

Cetec
Capacitações

CENTRO PAULA SOUZA

GOVERNO DO ESTADO
SÃO PAULO

GOVERNO FEDERAL
BRASIL
PAÍS RICO É PAÍS SEM POBREZA



Atualização Técnica e Pedagógica de Professores no componente de Lógica de Programação com C# (console)

Semana 4

Vetor e Matriz

Prof. Tiago Jesus de Souza

Introdução

Vetores e Matrizes em geral são caracterizadas por se tratarem de uma única variável de um determinado tamanho que armazena várias informações do mesmo tipo.

Essas informações são gravadas na memória sequencialmente e são referenciadas através de índices.

Em vez de declarar 100 variáveis para armazenar por exemplo o nome de 100 alunos, utilizando vetor, basta declarar apenas uma única variável de tamanho 100. Ou seja, em cada posição do vetor será armazenado o nome de cada aluno.

Vetores – Unidimensionais

Matrizes – Multidimensionais

São muito utilizados por exemplo, em desenvolvimento de jogos, análise de conjunto de dados, desenhos, cálculos matemáticos entre outros.

Vetor

É uma variável que possui apenas uma única dimensão.

A declaração de vetores em C# deve obedecer a seguinte sintaxe:

```
Tipo[ ] nome_variável = new Tipo[tamanho];
```

O Tipo deve ser especificado de acordo com o tipo de informação que será armazenada no vetor (ex. int float, char, string...).

E o tamanho representa a quantidade máxima de elementos que podem ser armazenados neste vetor.

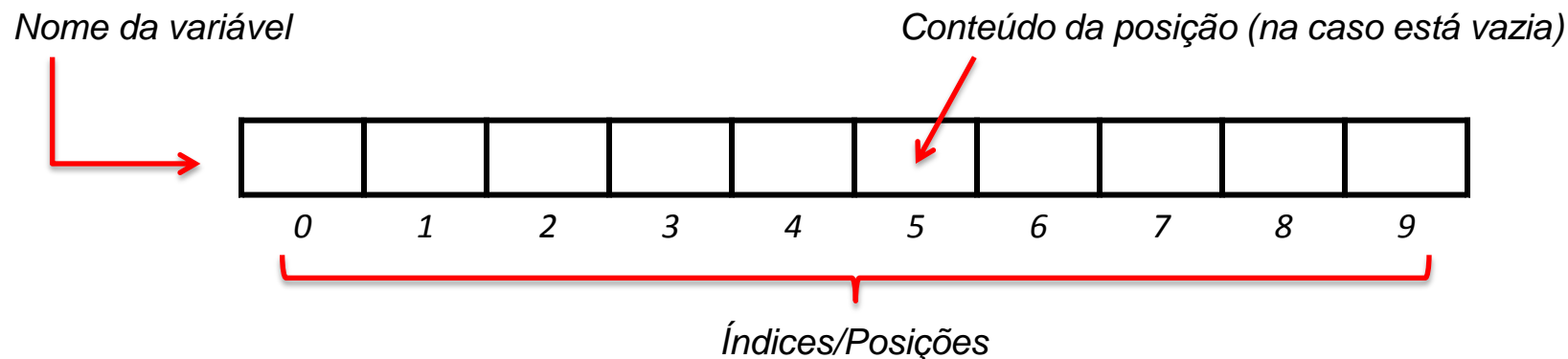
É importante dizer que na linguagem C# os vetores começam pelo índice 0 (zero) que guarda o primeiro elemento do vetor.

Vetor – Exemplo de Declaração

Para entender melhor, considere que seja necessário declarar um vetor do tipo inteiro que contenha 10 elementos. Isto é feito da seguinte forma:

```
int[ ] vetor_exemplo = new int[10];
```

A variável **vetor_exemplo** possui 10 posições, sendo que seus índices vão de 0 a 9.



Exemplo 1 – Sem uso de Vetor

Neste exemplo, o programa irá armazenar o nome de 10 pessoas e depois exibi-los na tela.

Observem que foram necessárias a declaração de 10 variáveis, onde cada uma armazenará o nome de uma pessoa.

```
static void Main(string[] args)
{
    string n1, n2, n3, n4, n5, n6, n7, n8, n9, n10;

    Console.WriteLine("Digite o 1º nome: "); n1 = Console.ReadLine();
    Console.WriteLine("Digite o 2º nome: "); n2 = Console.ReadLine();
    Console.WriteLine("Digite o 3º nome: "); n3 = Console.ReadLine();
    Console.WriteLine("Digite o 4º nome: "); n4 = Console.ReadLine();
    Console.WriteLine("Digite o 5º nome: "); n5 = Console.ReadLine();
    Console.WriteLine("Digite o 6º nome: "); n6 = Console.ReadLine();
    Console.WriteLine("Digite o 7º nome: "); n7 = Console.ReadLine();
    Console.WriteLine("Digite o 8º nome: "); n8 = Console.ReadLine();
    Console.WriteLine("Digite o 9º nome: "); n9 = Console.ReadLine();
    Console.WriteLine("Digite o 10º nome: "); n10 = Console.ReadLine();
    Console.WriteLine();
    Console.WriteLine("1º nome: " + n1);
    Console.WriteLine("2º nome: " + n2);
    Console.WriteLine("3º nome: " + n3);
    Console.WriteLine("4º nome: " + n4);
    Console.WriteLine("5º nome: " + n5);
    Console.WriteLine("6º nome: " + n6);
    Console.WriteLine("7º nome: " + n7);
    Console.WriteLine("8º nome: " + n8);
    Console.WriteLine("9º nome: " + n9);
    Console.WriteLine("10º nome: " + n10);

    Console.ReadKey();
}
```

