



# Fundamentos de Programação

Vetores Numéricos

# Programa sem vetores

Programa que lê as notas de 4 alunos e calcula a sua média.

```
static void Main(string[] args)
{
    int i;
    double nota, media, soma = 0;
    for (i = 0; i < 4; i++)
    {
        Console.Write("Digite uma nota:");
        nota = Convert.ToDouble(Console.ReadLine());
        soma += nota;
    }
    media = soma / 4;
    Console.Write("Media = {0}", media);
}
```

# Programa sem vetores

Programa que lê as notas de 4 alunos e calcula a sua média.

|  |
|--|
| Assuma que seja<br>necessário imprimir,<br>no final do programa<br>o número de notas<br>acima da média<br>calculada. |
|--|

```
static void Main(string[] args)
{
    int i;
    double nota, media, soma = 0;
    for (i = 0; i < 4; i++)
    {
        Console.Write("Digite uma nota:");
        nota = Convert.ToDouble(Console.ReadLine());
        soma += nota;
    }
    media = soma / 4;
    Console.Write("Media = {0}", media);
}
```

# Programa sem vetores

Programa que lê as notas de 4 alunos e calcula a sua média.

```
static void Main(string[] args)
{
    int i;
    double nota, media, soma = 0;
    for (i = 0; i < 4; i++)
    {
        Console.Write("Digite uma nota:");
        nota = Convert.ToDouble(Console.ReadLine());
        soma += nota;
    }
    media = soma / 4;
    Console.Write("Media = {0}", media);
}
```

Assuma que seja necessário imprimir, no final do programa, o número de notas acima da média calculada.

Que modificações são necessárias no programa?

# Programa sem vetores

Programa que lê notas de 4 alunos, calcula sua média e imprime o número de notas acima da média.

```
static void Main(string[] args)
{
    int i, cont = 0;
    double nota, media, soma = 0;
    for (i = 0; i < 4; i++)
    {
        Console.Write("Digite uma nota:");
        nota = Convert.ToDouble(Console.ReadLine());
        soma += nota;
    }
    media = soma / 4;
    Console.Write("Media = {0}", media);
}
```

É necessário inicializar um contador e inicializá-lo.

# Programa sem vetores

Programa que lê notas de 4 alunos, calcula sua média e imprime o número de notas acima da média.

```
static void Main(string[] args)
{
    int i, cont = 0;
    double nota, media, soma = 0;
    for (i = 0; i < 4; i++)
    {
        Console.Write("Digite uma nota:");
        nota = Convert.ToDouble(Console.ReadLine());
        soma += nota;
    }
    media = soma / 4;
    Console.Write("Media = {0}", media);
}
```

É necessário contar o número de notas acima da média.

***Isso só pode ser feito após o cálculo da média!***

# Programa sem vetores

Programa que lê notas de 4 alunos, calcula sua média e imprime o número de notas acima da média.

```
static void Main(string[] args)
{
    int i, cont = 0;
    double nota, media, soma = 0;
    for (i = 0; i < 4; i++)
    {
        Console.Write("Digite uma nota: ");
        nota = Convert.ToDouble(Console.ReadLine());
        soma += nota;
    }
    media = soma / 4;
    Console.WriteLine("Média = {0}", media);
}
```

Duas opções:

1) Ler o valor de cada

nota outra vez

2) Ler apenas um  
cada valor,  
armazenando  
nota em uma  
distinta

z

a vez

o cada  
variável

# Programa sem vetores

Programa que lê notas de 4 alunos, calcula sua média e imprime o número de notas acima da média.

```
// Opcao 1: le duas vezes cada valor
static void Main(string[] args)
{
    int i, cont=0;
    double nota, media, soma = 0;
    for (i = 0; i < 4; i++){
        Console.Write("Digite uma nota:");
        nota = Convert.ToDouble(Console.ReadLine());
        soma += nota;
    }
    media = soma / 4;
    Console.Write("Digite tudo de novo!");
    for (i = 0; i < 4; i++){
        Console.Write("Digite uma nota:");
        nota = Convert.ToDouble(Console.ReadLine());
        if( nota > media )
            cont++;
    }
    Console.Write("Media = {0}", media);
    Console.Write("{0} notas acima", cont);
}
```

```
//Opcao 2: uma variavel por nota
static void Main(string[] args)
{
    int i, cont=0;
    double n1, n2, n3, n4;
    double media, soma = 0;

    Console.Write("Digite as notas:");
    n1 = Convert.ToDouble(Console.ReadLine());
    n2 = Convert.ToDouble(Console.ReadLine());
    n3 = Convert.ToDouble(Console.ReadLine());
    n4 = Convert.ToDouble(Console.ReadLine());

    media = (n1 + n2 + n3 + n4) / 4;
    Console.Write("Media = {0}", media);

    if( n1 > media ) cont++;
    if( n2 > media ) cont++;
    if( n3 > media ) cont++;
    if( n4 > media ) cont++;

    Console.Write("{0} notas acima", cont);
}
```



# Programa sem vetores

Programa que lê notas de 4 alunos, calcula sua média e imprime o número de notas acima da média.

```
// Opcao 1: le duas vezes cada valor
static void Main(string[] args)
{
    int i, cont=0;
    double nota, media, soma = 0;
    for (i = 0; i < 4; i++){
        Console.WriteLine("Digite uma nota:");
        nota = Convert.ToDouble(Console.ReadLine());
        soma += nota;
    }
    media = soma / 4;
    Console.WriteLine("Digite tudo de novo!");
    for (i = 0; i < 4; i++){
        Console.WriteLine("Digite uma nota:");
        nota = Convert.ToDouble(Console.ReadLine());
        if( nota > media )
            cont++;
    }
    Console.WriteLine("Media = {0}", media);
    Console.WriteLine("{0} notas acima", cont);
}
```

```
//Opcao 2: uma variavel por nota
static void Main(string[] args)
{
    int i, cont=0;
    double n1, n2, n3, n4;
    double media, soma = 0;

    Console.WriteLine("Digite as notas:");
    n1 = Convert.ToDouble(Console.ReadLine());
    n2 = Convert.ToDouble(Console.ReadLine());
    n3 = Convert.ToDouble(Console.ReadLine());
    n4 = Convert.ToDouble(Console.ReadLine());

    media = (n1 + n2 + n3 + n4) / 4;
    Console.WriteLine("Media = {0}", media);

    if( n1 > media ) cont++;
    if( n2 > media ) cont++;
    if( n3 > media ) cont++;
    if( n4 > media ) cont++;
}
```

*Na opção 1, o usuário vai ter o trabalho de redigitar os valores.*

## Programa sem vetores

Programa que lê notas de 4 alunos, calcula sua média e imprime o número de notas acima da média.

```
// Opcao 1: le duas vezes cada valor
static void Main(string[] args)
{
    int i, cont=0;
    double nota, media, soma = 0;
    for (i = 0; i < 4; i++){
        Console.Write("Digite uma nota:");
        nota = Convert.ToDouble(Console.ReadLine());
        soma += nota;
    }
    media = soma / 4;
    Console.Write("Digite tudo de novo!");
    for (i = 0; i < 4; i++){
        Console.Write("Digite uma nota:");
        nota = Convert.ToDouble(Console.ReadLine());
        if( nota > media )
            cont++;
    }
}
```

|       |  |
|-------|--|
| Conso | <i>Na opção 2, o programador precisa repetir os comandos para cada variável.</i> |
| Conso |  |
|       |  |

```
//Opcao 2: uma variavel por nota
static void Main(string[] args)
{
    int i, cont=0;
    double n1, n2, n3, n4;
    double media, soma = 0;

    Console.Write("Digite as notas:");
    n1 = Convert.ToDouble(Console.ReadLine);
    n2 = Convert.ToDouble(Console.ReadLine);
    n3 = Convert.ToDouble(Console.ReadLine);
    n4 = Convert.ToDouble(Console.ReadLine);

    media = (n1 + n2 + n3 + n4) / 4;
    Console.Write("Media = {0}", media);

    if( n1 > media ) cont++;
    if( n2 > media ) cont++;
    if( n3 > media ) cont++;
    if( n4 > media ) cont++;

    Console.Write("{0} notas acima", cont)
}
```

# Programa sem vetores

Programa que lê notas de 4 alunos, calcula sua média e imprime o número de notas acima da média.

```
// Opcao 1: le duas vezes cada valor
static void Main(string[] args)
{
    int i, cont=0;
    double nota, media, soma = 0;
    for (i = 0; i < 4; i++){
        Console.Write("Digite uma nota:");
        nota = Convert.ToDouble(Console.ReadLine());
        soma += nota;
    }
    media = soma / 4;
    Console.Write("Digite tudo de novo!");
    for (i = 0; i < 4; i++){
        Console.Write("Digite uma nota:");
        nota = Convert.ToDouble(Console.ReadLine());
        if( nota > media )
            cont++;
    }
    Console.Write("Media = {0}", media);
    Console.Wr
```

```
//Opcao 2: uma variavel por nota
static void Main(string[] args)
{
    int i, cont=0;
    double n1, n2, n3, n4;
    double media, soma = 0;

    Console.Write("Digite as notas:");
    n1 = Convert.ToDouble(Console.ReadLine());
    n2 = Convert.ToDouble(Console.ReadLine());
    n3 = Convert.ToDouble(Console.ReadLine());
    n4 = Convert.ToDouble(Console.ReadLine());

    media = (n1 + n2 + n3 + n4) / 4;
    Console.Write("Media = {0}", media);

    if( n1 > media ) cont++;
    if( n2 > media ) cont++;
    if( n3 > media ) cont++;
    if( n4 > media ) cont++;

    Console.Write("Número de notas acima", cont);
}
```

*E se o número de notas aumentar?*

# Programa sem vetores

```
// Opcao 1: le duas vezes cada valor
static void Main(string[] args)
```

```
{
    int i, cont=0;
    double nota, media;
    for (i = 0; i < 4; i++)
    {
        Console.WriteLine("Digite uma nota:");
        nota = Convert.ToDouble(Console.ReadLine());
        soma += nota;
    }
    media = soma / 4;
    Console.WriteLine("Media = {0}", media);
    Console.WriteLine("{0} notas acima", cont);
}
```

**Vetores permitem  
resolver esse  
problema de forma  
conveniente.**

```
//Opcao 2: uma variavel por nota
static void Main(string[] args)
```

```
{
    int i, cont=0;
    double n1, n2, n3, n4;
    double soma = 0;

    Console.WriteLine("Digite as notas:");
    n1 = Convert.ToDouble(Console.ReadLine());
    n2 = Convert.ToDouble(Console.ReadLine());
    n3 = Convert.ToDouble(Console.ReadLine());
    n4 = Convert.ToDouble(Console.ReadLine());

    media = (n1 + n2 + n3 + n4) / 4;
    Console.WriteLine("Media = {0}", media);

    if ( n1 > media ) cont++;
    if ( n2 > media ) cont++;
    if ( n3 > media ) cont++;
    if ( n4 > media ) cont++;

    Console.WriteLine("{0} notas acima", cont);
}
```

## *Um vetor...*

- é uma estrutura que permite armazenar uma sequência de dados de um mesmo tipo.
- permite que dados sejam armazenados e estruturados de forma simples.
- permite que cada um dos dados armazenados seja acessado diretamente.

