# UNIT – 1

## .NET Framework and Visual Studio IDE, Language Basics

# Programming with C#

Code : CS-23

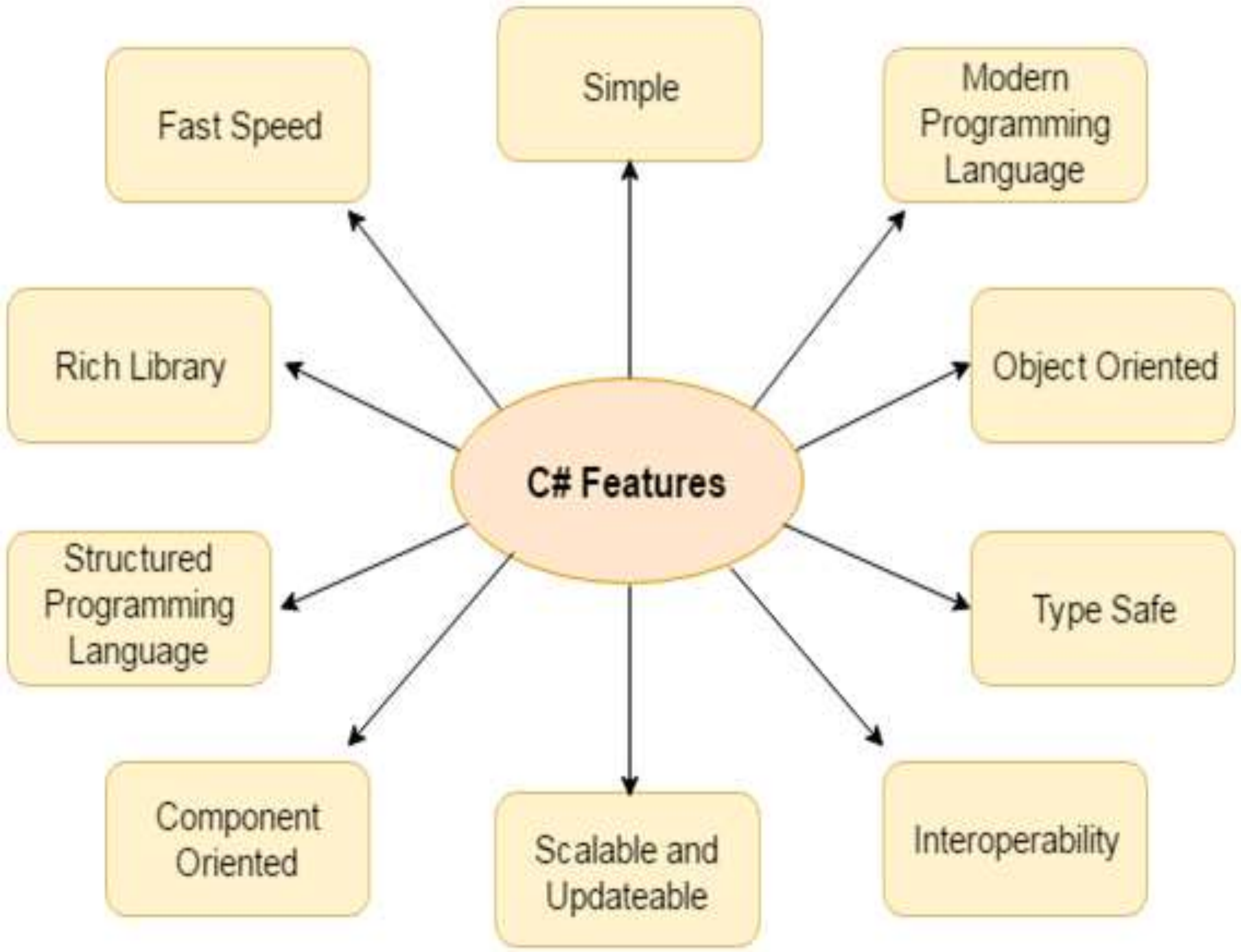
## What is C# :

- C# is pronounced "C-Sharp".
- It is an object-oriented programming language created by Microsoft that runs on the .NET Framework.
- **Anders Hejlsberg** is known as the **founder of C# language.**
- C# has roots from the C family, and the language is close to other popular languages like C++ and Java.
- The first version was released in year 2002. The latest version, **C# 13.0**, was released in November 2024.
- C# is used for:

  Mobile applications
  Desktop applications
  Web applications
  Web services
  Web sites
  Games
  VR
  Database applications

## ❖ Why Use C# :

- It is one of the most popular programming language in the world
- It is easy to learn and simple to use
- It has a huge community support
- C# is an object oriented language which gives a clear structure to programs and allows code to be reused, lowering development costs
- As C# is close to C,C++ and Java, it makes it easy for programmers to switch to C# or vice versa.


C# Features
Simple
Modern Programming Language
Object Oriented
Type Safe
Interoperability
Scalable and Updateable
Component Oriented
Structured Programming Language
Rich Library
Fast Speed

## ❖ Introduction to .NET Framework :

- The **.NET Framework** is a software development platform that was introduced by Microsoft in the late 1990 under the NGWS.
- In 13 February 2002, Microsoft launched the first version of the .NET Framework, referred to as the **.NET Framework 1.0**.
- .NET Framework is a technology that supports building and running Windows apps and web services.
- It is a virtual machine that provide a common platform to run an application that was built using the different language such as C#, VB.NET, Visual Basic, etc.
- It is also used to create a form based, console-based, mobile and web-based application or services that are available in Microsoft environment.
- Furthermore, the .NET Framework is a pure object oriented, that similar to the Java Language.
- But it is not a platform independent as the Java. So, its application runs only to the windows platform.

## Evolution and Overview of .NET :

- .NET is a framework that used to develop various software applications.
- This product was designed  and developed by Microsoft around 1990's.
- The class libraries present in this framework is known as FCL-Framework Class Library.
- It is layer between OS and language.
- Execution takes place in environment called CLR-Common Language Runtime.
- Latest version released in 2022 is .NET Framework 4.8.1
- .NET Framework supports 60+ Programming Languages.
- It is called as framework because it supports multiple languages , multiple technologies and multiple databases.

- Languages :  60+ languages like : 9 languages given by microsoft and other third party languages

  EX :c#.net , J#.net , F#.net,VB.net etc….
- Technologies : asp.net,ADO.net
- Database : sql server, oracle ,ms access,
- It provide common environment for devleop, deploy and execute application using .net framework that's why we can say it is CLR(common language runtime).
- It provides a library classes and objects.
- Library is called framework class library FCL.
- Run-time environment that is Common Language Run-time CLR.
- It provides language interoperability / language independence.

## Features of .NET Framework :

- **1. Cross- Language Interoperability:**
  Language interoperability is the ability of a code to interact with another code that is written by using a different programming language irrespective of the language in which it is created. Language interoperability can help maximize code reuse and improve the overall efficiency of the development process.
- **2. Multi-language Support**
  The .NET platform supports many programming languages.
- A new compiler must be implemented for each language , Framework 60+ languages.

- **3. Automatic Resource Management**
  In the common language runtime (CLR), the garbage collector serves as an automatic memory manager.
- The Features of .Net Framework's garbage collector manages the allocation and release of memory for our application. Each time we create a new object, the common language runtime allocates memory for the object from the managed heap.
- The runtime automatically handles object layout and manages references to objects, releasing them when they are no longer being used. This automatic memory management resolves the two most common application errors, memory leaks and invalid memory references.
- **4. Type Safety**
  Data structure in .NET supported languages has the same layout. This means that some code can consume types and instances declared in other languages.

- **5. Debugging**
  A debugger is a computer program that lets you run your program, line by line and examine the values of variables or look at values passed into methods and let you figure out why it isn't running the way you expected it to.
- **6. Elimination of DLL Hell**
  DLL Hell happens when multiple applications attempt to share a common DLL(Dynamic Link Library).
  Features of .NET Framework solves this problem by allowing multiple versions of the same DLL to coexist.
- **7. Security**
  Managed components are awarded varying degrees of trust, depending on a number of factors that include their origin (such as the Internet, enterprise network, or local computer). This feature empowers a developer to determine whether a managed component performs file-access operations, registry-access operations, and other sensitive functions or not.

## Advantages of .NET Framework :

- **1. Object-oriented Programming :**

  One of the advantages of the .NET framework is that it is based on an object-oriented programming module where the software is divided into small manageable pieces. The developers can work on one piece and then move on to the next. The .NET framework simplifies work for developers by eliminating programming and making code manageable.

- **2. Cross-platform Design** :

  when first launched, the Microsoft dot net framework was for windows. But it is now a cross-platform development that runs on platforms like Linux, iOS, Android and many more. The developers can build applications for multiple operating systems with any programming language like C#, F# and Visual Studio.

- **3. Support Applications** :

  With the .NET framework, the developers can create applications on multiple domains such as gaming, business, mobile, etc.
- **4. Easy Maintenance and Deployment** :

  The modular design of the .NET framework allows developers to work on applications separately. You can take apart applications and then fix them for updation and put them together.
- **5. Cost-effective** :

  Visual Studio by Microsoft is preferred by most developers as it is small, extensible and better than most programming languages. The .NET framework provides cheaper infrastructure and offers the freedom for developers to pick the provider of their choice.

