

# MC102

## Algoritmos e Programação de Computadores

---

Aula de Laboratório 10  
Instituto de Computação  
Primeiro Semestre de 2012

---

14 de maio de 2012



# Conteúdo

- 1 Bubble-Sort
- 2 Insertion-Sort



# Ordenação

- **Ordenação** está presente em vários problemas cotidianos;

## Definição

Dado uma coleção de elementos com uma relação de ordem entre si, devemos gerar uma saída com os elementos ordenados.



# Bubble-Sort

- Dado um vetor, **vet**, de inteiros;
- A ideia do algoritmo é:
  - 1 Compare **vet[0]** com **vet[1]** e troque-os se **vet[0] > vet[1]**;
  - 2 Compare **vet[1]** com **vet[2]** e troque-os se **vet[1] > vet[2]**;
  - 3 ...
  - 4 Compare **vet[tam - 2]** com **vet[tam - 1]** e troque-os se **vet[tam - 2] > vet[tam - 1]**;



# Bubble-Sort

- Dado um vetor, **vet**, de inteiros;
- A ideia do algoritmo é:
  - 1 Compare **vet[0]** com **vet[1]** e troque-os se **vet[0] > vet[1]**;
  - 2 Compare **vet[1]** com **vet[2]** e troque-os se **vet[1] > vet[2]**;
  - 3 ...
  - 4 Compare **vet[tam - 2]** com **vet[tam - 1]** e troque-os se **vet[tam - 2] > vet[tam - 1]**;
- No final da primeira interação o maior elemento estará na última posição do vetor;

## Ideia do Bubble-Sort

O bubble-sort percorre o vetor **tam-1** vezes, e em cada uma delas colocar o maior elemento no lugar correto.