# Declarando um vetor

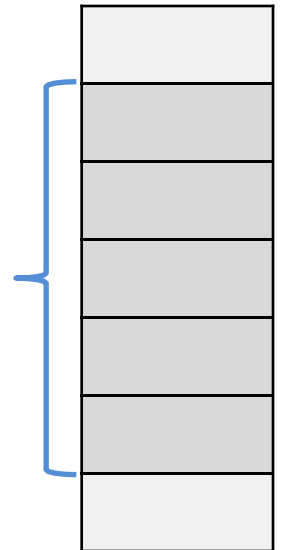
- Ao criar um vetor, precisamos informar o tamanho deste, isto é, precisamos informar quantos valores serão armazenados.
- Ao saber o número de valores, o vetor separa o espaço necessário na memória.
- A sintaxe para criação de um vetor é:

```
tipo [] nomeDoVetor = new tipo [TAMANHO];
```

- Exemplo:

```
int [] dados = new int [5];
```

dados



# *Declarando um vetor*

- Ao reservar a memória para o programa, antes de iniciar a execução, o programa precisa saber quanto espaço de memória precisa ser separado.
- O tamanho deve ser especificado através de um valor constante.

```
static void Main(string[] args)
{
    int[] matricula = new int[50];
    double[] notaP1 = new double[50];
    double[] notaP2 = new double[50];
    double[] mediaProvas = new double[3];
    ...
}
```

# *Declarando um vetor*

```
const int NUM_ALUNOS = 50;

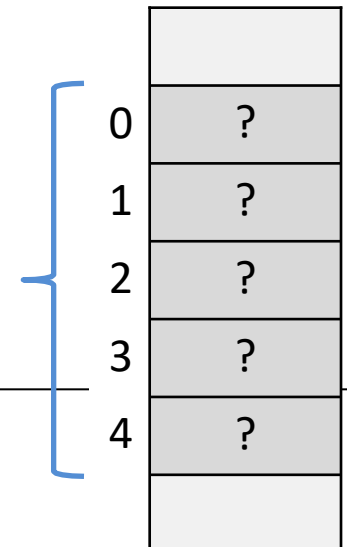
static void Main(string[] args)
{
    int[] matricula = new int[NUM_ALUNOS];
    double[] notaP1 = new double[NUM_ALUNOS];
    double[] notaP2 = new double[NUM_ALUNOS];
    double[] mediaProvas = new double[3];
    ...
}
```



# Acessando elementos

- Ao acessar um elemento de um vetor, é necessário especificar a posição do elemento (ou índice do elemento).
- Em um vetor com N elementos, os índices são numerados de 0 a N-1.

```
int[] dados = new int[5];  
dados[0] = Convert.ToInt32(Console.ReadLine());  
dados[1] = dados[0];  
dados[2] = dados[1]*2;  
dados[3] = dados[2]-1;  
dados[4] = dados[3]/3;  
Console.Write("{0}", dados[4]);
```

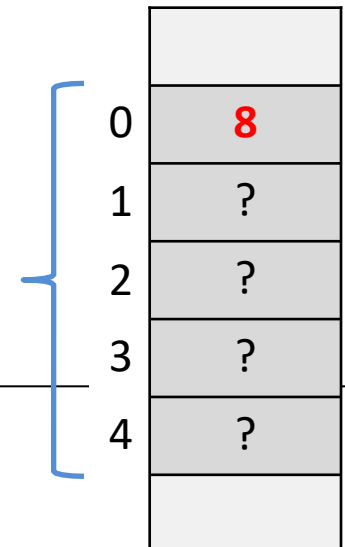


# Acessando elementos

- Ao acessar um elemento de um vetor, é necessário especificar a posição do elemento (ou índice do elemento).
- Em um vetor com N elementos, os índices são numerados de 0 a N-1.

É possível armazenar valores fornecidos através do teclado.

```
int[] dados = new int[5];  
dados[0] = Convert.ToInt32(Console.ReadLine());  
dados[1] = dados[0];  
dados[2] = dados[1]*2;  
dados[3] = dados[2]-1;  
dados[4] = dados[3]/3;  
Console.WriteLine("{0}", dados[4]);
```



|   |   |
|---|---|
|   |   |
| 0 | 8 |
| 1 | ? |
| 2 | ? |
| 3 | ? |
| 4 | ? |
|   |   |

# Acessando elementos

- Ao acessar um elemento de um vetor, é necessário especificar a posição do elemento (ou índice do elemento).
- Em um vetor com N elementos, os índices são numerados de 0 a N-1.

É possível fazer atribuição diretamente a uma posição.

```
int[] dados = new int[5];  
dados[0] = Convert.ToInt32(Console.ReadLine());  
dados[1] = dados[0];  
dados[2] = dados[1]*2;  
dados[3] = dados[2]-1;  
dados[4] = dados[3]/3;  
Console.WriteLine("{0}", dados[4]);
```

|   |    |
|---|----|
|   |    |
| 0 | 8  |
| 1 | 8  |
| 2 | ?  |
| 3 | ?  |
| 4 | ?  |
|   | 19 |

# Acessando elementos

- Ao acessar um elemento de um vetor, é necessário especificar a posição do elemento (ou índice do elemento).
- Em um vetor com N elementos, os índices são numerados de 0 a N-1.

É possível acessar o valor armazenado em uma posição.

```
int[] dados = new int[5];  
dados[0] = Convert.ToInt32(Console.ReadLine());  
dados[1] = dados[0];  
dados[2] = dados[1]*2;  
dados[3] = dados[2]-1;  
dados[4] = dados[3]/3;  
Console.WriteLine("{0}", dados[4]);
```

|   |    |
|---|----|
|   |    |
| 0 | 8  |
| 1 | 8  |
| 2 | ?  |
| 3 | ?  |
| 4 | ?  |
|   | 19 |

# Acessando elementos

- Ao acessar um elemento de um vetor, é necessário especificar a posição do elemento (ou índice do elemento).
- Em um vetor com N elementos, os índices são numerados de 0 a N-1.

É possível imprimir o valor armazenado em uma posição.

```
int[] dados = new int[5];  
dados[0] = Convert.ToInt32(Console.ReadLine);  
dados[1] = dados[0];  
dados[2] = dados[1]*2;  
dados[3] = dados[2]-1;  
dados[4] = dados[3]/3;  
Console.WriteLine("{0}", dados[4]);
```

|   |    |
|---|----|
|   |    |
| 0 | 8  |
| 1 | 8  |
| 2 | 16 |
| 3 | 15 |
| 4 | 5  |
|   |    |

# Acessando elementos

- Uma das maiores vantagens do uso de vetores, é a possibilidade de acessar os índices de 0 a N-1 utilizando uma variável (em geral, um contador).

```
const int N = 5;

static void Main(string[] args)
{
    int i;
    int[] dados = new int[N];
    for( i=0; i<N; i++ )
        dados[i] = i;
    ...
}
```

A cada iteração do laço, a variável *i* varia entre 0 e N-1.

Assim, a cada iteração, uma posição distinta do vetor é acessada.

# Exemplo

- Com o uso de vetores, o programa do início da aula pode ser implementado de forma simples.

```
const int NUM_ALUNOS = 10;
static void Main(string[] args)
{
    int i, cont=0;
    double[] nota = new double[NUM_ALUNOS];
    double media, soma = 0;
    for (i = 0; i < NUM_ALUNOS; i++)
    {
        Console.Write("Digite uma nota:");
        nota[i] = Convert.ToDouble(Console.ReadLine());
        soma += nota[i];
    }
    media = soma / NUM_ALUNOS;
    for (i = 0; i < NUM_ALUNOS; i++)
    {
        if( nota[i] > media ) cont++;
    }
    Console.Write("Media = {0}", media);
    Console.Write("{0} notas acima", cont);
}
```

# Exemplo

- Inconvenientes anteriores resolvidos:
- Não há necessidade de o usuário redigitar os valores.
- O programador pode utilizar laços para acessar cada posição, ao invés de inserir uma linha de código específica para cada valor de uma sequência.

```
const int NUM_ALUNOS = 10;
static void Main(string[] args)
{
    int i, cont=0;
    double[] nota = new double[NUM_ALUNOS];
    double media, soma = 0;
    for (i = 0; i < NUM_ALUNOS; i++)
    {
        Console.Write("Digite uma nota:");
        nota[i] = Convert.ToDouble(Console.ReadLine());
        soma += nota[i];
    }
    media = soma / NUM_ALUNOS;
    for (i = 0; i < NUM_ALUNOS; i++)
    {
        if( nota[i] > media ) cont++;
    }
    Console.Write("Media = {0}", media);
    Console.Write("{0} notas acima", cont);
}
```



# Exemplo

- O uso da constante declarada NUM\_ALUNOS para especificar o tamanho do vetor facilita o trabalho do programador em caso de necessidade de alteração do tamanho do vetor.

```
const int NUM_ALUNOS = 10;
static void Main(string[] args)
{
    int i, cont=0;
    double[] nota = new double[NUM_ALUNOS];
    double media, soma = 0;
    for (i = 0; i < NUM_ALUNOS; i++)
    {
        Console.Write("Digite uma nota:");
        nota[i] = Convert.ToDouble(Console.ReadLine());
        soma += nota[i];
    }
    media = soma / NUM_ALUNOS;
    for (i = 0; i < NUM_ALUNOS; i++)
    {
        if( nota[i] > media ) cont++;
    }
    Console.Write("Media = {0}", media);
    Console.Write("{0} notas acima", cont);
}
```

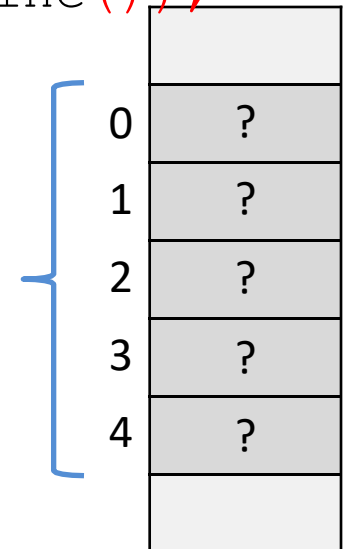
# *Inicialização de vetores*

- De forma diferente à que ocorre com variáveis, os elementos de vetores em C# são automaticamente inicializados assim que o operador `new` é utilizado para instanciar um vetor. Dessa forma, após a criação de um vetor, todas as suas posições contém o valor padrão para aquele tipo.
- Vetores numéricos são inicializados com 0 (zero) e vetores de caracteres são inicializados com ' ' (espaço).