## Disadvantages of .NET Framework :

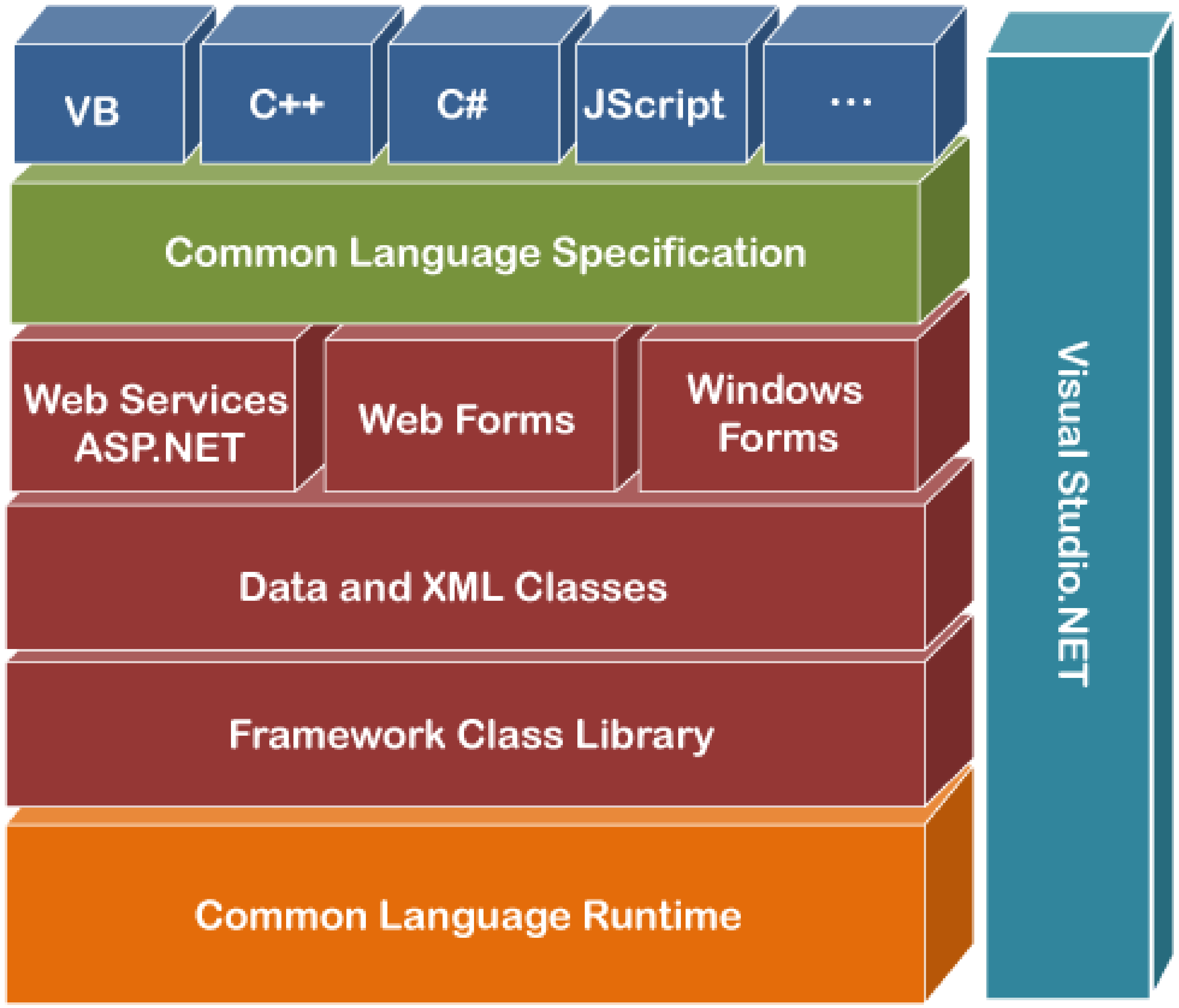
- **1. Licensing Cost :** The .NET framework costs money depending on the project and its demands. They cost money in terms of licensing fees. So, it is possible that .NET can be expensive.
- **2. Memory Leaks**: Although .NET possesses a garbage collector for memory-related issues, they are criticized for memory leaks. This can jeopardize the stability of the projects.
- **3. Object-Relational Programming Issues** : Since the .NET framework is object-oriented programming, it supports data-oriented software application development provided by Entity Framework. However, there have been concerns regarding the flexibility of the framework that supports new databases.
- **4. Vendor Lock-In** : Microsoft .NET framework impose limitations that impact the framework for .NET application. The developer has less control over the project and will depend on the decisions of Microsoft.

## ❖ .NET Framework Architecture :

- **.NET Framework Architecture** is a programming model for the .NET platform that provides an execution environment and integration with various programming languages for simple development and deployment of various Windows and desktop applications.
- It consists of class libraries and reusable components.

- There are Main Two Components of .NET framework :

1. CLR : Common Language Runtime.
2. FCL : Framework Class Library.


VB
C++
C#
JScript
…
Common Language Specification
Web Services ASP.NET
Web Forms
Windows Forms
Data and XML Classes
Framework Class Library
Common Language Runtime
Visual Studio.NET

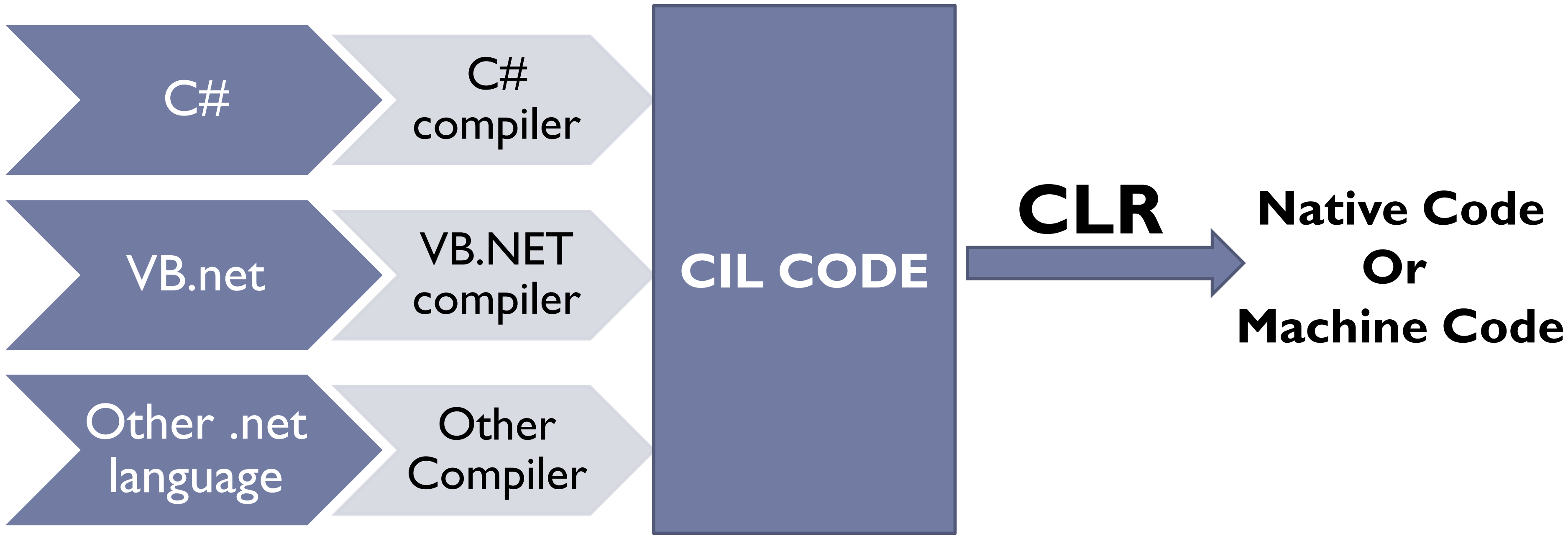
## Components of .NET Framework :

1. CLR (Common Language Runtime)
2. CTS (Common Type System)
3. CLS (Common Language Specification)
4. FCL (Framework Class Library)
5. BCL (Base Class Library)

## ❖ What is Common Language Runtime (CLR):

- CLR is execution engine that loads and executes the program.
- It is the hart of the .NET.
- It is runtime environment.
- CLR is a virtual machine.
- CLR is runtime engine provided by the .NET framework.
- It provides an infrastructure for running programs and allows them to communicate with other parts of the .NET framework.
- It acts as an interface between the framework and operating system.
- A CLR also helps to convert a source code into the byte code, and this byte code is known as CIL (Common Intermediate Language) or MSIL (Microsoft Intermediate Language).
- After converting into a byte code, a CLR uses a JIT(Just In Time) compiler at run time that helps to convert a CIL or MSIL code into the machine or native code.

- .NET Running Environment :



## Components of .NET CLR :

- The key components of CLR include the following:
- Class Loader :

  Used to load all classes at run time.
- MSIL(Microsoft Intermediate Language) to Native code :

  The Just In Time (JIT) compiler will convert MSIL code into native code.
- Code Manager :

  It manages the code at run time.
- Garbage Collector :

  It manages the memory. Collect all unused objects and de-allocate them to reduce memory.
- Thread Support
- Exception Handler :

  It handles exceptions at run time.

## ❖ Common Type System (CTS) :

- Common Type System (CTS) describes the datatypes 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 can support two types of category.

1. **Value Type**
2. **Reference Type**

- **Value Types :**

  Contain the values that need to be stored directly on the stack or allocated inline in a structure. They can be built-in (standard primitive types), user-defined (defined in source code) or enumerations (sets of enumerated values that are represented by labels but stored as a numeric type).
- Example :

```
int a;
```

- **Reference Types :**
- Store a reference to the value's memory address and are allocated on the heap. Reference types can be any of the pointer types, interface types or self-describing types (arrays and class types such as user-defined classes, boxed value types and delegates).
- Example :

```
student obj = new student();
```

Here, student is a class and obj is object.

## ❖ Common Language Specification (CLS):

- CLS is a sub set of CTS.
- The Common Language Specification (CLS) is a fundamental set of language features supported by the Common Language Runtime (CLR) of the .NET Framework.
- It means that all of the rules in CTS also apply to the CLS.
- CLS is a part of the specifications of the .NET Framework.
- CLS was designed to support language constructs commonly used by developers and to produce verifiable code, which allows all CLS-compliant languages to ensure the type safety of code.
- CLS includes features common to many object-oriented programming languages.

