

Arreglos

Contenido

Pasando arreglos a los métodos

Ordenar y realizar búsquedas en arreglos

Arreglos de varias dimensiones

La estructura de repetición foreach



Pasando arreglos a métodos

- Pasar arreglos como parámetros a los métodos especificando el nombre del arreglo sin paréntesis cuadrados
- Los arreglos son pasados por referencia
- Los elementos individuales del arreglo son pasados por valor



```
using System;
using System.Collections.Generic;
using System.Text;
using System.Windows.Forms;
namespace Arreglos
{
    class PasarArreglo
    {
        static void Main(string[] args)
        {
            int[] a = { 1, 2, 3, 4, 5 };

            Console.WriteLine("Efectos de pasar el arreglo entero" +
                "llamada por referencia\n\nLos elementos del arreglo " +
                "original son \n\t");

            for (int i = 0; i < a.Length; i++)
                Console.WriteLine(" {0} ", a[i]);

            ModificarArreglo(a); // el arreglo es pasado por referencia
            Console.WriteLine("\n\nLos valores del arreglo modificado son:\n\t");

            // despliega los elementos del arreglo a
            for (int i = 0; i < a.Length; i++)
                Console.WriteLine(" {0} ", a[i]);
            Console.WriteLine("\n\nEfectos de pasar un elemento " +
                "del arreglo, llamada-por-valor \n\n" +
                "a[ 3 ] antes de ModificarElemento: {0} ", a[3]);

            // elemento del arreglo pasado en llamada-por-valor
            ModificarElemento(a[3]);

            Console.WriteLine("\na[ 3 ] despues de ModificarElemento: {0} ", a[3]);
            Console.ReadLine();
        }
    }
}
```

Pasar un arreglo a un método (1)



Outline

Pasar un arreglo a un método (2)

```
// método que modifica el arreglo que recibe,
// se modifica el arreglo original
public static void ModificarArreglo(int[] b)
{
    for (int j = 0; j < b.Length; j++)
        b[j] *= 2;
}
// método que modifica el entero pasado a este
// el original no será modificado
public static void ModificarElemento(int e)
{
    Console.WriteLine("\nvalor recibido en ModificarElemento: {0}", e);
    e *= 2;
    Console.WriteLine("\nvalor calculado en ModificarElemento: {0}", e);
}
}
```

Efectos de pasar el arreglo entero llamada por referencia

Los elementos del arreglo original son

1 2 3 4 5

Los valores del arreglo modificado son:

2 4 6 8 10

Efectos de pasar un elemento del arreglo, llamada-por-valor

a[3] antes de ModificarElemento: 8

valor recibido en ModificarElemento: 8

valor calculado en ModificarElemento: 16

a[3] despues de ModificarElemento: 8

Ordenar y realizar búsquedas en arreglos

- .NET Framework incluye capacidades de ordenación y búsqueda de arreglos
 - Clase Array
 - Array.Sort
 - Ordena los elementos del arreglo
 - Array.IndexOf
 - Busca un elemento en el arreglo y regresa la posición de la primera ocurrencia, regresa -1 si no lo encuentra
 - Array.BinarySearch
 - Realiza una búsqueda binaria en un arreglo ordenado, mas eficiente que IndexOf



Outline

Ordenación y búsqueda en arreglos (1)

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace Arreglos02
{
    public partial class frmArreglos : Form
    {
        int[] arreglo = new int[14];
        public frmArreglos()
        {
            InitializeComponent();
        }

        // Procedimiento que muestra un arreglo en un cuadro de texto
        public void MostrarArreglo(int[] a, TextBox txt)
        {
            txt.Clear();
            for(int i=0; i<arreglo.Length; i++)
                txt.Text += string.Format("[{0}] = {1}\r\n", i, arreglo[i]);
        }

        private void btnOrdenar_Click(object sender, EventArgs e)
        {
            Array.Sort(arreglo); // Ordena un arreglo unidimensional
            MostrarArreglo(arreglo, txtOrdenado);
        }
    }
}
```

Outline

Ordenación y búsqueda en arreglos (2)

