INE5231 Computação Científica I

Prof. A. G. Silva

25 de abril de 2017

Conteúdo programático

- **O computador - [3 horas-aula]**
- **Representação de algoritmos - [3 horas-aula]:**
- **Linguagens de programação estruturadas [3 horas-aula]**
- **Introdução à programação em C [6 horas-aula]**
- **Programas envolvendo processos de repetição e seleção [6 horas-aula]**
- **Variáveis estruturadas unidimensionais homogêneas [9 horas-aula]**
- **Variáveis estruturadas multidimensionais homogêneas [6 horas-aula]**
- **Variáveis estruturadas heterogêneas [6 horas-aula]**
- **Subdivisão de problemas e subprogramação [6 horas-aula]**
- **Programação utilizando uma linguagem de computação técnica numérica [6 horas-aula]**

Curso de C

Vetores

Vetores

Roteiro:

- Declaração de vetores
- Uso correto de vetores
- Vetores com tamanho variável
- Matrizes
- Vetores de caracteres (texto)

Introdução

Tipos de Dados:

- Da linguagem C:
  - Números inteiros: `int`, `long int`, `unsigned int`, ...
  - Números fracionários: `float`, `double`, ...
  - Caracteres: `char`
- Do programador:
  - Vetores, estruturas, enumerações, etc
  - Sinônimos

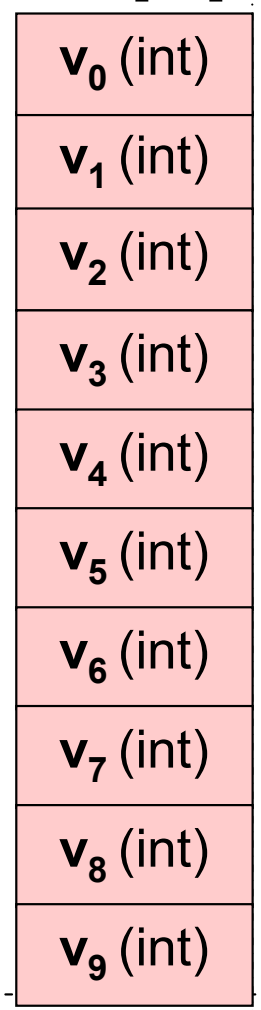
Vetores

Declaração e Uso

Vetores

Conceitos:

- Seqüência de valores
- Todos do mesmo tipo
- Nome único para a variável
- Acesso por índice
- Tamanho fixo
- Numeração de 0 até *tamanho*-1
- Alocados sequencialmente na memória



Declaração

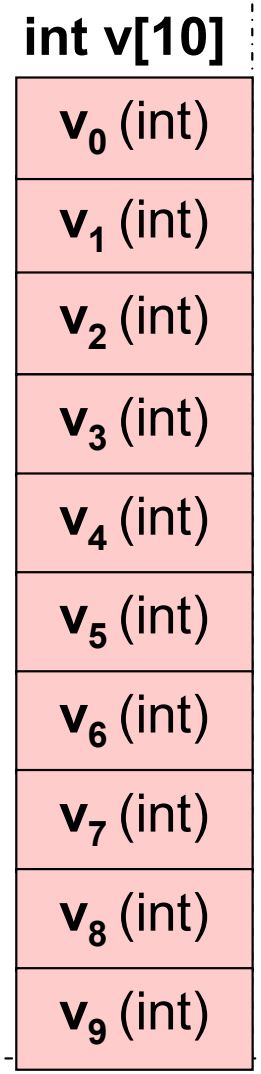
Declaração de vetor:

```
tipo  variavel  [tamanho]
```

- `tipo`: Qualquer tipo da linguagem C
- `variavel`: Nome da variável
- `[tamanho]`: Número de elementos

Exemplo:

```
int vetor[10];
double medidas[100];
```



Acesso

Elementos com índice:

Elementos do vetor:

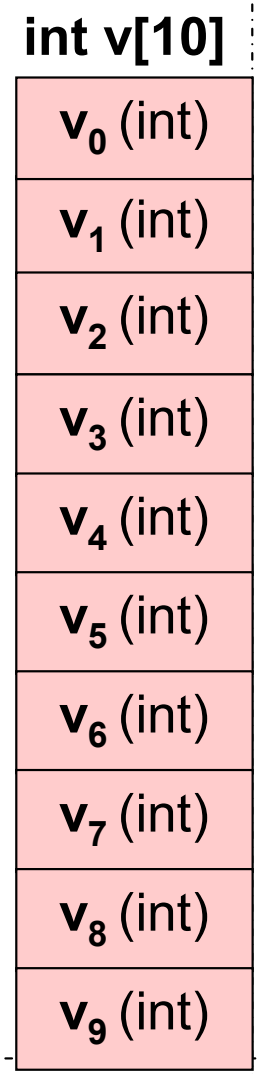
```
vetor[0], vetor[1], vetor[2], ...
```

Atribuição:

```
vetor[indice] = valor;
```

Exemplo:

```
int vetor[10];
vetor[5] = 3;
vetor[0] = vetor[1] + vetor[2];
```



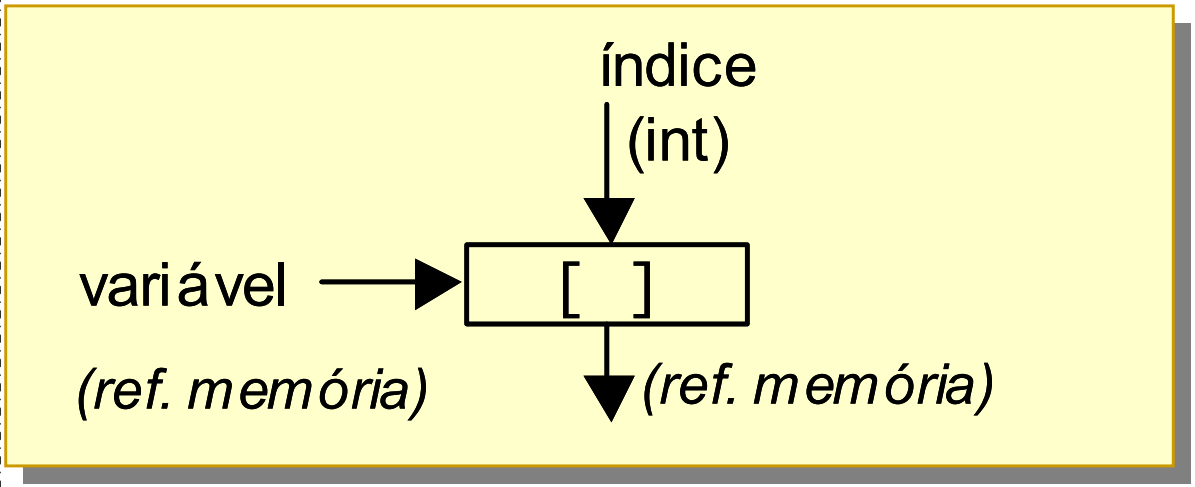
Acesso

Exemplo:

