Aula 29 – Ordenação

Norton T. Roman & Luciano A. Digiampietri

Ordenação

- Na última aula aprendemos a realizar Busca Binária

Ordenação

- Na última aula aprendemos a realizar Busca Binária
- Porém, para se utilizar a Busca Binária o arranjo precisa estar ordenado

Ordenação

- Na última aula aprendemos a realizar Busca Binária
- Porém, para se utilizar a Busca Binária o arranjo precisa estar ordenado
- De fato, conjuntos de dados ordenados são utilizados por diferentes algoritmos

Método da Bolha (Bubble Sort)

```
void bolha(int v[], int tam) {
  int ult, i, aux;
  for (ult = tam-1; ult>0; ult--)
    for (i=0; i<ult; i++)
      if (v[i] > v[i+1]) {
        aux = v[i];
        v[i] = v[i+1];
        v[i+1] = aux;
      }
}
```

Método da Bolha (Bubble Sort)

- Percorra todo o arranjo tomando seus elementos adjacentes para a par

```
void bolha(int v[], int tam) {
  int ult, i, aux;
  for (ult = tam-1; ult>0; ult--)
    for (i=0; i<ult; i++)
      if (v[i] > v[i+1]) {
        aux = v[i];
        v[i] = v[i+1];
        v[i+1] = aux;
      }
}
```

Método da Bolha (Bubble Sort)

- Percorra todo o arranjo tomando seus elementos adjacentes para a par
- Se os elemento no par estiverem ordenados, siga ao próximo par

```
void bolha(int v[], int tam) {
  int ult, i, aux;
  for (ult = tam-1; ult>0; ult--)
    for (i=0; i<ult; i++)
      if (v[i] > v[i+1]) {
        aux = v[i];
        v[i] = v[i+1];
        v[i+1] = aux;
      }
}
```

Método da Bolha (Bubble Sort)

- Percorra todo o arranjo tomando seus elementos adjacentes para a par
- Se os elemento no par estiverem ordenados, siga ao próximo par
- Senão, troque-os de lugar

```
void bolha(int v[], int tam) {
  int ult, i, aux;
  for (ult = tam-1; ult>0; ult--)
    for (i=0; i<ult; i++)
      if (v[i] > v[i+1]) {
        aux = v[i];
        v[i] = v[i+1];
        v[i+1] = aux;
      }
}
```

Método da Bolha (Bubble Sort)

- Percorra todo o arranjo tomando seus elementos adjacentes para a par
- Se os elemento no par estiverem ordenados, siga ao próximo par
- Senão, troque-os de lugar
- Repita a operação até que nenhuma troca possa ser feita no arranjo inteiro

```
void bolha(int v[], int tam) {
  int ult, i, aux;
  for (ult = tam-1; ult>0; ult--)
    for (i=0; i<ult; i++)
      if (v[i] > v[i+1]) {
        aux = v[i];
        v[i] = v[i+1];
        v[i+1] = aux;
      }
}
```

Método da Bolha (Bubble Sort)

Ex: ordene em ordem crescente

9 8 4 6 3

Método da Bolha (Bubble Sort)

Ex: ordene em ordem crescente

- Executando a primeira passada:

9 8 4 6 3

Método da Bolha (Bubble Sort)

Ex: ordene em ordem crescente

- Executando a primeira passada:

$$\underline{9 \quad 8} \quad 4 \quad 6 \quad 3$$

- Comparando os dois primeiros números

Método da Bolha (Bubble Sort)

Ex: ordene em ordem crescente

- Executando a primeira passada:

$$\underline{8 \quad 9} \quad 4 \quad 6 \quad 3$$

- Trocando porque $8 < 9$

Método da Bolha (Bubble Sort)

Ex: ordene em ordem crescente

- Executando a primeira passada:

$$8 \quad \underline{9 \quad 4} \quad 6 \quad 3$$

- Comparando segundo e terceiro números

Método da Bolha (Bubble Sort)

