

(Parte II) - Arrays

Nesta segunda parte serão abordadas algumas estruturas de dados, nomeadamente arrays uni-dimensionais

Sumário:

Arrays uni-dimensionais:
Manipulação de Arrays - Exemplos
Exemplo Completo

Arrays

Um array (Vetor ou Matriz) é uma estrutura de dados que possibilita o armazenamento de dados de um determinado tipo. Cada valor armazenado numa unidade desse array é acedido ou indexado por um indexador do tipo inteiro, correspondente a uma posição.

Arrays uni-dimensionais:

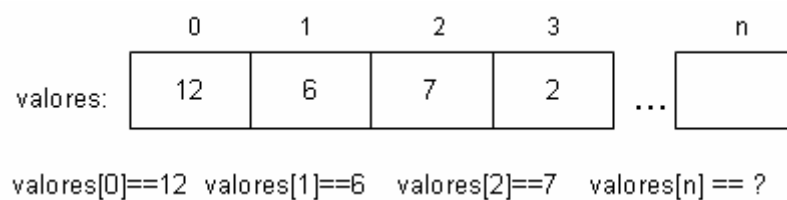


Figura 1 - Array uni-dimensional

Array que comporta uma única dimensão de dados.

Sintaxe:

*tipoDados [] nomeArray = **new** TipoDados[dimensãoArray] Classe []*

*nomeArray = **new** Classe[dimensãoArray]*

Declarar e Activar (reservar espaço com o operador **new()**)

```
double[] alturas = new double[10];  
alturas[3]=12.3;
```

```
string[] nomes = new string[3];  
nomes[0] = "lufer";
```

Declarar, inicializar e omitir o tamanho do array.

```
string[] cp = {"4900","4500","4750"};
int[] valores = {1,2,3,4,5,6,7,8};
```

Manipulação de Arrays - Exemplos

Declarações:

```
//SINTAXE:
//type[] array_name = new type[size];
//type[] array_name = {val1,val2,...,valN};

// declarar
int[] valores;
valores = new int[20];
//int[] valores = new int[20]; Equivalente

//declarar e inicializar sem operador new()
int[] notas ={12,13,10,15};

// array de strings
string[] nomes ={ "ola", "ole" };
```

Operadores e Operações:

```
//dimensão de um array: Length()
Console.WriteLine("Tamanho do array notas=" + notas.Length);

// Inserir valores num array
for (int i = 0; i < valores.Length; i++)
    valores[i] = i;
```

Percorrer um array:

```
// Inserir valores num array
for (int i = 0; i < valores.Length; i++)
    valores[i] = i;

// Mostrar todos os valores de um array
int j = 0;
while (j < valores.Length)
{
    Console.WriteLine("aux[{0}]={1}",j,valores[j]);
    j++;
}

// Mostrar todos os valores de um array com foreach
int x = 0;
foreach (int i in valores)
{
    Console.WriteLine("valores[{0}]={1}", x++, i);
}
```

Copiar arrays:

O array destino deverá ter a dimensão suficiente para conter o array origem.

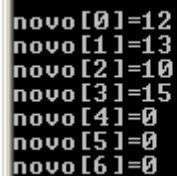
Exemplo 1:

```
// Copiar um array: copiar array notas para array novo
int[] novo=new int[notas.Length+3];
notas.CopyTo(novo, 0); //copia a partir da posição "0" do array notas

// Mostrar todos os valores de um array com foreach
//int x = 0;
x = 0;
foreach (int i in novo)
{
    Console.WriteLine("novo[{0}]={1}", x++, i);
}
```

Resultado:

Repare-se que o array novo fica somente com as quatro posições ocupadas, correspondente a todos os valores que existiam no array notas



```
novo[0]=12
novo[1]=13
novo[2]=10
novo[3]=15
novo[4]=0
novo[5]=0
novo[6]=0
```

Figura 2 - Resultado da cópia de arrays (I)

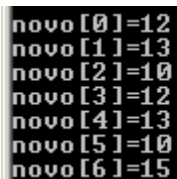
Exemplo 2:

```
notas.CopyTo(novo, 3); //insere a partir da posição "3" do array novo

// Mostrar todos os valores de um array com foreach
//int
x = 0;
foreach (int i in novo)
{
    Console.WriteLine("novo[{0}]={1}", x++, i);
}
```

Resultado

Neste caso é feita uma cópia do array notas mas inseridas a partir da posição 3 do novo array.



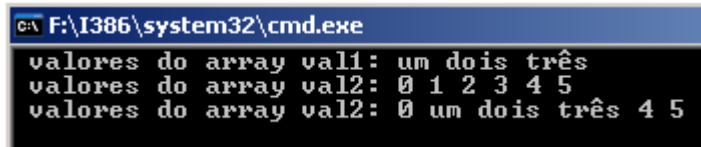
```
novo[0]=12
novo[1]=13
novo[2]=10
novo[3]=12
novo[4]=13
novo[5]=10
novo[6]=15
```

Figura 3 - Resultado da cópia de arrays (II)

Exemplo Completo

```
public class CopyToArray
{
    public static void Main()
    {
        //val1 e val2 são arrays com valores do tipo object
        object[] val1 = {"um", "dois", "três"};
        object[] val2 = {0, 1, 2, 3, 4, 5};
        Console.Write (" valores do array val1: ");
        foreach(object o in val1)
        {
            Console.Write (" {0} ", o);
        }
        Console.Write ("\n valores do array val2: ");
        foreach (object o in val2)
        {
            Console.Write (" {0} ", o);
        }
        val1.CopyTo (val2, 1);
        Console.Write ("\n valores do array val2: ");
        foreach (object o in val2)
        {
            Console.Write (" {0} ", o);
        }
        Console.WriteLine();
    }
}
```

Resultado



```
c:\ F:\I386\system32\cmd.exe
valores do array val1: um dois três
valores do array val2: 0 1 2 3 4 5
valores do array val2: 0 um dois três 4 5
```

Figura 4 – Resultado do Exemplo completo sobre arrays simples

Nota

Uma exceção (Exception) ocorre sempre que surge um erro na execução de um programa. Contudo é possível evitar que o programa pare de forma anormal na sequência desse erro. Chama-se a isto suportar Excepções. Quando a referência utilizada num array é inválida (ex: $i > \text{array.Length}$ ou $i < 0$), o C# gera a exceção `IndexOutOfRangeException`.