Loops

Conditions

Break continue

Collections

Class

Break : used to exit out of a loop based on some condition

Break is used in loops and in case construct in switch

Continue : used to skip the statements of current pass of the loop and it takes you to the next iteration or pass of the loop

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Demos10Nov

{

class Program

{

static void Main(string[] args)

{

// Give me sum of some numbers of an array

int[] num = new int[10];

int sum = 0;

Console.WriteLine("Enter Elements");

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

{

num[i] = Convert.ToInt16(Console.ReadLine());

sum += num[i];

}

Console.WriteLine("Sum is " + sum);

}

}

}

// break and continue

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Demos10Nov

{

class Program

{

static void Main(string[] args)

{

// Give me sum of positive numbers of an array , and exit if number is 0

int[] num = new int[10];

int sum = 0;

Console.WriteLine("Enter Elements");

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

{

num[i] = Convert.ToInt16(Console.ReadLine());

if (num[i] == 0) break;

if (num[i] < 0) continue;

sum += num[i];

}

Console.WriteLine("Sum is " + sum);

Console.Read();

}

}

}

Advantages of Arrays:

Store n no. of values in single variable

Continuous memory allocation

// Searching : Linear , binary we can access elements thru their index

**Elements are of same type**

Limitations of Arrays

Memory is fixed at compilation time

Memory could be wasted

It is time consuming to insert or delete elements in an array

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 3 | 5 | 6 | 7 | 9 | 7 |  |  |  |

Because it requires lots of shifting of elements of elements

How do we solve this problem?? Collections

Collections are structures which stores elements.

Advantages

Their size is not fixed.

Their size is dynamic.

Memory is not wasted

Insertion / Deletion is not time consuming , we can simply call the methods

Ex: ArrayList , Stack , Queue , HashSet

Boxing , unboxing

Boxing “ Converting a variable from value type to reference type

UnBoxing “ Converting a variable from reference type to value type

Object : Base type for all the types

String is Immutable

StringBuilder is mutable

Int[] n = new int[10]

Demo

using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Demos10Nov

{

class Collections

{

static void Main()

{

Console.WriteLine("ArrayList");

ArrayList list = new ArrayList();

list.Add(1);

list.Add(2);

list.Add(3);

list.Add(4);

list.Add(5);

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

// Console.WriteLine(list[i]);

foreach(int temp in list)

Console.WriteLine(temp);

list.Insert(1, 10);

list.RemoveAt(2);

list.Remove(5);

Console.WriteLine("Stack");

// Stack LIFO

Stack stack = new Stack();

stack.Push(1);

stack.Push(2);

stack.Push(3);

stack.Pop();

foreach(int temp in stack)

Console.WriteLine(temp);

Console.WriteLine("Queue");

// Queue FIFO

Queue queue = new Queue();

queue.Enqueue(1);

queue.Enqueue(2);

queue.Enqueue(3);

queue.Dequeue();

foreach (int temp in queue)

Console.WriteLine(temp);

Console.WriteLine("HashTable");

// Hashtable

// Random Searching

//Elements are stored as Key and Value Pair

Hashtable ht = new Hashtable();

ht[1] = 20;

ht[2] = 30;

ht[3] = 89;

foreach(int temp in ht.Keys)

{

Console.WriteLine(temp);

}

Console.WriteLine("The Keys are");

foreach(int key in ht.Keys)

Console.WriteLine(key + " " + ht[key]);

Console.Read();

Console.WriteLine(ht[3]);

foreach (var temp in list)

Console.WriteLine(temp);

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

// Console.WriteLine(list[i]);

list.Add("AAAA");

list.Add(20.9);

}

}

}

Disadvantage of Collections :

Types are not same

While storing, it does Boxing

While retrieving it does unboxing

To solve this, We use generic collections

Generic collection take advantages of Arrays and collections

// ArrayList

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

list.Add(1);

list.Add(2);

list.Add(3);

It takes advantages of both Arrays and Collections

**Elements are of same type**

Their size is not fixed.

Their size is dynamic.

Memory is not wasted

Insertion / Deletion is not time consuming , we can simply call the methods

List, Stack, Queue, Dictionary