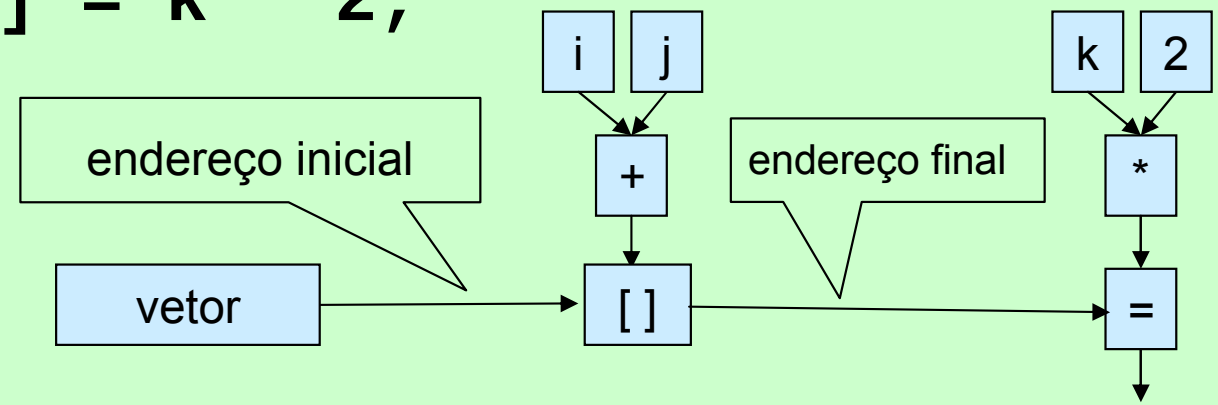
```
int vetor[100];
...
for (indice = 0;
     indice < 100;
     indice++) {

    printf("%d, ", vetor[indice]);

}
```

Acesso

Operador de índice:




```
vetor[i+j] = k * 2;
```



Acesso

Imprimir uma listra de trás para frente:

```
int main(int argc, char *argv[]) {
      int valores[10];
      int indice;
      printf("Escreva 10 números inteiros: ");
      for (indice = 0; indice < 10; indice++) {
            scanf("%d", &valores[indice] );
      }
      printf("Valores em ordem reversa:\n");
      for (indice = 9; indice >= 0; indice--) {
            printf("%d ", valores[indice]);
      }
      return 0;
}
```

Vetores\Reverso01\Reverso01.vcproj

Conteúdo inicial

Declaração com conteúdo inicial:

```
tipo vetor [n] = {elem_0, elem_1, ..., elem_n-1}
```

Tamanho do vetor

Lista de n valores

Exemplo:

```
int impares[5] = {1, 3, 5, 7, 9};
```

Conteúdo inicial

Declaração com conteúdo inicial :

```
tipo vetor [ ] = {elem0, elem1, ..., elemn-1}
```

Tamanho do vetor omitido

Lista de n valores

Exemplo:

```
int impares[] = {1, 3, 5, 7, 9};
```

Vetores

Regras para uso correto

Cuidado com os índices:

Errado:

```
int vetor[10];
...
vetor[-2] = 3;
vetor[20] = 6;
```

Efeitos imprevisíveis!

Correto:

```
vetor[4] = 3;
vetor[7] = 6;
```

Vetores

Atribuir todos os valores:

Errado:

```
int vetor[10];
...
vetor = 0;
```

Correto:

```
int indice;
for (indice = 0; indice < 10; indice++) {
   vetor[indice] = 0;
}
```

Copiar todos os valores:

Errado:

```
int vetorA[10], vetorB[10];
...
vetorA = vetorB;
```

Correto:

```
int indice;
for (indice = 0; indice < 10; indice++) {
   vetorA[indice] = vetorB[indice];
}
```

Vetores

Vetor de tamanho variável

Vetor de tamanho variável

Solução Simples:

```
int valores[100];
int numero_elementos;


int i;
for (i = 0; i < numero_elementos; i++) {
     printf("%d", valores[i]);
}
```

Limite superior para tamanho

Tamanho utilizado

Vetor de tamanho variável

Imprimir uma lista de números de trás para frente:

```
int main(int argc, char *argv[]) {
        int valores[100];
        int numero_valores;
        int i;

        printf("Quantos valores? (no máximo 100) ");
        scanf("%d", &numero_valores);
        if ( (numero_valores < 1)||(numero_valores > 100)) {
                printf(“Quantidade invalida!\n");
                return 1;
        }
...
```

Vetores

Imprimir uma lista de números de trás para frente:

```
...
        printf("Escreva os números: “);
        for (i = 0; i < numero_valores; i++) {
              scanf("%d", &valores[i] );
        }
        printf("Valores em ordem reversa:\n");
        for (i = numero_valores-1; i >= 0; i--) {
              printf("%d ", valores[i]);
        }
        printf(“\n”);
        return 0;
}
```

Vetores\Reverso02\Reverso02.vcproj

Vetores

Declaração de Matrizes

Matrizes

Conceitos:

- *Tabela* de valores
- Acesso por *dois* índices: linha e coluna
- Numeração: 0 até *linhas* – 1; 0 até *colunas* – 1

Outras propriedades:

- Valores do mesmo tipo
- Acesso aos valores através de um único nome de variável
- Dimensões fixas

```
int a[4][10];
```

a_{00}	a_{01}	a_{02}	a_{03}	a_{04}	a_{05}	a_{06}	a_{07}	a_{08}	a_{09}
a_{10}	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}	a_{16}	a_{17}	a_{18}	a_{19}
a_{20}	a_{21}	a_{22}	a_{23}	a_{24}	a_{25}	a_{26}	a_{27}	a_{28}	a_{29}
a_{30}	a_{31}	a_{32}	a_{33}	a_{34}	a_{35}	a_{36}	a_{37}	a_{38}	a_{39}

Matrizes

Declaração:

```
tipo variavel [linhas] [colunas]
```

- **tipo**: Qualquer tipo C
- ***variavel***: Nome da variável
- **[linhas]**: Número de linhas
- **[colunas]**: Número de colunas

Exemplo:

```
int matriz[4][10];
```

```
int a[4][10];
```

a_{00}	a_{01}	a_{02}	a_{03}	a_{04}	a_{05}	a_{06}	a_{07}	a_{08}	a_{09}
a_{10}	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}	a_{16}	a_{17}	a_{18}	a_{19}
a_{20}	a_{21}	a_{22}	a_{23}	a_{24}	a_{25}	a_{26}	a_{27}	a_{28}	a_{29}
a_{30}	a_{31}	a_{32}	a_{33}	a_{34}	a_{35}	a_{36}	a_{37}	a_{38}	a_{39}

Matrizes

Acesso:

Elementos do vetor:

```
vetor[0][0], vetor[0][1], vetor[0][2], ...
vetor[linhas-1][colunas-1]
```

Atribuição:

```
vetor[linha][coluna] = valor;
```

Exemplo:

```
int vetor[6][10];
vetor[5][1] = 3;
vetor[0][2] = vetor[1][0] + vetor[2][3];
```

Matrizes

Acesso:

Exemplo:

```
int lin, col;
int matriz[4][10];
...
for (lin = 0; lin < 4; lin++) {
   for (col = 0; col < 10; col++) {
      printf("%d ", matriz[lin][col];
   }
   printf("\n");
}
```

Vetores multidimensionais

Vetor com *n* dimensões:

```
tipo variavel [dim_1] [dim_2] ... [dim_N] ;
```

- `tipo`: Qualquer tipo C
- *`variavel`*: Nome da variável
- $[dim_1]$: Dimensão 1
- $[dim_2]$: Dimensão 2
- $[dim_N]$: Dimensão *N*

Vetores

Vetor de caracteres

(Texto)

Texto

Conceitos:

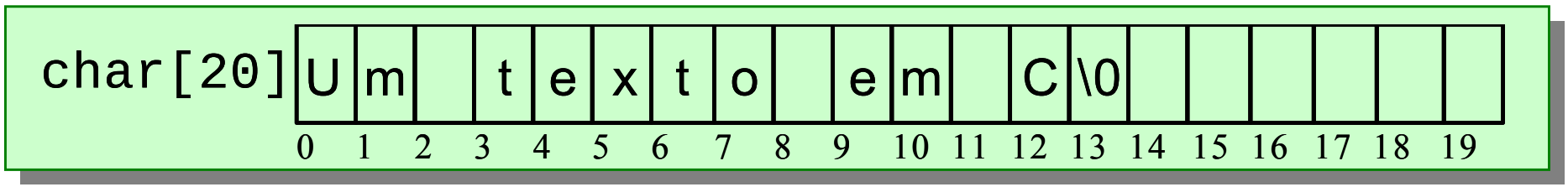
- Caracteres individuais:
  - `'a'`, `'A'`, `'f'`, `'4'`, `'.'` ⟵ aspas simples!
  - Representam apenas um símbolo, letra ou dígito
- Texto:
  - Seqüência de caracteres
  - `"Algoritmos e Programação"`

aspas duplas!

Texto

Armazenamento:

- Vetor de caracteres
- Cada caractere do texto $\Rightarrow$ um elemento do vetor
- Último caractere: nulo (‘\0’)
- Tamanho mínimo do vetor: comprimento do texto + 1



Texto

Declaração:

```
char  variavel [tamanho];
```

Tamanho máximo do texto menos um

Exemplo:

```
char texto[50];
```

Texto

Declaração:

```
char  variavel [tamanho]  = "texto";
```

Exemplo: `char texto[50] = "João da Silva";`

```
char  variavel [] = "texto";
```

Exemplo: `char texto[] = “João da Silva“;`

Texto

Ler e imprimir:

Impressão:

```
char texto[] = “Um texto em C”;
printf(“A variável: %s”, texto);
```

Leitura:

```
char texto[20];
printf(“Escreva uma palavra: ”);
scanf(“%s”, texto);
```

OBS:

- Leitura por palavra
- Não usa &
- Cuidado com tamanho da variável

Texto

Atribuir à uma variável tipo texto:

Errado:

```
char texto[100];
...
texto = “João da Silva”;
```

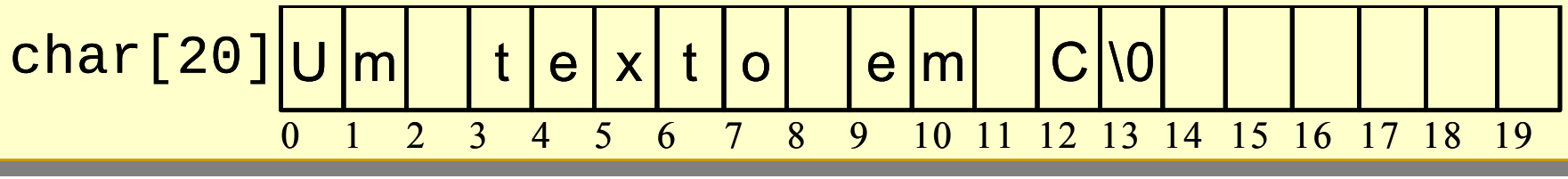
Correto:

```
#include <string.h>
...
strcpy(texto, “João da Silva”);
```

Texto

Acessar caracteres:

```
char texto[20] = “Um texto em C”;
texto[12] = ‘B’;
printf(“%s”, texto);
```




```
Um texto em B
```

