Collection

So far we have seen that arrangement to store multiple values is done using an array

An array Is a group of values with same data type

There are some limitations with array

* Size in array is fixed
* Only values of same data type can be stored

Lets look at an example

class Stack

{

int top;

int[] arr;

public Stack()

{

top = -1;

arr = new int[10];

}

public void Push(int i)

{

top++;

arr[top] = i;

}

public int Pop()

{

return arr[top--];

}

}

class Program

{

static void Main(string[] args)

{

Stack obj = new Stack();

obj.Push(3);

obj.Push(2);

obj.Push(11);

obj.Push(4);

obj.Push(1);

obj.Push(5);

obj.Push(9);

obj.Push(6);

obj.Push(7);

obj.Push(4);

for(int i=0;i<10;i++)

{

Console.WriteLine(obj.Pop());

}

Console.ReadLine();

}

}

In the above example a class Stack is created that has an array of size 10;

Two functions push and pop are used to insert values in the array and extract values from the array

In this program values that are stored inside array are of type Integer.and not more then 10 values can be stored

Such an arrangement has a limited capacity to hold and process value.

To overcome these limitations .NET provides us with System.Collections namespace

A Collections namespace has prebuilt classes that can be used to create objects whose capacity can be increased dynamically .Also these classes can be used to create objects that can interact with different data types

Lets see an example of Stack class in System.Collections.Generic Namespace

class Program

{

static void Main(string[] args)

{

Stack<Int32> obj = new Stack<Int32>();

obj.Push(10);

obj.Push(23);

obj.Push(11);

obj.Push(15);

obj.Push(19);

foreach(int i in obj)

{

Console.WriteLine(i);

}

Console.ReadLine();

}

}

The same objective has been attained using Stack class in System.Collections namespace

Collection name space provides us with three types of collections

Kinds of Collections

Standard

Generic

Concurrent

System.Collections :-Used to create a collection of Objects such as Lists ,Queues,BitArrays,Hashtable and dictionaries

The standard collections are found under the System.Collections. They do not store elements as specifically typed objects, but as objects of type Object. Standard collections include ArrayList, Hashtable, Queue, and Stack

Classes in System.Collections

ArrayList :- Used to create a dynamic array.Implements IList interface

|  |  |  |
| --- | --- | --- |
| Function /Property | Usage | Example |
| Add | Used to add an element to arraylist | numberlist.Add(100) |
| ToArray | Used to convert an arraylist to Array object | Array array = numberlist.ToArray();  for(int i=0;i<array.Length;i++)  {  Console.WriteLine(array.GetValue(i));  } |
| Count | Returns Actual number of elements in an arraylist | Int i=numberlist.Count  Console.WriteLine(i); |
| AddRange | Used to append a range of ICollection elements at the end of ArrayList | ArrayList n = new ArrayList();  n.Add(55);  n.Add(67);  numberlist.AddRange(n); |
| BinarySearch | Used to search and return position of an element in a Sorted ArrayList | numberlist.Sort();  int i = numberlist.BinarySearch(11); |
| Sort | Used to Sort an ArrayList | numberlist.Sort() |
| Insert | Used to insert an element at an index position | numberlist.Insert(3,57) |
| Remove | Used to remove an element at the first occurrence from an ArrayList | numberlist.Remove(57) |
| Reverse | Used to reverse elements of an ArrayList | numberlist.Reverse() |

Lets see an Example

C#

class Program

{

static void Main(string[] args)

{

ArrayList numberlist = new ArrayList();

numberlist.Add(12);

numberlist.Add(23);

numberlist.Add(24);

numberlist.Add(11);

numberlist.Add(33);

foreach(var i in numberlist)

{

Console.WriteLine(i);

}

Console.ReadLine();

}

}

In the above program an ArrayList instance is created and add function is used to store values in ArrayList.A foreach loop is used to iterate through ArrayList

To retrieve value from an ArrayList,index position can be used

C#

class Program

{

static void Main(string[] args)

{

ArrayList numberlist = new ArrayList();

numberlist.Add(12);

numberlist.Add(23);

numberlist.Add(24);

numberlist.Add(11);

numberlist.Add(33);

for(int i=0;i<numberlist.Count;i++)

{

Console.WriteLine("value at position {0} is {1} ", i, numberlist[i]);

}

Console.ReadLine();

}

}

In the above example a for loops is used to iterate through arraylist .Inside loop variable I is used as index position of arraylist ,to retrieve a value