Ex: ordene em ordem crescente

- Executando a primeira passada:

$$8 \quad \underline{4 \quad 9} \quad 6 \quad 3$$

- Trocando porque $4 < 9$

Método da Bolha (Bubble Sort)

Ex: ordene em ordem crescente

- Executando a primeira passada:

$$8 \quad 4 \quad \underline{9 \quad 6} \quad 3$$

- Comparando terceiro e quarto números

Método da Bolha (Bubble Sort)

Ex: ordene em ordem crescente

- Executando a primeira passada:

$$8 \quad 4 \quad \underline{6 \quad 9} \quad 3$$

- Trocando porque $6 < 9$

Método da Bolha (Bubble Sort)

Ex: ordene em ordem crescente

- Executando a primeira passada:

$$8 \quad 4 \quad 6 \quad \underline{9 \quad 3}$$

- Comparando quarto e quinto números

Método da Bolha (Bubble Sort)

Ex: ordene em ordem crescente

- Executando a primeira passada:

$$8 \quad 4 \quad 6 \quad \underline{3 \quad 9}$$

- Trocando porque $3 < 9$

Método da Bolha (Bubble Sort)

Ex: ordene em ordem crescente

- Primeira passada completa. Último elemento fixado:Executando a segunda passada:

$$8 \quad 4 \quad 6 \quad \underline{3 \quad \mathbf{9}}$$

Método da Bolha (Bubble Sort)

Ex: ordene em ordem crescente

- Executando a segunda passada:

8	4	6	3	**9**
8	4	6	3	**9**

Método da Bolha (Bubble Sort)

Ex: ordene em ordem crescente

- Executando a segunda passada:

$$\begin{array}{ccccc} 8 & 4 & 6 & 3 & \mathbf{9} \\ \underline{8} & \underline{4} & 6 & 3 & \mathbf{9} \end{array}$$

- Comparando os dois primeiros números

Método da Bolha (Bubble Sort)

Ex: ordene em ordem crescente

- Executando a segunda passada:

$$\begin{array}{ccccc} 8 & 4 & 6 & 3 & \mathbf{9} \\ \underline{4} & \underline{8} & 6 & 3 & \mathbf{9} \end{array}$$

- Trocando porque $4 < 8$

Método da Bolha (Bubble Sort)

Ex: ordene em ordem crescente

- Executando a segunda passada:

$$\begin{array}{ccccc} 8 & 4 & 6 & 3 & \mathbf{9} \\ 4 & \underline{8} & \underline{6} & 3 & \mathbf{9} \end{array}$$

- Comparando segundo e terceiro números

Método da Bolha (Bubble Sort)

Ex: ordene em ordem crescente

- Executando a segunda passada:

$$\begin{array}{cccccc} 8 & 4 & 6 & 3 & \mathbf{9} \\ 4 & \underline{6} & \underline{8} & 3 & \mathbf{9} \end{array}$$

- Trocando porque $6<8$

Método da Bolha (Bubble Sort)

Ex: ordene em ordem crescente

- Executando a segunda passada:

$$\begin{array}{cccc} 8 & 4 & 6 & 3 & \mathbf{9} \\ 4 & 6 & \underline{8} & \underline{3} & \mathbf{9} \end{array}$$

- Comparando terceiro e quarto números

Método da Bolha (Bubble Sort)

Ex: ordene em ordem crescente

- Executando a segunda passada:

$$\begin{array}{cccc} 8 & 4 & 6 & 3 \\ 4 & 6 & 3 & 8 \end{array} \quad \mathbf{9}$$

- Trocando porque $3 < 8$

Método da Bolha (Bubble Sort)

Ex: ordene em ordem crescente

- Segunda passada completa. Último elemento fixado:

$$\begin{array}{cccc} 8 & 4 & 6 & 3 & \mathbf{9} \\ 4 & 6 & 3 & \mathbf{8} & \mathbf{9} \end{array}$$

