

Bloque2: Programación con VS.NET (C#)

Colecciones en C# (Introducción)



```
//ArrayList
Console.WriteLine("ArrayList");
ArrayList arrayList = new ArrayList();
arrayList.Add("hola1");
arrayList.Add("hola2");
arrayList.Add("hola3");
arrayList.Add("hola4");
arrayList.Add("hola5");
arrayList.Add("hola6");
arrayList.Add("hola7");
arrayList.Add("hola8");
arrayList.Add("hola9");
```



C# → Uso de COLECCIONES:

- Sirven para manipular grandes cantidades de datos.
- El Framework.NET provee de varias estructuras de datos, llamadas COLECCIONES, adaptadas a diferentes tipos de situaciones:
 - Almacenamiento de datos dispares y desordenados.
 - Almacenamiento de datos en base a su tipo.
 - Almacenamiento de datos por nombre, etc.
- La clase Array (que hemos utilizado hasta el momento) se puede considerar como un tipo de colección.

C# → ¿Dónde están las COLECCIONES?

- Las clases que permiten gestionar las colecciones **se agrupan en dos espacios de nombres** (namespaces):
 - System.Collections
 - System.Collections.Generic

System.Collections Namespace

El espacio de nombres [System.Collections](#) contiene interfaces y clases que definen varias colecciones de objetos, como listas, colas, matrices de bits, tablas hash y diccionarios.

System.Collections.Generic Namespace

El espacio de nombres [System.Collections.Generic](#) contiene interfaces y clases que definen colecciones genéricas, lo que permite que los usuarios creen colecciones fuertemente tipadas para proporcionar una mayor seguridad de tipos y un rendimiento mejor que los de las colecciones no genéricas fuertemente tipadas.

C# → ¿Qué tipo de COLECCIÓN selecciono?

- Es necesario identificar las necesidades relativas a la colección de objetos que debemos representar para escoger un tipo u otro:
 - Si es necesario acceder por un índice a un elemento de la colección, optemos por un **Array**, un **ArrayList** o un **List<T>**.
 - Si deseo almacenar parejas del tipo “clave única & elemento” utilizo una **Hashtable** o un **dictionary<TKey,TValue>**.
 - Si deseo acceder a los elementos en el mismo orden en que se han insertado en la colección, uso un **Queue** o **Queue<T>** (Cola).
 - Si deseo acceder a los elementos en orden inverso, utilizo una **Stack** o **Stack<T>** (Pila).

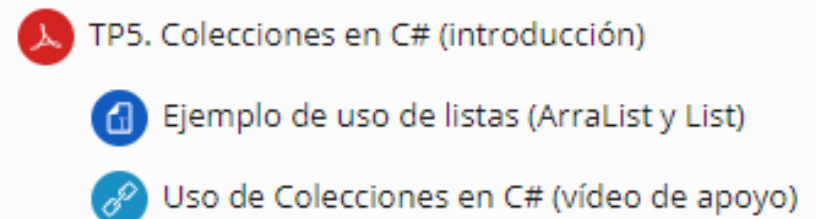



Las clases **ArrayList** y **List<T>** en C#

ArrayList Clase

- Son evoluciones de la clase **Array** y aportan ciertas mejoras:
 - El tamaño de un objeto **ArrayList** o un **List<T>** es dinámico y se ajusta en función de las necesidades.
 - Estas clases aportan métodos para agregar, incluir y eliminar varios elementos.
 - Durante su uso es posible obtener el número de elementos de un **ArrayList** o **List<T>** utilizando la propiedad **Count**.

ANÁLISIS

Ejemplo de funcionamiento de un
ArrayList / List<T>

- 
-  TP5. Colecciones en C# (introducción)
 -  Ejemplo de uso de listas (ArraList y List)
 -  Uso de Colecciones en C# (vídeo de apoyo)

Las clases **Stack** y **Stack<T>** en C#

- Las pilas (stack) siguen el principio LIFO (Last In, First Out).
- Analogía para representar este objeto (**una pila de platos**):
 - Cada plato que se agrega se sitúa en lo alto de la pila y cuando se desea tomar un plato de la pila, se coge el que hay encima de todos los demás. El último plato insertado es el primero en salir de la pila.
 - Tiene tres operaciones principales:

Push → Agrega un elemento en lo alto de la pila.

