |
|---|
| 1 |
|---|

 $\Rightarrow$  Deve continuar.  
troca

# Bubble Sort v. 2: exemplo

2ª execução do laço  
 $n = 2$  e troca = 1

Início  $\Rightarrow$ 

|   |
|---|
| 0 |
|---|

  
troca

|       |   |   |   |   |
|-------|---|---|---|---|
| 0     | 1 | 2                                       | 3 | 4 |
| 9     | 7 | 5                                       | 3 | 1 |
| 9 < 7 |   | $\Rightarrow$ F $\Rightarrow$ não troca |   |   |

|       |   |   |   |   |
|-------|---|---|---|---|
| 0     | 1 | 2                                       | 3 | 4 |
| 9     | 7 | 5                                       | 3 | 1 |
| 7 < 5 |   | $\Rightarrow$ F $\Rightarrow$ não troca |   |   |

|       |   |   |   |   |
|-------|---|---|---|---|
| 0     | 1 | 2                                       | 3 | 4 |
| 9     | 7 | 5                                       | 3 | 1 |
| 5 < 3 |   | $\Rightarrow$ F $\Rightarrow$ não troca |   |   |

|       |   |   |   |   |
|-------|---|---|---|---|
| 0     | 1 | 2                                       | 3 | 4 |
| 9     | 7 | 5                                       | 3 | 1 |
| 3 < 1 |   | $\Rightarrow$ F $\Rightarrow$ não troca |   |   |

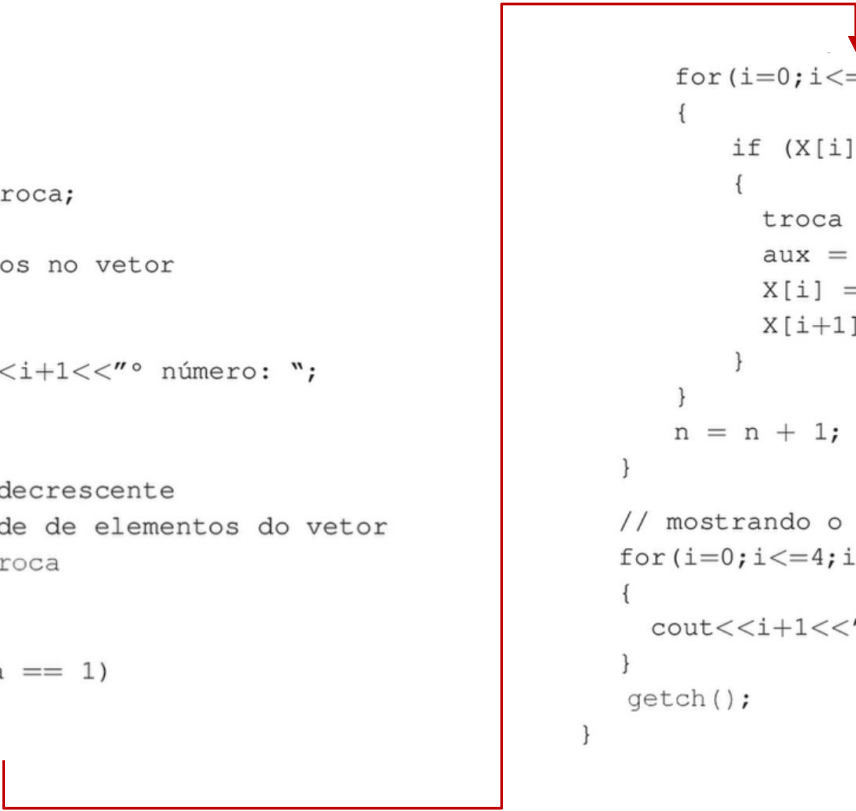
Fim = 

|   |
|---|
| 0 |
|---|

 $\Rightarrow$  Vetor ordenado, sai do laço.  
troca

# Bubble Sort v. 2: código

```
#include <iostream.h>
#include <conio.h>
void main()
{
    int X[5], n, i, aux, troca;
    clrscr();
    // carregando os números no vetor
    for(i=0;i<=4;i++)
    {
        cout<<"Digite o "<<i+1<<"º número: ";
        cin>>X[i];
    }
    // ordenando de forma decrescente
    // laço com a quantidade de elementos do vetor
    // e enquanto houver troca
    n = 1;
    troca = 1;
    while (n <= 5 && troca == 1)
    {
        troca = 0;
```



```
        for(i=0;i<=3;i++)
        {
            if (X[i] < X[i+1])
            {
                troca = 1;
                aux = X[i];
                X[i] = X[i+1];
                X[i+1] = aux;
            }
        }
        n = n + 1;
    }
    // mostrando o vetor ordenado
    for(i=0;i<=4;i++)
    {
        cout<<i+1<<"º número: "<<X[i]<<"\n";
    }
    getch();
}
```

# Insertion Sort

- Funcionamento

- ▶ As comparações iniciam a partir do segundo número do vetor de tamanho  $n$ .
- ▶ Números à esquerda do número escolhido, estão sempre ordenados.
- ▶ Um laço com as comparações será executado do segundo ao último elemento.

```
for(i=1; i<n; i++)
```

# Insertion Sort

- Funcionamento (cont.)
  - ▶ Condições para executar o laço
    - ▶ Enquanto existirem elementos à esquerda do número eleito para comparações.
    - ▶ A posição que atende a ordenação que se busca não for encontrada.