Método da Bolha (Bubble Sort)

Ex: ordene em ordem crescente

- Executando a terceira passada:

8	4	6	3	**9**
4	6	3	**8**	**9**
4	6	3	**8**	**9**

Método da Bolha (Bubble Sort)

Ex: ordene em ordem crescente

- Executando a terceira passada:

8	4	6	3	**9**
4	6	3	**8**	**9**
4	6	3	**8**	**9**

- Comparando os dois primeiros números

Método da Bolha (Bubble Sort)

Ex: ordene em ordem crescente

- Executando a terceira passada:

8	4	6	3	**9**
4	6	3	**8**	**9**
4	6	3	**8**	**9**

- 6>4, logo não há necessidade de troca

Método da Bolha (Bubble Sort)

Ex: ordene em ordem crescente

- Executando a terceira passada:

$$\begin{array}{ccccc} 8 & 4 & 6 & 3 & \mathbf{9} \\ 4 & 6 & 3 & \mathbf{8} & \mathbf{9} \\ 4 & 6 & 3 & \mathbf{8} & \mathbf{9} \end{array}$$

- Comparando os segundo e terceiro números

Método da Bolha (Bubble Sort)

Ex: ordene em ordem crescente

- Executando a terceira passada:

$$\begin{array}{ccccc} 8 & 4 & 6 & 3 & \mathbf{9} \\ 4 & 6 & 3 & \mathbf{8} & \mathbf{9} \\ 4 & \underline{3} & \underline{6} & \mathbf{8} & \mathbf{9} \end{array}$$

- Trocando porque $3<6$

Método da Bolha (Bubble Sort)

Ex: ordene em ordem crescente

- Terceira passada completa. Último elemento fixado:

8	4	6	3	**9**
4	6	3	**8**	**9**
4	3	**6**	**8**	**9**

Método da Bolha (Bubble Sort)

Ex: ordene em ordem crescente

- Executando a quarta passada:

8	4	6	3	**9**
4	6	3	**8**	**9**
4	3	**6**	**8**	**9**
4	3	**6**	**8**	**9**

Método da Bolha (Bubble Sort)

Ex: ordene em ordem crescente

- Executando a quarta passada:

8	4	6	3	**9**
4	6	3	**8**	**9**
4	3	**6**	**8**	**9**
4	3	**6**	**8**	**9**

- Comparando os dois primeiros números

Método da Bolha (Bubble Sort)

Ex: ordene em ordem crescente

- Executando a quarta passada:

8	4	6	3	**9**
4	6	3	**8**	**9**
4	3	**6**	**8**	**9**
3	4	**6**	**8**	**9**

- Trocando porque $3<4$

Método da Bolha (Bubble Sort)

Ex: ordene em ordem crescente

- Quarta passada completa. Último elemento fixado:

8	4	6	3	**9**
4	6	3	**8**	**9**
4	3	**6**	**8**	**9**
3	**4**	**6**	**8**	**9**

Método da Bolha (Bubble Sort)

Ex: ordene em ordem crescente

- Quarta passada completa. Último elemento fixado:

8	4	6	3	**9**
4	6	3	**8**	**9**
4	3	**6**	**8**	**9**
3	**4**	**6**	**8**	**9**

- Há somente um elemento no arranjo. O algoritmo para. O arranjo está ordenado.