```
private void btnGenerar_Click(object sender, EventArgs e)
{
    // Llena el arreglo de numeros enteros aleatorios entre 0 y 100

    Random aleatorio = new Random();
    for(int i=0; i<arreglo.Length; i++)
        arreglo[i] = aleatorio.Next(0, 100);
    MostrarArreglo(arreglo, txtOriginal);
}

private void btnBuscar_Click(object sender, EventArgs e)
{
    // efectua una búsqueda lineal en el arreglo
    int pos=Array.IndexOf(arreglo, int.Parse(txtBuscar.Text));
    if (pos < 0)
        lblRes.Text = "Número encontrado";
    else
        lblRes.Text = string.Format("El {0} esta en la posición [{1}]",
            arreglo[pos], pos);
}
}
```

frmArreglos

Arreglo Original	Arreglo Ordenado
[0] = 38	[0] = 21
[1] = 70	[1] = 23
[2] = 74	[2] = 34
[3] = 34	[3] = 35
[4] = 36	[4] = 36
[5] = 52	[5] = 38
[6] = 21	[6] = 46
[7] = 92	[7] = 52
[8] = 60	[8] = 52
[9] = 35	[9] = 58
[10] = 52	[10] = 60
[11] = 46	[11] = 70
[12] = 58	[12] = 74
[13] = 23	[13] = 92

Buscar

38

El 38 esta en la posición

Buscar Elemento

Generar Ordenar

Arreglos de varias dimensiones

- Requiere dos o mas índices para identificar un elemento en particular
- Arreglos rectangulares
 - Con frecuencia representan tablas en la cuales cada renglón es del mismo tamaño y cada columna es del mismo tamaño
 - Por convención el primer índice identifica elementos del reglón y el segundo índice identifica elementos de la columna
- Arreglos dentados (Jagged Arrays)
 - Arreglos de arreglos
 - Los arreglos que compomen a los arreglos dentados pueden ser arreglos de diferentes tamaños



Arreglos de varias dimensiones

	Columna 0	Columna	Columna 2	Columna 3
Renglón 0	a[0, 0]	a[0, 1]	a[0, 2]	a[0, 3]
Renglón 1	a[1, 0]	a[1, 1]	a[1, 2]	a[1, 3]
Renglón 2	a[2, 0]	a[2, 1]	a[2, 2]	a[2, 3]

Diagram illustrating the structure of a 2D array (matrix) with 3 rows and 4 columns. The array is labeled 'a'.

The diagram shows the following structure:

- Columns: Columna 0, Columna, Columna 2, Columna 3
- Rows: Renglón 0, Renglón 1, Renglón 2

The elements are represented as $a[\text{row index}, \text{column index}]$. For example, the element at row 0, column 1 is $a[0, 1]$.

Labels and arrows indicate the components of the array notation:

- Nombre del arreglo (Name of the array): 'a'
- Índice de renglón (Row index): The first index (e.g., 0, 1, 2)
- Índice columna (Column index): The second index (e.g., 0, 1, 2, 3)

Arreglo con doble con tres renglones y cuatro columnas





Outline

Arreglo Dos Dimensiones

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Windows.Forms;
namespace Arreglos
{
    class ArreglosDosDimensiones
    {
        static void Main(string[] args)
        {
            // declaración e inicialización de un arreglo rectangular
            int[,] arreglo1 = new int[,] { { 1, 2, 3 }, { 4, 5, 6 } };

            // declaración e inicialización de arreglos dentados
            int[][] arreglo2 = new int[3][];
            arreglo2[0] = new int[] { 1, 2 };
            arreglo2[1] = new int[] { 3 };
            arreglo2[2] = new int[] { 4, 5, 6 };
            Console.WriteLine("\nValores en arreglo1 por renglón son\n");

            // output values in arreglo1
            for (int i = 0; i < arreglo1.GetLength(0); i++)
            {
                for (int j = 0; j < arreglo1.GetLength(1); j++)
                {
                    Console.Write(" {0} ", arreglo1[i, j]);
                }
                Console.WriteLine();
            }
            Console.WriteLine("\nValores en arreglo2 por renglón son\n");

            // salida de los elementos en el arreglo2
            for (int i = 0; i < arreglo2.Length; i++)
            {
                for (int j = 0; j < arreglo2[i].Length; j++)
                {
                    Console.Write(" {0} ", arreglo2[i][j]);
                }
                Console.WriteLine();
            }
            Console.ReadLine();
        }
    }
}
```

