Fundamentos de Sistemas de Información

Arreglos

Logro de sesion

Al finalizar la sesión, el alumno implementa aplicaciones visuales, en C#, haciendo uso de la Plataforma Microsoft.NET **y** de Arreglos como estructura de datos.

Arreglos

En C# podemos trabajar con Arreglos definiéndolos de la siguiente forma:

<tipo de dato> [] **Nombre_del_arreglo**;

Nombre_del_arreglo = new **<tipo de dato>** [<cantidad>];

Pero una mejor alternativa es hacerlo con la clase **Array** que viene en el .Net Framework

Clase Array

Características:

- ✓ Se encuentra definida en el namespace System.
- ✓ Tiene métodos predefinidos para crear, insertar, buscar y ordenar los elementos de un arreglo.
- ✓ Trabaja con cualquier tipo de dato.
- ✓ Crea arreglos de una o varias dimensiones.

Clase Array – Creando una instancia

Para crear una instancia de esta clase debemos de hacer lo siguiente:

Array **arreglo**=Array.CreateInstance **(typeof(<tipo de dato>), elementos)**

Ejemplo:

```
//Arreglo de entero de 30 elementos
Array mi_arreglo= Array.CreateInstance(typeof(Int32),30);
// Arreglo de Char de 15 elementos
Array mi_arreglo= Array.CreateInstance(typeof(char),15);
```

Clase Array – Recorrido del Arreglo

Para recorrer un Array, es necesario conocer lo siguiente:

- La dimensión del arreglo que deseamos recorrer.
- El primer índice de la dimensión y
- El último índice de la dimensión.

Clase Array – Recorrido del Arreglo

Por lo tanto necesitamos conocer que:

- ✓ Las dimensiones de un arreglo empiezan en **0**
- ✓ **GetLowerBound(<dimensión>)**: Retorna el primer índice de la dimensión indicada
- ✓ **GetUpperBound(<dimensión>)**: Retorna el último índice de la dimensión indicada.

Clase Array – Recorrido del Arreglo

Ejemplo 1: Arreglo

```
Array arr= Array.CreateInstance(typeof(Int32),30);
for(int i=arr.GetLowerBound(0); i<=arr.GetUpperBound(0);i++)
```

Ejemplo 2: Matrices

```
Array mat= Array.CreateInstance(typeof(Int32),30,10);
for ( int i=mat.GetLowerBound(0); i <= mat.GetUpperBound(0);i++)
   for ( int j = mat.GetLowerBound(1); j <= mat.GetUpperBound(1); j++)
```

Clase Array – Insertar un elemento

Para insertar un nuevo elemento del arreglo debemos de utilizar el método **SetValue** definido en la clase Array

<nombre_arreglo>. **SetValue (<valor>,<índice>);**

Clase Array – Insertar un elemento

Ejemplo 1: Arreglo

```
Array arr= Array.CreateInstance(typeof(Int32),30);
for(int i=arr.GetLowerBound(0); i<=arr.GetUpperBound(0);i++)
        arr.SetValue(5,i);
```

Ejemplo 2: Matrices

```
Array mat= Array.CreateInstance(typeof(Int32),30,10);
for ( int i=mat.GetLowerBound(0); i <= mat.GetUpperBound(0);i++)
  for ( int j = mat.GetLowerBound(1); j <= mat.GetUpperBound(1); j++)
      mat.SetValue(5,i,j);
```

Clase Array – Obtener el valor de un elemento

Para conocer el valor de un elemento del arreglo debemos de utilizar el método **GetValue**, definido en la clase Array

<nombre_arreglo>. **GetValue (<índice>);**

Clase Array – Obtener el valor de un elemento

Ejemplo 1: Arreglo

```
Array arr= Array.CreateInstance(typeof(Int32),30);
for(int i=arr.GetLowerBound(0); i<=arr.GetUpperBound(0);i++)
        Console.WriteLine(arr.GetValue(i));
```

Ejemplo 2: Matrices

```
Array mat= Array.CreateInstance(typeof(Int32),30,10);
for ( int i=mat.GetLowerBound(0); i <= mat.GetUpperBound(0);i++)
  for ( int j = mat.GetLowerBound(1); j <= mat.GetUpperBound(1); j++)
      Console.WriteLine(mat.GetValue(i,j));
```

Clase Array – Usando Foreach

- ✓ A parte de utilizar for para recorrer los elementos de un arreglo, podemos utilizar la instrucción **foreach.**

Pero debemos tener en cuenta que:

- ✓ Solo tendría sentido utilizar el foreach si deseamos obtener los elementos del arreglo.
- ✓ Foreach no maneja los índices del arreglo.