Método da Bolha (Bubble Sort)

- Observe que não precisamos correr sempre o arranjo até o final

```
void bolha(int v[], int tam) {
  int ult, i, aux;
  for (ult = tam-1; ult>0; ult--)
    for (i=0; i<ult; i++)
      if (v[i] > v[i+1]) {
        aux = v[i];
        v[i] = v[i+1];
        v[i+1] = aux;
      }
}

int main() {
  int i;
  int v[] = {55,0,-78,-4,32,200,52,63,
                                    69,125};
  bolha(v,10);

  for (i=0;i<10;i++) printf("%i ",v[i]);
  printf("\n");
  return 0;
}
```

Método da Bolha (Bubble Sort)

- Observe que não precisamos correr sempre o arranjo até o final
  - Basta irmos até onde garantimos estar ordenado...

```
void bolha(int v[], int tam) {
  int ult, i, aux;
  for (ult = tam-1; ult>0; ult--)
    for (i=0; i<ult; i++)
      if (v[i] > v[i+1]) {
        aux = v[i];
        v[i] = v[i+1];
        v[i+1] = aux;
      }
}

int main() {
  int i;
  int v[] = {55,0,-78,-4,32,200,52,63,
                                    69,125};
  bolha(v,10);

  for (i=0;i<10;i++) printf("%i ",v[i]);
  printf("\n");
  return 0;
}
```

Método da Bolha (Bubble Sort)

- Observe que não precisamos correr sempre o arranjo até o final
  - Basta irmos até onde garantimos estar ordenado...
  - Marcando esse final

```
void bolha(int v[], int tam) {
  int ult, i, aux;
  for (ult = tam-1; ult>0; ult--)
    for (i=0; i<ult; i++)
      if (v[i] > v[i+1]) {
        aux = v[i];
        v[i] = v[i+1];
        v[i+1] = aux;
      }
}

int main() {
  int i;
  int v[] = {55,0,-78,-4,32,200,52,63,
                                    69,125};
  bolha(v,10);

  for (i=0;i<10;i++) printf("%i ",v[i]);
  printf("\n");
  return 0;
}
```

Método da Bolha (Bubble Sort)

- Observe que não precisamos correr sempre o arranjo até o final
  - Basta irmos até onde garantimos estar ordenado...
  - Marcando esse final
  - Sempre corremos de 0 a `ult`

```
void bolha(int v[], int tam) {
  int ult, i, aux;
  for (ult = tam-1; ult>0; ult--)
    for (i=0; i<ult; i++)
      if (v[i] > v[i+1]) {
        aux = v[i];
        v[i] = v[i+1];
        v[i+1] = aux;
      }
}

int main() {
  int i;
  int v[] = {55,0,-78,-4,32,200,52,63,
                                    69,125};
  bolha(v,10);

  for (i=0;i<10;i++) printf("%i ",v[i]);
  printf("\n");
  return 0;
}
```

Método da Bolha (Bubble Sort)

- Observe que não precisamos correr sempre o arranjo até o final
  - Basta irmos até onde garantimos estar ordenado...
  - Marcando esse final
  - Sempre corremos de 0 a `ult`
  - E a cada passada, decrementamos `ult`

```
void bolha(int v[], int tam) {
  int ult, i, aux;
  for (ult = tam-1; ult>0; ult--)
    for (i=0; i<ult; i++)
      if (v[i] > v[i+1]) {
        aux = v[i];
        v[i] = v[i+1];
        v[i+1] = aux;
      }
}

int main() {
  int i;
  int v[] = {55,0,-78,-4,32,200,52,63,
                                    69,125};
  bolha(v,10);

  for (i=0;i<10;i++) printf("%i ",v[i]);
  printf("\n");
  return 0;
}
```

Método da Bolha (Bubble Sort)

Saída

```
-78 -4 0 32 52 55 63 69 125 200
```

```
void bolha(int v[], int tam) {
  int ult, i, aux;
  for (ult = tam-1; ult>0; ult--)
    for (i=0; i<ult; i++)
      if (v[i] > v[i+1]) {
        aux = v[i];
        v[i] = v[i+1];
        v[i+1] = aux;
      }
}

int main() {
  int i;
  int v[] = {55,0,-78,-4,32,200,52,63,
                                    69,125};
  bolha(v,10);

  for (i=0;i<10;i++) printf("%i ",v[i]);
  printf("\n");
  return 0;
}
```

Método da Bolha (Bubble Sort)

Saída

```
-78 -4 0 32 52 55 63 69 125 200
```

Cuidado!