Vetores

Estatistica [calcula média e desvio padrão]
Palind [testa palindromos]
Primos [pegar primos, método da peneira]
Strings1 [teste com cadeias e funções sobre cadeias]
Concat [concatena cadeias]
Anagrama [testa anagramas]
SomaMat [soma de matrizes]
QuadradMag [testa quadrado mágico]

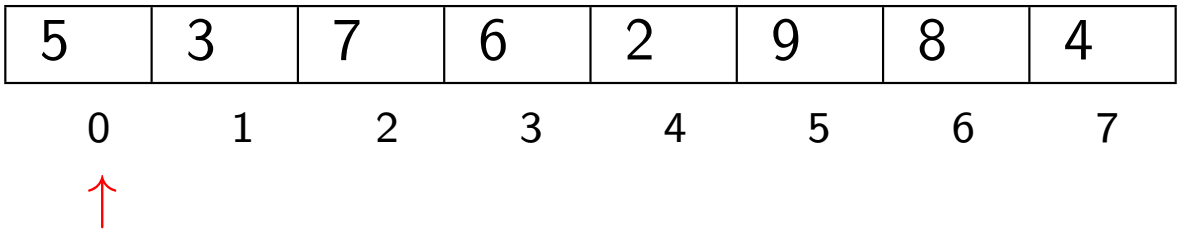
Processamentos de vetor

- Média e desvio padrão (exercício)
- Máximo (mínimo)
- Argmax
- Ordenação
  - Bolha
  - Seleção
  - Inserção
- Métodos de busca – busca binária

Algoritmos em vetor – valor máximo (I)

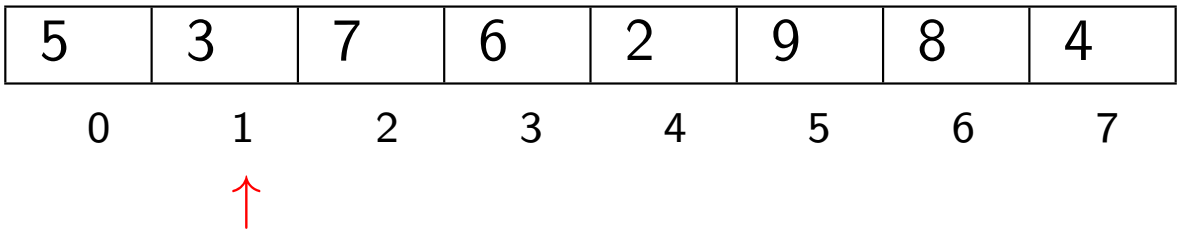
- Tomar o primeiro elemento (posição `i=0`) como o maior, guardando seu valor em uma variável `máximo`
- Verificar se o valor do elemento do vetor da posição seguinte (`i=i+1`) é maior que `máximo`
  - Se for, substituir `máximo` pelo valor do elemento desta posição
  - Se não for, verificar o elemento da posição seguinte (`i=i+1`)
  - Repetir até que não haja mais elemento a verificar (após atingir o comprimento total do vetor)
- A variável `máximo` conterá o maior valor do vetor

Algoritmos em vetor – valor máximo (exemplo)



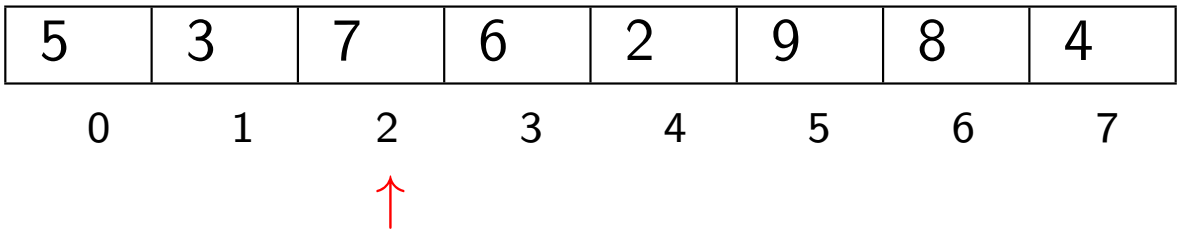
- i = 0
- maximo = v[0] = 5

Algoritmos em vetor – valor máximo (exemplo)



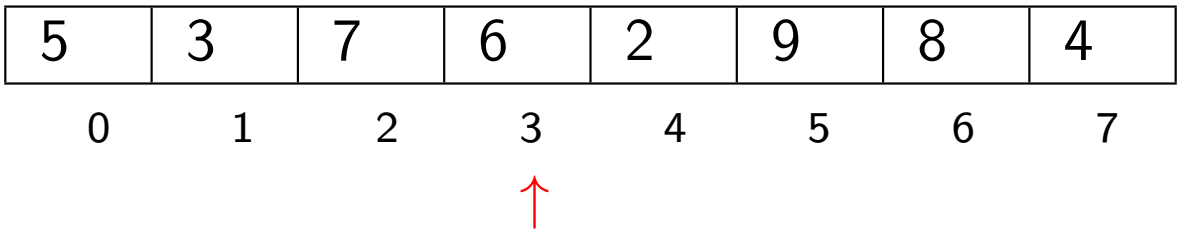
- i $= 1$
- maximo $= 5$
- v[1] $>$ maximo ?
  - *Não (3 $<$ 5): nada a fazer*

Algoritmos em vetor – valor máximo (exemplo)



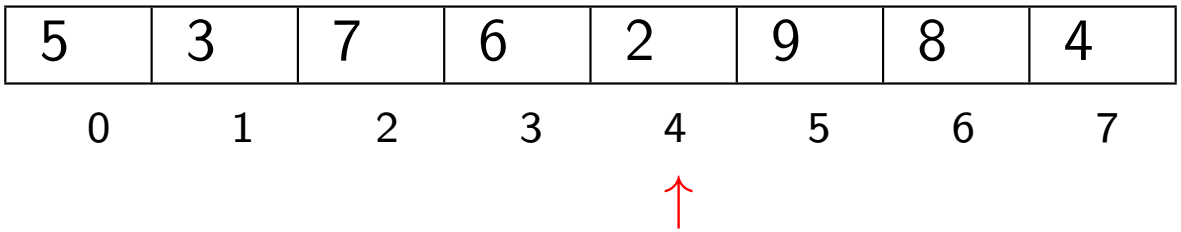
- i $= 2$
- maximo $= 5$
- v[2] $>$ maximo ?
  - *Sim (7 $>$ 5):* maximo $= 7$