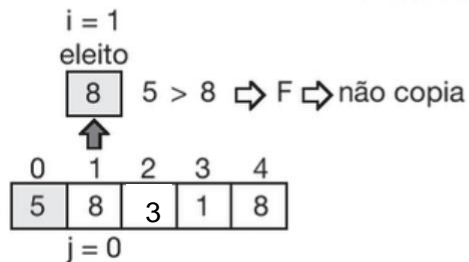


# Insertion Sort

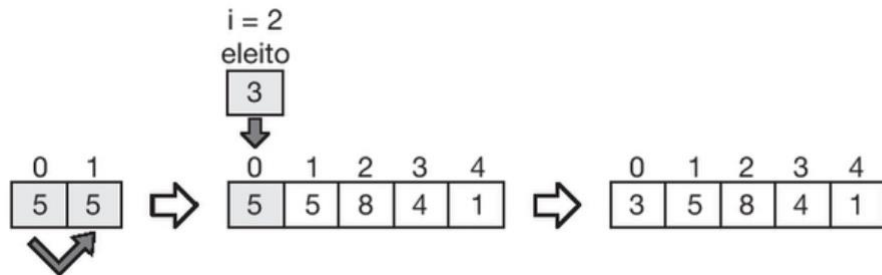
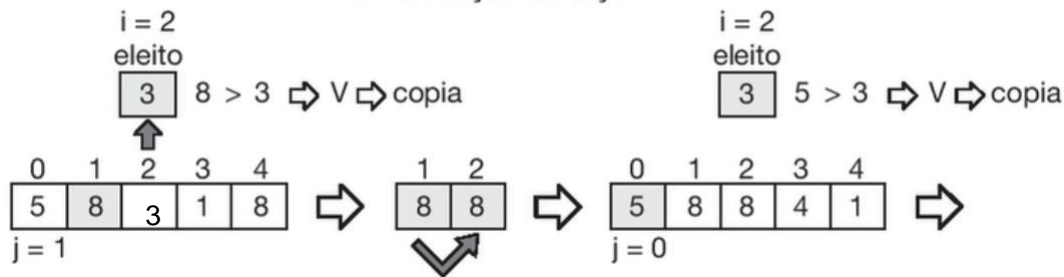
- Funcionamento (cont.)
  - ▶ O número eleito está na posição  $i$  e os números à esquerda estão na posição  $i-1$  a  $0$ .
  - ▶ O segundo laço será:  
 $j=i-1$  e `while(j >= 0 && elemento[j] > eleito)`

# Insertion Sort: exemplo

## 1ª execução do laço

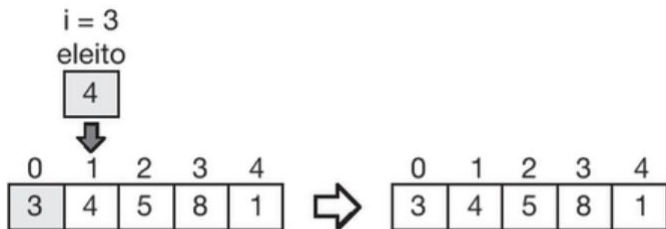
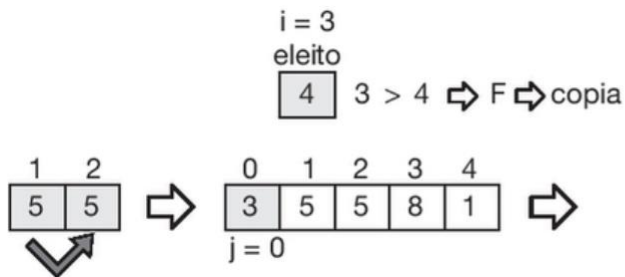
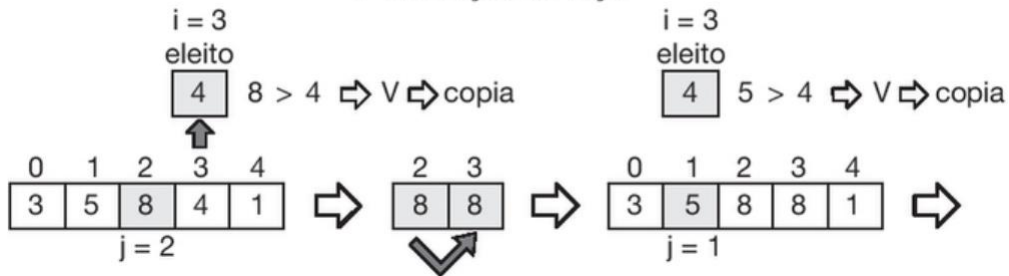


## 2ª execução do laço



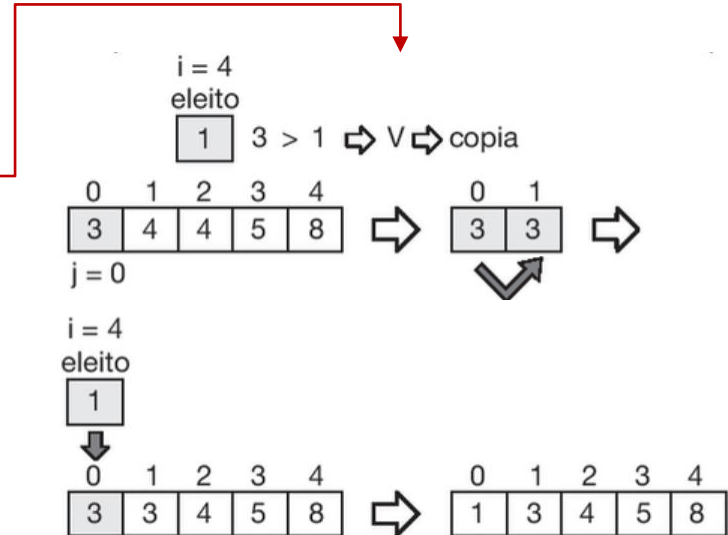
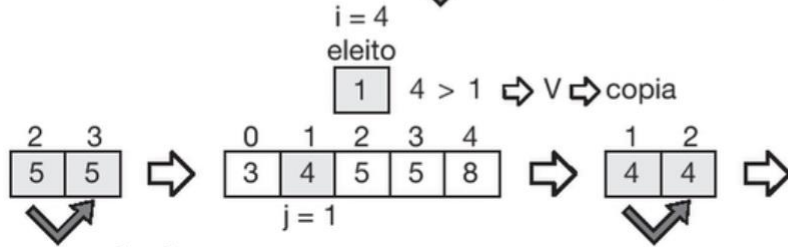
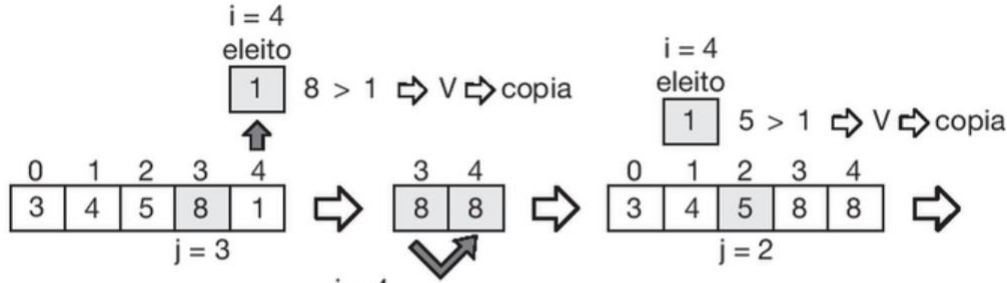
# Insertion Sort: exemplo