## ❖ Framework Class Library (FCL) :

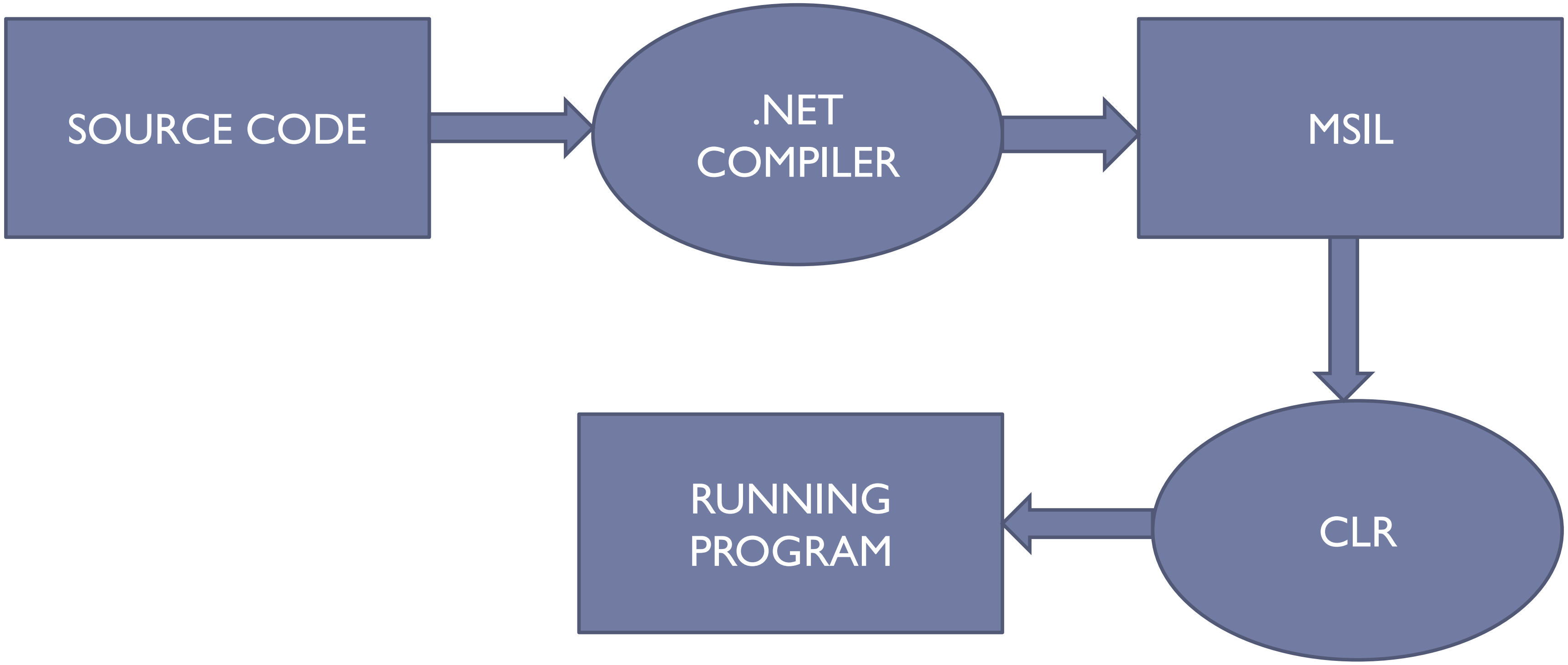
- Framework Class Library is the collection of classes, namespaces, interfaces and value types that are used for .NET applications.
- FCL is provide various types of functionality to the .NET code.
- In C <conio.h>,<stdio.h> are header files, they are added program to use inbuilt functions.

## ❖ Base Class Library (BCL) :

- BCL is Subset of FCL.
- Base Class Library is the sub part of the Framework that provides library support to Common Language Runtime to work properly.
- It includes the System namespace and core types of the .NET framework.

## ❖ Microsoft Intermediate Language (MSIL):

- MSIL, which stands for Microsoft Intermediate Language, serves as a crucial component within the .Net Framework.
- This versatile language is also referred to as **Intermediate Language (IL)** or **Common Intermediate Language (CIL)**.
- During the compilation process, the source code written in various .Net languages undergoes a transformation, converting it into Microsoft Intermediate Language (MSIL).
- This intermediate representation acts as an intermediary between the high-level source code and the native code specific to the underlying CPU architecture.
- MSIL is the CPU independent instruction set into .net framework program.
- MSIL remarkable attribute of being completely independent of any particular CPU architecture
- With the magic of MSIL, it becomes possible to enable programmers to "**write once, run anywhere**".


SOURCE CODE
.NET COMPILER
MSIL
CLR
RUNNING PROGRAM

## ❖ Namespace :

- Namespaces are used to organize the classes.
- It helps to control the scope of methods and classes in larger .Net programming projects.
- In simpler words you can say that it provides a way to keep one set of names(like class names) different from other sets of names.
- The biggest advantage of using namespace is that the class names which are declared in one namespace will not clash with the same class names declared in another namespace.
- It is also referred as **named group of classes** having common features.

- Namespaces are heavily used in C# programming in two ways.
- First Way :
- .NET uses namespaces to organize its many classes, as follows:

```
System.Console.WriteLine("Hello World!");
```

- Second Way :
- System is a namespace and Console is a class in that namespace.
- The using keyword can be used so that the complete name isn't required, as in the following example:

```
using System;
Console.WriteLine("Hello World!");
```

- **Namespaces overview**
- Namespaces have the following properties:
- They organize large code projects.
- They're delimited by using the . operator.
- The using directive obviates the requirement to specify the name of the namespace for every class.
- The global namespace is the "root" namespace: global::System will always refer to the .NET System namespace.

## ❖ Assembly :

- **Assembly** is a file that is automatically generated by compiler when compilation cycle successfully execute of every .NET application.
- **Assembly** is a collection of code files that are compiled into an **executable(.exe)** or **Dynamic Link Library (.DLL)**.
- The assembly file generate only one in execute of .NET application.
- Depending on their location and intended use, assemblies can be divided into many categories.
- There are Two Types of Assembly :

1. Private assembly
2. Share assembly

**1. Private Assembly :**

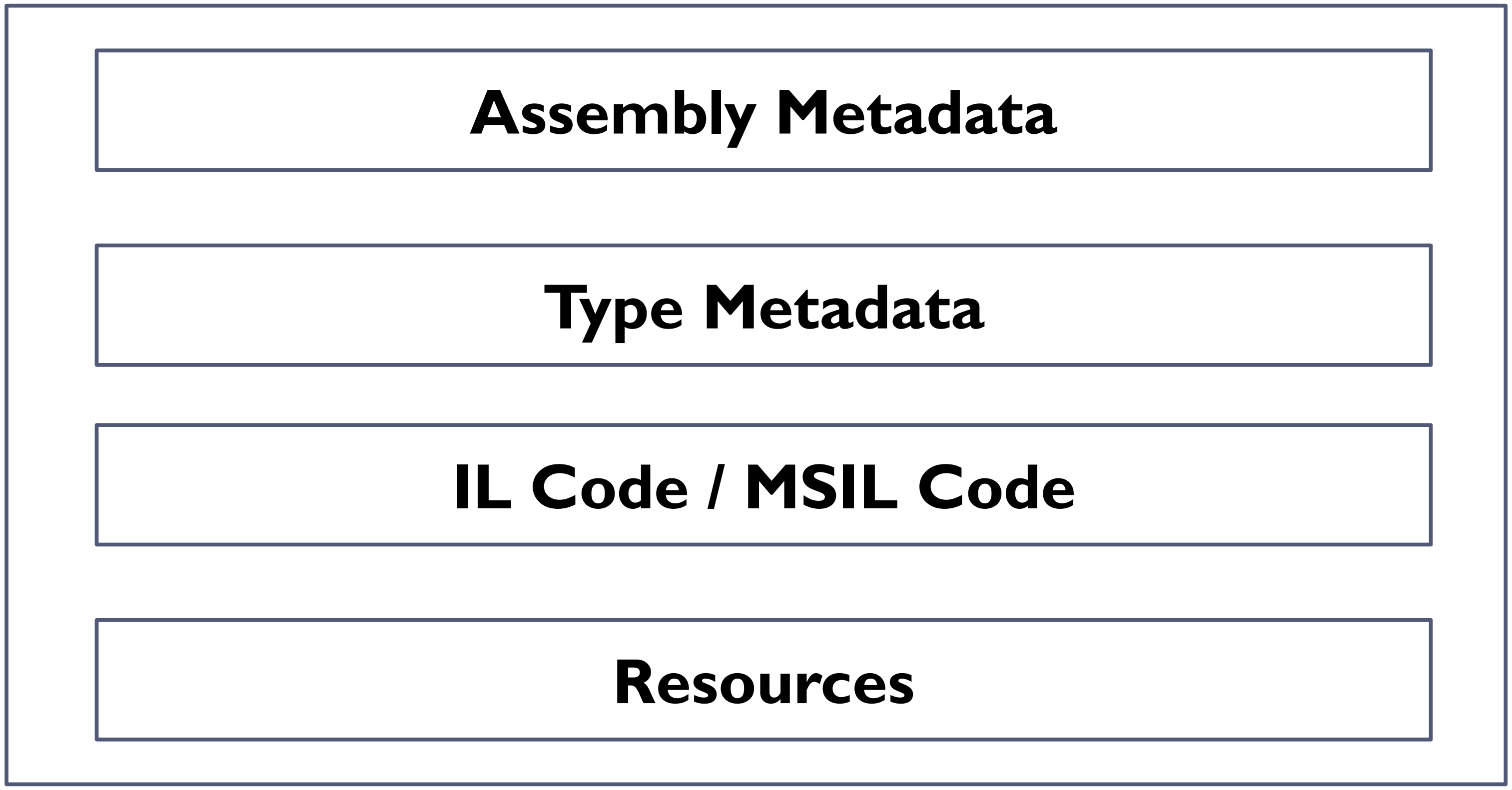
- It is typically found in the directory for the application or a subdirectory of the directory for the program.
- Private Assemblies are not intended to be shared with other applications.
- They are used to store application-specific code and resources.
- EX : .NET

**2. Shared assembly :**

- It is typically found in the **Global Assembly Cache (GAC)** or a common directory.
- Multiple applications are supposed to share a shared assembly.
- They are used to store resources and code that are shared by various applications.
- EX : MAC OS , LINUX , UNIX

- **What is Dynamic Link Library (DLL) ?**
- A **Dynamic Link Library (DLL)** is a type of assembly contains code that can be used by multiple applications.
- Similar to shared assemblies, **DLLs** are created with the intention of being used by numerous applications.
- **DLLs**, however, differ from shared assemblies in that they do not have a distinctive name.
- **DLLs** are kept in the directory of the application or a subdirectory of the directory of the application.
- They can be used by multiple applications by simply copying the **DLL** to the application's directory.

