**C#:**

.NET is a framework to develop software applications. It is designed and developed by Microsoft and the first beta version released in 2000.

C# works on the basis of .NET framework.

The software programs written in .NET are executed in the execution environment, which is called CLR (Common Language Runtime).

CLR: It is a program execution engine that loads and executes the program. It converts the program into native code

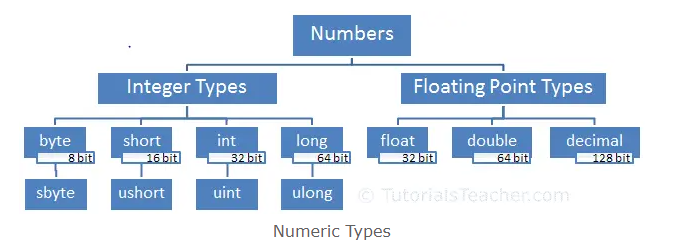
Feature of C#:

* C# is a simple language
* Object oriented programing language
* C# is a structured programming language in the sense that we can break the program into parts using functions.
* C# provides a lot of inbuilt functions that makes the development fast

# C# Data Types:

VALUE DATATYPE: 1: Predefined data types

2: User defined data types



1. Integer type numbers are positive or negative whole numbers without decimal point
2. Floating point type numbers are positive or negative whole numbers with decimal point

Short s1 = -32768;

Short s2 = 32767;

Ushort us1 = 65535;

Ushort us2 = -32000; //Compile-time

Byte b1 = 255;

Byte b2 = -128; // compile-time

Sbyte sb1 = -128;

Sbyte sb2 = 127;

long s1 = -32768;

long s2 = 32767;

Ulong us1 = 65535;

Ulong us2 = -32000; //Compile-time

int i1 = 255;

int i2 = -128;

uint sb1 = -128;// compile-time

uint sb2 = 127;

1. The uint type stores only positive numbers
2. The ushort type stores only positive numbers
3. The ulong type stores only positive numbers
4. The Byte type stores only positive numbers

**STRUCTURE:**

1. Struct is be used to hold small data values.
2. A structure is declared using struct keyword. The default modifier is internal for the struct and its members.
3. Struct is by default non static
4. EX:

struct Coordinate

{

Public int x;

Public int y;

}

Coordinate point = new Coordinate ();

;

Console. Write(point.x); // Compile time error

Point.x = 10;

point.y = 20;

Console.Write(point.x); //output: 10

Console.Write(point.y); //output: 20

**ENUM:**

1. In C#, an ENUM (or enumeration type) is used to assign constant names to a group of numeric integer values.
2. Enum is by default static.
3. EX:

enum Weekdays

{

Monday,

Tuesday,

Wednesday,

Thursday,

Friday,

Saturday,

Sunday

}

Console.WriteLine(Weekdays.Friday); //output: Friday

int day = (int) Weekdays.Friday; // enum to int conversion

Console.WriteLine(day); //output: 4

var wd = (WeekDays) 5; // int to enum conversion

Console.WriteLine(wd); //output: Saturday

**COLLECTION:** collection is an interface and its an unified architecture of classes and interface in order to overcome the drawback of arrays we will go for collection,

In Collection size is dynamic and we can store the heterogeneous type of data and also we can perform the CRUD operation on the collection

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | To store | Delete | Retrieve | Search | Sort |
| **List():** to store the value with duplicate and null ,to maintain order of insertion we use list  Memory wasted | Add() | Remove()  RemoveAt()  removeAll() | elementAt(); | Contains(); | Sort() |
| List<Object> list = new List<Object>();  list.Add("rajeev");  list.Add(1);  list.Add(2);  list.Add(3);  list.Add(4);  //to identify which element present in the perticuler index  //to check weather element present  Console.WriteLine(list.ElementAt(0));  list.Contains(1);  //to remove the element at the perticuler index  list.RemoveAt(3);  list.Remove(1);  foreach (var item in list)  {  Console.WriteLine(item);  } | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | To store | Delete | Retrieve | Search | Sort |
| **LinkedList():**store the value with duplicate and null we use list  Whenever data is added in linked list it will create a node for every data, memory optimized | Addfisrt()  Addlast()  Addafter()  Addbefore() | Remove()  RemoveAt()  removeAll() | elementAt(); | Contains(); | N/A |
| LinkedList<Object> list = new LinkedList<Object>();  list.AddLast("rajeev");  list.AddLast(1);  list.AddFirst(2);  Console.WriteLine(list.ElementAt(0));  list.Contains(1);  LinkedListNode<Object> node=list.Find(1);  list.AddAfter(node, 5);  //to remove the element at the perticuler index  list.RemoveAt(3);  list.Remove(1);  foreach (var item in list)  {  Console.WriteLine(item);  } | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | To store | Delete | Retrieve | Search | Sort |
| **Hashset<T>:**it will not allow duplicate and null value | Add() | Remove() | elementAt(); | Contains(); | N/A |
| HashSet<Object> hashSet = new HashSet<object>();  //to add the element into the set  hashSet.Add(1);  hashSet.Add(2);  hashSet.Add(4);  hashSet.Add(0);  hashSet.Add(7);  hashSet.Add(8);  //remove the element in the set  hashSet.Remove(0);  //to get the element of the perticuler posiotion  hashSet.ElementAt(4);  //to check wether the element present or not  hashSet.Contains(8);  //printing every data  foreach (var item in hashSet)  {  Console.WriteLine(item);  } | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | To store | Delete | Retrieve | Search | Sort |
| **Sortedset<T>:**it will not allow duplicate and not allow null value.  It is completely Auto sorted | Add() | Remove() | elementAt(); | Contains(); | N/A |
| SortedSet<Object> sortSet = new SortedSet<Object>();  sortSet.Add(1);  sortSet.Add(7);  sortSet.Add(3);  sortSet.Add(8);  sortSet.Add(9);  sortSet.Add(5); //remove the element in the set  hashSet.Remove(0);  //to get the element of the perticuler posiotion  hashSet.ElementAt(4);  //to check wether the element present or not  hashSet.Contains(8);  //printing every data  foreach (var item in sortSet)  {  Console.WriteLine(item);  } | | | | | |

**DATA DRIVEN TESTING:** testing application for multiple test data by using external resources like excel, data provider, etc.

