VOLTAR

Criação de Vetores e Matrizes

Apresentar os conceitos e criação de estruturas de dados do tipo vetor e matriz.

NESTE TÓPICO

Marcar tópico

Introdução

Até o momento você conheceu e trabalhou com variáveis que armazenam um valor por vez.

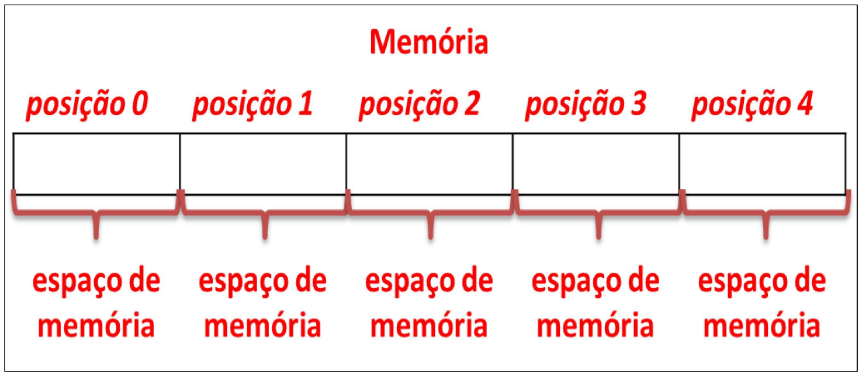
Vetores e **Matrizes** são **estruturas de dados** homogêneas, isto é, são estruturas de dados que armazenam dados do mesmo tipo. Variáveis do tipo **Vetores** e **Matrizes** são estáticas porque uma vez declaradas elas permanecem do mesmo tamanho. Elas são utilizadas para armazenar uma coleção de valores do mesmo tipo de dados.

Vetor

Vetor ou *array* é uma variável composta homogênea unidimensional, ou seja, vetor é um conjunto de variáveis de mesmo tipo, que possuem um mesmo identificador (nome) e são alocadas sequencialmente na memória do computador. Para nos referirmos a um elemento (valor) do vetor ou a uma posição no vetor, especificamos o nome do vetor e o número da posição (índice - um valor que indica cada posição específica dentro do vetor) do elemento ao qual nos referimos.

Podemos dizer que, a variável do tipo vetor é dividida em várias células, denominadas posições, cada uma identificada por índice numérico do tipo inteiro positivo. Todas as posições pertencem à mesma variável. Cada

posição tem acesso individual e independente. Veja a ilustração a seguir que representa um vetor em memória.



Declaração de Vetor

Em C# os **vetores** (**arrays**) possuem o índice iniciando em zero, portanto, o primeiro elemento do vetor possui o índice zero (0).

A forma para se declarar um vetor é: o tipo de dados, um par de colchetes ([ ]) e o nome da variável.

```
1.  // Declaração de um vetor de nome x e do tipo int
2.
3.  int[] x;
```

Alocando vetor em memória

Os vetores são variáveis e ocupam espaço em memória. No desenvolvimento devemos especificar o tipo dos elementos e usar o operador *new* para alocar o número de elementos para o vetor. Os vetores são alocados com o operador *new* porque são objetos, e todos os objetos devem ser criados com *new*. Veja a declaração e alocação de um vetor de três posições que armazenará números inteiros.

```
1.  int[] x = new int[3];
```

