UNIT -1

1.1 Introduction to C#
Understanding C# Environment
Overview of C#++

1.1 Introduction to C#

What is C#

- C# is pronounced as C-Sharp
- It is an object-oriented programming language created by Microsoft that runs on the .NET Framework.
- C# was developed by Anders Hejlsberg and his team during the development of .NET framework in the year 2000.
- C# has root from the C family and the language is close to other popular language C++ and Java.
- By the help of C# programming language we can develop different types of secured and robust applications like windows application, web application, database application, web services etc.
- C# is designed for Common Language Infrastructure (CLI), which consists of the executable code and runtime environment that allows use of various high-level languages on different computer platforms and architectures.

- The following reasons make C# a widely used professional language −
  - It is a modern, general-purpose programming language
  - It is object oriented.
  - It is component oriented.
  - It is easy to learn.
  - It is a structured language.
  - It produces efficient programs.
  - It can be compiled on a variety of computer platforms.
  - It is a part of .Net Framework.

Why C#

- A large number of computer languages, starting from FORTRAN developed in 1957 to the object-oriented language Java introduced in 1995 are being used for various applications.

  The choice of a language depends upon many factor such as hardware environment, business environment and user requirement so on.
- If we see the history of major languages which was developed during last two decades like C++ and C was the two most popular language which widely used in the software industry for the past two decades.

- With the help of this languages. The programmers can develop business, commercial application and able to control them in scientific way. However for certain reasons these languages cannot meet some requirement and standards of WWW.

  Some of the are-

  - They are not complete object oriented language
  - They are not suitable for working with new web technologies
  - They have poor-type-safety
  - Weak inconsistency and some more
- VB.Net (Visual basic) is a language which was promoted by Microsoft to overcome the problems but even this language could not able to meet some of the requirement of the www. Even VB is not fully an object oriented language.
- Later JAVA is a language derived from C , C++ which is truly an object oriented programming which is been widely used for web application. Even JAVA did not retain some C++ features such as operator overloading. Even it lacks in inter-operability with code developed in other languages.

So Microsoft decided to design a new language starting with clean slate. The result is C#, a simple and modern language that directly addresses the needs of component-based software development.

C# has many other reasons for being popular and in demand.

- **Easy to start:** C# is a high-level language so it is closer to other popular programming languages like C, C++, and Java and thus becomes easy to learn for anyone.
- **Widely used for developing Desktop and Web Application:** C# is widely used for developing web applications and Desktop applications. It is one of the most popular languages that is used in professional desktop. If anyone wants to create Microsoft apps, C# is their first choice.
- **Community:** The larger the community the better it is as new tools and software will be developing to make it better. C# has a large community so the developments are done to make it exist in the system and not become extinct.
- **Game Development:** C# is widely used in game development and will continue to dominate. C# integrates with Microsoft and thus has a large target audience. The C# features such as Automatic Garbage Collection, interfaces, object-oriented, etc. make C# a popular game developing language.

Evolution of C#

There was number of limitations in using the www over the Internet.

- We can see only one site at a time.
- The site has to be authored to our hardware environment.
- The information we get is basically read-only
- We cannot compare dynamically similar information stored in different sites.

So Microsoft Chairman Bill Gates, wanted to develop a software platform which overcome these limitations and enable users to get information anytime and anywhere, using a natural interface.

- The platform should be a collection of readily available Web services that can distributed and accessed via standard internet protocol. He wanted to make the Web both programmable and intelligent. The outcome is new generation platform called .NET
- Microsoft introduced C# which has been particularly designed to build software components for .NET and it supports key features of .NET natively.
- Like JAVA, C# is a descendant of C++ which in turn is a descendant of C. C is the mother of all the three modern languages.
- C# borrows Java's features such as grouping of classes, interfaces and implementation together in one file so the programmer can edit the code more easily.
- C# also handles objects using references, the same way as Java.
- C# uses VB's approach to form design, namely, dragging controls from a tool box, dropping them onto form and writing event handlers for them.
- C# modernizes C++ by enhancing some of its features and adding a few new features so as to help developers do more with fewer lines of code and fewer opportunities for error.

