Introduction To C#.NET

- Microsoft.Net was formerly known as Next Generation Windows Services(NGWS) .It is a completely new platform for developing the next generation of windows/web applications.

- However this new platform required a language which would take its full advantage; this is one of the factors that led to the development of C#.

- C# has evolved from C/C++.

- The #(hash symbol) in musical notations is used to refer to a sharp note and is called “Sharp”, hence the name is pronounced as C Sharp

- The C# compiler is considered to be the most efficient compiler in the .net family and a major part of the .net base class libraries itself are written in C#.

- Moreover, C# could be considered as a modern replacement for C/C++

- It is important to look at C# not just a programming language but as an integral part of the .net platform .
- Another important feature of C# is that it is a real object oriented programming language. This may not sound very exciting since we do have powerful programming languages like C++, and C# has its root in C++.
- The key point of C# however is that it enhances developer productivity and increases safety by enforcing strict type checking.
- It features a garbage collector, which relieves the programmer of the burden of manual memory management.
- C# offers extensive interoperability.

Program Flow Of C#

To understand the flow of C# program, let us begin analyzing a very simple program in C# which simply displays a message on the users screen.

```
/* This is a simple program in C# */
using System;
class First
{
    public static void Main()
    {
      Console.WriteLine (“My First C# Program”);
    }
}
```

Line 1: This is a comment in C#. Comments can be included in any part of C# program. The comment begins with “ /* ” and ends with “ */ ”. The comment can span multiple lines.

Line 2: This line using System is quite similar to the # include statement used in C/C++. The System namespace contains the classes that most applications used for interacting with the operating system. (Namespaces are nothing but a collection of classes). The classes that are used more often are the ones required for basic input/output. Note that there is a semicolon at the end of this line. All lines of code in C# must end with a semicolon just like C/C++.

Line 3: This line defines a class

Line 4: This line defines a open curly braces for a class

Line 5: Each class has one static void Main() function. This function is the entry point of a C# program.

Line 6: Next we open the scope of the method or function using curly braces

Line 7:Within the ***Main()*** function we call the ***WriteLine*** method of the ***Console*** class and pass the text “My First C# program” as its parameter. The ***WriteLine*** function displays text on the console. The ***WriteLine*** method is a part of the ***Console*** class, which in turn is a part of the ***System*** namespace.

Line 8: ***The Main()*** function is terminated using the closing braces.

Line 9: The class is terminated using the closing braces.

- ✓ Note: C# is case sensitive

  For Example: the keyword “using ” is not the same as “Using”

Basic C# Programming Concepts

- Declaring Variables In C#
- Basic Input/Output in C#
- Selection Statements in C#
- Iteration Constructs in C#
- Constructors in C#
- Destructors in C#
- Fundamentals Types of C#
- Data Types In C#
- Arrays in C#
- Structs in C#
- Enumerators In C#

- Declaring Variables In C#

  Variables in C# are declared in the following way:

  AccessModifier DataType VariableName;
  Ex: public String MyVar;

Access Modifier	Description
Public	Makes a member accessible from anywhere
Protected	Makes a member accessible within the class and all classes derived from it
Private	Makes a member accessible only within a class

- The *DataType* could be any of the variable types in C#. A few of the valid intrinsic C# datatypes are listed below

C# Data Types	Description
Int	Declares a variable which stores integer values. It takes upto 4 bytes of memory
String	Declares a variable which stores string values
Float	Declares a variable which stores integer values with decimals

- The DataType could also be an array definition or a custom data type like enumerators or even a class name for creating an object.
- The variable name should be a valid C# variable name.
- The rules for naming variables in C# are just the same as in C.
- In almost all programming languages, keywords are not allowed to be used as identifiers.
- However, C# has work around on this issue .
- You can use variables which clash with the keywords by prefixing the identifier with an @symbol
- Example: You can declare a variable like *@int* using keyword int as a variable name.

- Basic Input/Output in C#:

  Basic input and output operations are performed in C# using the methods of Console class in the System namespace.

The two methods used are

- Console.WriteLine()
- Console.ReadLIne()

Console.WriteLine() method is used for displaying text on the user screen. It can be used to format text before displaying it. It takes upto four additional parameters, which are known as the format string specifiers. The format string specifiers specify how data will be displayed and it can also serve as a placeholders, which determine where the values of the specified variable will be displayed in the string.

Console.ReadLine() method is used to get the input from the user which reads all characters upto a carriage return and returned it back as a string.

- Selection Statements In C#:

  The selection statements are used to perform operations based on the value of an expression as we have done in C programming.

  Types of selection statements used:

  - If Condition
  - If-Else Condition
  - Switch Case Construct

- Iteration Constructs In C#:

  The iteration or looping statements are used to perform certain set of instructions a certain number of times or while a specific condition is true.

  Types of iteration/looping constructs used:

  - The for loop
  - The while loop
  - The do loop
  - The foreach loop

  Except foreach loop all others are very similar to the ones in C.

- Constructors In C#:

  As we already know constructors are special methods which has the same name as the class name as in case of C++, Java.

- Destructors In C#:

  Destructors in C# are written in the same way as constructors are written. It has the same name as the class except there is a ~(tilde) symbol prefixed to it.

- Syntax Of Constructors/Destructors In C#:

```
using System;
Class MyClass
{
    public MyClass()
    {
    //My Class Constructor
    }
    public ~MyClass()
    {
    // My Class Destructor
    }
}
```

- Fundamental Types Of C#:

C # divides data types into two fundamental categories-
value types and reference types.

Most of the basic data types in C# for example (int, char) are value types.
Structures are also value types.

Reference types includes classes, interfaces , arrays and strings.

A value type represents the actual data stored on the stack, whereas reference type represents a pointer or reference to the data and are stored on the heap.

The basic difference between the two is in the way they are stored on the memory.

Boxing and Unboxing- Boxing in simple terms is nothing but conversion of a value type into a reference type. Similarly, unboxing is all about converting a reference type into a value type.

- <u>Data Types In C#</u>:

  C# provides us with an Unified Type System. This means that all the data types in C#, be it a reference type or value type are derived from just one class, the object class which means that the object class is the ultimate base class for all data types.

- <u>Arrays In C#</u>:

  Arrays are group of values of similar data type. These values are stored in contiguous memory locations, thereby ,making it easier to access and manipulate values. Arrays in C# belong to the reference type and hence are stored on heap.

  Array declaration in C#-

  DataType [number of elements] variablename;

  For Example-

  int [5] myarr;

- Structs In C#:

  Unlike the structs of C/C++, the structures in C# can have methods defined within them and they can have a constructor too.

- Enumerators In C#:

  Enums short for Enumerators are a set of named numeric constants which are useful when some value in a program can have a specific set of values.