## Structure of Assembly :



## ❖ What is Metadata :

- Metadata stored within the Assembly.
- .NET records information about compiled classes as Metadata.
- Metadata means data about data.
- A .NET language compiler will generate the metadata and store this in the assembly.
- On the .NET Platform programs are compiled into .NET PE (Portable Executable) files.
- The header section of every .NET PE file contains a special new section for Metadata.
- Metadata is nothing but a description of every namespace, class, method, property etc. contained within the PE file.
- The CLR uses this metadata to

- Locate classes
- Load classes
- Generate native code
- Provide security

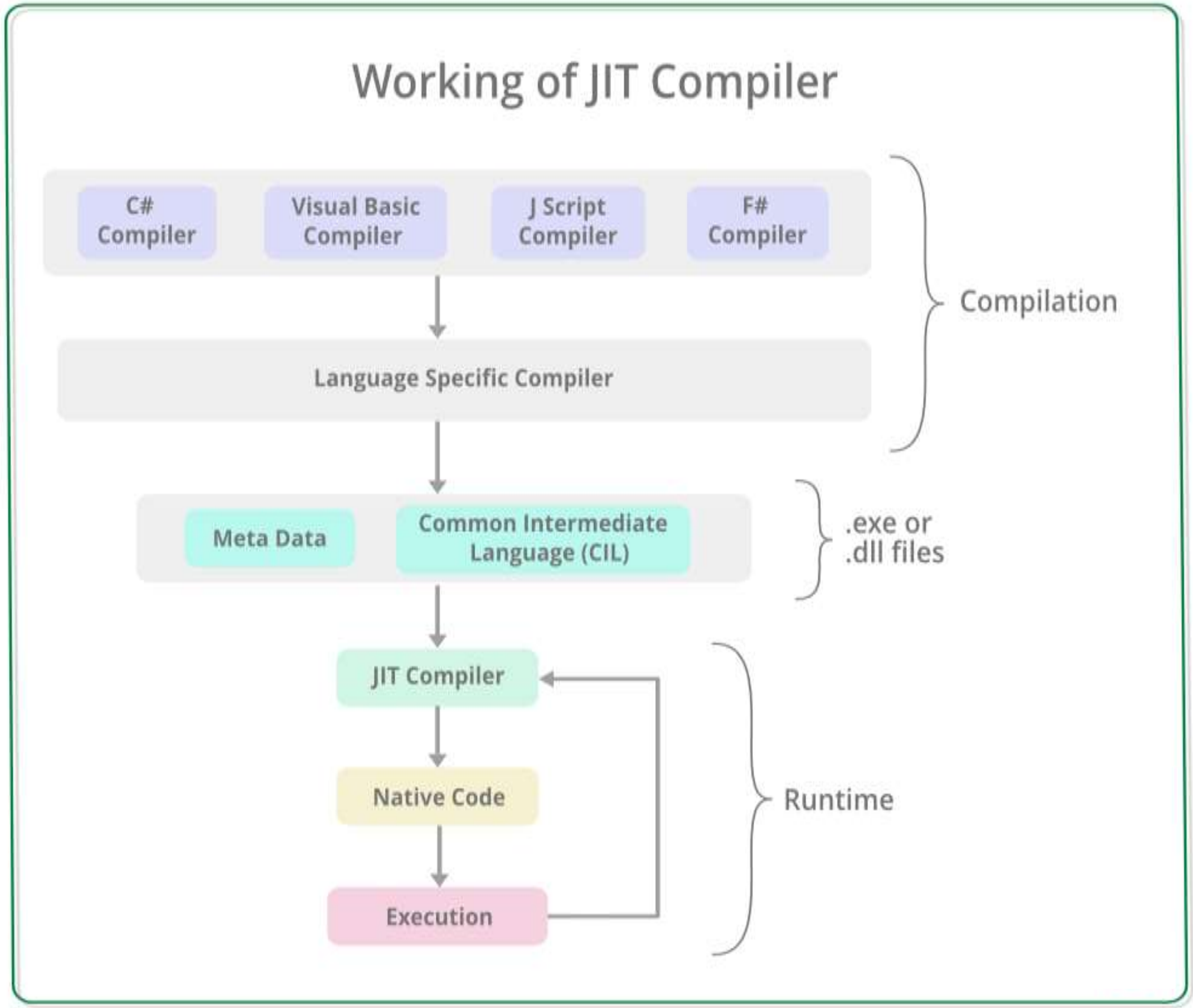
## ❖ Garbage Collector :

- Garbage collection is a memory management technique used in the .NET Framework and many other programming languages.
- In C#, the garbage collector is responsible for managing memory and automatically freeing up memory that is no longer being used by the application.
- The garbage collector works by periodically scanning the application's memory to determine which objects are still being used and which are no longer needed.
- Objects that are no longer being used are marked for garbage collection, and their memory is freed up automatically by the garbage collector.

## ❖ JIT Compiler :

- Just-In-Time compiler(JIT) is a part of **Common Language Runtime (CLR)** .
- in *.NET* which is responsible for managing the execution of *.NET* programs regardless of any *.NET* programming language.
- A language-specific compiler converts the source code to the intermediate language.
- This intermediate language is then converted into the machine code by the Just-In-Time (JIT) compiler.
- This machine code is specific to the computer environment that the JIT compiler runs on.
- The JIT compiler also enforces type-safety in the runtime environment of the .NET Framework.

- **Working of JIT Compiler:** The JIT compiler is required to speed up the code execution and provide support for multiple platforms.

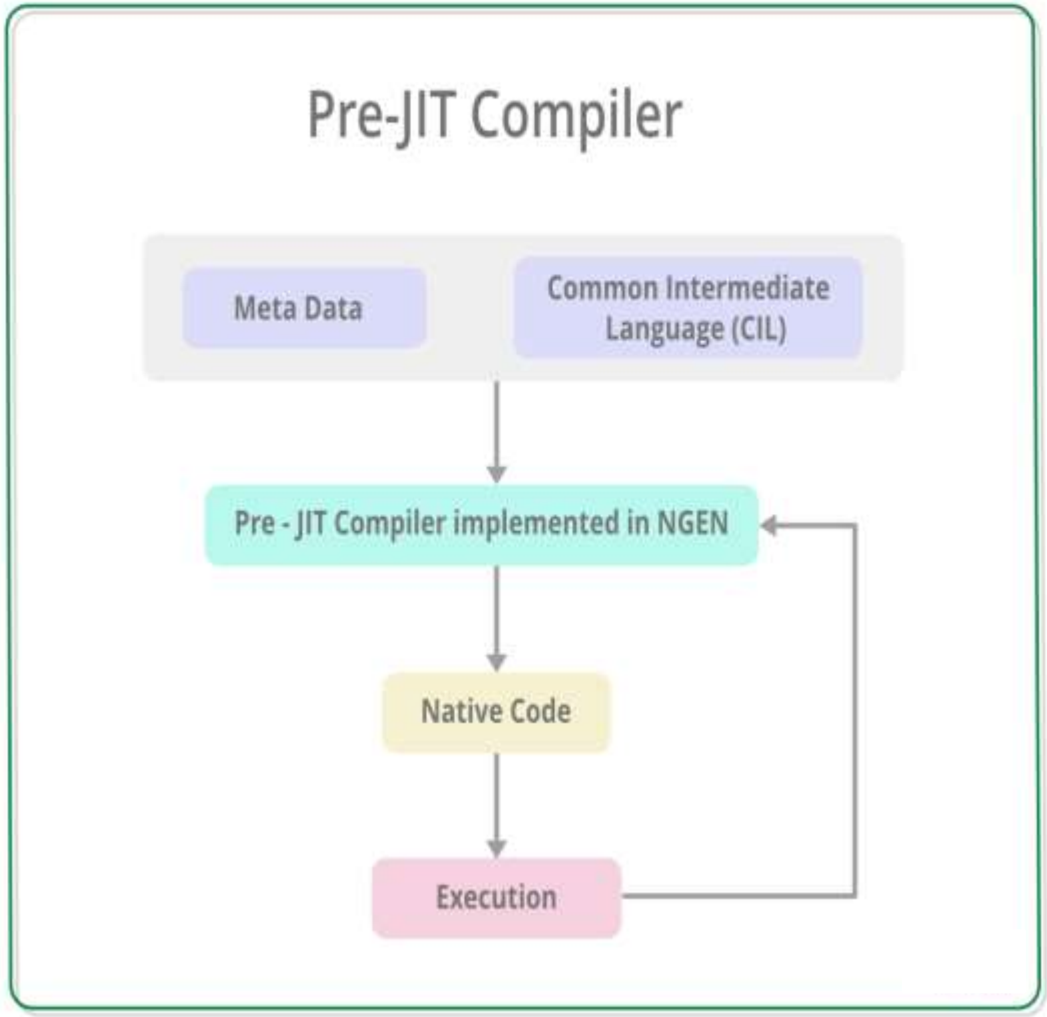


- **Types of Just-In-Time Compiler :**
- There are Three types of JIT Compiler.

1. **Pre JIT Compiler** It compiles complete program into native code in a single compilation cycle. This work is done at the time of deployment of the program.
2. **Econo JIT Compiler** It compiles only those methods that are called at runtime.
3. **Normal JIT Compiler** It's like Econo-JIT. The methods are compiled the 1st time they are stored in cache. When the same methods are called again the- compiled code from cache is used for execution.