Algoritmos em vetor – valor máximo (exemplo)



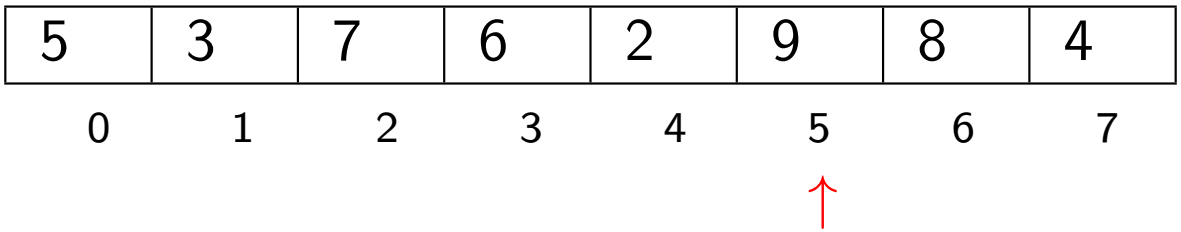
- i $= 3$
- maximo $= 7$
- v[3] $>$ maximo ?
  - *Não ($6 < 7$): nada a fazer*

Algoritmos em vetor – valor máximo (exemplo)



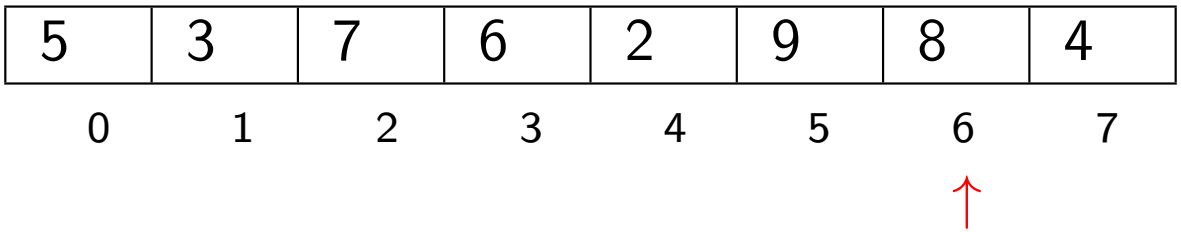
- i $= 4$
- maximo $= 7$
- v[4] $>$ maximo ?
  - *Não (2 $<$ 7): nada a fazer*

Algoritmos em vetor – valor máximo (exemplo)



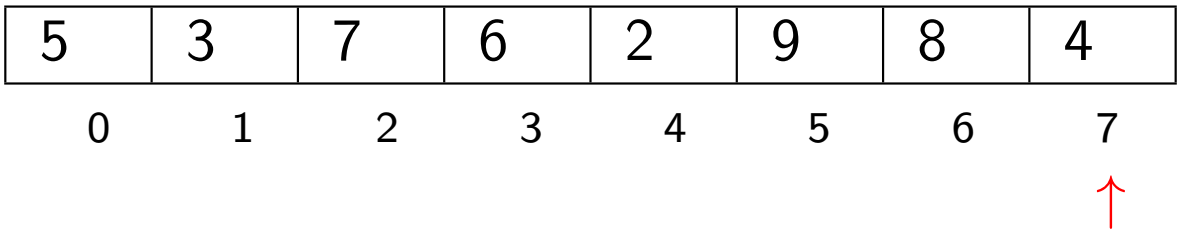
- i $= 5$
- maximo $= 7$
- v[5] $>$ maximo ?
  - *Sim (9 $>$ 7):* maximo $= 9$

Algoritmos em vetor – valor máximo (exemplo)



- i $= 6$
- maximo $= 9$
- v[6] $>$ maximo ?
  - *Não ($8 < 9$): nada a fazer*

Algoritmos em vetor – valor máximo (exemplo)



- i = 7
- maximo = 9
- v[7] > maximo ?
  - *Não (4 < 9): nada a fazer*

Algoritmos em vetor – valor máximo (III)

- Implementação

```
int v[];

/*

...

*/

    int maximo = v[0];
    for (i=1; i<N; i++) {
        if (v[i] > maximo) {
            maximo = v[i];
        }
    }
```

Métodos simples de ordenação

- Tempo de execução proporcional a n^2, ou seja $\mathcal{O}(n^2)$, para n valores
- Alguns algoritmos:
  - Bolha e bolha melhorado (bubble sort)
  - Seleção (selection sort)
  - Inserção (insertion sort)

Método da bolha – *bubble sort*

```
//dado um vetor v ...

    int aux;
    for (int i=0; i<N; i++) {
        for (int j=0; j<N-i-1; j++) {
            if (v[j] > v[j+1]) {
                aux = v[j];
                v[j] = v[j+1];
                v[j+1] = aux;
            }
        }
    }
```

Método da bolha melhorado – *bubble sort*

```
//dado um vetor v ...

    int aux;
    int TROCA;
    for (i=0; i<N; i++) {
        TROCA = 0;
        for (int j=0; j<N-i-1; j++) {
            if (v[j] > v[j+1]) {
                aux = v[j];
                v[j] = v[j+1];
                v[j+1] = aux;
                TROCA = 1;
            }
        }
        if ( ! TROCA )
            break;
    }
```

Método da bolha – *bubble sort*

- Normal

  20 30 28 31 24 60 45 50
  20 28 30 24 31 45 50 60
  20 28 24 30 31 45 50 60
  20 24 28 30 31 45 50 60
  20 24 28 30 31 45 50 60
  20 24 28 30 31 45 50 60
  20 24 28 30 31 45 50 60
  20 24 28 30 31 45 50 60
  20 24 28 30 31 45 50 60