# Bubble-Sort

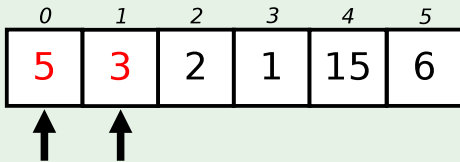
## Exemplo

<i>0</i>	<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>
5	3	2	1	15	6



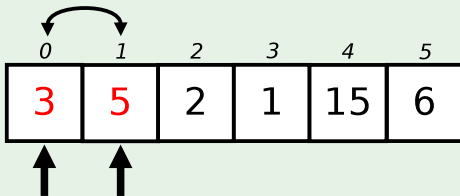
# Bubble-Sort

## Exemplo



# Bubble-Sort

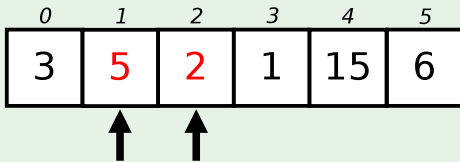
## Exemplo





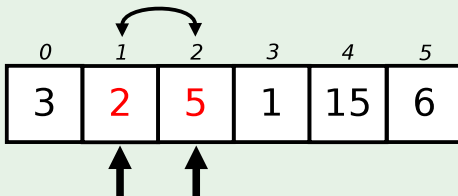
# Bubble-Sort

## Exemplo



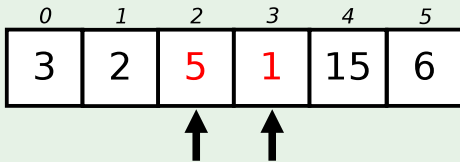
# Bubble-Sort

## Exemplo



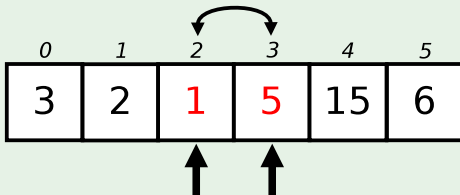
# Bubble-Sort

## Exemplo



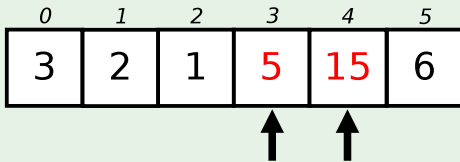
# Bubble-Sort

## Exemplo



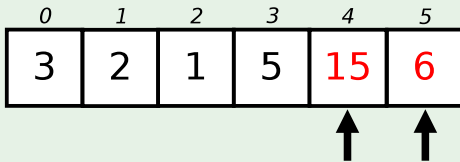
# Bubble-Sort

## Exemplo



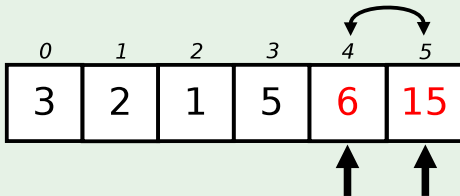
# Bubble-Sort

## Exemplo



# Bubble-Sort

## Exemplo



# Bubble-Sort

## Problema

Como ficaria as trocas de uma interação do **Bubble-Sort**, na qual apenas são comparados e trocados os elementos das posições: 0 e 1; 1 e 2; ...;  $i - 1$  e  $i$ .





# Bubble-Sort

## Solução

```
1 #include <stdio.h>
2
3 int main(){
4     int j, aux, i=5, vet[]={5,3,2,1,15,6};
5     for(j = 0; j < i; j++){
6         if(vet[j] > vet[j+1]){
7             aux = vet[j];
8             vet[j] = vet[j+1];
9             vet[j+1] = aux;
10        }
11    }
12    for(j = 0; j < 6; j++)
13        printf(" %d",vet[j]);
14    printf("\n");
15 }
```



# Bubble-Sort

## Problema

Utilizando o código anterior, faça o algoritmo **Bubble-Sort** para a ordenação do vetor  $\text{vet}[] = \{5, 3, 2, 1, 15, 6\}$ .

**Notem que:**

- As trocas na primeira iteração ocorrem até a **última posição**;
- Na segunda iteração ocorrem até a **penúltima posição**;
- E assim sucessivamente....



# Bubble-Sort

## Solução

```
1 #include <stdio.h>
2
3 int main(){
4     int j, aux, i, vet[]={5,3,2,1,15,6};
5     for(i = 0; i < 6; i++){
6         for(j = 0; j < i; j++){
7             if(vet[j] > vet[j+1]){
8                 aux = vet[j];
9                 vet[j] = vet[j+1];
10                vet[j+1] = aux;
11            }
12        }
13    }
14    for(i = 0; i < 6; i++)
15        printf(" %d",vet[i]);
16    printf("\n");
17 }
```

# Insertion-Sort

- Dado um vetor, **vet**, de inteiros;
- A ideia do algoritmo é:
  - 1 Devemos inserir o item da posição ***i*** na posição;
  - 2 Para isso, armazena-se o valor da posição ***i*** em uma variável auxiliar, **aux**, e em seguida:
    - 1 Compara a posição ***i-1*** com **aux**, caso **aux** seja menor deve-se copiar o valor da posição ***i-1*** para a posição ***i***;
    - 2 De forma semelhante, compara a posição ***i-2*** com **aux**, caso **aux** seja menor deve-se copiar o valor da posição ***i-2*** para a posição ***i-1***;
    - 3 Assim sucessivamente até que **aux** seja maior.

## Ideia do Insertion-Sort

O Insertion-Sort percorre o vetor de elementos da esquerda para a direita e à medida que avança vai deixando os elementos mais à esquerda ordenados.