Exemplo 1 – Sem uso de Vetor - Resultado

Foram digitados os 10 nomes e na sequência foram exibidos na tela, todos os nomes na ordem em que foram digitados.

```
file:///C:/Aulas C#/Aula4_Exemplo1/Aula4_Exemplo1/bin/Debug/Aula4_Exemplo1.EXE
Digite o 1º nome: Chico
Digite o 2º nome: Angela
Digite o 3º nome: Morge
Digite o 4º nome: Vera
Digite o 5º nome: Edson
Digite o 6º nome: Fátima
Digite o 7º nome: Zinho
Digite o 8º nome: Iraci
Digite o 9º nome: Carlos
Digite o 10º nome: Cida

1º nome: Chico
2º nome: Angela
3º nome: Morge
4º nome: Vera
5º nome: Edson
6º nome: Fátima
7º nome: Zinho
8º nome: Iraci
9º nome: Carlos
10º nome: Cida
```

Exemplo 1 – Com uso de Vetor

Com base no exemplo anterior, o propósito é o mesmo, o programa irá armazenar o nome de 10 pessoas e depois exibi-los na tela.

Observem que foi necessária a declaração de apenas 1 variável para armazenar os 10 nomes e uma outra do tipo *int* para referenciar o índice no vetor.

```
static void Main(string[] args)
{
    string[] nomes = new string[10];
    int i;

    for (i = 0; i < 10; i++)
    {
        Console.Write("Digite o {0}º nome: ", i+1);
        nomes[i] = Console.ReadLine();
    }

    Console.WriteLine();

    for (i = 0; i < 10; i++)
    {
        Console.WriteLine("{0}º nome:{1} ", i+1, nomes[i]);
    }

    Console.ReadKey();
}
```


Exemplo 2 – Com uso de Vetor - Resultado

Foram digitados os 10 nomes e na sequência foram exibidos na tela, todos os nomes na ordem em que foram digitados.

Se compararmos com o resultado do exemplo 1, veremos que o formato de saída são idênticos.

```
file:///C:/Aulas C#/Aula4_Exemplo2/Aula4_Exemplo2/bin/Debug/Aula4_Exemplo2.EXE
Digite o 1º nome: Claudinei
Digite o 2º nome: Luzia
Digite o 3º nome: João
Digite o 4º nome: Carmem
Digite o 5º nome: Zé
Digite o 6º nome: Susi
Digite o 7º nome: Cidinho
Digite o 8º nome: Tiana
Digite o 9º nome: Dito
Digite o 10º nome: Áurea

1º nome: Claudinei
2º nome: Luzia
3º nome: João
4º nome: Carmem
5º nome: Zé
6º nome: Susi
7º nome: Cidinho
8º nome: Tiana
9º nome: Dito
10º nome: Áurea
```

Exemplo 2 – Com uso de Vetor – Analisando o 1º Bloco

Imediatamente após a declaração das variáveis, este bloco permite que sejam armazenados os 10 nomes de pessoas.

A estrutura de repetição **for** é responsável por repetir as instruções por 10 vezes, onde *i* é a variável de controle deste bloco e será incrementada até chegar em 10, ou seja, até que a condição seja falsa.

Na formatação de saída {0}, irá aparecer o resultado da soma de *i*+1, ou seja, a primeira vez *i* vale **0**, então somando 0 e 1 aparecerá **1** na tela. “*Digite o 1º nome:* “. O usuário digitará o primeiro nome que será armazenado na posição **0** do vetor.

No segundo momento, após *i* ser incrementada, a mesma passa a valer **1**, aparecendo **2** na tela “*Digite o 2º nome:* “. O usuário digitará o

segundo nome que será armazenado na posição **1** do vetor. E assim sucessivamente.

```
for (i = 0; i < 10; i++)  
{  
    Console.WriteLine("Digite o {0}º nome: ", i+1);  
    nomes[i] = Console.ReadLine();  
}
```

Exemplo 2 – Com uso de Vetor – Analisando o 2º Bloco

Depois da execução do 1º Bloco, que foi responsável por armazenar os nomes no vetor, este 2º Bloco é responsável por exibir os dados que estão armazenados no vetor.

Na formatação de saída {0}, segue o mesmo adotado no 1º Bloco, apenas para identificar ao usuário uma sequência.

Na formatação de saída {1}, será exibido primeiramente o nome que está armazenado no vetor na posição 0, pois *i* inicia com valor 0. Em uma segunda iteração, será exibido o nome que está na posição 1 do vetor e assim sucessivamente.

```
for (i = 0; i < 10; i++)  
{  
    Console.WriteLine("{0}º nome:{1} ", i+1, nomes[i]);  
}
```

Com ou Sem uso de Vetor? - Comparativos

Analisando as modificações necessárias para tal adaptação, o uso de vetor se faz necessário para uma melhor codificação, onde no desenvolvimento não precisou realizar grandes modificações. Sem o uso de vetor seriam necessárias por volta 270 linhas de código a mais para adaptar ao novo cenário, o que não é viável.