Clase Array – Usando Foreach

De manera genérica sería:

Array **arr**= Array.CreateInstance(typeof(<**tipo de dato**>), <cantidad>);

//Asumiendo que el arreglo ya tiene valores

foreach(<**tipo de dato**> Nombre in **arr**)

Clase Array – Usando Foreach

Ejemplo:

```
Array arr= Array.CreateInstance(typeof(Int32),30);

//Asumiendo que el arreglo ya tiene valores
foreach(Int32 mi_variable in arr)
        Console.WriteLine(mi_variable);
```

Clase Array – Cantidad de elementos

Nombre_del_Arreglo.GetLength(dimensión)

Retorna el número total de elementos del arreglo en una dimensión indicada.

El valor obtenido debe ser asignado a una variable para no perderlo.

Clase Array – Buscar un elemento

Array.Find((Tipo_dato []) nombre_arreglo, método_comparación)

Find es una función definida en la clase Array.

Recibe como parámetro:

- ✓ El nombre de una variable del tipo Array. Mire el typecasting.
- ✓ Función que define la condición para encontrar el valor.

Retorna:

- ✓ El primer elemento que cumpla con la condición.

Clase Array – Buscar un elemento

```
private bool Comparacion(Int32 valor)
 {    return valor > 30 && valor < 40;      }
public void Ejemplo_Find()
 {
     Array m_arr = Array.CreateInstance(typeof(System.Int32), 5);
     m_arr.SetValue(10, 0);
     m_arr.SetValue(20, 1);
     m_arr.SetValue(35, 2);
     m_arr.SetValue(40,3);
     m_arr.SetValue(50, 4);
     int dato=Array.Find((Int32[])m_arr,Comparacion );
     Console.WriteLine(dato);
     Console.ReadKey();
 }
```

Clase Array – Buscar varios elementos

Array.FindAll((Tipo_dato []) nombre_arreglo,método_comparación)

FindAll es una función definida en la clase Array.

Recibe como parámetro:

- ✓ El nombre de una variable del tipo Array. Mire el typecasting
- ✓ Función que define la condición para encontrar el valor.

Retorna un arreglo con los elementos que cumplen la condición.

Clase Array – Buscar varios elementos

```
private bool Comparacion(Int32 valor)
 {    return valor > 30 && valor < 40;     }
public void Ejemplo_FindAll()
 {
    Array m_arr = Array.CreateInstance(typeof(System.Int32), 5);
    m_arr.SetValue(10, 0);
    m_arr.SetValue(20, 1);
    m_arr.SetValue(35, 2);
    m_arr.SetValue(40,3);
    m_arr.SetValue(50, 4);
    Int32[] dato=Array.FindAll((Int32[])m_arr,Comparacion );
    Console.WriteLine(dato);
    Console.ReadKey();
  }
```

Clase Array – Existencia de un elemento

Array.Exists((Tipo_dato [])nombre_arreglo,método_comparación)

Exists es una función definida en la clase Array.

Recibe como parámetro:

- ✓ El nombre de una variable del tipo Array. Mire el typecasting
- ✓ Función que define la condición para encontrar el valor.

Retorna True si el dato existe dentro del arreglo, en caso contrario False.

Clase Array – Existencia de un elemento

```
private bool Comparacion(Int32 valor)
 {   return valor == 30;    }
public void Ejemplo_Exists()
 {   Array m_arr = Array.CreateInstance(typeof(System.Int32), 5);
     m_arr.SetValue(10, 0);
     m_arr.SetValue(20, 1);
     m_arr.SetValue(35, 2);
     m_arr.SetValue(40,3);
     m_arr.SetValue(50, 4);
     bool resultado=Array.Exists((Int32[])m_arr,Comparacion );
     Console.WriteLine(resultado);
     Console.ReadKey();
}
```

Clase Array – Ordenar los datos

Array.Sort(nombre_arreglo,objeto_de_Clase_que_implementa_Compare)

Sort es una función definida en la clase Array.

Recibe como parámetro:

- ✓ El nombre de un Array.
- ✓ Objeto de la clase que implementa el método compare de la interfaz ICompare.

Resultado Array ordenado.

Clase Array – Ordenar los datos

```
Using System.Collections;

public class Metodo_Comparacion : IComparer
  {
     int IComparer.Compare(Object x, Object y)
      {
         return (Int32)x == (Int32)y ? 0 : (Int32)x > (Int32)y ?
1 : -1;
      }
  }
  ……….……
```

```
void Ejemplo_Sort(string[] args)
 {
    Array m_arr =
    Array.CreateInstance(typeof(System.Int32), 5);
    IComparer  Comparar=new Metodo_Comparacion();
          m_arr.SetValue(10, 4);
          m_arr.SetValue(20, 3);
          m_arr.SetValue(35, 2);
          m_arr.SetValue(40,1);
          m_arr.SetValue(50, 0);
           Array.Sort(m_arr,Comparar );
          foreach (Int32 dato in m_arr)
             Console.WriteLine(dato);
}
```