BitArray : Represent an array of bits which are represented as Booleans where 1 is true and 0 is false

Lets see an example

class Program

{

static void Main(string[] args)

{

BitArray obj = new BitArray(5);

obj[2] = true;

obj[4] = true;

for(int i=0;i<obj.Length;i++)

{

Console.WriteLine(obj[i]);

}

Console.ReadLine();

}

}

Functions inside a BitArray

|  |  |  |
| --- | --- | --- |
| Functions/Properties | Usage | Example |
| And | Used to perform and operation over two bitarrays | BitArray obj1 = new BitArray(5);    obj1[2] = true;  obj1[4] = true;  BitArray obj2 = new BitArray(5);  obj2[1] = true;  obj2[4] = true;  BitArray obj3=obj1.And(obj2);  for(int i=0;i<obj3.Length;i++)  {  Console.WriteLine(obj3[i]);  } |
| Or | Used to perform Or operation over two bit arrays | BitArray obj1 = new BitArray(5);    obj1[2] = true;  obj1[4] = true;  BitArray obj2 = new BitArray(5);  obj2[1] = true;  obj2[4] = true;  BitArray obj3=obj1.Or(obj2);  for(int i=0;i<obj3.Length;i++)  {  Console.WriteLine(obj3[i]);  } |
| Not | Used to perform Not Operation over a BitArray | BitArray obj1 = new BitArray(5);    obj1[2] = true;  obj1[4] = true;    BitArray obj3 = obj1.Not();  for(int i=0;i<obj3.Length;i++)  {  Console.WriteLine(obj3[i]);  } |

CaseInsensitiveComparer:-Used to Compare two objects ,ignoring cases for String

CaseInsensitiveComparer obj = new CaseInsensitiveComparer();

int i = obj.Compare("Hello World", "Hello World");

Console.WriteLine(i);

Hashtable:-Represents collection of Key/Value pairs organized based on hashcode of key

C#

Hashtable hashtable = new Hashtable();

hashtable.Add(1, "Apple");

hashtable.Add(2, "Banana");

hashtable.Add(3, "Orange");

hashtable.Add(4, "Pappaya");

Keys in hashtable can retrieved using property Key

C#

var keys = hashtable.Keys;

foreach(var x in keys)

{

Console.WriteLine(x);

}

Values can be retrieved from Hashtable using keys

C#

int key = Convert.ToInt32(Console.ReadLine());

Console.WriteLine(hashtable[key]);

Queue : -It is used to represent a fist in ,first out collection of objects

Functions Properties in Queue

|  |  |  |
| --- | --- | --- |
| Functions/Properties | Usage | Example |
| Enqueue | Adds a an object to the end of queue | Queue queue=new Queue()  queue.Enqueue(“First”)  queue.Enqueue(“Second”)  queue.Enqueue(“Third”) |
| Deque | Removes and returns object from the beginning of the queue | var x=queue.Deque() |

System.Collections.Generic classes are used to create collections in which values will be of same data type

A generic collection is used to implement strong typing by allowing only the desired data type to be added

Some Generic Collection Classes are

Dictionary<TKey,TValue> :Represents collection of key/value pairs that are organized based on key

Lets see an example

C#

Dictionary<Int32, String> dictionary = new Dictionary<Int32, String>();

dictionary.Add(1, "Apple");

dictionary.Add(2, "Banana");

dictionary.Add(3, "Orange");

dictionary.Add(4, "PineApple");

String value = dictionary[1];

Console.WriteLine(value);

List<T> represents a strongly typed list of objects

C#

class Program

{

static void Main(string[] args)

{

List<Int32> list = new List<Int32>();

list.Add(12);

list.Add(23);

list.Add(34);

list.Add(11);

list.Add(56);

list.Add(22);

list.Add(11);

foreach(int i in list)

{

Console.WriteLine(i);

}

Console.ReadLine();

}

}

LinkedList

This class is used to create a double LinkedList

A Linked List is a data structure in which data is captured inside a Node

A node has data as well as address for next node

Let understand working of Linked List

Data/Next Node

Data/next Node

Data /next Node

Data/next Node

In the diagram above nodes are depicted as having data and reference for next node as well as previous node .This is a doubly linked list

Lets create and use one doubly linked list using LinkedList<T> class

Some Concurrent collections are

BlockingCollection<T>, ConcurrentDictionary<T, T>, ConcurrentQueue<T>, and ConcurrentStack<T>.