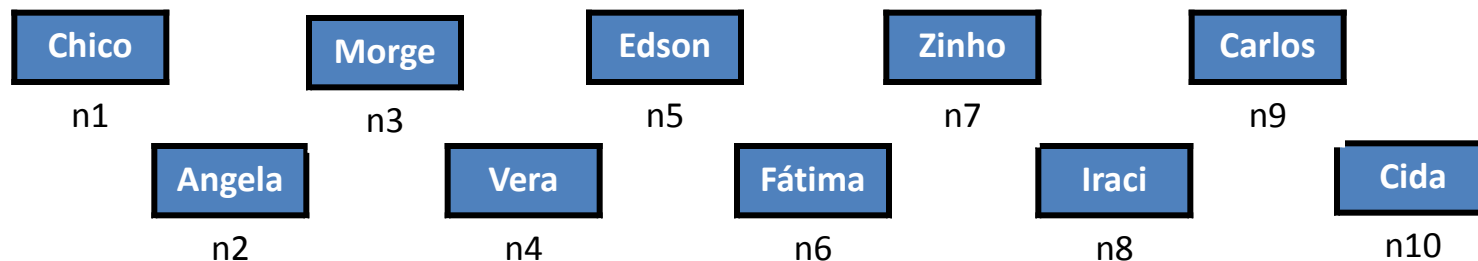
Além do mais, se o cliente ainda quiser solicitar mais adaptações, como por exemplo, saber qual é o nome que possui mais caracteres, o nome que possui menos caracteres, quantidade de nomes que iniciam com vogal ou mesmo para fazer uma simples pesquisa para saber se existe um nome cadastrado.

Então, se não usarmos vetores, as implementações sugeridas pelo cliente com certeza não atenderão as expectativas.

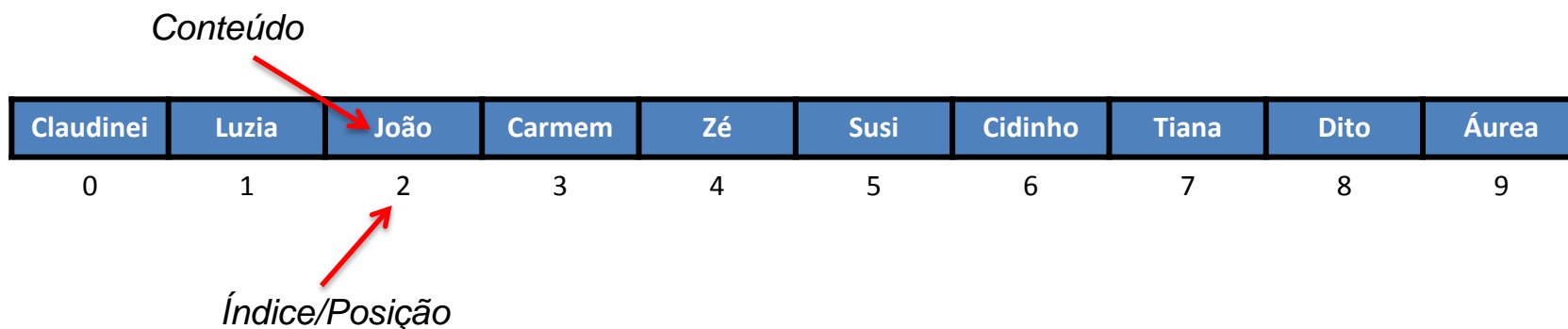
Neste caso e outros que veremos, é necessário o uso de vetores.

Diferença no Armazenamento

Sem uso de Vetor: os valores são armazenados em variáveis independentes.



Com uso de Vetor: os valores são armazenados em uma única variável.



Exemplo 3 – Transformando string em vetor de char

Neste exemplo será apresentado um método que transforma/converte uma string em um vetor de caracteres. O usuário digitará uma palavra e serão exibidas apenas as vogais desta palavra.

```
static void Main(string[] args)
{
    string palavra;
    int i;

    Console.Write("Digite uma palavra: ");
    palavra = Console.ReadLine();

    Console.WriteLine("\nVogais da palavra: {0} \n", palavra);

    char[] vetor = palavra.ToCharArray();

    int tamanho = vetor.Length;

    for (i = 0; i < tamanho; i++)
    {
        if (vetor[i] == 'a' || vetor[i] == 'e' || vetor[i] == 'i' || vetor[i] == 'o' || vetor[i] == 'u')
            Console.Write(" {0}", vetor[i]);
    }

    Console.ReadKey();
}
```


Exemplo 3 – Transformando string em vetor de char - Resultado

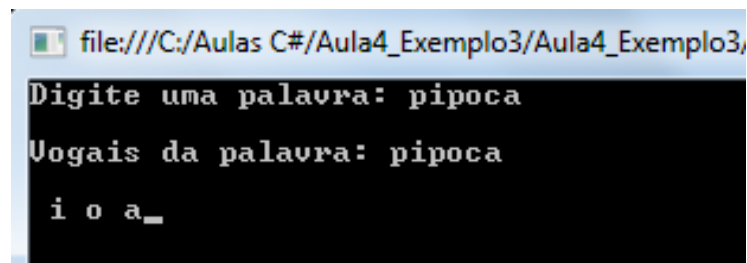
O resultado apresentado exibiu apenas as vogais da palavra digitada pelo usuário.

Neste exemplo somente foram tratadas vogais

minúsculas, mas poderiam ser realizados outros testes como comparar maiúsculas e também vogais com acentuação.

A instrução: `char[] vetor = palavra.ToCharArray();` realiza a transformação de uma string em um vetor de caracteres, pois para realizar o teste para saber se cada elemento do vetor é uma vogal, deve ser comparado caractere por caractere.