3ª execução do laço



# Insertion Sort: exemplo

4ª execução do laço



# Insertion Sort: código

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

void main()
{
    int X[5];
    int i, j, eleito;
    clrscr();
    // carregando os números no vetor
    for(i=0;i<=4;i++)
    {
        cout<<"Digite o "<<i+1<<"º número: ";
        cin>>X[i];
    }
    // ordenando de forma crescente
    // laço com a quantidade de elementos do vetor - 1
    for(i=1;i<=4;i++)
    {
        eleito = X[i];
        j = i - 1;
        // laço que percorre os elementos à
        // esquerda do número eleito
        // ou até encontrar a posição para
        // recolocação do número eleito
        // respeitando a ordenação procurada
        while (j >= 0 && X[j] > eleito)
        {
            X[j+1] = X[j];
            j = j - 1;
        }
        X[j+1] = eleito;
    }
    // mostrando o vetor ordenado
    for(i=0;i<=4;i++)
    {
        cout<<"\n"<<i+1<<"º número: "<<X[i];
    }
    getch();
}
```

# Selection Sort

- Funcionamento

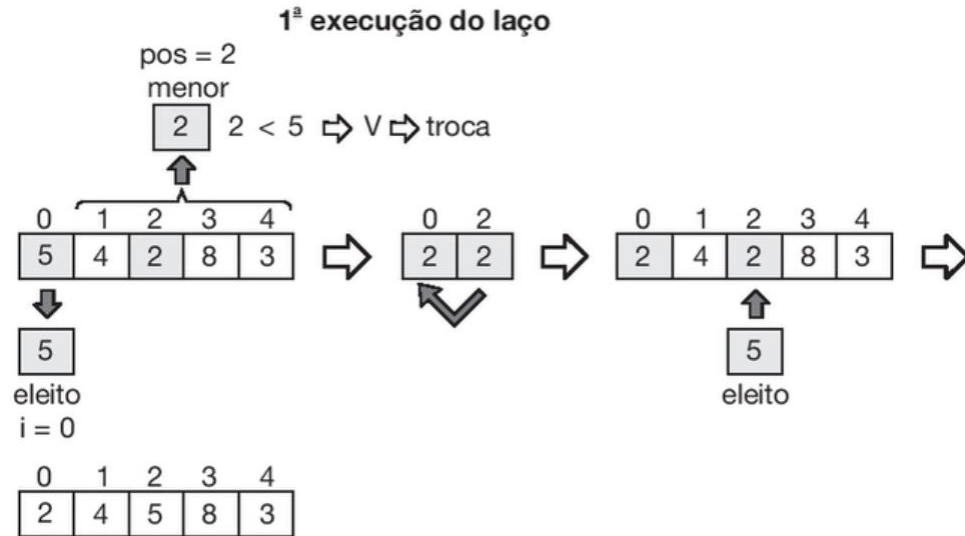
- ▶ Cada número do vetor, a partir do primeiro, será eleito e comparado com o menor ou maior, dentre os que estão à direita do eleito.
- ▶ Procura-se um número menor (crescente) ou maior (decrescente) que o eleito.

# Selection Sort

- Funcionamento (cont.)
  - ▶ Quando satisfaz as condições de ordenação, este trocará de posição com o eleito.
    - ▶ Todos à esquerda do eleito estarão ordenados.
  - ▶ Laços

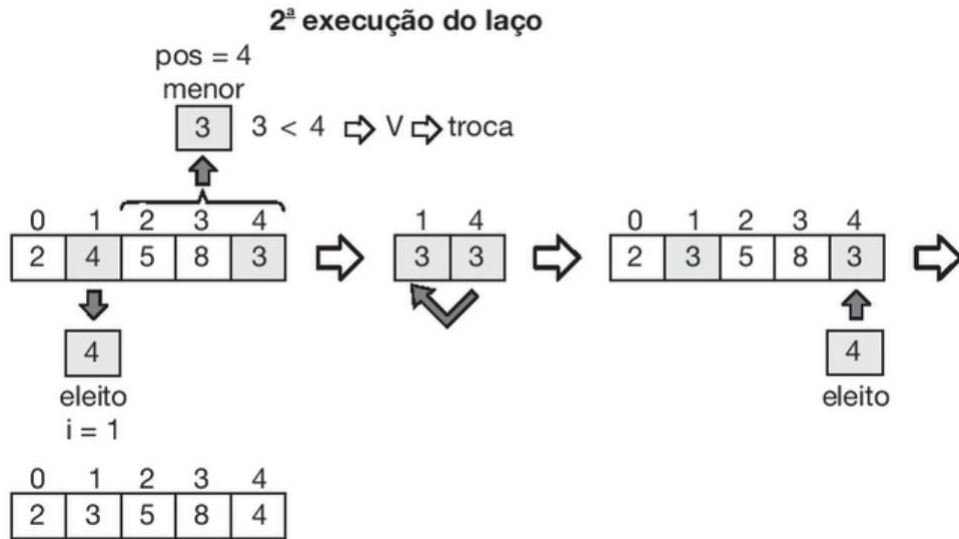
```
for(i=0; i<n-1; i++)  