Esse método modifica o arranjo original!

```
void bolha(int v[], int tam) {
  int ult, i, aux;
  for (ult = tam-1; ult>0; ult--)
    for (i=0; i<ult; i++)
      if (v[i] > v[i+1]) {
        aux = v[i];
        v[i] = v[i+1];
        v[i+1] = aux;
      }
}

int main() {
  int i;
  int v[] = {55,0,-78,-4,32,200,52,63,
                                    69,125};
  bolha(v,10);

  for (i=0;i<10;i++) printf("%i ",v[i]);
  printf("\n");
  return 0;
}
```

Ordenação por Seleção (Selection Sort)

Algoritmo:

Ordenação por Seleção (Selection Sort)

Algoritmo:

- Primeiro encontre o menor elemento do arranjo

Ordenação por Seleção (Selection Sort)

Algoritmo:

- Primeiro encontre o menor elemento do arranjo
- Então troque esse elemento de lugar com o que está na primeira posição

Ordenação por Seleção (Selection Sort)

Algoritmo:

- Primeiro encontre o menor elemento do arranjo
- Então troque esse elemento de lugar com o que está na primeira posição
- Encontre o segundo menor do arranjo

Ordenação por Seleção (Selection Sort)

Algoritmo:

- Primeiro encontre o menor elemento do arranjo
- Então troque esse elemento de lugar com o que está na primeira posição
- Encontre o segundo menor do arranjo
- Troque com o da segunda posição

Ordenação por Seleção (Selection Sort)

Algoritmo:

- Primeiro encontre o menor elemento do arranjo
- Então troque esse elemento de lugar com o que está na primeira posição
- Encontre o segundo menor do arranjo
- Troque com o da segunda posição
- E assim por diante, até chegar ao fim do arranjo

Ordenação por Seleção (Selection Sort)

Ex: ordene em ordem crescente

9 8 4 6 3

Ordenação por Seleção (Selection Sort)

Ex: ordene em ordem crescente

- Executando a primeira passada:

$$\begin{array}{ccccc} 9 & 8 & 4 & 6 & 3 \\ \underline{9} & 8 & 4 & 6 & 3 \end{array}$$

Ordenação por Seleção (Selection Sort)

Ex: ordene em ordem crescente

- Executando a primeira passada:

$$\begin{array}{ccccc} 9 & 8 & 4 & 6 & 3 \\ \underline{9} & 8 & 4 & 6 & \underline{3} \end{array}$$

- Encontrando o menor elemento

Ordenação por Seleção (Selection Sort)

Ex: ordene em ordem crescente

- Executando a primeira passada:

$$\begin{array}{ccccc} 9 & 8 & 4 & 6 & 3 \\ \underline{3} & 8 & 4 & 6 & \underline{9} \end{array}$$

- Trocando com o primeiro elemento, pois $3 < 9$

Ordenação por Seleção (Selection Sort)

Ex: ordene em ordem crescente

- Primeira passada completa. Primeiro elemento fixado:

9	8	4	6	3
3	8	4	6	9

Ordenação por Seleção (Selection Sort)

Ex: ordene em ordem crescente

- Executando a segunda passada:

9	8	4	6	3
3	8	4	6	9
3	8	4	6	9

Ordenação por Seleção (Selection Sort)

Ex: ordene em ordem crescente

- Executando a segunda passada:

9	8	4	6	3
3	8	4	6	9
3	8	4	6	9

- Encontrando o segundo menor elemento

Ordenação por Seleção (Selection Sort)

Ex: ordene em ordem crescente

- Executando a segunda passada:

$$\begin{array}{ccccc} 9 & 8 & 4 & 6 & 3 \\ \mathbf{3} & 8 & 4 & 6 & 9 \\ \mathbf{3} & 4 & 8 & 6 & 9 \end{array}$$