A instrução: `int tamanho = vetor.Length;` realiza a verificação de quantos caracteres existem no vetor e armazena esta quantidade na variável **tamanho**, para que possa ser usada como limite para o número de iterações na estrutura de repetição **for**.



```
file:///C:/Aulas C#/Aula4_Exemplo3/Aula4_Exemplo3
Digite uma palavra: pipoca
Vogais da palavra: pipoca
i o a_
```

Exemplo 4 – De 1 a 10 Salários Mínimos

Neste exemplo, apenas serão apresentados os valores correspondentes a quantidade de 1 a 10 salários mínimos.

```
static void Main(string[] args)
{
    int i;
    double sal_minimo = 678.00;
    double[] salarios = new double[10];

    //Atribuição do calculo no vetor salarios
    for (i = 0; i < salarios.Length; i++)
        salarios[i] = (i + 1) * sal_minimo;

    //Exibição dos dados calculados
    for (i = 0; i < salarios.Length; i++)
        Console.WriteLine("\n{0} Salario(s) Minimo(s) = {1}", (i + 1), salarios[i].ToString("#00.00"));

    Console.ReadKey();
}
```

Exemplo 4 – De 1 a 10 Salários Mínimos – Resultado

Uma particularidade apresentada neste exemplo, é a forma de exibição dos valores, onde foi realizada a transformação de um valor real (*double*) em uma *string*, de acordo com uma máscara pré-determinada.

```
file:///C:/Aulas C#/Aula4_Exemplo4/Aula4_Exemplo4/
1 Salario(s) Minimo(s) = 678,00
2 Salario(s) Minimo(s) = 1356,00
3 Salario(s) Minimo(s) = 2034,00
4 Salario(s) Minimo(s) = 2712,00
5 Salario(s) Minimo(s) = 3390,00
6 Salario(s) Minimo(s) = 4068,00
7 Salario(s) Minimo(s) = 4746,00
8 Salario(s) Minimo(s) = 5424,00
9 Salario(s) Minimo(s) = 6102,00
10 Salario(s) Minimo(s) = 6780,00
```

Exemplo 5 – Passando vetor por parâmetros

Neste exemplo será enviado um vetor por parâmetro para o método *exibirDados*.

Este método irá receber o vetor e por meio de uma estrutura de repetição, irá exibir os dados que estão armazenados neste Vetor.

```
static void Main(string[] args)
{
    int i;
    const int tam = 10;
    string[] times = new string[tam];

    Console.WriteLine("*** Times de Futebol ***\n");

    //Atribuição do calculo no vetor salarios
    for (i = 0; i < tam; i++)
    {
        Console.Write("Nome do {0}º time: ", i + 1);
        times[i] = Console.ReadLine();
    }
    Console.WriteLine("\n *** Times cadastrados no Vetor ***\n");
    exibirDados(times);

    Console.ReadKey();
}

static void exibirDados(string[] t)
{
    //Exibição dos dados do vetor
    for (int x = 0; x < t.Length; x++)
        Console.WriteLine("{0}º time: {1}", x+1, t[x]);
}
```

Exemplo 5 – Passando vetor por parâmetros - Resultado

Com o resultado apresentado, nota-se que a sequência que foram exibidos os nomes dos times na tela, foi a mesma sequência do cadastro.

Isso se deve pelo fato do acesso sequencial à cada posição do vetor, ou seja, das posições de 0 a 9.

```
file:///C:/Aulas C#/Aula4_Exemplo5/Aula4_Exemplo5/
*** Times de Futebol ***
Nome do 1º time: Guarani
Nome do 2º time: Barcelona
Nome do 3º time: Real Madri
Nome do 4º time: XU de Piracicaba
Nome do 5º time: Santos
Nome do 6º time: Bahia
Nome do 7º time: Internacional
Nome do 8º time: Cruzeiro
Nome do 9º time: Ituano
Nome do 10º time: Ponte Preta

*** Times cadastrados no Vetor ***
1º time: Guarani
2º time: Barcelona
3º time: Real Madri
4º time: XU de Piracicaba
5º time: Santos
6º time: Bahia
7º time: Internacional
8º time: Cruzeiro
9º time: Ituano
10º time: Ponte Preta
-
```

Exemplo 6 – Método que retorna um vetor

Neste exemplo foi criado um método chamado *cadastroCurso*, que será responsável por realizar a entrada de dados e retornar todos os dados cadastrados para o método principal.

Na sequência os dados serão exibidos.

```
static void Main(string[] args)
{
    int i;
    const int tam = 5;
    string[] cursos = new string[tam];

    Console.WriteLine("*** Cursos Técnicos ***\n");

    //Chamada para execução do método para cadastro de cursos
    cursos = cadastroCursos();

    Console.WriteLine("\n *** Cursos cadastrados no vetor ***\n");
    for (int x = 0; x < cursos.Length; x++)
        Console.WriteLine("{0}º Curso: {1}", x + 1, cursos[x]);

    Console.ReadKey();
}

static string[] cadastroCursos()
{
    string[] vetorAuxiliar = new string[5];
    //Entrada dos dados no vetor
    for (int x = 0; x < 5; x++)
    {
        Console.Write("Digite o nome do {0}º Curso: ", x + 1);
        vetorAuxiliar[x] = Console.ReadLine();
    }
    return vetorAuxiliar;
}
```


Exemplo 6 – Método que retorna um vetor - Resultado

O método que realiza o cadastro dos cursos, possui uma variável auxiliar (vetor) para armazenar os dados digitados pelo usuário.

Logo após todas as entradas serem realizadas pelo usuário, este método retornará um vetor de string para o método principal, cujos valores serão atribuídos em outra variável (cursos).

```
file:///C:/Aulas C#/Aula4_Exemplo6/Aula4_Exemplo6/bin/Debug/Aula4_Exemplo6.EXE
*** Cursos Técnicos ***
Digite o nome do 1º Curso: Informática
Digite o nome do 2º Curso: Eletrônica
Digite o nome do 3º Curso: Games
Digite o nome do 4º Curso: Logística
Digite o nome do 5º Curso: Administração

*** Cursos cadastrados no Vetor ***
1º Curso: Informática
2º Curso: Eletrônica
3º Curso: Games
4º Curso: Logística
5º Curso: Administração
```

Matriz

É uma variável que pode possuir mais de uma dimensão.