# Inicialização de vetores

- Você pode atribuir valores informados pelo usuário em um vetor da seguinte forma:

```
static void Main(string[] args)
{
    int i;
    int[] dados = new int[5];
    // Leitura dos dados do vetor
    for(i=0; i < 5; i++)
        dados = Convert.ToInt32(Console.ReadLine());
    ...
}
```



# Inicialização de vetores

- Outra forma de inicializar vetores com valores constantes é indicar valores para cada posição no momento da declaração do vetor.
- Em geral, esta alternativa só é utilizada para vetores pequenos.

```
static void Main(string[] args)
{
    // Declara e inicializa o vetor
    int[] dados = {0,0,0,0,0};
    ...
}
```

```
static void Main(string[] args)
{
    int i;
    int dados = new int[5];
    // Inicializa todo o vetor com zero
    for( i=0; i<5; i++ )
        dados[i] = 0;
    ...
}
```

# Vetores: Exemplo 1

O programa a seguir, usa o comando **for** para inicializar com zeros os elementos de um array inteiro **n** de 10 elementos e o imprime sob a forma de uma tabela.

```
static void Main(string[] args)
{
    int i;
    int[] n = new int[10];
    for (i=0; i<10; i++)
    {
        n[i] = 0;
    }
    Console.WriteLine("Elemento    Valor");
    for (i=0; i<10; i++)
    {
        Console.WriteLine("{0,8}{1,8}\n", i , n[i]);
    }
}
```

| Elemento | Valor |
|----------|-------|
| 0        | 0     |
| 1        | 0     |
| 2        | 0     |
| 3        | 0     |
| 4        | 0     |
| 5        | 0     |
| 6        | 0     |
| 7        | 0     |
| 8        | 0     |
| 9        | 0     |

# Vetores: Exemplo 2

O programa abaixo inicializa os dez elementos de um array `s` com os valores: 2, 4, 6, ..., 20 e imprime o vetor em um formato de tabela.

```
const int TAMANHO = 10;
static void Main(string[] args)
{
    int j;
    int[] s = new int[TAMANHO];
    for (j=0; j <= TAMANHO - 1; j++)
    {
        s[j] = 2 + 2 * j;
    }
    Console.WriteLine("Elemento    Valor\n");

    for (j=0; j <= TAMANHO - 1; j++)
        Console.WriteLine("{0,8}{1,8}\n", j, s[j]);
}
```

| Elemento | Valor |
|----------|-------|
| 0        | 2     |
| 1        | 4     |
| 2        | 6     |
| 3        | 8     |
| 4        | 10    |
| 5        | 12    |
| 6        | 14    |
| 7        | 16    |
| 8        | 18    |
| 9        | 20    |

# Vetores: Exemplo 3

O programa abaixo cria um vetor com um valor máximo de tamanho e utiliza apenas uma parte das posições.

```
const int MAX_ALUNOS = 100;
static void Main(string[] args)
{
    int j, numAlunos;
    double[] notas = new double[MAX_ALUNOS];
    Console.Write("Número de alunos: ");
    numAlunos = Convert.ToInt32(Console.ReadLine());
    for (j=0; j < numAlunos; j++)
    {
        Console.Write("Nota do {0}. aluno: ", j+1);
        notas[j] = Convert.ToDouble(Console.ReadLine());
    }
    Console.Write("Fim da leitura");
}
```

```
Número de alunos : 5
Nota do 1. aluno : 7
Nota do 2. aluno : 9
Nota do 3. aluno: 10
Nota do 4. aluno: 4
Nota do 5. aluno: 7
Fim da leitura das notas
```

# Vetores: Exemplo 4

O programa abaixo cria um vetor onde cada posição exerce a função de um contador.

```
const int NUM_CANDIDATOS = 5;
const int NUM_VOTOS = 25 ;
static void Main(string[] args)
{
    int i, cand;
    int[] contador = new int[NUM_CANDIDATOS];

    for(i = 0; i < NUM_VOTOS; i++)
    {
        Console.Write("Entre com {0}o voto:", i);
        cand = Convert.ToInt32(Console.ReadLine());
        contador[cand]++;
    }

    for(i = 0; i < NUM_CANDIDATOS; i++)
        Console.Write("\nCandidato {0}: {1} votos",
            i, contador[i]);
}
```

```
Entre com o 1o voto: 3
Entre com o 2o voto: 1
Entre com o 3o voto: 1
Entre com o 4o voto: 2
Entre com o 5o voto: 1
...
Entre com o 22o voto: 4
Entre com o 23o voto: 0
Entre com o 24o voto: 0

Candidato 0: 5 votos
Candidato 1: 11 votos
Candidato 2: 3 votos
Candidato 3: 2 votos
Candidato 4: 4 votos
...
```



# *Vetores e Sub-rotinas*

- Além de passarmos variáveis simples como parâmetros, também podemos passar **vetores**:
  - A linguagem C# não faz uma cópia dos elementos do vetor na chamada da função.
  - Quando passamos um vetor como parâmetro, a função chamada recebe uma **referência para o vetor**.
  - Isso significa que a função chamada, quando acessa os elementos, acessa as mesmas posições de memória que a função que declarou o vetor.

# Vetores e Sub-rotinas

```
const int TAMANHO = 10;

static void imprimeVetor (int[] vet, int tam)
{
    int i;
    for (i = 0; i <= tam - 1; i++)
    {
        Console.WriteLine("{0}\n", vet[i]);
    }
}

static void Main(string[] args)
{
    int[] s = new int[TAMANHO];
    int i;
    for (i = 0; i <= TAMANHO - 1; i++)
    {
        Console.Write ("Informe o valor do vetor na posicao {0}: ", i);
        s[i] = Convert.ToInt32(Console.ReadLine());
    }
    imprimeVetor(s, TAMANHO);
}
```

# *Exercício resolvido 1*

- Problema: Criar uma função que receba um vetor de números reais e seu tamanho e retorne o índice do maior valor contido no vetor. Se houver mais de uma ocorrência do maior valor, retornar o índice do primeiro. Faça um programa principal para testar a função.
- Vamos ver agora o resultado do teste de mesa para o problema.

# Exercício 1: solução proposta

```
static int encontraMaior (double[] vet, int tam)
{
    int i, indice = 0;
    double maior = vet[0];
    for (i = 1; i < tam; i++)
    {
        if (vet[i] > maior)
        {
            maior = vet[i];
            indice = i;
        }
    }
    return indice;
}

static void Main(string[] args)
{
    double[] vetor = {3.0, 4.3, 5.6, 2.8, 7.9, 3.4};
    int posicao;
    posicao = encontraMaior(vetor, 6);
    Console.WriteLine("Maior valor esta na posicao {0}", posicao);
}
```

# Exercício 1: Teste de Mesa

```
1 static int encontraMaior (double[] vet, int tam)
2 {
3     int i, indice = 0;
4     double maior = vet[0];
5     for (i = 1; i < tam; i++)
6     {
7         if (vet[i] > maior)
8         {
9             maior = vet[i];
10            indice = i;
11        }
12    }
13    return indice;
14 }
15
16 static void Main(string[] args)
17 {
18     double[] vetor = {3.0, 4.3, 5.6,
19                       2.8, 7.9, 3.4};
20     int pos;
21     pos = encontraMaior( vetor, 6);
22     Console.WriteLine("Maior valor na posicao {0}", pos);
23 }
```

**Entrada:**

vetor de tamanho = 6

| i   | 0   | 1   | 2   | 3   | 4   | 5   |
|-----|-----|-----|-----|-----|-----|-----|
| vet | 3.0 | 4.3 | 5.6 | 2.8 | 7.9 | 3.4 |

# Exercício 1: Teste de Mesa

```
1 static int encontraMaior (double[] vet, int tam)
2 {
3     int i, indice = 0;
4     double maior = vet[0];
5     for (i = 1; i < tam; i++)
6     {
7         if (vet[i] > maior)
8         {
9             maior = vet[i];
10            indice = i;
11        }
12    }
13    return indice;
14 }
15
16 static void Main(string[] args)
17 {
18     double[] vetor = {3.0, 4.3, 5.6,
19                       2.8, 7.9, 3.4};
20     int pos;
21     pos = encontraMaior( vetor, 6);
22     Console.WriteLine("Maior valor na posicao {0}", pos);
23 }
```

## Entrada:

vetor de tamanho = 6

## Variáveis da Sub-Rotina:

tam =

i =

indice =

maior =

| i   | 0   | 1   | 2   | 3   | 4   | 5   |
|-----|-----|-----|-----|-----|-----|-----|
| vet | 3.0 | 4.3 | 5.6 | 2.8 | 7.9 | 3.4 |

# Exercício 1: Teste de Mesa

```

1  static int encontraMaior (double[] vet, int tam)
2  {
3      int i, indice = 0;
4      double maior = vet[0];
5      for (i = 1; i < tam; i++)
6      {
7          if (vet[i] > maior)
8          {
9              maior = vet[i];
10             indice = i;
11         }
12     }
13     return indice;
14 }
15
16 static void Main(string[] args)
17 {
18     double[] vetor = {3.0, 4.3, 5.6,
19                       2.8, 7.9, 3.4};
20     int pos;
21     pos = encontraMaior( vetor, 6);
22     Console.WriteLine("Maior valor na posicao {0}", pos);
23 }

```

**Entrada:**  
vetor de tamanho = 6

**Variáveis da Sub-Rotina:**  
tam = 6  
i =  
indice =  
maior =

| i   |     | 0   | 1   | 2   | 3   | 4   | 5 |
|-----|-----|-----|-----|-----|-----|-----|---|
| vet | 3.0 | 4.3 | 5.6 | 2.8 | 7.9 | 3.4 |   |

# Exercício 1: Teste de Mesa

```
1 static int encontraMaior (double[] vet, int tam)
2 {
3     int i, indice = 0;
4     double maior = vet[0];
5     for (i = 1; i < tam; i++)
6     {
7         if (vet[i] > maior)
8         {
9             maior = vet[i];
10            indice = i;
11        }
12    }
13    return indice;
14 }
15
16 static void Main(string[] args)
17 {
18     double[] vetor = {3.0, 4.3, 5.6,
19                       2.8, 7.9, 3.4};
20     int pos;
21     pos = encontraMaior( vetor, 6);
22     Console.WriteLine("Maior valor na posicao {0}", pos);
23 }
```

## Entrada:

vetor de tamanho = 6

## Variáveis da Sub-Rotina:

tam = 6

i = ?

indice = 0

maior = 3.0

| i   | 0   | 1   | 2   | 3   | 4   | 5   |
|-----|-----|-----|-----|-----|-----|-----|
| vet | 3.0 | 4.3 | 5.6 | 2.8 | 7.9 | 3.4 |



# Exercício 1: Teste de Mesa

```
1 static int encontraMaior (double[] vet, int tam)
2 {
3     int i, indice = 0;
4     double maior = vet[0];
5     for (i = 1; i < tam; i++)
6     {
7         if (vet[i] > maior)
8         {
9             maior = vet[i];
10            indice = i;
11        }
12    }
13    return indice;
14 }
15
16 static void Main(string[] args)
17 {
18     double[] vetor = {3.0, 4.3, 5.6,
19                       2.8, 7.9, 3.4};
20     int pos;
21     pos = encontraMaior( vetor, 6);
22     Console.WriteLine("Maior valor na posicao {0}", pos);
23 }
```

## Entrada:

vetor de tamanho = 6

## Variáveis da Sub-Rotina:

tam = 6

i = 1

indice = 0

maior = 3.0

| i   | 0   | 1   | 2   | 3   | 4   | 5   |
|-----|-----|-----|-----|-----|-----|-----|
| vet | 3.0 | 4.3 | 5.6 | 2.8 | 7.9 | 3.4 |

# Exercício 1: Teste de Mesa

```
1 static int encontraMaior (double[] vet, int tam)
2 {
3     int i, indice = 0;
4     double maior = vet[0];
5     for (i = 1; i < tam; i++)
6     {
7         if (vet[i] > maior)
8         {
9             maior = vet[i];
10            indice = i;
11        }
12    }
13    return indice;
14 }
15
16 static void Main(string[] args)
17 {
18     double[] vetor = {3.0, 4.3, 5.6,
19                       2.8, 7.9, 3.4};
20     int pos;
21     pos = encontraMaior( vetor, 6);
22     Console.WriteLine("Maior valor na posicao {0}", pos);
23 }
```