- Trocando com o segundo elemento, pois $4 < 8$

Ordenação por Seleção (Selection Sort)

Ex: ordene em ordem crescente

- Segunda passada completa. Segundo elemento fixado:

$$\begin{array}{ccccc} 9 & 8 & 4 & 6 & 3 \\ \mathbf{3} & 8 & 4 & 6 & 9 \\ \mathbf{3} & \underline{\mathbf{4}} & 8 & 6 & 9 \end{array}$$

Ordenação por Seleção (Selection Sort)

Ex: ordene em ordem crescente

- Executando a terceira passada:

9	8	4	6	3
3	8	4	6	9
3	**4**	8	6	9
3	**4**	8	6	9

Ordenação por Seleção (Selection Sort)

Ex: ordene em ordem crescente

- Executando a terceira passada:

9	8	4	6	3
3	8	4	6	9
3	**4**	8	6	9
3	**4**	8	6	9

- Encontrando o terceiro menor elemento

Ordenação por Seleção (Selection Sort)

Ex: ordene em ordem crescente

- Executando a terceira passada:

$$\begin{array}{ccccc} 9 & 8 & 4 & 6 & 3 \\ \mathbf{3} & 8 & 4 & 6 & 9 \\ \mathbf{3} & \mathbf{4} & 8 & 6 & 9 \\ \mathbf{3} & \mathbf{4} & 6 & 8 & 9 \end{array}$$

- Trocando com o terceiro elemento, pois $6 < 8$

Ordenação por Seleção (Selection Sort)

Ex: ordene em ordem crescente

- Terceira passada completa. Terceiro elemento fixado:

9	8	4	6	3
3	8	4	6	9
3	**4**	8	6	9
3	**4**	**6**	8	9

Ordenação por Seleção (Selection Sort)

Ex: ordene em ordem crescente

- Executando a quarta passada:

9	8	4	6	3
3	8	4	6	9
3	**4**	8	6	9
3	**4**	**6**	8	9
3	**4**	**6**	8	9

Ordenação por Seleção (Selection Sort)

Ex: ordene em ordem crescente

- Executando a quarta passada:

9	8	4	6	3
3	8	4	6	9
3	**4**	8	6	9
3	**4**	**6**	8	9
3	**4**	**6**	8	9

- Encontrando o quarto menor elemento

Ordenação por Seleção (Selection Sort)

Ex: ordene em ordem crescente

- Executando a quarta passada:

9	8	4	6	3
3	8	4	6	9
3	**4**	8	6	9
3	**4**	**6**	8	9
3	**4**	**6**	8	9

- Não há troca, pois já está na quarta posição

Ordenação por Seleção (Selection Sort)

Ex: ordene em ordem crescente

- Quarta passada completa. Quarto elemento fixado:

9	8	4	6	3
3	8	4	6	9
3	**4**	8	6	9
3	**4**	**6**	8	9
3	**4**	**6**	**8**	9

Ordenação por Seleção (Selection Sort)

Ex: ordene em ordem crescente

- Executando a quinta passada:

9	8	4	6	3
3	8	4	6	9
3	**4**	8	6	9
3	**4**	**6**	8	9
3	**4**	**6**	**8**	9
3	**4**	**6**	**8**	9

Ordenação por Seleção (Selection Sort)

Ex: ordene em ordem crescente

- Executando a quinta passada:

9	8	4	6	3
3	8	4	6	9
3	**4**	8	6	9
3	**4**	**6**	8	9
3	**4**	**6**	**8**	9
3	**4**	**6**	**8**	9

- Última posição do arranjo. O algoritmo para.