**Ex1: simple hardcoded for data driven**

public class UnitTest1

{

[DataTestMethod]

[DataRow ("Rajeev", 1)]

[DataRow("naga",2)]

[DataRow("Exam",3)]

[TestCategory("DataDriven")]

[TestMethod]

public void DataDriven1(String a,int b)

{

Console.WriteLine(a + " " + b);

MessageBox.Show(a + " " + b);

}

}

**Ex2. For Dynamic data**

[TestClass]

public class DynamicDatadriven

{

public static IEnumerable<Object[]> data

{

get

{

return new[]

{

new Object[] { 1, 2 },

new Object[] { 2, 3 },

new Object[] { 3, 4 },

new Object[] { 4, 5 },

};

}

}

[TestMethod]

[TestCategory("DynamicDataDriven")]

[DynamicData(nameof(data))]

public void TestMethod3(int a, int b)

{

Console.WriteLine(a + " " + b);

MessageBox.Show(a + " " + b);

}

}

**To fetch the single data from Excel:(** dependency: Bytescout.Spreadsheet)

public static string fPath = "C:\\Users\\Rajeev\\source\\repos\\SAIPCsharp\\SAIPCsharp\\file\\dataex.xlsx";

public static string sname = "Sheet1";

[TestMethod]

[TestCategory("Read\_Singe\_Data\_From\_Excell")]

public void TestMethod1()

{

int row = 3;

int cel= 0;

Spreadsheet sp = new Spreadsheet();

sp.LoadFromFile(fPath);

string sdata=sp.Workbook.Worksheets.ByName(sname).Cell(row, cel).ToString();

MessageBox.Show(sdata);

}

**To Read Multiple Data from Excel:**

[TestMethod]

[TestCategory("Read\_multiple\_data\_from\_excell")]

public static string fPath = "C:\\Users\\Rajeev\\source\\repos\\SAIPCsharp\\SAIPCsharp\\file\\dataex.xlsx";

public static string sname = "Sheet1";

public void TestMethod2()

{

Spreadsheet sp = new Spreadsheet();

sp.LoadFromFile(fPath);

int lrow = sp.Workbook.Worksheets.ByName(sname).UsedRangeRowMax;

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

{

tring sdata = sp.Workbook.Worksheets.ByName(sname).Cell(i, 0).ToString();

Console.WriteLine("data " + i + ":" + sdata);

}

}

namespace SAIPCsharp.Practice

{

[TestClass]

public class UnitTest2

{

//to access the excell path and sheet name for multiple testmethods am using outside the method and inside the class

public static string fPath = "C:\\Users\\Rajeev\\source\\repos\\SAIPCsharp\\SAIPCsharp\\file\\dataex.xlsx";

public static string sname = "Sheet1";

//fetching the data from the excell and returning it to another method

public static IEnumerable<Object[]> dataprovide()

{

//create a object for the spreadsheet

Spreadsheet sp = new Spreadsheet();

//load the excell to that spreadsheet instance

sp.LoadFromFile(fPath);

//by using 'WORKBOOK' and 'WORKSHEET' property and 'BYNAME' method am opening the perticuler sheet

Worksheet book = sp.Workbook.Worksheets.ByName(sname);

//to fetch the last used row num

int lrow = book.UsedRangeRowMax;

//to fetch the last used cel num

int lcol = book.UsedRangeColumnMax;

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

{

for (int j = 0; j <=lcol; j++)

{

string key= sp.Workbook.Worksheets.ByName(sname).Cell(i, j).ToString();

yield return new object[] {key };

}

}

}

public static IEnumerable<Object[]> dataprovidebykeyandvalue()

{

Spreadsheet sp = new Spreadsheet();

sp.LoadFromFile(fPath);

Worksheet book = sp.Workbook.Worksheets.ByName(sname);

int lrow = book.UsedRangeRowMax;

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

{

string key = sp.Workbook.Worksheets.ByName(sname).Cell(i, 0).ToString();

string value = sp.Workbook.Worksheets.ByName(sname).Cell(i, 1).ToString();

yield return new object[] { key,value};

}

}

//here am grouping this test method

[TestCategory("DataDrivenassignment1")]

[TestMethod]

//by using DynamicData anotation am fetching the method from where am reacieving the data

[DynamicData(nameof(dataprovide), DynamicDataSourceType.Method)]

//using string paramater am recieving the data

public void TestMethod1(String data)

{

//Console.WriteLine(data);

MessageBox.Show(data);

}

[TestCategory("DataDrivenassignment2")]

[TestMethod]

[DynamicData(nameof(dataprovidebykeyandvalue), DynamicDataSourceType.Method)]

public void TestMethod2(String key,string value)

{

Console.WriteLine(key+""+value);

MessageBox.Show(key + " " + value);

}

}

}