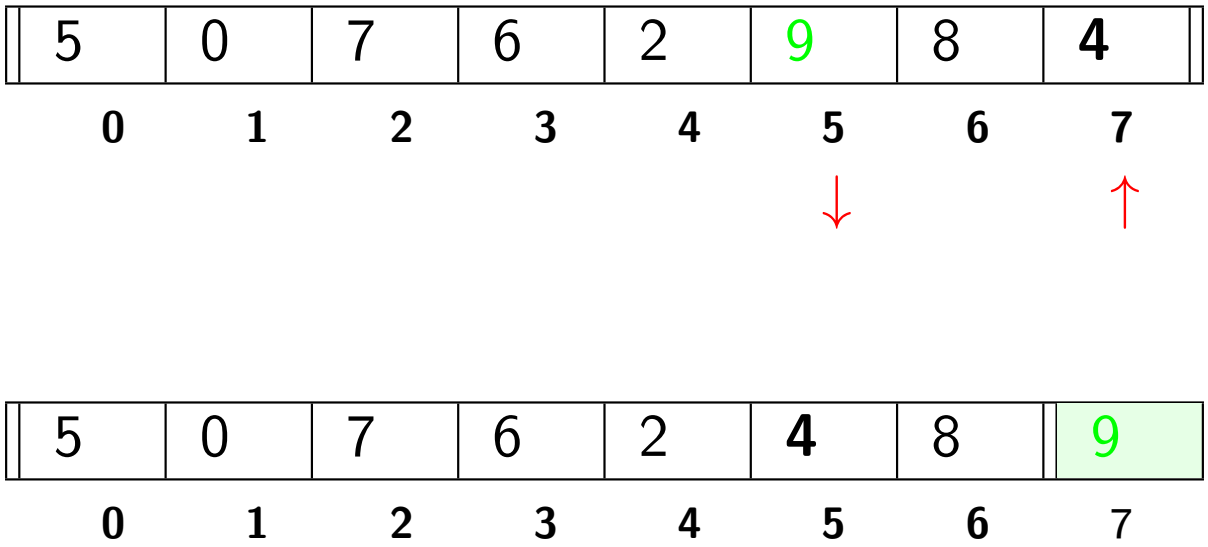
- Melhorado

  20 30 28 31 24 60 45 50
  20 28 30 24 31 45 50 60
  20 28 24 30 31 45 50 60
  20 24 28 30 31 45 50 60
  20 24 28 30 31 45 50 60

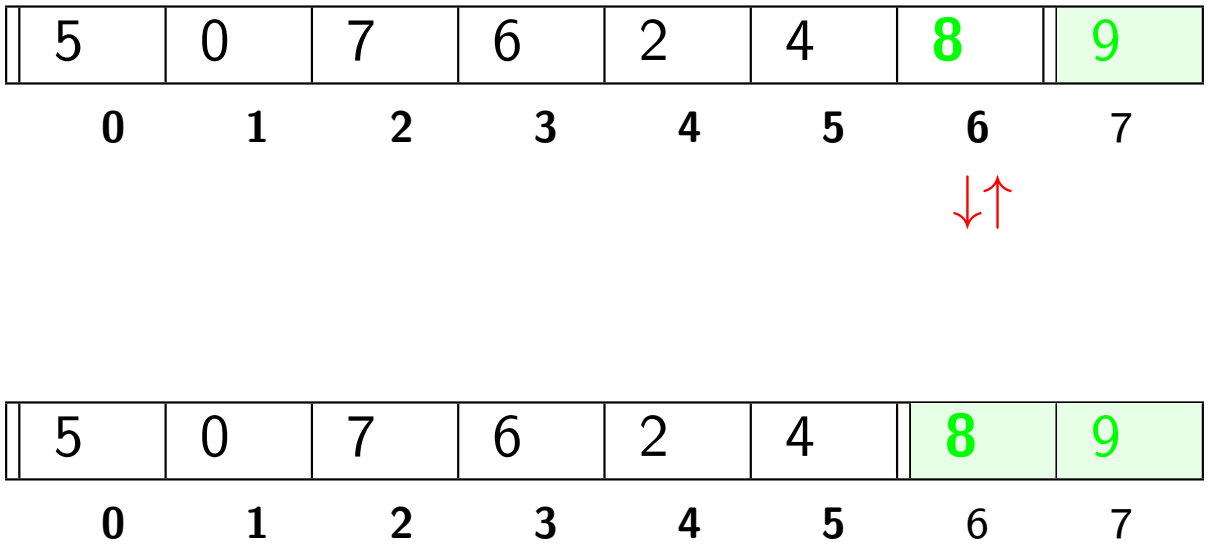
Método da seleção – *selection sort*

1. Partição vai de 0 até k (onde k é o tamanho do vetor menos um)
2. Trocar o maior elemento do vetor (máximo) com a última posição da partição
3. $k = k - 1$, voltar ao passo 1