## Entrada:

vetor de tamanho = 6

## Variáveis da Sub-Rotina:

tam = 6

i = 1

indice = 0

maior = 3.0

| i   | 0   | 1   | 2   | 3   | 4   | 5   |
|-----|-----|-----|-----|-----|-----|-----|
| vet | 3.0 | 4.3 | 5.6 | 2.8 | 7.9 | 3.4 |

# Exercício 1: Teste de Mesa

```
1 static int encontraMaior (double[] vet, int tam)
2 {
3     int i, indice = 0;
4     double maior = vet[0];
5     for (i = 1; i < tam; i++)
6     {
7         if (vet[i] > maior)
8         {
9             maior = vet[i];
10            indice = i;
11        }
12    }
13    return indice;
14 }
15
16 static void Main(string[] args)
17 {
18     double[] vetor = {3.0, 4.3, 5.6,
19                       2.8, 7.9, 3.4};
20     int pos;
21     pos = encontraMaior( vetor, 6);
22     Console.WriteLine("Maior valor na posicao {0}", pos);
23 }
```

## Entrada:

vetor de tamanho = 6

## Variáveis da Sub-Rotina:

tam = 6

i = 1

indice = 1

maior = 4.3

| i   | 0   | 1   | 2   | 3   | 4   | 5   |
|-----|-----|-----|-----|-----|-----|-----|
| vet | 3.0 | 4.3 | 5.6 | 2.8 | 7.9 | 3.4 |

# Exercício 1: Teste de Mesa

```
1 static int encontraMaior (double[] vet, int tam)
2 {
3     int i, indice = 0;
4     double maior = vet[0];
5     for (i = 1; i < tam; i++)
6     {
7         if (vet[i] > maior)
8         {
9             maior = vet[i];
10            indice = i;
11        }
12    }
13    return indice;
14 }
15
16 static void Main(string[] args)
17 {
18     double[] vetor = {3.0, 4.3, 5.6,
19                       2.8, 7.9, 3.4};
20     int pos;
21     pos = encontraMaior( vetor, 6);
22     Console.WriteLine("Maior valor na posicao {0}", pos);
23 }
```

## Entrada:

vetor de tamanho = 6

## Variáveis da Sub-Rotina:

tam = 6

i = 2

indice = 1

maior = 4.3

| i   | 0   | 1   | 2   | 3   | 4   | 5   |
|-----|-----|-----|-----|-----|-----|-----|
| vet | 3.0 | 4.3 | 5.6 | 2.8 | 7.9 | 3.4 |

# Exercício 1: Teste de Mesa

```
1 static int encontraMaior (double[] vet, int tam)
2 {
3     int i, indice = 0;
4     double maior = vet[0];
5     for (i = 1; i < tam; i++)
6     {
7         if (vet[i] > maior)
8         {
9             maior = vet[i];
10            indice = i;
11        }
12    }
13    return indice;
14 }
15
16 static void Main(string[] args)
17 {
18     double[] vetor = {3.0, 4.3, 5.6,
19                       2.8, 7.9, 3.4};
20     int pos;
21     pos = encontraMaior( vetor, 6);
22     Console.WriteLine("Maior valor na posicao {0}", pos);
23 }
```

## Entrada:

vetor de tamanho = 6

## Variáveis da Sub-Rotina:

tam = 6

i = 2

indice = 1

maior = 4.3

| i   | 0   | 1   | 2   | 3   | 4   | 5   |
|-----|-----|-----|-----|-----|-----|-----|
| vet | 3.0 | 4.3 | 5.6 | 2.8 | 7.9 | 3.4 |

# Exercício 1: Teste de Mesa

```
1 static int encontraMaior (double[] vet, int tam)
2 {
3     int i, indice = 0;
4     double maior = vet[0];
5     for (i = 1; i < tam; i++)
6     {
7         if (vet[i] > maior)
8         {
9             maior = vet[i];
10            indice = i;
11        }
12    }
13    return indice;
14 }
15
16 static void Main(string[] args)
17 {
18     double[] vetor = {3.0, 4.3, 5.6,
19                       2.8, 7.9, 3.4};
20     int pos;
21     pos = encontraMaior( vetor, 6);
22     Console.WriteLine("Maior valor na posicao {0}", pos);
23 }
```

**Entrada:**  
vetor

**a:**  
r de tamanho = 6

**Variáveis da Sub-Rotina:**  
tam  
i

**Valores:**  
= 6  
= 2  
= 2  
= 5.6

| i   |     | 0 | 1   | 2   | 3   | 4   | 5   |
|-----|-----|---|-----|-----|-----|-----|-----|
| vet | 3.0 | 0 | 4.3 | 5.6 | 2.8 | 7.9 | 3.4 |

# Exercício 1: Teste de Mesa

```
1 static int encontraMaior (double[] vet, int tam)
2 {
3     int i, indice = 0;
4     double maior = vet[0];
5     for (i = 1; i < tam; i++)
6     {
7         if (vet[i] > maior)
8         {
9             maior = vet[i];
10            indice = i;
11        }
12    }
13    return indice;
14 }
15
16 static void Main(string[] args)
17 {
18     double[] vetor = {3.0, 4.3, 5.6,
19                       2.8, 7.9, 3.4};
20     int pos;
21     pos = encontraMaior( vetor, 6);
22     Console.WriteLine("Maior valor na posicao {0}", pos);
23 }
```

## Entrada:

vetor de tamanho = 6

## Variáveis da Sub-Rotina:

**tam = 6**

**i = 3**

indice = 2

maior = 5.6

| i   | 0   | 1   | 2   | 3   | 4   | 5   |
|-----|-----|-----|-----|-----|-----|-----|
| vet | 3.0 | 4.3 | 5.6 | 2.8 | 7.9 | 3.4 |

# Exercício 1: Teste de Mesa

```
1 static int encontraMaior (double[] vet, int tam)
2 {
3     int i, indice = 0;
4     double maior = vet[0];
5     for (i = 1; i < tam; i++)
6     {
7         if (vet[i] > maior)
8         {
9             maior = vet[i];
10            indice = i;
11        }
12    }
13    return indice;
14 }
15
16 static void Main(string[] args)
17 {
18     double[] vetor = {3.0, 4.3, 5.6,
19                       2.8, 7.9, 3.4};
20     int pos;
21     pos = encontraMaior( vetor, 6);
22     Console.WriteLine("Maior valor na posicao {0}", pos);
23 }
```

## Entrada:

vetor de tamanho = 6

## Variáveis da Sub-Rotina:

tam = 6

i = 3

indice = 2

maior = 5.6

| i   | 0   | 1   | 2   | 3   | 4   | 5   |
|-----|-----|-----|-----|-----|-----|-----|
| vet | 3.0 | 4.3 | 5.6 | 2.8 | 7.9 | 3.4 |



# Exercício 1: Teste de Mesa

```
1 static int encontraMaior (double[] vet, int tam)
2 {
3     int i, indice = 0;
4     double maior = vet[0];
5     for (i = 1; i < tam; i++)
6     {
7         if (vet[i] > maior)
8         {
9             maior = vet[i];
10            indice = i;
11        }
12    }
13    return indice;
14 }
15
16 static void Main(string[] args)
17 {
18     double[] vetor = {3.0, 4.3, 5.6,
19                       2.8, 7.9, 3.4};
20     int pos;
21     pos = encontraMaior( vetor, 6);
22     Console.WriteLine("Maior valor na posicao {0}", pos);
23 }
```

## Entrada:

vetor de tamanho = 6

## Variáveis da Sub-Rotina:

tam = 6

i = 4

indice = 2

maior = 5.6

| i   | 0   | 1   | 2   | 3   | 4   | 5   |
|-----|-----|-----|-----|-----|-----|-----|
| vet | 3.0 | 4.3 | 5.6 | 2.8 | 7.9 | 3.4 |

# Exercício 1: Teste de Mesa

```
1 static int encontraMaior (double[] vet, int tam)
2 {
3     int i, indice = 0;
4     double maior = vet[0];
5     for (i = 1; i < tam; i++)
6     {
7         if (vet[i] > maior)
8         {
9             maior = vet[i];
10            indice = i;
11        }
12    }
13    return indice;
14 }
15
16 static void Main(string[] args)
17 {
18     double[] vetor = {3.0, 4.3, 5.6,
19                       2.8, 7.9, 3.4};
20     int pos;
21     pos = encontraMaior( vetor, 6);
22     Console.WriteLine("Maior valor na posicao {0}", pos);
23 }
```

## Entrada:

vetor de tamanho = 6

## Variáveis da Sub-Rotina:

tam = 6

i = 4

indice = 2

maior = 5.6

| i   | 0   | 1   | 2   | 3   | 4   | 5   |
|-----|-----|-----|-----|-----|-----|-----|
| vet | 3.0 | 4.3 | 5.6 | 2.8 | 7.9 | 3.4 |

# Exercício 1: Teste de Mesa

```
1 static int encontraMaior (double[] vet, int tam)
2 {
3     int i, indice = 0;
4     double maior = vet[0];
5     for (i = 1; i < tam; i++)
6     {
7         if (vet[i] > maior)
8         {
9             maior = vet[i];
10            indice = i;
11        }
12    }
13    return indice;
14 }
15
16 static void Main(string[] args)
17 {
18     double[] vetor = {3.0, 4.3, 5.6,
19                       2.8, 7.9, 3.4};
20     int pos;
21     pos = encontraMaior( vetor, 6);
22     Console.WriteLine("Maior valor na posicao {0}", pos);
23 }
```

## Entrada:

vetor de tamanho = 6

## Variáveis da Sub-Rotina:

tam = 6

i = 4

indice = 4

maior = 7.9

| i   | 0   | 1   | 2   | 3   | 4   | 5   |
|-----|-----|-----|-----|-----|-----|-----|
| vet | 3.0 | 4.3 | 5.6 | 2.8 | 7.9 | 3.4 |

# Exercício 1: Teste de Mesa

```
1 static int encontraMaior (double[] vet, int tam)
2 {
3     int i, indice = 0;
4     double maior = vet[0];
5     for (i = 1; i < tam; i++)
6     {
7         if (vet[i] > maior)
8         {
9             maior = vet[i];
10            indice = i;
11        }
12    }
13    return indice;
14 }
15
16 static void Main(string[] args)
17 {
18     double[] vetor = {3.0, 4.3, 5.6,
19                       2.8, 7.9, 3.4};
20     int pos;
21     pos = encontraMaior( vetor, 6);
22     Console.WriteLine("Maior valor na posicao {0}", pos);
23 }
```

## Entrada:

vetor de tamanho = 6

## Variáveis da Sub-Rotina:

tam = 6

i = 5

indice = 4

maior = 7.9

| i   | 0   | 1   | 2   | 3   | 4   | 5   |
|-----|-----|-----|-----|-----|-----|-----|
| vet | 3.0 | 4.3 | 5.6 | 2.8 | 7.9 | 3.4 |

# Exercício 1: Teste de Mesa

```
1 static int encontraMaior (double[] vet, int tam)
2 {
3     int i, indice = 0;
4     double maior = vet[0];
5     for (i = 1; i < tam; i++)
6     {
7         if (vet[i] > maior)
8         {
9             maior = vet[i];
10            indice = i;
11        }
12    }
13    return indice;
14 }
15
16 static void Main(string[] args)
17 {
18     double[] vetor = {3.0, 4.3, 5.6,
19                       2.8, 7.9, 3.4};
20     int pos;
21     pos = encontraMaior( vetor, 6);
22     Console.WriteLine("Maior valor na posicao {0}", pos);
23 }
```