A declaração de matrizes bidimensionais em C# deve obedecer a seguinte sintaxe:

```
Tipo[, ] nome_variável = new Tipo[qtde_linhas, qtde_colunas];
```

O Tipo deve ser especificado de acordo com o tipo de informação que será armazenada na matriz (ex. int float, char, string...).

E a *qtde_linha* representa a quantidade máxima de linhas da matriz e *qtde_colunas* representa a quantidade máxima de colunas da matriz.

É importante dizer que na linguagem C# as matrizes começam pelo índice 0 (zero) tanto na linha quanto na coluna, para indicar a posição do primeiro elemento na matriz.

Matriz – Exemplo de Declaração e Inicialização

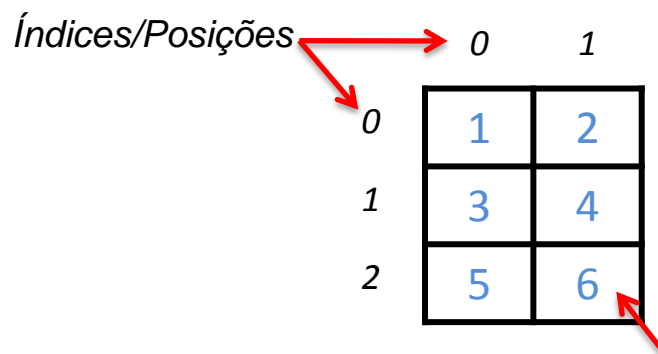
Também é possível inicializar o vetor no momento de sua declaração. Para isso veja a sintaxe abaixo:

```
Tipo [, ] nome_matriz = { { valor1, valor2}, {valor3, valor4}, {valor5, valor6} };
```

Sendo que cada chave representa uma linha da matriz, portanto neste exemplo, a matriz é do tamanho 3 (linhas) x 2 (colunas)

Declaração da variável **matriz_exemplo** com os valores atribuídos.

```
int [, ] matriz_exemplo = { { 1, 2 }, { 3, 4 }, { 5, 6 } };
```



Conteúdo da posição linha 2 e coluna 1

Exemplo 7 – Cadastro e Exibição de dados usando Matriz

Neste exemplo, o propósito é cadastrar 9 números Inteiros em uma matriz e depois exibi-los na tela.

Observem que foi necessária a declaração de apenas 1 variável para armazenar os 9 números inteiros e outras 2 variáveis do tipo *int* para referenciar as posições Linha e coluna na matriz.

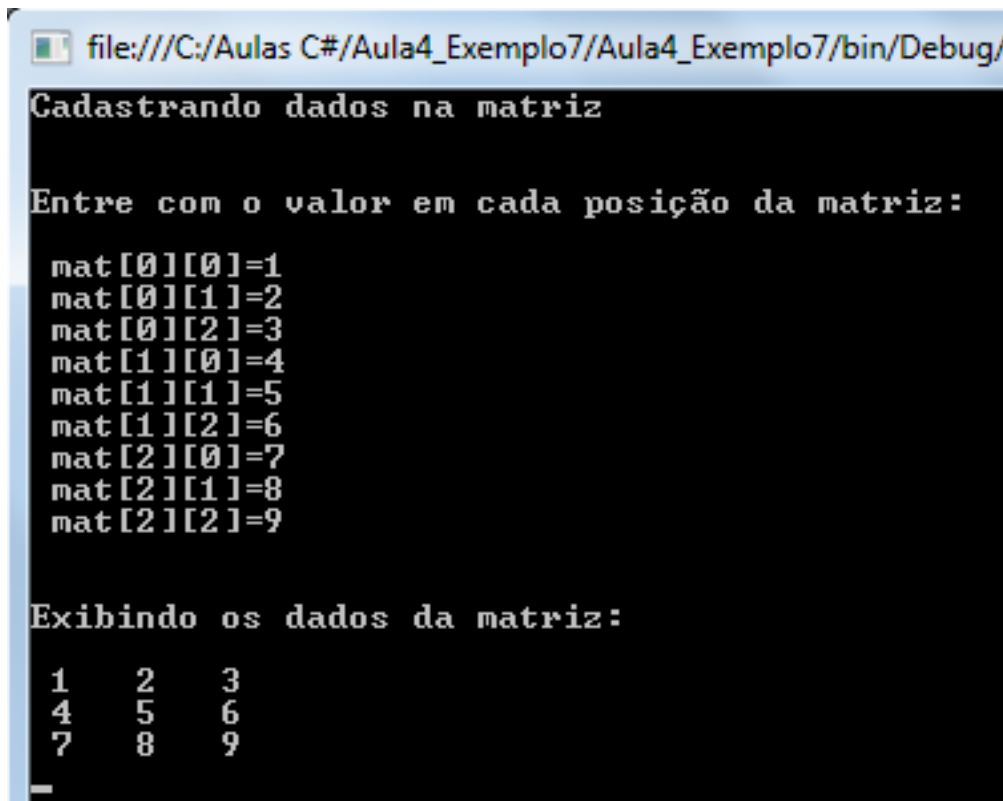
```
static void Main(string[] args)
{
    const int N_LIN = 3, N_COL = 3;
    int[,] mat = new int[N_LIN, N_COL];
    int l,c;
    Console.WriteLine("Cadastrando dados na matriz\n");
    Console.WriteLine("\nEntre com o valor em cada posição da matriz:\n");
    for(l=0; l<N_LIN; l++)
    {
        for(c=0; c<N_COL; c++)
        {
            Console.Write(" mat[{0}][{1}]=", l, c);
            mat[l,c] = int.Parse(Console.ReadLine());
        } /*fim do for c*/
    } /*fim do for l*/

    Console.WriteLine("\n\nExibindo os dados da matriz:\n");
    for(l=0; l<N_LIN;l++)
    {
        for(c=0;c<N_COL;c++)
        {
            Console.Write(" {0} ", mat[l,c]);
        }
        Console.WriteLine();
    }
    Console.ReadKey();
}
```

Exemplo 7 – Cadastro e Exibição de dados usando Matriz - Resultado

Foram digitados 9 números inteiros e na sequência todos os números foram exibidos na tela, de acordo com o conteúdo de cada posição da matriz.