Summary

Summary

* Array can store only limited values with fixed data type
* .Net provides us with Collection API to store values dynamically ,in different arrangements
* System.Collections is namespace for collections
* Different types of collections are ArrayList,BitArray,Hashtable,Queue,List
* System.Collections.Generic classes are used to create collections in which values will be of same data type

FAQ

**. What is the difference between Array and Arraylist?**

In an array, we can have items of the same type only. The size of the array is fixed when compared. To an arraylist is similar to an array, but it doesn't have a fixed size.

**What are generics in C#.NET?**

Generics are used to make reusable code classes to decrease the code redundancy, increase type safety, and performance. Using generics, we can create collection classes. To create generic collection, System.Collections.Generic namespace should be used instead of classes such as ArrayList in the System.Collections namespace. Generics promotes the usage of parameterized types.

**What are different types of collections in .NET?**

There are five important collections in .NET Arrays, Lists, Hashtable, stacks and queues.

**What are hashtable collections?**

In arraylist or array if we have to access any data we need to use the internal index id generated by the array list collection. For instance the below code snippet shows how the internal id is used to fetch data from array list.

In actual scenarios we hardly remember internal id’s generated by collection we would like to fetch the data by using some application defined key. There’s where hash table comes in to picture.

string str = MyList[1].ToString();

Hash table helps to locate data using keys as shown below. When we add data to hash table it also has a provision where we can add key with the data. This key will help us to fetch data later using key rather than using internal index id’s generated by collections.

objHashtable.Add(“p001”,”MyData”);

This key is converted in to numeric hash value which is mapped with the key for quick lookup.

**What are Queues and stack collection?**

Queues are collection which helps us to add object and retrieve them in the manner they were added. In other word queues helps us to achieve the first in first out collection behavior.

Stack collection helps us to achieve first in last out behavior.

Can you serialize hashtable and Why?

No, You can’t Serialize Hash table.Because, the .NET Framework does not allow serialization of any object that implements the IDictionary interface

### What all are the Advantages and Disadvantages of Generics in C#?

#### Advantages of Generics:

* Generics provide type safety without the overhead of multiple implementations.
* Generics eliminates boxing and unboxing.
* There is no need to write code to test for the correct data type because it is enforced at compile time. The need for type casting and the possibility of run-time errors are reduced.
* By providing strong typing, a class built from a generic lets visual studio provide IntelliSense.
* Generic collection types generally perform better for storing and manipulating value types because there is no need to box the value types
* Generic delegates enable type-safe callbacks without the need to create multiple delegate classes.

#### Disdvantages of Generics:

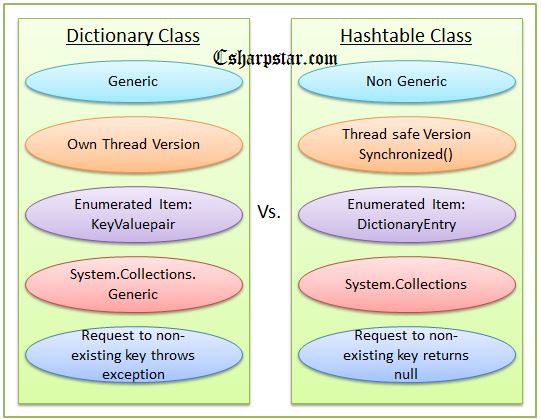
 Generic types can be derived from most base classes, such as MarshalByRefObject (and constraints can be used to require that generic type parameters derive from base classes like MarshalByRefObject). However, the .NET Framework does not support context-bound generic types. A generic type can be derived from ContextBoundObject, but trying to create an instance of that type causes a TypeLoadException.

 Enumerations cannot have generic type parameters.

 Lightweight dynamic methods cannot be generic.

 In C#, a nested type that is enclosed in a generic type cannot be instantiated unless types have been assigned to the type parameters of all enclosing types

### What all are the difference between Dictionary class and Hashtable class?

[](https://www.csharpstar.com/wp-content/uploads/2016/07/Dic_vs_Hashtable.jpg)

Case Study for Generics and Collections

Have a look at below given class diagram

Products

Productname

Price

quantity

Customer

customername

email

shipping address

billing address

Order

CartList:List<Cart>

Cart

Customer

ProductList:List<Product>

total\_amount:float

Status:true

ConfirmOrder(boolean):

Order---------->CarList----------->Cart------>Customer---->List of Products