Ordenação por Seleção (Selection Sort)

Ex: ordene em ordem crescente

- Quinta passada completa. Quinto elemento fixado:

9	8	4	6	3
3	8	4	6	9
3	**4**	8	6	9
3	**4**	**6**	8	9
3	**4**	**6**	**8**	9
3	**4**	**6**	**8**	**9**

Ordenação por Seleção (Selection Sort)

```
int posMenorEl(int v[], int tam, int inicio){
  int i, posMenor;
  posMenor = -1;
  if ((inicio>=0) && (inicio < tam)) {
    posMenor = inicio;
    for (i=inicio+1; i<tam; i++)
      if (v[i] < v[posMenor]) posMenor = i;
  }
  return(posMenor);
}

void selecao(int v[], int tam) {
  int i, posMenor, aux;
  for (i=0; i<tam-1; i++) {
    posMenor = posMenorEl(v,tam,i);
    aux = v[i];
    v[i] = v[posMenor];
    v[posMenor] = aux;
  }
}
```

Ordenação por Seleção (Selection Sort)

- Criamos um método que diz a posição do menor elemento em subvetor
  $inicio \leq i < fim$

```
int posMenorEl(int v[], int tam, int inicio){
  int i, posMenor;
  posMenor = -1;
  if ((inicio>=0) && (inicio < tam)) {
    posMenor = inicio;
    for (i=inicio+1; i<tam; i++)
      if (v[i] < v[posMenor]) posMenor = i;
  }
  return(posMenor);
}

void selecao(int v[], int tam) {
  int i, posMenor, aux;
  for (i=0; i<tam-1; i++) {
    posMenor = posMenorEl(v,tam,i);
    aux = v[i];
    v[i] = v[posMenor];
    v[posMenor] = aux;
  }
}
```

Ordenação por Seleção (Selection Sort)

- Criamos um método que diz a posição do menor elemento em subvetor
  $inicio \leq i < fim$
  - Sempre é bom testar a entrada

```
int posMenorEl(int v[], int tam, int inicio){
  int i, posMenor;
  posMenor = -1;
  if ((inicio>=0) && (inicio < tam)) {
    posMenor = inicio;
    for (i=inicio+1; i<tam; i++)
      if (v[i] < v[posMenor]) posMenor = i;
  }
  return(posMenor);
}

void selecao(int v[], int tam) {
  int i, posMenor, aux;
  for (i=0; i<tam-1; i++) {
    posMenor = posMenorEl(v,tam,i);
    aux = v[i];
    v[i] = v[posMenor];
    v[posMenor] = aux;
  }
}
```

Ordenação por Seleção (Selection Sort)

- Para cada elemento do arranjo (exceto o último, que sobra já ordenado)

```
int posMenorEl(int v[], int tam, int inicio){
  int i, int posMenor;
  posMenor = -1;
  if ((inicio>=0) && (inicio < tam)) {
    posMenor = inicio;
    for (i=inicio+1; i<tam; i++)
      if (v[i] < v[posMenor]) posMenor = i;
  }
  return(posMenor);
}

void selecao(int v[], int tam) {
  int i, posMenor, aux;
  for (i=0; i<tam-1; i++) {
    posMenor = posMenorEl(v,tam,i);
    aux = v[i];
    v[i] = v[posMenor];
    v[posMenor] = aux;
  }
}
```

Ordenação por Seleção (Selection Sort)

- Para cada elemento do arranjo (exceto o último, que sobra já ordenado)
  - Busca o menor elemento a partir deste

```
int posMenorEl(int v[], int tam, int inicio){
  int i, int posMenor;
  posMenor = -1;
  if ((inicio>=0) && (inicio < tam)) {
    posMenor = inicio;
    for (i=inicio+1; i<tam; i++)
      if (v[i] < v[posMenor]) posMenor = i;
  }
  return(posMenor);
}

void selecao(int v[], int tam) {
  int i, posMenor, aux;
  for (i=0; i<tam-1; i++) {
    posMenor = posMenorEl(v,tam,i);
    aux = v[i];
    v[i] = v[posMenor];
    v[posMenor] = aux;
  }
}
```

