Clase 12

Martes, 3 Octubre 2023

Indice

Inicializadores de Listas

- Podemos declarar e inicializar las Listas/Arrays de diferentes maneras:
-

```
1.-
'
public static void Main()
{
    List<int> list;
    list = new List<int>();

    list.Add(1);
    list.Add(5);
    list.Add(10);
}
'
```

-

```
2.-
'
public static void Main()
{
    List<int> list = new List<int>();

    list.Add(1);
    list.Add(5);
    list.Add(10);
}
'
```

-

```
3.-
'
public static void Main()
{
    List<int> list = new List<int>{1, 5, 10};
}
'
```

-

```
4.-
'
public static void Main()
{
    List<int> list = new List<int>
    {
        1,
        5,
        10
    };
}
'
```

Inicializadores de Arrays

-

```
1.-
'
public static void Main()
{
    int[] list;
    list = new int[3];
```

```
    list[0] = 1;
    list[1] = 5;
    list[2] = 10;
}
'
```

- ```
  2.-
  '
  public static void Main()
  {
      int[] list = new int[3];

      list[0] = 1;
      list[1] = 5;
      list[2] = 10;
  }
  '
  ```

- ```
  3-
  '
  public static void Main()
  {
      int[] list = new int[]{1, 5, 10};
  }
  '
  ```

- ```
  4.-
  '
  public static void Main()
  {
      int[] list = new int[]
      {
      1,
      5,
      10
      };
  }
  '
  ```

Funciones que se utilizan habitualmente en Listas y Arrays

- **Función 1:** Crear una función que se le pase una lista de enteros y un valor. Devuelve verdadero o falso si el valor está dentro de la lista.
- Es buena idea utilizar el nombre de *'Contains'* para evaluar si una lista contiene un elemento.
- ```
  '
  public class Functions
  {
      public static bool ContainsNumber(List<int> list, int number)
      {
          if( list == null || list.Count == 0)
              return false;

          for(int i = 0; i < list.Count; i++)
          {
              if (list[i] == number)
                  return true;
          }
          return false;
      }
  }
  '
  ```

- Es aconsejable **validar los parámetros de entrada** de la función. Hay que comprobar que la lista no apunte a null y que no esté vacía.

- Queda reservado el uso de la función Remove(). Será motivo de suspenso su uso.
- Podemos hacer 'folding' con snippets de código siempre que queramos simplificar el código escrito. Para ello empleamos #region y #endregion

- **Funcion 2:** Se le pasa una lista de enteros y te devuelve el valor mayor.

```
'
public class ListExample
{
    public static int GetMajor(List<int> list)
    {
        if( list == null || list.Count == 0)
            return int.MinValue;

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

- **Funcion 3:** Función que devuelve la posición del valor mayor.

```
'
public class ListExample
{
    public static int GetMajor(List<int> list)
    {
        if( list == null || list.Count == 0)
            return -1;

        int aux = list[0];
        int index = 0;
        for(int i = 0; i <& list.Count; i++)
        {
            if(list[i] > aux)
            {
                index = i;
                aux = list[i];
            }
        }
        return index;
    }
}
'
```

- Los valores de índice se establecen por convención como *index*.
- Podemos llamar a la función anterior para buscar el valor mayor, pero tendríamos el problema de recorrer dos *for* distintos, con el doble de carga para el programa. Es preferible crear un único for que resuelva este problema.

- **Funcion 4:** Función que devuelve verdadero o falso si una lista está ordenada

```
'
public class ListExample
{
    public static bool IsOrdered(List<int> list)
    {
        if (list == null || list.Count == 0)
            return false;

        bool result = true;

        for(int i = 0; i < list.Count - 1; i++)
        {
```

```
            if (list[i] > list[i + 1])
                result = false;
        }
        if(list[list.Count - 1] > list[list.Count])
            result = false;
        return result;
    }
}
'
```

- **Funcion 5:** Función que ordena los valores de una lista de manera *ascendente*.

```
'
public class ListExample
{
    public static List<int> Sort(List<int> list)
    {
        if (list == null || list.Count == 0)
            throw new Exception("Error de validación de parámetros");

        List <int> result = new List <int>();
        int aux;

        for(int i = 0; i < list.Count - 1; i++)
        {
            for(int j = list.Count - 1; j > 0; j--)
            {
                if (list[j] < list[j - 1])
                    {
                        aux = list[j - 1];
                        list[j - 1] = list[j];
                        list[j] = aux;
                    }
            }
        list.Add(list[i]);
        }
        return result;
    }
}
'
```