Characteristic of C#

1. Simple
2. Modern programming language
3. Object oriented
4. Type safe
5. Version able
6. Interoperability
7. Component oriented
8. Structured programming language
9. Rich Library
10. Fast speed

- **Simple**

C# is a simple language in the sense that it provides structured approach (to break the problem into parts), rich set of library functions, data types etc.
C# simplifies C++ by eliminating certain operators -> :: and pointers. C# treats integer and Boolean as two different datatype.

- **Modern programming language**

C# is called as modern language due to a number of features it supports like

- Automatic garbage collection
- Robust security mode

- Modern approach to debugging
- Decimal datatype for financial application

- **Object oriented**
  C# is truly object- oriented programing language. It supports three main principle of OOP's concepts
  - Encapsulation
  - Inheritance
  - Polymorphism
- **Type-safe**
  Type-safety promotes robust programs. C# incorporates a number to type-safe measures.
  - All dynamically allocated objects and arrays are initialized to zero
  - Use of any uninitialized variables produces an error message by the compiler
  - Access to arrays are range-checked and warned if it goes out-of-bounds
  - C# does not permit unsafe casts
  - C# enforces overflow checking in arithmetic operations
  - Reference parameters that are passed are type-safe
  - C# supports automatic garbage collection
- **Versionable**
  Making new versions of software modules work with the existing application is known as versioning. C# provides support for versioning with the help of **new** and **override** keyword. With this support, a programmer can guarantee that his new class library will maintain binary compatibility with the existing client application.

- **Interoperability**
  Interoperability of programming languages is the ability for two or more languages to interact as part of the same system. Frequently, this means passing messages and data between potentially very different languages..
- **Component Oriented**
  C# is component oriented programming language. It is the predominant software development methodology used to develop more robust and highly scalable applications.
- **Structured Programming Language**
  C# is a structured programming language in the sense that we can break the program into parts using functions. So, it is easy to understand and modify.

- **Rich Library**
  C# provides a lot of inbuilt functions that makes the development fast.
- **Fast Speed**
  The compilation and execution time of C# language is fast.

Advantages of C#

- Object-Oriented Language
- Automatic Garbage Collection
- Cross Platform
- Backward Compatibility
- Better Integrity and Interoperability

Applications of C#

- Games using Unity
- Web Applications Client-Server Applications
- Windows Applications that run on desktops
- Web Services Applications
- Console Applications
- Class Libraries

How does C# differ from C++?

Compiling	C++ compiles down to machine code	C# 'compiles' down to CLR (Common Language Runtime), which is interpreted by JIT in ASP.NET
Multiple inheritances	C++ support the multiple inheritances	C# does not support multiple inheritances.
Use of pointers	You can use pointers anywhere in the program.	You can use pointer only in the unsafe mode.
Language Type	C++ comprises both High-level language and Low-level language as it is built directly over C. It is an intermediate language	C# is a High-level programming language
Object-Oriented	C++ is not a completely object-oriented language.	C# is purely an object-oriented language

Garbage Collection	Does not support cleaning of unreferenced parts of a program	Supports cleaning of unreferenced parts of a program

How does C# differ from Java?

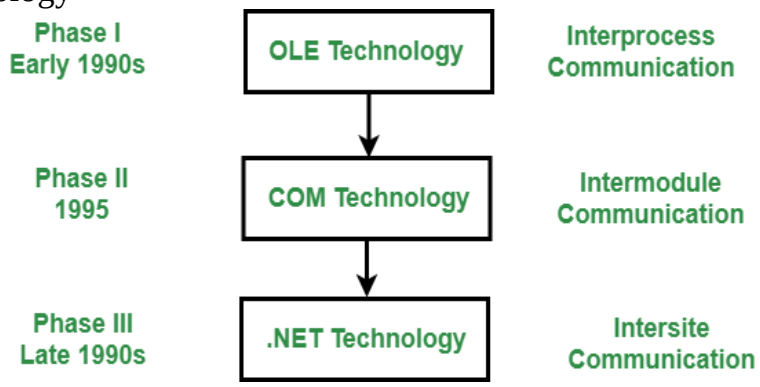
Polymorphism	Invokes the “virtual” keyword in a base class and “override” keyword in a derived class.	Enables polymorphism by default.
Designed for	Java programming language is intended to be run on a Java platform, by the help of Java Runtime Environment (JRE).	The C# programming language is designed to be run on the Common Language Runtime (CLR).
Safety type	Java type safety is safe.	C# type of safety is unsafe.
Built-in Datatype	Built-in data types that are passed by value are called simple types.	Built-in data types that are passed by value are known as primitive types.
Support for Goto statement.	Java doesn’t support the goto statement.	C# supports the goto statement.
Structure and unions	Java doesn’t support structures and unions.	C# supports structures and unions.
Installation	Requires JDK to run Java.	.Net framework provides a vast library of codes used by C#
IDE	Eclipse, NetBeans, IntelliJ IDEA	Visual Studio, MonoDevelop
Operator Overloading	No support for operator overloading	C# provides support for operator overloading for multiple operators.
Runtime Environment	Java supports JVM(Java Virtual Machine).	C# supports CLR(Common Language Runtime).
Platform Dependency	Java is a robust and platform independent language.	Code written in C# is windows specific.
Pointers	Java does not support pointers.	In C# you can use pointer only in an unsafe mode.