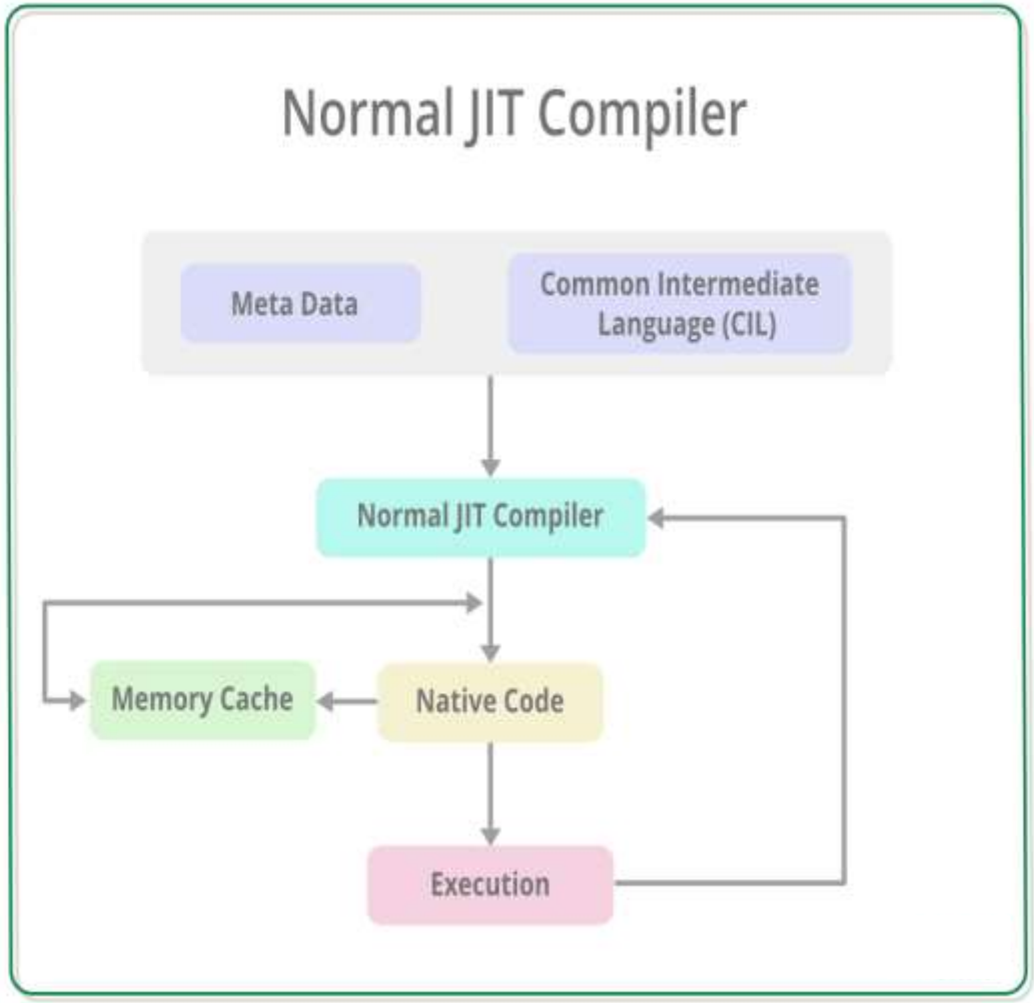
## 1. PRE JIT Compiler :

- All the source code is compiled into the machine code at the same time in a single compilation cycle using the Pre-JIT Compiler.
- This compilation process is performed at application deployment time, And this compiler is always implemented in the ***Ngen.exe (Native Image Generator)***.


Pre-JIT Compiler
Meta Data
Common Intermediate Language (CIL)
Pre - JIT Compiler implemented in NGEN
Native Code
Execution

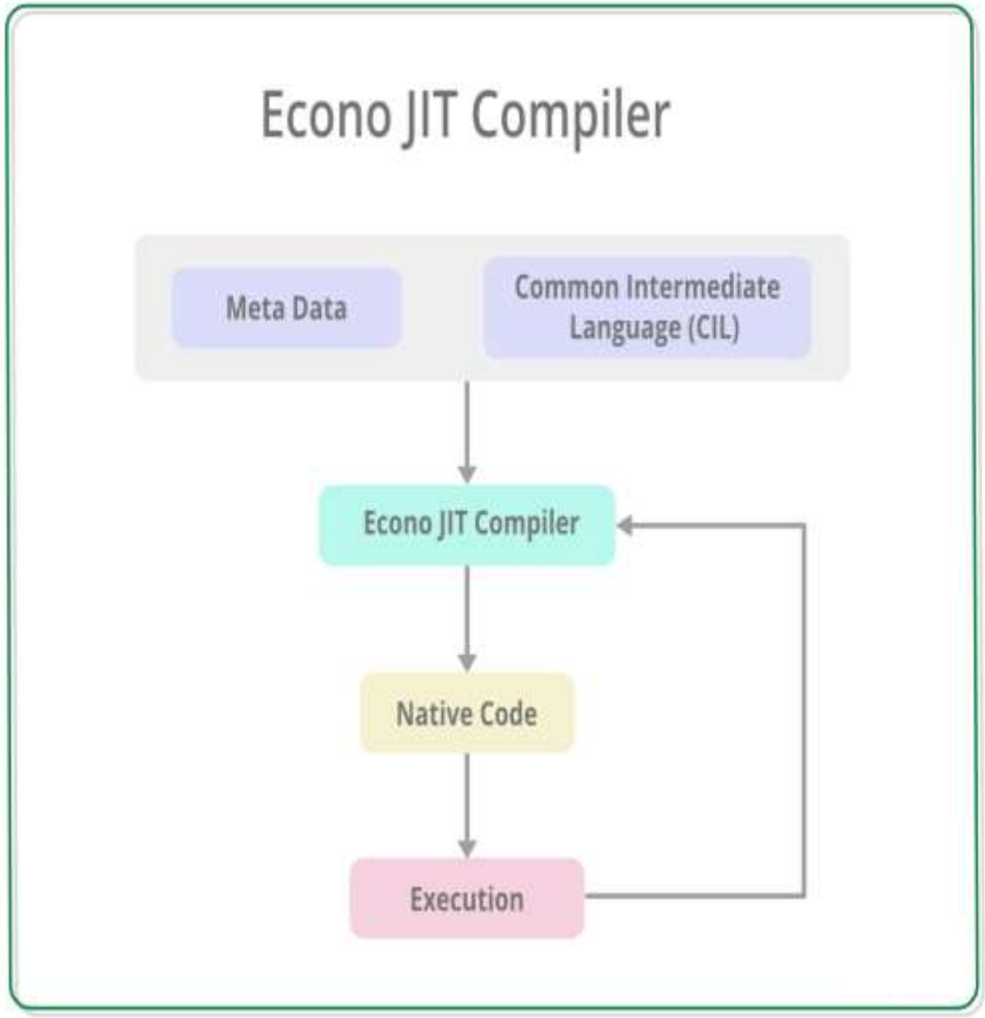
## 2. Normal JIT Compiler :

- The source code methods that are required at run-time are compiled into machine code the first time they are called by the Normal JIT Compiler. After that, they are stored in the cache and used whenever they are called again.



## 3. Econo JIT Compiler :

- The source code methods that are required at run-time are compiled into machine code by the Econo JIT Compiler.
- After these methods are not required anymore, they are removed.
- This JIT compiler is obsolete starting from .NET 2.0



- **Advantages of JIT Compiler:**
- The JIT compiler requires less memory usage as only the methods that are required at run-time are compiled into machine code by the JIT Compiler.
- Page faults are reduced by using the JIT compiler as the methods required together are most probably in the same memory page.
- Code optimization based on statistical analysis can be performed by the JIT compiler while the code is running.
- **Disadvantages of JIT compiler:**
- The JIT compiler requires more startup time while the application is executed initially.
- The cache memory is heavily used by the JIT compiler to store the source code methods that are required at run-time.

- **Note:** Much of the disadvantages of the JIT compiler can be handled using the Ahead-of-time (AOT) compilation. This involves compiling the MSIL into machine code so that runtime compilation is not required and the machine code file can be executed natively.

## ❖ Managed Code :

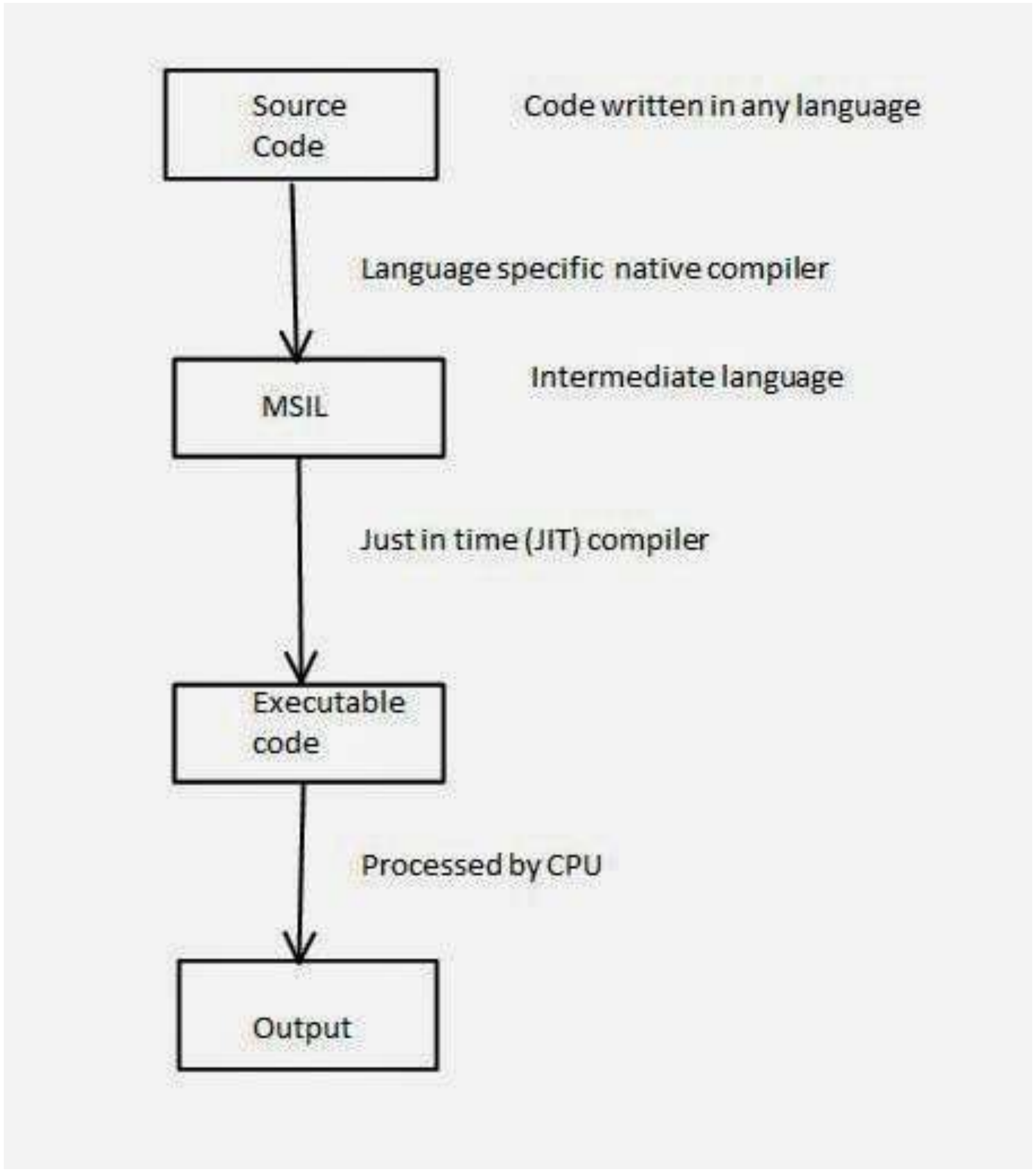
- A manage code which are created on .NET framework is known as Managed Code.
- A code which are execute under control by CLR is known as managed code.
- EX :