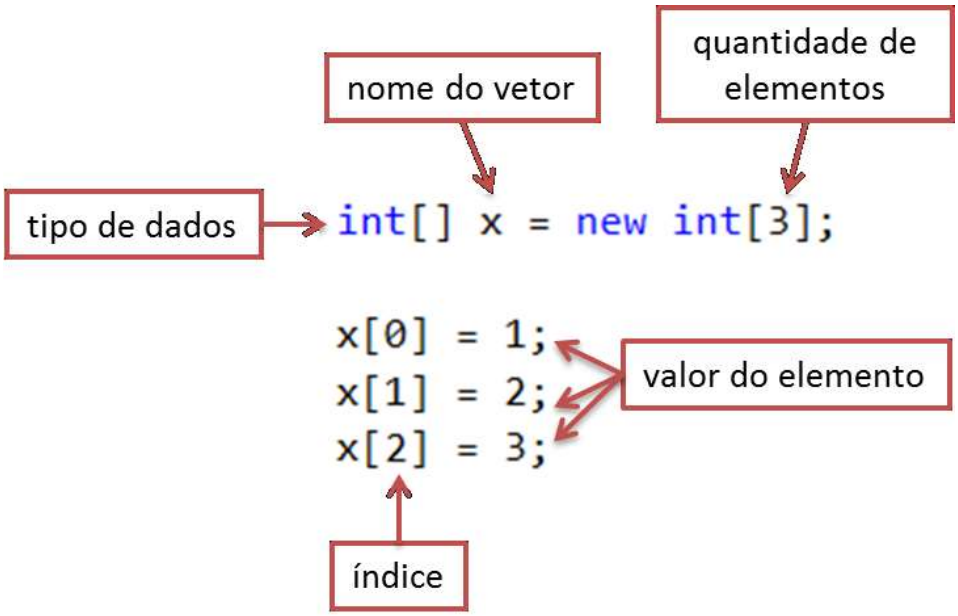
ou também podemos declarar um vetor em duas etapas:

```
1.  int[] x;
2.  x = new int[3];
```

A declaração acima aloca 3 elementos (valores) do tipo inteiro para o vetor inteiro denominado **x.**

Para atribuir ou acessar (ler) o valor do vetor fazemos referência a cada posição do vetor utilizando seu nome seguido de seu índice, delimitado pelos caracteres [ ].

Veja a figura a seguir que atribui os valores ao vetor denominado **x**.



Você também pode declarar e inicializar o vetor. Veja o exemplo comentado.

```
1.  //declarando e inicializando um vetor
2.  // a lista inicializadora especifica o número de elementos e o valor de cada element
    o
3.  int[] y = { 5, 8, 25, -32, 40 };
```

Quando declaramos e inicializamos os vetores não é necessário a utilização do operador *new*, o compilador aloca memória para o objeto quando encontra uma declaração de vetor que inclui uma lista inicializadora.

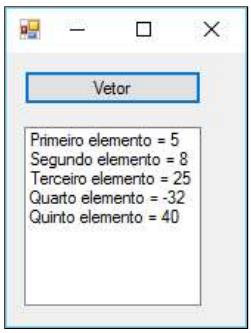
Outros exemplos de declaração de vetores:

```
1.  //vetor para armazenar 50 notas
2.  double[] notas = new double[50];
3.
4.  //vetor para armazenar 50 nomes
5.  string[] nomes = new string[50];
6.
7.  //declarando e inicializando vetores
8.
9.  double[] pesos = new double[] {75.9, 85, 45.3, 64.9, 58};
10.
11. string[] sobrenomes = {"Silva", "Gonzales", "Oliveira" };
```

Acessando e percorrendo os dados de um vetor

Como vimos anteriormente s elementos contidos nos vetores são identificados por um **índice** e é por meio do número do índice que acessamos um determinado elemento no vetor tanto para leitura quanto para atribuir valores ao vetor. Veja o exemplo a seguir que declara e inicializa um vetor e apresenta os seus elementos em um ListBox.

```
1.  //declarando e inicializando um vetor
2.  // a lista inicializadora especifica o número de elementos e o valor de cada element
    o
3.  int[] vetor = { 5, 8, 25, -32, 40 };
4.
5.  //acessando os elementos do vetor por meio do índice e apresentado seus elementos no
    ListBox
6.
7.  lstVetor.Items.Add("Primeiro elemento = " + vetor[0]);
8.  lstVetor.Items.Add("Segundo elemento = " + vetor[1]);
9.  lstVetor.Items.Add("Terceiro elemento = " + vetor[2]);
10. lstVetor.Items.Add("Quarto elemento = " + vetor[3]);
11. lstVetor.Items.Add("Quinto elemento = " + vetor[4]);
```