Understanding C# environment

Origins of .NET technology

There are three significant phases of the development of .NET technology.

- OLE Technology
- COM Technology
- .NET Technology



OLE Technology: Object linking and embedding (OLE) is a Microsoft technology that facilitates the sharing of application data and objects written in different formats from multiple sources. Linking establishes a connection between two objects, and embedding facilitates application data insertion..

COM Technology: Microsoft COM (Common Object Model) enables various software components to communicate. COM is mostly used by developers for various purposes like creating reusable software components, linking components together to build applications, and also taking advantage of Windows services. The objects of COM can be created with a wide range of programming languages.

.NET Technology: .NET technology of collection or set of technologies to develop and windows and web applications. The technology of .Net is developed by Microsoft and was launched in Feb. 2002, by basic definition, Microsoft's new Internet Strategy. It was originally called NGWS (Next Generation Web Services). It is considered to be as one of the powerful, popular and very much useful Internet Technology available today.

.Net framework

The .NET framework provides an environment for building, deploying and running web services and other applications.

Or

It is a virtual machine for compiling and executing programs written in different languages like C#**, VB.Net, etc.**

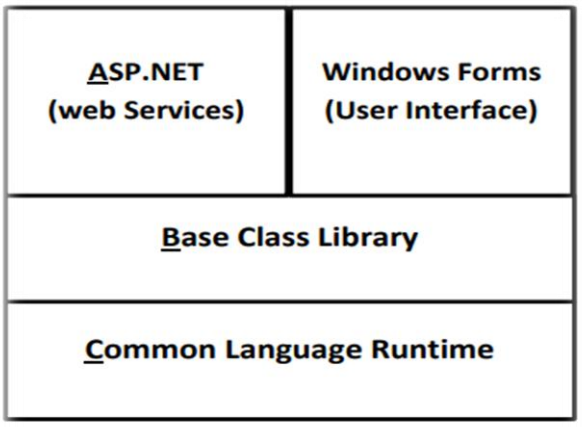
- **.NET** is a software framework that is designed and developed by Microsoft. The first version of the .Net framework was 1.0 which came in the year 2002.
- It is used to develop Form-based applications, Web-based applications, and Web services.

- There is a variety of programming languages available on the .Net platform, VB.Net and C# being the most common ones. It is used to build applications for Windows, phones, web, etc.
- .NET Framework supports more than 60 programming languages in which 11 programming languages are designed and developed by Microsoft.
- The remaining **Non-Microsoft Languages** are supported by .NET Framework but not designed and developed by Microsoft.

Architecture of .Net Framework

The .NET Framework provides a environment for building, deploying and running web services and other applications. It consists of three distinct technologies

- Common Language Runtime(CLR)
- Framework Base Classes
- User and program interfaces(ASP.Net and Windows)



Architecture of .NET Framework

- **Common Language Runtime**

The Common Language Runtime is properly known as CLR which is the heart and soul of .Net Framework.
.NET CLR is a run-time environment that manages and executes the code written in any .NET programming language. It makes the development process easier by providing various services such as remoting, thread management, type-safety, memory management, robustness, etc. It converts code into native code which further can be executed by the CPU.