## Entrada:

vetor de tamanho = 6

## Variáveis da Sub-Rotina:

tam = 6

i = 5

indice = 4

maior = 7.9

| i   | 0   | 1   | 2   | 3   | 4   | 5   |
|-----|-----|-----|-----|-----|-----|-----|
| vet | 3.0 | 4.3 | 5.6 | 2.8 | 7.9 | 3.4 |

# Exercício 1: Teste de Mesa

```
1 static int encontraMaior (double[] vet, int tam)
2 {
3     int i, indice = 0;
4     double maior = vet[0];
5     for (i = 1; i < tam; i++)
6     {
7         if (vet[i] > maior)
8         {
9             maior = vet[i];
10            indice = i;
11        }
12    }
13    return indice;
14 }
15
16 static void Main(string[] args)
17 {
18     double[] vetor = {3.0, 4.3, 5.6,
19                       2.8, 7.9, 3.4};
20     int pos;
21     pos = encontraMaior( vetor, 6);
22     Console.WriteLine("Maior valor na posicao {0}", pos);
23 }
```

## Entrada:

vetor de tamanho = 6

## Variáveis da Sub-Rotina:

**tam = 6**

**i = 6**

indice = 4

maior = 7.9

| i   | 0   | 1   | 2   | 3   | 4   | 5   |
|-----|-----|-----|-----|-----|-----|-----|
| vet | 3.0 | 4.3 | 5.6 | 2.8 | 7.9 | 3.4 |

# Exercício 1: Teste de Mesa

```
1 static int encontraMaior (double[] vet, int tam)
2 {
3     int i, indice = 0;
4     double maior = vet[0];
5     for (i = 1; i < tam; i++)
6     {
7         if (vet[i] > maior)
8         {
9             maior = vet[i];
10            indice = i;
11        }
12    }
13    return indice;
14 }
15
16 static void Main(string[] args)
17 {
18     double[] vetor = {3.0, 4.3, 5.6,
19                       2.8, 7.9, 3.4};
20     int pos;
21     pos = encontraMaior( vetor, 6);
22     Console.WriteLine("Maior valor na posicao {0}", pos);
23 }
```

## Entrada:

vetor de tamanho = 6

## Variáveis da Sub-Rotina:

tam = 6

i = 6

indice = 4

maior = 7.9

| i   | 0   | 1   | 2   | 3   | 4   | 5   |
|-----|-----|-----|-----|-----|-----|-----|
| vet | 3.0 | 4.3 | 5.6 | 2.8 | 7.9 | 3.4 |

# Exercício 1: Teste de Mesa

```

1  static int encontraMaior (double[] vet, int tam
2  {
3      int i, indice = 0;
4      double maior = vet[0];
5      for (i = 1; i < tam; i++)
6      {
7          if (vet[i] > maior)
8          {
9              maior = vet[i];
10             indice = i;
11         }
12     }
13     return indice;
14 }
15
16 static void Main(string[] args)
17 {
18     double[] vetor = {3.0, 4.3, 5.6,
19                       2.8, 7.9, 3.4};
20     int pos;
21     pos = encontraMaior( vetor, 6);
22     Console.WriteLine("Maior valor na posicao {0}", pos);
23 }

```

**Entrada**  
vetor

**a:**  
r de tamanho = 6

**Variáveis**  
tam  
i  
indice  
maior

**valores da Sub-Rotina**  
= 6  
= 6  
= 4  
= 7.9

| i   |    | 0 | 1   | 2   | 3   | 4   |   |
|-----|----|---|-----|-----|-----|-----|---|
| vet | 3. | 0 | 4.3 | 5.6 | 2.8 | 7.9 | 3 |

**Saída**  
pos

**:**  
= 4



# *A propriedade Length*

- Para identificar o tamanho de um vetor, basta utilizar a propriedade Length do vetor.
- Exemplo:

```
double[] vetor = new double[5];  
int tamanhoVetor = vetor.Length; //Receberá 5
```

## *Exercício resolvido 2*

- Problema: Criar uma função em C# que receba um vetor de números reais. Essa função deverá ordenar o vetor em ordem crescente.

## Exercício 2: solução proposta

```
void ordena(double[] vet)
{
    int i, j;
    double aux;
    for(i = 0; i < vet.Length; i++)
    {
        for(j = vet.Length - 1; j > i; j--)
        {
            if ( vet[j] < vet[j-1] )
            {
                aux=vet[j];
                vet[j]= vet[j-1];
                vet[j-1]=aux;
            }
        }
    }
}
```

# Exercício 2: Teste de Mesa

```
1 void ordena (double[] vet)
2 {
3     int i, j;
4     double aux;
5     for(i = 0; i < vet.Length; i++)
6     {
7         for(j=vet.Length-1; j>i; j--)
8         {
9             if ( vet[j] < vet[j-1])
10            {
11                aux=vet[j];
12                vet[j] = vet[j-1];
13vet[j-1]=aux; 14 }
15            }
16        }
17 }
```

## Entrada:

vetor de tamanho = 5

## Variáveis:

tam =

i =

j =

aux =

|     | 0    | 1    | 2   | 3    | 4   |
|-----|------|------|-----|------|-----|
| vet | 11.0 | 22.0 | 3.0 | 44.0 | 5.0 |

Vamos supor para esse exercício que o vetor de entrada tenha 5 posições e os seguintes valores:

# Exercício 2: Teste de Mesa

```
1 void ordena (double[] vet)
2 {
3     int i, j;
4     double aux;
5     for(i = 0; i < vet.Length; i++)
6     {
7         for(j=vet.Length-1; j>i; j--)
8         {
9             if ( vet[j] < vet[j-1])
10            {
11                aux=vet[j];
12                vet[j] = vet[j-1];
13                vet[j-1]=aux;
14            }
15        }
16    }
17 }
```

## Entrada:

vetor de tamanho = 5

## Variáveis:

tam = 5

i =

j =

aux =

|     | 0    | 1    | 2   | 3    | 4   |
|-----|------|------|-----|------|-----|
| vet | 11.0 | 22.0 | 3.0 | 44.0 | 5.0 |

# Exercício 2: Teste de Mesa

```
1 void ordena (double[] vet)
2 {
3     int i, j;
4     double aux;
5     for(i = 0; i < vet.Length; i++)
6     {
7         for(j=vet.Length-1; j>i; j--)
8         {
9             if ( vet[j] < vet[j-1])
10            {
11                aux=vet[j];
12                vet[j] = vet[j-1];
13vet[j-1]=aux; 14 }
15         }
16     }
17 }
```

## Entrada:

vetor de tamanho = 5

## Variáveis:

tam = 5

i = ?

j = ?

aux = ?

|     | 0    | 1    | 2   | 3    | 4   |
|-----|------|------|-----|------|-----|
| vet | 11.0 | 22.0 | 3.0 | 44.0 | 5.0 |

# Exercício 2: Teste de Mesa

```
1 void ordena (double[] vet)
2 {
3     int i, j;
4     double aux;
5     for(i = 0; i < vet.Length; i++)
6     {
7         for(j=vet.Length-1; j>i; j--)
8         {
9             if ( vet[j] < vet[j-1])
10            {
11                aux=vet[j];
12                vet[j] = vet[j-1];
13vet[j-1]=aux; 14 }
15            }
16        }
17    }
```

## Entrada:

vetor de tamanho = 5

## Variáveis:

tam = 5

i = 0

j = ?

aux = ?

|     | 0    | 1    | 2   | 3    | 4   |
|-----|------|------|-----|------|-----|
| vet | 11.0 | 22.0 | 3.0 | 44.0 | 5.0 |

↑  
i

# Exercício 2: Teste de Mesa

```
1 void ordena (double[] vet)
2 {
3     int i, j;
4     double aux;
5     for(i = 0; i < vet.Length; i++)
6     {
7         for(j=vet.Length-1; j>i; j--)
8         {
9             if ( vet[j] < vet[j-1])
10            {
11                aux=vet[j];
12                vet[j] = vet[j-1];
13                vet[j-1]=aux;
14            }
15        }
16    }
17 }
```

## Entrada:

vetor de tamanho = 5

## Variáveis:

tam = 5

i = 0

j = 4

aux = ?

|     | 0    | 1    | 2   | 3    | 4   |
|-----|------|------|-----|------|-----|
| vet | 11.0 | 22.0 | 3.0 | 44.0 | 5.0 |

↑ i

↑ j



# Exercício 2: Teste de Mesa

```
1 void ordena (double[] vet)
2 {
3     int i, j;
4     double aux;
5     for(i = 0; i < vet.Length; i++)
6     {
7         for(j=vet.Length-1; j>i; j--)
8         {
9             if ( vet[j] < vet[j-1])
10            {
11                aux=vet[j];
12                vet[j] = vet[j-1];
13                vet[j-1]=aux;
14            }
15        }
16    }
17 }
```

## Entrada:

vetor de tamanho = 5

## Variáveis:

tam = 5

i = 0

j = 4

aux = ?

|     | 0    | 1    | 2   | 3    | 4   |
|-----|------|------|-----|------|-----|
| vet | 11.0 | 22.0 | 3.0 | 44.0 | 5.0 |

↑ i

↑ j

# Exercício 2: Teste de Mesa

```
1 void ordena (double[] vet)
2 {
3     int i, j;
4     double aux;
5     for(i = 0; i < vet.Length; i++)
6     {
7         for(j=vet.Length-1; j>i; j--)
8         {
9             if ( vet[j] < vet[j-1])
10            {
11                aux=vet[j];
12                vet[j] = vet[j-1];
13                vet[j-1]=aux;
14            }
15        }
16    }
17 }
```

## Entrada:

vetor de tamanho = 5

## Variáveis:

tam = 5

i = 0

j = 4

aux = 5.0

|     | 0    | 1    | 2   | 3    | 4   |
|-----|------|------|-----|------|-----|
| vet | 11.0 | 22.0 | 3.0 | 44.0 | 5.0 |

↑ i

↑ j

# Exercício 2: Teste de Mesa

```
1 void ordena (double[] vet)
2 {
3     int i, j;
4     double aux;
5     for(i = 0; i < vet.Length; i++)
6     {
7         for(j=vet.Length-1; j>i; j--)
8         {
9             if ( vet[j] < vet[j-1])
10            {
11                aux=vet[j];
12                vet[j] = vet[j-1];
13                vet[j-1]=aux;
14            }
15        }
16    }
17 }
```

## Entrada:

vetor de tamanho = 5

## Variáveis:

tam = 5

i = 0

j = 4

aux = 5.0

|     | 0    | 1    | 2   | 3  | 4  |
|-----|------|------|-----|----|----|
| vet | 11.0 | 22.0 | 3.0 | 44 | 44 |
|     | ↑    |      |     |    | ↑  |
|     | i    |      |     |    | j  |

# Exercício 2: Teste de Mesa

```
1 void ordena (double[] vet)
2 {
3     int i, j;
4     double aux;
5     for(i = 0; i < vet.Length; i++)
6     {
7         for(j=vet.Length-1; j>i; j--)
8         {
9             if ( vet[j] < vet[j-1])
10            {
11                aux=vet[j];
12                vet[j] = vet[j-1];
13                vet[j-1]=aux;
14            }
15        }
16    }
17 }
```