  C# .NET

  F# .NET

  VB .NET
- A CLR provides following services for manage code.

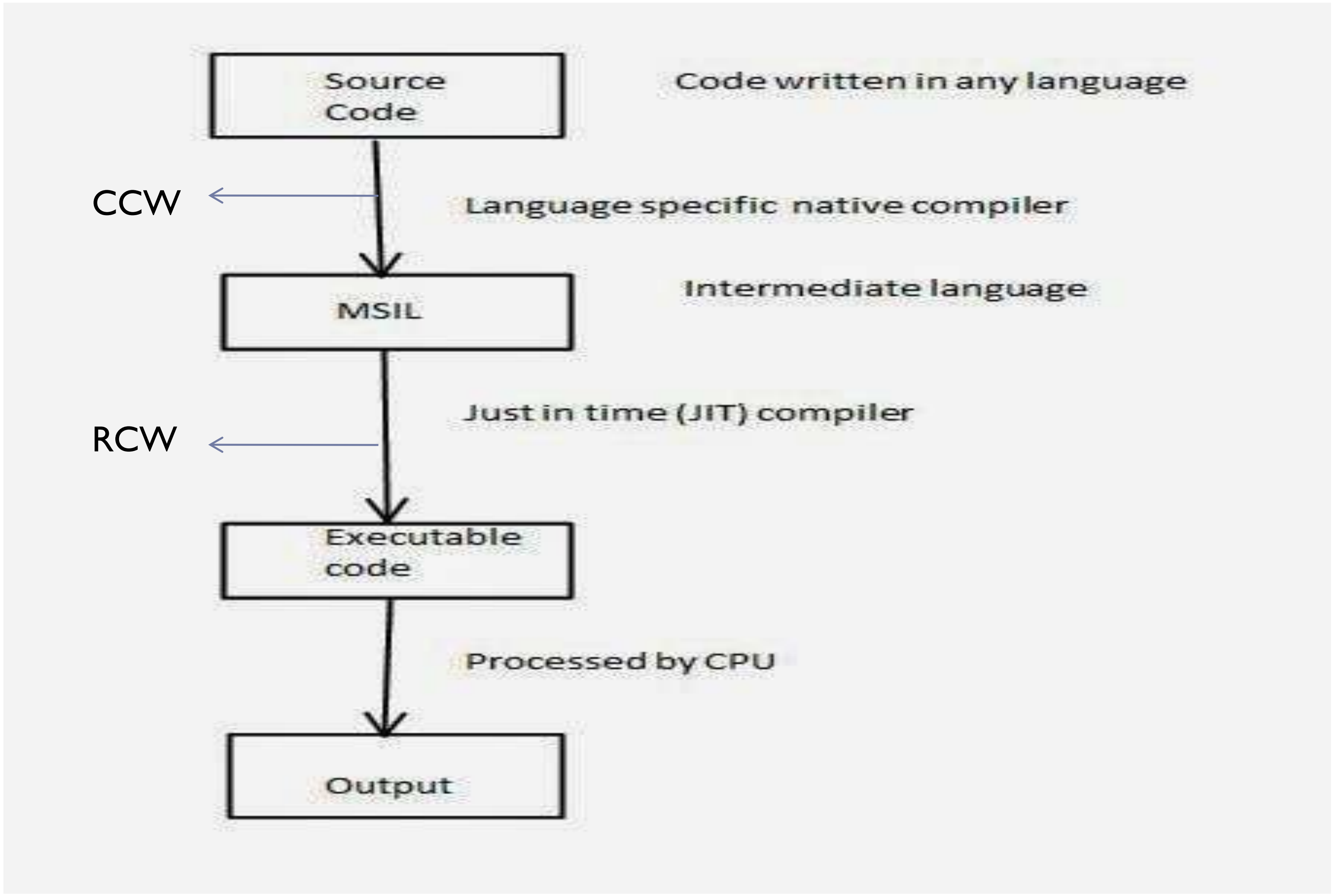
1. Automatic Memory Management.
2. Type Checking
3. Exception Handling.
4. Thread support.
5. Debug Support.


Source Code
Code written in any language
Language specific native compiler
MSIL
Intermediate language
Just in time (JIT) compiler
Executable code
Processed by CPU
Output

## ❖ Unmanaged Code :

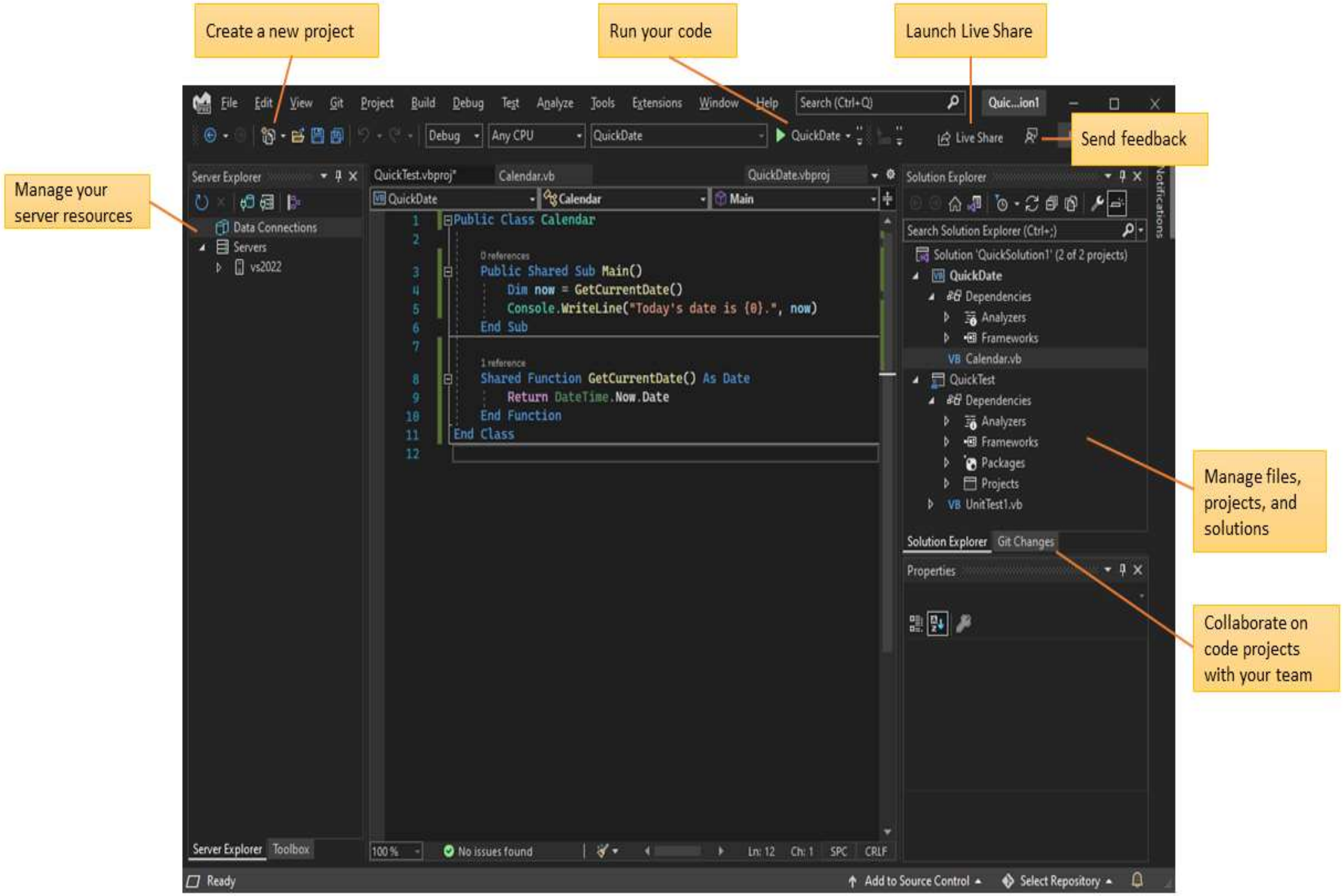
- Unmanaged Code is code that is executed outside the **.NET** runtime environment.
- This means that the code is not managed by the **Common Language Runtime (CLR)**, which is the virtual machine that executes **.NET** applications.
- Unmanaged Code can be written in any programming language that can produce native code, such as C, C++, or assembly language.
- Unmanaged code is executed with help of wrapper classes.
- Wrapper classes are of two types:

1. **CCW (COM Callable Wrapper)**
2. **RCW (Runtime Callable Wrapper).**


Source Code
Code written in any language
CCW
Language specific native compiler
MSIL
Intermediate language
Just in time (JIT) compiler
RCW
Executable code
Processed by CPU
Output

## ❖ What is **IDE** :

- Integrated development environments (IDE) are applications that facilitates the development of other applications.
- Designed to encompass all programming tasks in one application, one of the main benefits of an IDE is that they offer a central interface with all the tools a developer needs.
- **Code editor:** Designed for writing and editing source code, these editors are distinguished from text editors because work to either simplify or enhance the process of writing and editing of code for developers
- **Compiler:** Compilers transform source code that is written in a human readable/writable language in a form that computers can execute.
- **Debugger:** Debuggers are used during testing and can help developers debug their application programs.
- **Build automation tools:** These can help automate developer tasks that are more common to save time.



```
Public Class Calendar

    0 references
    Public Shared Sub Main()
        Dim now = GetCurrentDate()
        Console.WriteLine("Today's date is {0}.", now)
    End Sub

    1 reference
    Shared Function GetCurrentDate() As Date
        Return DateTime.Now.Date
    End Function
End Class
```

## ❖ Types of IDE :

1. Console Application
2. Windows Applications
3. Web Applications
4. Setup

## 1. Console Application :

- A Console Application, is an application that takes inputs and displays output at a command-line console with access to three basic streams like :

  standard input

  standard output

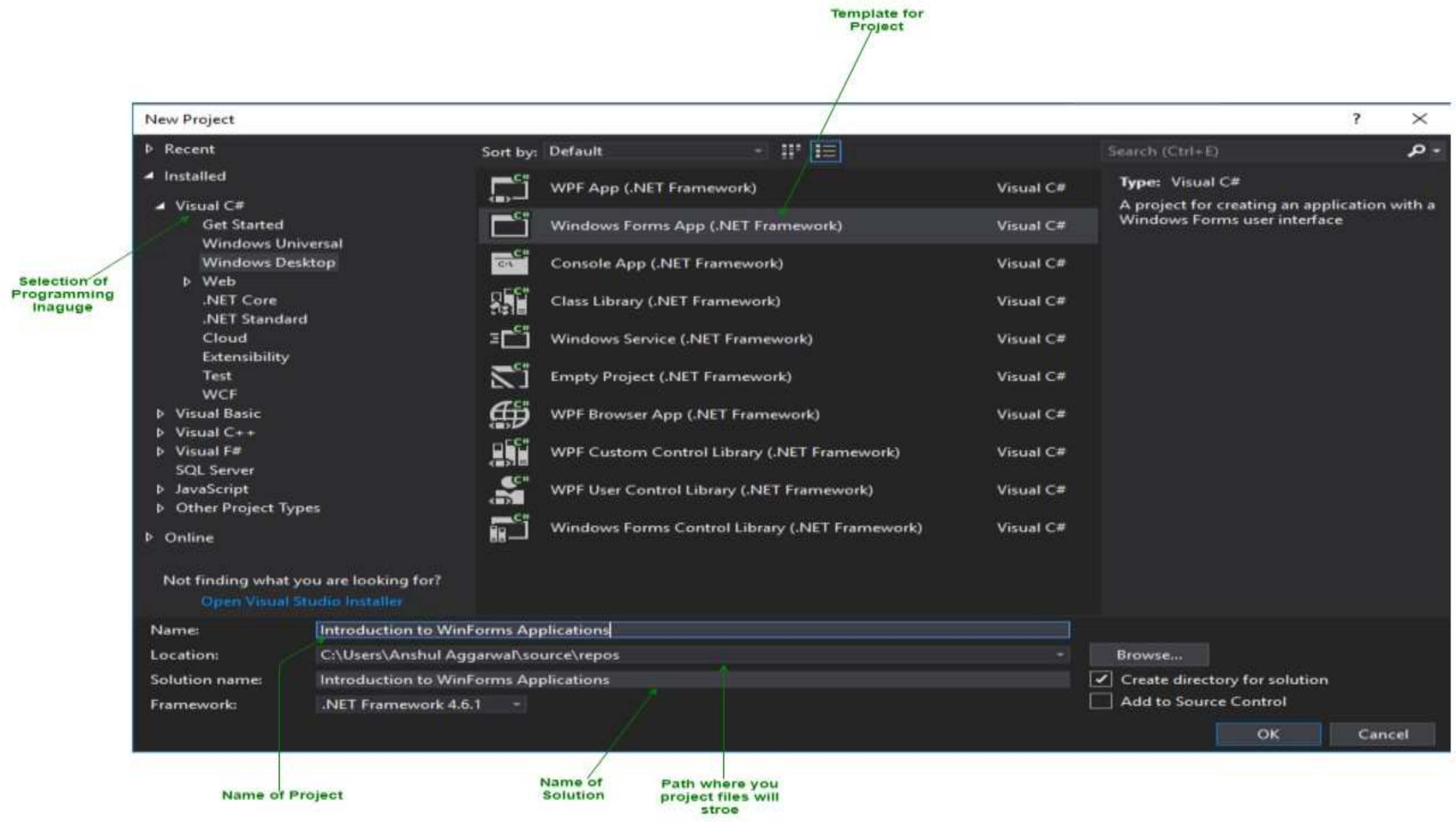
  standard error.
- The console application primarily is designed for the following reasons,
  - To provide a simple user interface for applications requiring little or no user interaction, such as samples for learning C# language features and command-line utility programs.
  - Automated testing, which can reuse automation implementation resources.
- Console applications don't have any graphical user interface, they have character-based interfaces.
- It is used for reading or writing characters from the console.

## 2. Windows Application :

- A Windows forms application is one that runs on the desktop computer.
- A Windows forms application will normally have a collection of controls such as labels, textboxes, list boxes, etc.
- Windows Forms is a Graphical User Interface(GUI) class library which is bundled in *.Net Framework*.
- Its main purpose is to provide an easier interface to develop the applications for desktop, tablet, PCs. It is also termed as the **WinForms**

## ➢ Steps for creating WinForms :

- First, open the Visual Studio then Go to **File → New → Project** to create a new project and then :


Template for Project
Selection of Programming inaguge
Name of Project
Name of Solution
Path where you project files will stroe
New Project
Recent
Installed
Visual C#
Get Started
Windows Universal
Windows Desktop
Web
.NET Core
.NET Standard
Cloud
Extensibility
Test
WCF
Visual Basic
Visual C++
Visual F#
SQL Server
JavaScript
Other Project Types
Online
Not finding what you are looking for?
Open Visual Studio Installer
Sort by: Default
Search (Ctrl+E)
WPF App (.NET Framework) Visual C#
Windows Forms App (.NET Framework) Visual C#
Console App (.NET Framework) Visual C#
Class Library (.NET Framework) Visual C#
Windows Service (.NET Framework) Visual C#
Empty Project (.NET Framework) Visual C#
WPF Browser App (.NET Framework) Visual C#
WPF Custom Control Library (.NET Framework) Visual C#
WPF User Control Library (.NET Framework) Visual C#
Windows Forms Control Library (.NET Framework) Visual C#
Browse...
OK
Cancel