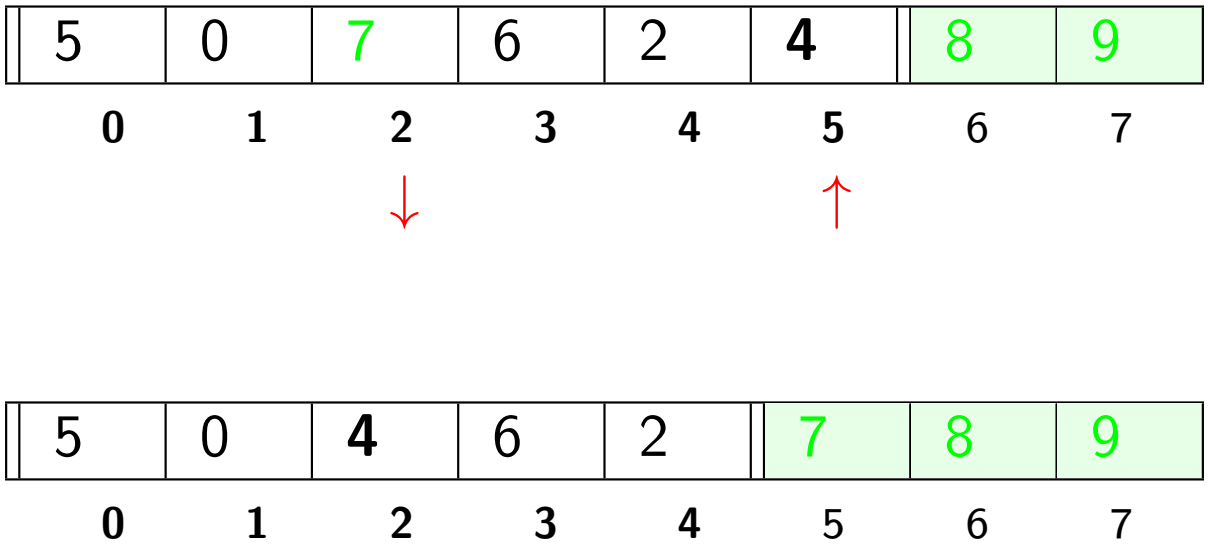
Método da seleção – *selection sort*



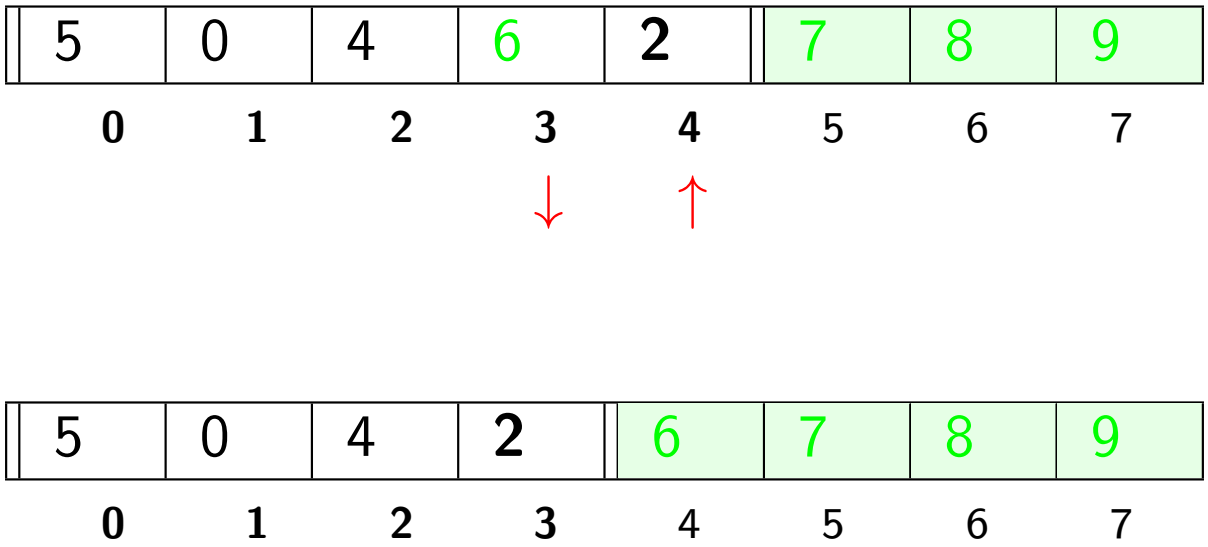
Método da seleção – *selection sort*



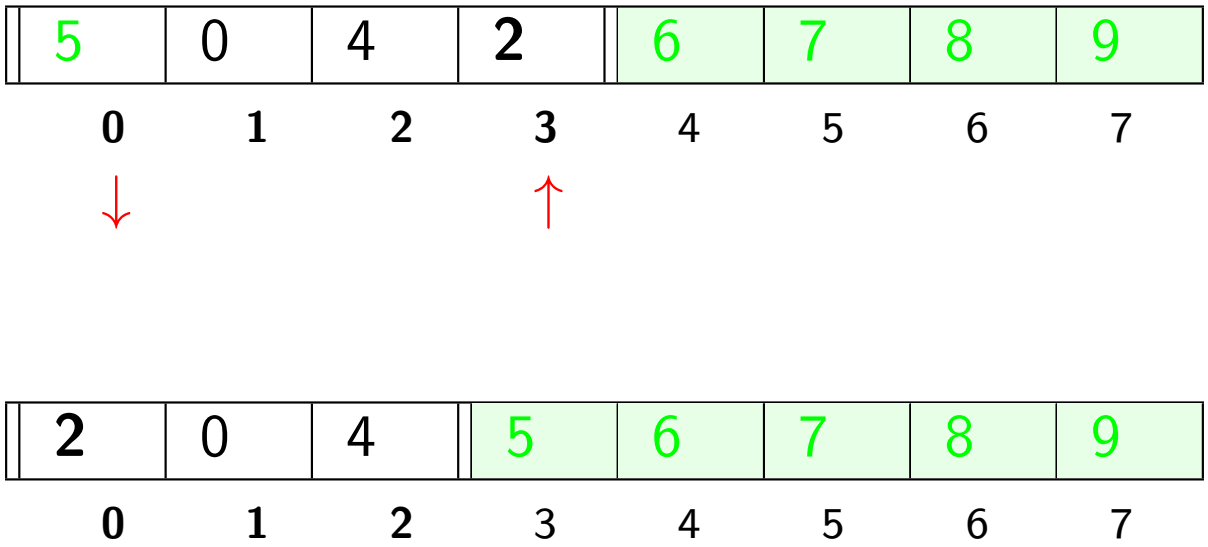
Método da seleção – *selection sort*



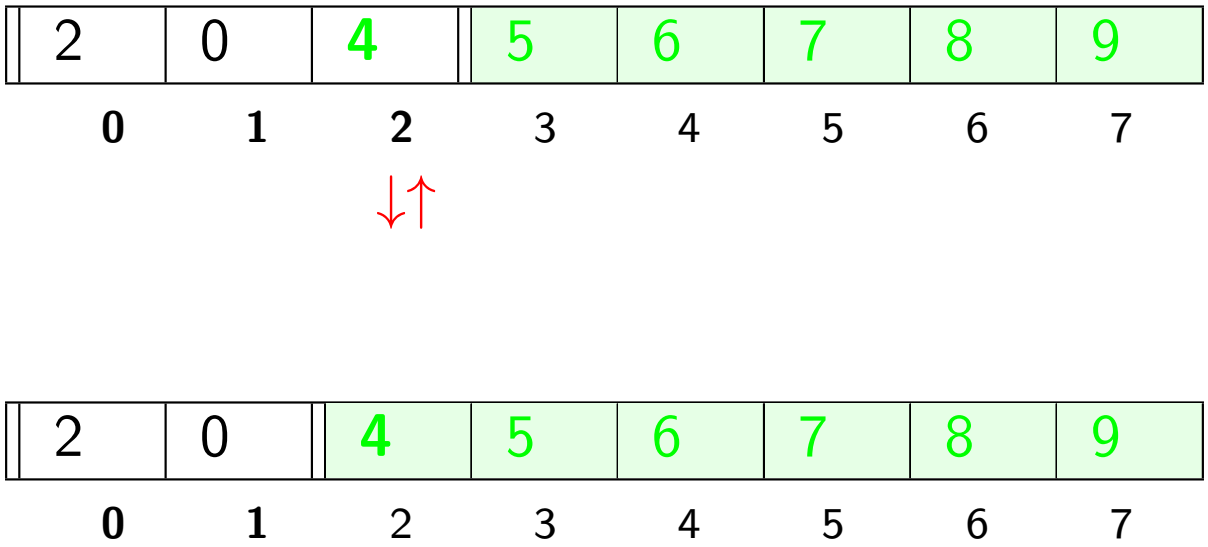
Método da seleção – *selection sort*



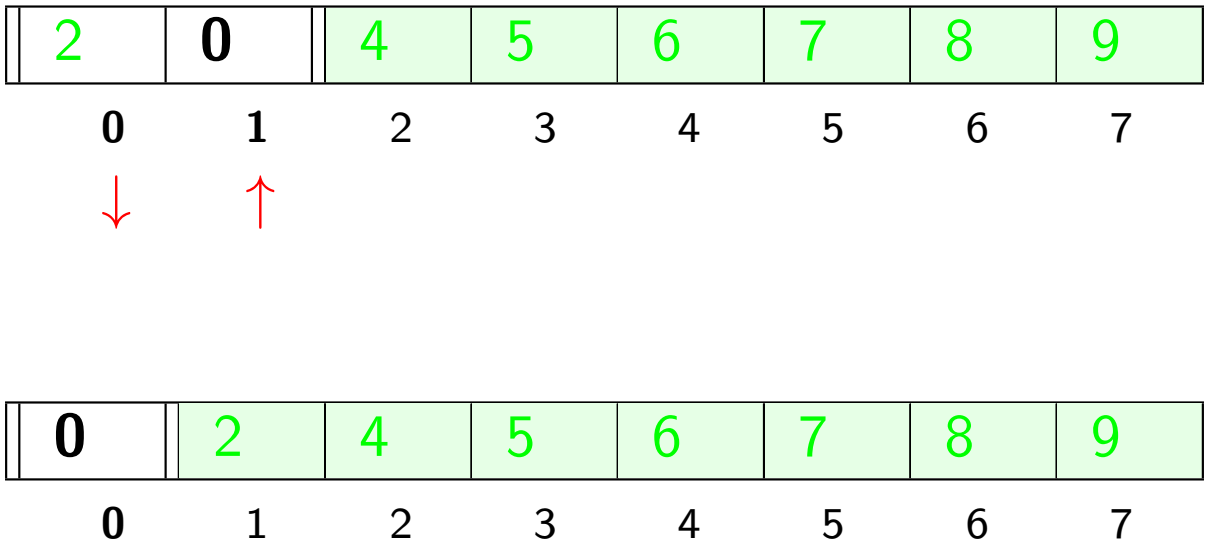
Método da seleção – *selection sort*



Método da seleção – *selection sort*



Método da seleção – *selection sort*



Método da seleção – *selection sort*

```
public void selecao() { //selection sort
    double aux;
    int posMaior;
    for (int i=0; i<v.length; i++) {
        posMaior = 0;
        for (int j=1; j<v.length-i; j++) {
            if (v[j] > v[posMaior])
                posMaior = j;
        }
        aux = v[posMaior];
        v[posMaior] = v[v.length-i-1];
        v[v.length-i-1] = aux;
    }
}
```

Método da seleção – *selection sort*

```
[20  30  28  31  24  60  45  50]
                     ^       ^

[20  30  28  31  24  50  45] 60
                     ^   ^

[20  30  28  31  24  45] 50  60
                     ^^

[20  30  28  31  24] 45  50  60
                 ^   ^

[20  30  28  24] 31  45  50  60
      ^      ^

[20  24  28] 30  31  45  50  60
         ^^

[20  24] 28  30  31  45  50  60
     ^^

[20] 24  28  30  31  45  50  60
```

Método da inserção – *insertion sort*

```
public void insercao() { //insertion sort
     double aux;
     int i, j;
     for (i=1; i<v.length; i++) {
          aux = v[i];
          j = i - 1;
          while (j >= 0 && v[j] > aux) {
               v[j+1] = v[j];
