Clase 11

Lunes, 2 Octubre 2023

Indice

Colecciones

- Las **colecciones** son agrupaciones de elementos. Pueden almacenar en su interior cualquiera de los tipos básicos.
- Las colecciones pueden se clasifican según su funcionamiento: Arrays, Listas, Diccionarios, Árboles
- Es importante recordar que las **colecciones se almacenan en una variable de tipo número, que apunta a una direccion de memoria.**
  Es en este lugar reservado de la memoria donde se guarda la información sobre la colección y sus elmentos.

Listas

- Son las colecciones más importantes.
- Su funcionamiento es similar al de un objeto, creado a partir de la clase LIST.
- El objeto LIST se crea en el momento que hacemos **NEW**. Esto quiere decir que la asignación a una variable no crea el objeto en sí mismo.
- Ejemplo de creación de lista

```
'
public void CreateList()
{
    List<int> list;            // se crea una variable llamada 'l'
                               // de tipo 'Lista de enteros'
    list = new List<int>();    // se crea la lista en una direccion de la memoria
    list = null;               // l apunta a la nada,
                               // se destruye la lista del programa
}
'
```

- Metodos que hacen funcionar una lista

```
'
public void UseList()
{
    Add(element)     // añade 'element' al final de la lista

    list.Add(40);
    list.Add(-10);
    list.Add(3);
                           // list: [40,-10,3]

    Remove(element)  // elimina el objeto 'element' de la lista
    RemoveAt(index)  // elimina el elemento que se encuentra
                           // en la posición 'index'

    l.RemoveAt(2);
                           // list: [40,-10]

    l[i] = value     // Actualiza el valor del elemento 'i' de una lista

    l[0] = 60;
    int i = 1;
    l[i + 0] = 3;
                           // list: [60,3]

    l.Count          // Hace el conteo del número de elementos
                           // que hay en una lista

    int n = l.Count;
                           // n = 2

    l.Insert(index, element)   // Inserta 'element' en la posicion 'index'
```

```
    l.Insert(1, -20);
                            // list: [60,-20,3]

    l[1] = l[2]      // el elemento de la posicion 1 es igual
                            // al elemento de la posicion 2

                            // list: [60,3,3]

    l.Clear()        // Se eliminan todos los elementos de la lista.
                            // Vacia la lista

                            // list: [0]
}
'
```

- Las ventajas de una **LISTA** es que pueden contener tantos elementos como necesitemos en su interior, esto es, no tienen límite de capacidad. Son más sencillas y ágiles de trabajar que el resto de colecciones. Su principal uso es para tratar con datos que se almacenan y se destruyen de manera dinámica.

Arrays

- Son colecciones de elementos, rígidas y estáticas, del tipo que se indique en su definición. Tienen la característica particular que no permiten cambiar el tamaño de la colección, ni añadir o eliminar elementos posteriormente a su creación. De hecho, es obligatorio definir su tamaño en el momento que creamos el objeto.

```
'
public void CreateArrays()
{
int [] array;           // Se crea la variable 'a' que es del tipo 'array de int'
array = new int[4];     // Se crea un array en una posición de la memoria
                        // donde apunta el puntero 'a'
                        // Es Imprescindible especificar el número de celdas
                        // que contiene el array.
                        // Este número no se puede cambiar.

                        // array: [0,0,0,0]
array[3] = -10;
array[0] = a[3];
                        // array: [-10,0,0,-10]

int n = array.Length;
                        // n = 4
int [] b = array;
                        // b = [-10,0,0,-10]
                        // b, del tipo 'array de int',
                        // apunta a la misma dirección que a.
}
'
```

Funciones de ejemplo

- **FUNCIÓN:** Se le pasa una lista de strings y devuelve el número de elementos que hay en su interior.

```
'
public class ListExample
{
    public static int GetListItems(List<string> list)
    {
        return list.Count;
    }
}
'
```

- 'list' es solo el puntero que apunta una dirección de la RAM donde se almacenan los datos particulares. Realmente es un número entero, por ejemplo 380.000. Esto quiere decir que hasta que no hacemos el 'NEW' no estamos creando ninguna lista

- **FUNCIÓN:** Se le pasa una lista de dobles y devuelve el número de elementos que son positivos.

```
'
public static int GetPositiveListItems(List<double> list)
{
    int result = 0;
    for(int i = 0; i < list.Count; i++)
    {
        if(list[i] > 0)
            result++;
    }
    return result;
}
'
```

- **FUNCIÓN:** Se le pasa un array de dobles y devuelve el número de elementos que son positivos.

```
'
public static int GetPositiveArrayItems(double[] array)
{
    int result = 0;
    for(int i = 0; i < array.Length; i++)
    {
        if(array[i] > 0)
            result++;
    }
    return result;
}
'
```

- La ventaja de un **ARRAY** es que permanecen inmutables a los cambios y definen una estructura muy rígida y estable de agrupación de datos. Son útiles para cálculos matemáticos y otras operaciones que mantengan los datos inmutables.