Valores en arreglo1 por renglón son

```
1 2 3
4 5 6
```

Valores en arreglo2 por renglón son

```
1 2
3
4 5 6
```



Outline

Arreglo Doble (1)

```
using System;
using System.Collections.Generic;
using System.Text;
namespace Arreglos
{
    class ArregloDoble
    {
        static int[][] califs;
        static int estudiantes, examenes;
        static void Main(string[] args)
        {
            califs = new int[3][];
            califs[0] = new int[] { 77, 68, 86, 73 };
            califs[1] = new int[] { 96, 87, 89, 81 };
            califs[2] = new int[] { 70, 90, 86, 81 };
            estudiantes = califs.Length; // numero de estudiantes
            examenes = califs[0].Length; // numero de examenes

            // relleno de linea de encabezados
            Console.Write(" ");

            // imprime encabezados de columnas
            for (int i = 0; i < examenes; i++)
                Console.Write(" [{0}] ", i);

            // imprime renglones
            for (int i = 0; i < estudiantes; i++)
            {
                Console.WriteLine("\ncalifs[{0}] ", i);
                for (int j = 0; j < examenes; j++)
                    Console.Write("{0} ", califs[i][j]);
            }

            Console.WriteLine("\n\nCalificación más baja: {0}\n", Minima());
            Console.WriteLine("Calificación más alta: {0} \n\n", Maxima());

            for (int i = 0; i < estudiantes; i++)
                Console.WriteLine("El promedio del estudiante {0} es {1} ",
                    i, Promedio(califs[i]));
            Console.ReadLine();
        }
    }
}
```

Outline



Arreglo Doble (2)

// encuentra la calificación más baja en el arreglo

```
static public int Minima()
```

```
{
    int califMinima = 100;
    for (int i = 0; i < estudiantes; i++)
        for (int j = 0; j < examenes; j++)
            if (califs[i][j] < califMinima)
                califMinima = califs[i][j];
    return califMinima;
}
```

// encuentra la calificación más alta en el arreglo

```
public static int Maxima()
```

```
{
    int califMaxima = 0;
    for (int i = 0; i < estudiantes; i++)
        for (int j = 0; j < examenes; j++)
            if (califs[i][j] > califMaxima)
                califMaxima = califs[i][j];
    return califMaxima;
}
```

// determina el promedio para un estudiante en particular

```
public static double Promedio(int[] conjuntoCalifs)
```

```
{
    int total = 0;
    for (int i = 0; i < conjuntoCalifs.Length; i++)
        total += conjuntoCalifs[i];
    return (double)total / conjuntoCalifs.Length;
}
}
```

	[0]	[1]	[2]	[3]
califs[0]	77	68	86	73
califs[1]	96	87	89	81
califs[2]	70	90	86	81

Calificación más baja: 68

Calificación más alta: 96

El promedio del estudiante 0 es 76

El promedio del estudiante 1 es 88.25

El promedio del estudiante 2 es 81.75

La estructura de repetición foreach

- Es utilizada para iterar através de valores en las estructuras de datos tales como arreglos
- No tiene contador
- Se utiliza una variable para representar el valor de cada elemento





Outline

For Each

```
using System;
using System.Collections.Generic;
using System.Text;
namespace Arreglos
{
    class ForEach
    {
        static void Main(string[] args)
        {
            int[,] arregloCalifs = { { 77, 68, 86, 73 },
                                      { 98, 87, 89, 81 }, { 70, 90, 86, 81 } };

            int califBaja = 100;
            foreach (int grade in arregloCalifs)
            {
                if (grade < califBaja)
                    califBaja = grade;
            }

            Console.WriteLine("La calificación mas baja es {0}", califBaja);
            Console.ReadLine();
        }
    }
}
```

La calificación mas baja es 68