# C#-11 - Collections

1. Array
2. Coleções

## 1. Array

Fornece métodos para criar, manipular, pesquisar e classificar matrizes, servindo assim como a classe base para todas as matrizes no Common Language Runtime.

### Matrizes unidimensionais

Você cria uma matriz unidimensional usando o novo operador que especifica o tipo de elemento de matriz e o número de elementos. O exemplo a seguir declara uma matriz de cinco inteiros:

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

Essa matriz contém os elementos de array[0] a array[4]. Os elementos da matriz são inicializados para o valor padrão do tipo de elemento, para inteiros.

É possível inicializar uma matriz unidirecional das seguintes formas:

```
int[] array2 = { 1, 3, 5, 7, 9 };
string[] weekDays2 = { "Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat" };
```

também é possivel declarar a variavel e instanciar seus valores depois:

```
int[] array3;
array3 = new int[] { 1, 3, 5, 7, 9 };
```

#### Retornando dados

Para retornar os dados de uma matriz é possível acessar pelo seu índice começando em 0, veja a seguir:

```
string[] weekDays2 = { "Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat" };

Console.WriteLine(weekDays2[0]);
Console.WriteLine(weekDays2[1]);
Console.WriteLine(weekDays2[2]);
Console.WriteLine(weekDays2[3]);
Console.WriteLine(weekDays2[4]);
Console.WriteLine(weekDays2[5]);
Console.WriteLine(weekDays2[6]);

/*Output:
Sun
Mon
Tue
Wed
```

```
Thu
Fri
Sat
*/
```

## Foreach com matrizes unidimensionais

A instrução **foreach** fornece uma maneira simples e limpa de iterar através dos elementos de uma matriz.

Para matrizes unidimensionais, a declaração processa elementos no aumento da ordem do índice, começando com o índice 0 e terminando com índice: **foreachLength - 1**

```
int[] numbers = { 4, 5, 6, 1, 2, 3, -2, -1, 0 };
foreach (int i in numbers)
{
    System.Console.Write("{0} ", i);
}
// Output: 4 5 6 1 2 3 -2 -1 0
```

## Matrizes multidimensionais

Arrays podem ter várias dimensões. Uma array unidimensional é chamada de vetor. Já uma array bidimensional é chamada de matriz. É possível inicializar uma matriz de diferentes maneiras, veja abaixo:

```
int[,] array2D = new int[,] {{ 1, 2 },
                             { 3, 4 },
                             { 5, 6 },
                             { 7, 8 }};

int[,] array2Da = new int[4, 2] {{ 1, 2 },
                                 { 3, 4 },
                                 { 5, 6 },
                                 { 7, 8 }};

string[,] array2Db = new string[3, 2] {{ "one", "two"},
                                       { "three", "four"},
                                       { "five", "six"}};

// Three-dimensional array.
int[,,] array3D = new int[,,] { { { 1, 2, 3 }, { 4, 5, 6 } },
                                { { 7, 8, 9 }, { 10, 11, 12 } } };
// The same array with dimensions specified.
int[,,] array3Da = new int[2, 2, 3] { { { 1, 2, 3 }, { 4, 5, 6 } },
                                      { { 7, 8, 9 }, { 10, 11, 12 } } };
```

É possível também inicializar uma matriz sem especificar a classificação:

```
int[,] array4 = { { 1, 2 }, { 3, 4 }, { 5, 6 }, { 7, 8 } };
```

também é possível declarar a variável e instanciar seus valores depois:

```
int[,] array5;
array5 = new int[,] { { 1, 2 }, { 3, 4 }, { 5, 6 }, { 7, 8 } };   // OK
```

### Retornando dados

Para retornar os dados de uma matriz é possível acessar pelo seu índice começando em 0, veja a seguir:

```
int[,] array2D = new int[,] { { 1, 2 }, { 3, 4 }, { 5, 6 }, { 7, 8 } };
string[,] array2Db = new string[3, 2] { { "one", "two" }, { "three", "four" },{
"five", "six" } };

int[,,] array3D = new int[,,] { { { 1, 2, 3 }, { 4, 5, 6 } },{ { 7, 8, 9 }, {
10, 11, 12 } } };
int[,,] array3Da = new int[2, 2, 3] { { { 1, 2, 3 }, { 4, 5, 6 } },{ { 7, 8, 9
}, { 10, 11, 12 } } };

// Accessing array elements.
System.Console.WriteLine(array2D[0, 0]);
System.Console.WriteLine(array2D[0, 1]);
System.Console.WriteLine(array2D[1, 0]);
System.Console.WriteLine(array2D[1, 1]);
System.Console.WriteLine(array2D[3, 0]);
System.Console.WriteLine(array2Db[1, 0]);
System.Console.WriteLine(array3Da[1, 0, 1]);
System.Console.WriteLine(array3D[1, 1, 2]);
```