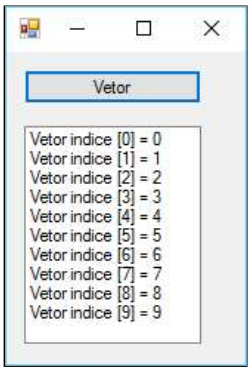


E se esse vetor tivesse 100 elementos ou mais? Geralmente utilizamos laço de repetição para acessar e manipular os elementos do vetor. No exemplo a seguir a variável *i* é utilizada para acessar o índice do vetor.

```
1.  //declarando e inicializando um vetor
2.  // a lista inicializadora especifica o número de elementos e o valor de cada element
    o
3.
4.  int[] vetor = { 5, 8, 25, -32, 40 };
5.
6.  int i;    //variável i para o índice
7.
8.  lstVetor.Items.Clear();
9.
10. for (i = 0; i < 5; i++)
11. {
12.     lstVetor.Items.Add("Elemento [" + (i + 1) + "] = " + vetor[i]);
13. }
```

Inicializando um vetor com valores gerados a partir do laço de repetição e apresentando os elementos do vetor no **ListBox**.

```
1.  //declarando um vetor de 10 posições
2.  int[] vetor = new int[10];
3.
4.  int i;    //variável i para o índice
5.
6.  lstVetor.Items.Clear();
7.
8.  for (i = 0; i < 10; i++)
9.  {
10.     //atribuindo o valor de i para o vetor no índice i
11.     vetor[i] = i;
12.
13.     lstVetor.Items.Add("Vetor indice [" + i + "] = " + vetor[i]);
14. }
```



O laço Foreach

O laço **foreach** é uma evolução do laço de repetição e pode ser usado para acessar cada elemento do vetor ou coleção.

Sintaxe do laço foreach

```
foreach (<tipo> <variavel> in <coleção>)
```

Veja o exemplo a seguir.

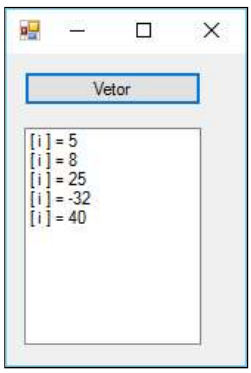
```
lstVetor.Items.Clear();

int[] vetor = { 5, 8, 25, -32, 40 };

foreach (int i in vetor)
{

    lstVetor.Items.Add("[ i ] = " + i);

}
```



Matriz

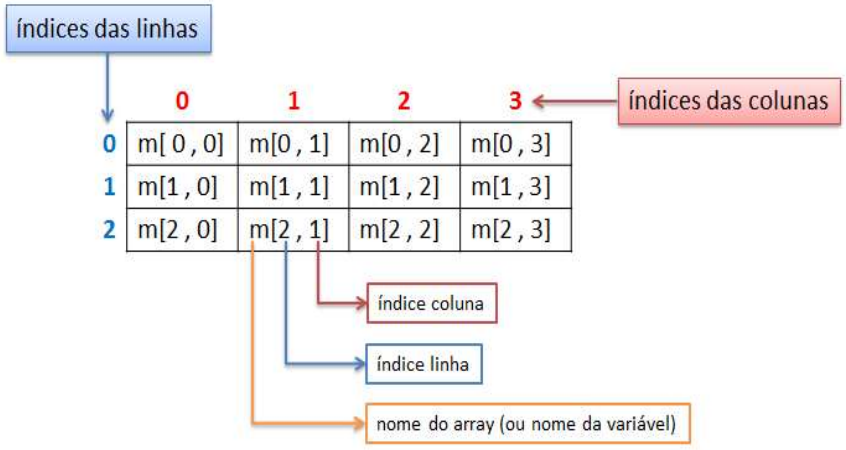
Matriz é uma variável composta homogenia multidimensional, ou seja, são **arrays** que possuem **dois ou mais índices** para identificar elementos específicos, e que armazenam uma coleção de dados do mesmo tipo.

Os **arrays** (**vetores**) que exigem **dois índices** para identificar um elemento específico são chamados de *arrays bidimensionais*, ou *vetores bidimensionais*. Existem dois tipos de arrays bidimensionais: os

retangulares e os *irregulares*. Vamos estudar os ***arrays retangulares***.

Os ***arrays retangulares*** com **dois índices** normalmente representam **matrizes** ou **tabelas** de valores e são organizados em *linhas* e *colunas*, no qual cada linha tem o mesmo tamanho e cada coluna também tem o mesmo tamanho.

Veja a ilustração a seguir que representam uma matriz denominada *m* de 3 linhas por 4 colunas.



Observe a seguinte regra:

Se **M** é uma matriz *m* x *n*, isto é, *m* é a quantidade de linhas e *n* é a quantidade de colunas, então suas linhas são indexadas de *0* a *m-1* e suas colunas de *0* a *n-1*. Sua dimensão é dada por: *m* x *n*.