## Entrada:

vetor de tamanho = 5

## Variáveis:

tam = 5

i = 0

j = 4

aux = 5.0

|     | 0    | 1    | 2   | 3   | 4    |
|-----|------|------|-----|-----|------|
| vet | 11.0 | 22.0 | 3.0 | 5.0 | 44.0 |

↑ i

↑ j

## Exercício 2: Teste de Mesa

```

1 void ordena (double[] vet)
2 {
3     int i, j;
4     double aux;
5     for(i = 0; i < vet.Length; i++)
6     {
7         for(j=vet.Length-1; j>i; j--)
8         {
9             if ( vet[j] < vet[j-1])
10            {
11                aux=vet[j];
12                vet[j] = vet[j-1];
13                vet[j-1]=aux;
14            }
15        }
16    }
17 }

```

## Entrada:

vetor de tamanho = 5

## Variáveis:

tam = 5

i = 0

j = 3

aux = 5.0

|     | 0    | 1    | 2   | 3   | 4    |
|-----|------|------|-----|-----|------|
| vet | 11.0 | 22.0 | 3.0 | 5.0 | 44.0 |

$i \quad j$

## Exercício 2: Teste de Mesa

```

1 void ordena (double[] vet)
2 {
3     int i, j;
4     double aux;
5     for(i = 0; i < vet.Length; i++)
6     {
7         for(j=vet.Length-1; j>i; j--)
8         {
9             if ( vet[j] < vet[j-1])
10            {
11                aux=vet[j];
12                vet[j] = vet[j-1];
13                vet[j-1]=aux;
14            }
15        }
16    }
17 }

```

## Entrada:

vetor de tamanho = 5

## Variáveis:

tam = 5

i = 0

**j = 3**

aux = 5.0

|     | 0    | 1    | 2   | 3   | 4    |
|-----|------|------|-----|-----|------|
| vet | 11.0 | 22.0 | 3.0 | 5.0 | 44.0 |

$\uparrow$                        $\uparrow$

$i$                                    $j$

# Exercício 2: Teste de Mesa

```
1 void ordena (double[] vet)
2 {
3     int i, j;
4     double aux;
5     for(i = 0; i < vet.Length; i++)
6     {
7         for(j=vet.Length-1; j>i; j--)
8         {
9             if ( vet[j] < vet[j-1])
10            {
11                aux=vet[j];
12                vet[j] = vet[j-1];
13                vet[j-1]=aux;
14            }
15        }
16    }
17 }
```

## Entrada:

vetor de tamanho = 5

## Variáveis:

tam = 5

i = 0

j = 2

aux = 5.0

|     | 0    | 1    | 2   | 3   | 4    |
|-----|------|------|-----|-----|------|
| vet | 11.0 | 22.0 | 3.0 | 5.0 | 44.0 |
|     | ↑    |      | ↑   |     |      |
|     | i    |      | j   |     |      |

# Exercício 2: Teste de Mesa

```
1 void ordena (double[] vet)
2 {
3     int i, j;
4     double aux;
5     for(i = 0; i < vet.Length; i++)
6     {
7         for(j=vet.Length-1; j>i; j--)
8         {
9             if ( vet[j] < vet[j-1])
10            {
11                aux=vet[j];
12                vet[j] = vet[j-1];
13                vet[j-1]=aux;
14            }
15        }
16    }
17 }
```

## Entrada:

vetor de tamanho = 5

## Variáveis:

tam = 5

i = 0

j = 2

aux = 5.0

|     | 0    | 1    | 2   | 3   | 4    |
|-----|------|------|-----|-----|------|
| vet | 11.0 | 22.0 | 3.0 | 5.0 | 44.0 |
|     | ↑    |      | ↑   |     |      |
|     | i    |      | j   |     |      |



# Exercício 2: Teste de Mesa

```
1 void ordena (double[] vet)
2 {
3     int i, j;
4     double aux;
5     for(i = 0; i < vet.Length; i++)
6     {
7         for(j=vet.Length-1; j>i; j--)
8         {
9             if ( vet[j] < vet[j-1])
10            {
11                aux=vet[j];
12                vet[j] = vet[j-1];
13                vet[j-1]=aux;
14            }
15        }
16    }
17 }
```

## Entrada:

vetor de tamanho = 5

## Variáveis:

tam = 5

i = 0

j = 2

aux = 3.0

|     | 0    | 1   | 2    | 3   | 4    |
|-----|------|-----|------|-----|------|
| vet | 11.0 | 3.0 | 22.0 | 5.0 | 44.0 |
|     | ↑    |     | ↑    |     |      |
|     | i    |     | j    |     |      |

# Exercício 2: Teste de Mesa

```
1 void ordena (double[] vet)
2 {
3     int i, j;
4     double aux;
5     for(i = 0; i < vet.Length; i++)
6     {
7         for(j=vet.Length-1; j>i; j--)
8         {
9             if ( vet[j] < vet[j-1])
10            {
11                aux=vet[j];
12                vet[j] = vet[j-1];
13                vet[j-1]=aux;
14            }
15        }
16    }
17 }
```

## Entrada:

vetor de tamanho = 5

## Variáveis:

tam = 5

i = 0

j = 1

aux = 3.0

|     | 0    | 1   | 2    | 3   | 4    |
|-----|------|-----|------|-----|------|
| vet | 11.0 | 3.0 | 22.0 | 5.0 | 44.0 |
|     | ↑    | ↑   |      |     |      |
|     | i    | j   |      |     |      |

# Exercício 2: Teste de Mesa

```
1 void ordena (double[] vet)
2 {
3     int i, j;
4     double aux;
5     for(i = 0; i < vet.Length; i++)
6     {
7         for(j=vet.Length-1; j>i; j--)
8         {
9             if ( vet[j] < vet[j-1])
10            {
11                aux=vet[j];
12                vet[j] = vet[j-1];
13                vet[j-1]=aux;
14            }
15        }
16    }
17 }
```

## Entrada:

vetor de tamanho = 5

## Variáveis:

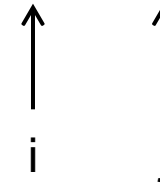
tam = 5

i = 0

j = 1

aux = 3.0

|     | 0    | 1   | 2    | 3   | 4    |
|-----|------|-----|------|-----|------|
| vet | 11.0 | 3.0 | 22.0 | 5.0 | 44.0 |



# Exercício 2: Teste de Mesa

```
1 void ordena (double[] vet)
2 {
3     int i, j;
4     double aux;
5     for(i = 0; i < vet.Length; i++)
6     {
7         for(j=vet.Length-1; j>i; j--)
8         {
9             if ( vet[j] < vet[j-1])
10            {
11                aux=vet[j];
12                vet[j] = vet[j-1];
13                vet[j-1]=aux;
14            }
15        }
16    }
17 }
```

## Entrada:

vetor de tamanho = 5

## Variáveis:

tam = 5

i = 0

j = 1

aux = 3.0

|     | 0   | 1    | 2    | 3   | 4    |
|-----|-----|------|------|-----|------|
| vet | 3.0 | 11.0 | 22.0 | 5.0 | 44.0 |
|     | ↑   | ↑    |      |     |      |
|     | i   | j    |      |     |      |

# Exercício 2: Teste de Mesa

```
1 void ordena (double[] vet)
2 {
3     int i, j;
4     double aux;
5     for(i = 0; i < vet.Length; i++)
6     {
7         for(j=vet.Length-1; j>i; j--)
8         {
9             if ( vet[j] < vet[j-1])
10            {
11                aux=vet[j];
12                vet[j] = vet[j-1];
13                vet[j-1]=aux;
14            }
15        }
16    }
17 }
```

## Entrada:

vetor de tamanho = 5

## Variáveis:

tam = 5

i = 0

j = 0

aux = 3.0

|     | 0   | 1    | 2    | 3   | 4    |
|-----|-----|------|------|-----|------|
| vet | 3.0 | 11.0 | 22.0 | 5.0 | 44.0 |

↑↑  
i j

# Exercício 2: Teste de Mesa

```
1 void ordena (double[] vet)
2 {
3     int i, j;
4     double aux;
5     for(i = 0; i < vet.Length; i++)
6     {
7         for(j=vet.Length-1; j>i; j--)
8         {
9             if ( vet[j] < vet[j-1])
10            {
11                aux=vet[j];
12                vet[j] = vet[j-1];
13vet[j-1]=aux; 14 }
15            }
16        }
17 }
```

## Entrada:

vetor de tamanho = 5

## Variáveis:

tam = 5

i = 1

j = 0

aux = 3.0

|     | 0   | 1    | 2    | 3   | 4    |
|-----|-----|------|------|-----|------|
| vet | 3.0 | 11.0 | 22.0 | 5.0 | 44.0 |
| j   | ↑   |      |      |     |      |
| i   |     | ↑    |      |     |      |

# Exercício 2: Teste de Mesa

```
1 void ordena (double[] vet)
2 {
3     int i, j;
4     double aux;
5     for(i = 0; i < vet.Length; i++)
6     {
7         for(j=vet.Length-1; j>i; j--)
8         {
9             if ( vet[j] < vet[j-1])
10            {
11                aux=vet[j];
12                vet[j] = vet[j-1];
13                vet[j-1]=aux;
14            }
15        }
16    }
17 }
```

## Entrada:

vetor de tamanho = 5

## Variáveis:

tam = 5

i = 1

j = 4

aux = 3.0

|     | 0   | 1    | 2    | 3   | 4    |
|-----|-----|------|------|-----|------|
| vet | 3.0 | 11.0 | 22.0 | 5.0 | 44.0 |

↑  
i

↑  
j

## Exercício 2: Teste de Mesa

```

1 void ordena (double[] vet)
2 {
3     int i, j;
4     double aux;
5     for(i = 0; i < vet.Length; i++)
6     {
7         for(j=vet.Length-1; j>i; j--)
8         {
9             if ( vet[j] < vet[j-1])
10            {
11                aux=vet[j];
12                vet[j] = vet[j-1];
13                vet[j-1]=aux;
14            }
15        }
16    }
17 }

```

## Entrada:

vetor de tamanho = 5

## Variáveis:

tam = 5

$$i = 1$$

j = 4

aux = 3.0

|     | 0   | 1    | 2    | 3   | 4    |
|-----|-----|------|------|-----|------|
| vet | 3.0 | 11.0 | 22.0 | 5.0 | 44.0 |

↑
↑  
i
j



# Exercício 2: Teste de Mesa

```
1 void ordena (double[] vet)
2 {
3     int i, j;
4     double aux;
5     for(i = 0; i < vet.Length; i++)
6     {
7         for(j=vet.Length-1; j>i; j--)
8         {
9             if ( vet[j] < vet[j-1])
10            {
11                aux=vet[j];
12                vet[j] = vet[j-1];
13                vet[j-1]=aux;
14            }
15        }
16    }
17 }
```

## Entrada:

vetor de tamanho = 5

## Variáveis:

tam = 5

i = 1

j = 3

aux = 3.0

|     | 0   | 1    | 2    | 3   | 4    |
|-----|-----|------|------|-----|------|
| vet | 3.0 | 11.0 | 22.0 | 5.0 | 44.0 |

↑  
i

↑  
j

# Exercício 2: Teste de Mesa

```
1 void ordena (double[] vet)
2 {
3     int i, j;
4     double aux;
5     for(i = 0; i < vet.Length; i++)
6     {
7         for(j=vet.Length-1; j>i; j--)
8         {
9             if ( vet[j] < vet[j-1])
10            {
11                aux=vet[j];
12                vet[j] = vet[j-1];
13                vet[j-1]=aux;
14            }
15        }
16    }
17 }
```

