Limitations of array

* **Fixed size**
* **Does not have its own behaviour.**
  + Sorting (Can be done but though array as an object but not array as datatype (ie) ‘String’ = Object not ‘string’ = datatype)
* Faster to search than collection.
* If no of data is small (less than 100), then array and if its large then Collection.
* If we don’t want it to be fixed, then go for collection.
* **Arrays are homogenous**, whereas collection is heterogenous.
* Array, Lower index is 0 and Upper is -1.
* Storing and retrieving in arrays is done through loops.
* Single Multi-Dimension and Jagged Arrays.
* If u have cube, then it is 3D.
* Possible to have Uneven Arrays (Every row has different no of columns).
* Jagged array

c# is strongly typed language, which means we need to declare the type of the variable before using it.

    arrays are efficient way of storing multiple values of same type.

    faster access to elements in array

    store multiple values of same type

    drawbacks:

    not have its own behaviour as own , no fuction in collection is applicable here

    problem :they are fixed in size

    memoery is allocated as low than other data structure

    dynamic arrays is called collection

    more efficient than collection

    search in array is faster than collection

    much effient in large data so array is efficent

    if data is less than 100 then array is better

    if data is more than 100 then collection is better

    not sure no of elements then collection is better

    arrays are homogenius, collection are hetrogenius

    lower index is 0

    upper index is size-1

    index is the position of the element in the array

    storing and retrieving through loop

    single, multi dimensional, jagged array

    matrix is 2d array  ,no of rows and columns are same

    table in rdbms is 2d array

    it is not suitable for data analysis

    cubes are 3d array it consist of x,y,z

    jagged array is array of array, uneven array, every row is different no of columns.

object Array can store any type of data ,

boxing and unboxing

boxing is converting value type to reference type

unboxing is converting reference type to value type

Why Collections?

* Collection classes are special classes for data storage and retrieval.
* These classes are not fixed size.
* The capacity of the collection can dynamically grow as the items added to them increases.
* Collections can be heterogeneous or homogeneous.
* These are stacks, List, Queues and hash table.
* All collection classes implement IEnumerable interface.
* That is extended by ICollection interface.
* IDictionary and IList are also interfaces for collection which are derived from ICollection.
* Collection is of two types,
  + Dictionary
    - The objects are in memory unordered but for the computer’s convenience it has a key to find the object, like finding books in a library.
  + List
* If no of items is small then go for List, if not then go for Dictionary.
* SET like list but no redundant values like in CPP.

Interfaces Used in Collection

* IEnumerable
* IEnumerator
* ICollection
* IList
* IDictionary
* IDictionaryEnumerator
* IComparer

List Types

* ArrayList
  + Implements the IList interface using an array whose size is dynamically increased as required.
* Queue
  + Represents a first-in, first-out collection of objects.
* Stack
  + Represents a simple Last-In-First-Out (LIFO) non-generic collection of objects.
* HashTable
  + Represents a collection of key/value pairs that are organized based on the hash code of the key.
* SortedList
  + Represents a collection of key/value pairs that are sorted by the keys and are accessible by key and by index.

Iterator

* A Common algorithm to navigate through data irrespective of data striucture.
* In java it has interface Iterator.
* For C#,
  + IEnumerable
  + IEnumerator