  for(j=i+2; j<=n-1; j++)
```
  - ▶ O número eleito é  $i$  e à direita são  $i+1$  a  $n-1$ .

# Selection Sort: exemplo

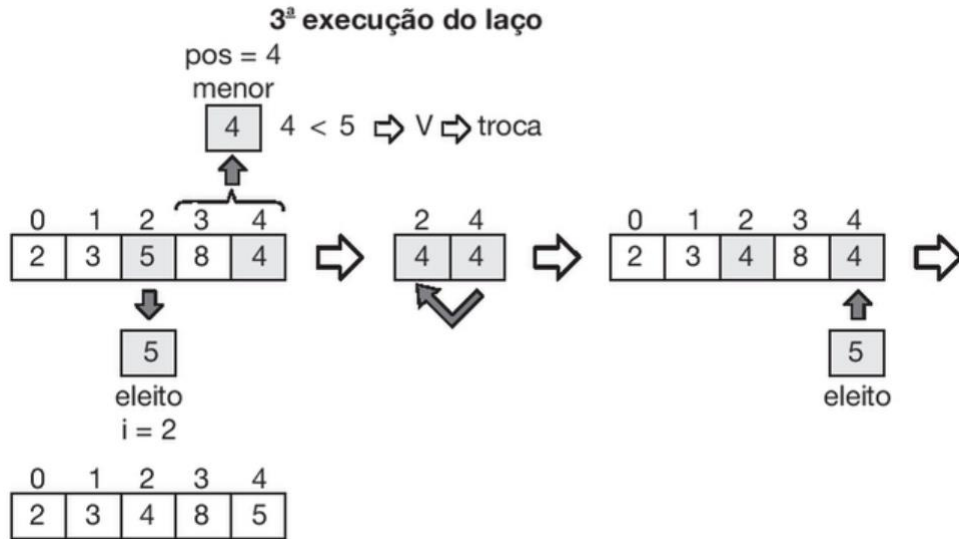




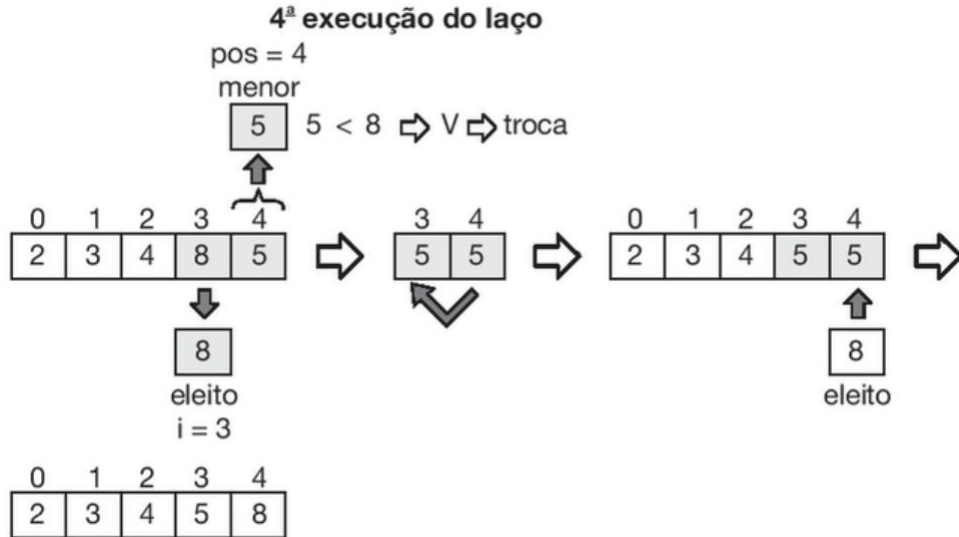
# Selection Sort: exemplo



# Selection Sort: exemplo



# Selection Sort: exemplo



# Selection Sort: código

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

void main ()
{
    int X[5];
    int i, j, eleito, menor, pos;
    clrscr();
    // carregando os números no vetor
    for(i=0;i<=4;i++)
    {
        cout<<"Digite o "<<i+1<<"º número: ";
        cin>>X[i];
    }
    // ordenando de forma crescente
    // laço que percorre da 1ª posição
    // à penúltima posição do vetor
    // elegendo um número para ser comparado
    for(i=0;i<=3;i++)
    {
        eleito = X[i];
        // encontrando o menor número à direita do eleito
        // com sua respectiva posição
```

```
        // posição do eleito = i
        // primeiro número à direita do
        // eleito na posição = i + 1
        menor = X[i+1];
        pos = i + 1;
        // laço que percorre os elementos
        // que estão à direita do
        // número eleito, retornando o
        // menor número à direita
        // e sua posição
        for (j=i+1;j<=4;j++)
        {
            if (X[j] < menor)
            {
                menor = X[j];
                pos = j;
            }
        }
        // troca do número eleito com o número da posição pos
        // o número da posição pos é o menor número à direita
        // do número eleito
```