Peek → Toma el elemento de lo alto de la pila sin eliminarlo de la colección.

Pop → Recupera el último elemento de la pila eliminándolo de la colección.

Clases Stack y Stack<T> en C#

```
using System;
using System.Collections;
public class SamplesStack {

    public static void Main() {

        // Creates and initializes a new Stack.
        Stack myStack = new Stack();
        myStack.Push("Hello");
        myStack.Push("World");
        myStack.Push("!");

        // Displays the properties and values of the Stack.
        Console.WriteLine( "myStack" );
        Console.WriteLine( "\tCount:    {0}", myStack.Count );
        Console.Write( "\tValues:" );
        PrintValues( myStack );

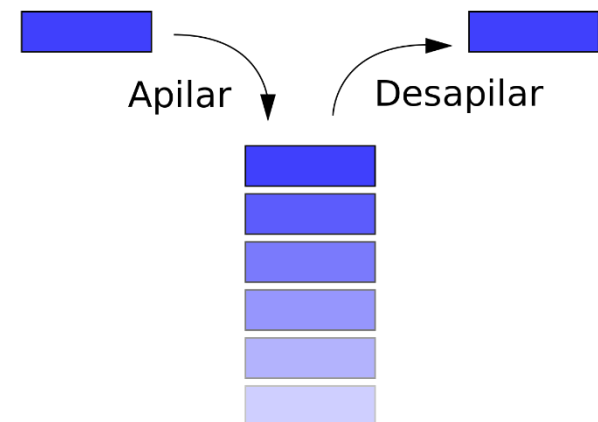
    }

    public static void PrintValues( IEnumerable myCollection ) {
        foreach ( Object obj in myCollection )
            Console.Write( "    {0}", obj );
        Console.WriteLine();
    }

}
```

Stack Class

Espacio de nombres: `System.Collections`



```
/*
This code produces the following output.

myStack
    Count:    3
    Values:    !    World    Hello
*/
```

Clases Queue y Queue<T> en C#

- Las colas (queue) siguen el principio FIFO (First In, First Out).
- Analog a para representar este objeto (**una cola de espera**):
 - La primera persona que llega y entra es la primera en salir de la cola y ser atendida. Con ello, el primer elemento agregado ser  el primero en salir de la cola.
 - Sus tres operaciones (m todos) son principales:

Enqueue → Agrega un elemento al final de la cola.

Peek → Toma el primer elemento de la cola sin eliminarlo de la colecci n.

Dequeue → Recupera el primer elemento de la cola elimin ndolo de la colecci n.

Clases Queue y Queue<T> en C#

```
using System;
using System.Collections;
public class SamplesQueue {

    public static void Main() {

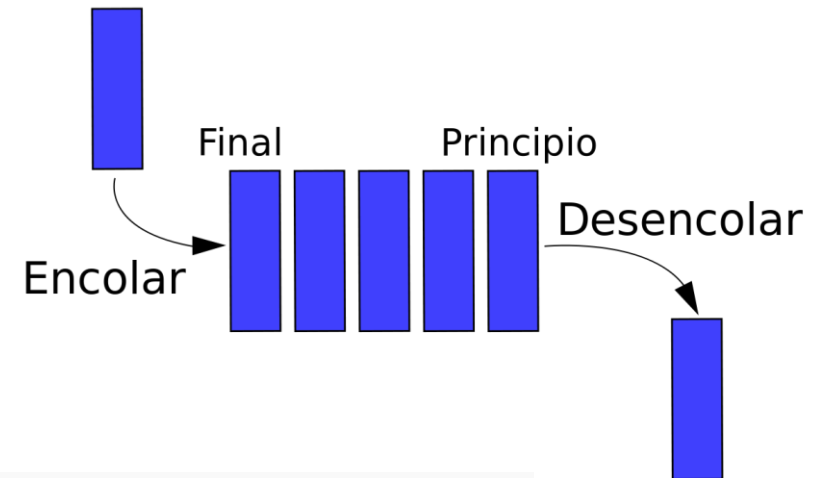
        // Creates and initializes a new Queue.
        Queue myQ = new Queue();
        myQ.Enqueue("Hello");
        myQ.Enqueue("World");
        myQ.Enqueue("!");

        // Displays the properties and values of the Queue.
        Console.WriteLine( "myQ" );
        Console.WriteLine( "\tCount:    {0}", myQ.Count );
        Console.Write( "\tValues:" );
        PrintValues( myQ );
    }

    public static void PrintValues( IEnumerable myCollection ) {
        foreach ( Object obj in myCollection )
            Console.Write( "    {0}", obj );
        Console.WriteLine();
    }
}
```