Notamos que para exibir no formato de matriz, logo após o término de cada iteração do segundo *for*, foi usando um `Console.WriteLine()`, ou seja, após percorrer todas as colunas de cada linha, é necessário pular uma linha.



```
file:///C:/Aulas C#/Aula4_Exemplo7/Aula4_Exemplo7/bin/Debug/
Cadastrando dados na matriz

Entre com o valor em cada posição da matriz:

mat[0][0]=1
mat[0][1]=2
mat[0][2]=3
mat[1][0]=4
mat[1][1]=5
mat[1][2]=6
mat[2][0]=7
mat[2][1]=8
mat[2][2]=9

Exibindo os dados da matriz:

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

Exemplo 8 – Matriz com valores pré-definidos

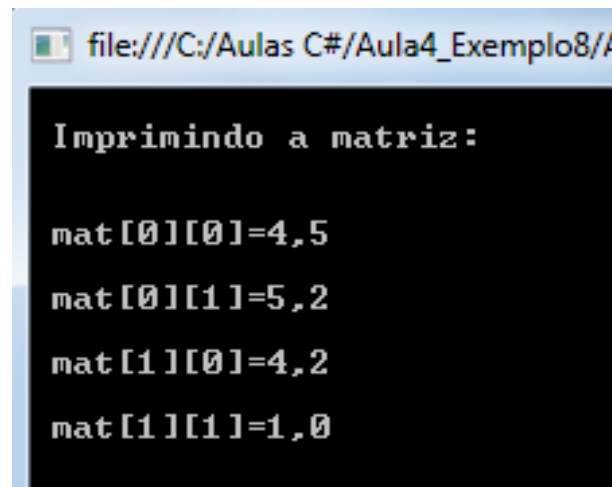
Neste exemplo, a matriz foi declarada e já inicializada com valores definidos em tempo de programação. E na sequência os dados são exibidos na tela.

```
static void Main(string[] args)
{
    double[,] matriz = { { 4.5, 5.2 }, { 4.2, 1 } };

    int x, y;
    Console.WriteLine("\n Imprimindo a matriz:\n");
    for (x = 0; x < 2; x++)
    {
        for (y = 0; y < 2; y++)
        {
            Console.WriteLine("\n mat[{0}][{1}]={2:N1}", x, y, matriz[x, y]);
        }
    }
    Console.ReadKey();
}
```


Exemplo 8 – Matriz com valores pré-definidos - Resultado

O resultado apresentado, mostra os valores reais sendo exibidos com apenas uma casa decimal após a vírgula.



```
file:///C:/Aulas C#/Aula4_Exemplo8/A  
Imprimindo a matriz:  
mat[0][0]=4,5  
mat[0][1]=5,2  
mat[1][0]=4,2  
mat[1][1]=1,0
```

Exemplo 9 – Testando elementos da Matriz

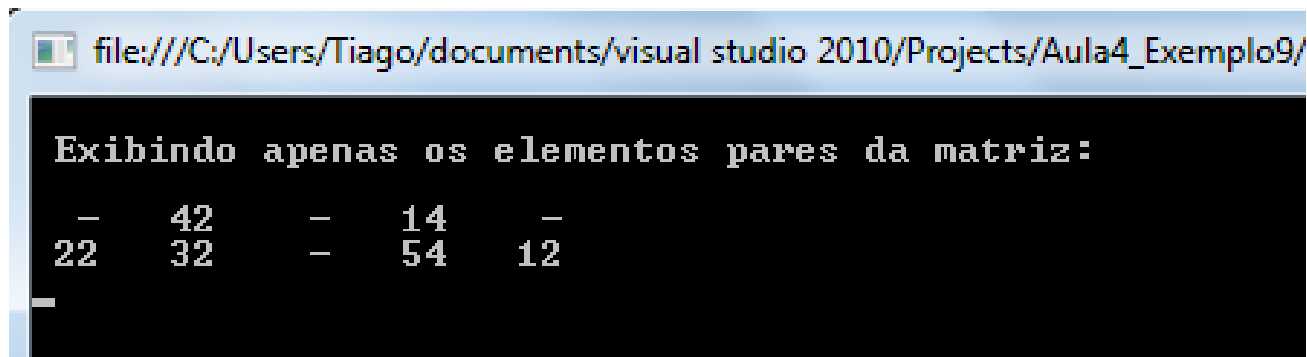
Neste exemplo, a matriz também foi declarada e já inicializada com valores definidos em tempo de programação. E na sequência os dados são exibidos na tela.

```
static void Main(string[] args)
{
    int[,] mat = { { 1, 42, 23, 14, 51 }, { 22, 32, 55, 54, 12 } };

    int x, y;
    Console.WriteLine("\n Exibindo apenas os elementos pares da matriz:\n");
    for (x = 0; x < 2; x++)
    {
        for (y = 0; y < 5; y++)
        {
            if (mat[x, y] % 2 == 0)
                Console.Write(" {0:D2} ", mat[x, y]);
            else
                Console.Write(" - ");
        }
        Console.WriteLine();
    }
    Console.ReadKey();
}
```

Exemplo 9 – Testando elementos da Matriz - Resultado

O resultado apresentado, mostra apenas os valores pares que estão armazenados na matriz.