# Selection Sort: código (cont.)

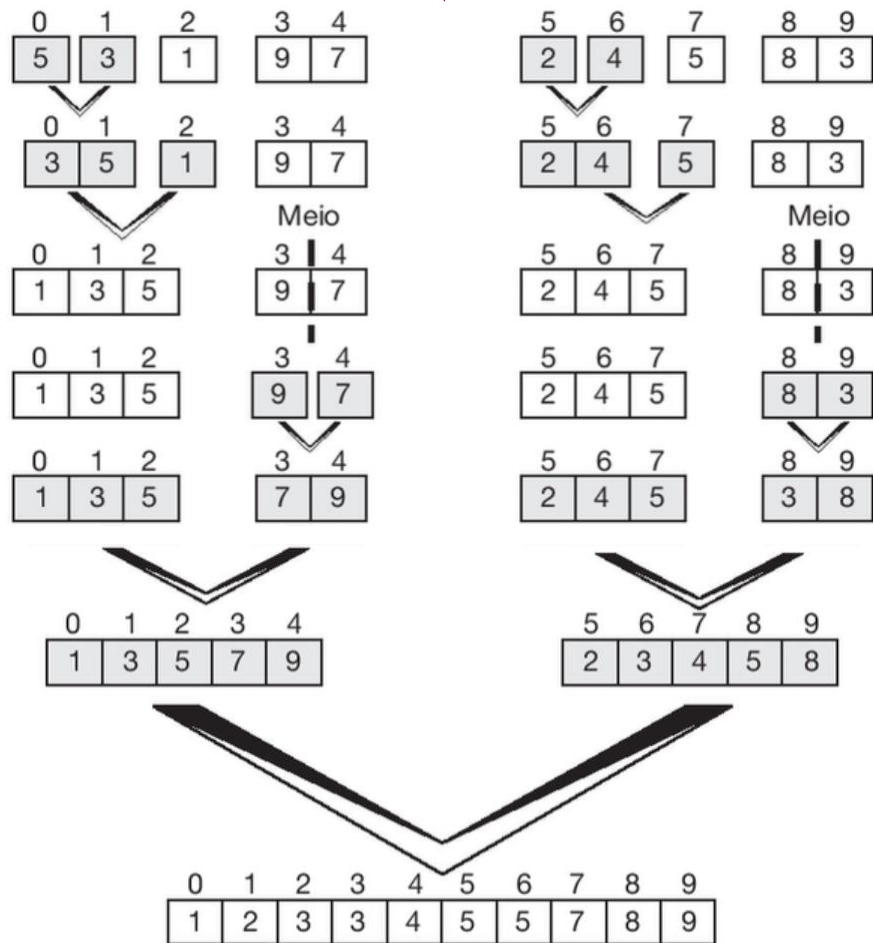
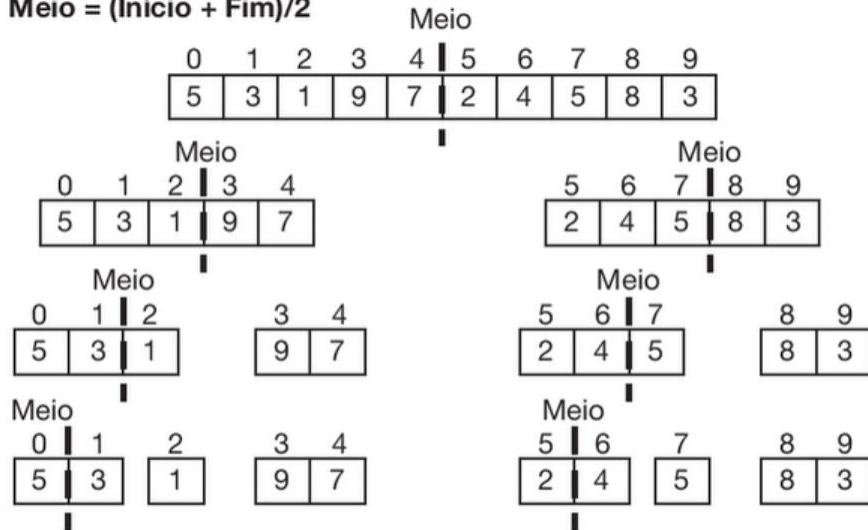
```
        if (menor < eleito)
        {
            X[i] = X[pos];
            X[pos] = eleito;
        }
    }
    // mostrando o vetor ordenado
    for(i=0; i<=4; i++)
    {
        cout<<"\n"<<i+1<<"º número: "<<X[i];
    }
    getch();
}
```

# Merge Sort

- Funcionamento
  - ▶ O vetor é dividido em vetores com a metade do tamanho original, de forma recursiva.
  - ▶ Isso ocorre até que o vetor fique com apenas um elemento e estes sejam ordenados e intercalados.

# Merge Sort: exemplo

**Meio = (Início + Fim)/2**



# Merge Sort: código

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

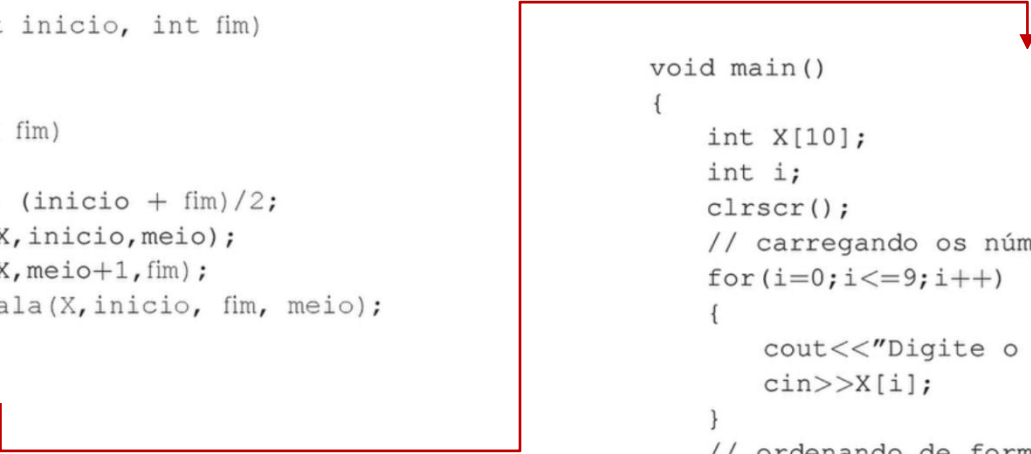
void intercala (int X[],int inicio, int fim, int meio)
{
    int poslivre, inicio_vetor1, inicio_vetor2, i;
    int aux[10];
    inicio_vetor1 = inicio;
    inicio_vetor2 = meio + 1;
    poslivre = inicio;
    while (inicio_vetor1 <= meio && inicio_vetor2 <= fim)
    {
        if (X[inicio_vetor1] <= X[inicio_vetor2])
        {
            aux[poslivre] = X[inicio_vetor1];
            inicio_vetor1 = inicio_vetor1 + 1;
        }
        else
        {
            aux[poslivre] = X[inicio_vetor2];
            inicio_vetor2 = inicio_vetor2 + 1;
        }
        poslivre = poslivre + 1;
    }
}
```

```
// se ainda existem números no primeiro vetor
// que não foram intercalados
for (i=inicio_vetor1;i<=meio;i++)
{
    aux[poslivre] = X[i];
    poslivre = poslivre + 1;
}
// se ainda existem números no segundo vetor
// que não foram intercalados
for (i=inicio_vetor2;i<=fim;i++)
{
    aux[poslivre] = X[i];
    poslivre = poslivre + 1;
}
// retorna os valores do vetor aux para o vetor X
for (i=inicio;i<=fim;i++)
{
    X[i] = aux[i];
}
}
```



# Merge Sort: código

```
void merge (int X[], int inicio, int fim)
{
    int meio;
    if (inicio < fim)
    {
        meio = (inicio + fim)/2;
        merge(X, inicio, meio);
        merge(X, meio+1, fim);
        intercala(X, inicio, fim, meio);
    }
}
```