Queue Class

Espacio de nombres: `System.Collections`







```
/*
This code produces the following output.

myQ
    Count:    3
    Values:   Hello    World    !
*/
```

Realiza el ejercicio 16 de la AP7 utilizando una Lista (ArrayList o List<T>)

EJERCICIO 16. Programa que lea temperaturas obtenidas en observatorios meteorológicos. Al finalizar las lecturas, debe informar de cuál ha sido la temperatura máxima y mínima correspondiente. El programa finalizará cuando introduzcamos como temperatura el siguiente valor '99'.

Solución disponible en

-  TP5. Colecciones en C# (introducción)
-  Ejemplo de uso de listas (ArraList y List)
-  Ejercicio 16 de la AP7 con ArraList / List
-  Uso de Colecciones en C# (vídeo de apoyo)

Ejercicio 16 de la AP7 utilizando una Lista (ArrayList o List<T>)

```
static void Main(string[] args)
{
    ArrayList temperatura = new ArrayList();
    int temp, tempMin, tempMax, promedio=0;

    // Introducci n de elementos en la lista.
    do
    {
        Console.WriteLine("Introduce una temperatura (99 para finalizar): ");
        temp = int.Parse(Console.ReadLine());
        if (temp!=99)
            temperatura.Add(temp);
    } while (temp != 99);

    // Recorrido y c lculo de temperatura m nima y m xima
    tempMin = (int) temperatura[0]; tempMax = (int) temperatura[0];
    foreach (int valor in temperatura)
    {
        if (valor > tempMax) tempMax = valor;
        if (valor < tempMin) tempMin = valor;
        promedio += valor;
    }
    promedio /= temperatura.Count;

    Console.WriteLine("\nEl promedio de temperaturas es {0:N}", promedio);
    Console.WriteLine("M xima: {0} ---- M nima: {1}", tempMax, tempMin);
}
```

Practicando con Colecciones (VIII)

```
using System;
using System.Collections;

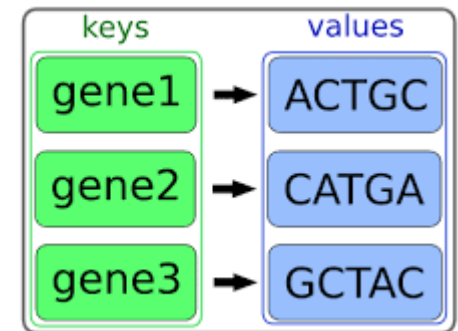
namespace Ejemplo_HashTable
{
    0 referencias
    class Program
    {
        0 referencias
        public static void Main()
        {
            // Creamos un diccionario e insertamos datos
            Hashtable miDiccio = new Hashtable();
            miDiccio.Add("byte", "8 bits");
            miDiccio.Add("pc", "personal computer");
            miDiccio.Add("kilobyte", "1024 bytes");

            // Mostramos algún dato
            Console.WriteLine("Cantidad de palabras en el diccionario: {0}",
                miDiccio.Count);
            if (miDiccio.Contains("pc"))
                Console.WriteLine("El significado de PC es: {0}",
                    miDiccio["pc"]);
            else
                Console.WriteLine("No existe la palabra PC");
            Console.ReadKey();
        }
    }
}
```

Ejemplo
disponible en
plataforma

Ejemplo de uso de colección Hashtable (Diccionario)

dict



key	value
firstName	Bugs
lastName	Bunny
location	Earth