Então, dada uma matriz de 3 linhas e 4 colunas, dizemos que é uma matriz com dimensão 3X4, ou seja, essa matriz armazenará **12** elementos.

Declaração de Matriz

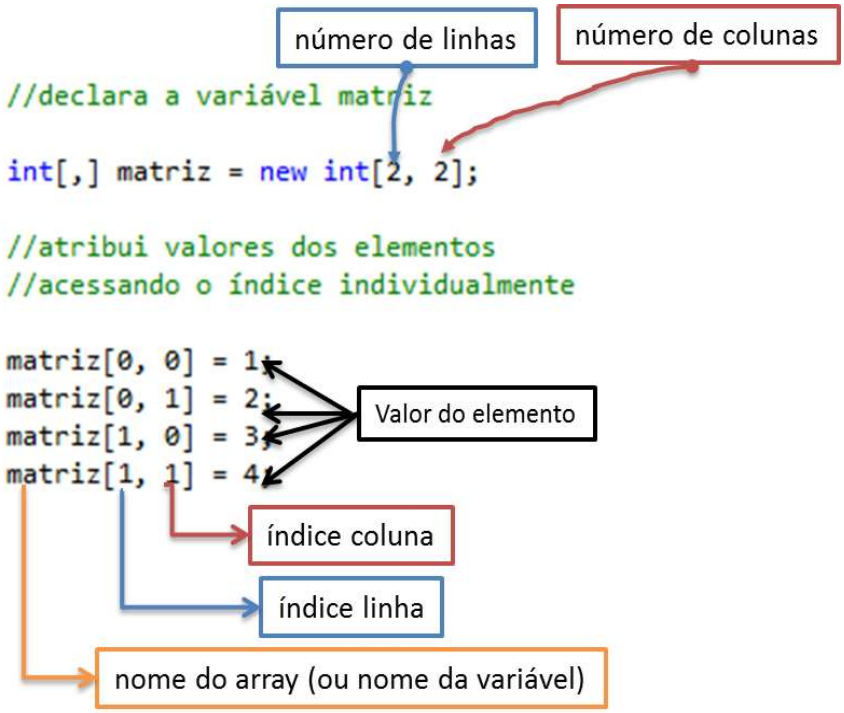
Em C# as matrizes possuem dois índices, que por convenção, o primeiro identifica a linha do elemento e o segundo identifica a coluna que iniciam em zero.

```
1.  //declara a variável matriz de duas dimensões
2.
3.  int[  ,  ] matriz = new int[2 , 2];
```

Podemos também declarar e inicializar a variável acessando cada posição individualmente.

```
1.  //declara a variável matriz com o operador new e define o seu tamanho
2.
3.  int[ , ] matriz = new int[2 , 2];
4.
5.  //atribui valores dos elementos acessando o índice individualmente
6.
7.  matriz[0 , 0] = 1;
8.  matriz[0 , 1] = 2;
9.  matriz[1 , 0] = 3;
10. matriz[1 , 1] = 4;
```

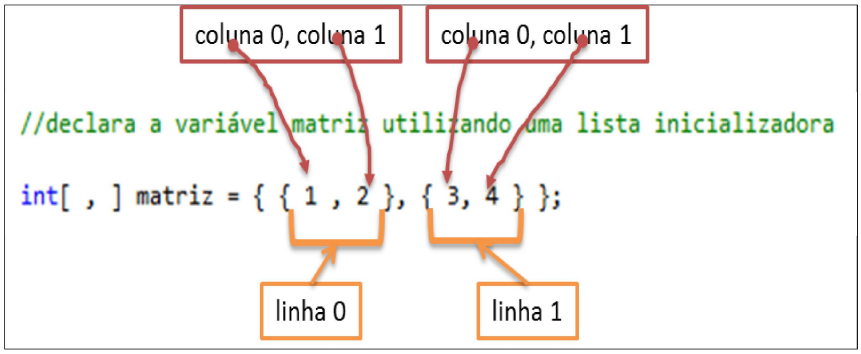
Veja a ilustração a seguir.