- **Base Class Library**

.NET Base Class Library is the collection of classes, namespaces, interfaces and value types that are used for .NET applications.
It contains thousands of classes that supports the following functions.

- Base and user-defined data types
- Support for exceptions handling
- input/output and stream operations
- Access to data
- Ability to create Windows-based GUI applications, web-client and server applications.
- Support for creating web services

- **ASP.NET (Web services)**

  ASP.NET is a web framework designed and developed by Microsoft. It is used to develop websites, web applications, and web services. It provides a fantastic integration of HTML, CSS, and JavaScript.

- **WinForms (User interface)**

Windows Forms is a smart client technology for the .NET Framework, a set of managed libraries that simplify common application tasks such as reading and writing to the file system.

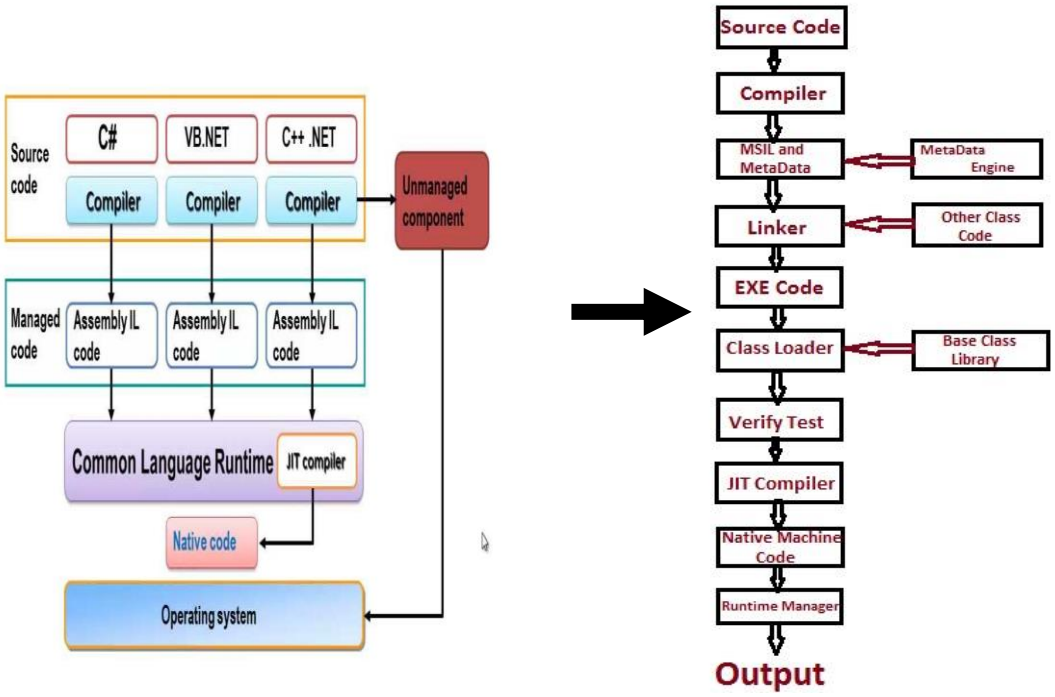
Common Language Runtime(CLR)

- The Common Language Runtime, popularly known as CLR is the heart and soul of the .NET Framework.
- CLR is a runtime environment in which programs written in C# and other .NET languages are executed.
- It also supports cross-interoperability

The CLR provides a number of services that include

- Loading and execution of programs
- Memory isolation for applications
- Verification of type safety
- Compilation of IL into native executable code
- Providing metadata
- Memory management (automatic garbage collection)
- Enforcement of security
- Interoperability with other systems
- Managing exception and errors
- Support for tasks such as debugging and profiling

CLR activities for executing a program



- Source code which is written in any .NET languages.
- Language specific compiler compiles the source code into the ***MSIL(Microsoft Intermediate Language)*** which is also known as the ***CIL(Common Intermediate Language)*** or ***IL(Intermediate Language)*** along with its metadata. *Metadata* includes all the types, actual implementation of each function of the program. MSIL is machine-independent code.
- IL and metadata are linked with other native code if required and the resultant IL code is saved.
- During execution, the IL code and any requirement from base class library are brought by the class loader.
- The combined code is tested for type-safety
- CLR includes the JIT(Just-In-Time) compiler which converts the MSIL code to machine code ,which is send to the runtime manager for execution.