## Entrada:

vetor de tamanho = 5

## Variáveis:

tam = 5

i = 1

j = 3

aux = 3.0

|     | 0   | 1    | 2    | 3   | 4    |
|-----|-----|------|------|-----|------|
| vet | 3.0 | 11.0 | 22.0 | 5.0 | 44.0 |

↑  
i

↑  
j

# Exercício 2: Teste de Mesa

```
1 void ordena (double[] vet)
2 {
3     int i, j;
4     double aux;
5     for(i = 0; i < vet.Length; i++)
6     {
7         for(j=vet.Length-1; j>i; j--)
8         {
9             if ( vet[j] < vet[j-1])
10            {
11                aux=vet[j];
12                vet[j] = vet[j-1];
13                vet[j-1]=aux;
14            }
15        }
16    }
17 }
```

## Entrada:

vetor de tamanho = 5

## Variáveis:

tam = 5

i = 1

j = 3

aux = 5.0

|     | 0   | 1    | 2   | 3    | 4    |
|-----|-----|------|-----|------|------|
| vet | 3.0 | 11.0 | 5.0 | 22.0 | 44.0 |

↑  
i

↑  
j

# Exercício 2: Teste de Mesa

```
1 void ordena (double[] vet)
2 {
3     int i, j;
4     double aux;
5     for(i = 0; i < vet.Length; i++)
6     {
7         for(j=vet.Length-1; j>i; j--)
8         {
9             if ( vet[j] < vet[j-1])
10            {
11                aux=vet[j];
12                vet[j] = vet[j-1];
13                vet[j-1]=aux;
14            }
15        }
16    }
17 }
```

## Entrada:

vetor de tamanho = 5

## Variáveis:

tam = 5

i = 1

j = 2

aux = 5.0

|     | 0   | 1    | 2   | 3    | 4    |
|-----|-----|------|-----|------|------|
| vet | 3.0 | 11.0 | 5.0 | 22.0 | 44.0 |

↑  
i

↑  
j

# Exercício 2: Teste de Mesa

```
1 void ordena (double[] vet)
2 {
3     int i, j;
4     double aux;
5     for(i = 0; i < vet.Length; i++)
6     {
7         for(j=vet.Length-1; j>i; j--)
8         {
9             if ( vet[j] < vet[j-1])
10            {
11                aux=vet[j];
12                vet[j] = vet[j-1];
13                vet[j-1]=aux;
14            }
15        }
16    }
17 }
```

## Entrada:

vetor de tamanho = 5

## Variáveis:

tam = 5

i = 1

j = 2

aux = 5.0

|     | 0   | 1    | 2   | 3    | 4    |
|-----|-----|------|-----|------|------|
| vet | 3.0 | 11.0 | 5.0 | 22.0 | 44.0 |

↑  
i

↑  
j

# Exercício 2: Teste de Mesa

```
1 void ordena (double[] vet)
2 {
3     int i, j;
4     double aux;
5     for(i = 0; i < vet.Length; i++)
6     {
7         for(j=vet.Length-1; j>i; j--)
8         {
9             if ( vet[j] < vet[j-1])
10            {
11                aux=vet[j];
12                vet[j] = vet[j-1];
13                vet[j-1]=aux;
14            }
15        }
16    }
17 }
```

## Entrada:

vetor de tamanho = 5

## Variáveis:

tam = 5

i = 1

j = 2

aux = 5.0

|     | 0   | 1   | 2    | 3    | 4    |
|-----|-----|-----|------|------|------|
| vet | 3.0 | 5.0 | 11.0 | 22.0 | 44.0 |

↑  
i

↑  
j

# Exercício 2: Teste de Mesa

```
1 void ordena (double[] vet)
2 {
3     int i, j;
4     double aux;
5     for(i = 0; i < vet.Length; i++)
6     {
7         for(j=vet.Length-1; j>i; j--)
8         {
9             if ( vet[j] < vet[j-1])
10            {
11                aux=vet[j];
12                vet[j] = vet[j-1];
13                vet[j-1]=aux;
14            }
15        }
16    }
17 }
```

## Entrada:

vetor de tamanho = 5

## Variáveis:

tam = 5

i = 1

j = 1

aux = 5.0

|     | 0   | 1   | 2    | 3    | 4    |
|-----|-----|-----|------|------|------|
| vet | 3.0 | 5.0 | 11.0 | 22.0 | 44.0 |

↑ ↑  
i j

# Exercício 2: Teste de Mesa

```
1 void ordena (double[] vet)
2 {
3     int i, j;
4     double aux;
5     for(i = 0; i < vet.Length; i++)
6     {
7         for(j=vet.Length-1; j>i; j--)
8         {
9             if ( vet[j] < vet[j-1])
10            {
11                aux=vet[j];
12                vet[j] = vet[j-1];
13vet[j-1]=aux; 14 }
15            }
16        }
17 }
```

## Entrada:

vetor de tamanho = 5

## Variáveis:

tam = 5

i = 2

j = 1

aux = 5.0

|     | 0   | 1   | 2    | 3    | 4    |
|-----|-----|-----|------|------|------|
| vet | 3.0 | 5.0 | 11.0 | 22.0 | 44.0 |

↑  
j

↑  
i



# Exercício 2: Teste de Mesa

```
1 void ordena (double[] vet)
2 {
3     int i, j;
4     double aux;
5     for(i = 0; i < vet.Length; i++)
6     {
7         for(j=vet.Length-1; j>i; j--)
8         {
9             if ( vet[j] < vet[j-1])
10            {
11                aux=vet[j];
12                vet[j] = vet[j-1];
13                vet[j-1]=aux;
14            }
15        }
16    }
17 }
```

## Entrada:

vetor de tamanho = 5

## Variáveis:

tam = 5

i = 2

j = 4

aux = 5.0

|     | 0   | 1   | 2    | 3    | 4    |
|-----|-----|-----|------|------|------|
| vet | 3.0 | 5.0 | 11.0 | 22.0 | 44.0 |

↑  
i

↑  
j

# Exercício 2: Teste de Mesa

```
1 void ordena (double[] vet)
2 {
3     int i, j;
4     double aux;
5     for(i = 0; i < vet.Length; i++)
6     {
7         for(j=vet.Length-1; j>i; j--)
8         {
9             if ( vet[j] < vet[j-1])
10            {
11                aux=vet[j];
12                vet[j] = vet[j-1];
13                vet[j-1]=aux;
14            }
15        }
16    }
17 }
```

## Entrada:

vetor de tamanho = 5

## Variáveis:

tam = 5

i = 2

j = 4

aux = 5.0

|     | 0   | 1   | 2    | 3    | 4    |
|-----|-----|-----|------|------|------|
| vet | 3.0 | 5.0 | 11.0 | 22.0 | 44.0 |

↑  
i

↑  
j

# Exercício 2: Teste de Mesa

```
1 void ordena (double[] vet)
2 {
3     int i, j;
4     double aux;
5     for(i = 0; i < vet.Length; i++)
6     {
7         for(j=vet.Length-1; j>i; j--)
8         {
9             if ( vet[j] < vet[j-1])
10            {
11                aux=vet[j];
12                vet[j] = vet[j-1];
13                vet[j-1]=aux;
14            }
15        }
16    }
17 }
```

## Entrada:

vetor de tamanho = 5

## Variáveis:

tam = 5

i = 2

j = 3

aux = 5.0

|     | 0   | 1   | 2    | 3    | 4    |
|-----|-----|-----|------|------|------|
| vet | 3.0 | 5.0 | 11.0 | 22.0 | 44.0 |

↑  
i

↑  
j

# Exercício 2: Teste de Mesa

```
1 void ordena (double[] vet)
2 {
3     int i, j;
4     double aux;
5     for(i = 0; i < vet.Length; i++)
6     {
7         for(j=vet.Length-1; j>i; j--)
8         {
9             if ( vet[j] < vet[j-1])
10            {
11                aux=vet[j];
12                vet[j] = vet[j-1];
13                vet[j-1]=aux;
14            }
15        }
16    }
17 }
```

## Entrada:

vetor de tamanho = 5

## Variáveis:

tam = 5

i = 2

j = 3

aux = 5.0

|     | 0   | 1   | 2    | 3    | 4    |
|-----|-----|-----|------|------|------|
| vet | 3.0 | 5.0 | 11.0 | 22.0 | 44.0 |

↑  
i

↑  
j

# Exercício 2: Teste de Mesa

```
1 void ordena (double[] vet)
2 {
3     int i, j;
4     double aux;
5     for(i = 0; i < vet.Length; i++)
6     {
7         for(j=vet.Length-1; j>i; j--)
8         {
9             if ( vet[j] < vet[j-1])
10            {
11                aux=vet[j];
12                vet[j] = vet[j-1];
13                vet[j-1]=aux;
14            }
15        }
16    }
17 }
```

## Entrada:

vetor de tamanho = 5

## Variáveis:

tam = 5

i = 2

j = 2

aux = 5.0

|     | 0   | 1   | 2    | 3    | 4    |
|-----|-----|-----|------|------|------|
| vet | 3.0 | 5.0 | 11.0 | 22.0 | 44.0 |

↑↑  
i j

# Exercício 2: Teste de Mesa

```
1 void ordena (double[] vet)
2 {
3     int i, j;
4     double aux;
5     for(i = 0; i < vet.Length; i++)
6     {
7         for(j=vet.Length-1; j>i; j--)
8         {
9             if ( vet[j] < vet[j-1])
10            {
11                aux=vet[j];
12                vet[j] = vet[j-1];
13vet[j-1]=aux; 14 }
15            }
16        }
17    }
```

## Entrada:

vetor de tamanho = 5

## Variáveis:

tam = 5

i = 3

j = 2

aux = 5.0

|     | 0   | 1   | 2    | 3    | 4    |
|-----|-----|-----|------|------|------|
| vet | 3.0 | 5.0 | 11.0 | 22.0 | 44.0 |

↑      ↑  
j      i

# Exercício 2: Teste de Mesa

```
1 void ordena (double[] vet)
2 {
3     int i, j;
4     double aux;
5     for(i = 0; i < vet.Length; i++)
6     {
7         for(j=vet.Length-1; j>i; j--)
8         {
9             if ( vet[j] < vet[j-1])
10            {
11                aux=vet[j];
12                vet[j] = vet[j-1];
13                vet[j-1]=aux;
14            }
15        }
16    }
17 }
```

## Entrada:

vetor de tamanho = 5

## Variáveis:

tam = 5

i = 3

j = 4

aux = 5.0

|     | 0   | 1   | 2    | 3    | 4    |
|-----|-----|-----|------|------|------|
| vet | 3.0 | 5.0 | 11.0 | 22.0 | 44.0 |

↑      ↑  
i      j

# Exercício 2: Teste de Mesa

```
1 void ordena (double[] vet)
2 {
3     int i, j;
4     double aux;
5     for(i = 0; i < vet.Length; i++)
6     {
7         for(j=vet.Length-1; j>i; j--)
8         {
9             if ( vet[j] < vet[j-1])
10            {
11                aux=vet[j];
12                vet[j] = vet[j-1];
13                vet[j-1]=aux;
14            }
15        }
16    }
17 }
```

## Entrada:

vetor de tamanho = 5

## Variáveis:

tam = 5

i = 3

j = 4

aux = 5.0

|     | 0   | 1   | 2    | 3    | 4    |
|-----|-----|-----|------|------|------|
| vet | 3.0 | 5.0 | 11.0 | 22.0 | 44.0 |