# Insertion-Sort

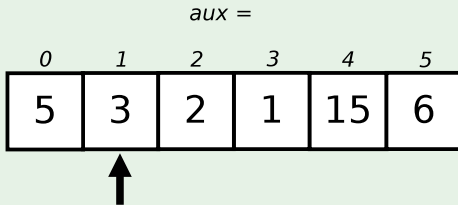
## Exemplo

<i>0</i>	<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>
5	3	2	1	15	6



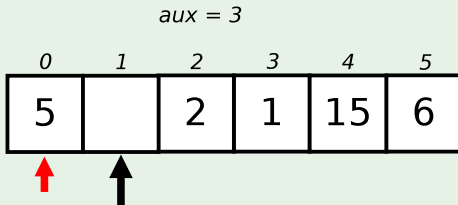
# Insertion-Sort

## Exemplo



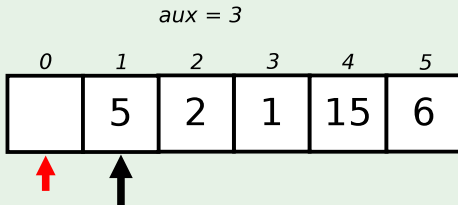
# Insertion-Sort

## Exemplo



# Insertion-Sort

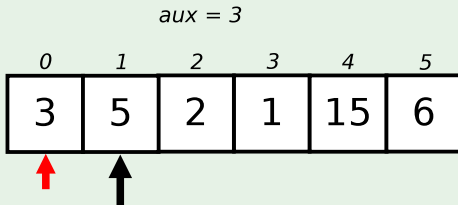
## Exemplo





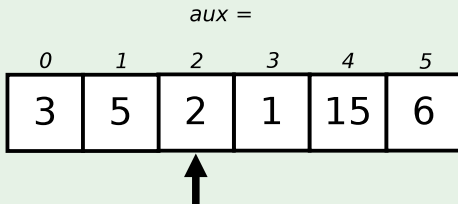
# Insertion-Sort

## Exemplo



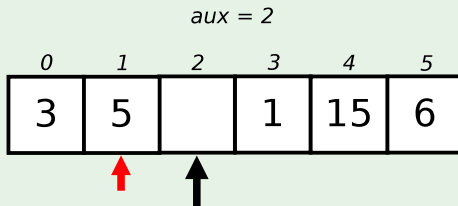
# Insertion-Sort

## Exemplo



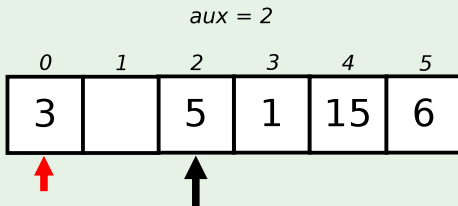
# Insertion-Sort

## Exemplo



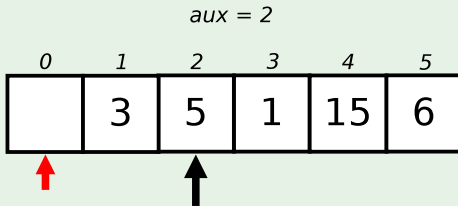
# Insertion-Sort

## Exemplo



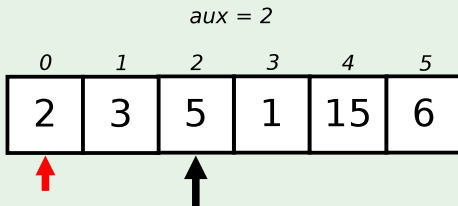
# Insertion-Sort

## Exemplo



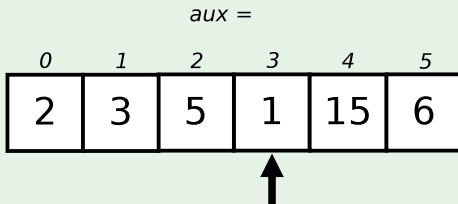
# Insertion-Sort

## Exemplo



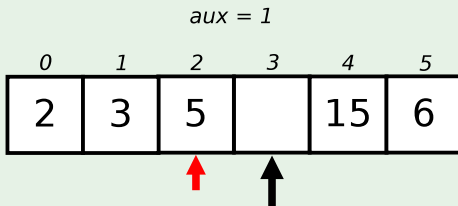
# Insertion-Sort

## Exemplo



# Insertion-Sort

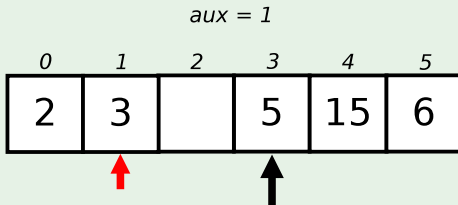
## Exemplo





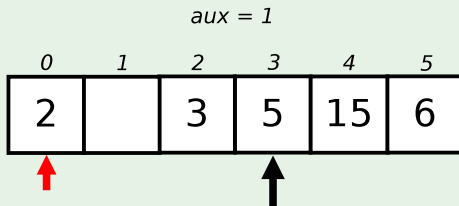
# Insertion-Sort

## Exemplo



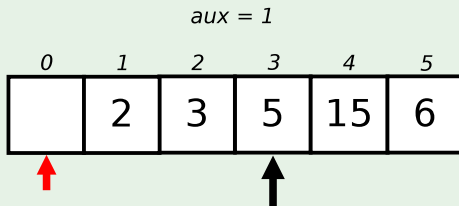
# Insertion-Sort

## Exemplo



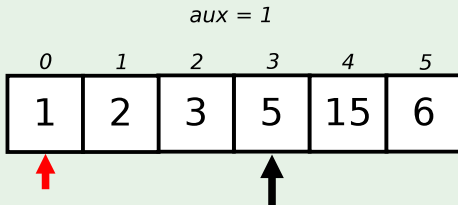
# Insertion-Sort

## Exemplo



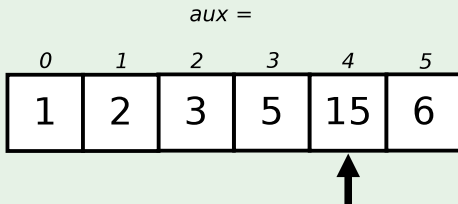
# Insertion-Sort

## Exemplo



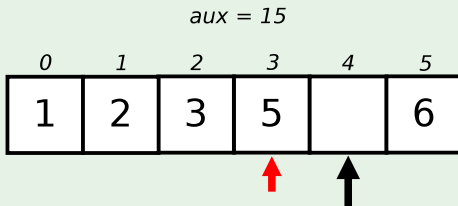
# Insertion-Sort

## Exemplo



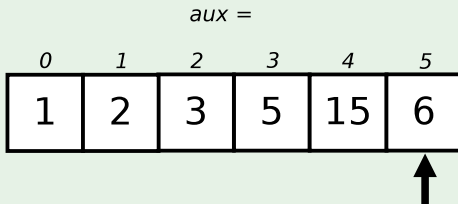
# Insertion-Sort

## Exemplo



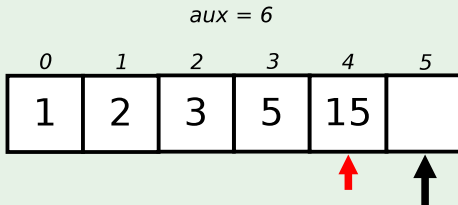