```
void main()
{
    int X[10];
    int i;
    clrscr();
    // carregando os números no vetor
    for(i=0; i<=9; i++)
    {
        cout<<"Digite o "<<i+1<<"º número: ";
        cin>>X[i];
    }
    // ordenando de forma crescente
    merge(X, 0, 9);
    // mostrando o vetor ordenado
    for(i=0; i<=9; i++)
    {
        cout<<"\n"<<i+1<<"º número: "<<X[i];
    }
    getch();
}
```

# Quick Sort

- Funcionamento
  - ▶ O vetor é particionado em 2 de modo recursivo até que o vetor fique com apenas 1 elemento.
  - ▶ É semelhante ao Merge Sort.

# Quick Sort: exemplo

1ª execução do laço

Vetor de 0 a 9

Posição do pivô = parte inteira  $[(0+9)/2] = 4$

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 5 | 8 | 3 | 1 | 6 | 2 | 4 | 9 | 7 | 5 |

↑  
pivô

5 <= 6 ⇒ V ⇒ para  
↓ j = 9

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 5 | 8 | 3 | 1 | 6 | 2 | 4 | 9 | 7 | 5 |

↑  
pivô  
pivô  
↓

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 5 | 8 | 3 | 1 | 6 | 2 | 4 | 9 | 7 | 5 |

↑ i = 0  
5 >= 6 ⇒ F ⇒ continua

pivô  
↓

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 5 | 8 | 3 | 1 | 6 | 2 | 4 | 9 | 7 | 5 |

↑ i = 1  
8 >= 6 ⇒ V ⇒ para  
pivô  
↓

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 5 | 8 | 3 | 1 | 6 | 2 | 4 | 9 | 7 | 5 |

i < j ⇒ 1 < 9 ⇒ V ⇒ troca

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 5 | 5 | 3 | 1 | 6 | 2 | 4 | 9 | 7 | 8 |

7 <= 6 ⇒ F ⇒ continua  
↓ j = 8

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 5 | 5 | 3 | 1 | 6 | 2 | 4 | 9 | 7 | 8 |

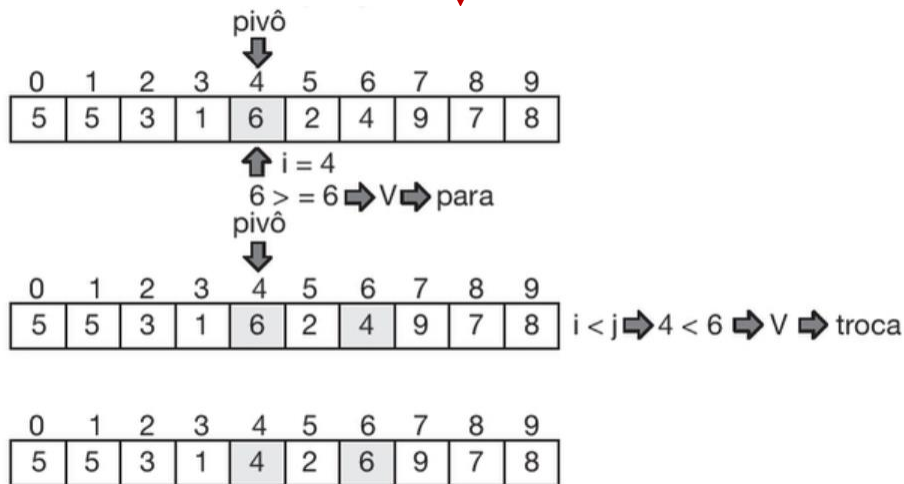
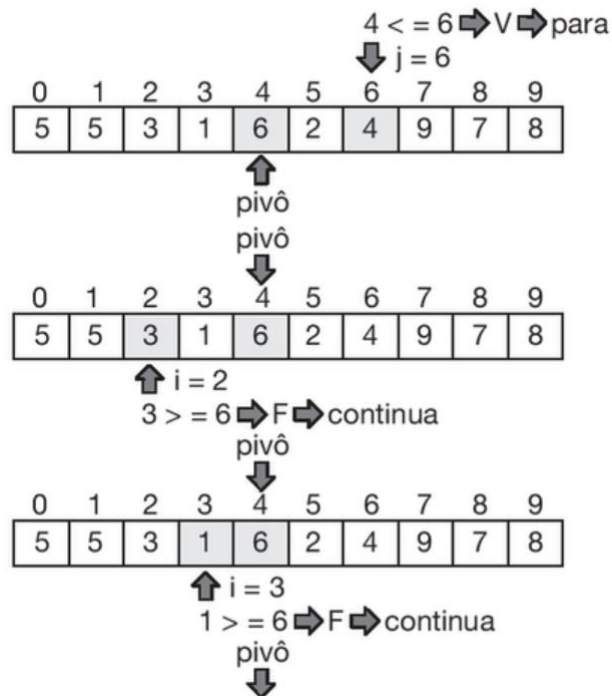
↑  
pivô

9 <= 6 ⇒ F ⇒ continua  
↓ j = 7

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 5 | 5 | 3 | 1 | 6 | 2 | 4 | 9 | 7 | 8 |

↑  
pivô

# Quick Sort: exemplo



# Quick Sort: exemplo

2ª execução do laço

Vetor de 0 a 5

Posição do pivô = parte inteira  $[(0+5)/2] = 2$

| 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 5 | 5 | 3 | 1 | 4 | 2 |

↑  
pivô

$2 \leq 3 \Rightarrow V \Rightarrow \text{para}$

↓  
 $j = 5$

| 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 5 | 5 | 3 | 1 | 4 | 2 |

↑  
pivô

| 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 5 | 5 | 3 | 1 | 4 | 2 |

↑  
 $i = 0$

$5 \geq 3 \Rightarrow V \Rightarrow \text{para}$

| 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 5 | 5 | 3 | 1 | 4 | 2 |

↑  
pivô

$i < j \Rightarrow 0 < 5 \Rightarrow V \Rightarrow \text{troca}$

| 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 2 | 5 | 3 | 1 | 4 | 5 |

↑  
pivô

$4 \leq 3 \Rightarrow F \Rightarrow \text{continua}$

↓  
 $j = 4$

| 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 2 | 5 | 3 | 1 | 4 | 5 |

↑  
pivô

$1 \leq 3 \Rightarrow V \Rightarrow \text{para}$

↓  
 $j = 3$

| 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 2 | 5 | 3 | 1 | 4 | 5 |

↑  
pivô

| 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 2 | 5 | 3 | 1 | 4 | 5 |

↑  
 $i = 1$

$5 \geq 3 \Rightarrow V \Rightarrow \text{para}$

| 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 2 | 5 | 3 | 1 | 4 | 2 |

$i < j \Rightarrow 1 < 3 \Rightarrow V \Rightarrow \text{troca}$

↑  
pivô

| 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 2 | 1 | 3 | 5 | 4 | 5 |

↑  
pivô

# Quick Sort: exemplo

## 3ª execução do laço

Vetor de 0 a 2

Posição do pivô = parte inteira  $[(0+2)/2] = 1$

|   |   |   |
|---|---|---|
| 0 | 1 | 2 |
| 2 | 1 | 3 |

↑  
pivô

$3 \leq 1 \Rightarrow F \Rightarrow \text{continua}$

↓  $j = 2$

|   |   |   |
|---|---|---|
| 0 | 1 | 2 |
| 2 | 1 | 3 |

↑  
pivô

$1 \leq 1 \Rightarrow V \Rightarrow \text{para}$

↓  $j = 1$

|   |   |   |
