

# Linguagem C: Algoritmos de Ordenação

Prof. Paulo R. S. L. Coelho

paulo@facom.ufu.br

Faculdade de Computação  
Universidade Federal de Uberlândia

GEQ007



# Organização

- 1 Introdução
- 2 Algoritmos de Ordenação
  - Algoritmo de Inserção
  - Algoritmo de Seleção
  - Algoritmo de Ordenação por Troca
- 3 Exercícios



# Organização

- 1 Introdução
- 2 Algoritmos de Ordenação
  - Algoritmo de Inserção
  - Algoritmo de Seleção
  - Algoritmo de Ordenação por Troca
- 3 Exercícios



# Introdução

- Uma das aplicações mais estudadas e realizadas sobre vetores é a **ordenação**.
- Ordenar um vetor significa permutar seus elementos de tal forma que eles fiquem em ordem crescente, ou seja,  $v[0] \leq v[1] \leq v[2] \leq \dots \leq v[n - 1]$ .
- Por exemplo, suponha o vetor  $v = 5, 6, -9, 9, 0, 4$ . Uma ordenação desse vetor resultaria em um rearranjo de seus elementos:  $v = -9, 0, 4, 5, 6, 9$ .



# Introdução

- Existem diversos algoritmos de ordenação para vetores.
- Eles variam em relação à dificuldade de implementação e desempenho. Usualmente algoritmos mais fáceis de serem implementados apresentam desempenho inferior.
- Veremos 3 algoritmos diferentes de ordenação:
  - 1 Algoritmo de Inserção;
  - 2 Algoritmo de Seleção; e
  - 3 Algoritmo de Intercalação.



# Organização

1 Introdução

2 Algoritmos de Ordenação

- Algoritmo de Inserção
- Algoritmo de Seleção
- Algoritmo de Ordenação por Troca

3 Exercícios



# Algoritmo de Inserção (Insertion Sort) I

- Um dos algoritmos de implementação mais simples. Método de ordenação semelhante ao que usamos para ordenar as cartas de um baralho.
- Idéia básica:
  - Compare a chave ( $x$ ) com os elementos à sua esquerda, deslocando para direita cada elemento maior do que a chave;
  - Insira a chave na posição correta à sua esquerda, onde os elementos já estão ordenados;
  - Repita os passos anteriores atualizando a chave para a próxima posição à direita até o fim do vetor.

- Exemplo: 

0	crescente	j-1	j	n-1						
444	555	555	666	777	222	999	222	999	222	999



## Algoritmo de Inserção (Insertion Sort) II

- PSEUDOCÓDIGO: suponha um vetor  $v$  de tamanho  $n$ .

```
DECLARE  $i, j, x, n, v[n]$  NUMERICO;  
PARA  $i = 1$  ATÉ  $i < n$  FAÇA  
  INICIO  
     $x = v[i];$   
     $j = i - 1;$   
    ENQUANTO  $j \geq 0$  e  $v[j] > x$  FAÇA  
      INÍCIO  
         $v[j+1] = v[j];$   
         $j = j-1;$   
      FIM  
     $v[j+1] = x;$   
  FIM
```





# Algoritmo de Seleção (Selection Sort) I

- Implementação muito simples.
- Idéia básica:
  - Selecione o menor elemento do vetor;
  - Troque esse elemento com o elemento da primeira posição do vetor;
  - Repita as duas operações anteriores considerando apenas os  $n-1$  elementos restantes, em seguida repita com os  $n-2$  elementos restantes; e assim sucessivamente até que reste apenas um elemento no vetor a ser considerado.

- Exemplo: 

110	120	120	130	140	999	666	999	666	999	666
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

- PSEUDOCÓDIGO: suponha um vetor  $v$  de tamanho  $n$ .



## Algoritmo de Seleção (Selection Sort) II

```
DECLARE i, j, aux, n, min, v[n] NUMERICO;  
PARA i = 0 ATÉ i < n-1 FAÇA  
  INICIO  
    min = i;  
    PARA j = i+1 ATÉ j < n FAÇA  
      INICIO  
        SE v[j] < v[min] ENTÃO  
          min = j;  
      FIM  
    FIM  
    aux = v[i]; v[i] = v[min]; v[min] = aux;  
  FIM
```



# Ordenação por Troca (BubbleSort) I

- Outro algoritmo simples, útil para ordenação de vetores pequenos (desempenho ruim).
- Idéia básica:
  - Compare o primeiro elemento com o segundo. Se estiverem desordenados, então efetue a troca de posição. Compare o segundo elemento com o terceiro e efetue a troca de posição, se necessário;
  - Repita a operação anterior até que o penúltimo elemento seja comparado com o último. Ao final desta repetição o elemento de maior valor estará em sua posição correta, a  $n$ -ésima posição do vetor;
  - Continue a ordenação posicionando o segundo maior elemento, o terceiro,..., até que todo o vetor esteja ordenado.