```
file:///C:/Users/Tiago/documents/visual studio 2010/Projects/Aula4_Exemplo9/  
Exibindo apenas os elementos pares da matriz:  
- 42 - 14 -  
22 32 - 54 12
```

Exemplo 10 – Passando matriz por parâmetros

Neste exemplo será enviada uma matriz por parâmetro para o método *exibirDados*.

Este método irá receber a matriz e por meio de uma estrutura de repetição, irá exibir os dados que estão armazenados nesta Matriz.

O método `GetLength` é utilizado para verificar a quantidade de linhas (0) e colunas (1) da matriz passada por parâmetro.

```
static void Main(string[] args)
{
    int l, c;
    const int QTD_COL = 3;
    const int QTD_LIN = 5;
    string[,] nomes = new string[QTD_LIN, QTD_COL];

    String n = "Fulano ";
    int cont = 1;
    //Atribuição de nomes na matriz
    for (l = 0; l < QTD_LIN; l++)
        for (c = 0; c < QTD_COL; c++)
            nomes[l, c] = n + cont++;

    Console.WriteLine("\n *** Nomes cadastrados na Matriz ***\n");
    exibirDados(nomes);
    Console.ReadKey();
}

static void exibirDados(string[,] m)
{
    //Exibição dos dados da matriz
    for (int i = 0; i < m.GetLength(0); i++)
    {
        for (int j = 0; j < m.GetLength(1); j++)
        {
            Console.Write("{0}\t", m[i, j]);
        }
        Console.WriteLine();
    }
}
```

Exemplo 10 – Passando matriz por parâmetros - Resultado

Esta matriz contém 15 nomes, que foram gerados automaticamente, usando uma string *n* com texto “Fulano” e a cada iteração, uma variável *cont* servindo de contador, foi concatenando um número a essa string.

```
file:///C:/Aulas C#/Aula4_Exemplo10/Aula4_Exemplo10/bin/Debug/Aula4_Exemplo10.EXE

*** Nomes cadastrados na Matriz ***

Fulano 1      Fulano 2      Fulano 3
Fulano 4      Fulano 5      Fulano 6
Fulano 7      Fulano 8      Fulano 9
Fulano 10     Fulano 11     Fulano 12
Fulano 13     Fulano 14     Fulano 15
-
```

Exemplo 11 – Método que retorna Matriz

Neste exemplo serão enviadas por parâmetro a quantidade de linhas e colunas da matriz para ser realizado o cadastro de notas de alunos (método *cadastrarNotas*) e na sequência serão exibidas as informações cadastradas na matriz (método *exibeDados*).

No método principal (Main) apenas realizada a chamada de outros métodos, não existe processamento neste método, o que torna o código fonte mais “limpo”/“claro” em relação as etapas que devem ser executadas.

```
static void Main(string[] args)
{
    const int QTD_COL = 3;
    const int QTD_LIN = 5;
    double[,] notas = new double[QTD_LIN, QTD_COL];

    notas = cadastrarNotas(QTD_LIN, QTD_COL);

    Console.WriteLine("\n *** Notas cadastradas na Matriz ***\n");
    exibirDados(notas);
    Console.ReadKey();
}
```


Exemplo 11 – Método que retorna Matriz (*continuação*)

O método *cadastrarNotas* será responsável por realizar a leitura das 3 notas para cada aluno e ao final retorna uma matriz com todas as notas cadastradas.

```
static double[,] cadastrarNotas(int nLin, int nCol)
{
    double[,] n = new double[nLin, nCol];
    //Atribuição de notas na matriz
    for (int l = 0; l < nLin; l++)
    {
        Console.WriteLine("\nNotas do {0}º aluno", l + 1);
        for (int c = 0; c < nCol; c++)
        {
            Console.Write(" {0}º nota: ", c + 1);
            n[l, c] = double.Parse(Console.ReadLine());
        }
    }
    return n;
}
```

Exemplo 11 – Método que retorna Matriz (*continuação*)

O método *exibirDados* será responsável por exibir todos os dados cadastrados na matriz.

```
static void exibirDados(double[,] nt)
{
    //Exibição dos dados da matriz
    Console.WriteLine("\nAluno \t\tNota1 \t\tNota2 \t\tNota3 \n");
    for (int i = 0; i < nt.GetLength(0); i++)
    {
        Console.Write("{0}ºAluno: ", i + 1);
        for (int j = 0; j < nt.GetLength(1); j++)
        {
            Console.Write("\t{0:N1}\t", nt[i, j]);
        }
        Console.WriteLine();
    }
}
```

Exemplo 11 – Método que retorna Matriz - Resultado

```

file:///C:/Aulas C#/Aula4_Exemplo11/Aula4_Exemplo11/bin/Debug/Aula4_Exemplo11.EXE

Notas do 1º aluno
1º nota: 3,5
2º nota: 5,6
3º nota: 9,3

Notas do 2º aluno
1º nota: 5
2º nota: 9,5
3º nota: 2,5

Notas do 3º aluno
1º nota: 10
2º nota: 10
3º nota: 10

Notas do 4º aluno
1º nota: 7,5
2º nota: 4,5
3º nota: 2,5

Notas do 5º aluno
1º nota: 6,3
2º nota: 8,1
3º nota: 0,3

*** Notas cadastradas na Matriz ***

Aluno          Nota1          Nota2          Nota3
1ºAluno:       3,5            5,6            9,3
2ºAluno:       5,0            9,5            2,5
3ºAluno:       10,0           10,0           10,0
4ºAluno:       7,5            4,5            2,5
5ºAluno:       6,3            8,1            0,3
    
```

Exemplo 12 – Gerando números aleatórios na Matriz

Neste exemplo serão gerados números aleatórios entre 0 e 50 e serão armazenados em uma matriz. Logo em seguida o usuário deverá tentar adivinhar um número que está em uma determinada posição da matriz. A posição linha x coluna será escolhida aleatoriamente.