|---|---|---|
| 0 | 1 | 2 |
| 2 | 1 | 3 |

↑  
pivô

|   |   |   |
|---|---|---|
| 0 | 1 | 2 |
| 2 | 1 | 3 |

↑  
pivô

$i = 0$

$2 \geq 1 \Rightarrow V \Rightarrow \text{para}$

|   |   |   |
|---|---|---|
| 0 | 1 | 2 |
| 2 | 1 | 3 |

↑  
pivô

$i < j \Rightarrow 0 < 1 \Rightarrow V \Rightarrow \text{troca}$

|   |   |   |
|---|---|---|
| 0 | 1 | 2 |
| 1 | 2 | 3 |

↑  
pivô

# Quick Sort: exemplo

4ª execução do laço

Vetor de 1 a 2

Posição do pivô = parte inteira  $[(1+2)/2] = 1$

|   |   |
|---|---|
| 1 | 2 |
| 2 | 3 |

↑  
pivô

$3 < 2 \Rightarrow F \Rightarrow \text{continua}$

↓  $j = 2$

|   |   |
|---|---|
| 1 | 2 |
| 2 | 3 |

↑  
pivô

$2 < 2 \Rightarrow V \Rightarrow \text{para}$

↓  $j = 1$

|   |   |
|---|---|
| 1 | 2 |
| 2 | 3 |

↑  
pivô

|   |   |
|---|---|
| 1 | 2 |
| 2 | 3 |

↗  
↑  
pivô

$i = 1$

$2 \geq 2 \Rightarrow V \Rightarrow \text{para}$

|   |   |
|---|---|
| 1 | 2 |
| 2 | 3 |

↑  
pivô

$i < j \Rightarrow 1 < 1 \Rightarrow F \Rightarrow \text{não troca}$

# Quick Sort: exemplo

5ª execução do laço

Vetor de 3 a 5

Posição do pivô = parte inteira  $[(3+5)/2] = 4$

|   |   |   |
|---|---|---|
| 3 | 4 | 5 |
| 5 | 4 | 5 |

↑  
pivô

$5 \leq 4 \Rightarrow F \Rightarrow \text{continua}$

↓  $j = 5$

|   |   |   |
|---|---|---|
| 3 | 4 | 5 |
| 5 | 4 | 5 |

↑  
pivô

$4 \leq 4 \Rightarrow V \Rightarrow \text{para}$

↓  $j = 4$

|   |   |   |
|---|---|---|
| 3 | 4 | 5 |
| 5 | 4 | 5 |

↑  
pivô

|   |   |   |
|---|---|---|
| 3 | 4 | 5 |
| 5 | 4 | 5 |

↑  
pivô

$i = 3$

$5 \geq 4 \Rightarrow V \Rightarrow \text{para}$

|   |   |   |
|---|---|---|
| 3 | 4 | 5 |
| 5 | 4 | 5 |

↑  
pivô

$i < j \Rightarrow 3 < 4 \Rightarrow V \Rightarrow \text{troca}$

|   |   |   |
|---|---|---|
| 3 | 4 | 5 |
| 4 | 5 | 5 |

↑  
pivô

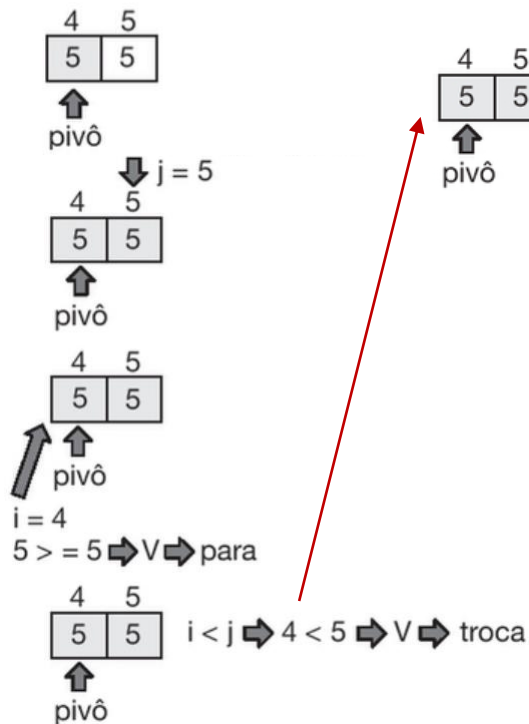


# Quick Sort: exemplo

## 6ª execução do laço

Vetor de 4 a 5

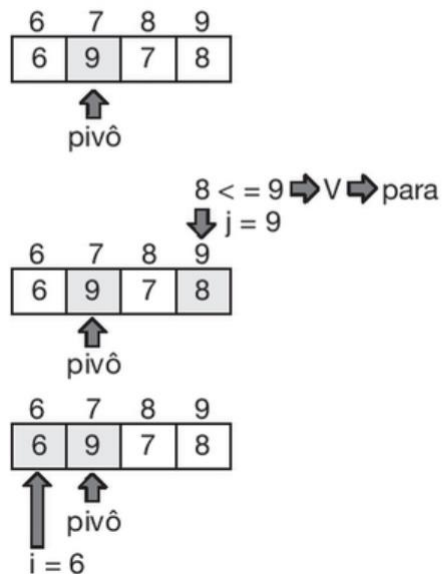
Posição do pivô = parte inteira  $[(4+5)/2] = 4$



## 7ª execução do laço

Vetor de 6 a 9

Posição do pivô = parte inteira  $[(6+9)/2] = 7$



# Quick Sort: exemplo

$6 \geq 9 \Rightarrow F \Rightarrow$  continua

|   |   |   |   |
|---|---|---|---|
| 6 | 7 | 8 | 9 |
| 6 | 9 | 7 | 8 |



$i = 7$

$9 \geq 9 \Rightarrow V \Rightarrow$  para

|   |   |   |   |
|---|---|---|---|
| 6 | 7 | 8 | 9 |
| 6 | 9 | 7 | 8 |

$i < j \Rightarrow 7 < 9 \Rightarrow V \Rightarrow$  troca

|   |   |   |   |
|---|---|---|---|
| 6 | 7 | 8 | 9 |
| 6 | 9 | 7 | 8 |

pivô

|   |   |   |   |
|---|---|---|---|
| 6 | 7 | 8 | 9 |
| 6 | 8 | 7 | 9 |

pivô

8ª execução do laço

Vetor de 6 a 8

Posição do pivô = parte inteira  $[(6+8)/2] = 7$

|   |   |   |
|---|---|---|
| 6 | 7 | 8 |
| 6 | 8 | 7 |

pivô

$7 \leq 8 \Rightarrow V \Rightarrow$  para

$j = 8$

|   |   |   |
|---|---|---|
| 6 | 7 | 8 |
| 6 | 8 | 7 |

pivô

|   |   |   |
|---|---|---|
| 6 | 7 | 8 |
| 6 | 8 | 7 |

pivô  
 $i = 6$

# Quick Sort: exemplo

$6 \geq 8 \Rightarrow F \Rightarrow$  continua

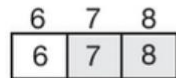


$i = 7$

$8 \geq 8 \Rightarrow V \Rightarrow$  para

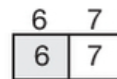


$i < j \Rightarrow 7 < 8 \Rightarrow V \Rightarrow$  troca



pivô

9ª execução do laço  
Vetor de 6 a 7  
Posição do pivô =  $\text{parte inteira}[(6+7)/2] = 7$



pivô

$7 \leq 6 \Rightarrow F \Rightarrow$  continua

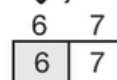
$j = 7$



pivô

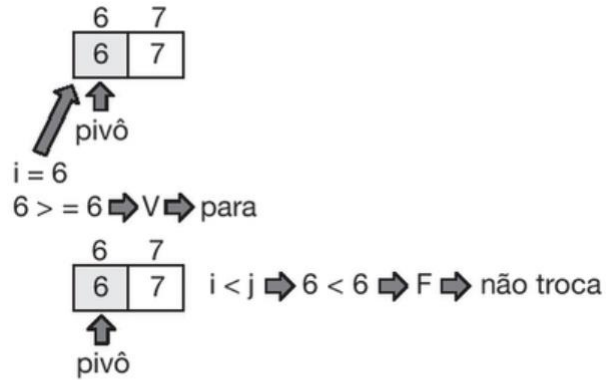
$6 \leq 6 \Rightarrow V \Rightarrow$  para

$j = 6$



pivô

# Quick Sort: exemplo



**Vetor ordenado**

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 5 | 6 | 7 | 8 | 9 |

# Quick Sort: código