# Insertion-Sort

## Exemplo



# Insertion-Sort

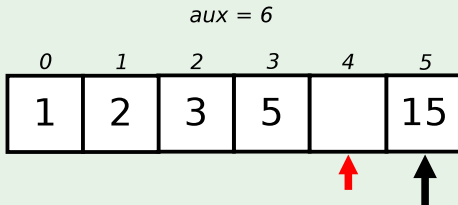
## Exemplo





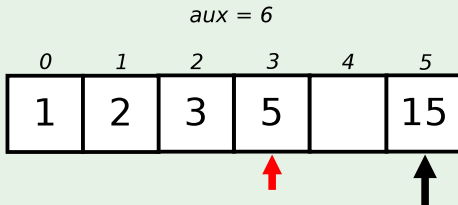
# Insertion-Sort

## Exemplo



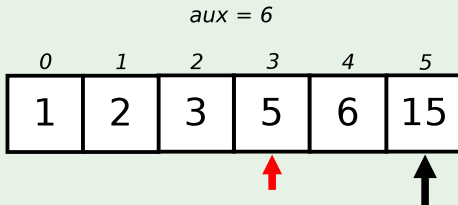
# Insertion-Sort

## Exemplo



# Insertion-Sort

## Exemplo



# Insertion-Sort

## Exemplo

<i>0</i>	<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>
1	2	3	5	6	15



# Insertion-Sort

## Problema

Assumindo um vetor ordenado dado,  $\text{vet}[6]=\{1,2,5,6,15\}$ . Inclua o número **3**, iniciando a verificação na posição  $i$  (nesse caso  $i$  seria a última posição, índice **5**).



# Insertion-Sort

## Solução

```
1 #include <stdio.h>
2
3 int main(){
4     int aux, j, i = 5, vet[6]={1,2,5,6,15};
5     aux = 3;
6     j = i - 1;
7     while((j >= 0) && (vet[j] > aux)){
8         vet[j+1] = vet[j];
9         j--;
10    }
11    vet[j + 1] = aux;
12    for(j = 0; j < 6; j++)
13        printf(" %d",vet[j]);
14    printf("\n");
15 }
```



# Insertion-Sort

## Problema

Utilizando o código anterior, faça o algoritmo **Insertion-Sort** para a ordenação do vetor  $\mathbf{vet}[] = \{5, 3, 2, 1, 15, 6\}$ .



# Insertion-Sort

## Solução

```
1 #include <stdio.h>
2
3 int main(){
4     int aux, j, i, vet[6]={5,3,2,1,15,6};
5
6     for(i = 1; i < 6; i++){
7         aux = vet[i];
8         j = i - 1;
9         while((j >= 0) && (vet[j] > aux)){
10             vet[j+1] = vet[j];
11             j--;
12         }
13         vet[j + 1] = aux;
14     }