A declaração acima também poderia ser feita utilizando uma lista inicializadora e omitindo o operador *new*.

```
1.  //declara a variável matriz utilizando uma lista inicializadora e omite o operador n
    ew
2.
3.  int[ , ] matriz = { { 1 , 2 }, { 3, 4 } };
```

Veja a ilustração a seguir.



Também podemos declarar a variável omitindo o seu tamanho, que será definido pela lista inicializadora.

```
1.  //declara a variável nomes omitindo o tamanho da variavel e utilizando uma lista ini
    cializadora
2.
3.  string[,] nomes = new string[,] { { "João", "Antonio" }, { "Maria", "Ana" } };
```

Acessando e percorrendo os elementos da matriz

De forma geral, utilizamos as variáveis para representar linhas e colunas (dois índices) que indicam a posição do elemento na matriz, então, para uma matriz denominada *inteiros*, podemos nos referenciar da seguinte forma:

inteiros[i , j]

no qual a variável *i* representa as linhas e a variável *j* representa as colunas.

Veja o exemplo comentado a seguir que declara e inicializa uma matriz e apresenta os seus elementos em um ListBox denominado lstMatriz.

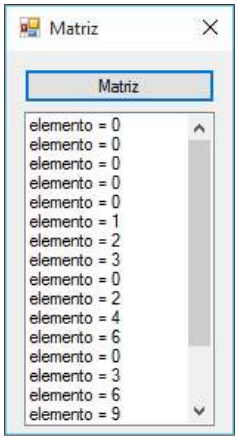
```
1.  lstMatriz.Items.Clear();
2.
3.  //variáveis para acessar as posições dos índices da matriz
4.
5.  // i - representa linha
6.  // j - representa coluna
7.
8.  int i, j;
9.
10. //declara a variável nomes omitindo o tamanho da variável e utilizando uma lista ini
    cializadora
11.
12. string[ , ] nomes = new string[ , ] { { "João", "Antonio" }, { "Maria", "Ana" } };
13.
14. for (i = 0; i < 2; i++)   //percorre as linhas
15. {
16.     for (j = 0; j < 2; j++)  //percorre as colunas
17.     {
18.         lstMatriz.Items.Add(nomes[i, j]);
19.     }
20. }
```

Veja a saída do programa.

Inicializando uma matriz com valores gerados a partir do laço de repetição e apresentando os elementos no **ListBox**.

```
1.  lstMatriz.Items.Clear();
2.
3.  //variáveis para acessar as posições dos índices da matriz
4.  // linha - representa linha
5.  // coluna - representa coluna
6.
7.  int linha, coluna;
8.
9.  //declara a variável numeros com 5 linhas e 4 colunas
10.
11. int[ , ] numeros = new int[5,4];    //dimensão da matriz 5 x 4
12.
13. for (linha = 0; linha < 5; linha++)
14. {
15.     for (coluna = 0; coluna < 4; coluna++)
16.     {
17.         numeros[linha, coluna] = linha * coluna;  //atribui o valor de linha * colun
    a
18.     }
19. }
20.
21. //exibindo os dados no ListBox
22.
23. for (linha = 0; linha < 5; linha++)
24. {
25.     for (coluna = 0; coluna < 4; coluna++)
26.     {
27.         lstMatriz.Items.Add("elemento = " + numeros[linha, coluna]);
28.     }
29. }
```

Veja a saída do programa.



Implemente os exemplos apresentados criando novos projetos.

Quiz

Exercício

Criação de Vetores e Matrizes

INICIAR

Referências

DEITEL, H. M. et al. **C# Como Programar**. São Paulo: Pearson Makron Books, 2003.

OLSSON, M. ***C# Quick Syntax Reference***. Berkeley, CA: Apress, 2013.

SITES

Microsoft <**https://msdn.microsoft.com/pt-br/library/9b9dty7d.aspx (https://msdn.microsoft.com/pt-br/library/9b9dty7d.aspx)**> Acesso em 30/10/2015

Microsoft <**https://msdn.microsoft.com/pt-br/library/2yd9wwz4.aspx (https://msdn.microsoft.com/pt-br/library/2yd9wwz4.aspx)**> Acesso em 30/10/2015

Macoratti <**http://www.macoratti.net/10/05/c_arrays.htm (http://www.macoratti.net/10/05/c_arrays.htm)**> Acesso em 30/10/2015

Avalie este tópico