```
#include <iostream.h>
```


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

```
void troca(int X[], int i, int j)
```

```
{
    int aux;
    aux = X[i];
    X[i] = X[j];
    X[j] = aux;
}
```

```
int particao(int X[],int p,int r)
```

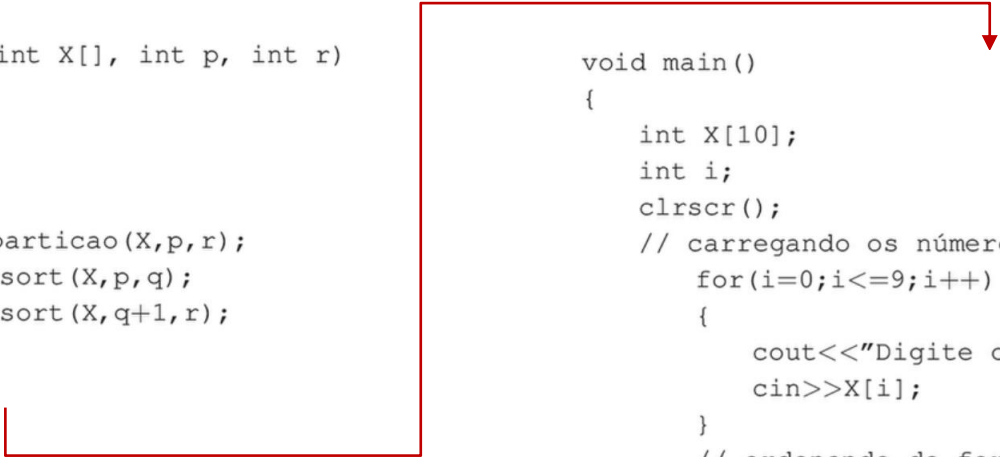
```
{
    int pivo, i, j;
    pivo = X[(p+r)/2];
    i = p-1;
    j = r+1;
    while (i < j)
    {
        do
        {
            j = j - 1;
        }
        while (X[j] > pivo);
    }
}
```



```
do
{
    i = i + 1;
}
while (X[i] < pivo);
if (i < j) troca(X,i,j);
}
return j;
}
```

# Quick Sort: código

```
void quicksort(int X[], int p, int r)
{
    int q;
    if (p < r)
    {
        q = particao(X,p,r);
        quicksort(X,p,q);
        quicksort(X,q+1,r);
    }
}
```



```
void main()
{
    int X[10];
    int i;
    clrscr();
    // carregando os números no vetor
    for(i=0;i<=9;i++)
    {
        cout<<"Digite o "<<i+1<<"º número: ";
        cin>>X[i];
    }
    // ordenando de forma crescente
    quicksort(X,0,9);
    // mostrando o vetor ordenado
    cout<<"Vetor Ordenado";
    for (i=0;i<=9;i++)
    {
        cout<<" "<<X[i];
    }
    getch();
}
```

**Perguntas?**

# Bibliografia da aula

- ASCENCIO, A. F. G.; ARAÚJO, G. S. Estrutura de dados. Algoritmos, análise da complexidade e implementação em Java e C/C++. Pearson. 2010.