

# *TECNOLOGIA EM INFORMÁTICA*

## ***Estrutura de Dados***

*Prof. Glauco da Silva*  
*contato: [glaucoslv@gmail.com](mailto:glaucoslv@gmail.com)*

# ***Roteiro***

- *Objetivos*
- *Métodos de ordenação elementares*
  - *Inserção direta*
  - *Seleção direta*
  - *Bubble sort*
- *Exercícios*

# ***Roteiro***

- ***Objetivos***
- *Métodos de ordenação elementares*
  - *Inserção direta*
  - *Seleção direta*
  - *Bubble sort*
- *Exercícios*

# ***Objetivos***

- *Apresentar os métodos de ordenação elementares, mostrando sua forma de trabalho e seu funcionamento para ordenação de valores*

# ***Roteiro***

- *Objetivos*
- ***Métodos de ordenação elementares***
  - ***Inserção direta***
  - *Seleção direta*
  - *Bubble sort*
- *Exercícios*

# *Inserção Direta*

- *Ordenar um arquivo pela inserção de registros no lugar correto*

25	57	48	37	12	92	86	33
25	57	48	37	12	92	86	33
25	48	57	37	12	92	86	33
25	37	48	57	12	92	86	33
12	25	37	48	57	92	86	33
12	25	37	48	57	92	86	33
12	25	37	48	57	86	92	33
12	25	33	37	48	57	86	92

# *Inserção Direta*

- *Algoritmo Básico*

*Repita  $i$ , de 1 até tamanho do arquivo*

*$y$  = elemento da vez*

*Localiza a posição da inserção entre  $i-1$  e 0*

*Desloca elementos 1 posição à direita*

*Insera  $y$*

# *Inserção Direta*

- *Possível implementação (Linguagem C)*

```
void insertsort (int x[], int tam)
{
    int i, j, z;
    for (i=1; i<tam; i++)
        {
            z=x[i];
            for (j=i-1; (j>=0) && (z<x[j]); j--)
                x[j+1]=x[j];
            x[j+1]=z;
        }
}
```

# ***Roteiro***

- *Objetivos*
- ***Métodos de ordenação elementares***
  - *Inserção direta*
  - ***Seleção direta***
  - *Bubble sort*
- *Exercícios*

# Seleção Direta

- *Selecionar o menor elemento, e colocá-lo no final dos elementos ordenados*

25	57	48	37	12	92	86	33
<b>12</b>	57	48	37	25	92	86	33
<b>12</b>	<b>25</b>	48	37	57	92	86	33
<b>12</b>	<b>25</b>	<b>33</b>	37	57	92	86	48
<b>12</b>	<b>25</b>	<b>33</b>	<b>37</b>	57	92	86	48
<b>12</b>	<b>25</b>	<b>33</b>	<b>37</b>	<b>48</b>	92	86	57
<b>12</b>	<b>25</b>	<b>33</b>	<b>37</b>	<b>48</b>	<b>57</b>	86	92
<b>12</b>	<b>25</b>	<b>33</b>	<b>37</b>	<b>48</b>	<b>57</b>	<b>86</b>	92
<b>12</b>	<b>25</b>	<b>33</b>	<b>37</b>	<b>48</b>	<b>57</b>	<b>86</b>	<b>92</b>

# *Seleção Direta*

- *Algoritmo Básico*

*Repita  $i$ , de 0 até tamanho do arquivo - 1*

*$x$  = menor elemento entre  $i$  e o final do arquivo*

*$px$  = posição do elemento  $x$*

*trocar elemento da posição  $i$  com elemento da posição  $px$*

# *Seleção Direta*

- *Possível implementação (Linguagem C)*

```
void selectsort (int x[], int tam){
    int i, j, z, p;
    tam--;
    for (i=0; i<tam; i++) {
        p=i;
        for (j=i+1; j<=tam; j++)
            if (x[j]<x[p])
                p=j;
        if (p!=i) {
            z=x[p];
            x[p]=x[i];
            x[i]=z;
        }
    }
}
```

# *Variações*

- *Realizando algumas modificações nos algoritmos de inserção e seleção diretas, obtemos algoritmos mais eficientes*
  - *O Shellsort é uma variante da inserção direta*
  - *O Heapsort é uma variante da seleção direta*

# ***Roteiro***

- *Objetivos*
- ***Métodos de ordenação elementares***
  - *Inserção direta*
  - *Seleção direta*
  - ***Bubble sort***
- *Exercícios*

# ***Bubble Sort***

- *Método de ordenação por troca (assim como o Quicksort)*
  - *Idéia básica*
    - *Fazer os mais “leves” borbulharem para cima*
  - *Um dos algoritmos de ordenação menos eficientes*
  - *Implementação*
    - *Várias passagens pelo arquivo, trocando registros fora de lugar*

# *Bubble Sort*

```
25 57 48 37 12 92 86 33
25 57
    57 48
        57 37
            57 12
                57 92
                    92 86
                        92 33

25 48 37 12 57 86 33 92
```

# *Bubble Sort*

25 48 37 12 57 86 33 **92**

25 48

48 37

48 12

48 57

57 86

86 33

25 37 12 48 57 33 **86 92**

# *Bubble Sort*

25 37 12 48 57 33 86 92  
25 12 37 48 33 57 86 92  
12 25 37 33 48 57 86 92  
12 25 33 37 48 57 86 92  
12 25 33 37 48 57 86 92  
12 25 33 37 48 57 86 92  
12 25 33 37 48 57 86 92

# ***Bubble Sort***

- *Algoritmo Básico*

*Repita i, de 0 até tamanho do arquivo*

*Repita j, de 0 até n-i*

*Se  $x[j] > x[j+1]$  troca elementos de posição*

# Bubble Sort

25 37 12 48 57 33 86 92  
25 12 37 48 33 57 86 92  
12 25 37 33 48 57 86 92  
12 25 33 37 48 57 86 92  
12 25 33 37 48 57 86 92  
12 25 33 37 48 57 86 92  
12 25 33 37 48 57 86 92

Passos desnecessários. O arquivo já  
está ordenado nesse ponto

# *Bubble Sort*

- *Algoritmo Básico*

*Repita i, de 0 até tamanho do arquivo ou **não haja troca***

*Repita j, de 0 até  $n-i$*

*Se  $x[j] > x[j+1]$  troca elementos de posição*

# Bubble Sort

- *Possível implementação (Linguagem C)*

```
void bubblesort(int num, int vetor[]) {
    int i, j;
    for (i=num-1; i>=1; i--) {
        int troca=0;
        for (j=0; j<i; j++)
            if (vetor[j]>vetor[j+1]) {
                int aux=vetor[j];
                vetor[j]=vetor[j+1];
                vetor[j+1]=aux;
                troca=1;
            }
        if (troca==0)
            return;
    }
}
```

# ***Roteiro***

- *Objetivos*
- *Métodos de ordenação elementares*
  - *Inserção direta*
  - *Seleção direta*
  - *Bubble sort*
- ***Exercícios***

# ***Exercícios***

- *Implementar os métodos apresentados em Java*
- *Ordenar o vetor a seguir pelos métodos apresentados em aula e verificar o número de trocas necessárias*

43	31	47	59	92	23	2
----	----	----	----	----	----	---