Main Components of CLR

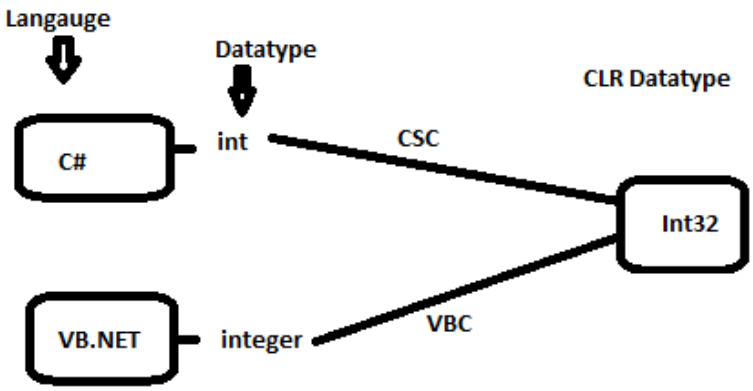
- CTS (Common Type System)
- CLS(Common Language Specification)

1. **CTS (Common Type System)**
   - CLS stands for Common Language Specification and it is a subset of CTS.
   - It defines a set of rules and restrictions that every language must follow which runs under the .NET framework.
   - The languages which follow these set of rules are said to be CLS Compliant.
   - In simple words, CLS enables cross-language integration or Interoperability.

   For Example

   - If we talk about C# and VB.NET then, in C# every statement must have to end with a semicolon. it is also called a statement Terminator, but in VB.NET each statement should not end with a semicolon(;).
   - So these syntax rules which you have to follow from language to language differ but CLR can understand all the language Syntax because in .NET each language is converted into MSIL code after compilation and the MSIL code is language specification of CLR.

2. **CLS (Common Language Specification)**
   - Common Type System (CTS) describes the data types that can be used by managed code.
   - CTS defines how these types are declared, used and managed in the runtime.
   - It facilitates cross-language integration, type safety, and high-performance code execution.
   - The rules defined in CTS can be used to define your own classes and values.
   - CTS deals with the data type. So here we have several languages and each and every language has its own data type and one language data type cannot be understandable by other languages but .NET Framework language can understand all the data types.
   - C# has an **int** data type and VB.NET has **Integer** data type. Hence a variable declared as an int in C# and Integer in VB.NET, finally after compilation, uses the same structure Int32 from CTS.



Microsoft Intermediate Language(MSIL)

- The Microsoft Intermediate Language (MSIL), also known as the Common Intermediate Language (CIL) is a set of instructions that are platform independent and are generated by the language-specific compiler from the source code.
- The MSIL is platform independent and consequently, it can be executed on any of the Common Language Infrastructure supported environments such as the Windows *.NET* runtime.
- The MSIL is converted into a particular computer environment specific machine code by the JIT compiler. This is done before the MSIL can be executed.

Managed Code and Unmanaged code

- A code which is written to aimed to get the services of the managed runtime environment execution like CLR(Common Language Runtime) in .NET Framework is known as **Managed Code**.
- A code which is directly executed by the operating system is known as **Unmanaged code**. It always aimed for the processor architecture and depends upon computer architecture.

Benefits of .NET approach

Microsoft has advanced the.NET strategy in order to provide a number benefits to developers and user

Some of the major benefits are

- Simple and faster systems development
- Rich object model
- Enhanced built-in functionality
- Many different ways to communicate with the outside world

- Integration of different languages into one platform
- Easy deployment and execution
- Wide range of scalability
- Interoperability with existing application
- Simple and easy-to-build sophisticated development tools
- Fewer bugs
- Potentially better performance

Overview of C#

Introduction

C# can be used to develop two categories of programs, namely

- Executable application programs
- Component libraries

➢ Executable programs are written to carry out certain tasks and require the method **Main** in one of the classes.