## Foreach com matrizes multidimensionais

Para matrizes multidimensionais, os elementos são atravessados de modo que os índices da dimensão mais direita são aumentados primeiro, depois a próxima dimensão esquerda, e assim por diante à esquerda:**

```
int[,] numbers2D = new int[3, 2] { { 9, 99 }, { 3, 33 }, { 5, 55 } };
// Or use the short form:
// int[,] numbers2D = { { 9, 99 }, { 3, 33 }, { 5, 55 } };

foreach (int i in numbers2D)
{
    System.Console.Write("{0} ", i);
}
// Output: 9 99 3 33 5 55
```

No entanto, com matrizes multidimensionais, usar um loop aninhado dá-lhe mais controle sobre a ordem em que processar os elementos de matriz.

# 2. Coleções

Para muitas aplicações, você deseja criar e gerenciar grupos de objetos relacionados. Existem duas maneiras de agrupar objetos: criando matrizes de objetos e criando coleções de objetos. Arrays são mais úteis para criar e trabalhar com um número fixo de objetos fortemente digitado. As coleções fornecem uma maneira mais flexível de trabalhar com grupos de objetos. Ao

contrário das matrizes, o grupo de objetos com os seus objetos pode crescer e encolher dinamicamente à medida que as necessidades do aplicativo mudam.

## List

Representa uma lista de objetos que podem ser acessados por índice. Fornece métodos para pesquisar, classificar e modificar listas.

```
// Criando uma lista de usuarios.
List<Usuario> usuarios = new List<Usuario>();

// Adicionando usuarios na lista
usuarios.Add(new Usuario("Gustavo", 31));
usuarios.Add(new Usuario("Clauber", 12));

//Listando usuarios (fazer override de ToString)
foreach (Usuario usuario in usuarios)
{
    Console.WriteLine(usuario);
}

//Inserindo usuario na posição 2
usuarios.Insert(2, new Usuario("Kaique", 34));

//Remover pelo objeto (fazer método de Equals)
usuarios.Remove(new Usuario("Gustavo", 31));

//Remove pelo indice
usuarios.RemoveAt(1);
```

## Queue

Representa uma primeira coleção de objetos (FIFO - First in, First out), onde o primeiro a entrar é o primeiro a sair.

```
// Criando uma Fila de string.
Queue<string> numeros = new Queue<string>();
numeros.Enqueue("um");
numeros.Enqueue("dois");
numeros.Enqueue("tres");
numeros.Enqueue("quatro");
numeros.Enqueue("cinco");

//Listando numeros
foreach (String numero in numeros)
{
    Console.WriteLine(numero);
}

//Valida quantos elementos tem na fila
numeros.Count

//Pegando o primeiro da fila
numeros.Dequeue();
```

```
//Removendo o primeiro da fila
numeros.Peek();

//Verificando se a Fila tem o numero cinco
numeros.Contains("cinco");

//Limpa a Fila
numeros.Clear();
```

## Stack

Representa uma coleção de instâncias de tamanho variável (LIFO - last-in firs-out), onde o ultimo a entrar é o primeiro a sair.

```
// Criando uma Pilha de string.
Stack<string> numeros = new Stack<string>();
numeros.Push("um");
numeros.Push("dois");
numeros.Push("tres");
numeros.Push("quatro");
numeros.Push("cinco");

// Listando elementos da pilha
foreach (String numero in numeros)
{
    Console.WriteLine(numero);
}

//Valida quantos elementos tem na Pilha
numeros.Count

// Removendo de cima da pilha
numeros.Pop();

// Validando proximo elemento da pilha
numeros.Peek();

//Verificando se a Pilha tem o numero um
numeros.Contains("um");

//Limpa a Pilha
numeros.Clear();
```

## Dictionary

Representa uma coleção de chaves e valores.

```
// Define um dicionario com chamve do tipo string e conteudo string
Dictionary<string, string> dicionario = new Dictionary<string, string>();

dicionario.Add("txt", "notepad.exe");
dicionario.Add("bmp", "paint.exe");
dicionario.Add("dib", "paint.exe");
dicionario.Add("rtf", "wordpad.exe");

// Listando elementos do Dicionario
```

```
foreach( KeyValuePair<string, string> kvp in dicionario )
{
    Console.WriteLine("Chave = {0}, Valor = {1}", kvp.Key, kvp.Value);
}

// Listar chaves que contem o dicionario
Dictionary<string, string>.KeyCollection chaves = dicionario.Keys;

foreach( string chave in chaves )
{
    Console.WriteLine("Key = {0}", chave);
}

// Mostra valor contido na chamve passada
Console.WriteLine(dicionario["bmp"]);

// Redefine conteudo da chave passada
dicionario["bmp"] = "<outro conteudo>";

// Remove elemento do dicionado pelo indice
dicionario.Remove("bmp");
```