Colas y Stacks

Representa una colección de objetos de tipo primero en entrar, primero en salir.

Ejemplos

El ejemplo siguiente muestra cómo crear y agregar valores a un Queue y cómo imprimir sus valores.

```
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();
   }
}
/*
This code produces the following output.

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

Comentarios

Esta clase implementa una cola como una matriz circular. Los objetos almacenados en un Queue se insertan en un extremo y se quitan de la otra.

Importante

No se recomienda que utilice el Queue clase para el nuevo desarrollo. En su lugar, se recomienda que use el tipo genérico Queue<T> clase.

Las colas y pilas son útiles cuando se necesita almacenamiento temporal de información; es decir, cuando desee descartar un elemento después de recuperar su valor. Use Queue si necesita acceder a la información en el mismo orden que se almacena en la colección. Use Stack si necesita acceder a la información en orden inverso.

Use ConcurrentQueue<T> o ConcurrentStack<T> si necesita tener acceso a la colección desde varios subprocesos simultáneamente.

Se pueden realizar tres operaciones principales en un Queue y sus elementos:

- Enqueue Agrega un elemento al final de la Queue.
- Dequeue Quita el elemento más antiguo desde el principio de la Queue.
- Peek Devuelve el elemento más antiguo que está al principio de la Queue pero no se quita de la Queue.

La capacidad de un Queue es el número de elementos de la Queue puede contener. Cuando se agregan elementos a un Queue, automáticamente se aumenta la capacidad según sea necesario mediante la reasignación. La capacidad puede reducirse mediante una llamada a TrimToSize.

El factor de crecimiento es el número por el que se multiplica la capacidad actual cuando se requiere una mayor capacidad. El factor de crecimiento se determina cuando el Queue se construye.El factor de crecimiento predeterminado es 2.0. La capacidad de la Queue aumentarán siempre en al menos un mínimo de cuatro, sin tener en cuenta el factor de crecimiento. Por ejemplo, un Queuecon un factor de crecimiento de 1.0 siempre aumentará en capacidad por cuatro cuando se requiere una mayor capacidad.

Queue acepta null como un valor válido y permite elementos duplicados.