➢ Component libraries do not require a **Main** declaration because they are not standalone application programs. They are written for use by other applications. This concept is something similar to applets and application programs in JAVA

Simple C# Program

```
using System;
namespace HelloWorld
{
  class Program
   {
      public static void Main(string[] args)
      {
        Console.WriteLine("Hello World!");
      }
   }
}
```

- using System; means that we can use classes from the System namespace. **using System** before the class, it means we don't need to specify System namespace for accessing any class of this namespace.

- namespace HelloWorld is used to organize your code, and it is a container for classes and other namespaces.

- The curly braces {} marks the beginning and the end of a block of code. C# is a block-structured language, meaning code blocks are always enclosed by braces { and } .Therefore every class definition in C# begins with an opening brace { and ends with a corresponding closing brace } that appears in the last line in the program

- class Program is a container for data and methods, which brings functionality to your program. Every line of code that runs in C# must be inside a class. C#

is a true object-oriented language and therefore "everything must be placed inside a class" .

- **class** is a keyword and declare that a new class definition . **Program** is a C# identifier that specifies the name of the class to be defined. A class is a template for what an object looks like and how it behaves

- public static void Main()
  defines a method named **Main.** Every C# executable program must include the **Main()** method in one of the classes. This is the starting point for executing the program. A C# application can have any number of classes but only one class can have Main method to initiate the execution.
  This line contains number of keywords: **public**, **static** and **void**

- **public:** modifier before class and Main() method. Now, it can be accessed from outside the class also.
- **static:** is a keyword which means object is not required to access static members. So it saves memory.
- **void:** is the return type of the method. It doesn't return any value. In such case, return statement is not required.
- **string[] args:** is used for command line arguments in C#. While running the C# program, we can pass values. These values are known as arguments which we can use in the program.

- **Console.WriteLine("Hello World!"):** Here, **Console** is the class defined in System namespace. The **WriteLine()** is the static method of Console class which is used to write the text on the console. The **WriteLine** always appends a new line character to the end of the string.
  using System before the class, it means we don't need to specify System namespace for accessing any class of this namespace. Here, we are using Console class without specifying System.Console.

```
namespace HelloWorld
{
  class Program
   {
      public static void Main(string[] args)
      {
        System.Console.WriteLine("Hello World!");
      }
```

```
    }
}
```

System.Console.WriteLine("Hello World!"): Here, System is the namespace. Console is the class defined in System namespace. The WriteLine() is the static method of Console class which is used to write the text on the console.

Adding Comments

The C# comments are statements that are not executed by the compiler. The comments in C# programming can be used to provide explanation of the code, variable, method or class. By the help of comments, you can hide the program code also.

There are two types of comments in C#.

- Single Line comment
- Multi Line comment

C# Single Line Comment

The single line comment starts with // (double slash). Let's see an example of single line comment in C#.

```
using System;
  public class CommentExample
  {
public static void Main(string[] args)
{
   int x = 10;//Here, x is a variable
   Console.WriteLine(x);
 }
 }
```

Output

10

C# Multi Line Comment

The C# multi line comment is used to comment multiple lines of code. It is surrounded by slash and asterisk (/* ..... */). Let's see an example of multi line comment in C#.

```
using System;
 public class CommentExample
  {
  public static void Main(string[] args)
  {
  /* Let's declare and
  print variable in C#. */
    int x=20;
```

```
Console.WriteLine(x);
}
  }
```

Output:

20

Executing the Program

- After creating the Program (source code), save it with .cs file extension in a folder in your system. We can give any suitable name for the file. In case, we may use the class name itself.
  Example—
  sample.cs
- For compiling the program, go to the folder where you have stored the file sample.cs and then type the following
  csc sample.cs
- The C# compiler (csc) complies your code and creates an executable file (IL code) by name
  sample.exe
  in the same directory (if the code is error free) . If there are errors, the complier will produce appropriate error messages. Errors should be corrected and the program should be complied again.
- For executing the program, simply type the name of the executable file at the command prompt. That is
  sample
  This will display the output

Main Returning a Value

- Main method return type is also one of the important thing in C# programming.
- Main can also return a value if it is declared as int type instead of void.
- If we use the return type as int in Main method, need to add return statement at the end of the method. Returned value serves as the program termination status code.
- Purpose of this return value is to allow communication of success or failure to the execution environment.

```
public class CommentExample
  {
public static int Main(string[] args)
{
   int x = 10;
```

```
  Console.WriteLine(x);
   return 0;  //Return statement
 }
 }
```