↑      ↑  
i      j



# Exercício 2: Teste de Mesa

```
1 void ordena (double[] vet)
2 {
3     int i, j;
4     double aux;
5     for(i = 0; i < vet.Length; i++)
6     {
7         for(j=vet.Length-1; j>i; j--)
8         {
9             if ( vet[j] < vet[j-1])
10            {
11                aux=vet[j];
12                vet[j] = vet[j-1];
13                vet[j-1]=aux;
14            }
15        }
16    }
17 }
```

## Entrada:

vetor de tamanho = 5

## Variáveis:

tam = 5

i = 3

j = 3

aux = 5.0

|     | 0   | 1   | 2    | 3    | 4    |
|-----|-----|-----|------|------|------|
| vet | 3.0 | 5.0 | 11.0 | 22.0 | 44.0 |

↑↑  
i j

# Exercício 2: Teste de Mesa

```
1 void ordena (double[] vet)
2 {
3     int i, j;
4     double aux;
5     for(i = 0; i < vet.Length; i++)
6     {
7         for(j=vet.Length-1; j>i; j--)
8         {
9             if ( vet[j] < vet[j-1])
10            {
11                aux=vet[j];
12                vet[j] = vet[j-1];
13vet[j-1]=aux; 14 }
15            }
16        }
17    }
```

## Entrada:

vetor de tamanho = 5

## Variáveis:

tam = 5

i = 4

j = 3

aux = 5.0

|     | 0   | 1   | 2    | 3    | 4    |
|-----|-----|-----|------|------|------|
| vet | 3.0 | 5.0 | 11.0 | 22.0 | 44.0 |

↑      ↑  
j      i

# Exercício 2: Teste de Mesa

```
1 void ordena (double[] vet)
2 {
3     int i, j;
4     double aux;
5     for(i = 0; i < vet.Length; i++)
6     {
7         for(j=vet.Length-1; j>i; j--)
8         {
9             if ( vet[j] < vet[j-1])
10            {
11                aux=vet[j];
12                vet[j] = vet[j-1];
13vet[j-1]=aux; 14 }
15            }
16        }
17 }
```

## Entrada:

vetor de tamanho = 5

## Variáveis:

tam = 5

i = 4

j = 3

aux = 5.0

**Saída:** vetor modificado  
(ordenado)

|     | 0   | 1   | 2    | 3    | 4    |
|-----|-----|-----|------|------|------|
| vet | 3.0 | 5.0 | 11.0 | 22.0 | 44.0 |

# Exercício 2

```
void ordena(double[] vet)
{
    int i, j;
    double aux;
    for(i = 0; i < vet.Length; i++)
    {
        for(j=vet.Length-1; j>i; j--)
        {
            if ( vet[j] < vet[j-1] )
            {
                aux=vet[j];
                vet[j]= vet[j-1];
                vet[j-1]=aux;
            }
        }
    }
}

static void Main(string[] args)
{
    int i;
    double[] vet={11.0,22.0,3.0,44.0,5.0};
    ordena(vet);
    for (i=0; i < 5; i++)
        Console.WriteLine("%.2f\n",vet[i]);
}
```

- Programa completo.
- Esse algoritmo de ordenação é conhecido como algoritmo da ordenação por bolha (ou *bubble sort*).

# Exercícios

1) Quais são os elementos do vetor referenciados pelas expressões abaixo ?

vet

|          |          |          |          |          |          |          |          |          |          |          |          |          |
|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|
| <u>1</u> | <u>2</u> | <u>4</u> | <u>7</u> | <u>4</u> | <u>2</u> | <u>8</u> | <u>9</u> | <u>0</u> | <u>6</u> | <u>5</u> | <u>4</u> | <u>3</u> |
|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|

a) vet[3]

b) vet[0]

c) vet[13]

2) Qual é a diferença entre os números "3" das duas instruções abaixo ?

```
int vet[] = new int[3];  
vet[3] = 5;
```

## *Exercícios*

- 3) Dada uma tabela contendo a idade de 10 alunos, faça um algoritmo que calcule o número de alunos com idade superior a média.
- 4) Faça um algoritmo para ler e somar dois vetores de 10 elementos inteiros. Imprima ao final os valores dessa soma, elemento a elemento.
- 5) Refaça o exercício anterior criando um procedimento para efetuar a leitura dos vetores e um segundo procedimento que imprimirá a soma dos vetores.

## *Exercícios*

- 6) Refaça o exercício (3) criando uma função que receba o vetor com a idade dos alunos e retorne a quantidade de alunos com idade superior a média.
- 7) Faça um algoritmo que leia, via teclado, 20 valores do tipo inteiro e determine qual o menor valor existente no vetor e imprima esse valor e seu índice no vetor.
- 8) Refaça o exercício anterior criando um procedimento que receba como parâmetro o vetor e imprima o menor valor e seu índice no vetor.

# Vetores Numéricos

Aula de Exercícios



# *Vetores*

- O vetor é uma estrutura:
  - Homogênea
  - Estática
- Todas as componentes são de um mesmo tipo e seu tamanho permanece o mesmo durante toda a execução do programa.

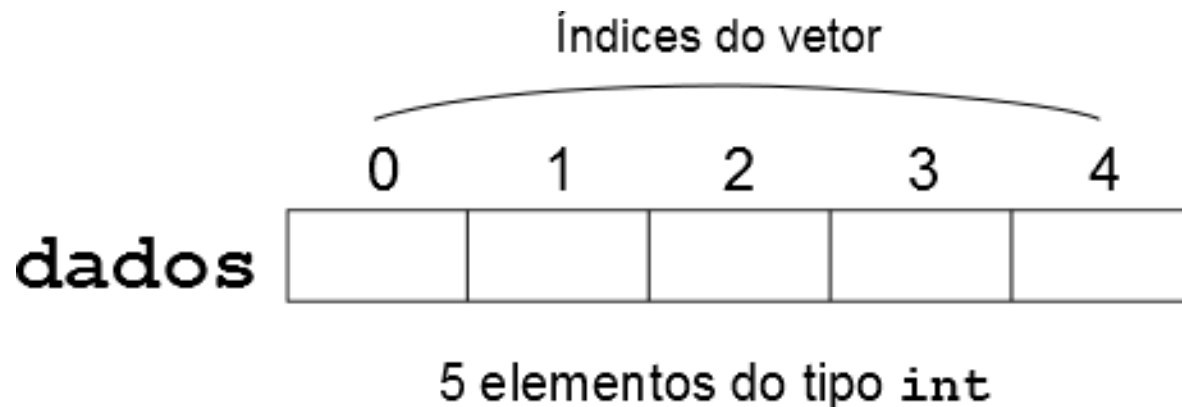
# Vetores: declaração

- A sintaxe em C# para declaração de variável do tipo vetor é a seguinte:

```
tipoPrimitivo[] identificador = new tipo[TAMANHO];
```

- Exemplo:

```
// vetor com 5 elementos do tipo int  
int[] dados = new int[5];
```



# Vetores: manipulação

- Cada um dos elementos de um vetor é referenciado individualmente por meio de um número inteiro entre colchetes após o nome do vetor.

|                |     |     |     |     |     |
|----------------|-----|-----|-----|-----|-----|
|                | 0   | 1   | 2   | 3   | 4   |
| <b>valores</b> | 3.6 | 2.7 | 4.2 | 7.9 | 1.2 |

- Exemplos:  
`X = valores[1];` //atribui a x o valor da posição 1 do vetor valores  
`Y = valores[4];` //atribui a x o valor da posição 4 do vetor valores  
`valores[0] = 3.2;` // a posição zero do vetor valores recebe o valor 3.2

# Vetores: exemplo

- Leia um vetor de 10 posições (inteiros) e imprima-o na ordem invertida (da última para a primeira posição).

```
static void Main(string[] args)
{
    int[] numeros = new int[10];
    int i;
    for (i=0; i<10;i++)
    {
        Console.Write("Digite valor {0}: ",i);
        numeros[i] = Convert.ToInt32(Console.ReadLine());
    }
    Console.Write("Vetor na ordem invertida:\n");

    for (i=9; i>=0; i--)
        Console.Write("{0}\n", numeros[i]);
}
```

# *Vetores e Sub-rotinas*

- Em C#, vetores são passados sempre por referência.
- Ou seja, as modificações feitas na subrotina refletem nos dados do vetor passado como parâmetro pela função chamadora.

# Vetores e Sub-rotinas: Exemplo

```
const int TAMANHO 10

// definicao de outras subrotinas
// leVetor, imprimeVetor, maiorElemento

double mediaVetor(int[] vet, int tam)
{
    int i, soma = 0;
    for(i = 0; i < tam; i++){
        soma = soma + vet[i];
    }
    return soma / (double)tam;
}

static void Main(string[] args)
{
    int[] v = new int[TAMANHO];
    leVetor(v, TAMANHO);
    imprimeVetor(v, TAMANHO);
    Console.WriteLine("\nMaior = {0}.", maiorElemento(v, TAMANHO));
    Console.WriteLine("\nMédia = {0}.", mediaVetor(v, TAMANHO));
}
```

## *Exercícios*

- 1) Desenvolva um programa que leia um vetor de números reais, um escalar e imprima o resultado da multiplicação do vetor pelo escalar.
- 2) Faça um procedimento que faça a leitura um vetor de 10 elementos inteiros e imprima somente os valores armazenados nos índices pares.
- 3) Faça um programa que leia um vetor com 15 valores reais. A seguir, encontre o menor elemento do vetor e a sua posição dentro do vetor, mostrando: "O menor elemento do vetor está na posição XXXX e tem o valor YYYYYY."
- 4) Faça um programa que leia um vetor de 15 posições (reais) e depois um valor a ser procurado no vetor. Imprima se o valor foi ou não encontrado e a quantidade de vezes que o valor está presente no vetor.

## Exercícios

- 5) Faça uma função que receba um vetor de números inteiros e um valor inteiro. A função deverá procurar este segundo valor neste vetor e retornar seu índice se for encontrado. Se o elemento não for encontrado, retornar -1.
- 6) Dada uma tabela com as notas de uma turma de 20 alunos, faça funções que retornem:
- a) A média da turma.
  - b) a quantidade de alunos aprovados ( $\geq 60$ )
  - c) a quantidade de alunos reprovados. ( $< 60$ )
- 7) Faça um programa que leia um conjunto de 20 valores e armazene-os num vetor V. Particione-o em dois outros vetores, P e I, conforme os valores de V forem pares ou ímpares. No final, imprima os valores dos 3 vetores.



## *Exercícios*

8) Faça um programa que leia um vetor  $G[13]$  que é o gabarito de um teste da loteria esportiva, contendo os valores 1 quando for coluna 1, 0 quando for coluna do meio e 2 quando for coluna 2.

Ler a seguir, para 5 apostadores, seu cartão de apostas ( $R[13]$ ) e depois da leitura imprimir quantos acertos o apostador teve.

Faça o teste através de funções.

9) Com relação ao exercício anterior, calcule e mostre o percentual dos apostadores que fizeram de 10 a 13 pontos e o percentual dos apostadores que fizeram menos do que 10 pontos.

10) Faça um programa que leia um vetor de valores inteiros e imprima-o na ordem crescente. O vetor deve ter tamanho  $N$  (declare e utilize uma constante  $N$ ).