Ordenação por Seleção (Selection Sort)

- Para cada elemento do arranjo (exceto o último, que sobra já ordenado)
  - Busca o menor elemento a partir deste
  - Troca com a posição desse elemento

```
int posMenorEl(int v[], int tam, int inicio){
  int i, int posMenor;
  posMenor = -1;
  if ((inicio>=0) && (inicio < tam)) {
    posMenor = inicio;
    for (i=inicio+1; i<tam; i++)
      if (v[i] < v[posMenor]) posMenor = i;
  }
  return(posMenor);
}

void selecao(int v[], int tam) {
  int i, posMenor, aux;
  for (i=0; i<tam-1; i++) {
    posMenor = posMenorEl(v,tam,i);
    aux = v[i];
    v[i] = v[posMenor];
    v[posMenor] = aux;
  }
}
```

Ordenação por Seleção (Selection Sort)

- E como fazer um *Selection Sort* sem método auxiliar?

```
int posMenorEl(int v[], int tam, int inicio){
  int i, posMenor;
  posMenor = -1;
  if ((inicio>=0) && (inicio < tam)) {
    posMenor = inicio;
    for (i=inicio+1; i<tam; i++)
      if (v[i] < v[posMenor]) posMenor = i;
  }
  return(posMenor);
}

void selecao(int v[], int tam) {
  int i, posMenor, aux;
  for (i=0; i<tam-1; i++) {
    posMenor = posMenorEl(v,tam,i);
    aux = v[i];
    v[i] = v[posMenor];
    v[posMenor] = aux;
  }
}
```

Ordenação por Seleção (Selection Sort)

- E como fazer um *Selection Sort* sem método auxiliar?

```
int posMenorEl(int v[], int tam, int inicio){
  int i, posMenor;
  posMenor = -1;
  if ((inicio>=0) && (inicio < tam)) {
    posMenor = inicio;
    for (i=inicio+1; i<tam; i++)
      if (v[i] < v[posMenor]) posMenor = i;
  }
  return(posMenor);
}

void selecao(int v[], int tam) {
  int i, posMenor, aux;
  for (i=0; i<tam-1; i++) {
    int posMenor = posMenorEl(v,tam,i);
    aux = v[i];
    v[i] = v[posMenor];
    v[posMenor] = aux;
  }
}
```

Ordenação por Seleção (Selection Sort)

- E como fazer um *Selection Sort* sem método auxiliar?

```
void selecao(int v[], int tam) {
  int i, p, aux, posMenor;
  for (i=0; i<tam-1; i++) {
    posMenor =  i;
    for (p=i+1; p<tam; p++)
      if (v[p] < v[posMenor])
        posMenor = p;
    aux = v[i];
    v[i] = v[posMenor];
    v[posMenor] = aux;
  }
}
```

```
int posMenorEl(int v[], int tam, int inicio){
  int i, posMenor;
  posMenor = -1;
  if ((inicio>=0) && (inicio < tam)) {
    posMenor = inicio;
    for (i=inicio+1; i<tam; i++)
      if (v[i] < v[posMenor]) posMenor = i;
  }
  return(posMenor);
}

void selecao(int v[], int tam) {
  int i, posMenor, aux;
  for (i=0; i<tam-1; i++) {
    int posMenor = posMenorEl(v,tam,i);
    aux = v[i];
    v[i] = v[posMenor];
    v[posMenor] = aux;
  }
}
```

Curiosidades

- Bolha:
  - `http://www.youtube.com/watch?v=lyZQPjUT5B4`
- Seleção:
  - `http://www.youtube.com/watch?v=Ns4TPTC8whw` (versão levemente diferente do algoritmo)

Aula 29 – Ordenação

Norton T. Roman & Luciano A. Digiampietri