Command Line arguments

- The arguments which are passed by the user or programmer to the **Main()** method is termed as Command-Line Arguments.
- Main() method is the entry point of execution of a program.
- Main() method accepts array of strings. But it never accepts parameters from any other method in the program.
- In C# the command line arguments are passed to the Main() methods by stating as follows:

```
static void Main(string[] args)

or

static int Main(string[] args)
```

```
using System;
namespace ComLineArg
{
   class Program
{
   static void Main(string[] args)
      {
       if(args.Length > 0)
         {
            Console.WriteLine("Arguments Passed by the Programmer:");
          foreach(Object obj in args)
            {
               Console.WriteLine(obj);
            }
         }
           else
         {
            Console.WriteLine("No command line arguments found.");
         }
   }   }
}
```

To Compile and execute the above program follow below commands :
Compile: csc Program.cs
Execute: Program.exe Welcome To Mysore!

Output :

```
Arguments Passed by the Programmer:

Welcome

To

Mysore
```

C# Basic Input and Output

C# Output

In order to output something in C#, we can use

```
System.Console.WriteLine()

OR

System.Console.Write()
```

Here, System is a namespace, Console is a class within namespace System and WriteLine and Write are methods of class Console.

Difference between WriteLine() and Write() method

The main difference between WriteLine() and Write() is that the Write() method only prints the string provided to it, while the WriteLine() method prints the string and moves to the start of next line as well.

Example : How to use WriteLine() and Write() method?

```
using System;
  namespace Sample
  {
  class Test
        {
        public static void Main(string[] args)
                {
                Console.WriteLine("Prints on ");
                Console.WriteLine("New line");
                   Console.Write("Prints on ");
                   Console.Write("Same line");
                }
        }
  }
```

When we run the program, the output will be

```
Prints on
New line
Prints on Same line
```

Printing Variables and Literals using WriteLine() and Write()

The WriteLine() and Write() method can be used to print variables and literals.

Example : Printing Variables and Literals

```
using System;
 namespace Sample
{
class Test
        {
        public static void Main(string[] args)
                {
                int value = 10;
Console.WriteLine(value);
Console.WriteLine(50.05);
                }
        }
}
```

When we run the program, the output will be

```
10
50.05
```

Combining (Concatenating) two strings using + operator and printing them

Strings can be combined/concatenated using the + operator while printing.

Example : Printing Concatenated String using + operator

```
using System;
 namespace Sample
```

```
{
class Test
        {
        public static void Main(string[] args)
                {
                int val = 55;
                Console.WriteLine("Hello " + "World");
                Console.WriteLine("Value = " + val);
                }
        }
}
```

When we run the program, the output will be

```
Hello World
Value = 55
```

Printing concatenated string using Formatted String [Better Alternative]

A better alternative for printing concatenated string is using formatted string. Formatted string allows programmer to use placeholders for variables.

For example,The following line,

```
Console.WriteLine("Value = " + val);
```

can be replaced by,

```
Console.WriteLine("Value = {0}", val);
```

{0} is the placeholder for variable val which will be replaced by value of val. Since only one variable is used so there is only one placeholder.

Multiple variables can be used in the formatted string.

Example : Printing Concatenated string using String formatting

```
using System;
 namespace Sample
{
class Test
```

```
{
        public static void Main(string[] args)
        {
                int firstNumber = 5, secondNumber = 10, result;
                result = firstNumber + secondNumber;
        Console.WriteLine("{0} + {1} = {2}", firstNumber, secondNumber, result);
                }
        }
}
```

When we run the program, the output will be

```
5 + 10 = 15
```

Here, {0} is replaced by firstNumber, {1} is replaced by secondNumber and {2} is replaced by result. This approach of printing output is more readable and less error prone than using + operator.

C# Input

In C#, the simplest method to get input from the user is by using the ReadLine() method of the Console class. They are also included in **Console** class.