               j--;
          }
          v[j+1] = aux;
     }
}
```

Método da inserção – *insertion sort*

```
[20]  30<  28   31   24   60   45   50
   ^
[20   30]  28<  31   24   60   45   50
    ^
[20   28   30]  31<  24   60   45   50
               ^
[20   28   30   31]  24<  60   45   50
    ^
[20   24   28   30   31]  60<  45   50
                     ^
[20   24   28   30   31   60]  45<  50
                          ^
[20   24   28   30   31   45   60]  50<
                               ^
[20   24   28   30   31   45   50   60]
```

Exercício

- Para cada método de ordenação, calcule:
  - Número de comparações realizadas
  - Número de trocas efetuadas

Métodos de busca

- Normalmente utiliza-se um valor numérico como chave primária (identificador exclusivo) de busca. Por exemplo, matrícula do aluno

- A busca por um determinado valor pode ser eficientemente implementada em um vetor ordenado por meio de:

  - Busca binária: verifica-se o elemento do meio da partição; se valor nesta posição é igual, encontrou (finaliza); se maior, a busca é repetida na primeira metade; se menor, é repetida na segunda metade

  - Busca por interpolação linear: verifica-se o elemento dado pela interpolação linear dos dois valores nas posições extremas da partição; se valor nesta posição é igual, encontrou (finaliza); se maior, repete-se para a primeira parte; caso contrário, repete-se para a segunda parte

Exercício

- Dado um vetor qualquer, ordene-o por qualquer método e implemente a busca binária.

Referências

- Notas do Prof. Arnaldo V. Moura e Daniel F. Ferber – Curso C – IC/Unicamp