

# Bubble Sort v.1

Murilo Dantas

# Bubble Sort v. 1

- Funcionamento

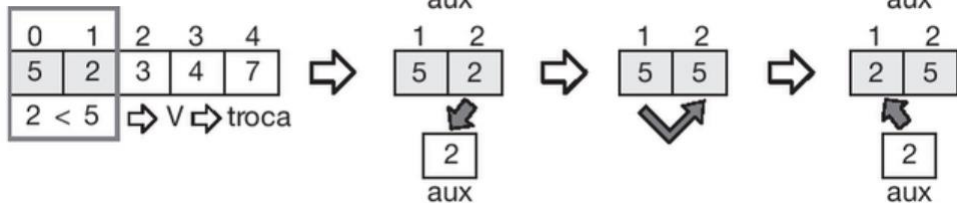
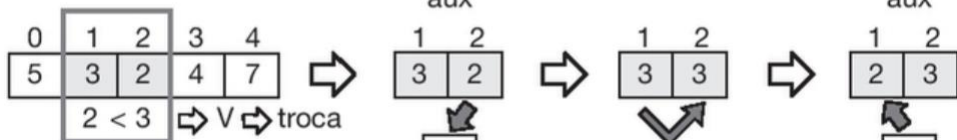
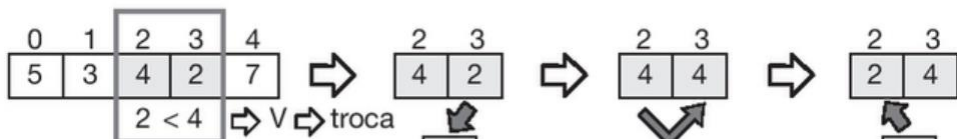
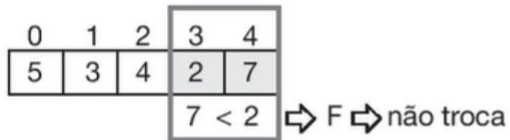
- ▶ São efetuadas comparações em um vetor de tamanho  $n$ .
- ▶ Cada elemento da posição  $i$  será comparado com o elemento da posição  $i-1$ .
  - ▶ Quando a ordenação procurada é encontrada, a troca será efetuada.

# 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--)

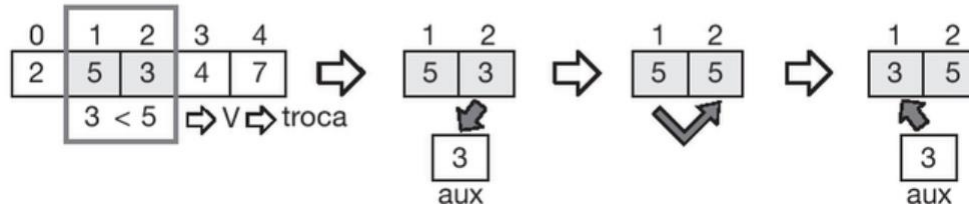
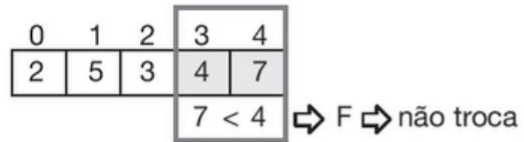
# Bubble Sort v. 1: exemplo

1ª execução do laço



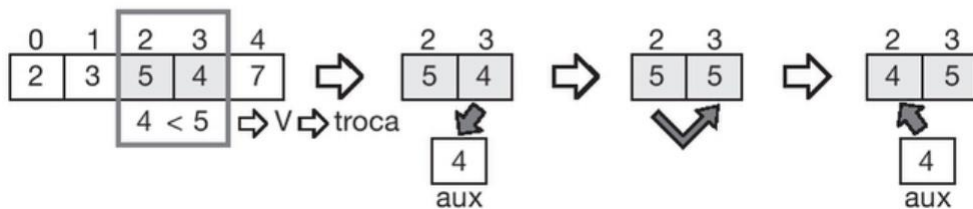
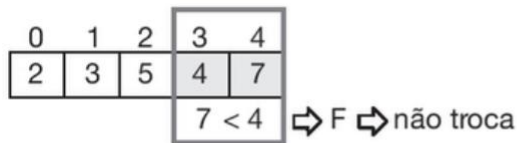
# Bubble Sort v. 1: exemplo

2ª execução do laço



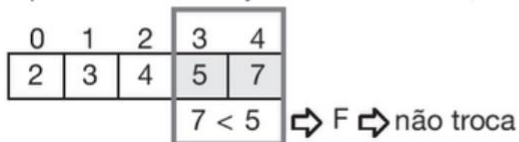
# Bubble Sort v. 1: exemplo

## 3ª execução do laço



## 4ª execução do laço

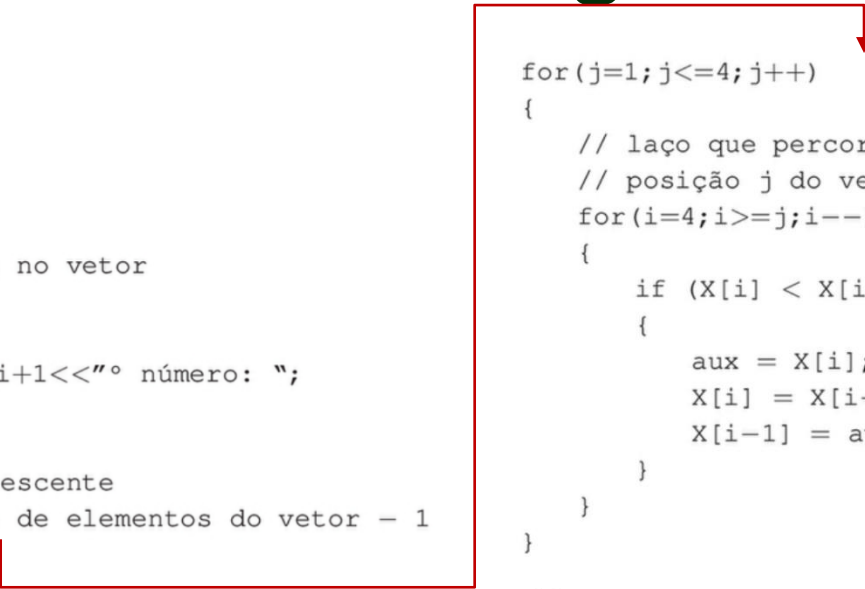
Apesar de o vetor já estar ordenado, mais uma execução do laço será realizada.



# 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();
}
```



**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.