Example : Get String Input From User

```
using System;
 namespace Sample
{
class Test
        {
        public static void Main(string[] args)
                {
                string testString;
                Console.Write("Enter a string - ");
                testString = Console.ReadLine();
                Console.WriteLine("You entered '{0}'", testString);
```

```
        }
    }
}
```

When we run the program, the output will be:

```
Enter a string - Hello World
You entered 'Hello World'
```

Difference between ReadLine(), Read() and ReadKey() method:

The difference between ReadLine(), Read() and ReadKey() method is:

ReadLine(): The ReadLine() method reads the next line of input from the standard input stream. It returns the same string.

Read(): The Read() method reads the next character from the standard input stream. It returns the ascii value of the character.

ReadKey(): The ReadKey() method obtains the next key pressed by user. This method is usually used to hold the screen until user press a key.

Example : Difference between Read() and ReadKey() method

```
using System;
 namespace Sample
{
class Test
    {
    public static void Main(string[] args)
        {
        int userInput;
        Console.WriteLine("Press any key to continue...");
        Console.ReadKey();
        Console.WriteLine();

        Console.Write("Input using Read() - ");
        userInput = Console.Read();
```

```
            Console.WriteLine("Ascii Value = {0}",userInput);
            }
        }
}
```

When we run the program, the output will be

```
Press any key to continue...
x
Input using Read() - Learning C#
Ascii Value = 76
```

While using ReadKey(), as soon as the key is pressed, it is displayed on the screen.

When Read() is used, it takes a whole line but only returns the ASCII value of first character. Hence, 76 (ASCII value of L) is printed.

Reading numeric values (integer and floating point types)

- We can use ReadLine() method for getting string values. But since the ReadLine() method receives the input as string, it needs to be converted into integer or floating point type.
- One simple approach for converting our input is using the methods of **Convert** class.

Example : Reading Numeric Values from User using Convert class

```
using System;
 namespace UserInput
{
class MyClass
        {
        public static void Main(string[] args)
                {
                        string userInput;
                        int intVal;
                        double doubleVal;
```

```
                    Console.Write("Enter integer value: ");
                    userInput = Console.ReadLine();
                    intVal = Convert.ToInt32(userInput);
                    Console.WriteLine("You entered {0}",intVal);

                    Console.Write("Enter double value: ");
                    userInput = Console.ReadLine();

                    doubleVal = Convert.ToDouble(userInput);
                    Console.WriteLine("You entered {0}",doubleVal);
                }
        }
}
```

When we run the program, the output will be

```
Enter integer value: 101
You entered 101
Enter double value: 59.412
You entered 59.412
```

The ToInt32() and ToDouble() method of Convert class converts the string input to integer and double type respectively.

Multiple Main Methods

C# includes a feature that enables us to define more than one class with the **Main** method .Since **Main** is the entry point for program execution, there are now more than one entry points. In fact there should be only one. This problem can be resolved by specifying which **Main** is to be used to the complier at the time of compilation

```
csc filename.cs/main:classname
```

Example: csc Program.cs/main:ClassA

```
using System;
class ClassA
{
```

```
public static void Main()
  {
Console.WriteLine(“Hello World”);
  }
}
class ClassB
{
public static void Main()
  {
Console.WriteLine(“Hello Mysore”);
  }
}
```

Output

```
Hello World
```

Types of Errors

Programming errors are mainly in three categories; Syntax Errors, Run time Errors and Logic Errors.

- **Syntax Errors**

  Syntax Errors, also known as Compilation errors are the most common type of errors. Most Syntax errors are caused by mistakes that you make when writing code.

  Ex: When you forgot to type a semicolon (;) after the statement, the compiler shows the syntax error and it would point out where the problem occurred.

- **Run-time Errors**

  Run-time errors are those that appear only after you compile and run your code. It will occur when your program attempts an operation that is impossible to carry out. You can fix most run-time errors by rewriting the faulty code, and then recompiling and running it.

  Ex: A program error may result from an attempt to divide by zero.

  When you compile this program, compiler won't show any Syntax Error. But when you run this code, the compiler show "Attempted to divide by zero".

- **Logic Errors**

  Code may compile and run without any Syntax Errors or Run-time Errors, but the output of an operation may produce unwanted or unexpected results in response to user actions. These types of errors are called Logic Errors. That means, Logic errors are those that appear once the application is in use.