## Lecture # 4:

**Objective:**

* Arrays
* Generics
* Stack
* Queue
* List
* Array List

## Arrays:

An array in Programing Language can be defined as number of memory locations, each of which can store the same data type and which can be references through the same variable name.

An array is a collective name given to a group of similar quantities. These similar quantities could be percentage marks of 100 students, number of chairs in home, or salaries of 300 employees or ages of 25 students.

## Syntax:

DATATYPE[ ] ARRAYNAME= new DATATYPE[SIZE] ;

## Example:

int[] arrayName=new int[10];

### Saving Data in an Array:

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

{

arrayName[i] = int.Parse(Console.ReadLine());

}

### Reterving Data from an Array:

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

{

Console.WriteLine(arrayName[i]);

}

### Fuctions and Properties Associated with Arrays:

int noOfElement=arrayName.Length; int maxValue=arrayName.Max<int>(); int minValue = arrayName.Min<int>(); bool flag=arrayName.Contains(10);

# GENERICS:

Generics are the most powerful feature of C# 2.0. Generics allow you to define type-safe data structures, without committing to actual data types. This results in a significant performance boost and higher quality code, because you get to reuse data processing algorithms without duplicating type-specific code.

Using generics is a technique that enriches your programs in the following ways:

* It helps you to maximize code reuse, type safety, and performance.
* You can create generic collection classes. The .NET Framework class library contains several new generic collection classes in the *System.Collections.Generic* namespace. You may use these generic collection classes instead of the collection classes in the *System.Collections* namespace.
* You can create your own generic classes and methods.

## Generic Method Example:

using System;

using System.Collections.Generic; using System.Linq;

using System.Text;

namespace GenericTask

{

class Program

{

static void swap<T>(ref T lhs, ref T rhs)

{

T temp; temp = lhs; lhs = rhs; rhs = temp;

}

static void Main(string[] args)

{

int ilhs = 30; int irhs = 10;

string slhs = "LHS"; string srhs = "RHS";

Console.WriteLine("Left Hand Side Value before swap = {0}\nRight Hand Side Value before swap = {1}", ilhs, irhs);

swap<int>(ref ilhs, ref irhs);

Console.WriteLine("Left Hand Side Value after swap = {0}\nRight Hand Side Value after swap = {1}", ilhs, irhs);

Console.WriteLine("Left Hand Side Value before swap = {0}\nRight Hand Side Value before swap = {1}", slhs, srhs);

swap<string>(ref slhs, ref srhs);

Console.WriteLine("Left Hand Side Value after swap = {0}\nRight Hand Side Value after swap = {1}", slhs, srhs);

}

}

}

# STACK:

Working principle is FIRST IN LAST OUT. Required namespace is : System.Collections.Generic;

To add value in stack ***push()*** method is used. To remove from stack **pop()** method is used. To show top element **peek()** method is used.

***C# Code:***

# QUEUE:

Stack<string> myStack = new Stack<string>(); myStack.Push("Element 1");

myStack.Push("Element 2"); string top=myStack.Peek(); Console.WriteLine(top); Console.WriteLine(myStack.Pop()); Console.WriteLine(myStack.Pop()); myStack.Count();

top = myStack.Peek(); Console.WriteLine(top);

Working principle is FIRST IN FIRST OUT. Required namespace is: System.Collections.Generic;

Too add value in queue **enqueue()** method is used. To remove from queue **dequeue()** method is used. To show front element **peek()** method is used.

***C# Code:***

Queue<string> myQueue = new Queue<string>(); myQueue.Enqueue("E1");

myQueue.Peek(); myQueue.Dequeue(); myQueue.Count(); myQueue.Contains("e1");

# LIST:

Required namespace is: System.Collections.Generic;

List<string> newList = new List<string>(); newList.Add("This"); newList.Add(" is "); newList.Add("my first "); newList.Add("application "); newList.Add("for Generics");

foreach (string item in newList)

{

Console.WriteLine(item);

}

//newList.Clear();

bool find=newList.Contains(" is "); Console.WriteLine(find);

int noOfItem=newList.Count(); Console.WriteLine(noOfItem);

string element=newList.ElementAt(4); Console.WriteLine(element);

newList.Remove("This"); newList.RemoveAt(0); newList.Reverse();