Listas en c#

C# tiene un amplio rango de clases para lidiar con listas. Implementando la interfaz de iList y la implementación más popular es la lista genérica, normalmente referida como List<T> La "T" especifica el tipo de objeto contenido en la lista, el cual tiene el beneficio añadido de que el compilador verificará y se asegurará que se agreguen únicamente objetos del tipo de la lista - en otras palabras, la lista tipo -List <Type> es segura para agregar elementos de un mismo tipo.

```
List<string> listOfStrings = new List<string>();
```

Inicializar una lista con items

```
List<string>listOfNames= newList<string>()
{
"Joh Doe",
 "Jane Doe",
 "Joe Doe"
};
```

De igual forma, declarar y añadir elementos a una matriz de tipo int[] es más eficiente que hacerlo a una de tipo List<int>, y más aún que a una de tipo System.Array.

List es peor que array, porque almacena el valor como object, y luego convierte el tipo valor a un tipo valor por referencia.

System.Array por su parte, penaliza aún más esta situación.

En términos generales, una matriz o array es recomendable cuando sabemos el número fijo de elementos que va a haber en la colección o el número máximo de elementos posibles. No es recomendable sin embargo, cuando el número de elementos máximo es desconocido o simplemente no lo sabemos, pudiendo incrementarse o no

según la ejecución de los procesos, ya que al cambiar su dimensión, copiaremos cada elemento en el nuevo array.

List<T> nos proporciona ventajas a la hora de acceder a los elementos de una colección, y aunque es menos práctica de cara al rendimiento cuando tenemos claro el número máximo de elementos, podría seguir siendo interesante su uso en esas circunstancias si las operaciones que tenemos que hacer con los elementos de la colección son costosas. No obstante, en el caso de desconocer el número máximo o fijo de elementos, es mejor que array.