15     for(j = 0; j < 6; j++)
16         printf(" %d",vet[j]);
17     printf("\n");
18 }
```



# Pegando o tempo de execução

## Exemplo

```
1 #include <time.h>
2 int main(){
3     clock_t start = clock(); //← antes de qualquer
        comando
4     double time = 0;
5
6     // todo seu codigo
7
8     //antes do return
9     time = ((double)clock()-start)/CLOCKS_PER_SEC);
10    return (0);
11 }
```



# Insertion-Sort

## Problema

Exiba na tela o tempo gasto pelo algoritmo **Insertion-Sort** para a ordenação do vetor  $\text{vet}[] = \{5, 3, 2, 1, 15, 6\}$ .



# Insertion-Sort

## Solução

```
1 #include <stdio.h>
2 #include <time.h>
3 void insertionSort(int vet[], int tam);
4 int main(){
5     clock_t start = clock();
6     int i, tam = 6, vetor [] = {5,3,2,1,15,6};
7     insertionSort(vetor, tam);
8     printf("Tempo: %f\n", ((double)clock()-start)/
9         CLOCKS_PER_SEC);
10 }
11 void insertionSort(int vet[], int tam){
12     int i, j, aux;
13     for(i=1; i<tam; i++){
14         aux = vet[i];
15         j=i-1;
16         while( (j>=0) && (vet[j] > aux)){
17             vet[j+1] = vet[j];
18             j--;
19         }
20         vet[j+1] = aux;
21     }
```

# Questões?

Obrigado!

*Para informação:*

**Página dos Laboratórios (Tarefas):** <http://susy.ic.unicamp.br:9999/mc102ab>

**Página do Curso:** <http://www.lrc.ic.unicamp.br/~geraldoms/mc102>

**E-mail:**

*geraldoms[at]lrc.ic.unicamp.br*

*brhenrique.fischer[at]gmail.com*