```
static void Main(string[] args)
{
    Random random = new Random(); //Objeto responsável por geração de números aleatórios

    const int QTD_COL = 3;
    const int QTD_LIN = 3;
    int[,] matriz = new int[QTD_LIN, QTD_COL];

    matriz = cadastrarNumeros(QTD_LIN, QTD_COL);

    Console.WriteLine("\n-----MEMORIZE-----\n");
    exibirDados(matriz);

    Thread.Sleep(5000); //aguarda 5 segundos
    Console.Clear(); //limpa a tela
}
```

Exemplo 12 – Gerando números aleatórios na Matriz (*continuação*)

Continuação do método *Main*.

```
int sorteioLinha = random.Next(0, QTD_LIN);    //gera número aleatório de 0 até QTD_LIN que no caso é 3
int sorteioColuna = random.Next(0, QTD_COL);    //gera número aleatório de 0 até QTD_COL que no caso é 3

Console.WriteLine("\nQual número consta na posição: linha {0}, coluna {1} da matriz: ", sorteioLinha, sorteioColuna);
int num = int.Parse(Console.ReadLine());

//método verifica passa por parâmetro o valor que está em uma posição da matriz e o número digitado pelo usuário,
//sendo que retornará true ou false
if (verifica(matriz[sorteioLinha, sorteioColuna], num) == true)
    Console.WriteLine("\nParabéns! Você acertou!\n");
else
    Console.WriteLine("\nQue pena! Você errou! Número correto seria: {0}\n", matriz[sorteioLinha, sorteioColuna]);

exibirDados(matriz);
Console.ReadKey();
}
```

Exemplo 12 – Gerando números aleatórios na Matriz (*continuação*)

Método *cadastrarNumeros*, responsável por gerar números aleatórios e armazenar na matriz..

Método *verifica*, responsável por testar se o número de uma determinada posição da matriz é igual ao número digitado pelo usuário

```
static int[,] cadastrarNumeros(int nLin, int nCol)
{
    int[,] n = new int[nLin, nCol];
    Random random = new Random();
    //Atribuição de números na matriz
    for (int l = 0; l < nLin; l++)
        for (int c = 0; c < nCol; c++)
            n[l, c] = random.Next(0, 50);

    return n;
}

static Boolean verifica(int numeroMatriz, int numUsuario)
{
    return numeroMatriz == numUsuario ? true : false;

    /* ou usando if
    if(numeroMatriz == numUsuario
        return true;
    else
        return false;
    */
}
```

Exemplo 12 – Gerando números aleatórios na Matriz (*continuação*)

Método *exibirDados*, responsável por exibir todos elementos armazenados na matriz.

```
static void exibirDados(int[,] n)
{
    //Exibição dos dados da matriz
    for (int i = 0; i < n.GetLength(0); i++)
    {
        for (int j = 0; j < n.GetLength(1); j++)
        {
            Console.Write("{0:D2}\t", n[i, j]);
        }
        Console.WriteLine();
    }
}
```


Exemplo 12 – Gerando números aleatórios na Matriz - Resultado

Apresentação dos resultados, levando em consideração o acerto do número sorteado.

```
file:///C:/Aulas C#/Aula4_Exemplo12/
-----MEMORIZE-----
19      03      21
02      26      33
05      12      05
-
```

```
file:///C:/Aulas C#/Aula4_Exemplo12/Aula4_Exemplo12/bin/Debug/Aula4_Exemplo12.EXE
Qual número consta na posição: linha 1, coluna 0 da matriz: 02
Parabéns! Você acertou!
19      03      21
02      26      33
05      12      05
-
```

Exemplo 12 – Gerando números aleatórios na Matriz - Resultado

Apresentação dos resultados, levando em consideração o erro do número sorteado.

```
file:///C:/Aulas C#/Aula4_Exemplo12/  
-----MEMORIZE-----  
30      42      45  
22      22      43  
24      01      06  
-
```

```
file:///C:/Aulas C#/Aula4_Exemplo12/Aula4_Exemplo12/bin/Debug/Aula4_Exemplo12.EXE  
Qual número consta na posição: linha 1, coluna 2 da matriz: 33  
Que pena! Você errou! Número correto seria: 43  
30      42      45  
22      22      43  
24      01      06
```

Bibliografia

- Manzano, José Augusto N. G., **Estudo Dirigido de Microsoft Visual C# 2010 Express. São Paulo, SP, Editora Érica, 2010.**
- MSDN, Microsoft. **Guia de Programação C#.** Disponível:
<[http://msdn.microsoft.com/en-us/library/system.threading.thread.sleep\(v=vs.71\).aspx](http://msdn.microsoft.com/en-us/library/system.threading.thread.sleep(v=vs.71).aspx)>. *Acesso em 03 abr 2013*

<<http://msdn.microsoft.com/en-us/library/system.random.aspx>>. *Acesso em 03 abr 2013*
- **Unicamp**
<<http://www.lsd.ic.unicamp.br/projetos/e-lane/introPascal/aula7.html>>. *Acesso em 02 abr 2013*