# Ordenação por Troca (BubbleSort) II

● Exemplo:

<b>j=1</b>	2	3	4	5	<b>i=6</b>
10	9	8	7	6	5

1	<b>j=2</b>	3	4	5	<b>i=6</b>
9	10	8	7	6	5

1	2	<b>j=3</b>	4	5	<b>i=6</b>
9	8	10	7	6	5

1	2	3	<b>j=4</b>	5	<b>i=6</b>
9	8	7	10	6	5

1	2	3	4	<b>j=5</b>	<b>i=6</b>
9	8	7	6	10	5

Final da primeira iteração

1	2	3	4	5	6
9	8	7	6	5	10

<b>j=1</b>	2	3	4	<b>i=5</b>	6
9	8	7	6	5	10

1	<b>j=2</b>	3	4	<b>i=5</b>	6
8	9	7	6	5	10

1	2	<b>j=3</b>	4	<b>i=5</b>	6
8	7	9	6	5	10

1	2	3	<b>j=4</b>	<b>i=5</b>	6
8	7	6	9	5	10

Final da segunda iteração

1	2	3	4	5	6
8	7	6	5	9	10



## Ordenação por Troca (BubbleSort) III

- PSEUDOCÓDIGO: suponha um vetor  $v$  de tamanho  $n$ .

```
DECLARE i, j, aux, n, v[n] NUMÉRICO;  
PARA i = n-1 ATÉ i > 0 FAÇA  
  INÍCIO  
    PARA j = 0 ATÉ j < i FAÇA  
      INÍCIO  
        SE v[j] > v[j+1]  
          aux = v[j]; v[j] = v[j+1]; v[j+1] = aux;  
        FIM  
      FIM  
    FIM  
  FIM
```



# Organização

- 1 Introdução
- 2 Algoritmos de Ordenação
  - Algoritmo de Inserção
  - Algoritmo de Seleção
  - Algoritmo de Ordenação por Troca
- 3 Exercícios



# Exercícios

- 1 Implemente na linguagem C o algoritmo de ordenação *insertion sort*. Utilize uma função auxiliar para implementar a ordenação.
- 2 Implemente na linguagem C o algoritmo de ordenação *selection sort*. Utilize uma função auxiliar para implementar a ordenação.
- 3 Implemente na linguagem C o algoritmo de ordenação *bubblesort*. Utilize uma função auxiliar para implementar a ordenação.



# Respostas I

```
1 #include <stdio.h>
#include <stdlib.h>

void insertionSort(int v[200], int n)
{
    int i, j, x;
    for(i = 1; i < n; i++) {
        x = v[i];
        j = i - 1;
        while(j >= 0 && v[j] > x) {
            v[j+1] = v[j];
            j--;
        }
        v[j+1] = x;
    }
}
```



# Respostas II

```
int main()
{
    int v[200], n, i;
    printf("Entre tamanho desejado do vetor: ");
    scanf("%d", &n);
    printf("Entre os %d elementos do vetor:\n", n);
    for(i = 0; i < n; i++) {
        scanf("%d", &v[i]);
    }
    insertionSort(v, n);
    printf("\n\nVetor ordenado:\n");
    for(i = 0; i < n; i++) {
        printf("%d\t", v[i]);
    }
    printf("\n");
    return 0;
}
```

# Respostas III

2

```
#include <stdio.h>
#include <stdlib.h>

void selectionSort(int v[200], int n)
{
    int i, j, aux, min;
    for(i = 0; i < n-1; i++) {
        min = i;
        for(j = i+1; j < n; j++) {
            if(v[j] < v[min]) {
                min = j;
            }
        }
        aux = v[i]; v[i] = v[min]; v[min] = aux; //troca
    }
}
```

# Respostas IV

```
int main()
{
    int v[200], n, i;
    printf("Entre tamanho desejado do vetor: ");
    scanf("%d", &n);
    printf("Entre os %d elementos do vetor:\n", n);
    for(i = 0; i < n; i++) {
        scanf("%d", &v[i]);
    }
    selectionSort(v, n);
    printf("\n\nVetor ordenado:\n");
    for(i = 0; i < n; i++) {
        printf("%d\t", v[i]);
    }
    printf("\n");
    return 0;
}
```

# Respostas V

```
3 #include <stdio.h>
#include <stdlib.h>

void bubbleSort(int v[200], int n)
{
    int i, j, aux;
    for(i = n-1; i > 0; i--) {
        for(j = 0; j < i; j++) {
            if(v[j] > v[j+1]) {
                aux = v[j]; v[j] = v[j+1]; v[j+1] = aux; //troca
            }
        }
    }
}
```

# Respostas VI

```
int main()
{
    int v[200], n, i;
    printf("Entre tamanho desejado do vetor: ");
    scanf("%d", &n);
    printf("Entre os %d elementos do vetor:\n", n);
    for(i = 0; i < n; i++) {
        scanf("%d", &v[i]);
    }
    bubbleSort(v, n);
    printf("\n\nVetor ordenado:\n");
    for(i = 0; i < n; i++) {
        printf("%d\t", v[i]);
    }
    printf("\n");
    return 0;
}
```