Type: Visual C#

A project for creating an application with a Windows Forms user interface

Name: Introduction to WinForms Applications

Location: C:\Users\Anshul Aggarwal\source\repos

Solution name: Introduction to WinForms Applications

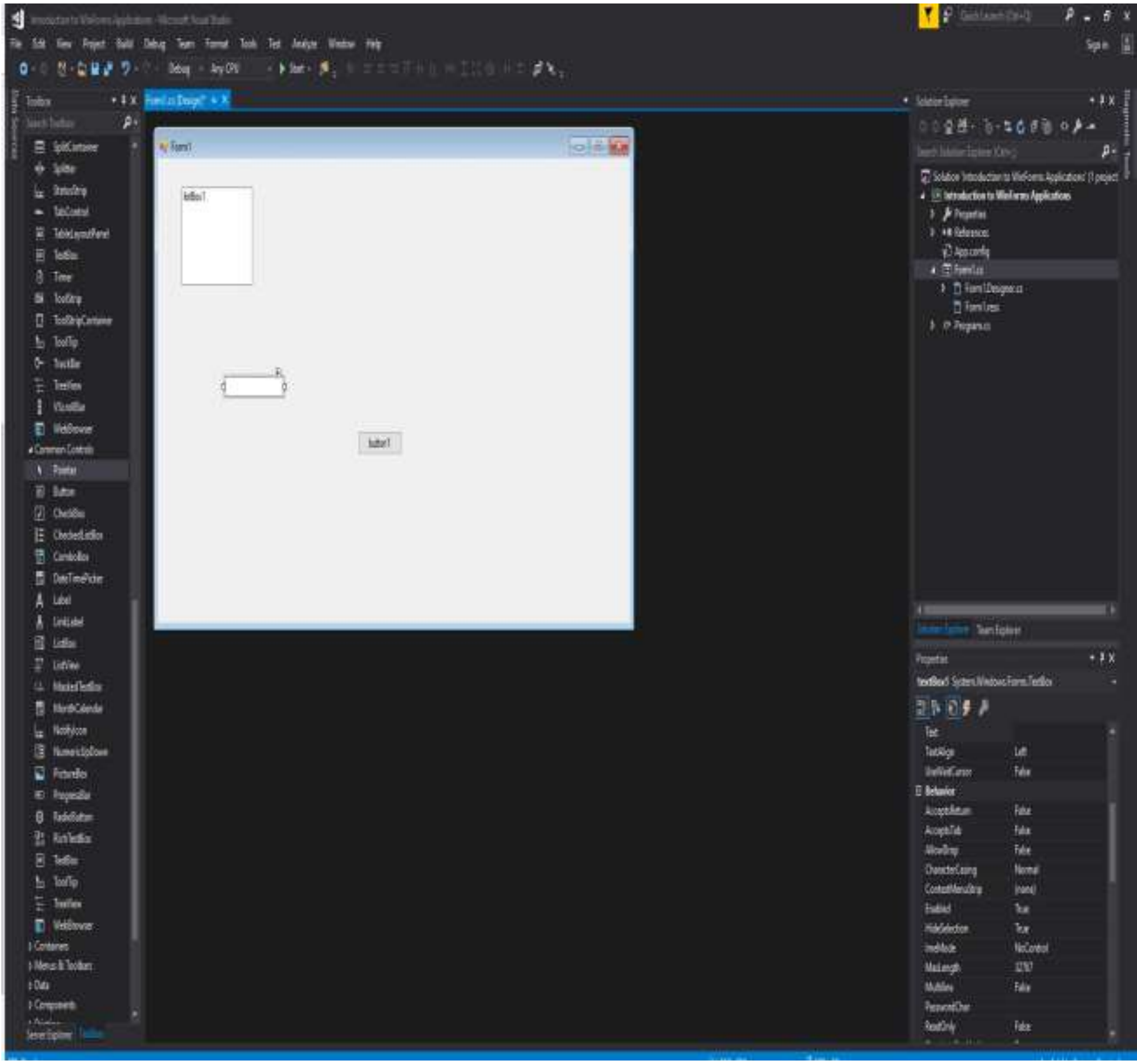
Framework: .NET Framework 4.6.1

[✓] Create directory for solution

[ ] Add to Source Control

- After that following window will display which will be divided into three parts as follows:
  - **Editor Window or Main Window:** Here, you will work with forms and code editing. You can notice the layout of form which is now blank. You will double click the form then it will open the code for that.
  - **Solution Explorer Window:** It is used to navigate between all items in solution. For example, if you will select a file form this window then particular information will be display in the property window.
  - **Properties Window:** This window is used to change the different properties of the selected item in the Solution Explorer. Also, you can change the properties of components or controls that you will add to the forms.
- Now **to add the controls to your WinForms application** go to **Toolbox** tab present in the extreme left side of Visual Studio.
- Now drag and drop the controls that you needed on created Form. For example, if you can add TextBox, ListBox, Button etc. as shown below.

- To run the program you can use an **F5 key** or **Play button** present in the toolbar of Visual Studio. To stop the program you can use pause button present in the ToolBar. You can also run the program by going to **Debug->Start Debugging** menu in the menubar.



## 3. Web Application :

- A web application is store on remote server and derived from standard protocol over the internet is known as web application.
- An application that is usable only with an internet connection and that uses HTTP as its primary communications protocol also called web application.

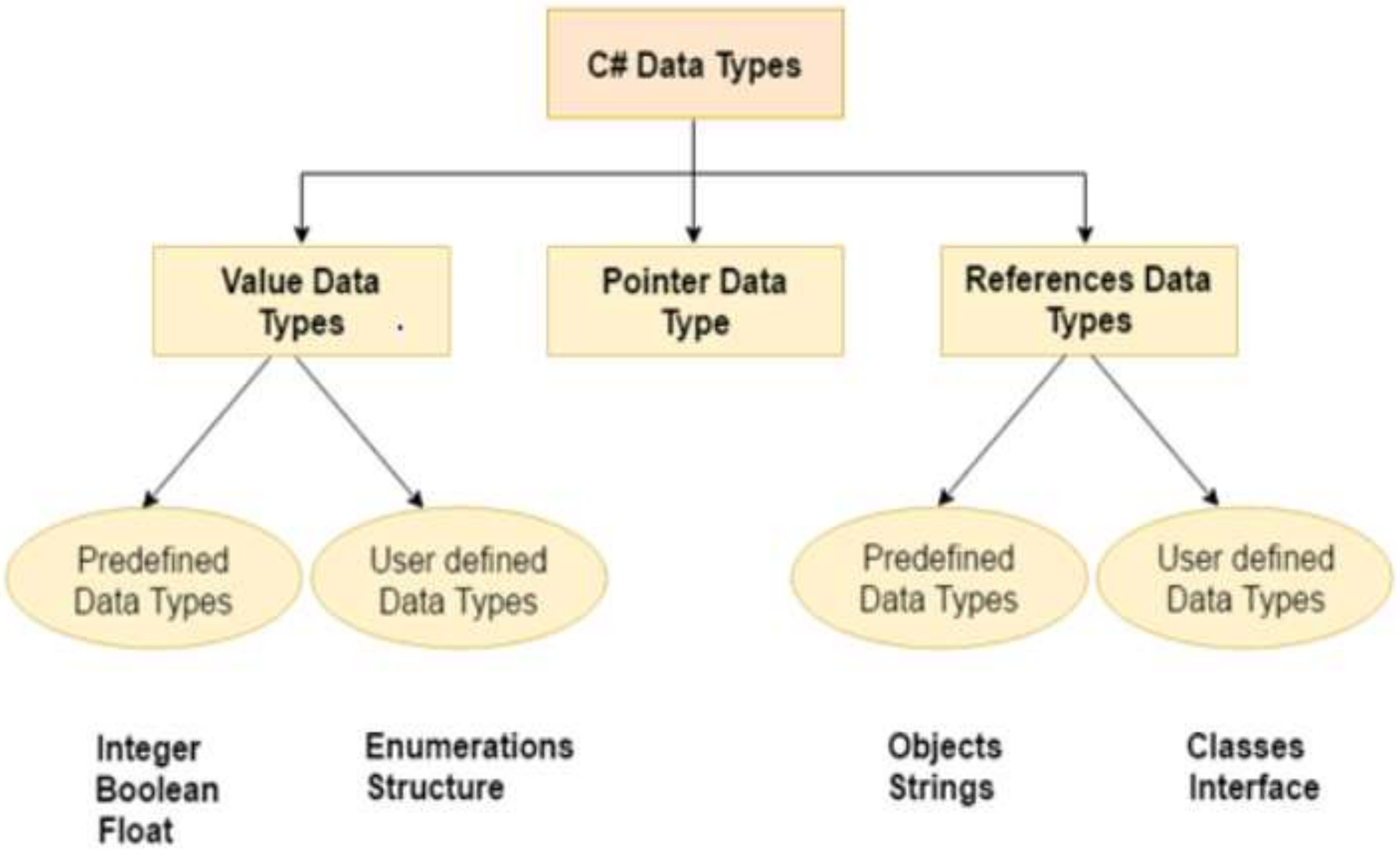
## 4. Setup File :

- Setup is the overall process of connecting and preparing a software program, hardware device or function properly.

## ❖ C# Data Types :

- A data type is a set of value and allowable operation on those value/their behaviour is known as data type.
- A data type specifies the size and type of variable values.
- It is important to use the correct data type for the corresponding variable; to avoid errors, to save time and memory, but it will also make your code more maintainable and readable.
- C# provides two types of data type :
  1. Primitive or predefined data type
  2. Non primitive or user defined data type

## Types of Data Types :



## Value Type Data type :

- Value type variables can be assigned a value directly. They are derived from the class **System.Value** Type.
- They can contain both signed and unsigned literals. Some of the value data types are int, float and char, which contain integers, floating point numbers and alphabets respectively.

| Data Types | Memory Size | Range |
|---|---|---|
| char or signed char | 1 byte | -128 to 127 |
| unsigned char | 1 byte | 0 to 127 |
| short or signed short | 2 byte | -32,768 to 32,767 |
| unsigned short | 2 byte | 0 to 65,535 |
| int or signed int | 4 byte | -2,147,483,648 to -2,147,483,647 |
| unsigned int | 4 byte | 0 to 4,294,967,295 |
| long or signed long | 8 byte | ?9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 |
| unsigned long | 8 byte | 0 - 18,446,744,073,709,551,615 |
| float | 4 byte | $1.5 * 10^{-45}$ - $3.4 * 10^{38}$, 7-digit precision |
| double | 8 byte | $5.0 * 10^{-324}$ - $1.7 * 10^{308}$, 15-digit precision |
| decimal | 16 byte | at least $-7.9 * 10^{?28}$ - $7.9 * 10^{28}$, with at least 28-digit precision |

## Example :

```
using System;
namespace demo
{
    class Program
    {
            static void Main()
            {
                    Console.WriteLine(sizeof(char));        //2
                    Console.WriteLine(sizeof(short));       //2
                    Console.WriteLine(sizeof(int));         //4
                    Console.WriteLine(sizeof(long));        //8
                    Console.WriteLine(sizeof(float));       //4
                    Console.WriteLine(sizeof(double));      //8
                    Console.WriteLine(sizeof(decimal));     //16
            }
    }
}
```

## Reference Data Type :

- The reference data types do not contain the actual data stored in a variable, but they contain a reference to the variables.
- If the data is changed by one of the variables, the other variable automatically reflects this change in value.
- Reference data type divided into three categories :
  1. Object Type
  2. Dynamic Type
  3. String Type

## 1. Dynamic Type :

- You can store any type of value in the dynamic data type variable. Type checking for these types of variables takes place at run-time.
- **Syntax :**

  dynamic <variable_name> = value; For example,

  dynamic d = 20;
- Dynamic types are similar to object types except that type checking for object type variables takes place at compile time, whereas that for the dynamic type variables takes place at run time.
- **Example :**

```
using System;
namespace DynamicType    {
        class Program      {
                static void Main() {
                        dynamic msg = "hello";
                        Console.WriteLine(msg);
                        msg=Console.ReadLine();
                        Console.WriteLine(msg);
                }
        }
}
```

## 2. Object Type :

- The object type is an alias for the **System.Object** class which is the base class for all data types in the C# CTS (Common Type System).
- We can assign values of any type to a variable that is of type object.

## Example :

```
using System;
namespace ObjectType
{
        class Program
        {
                static void Main()
                {
                        object message = "Hello World";
                        Console.WriteLine(message);
                        Console.ReadLine();
                }
        }
}
```

## 3. String Type :

- The **String Type** allows you to assign any string values to a variable. The string type is an alias for the System.String class. It is derived from object type.

## Example :

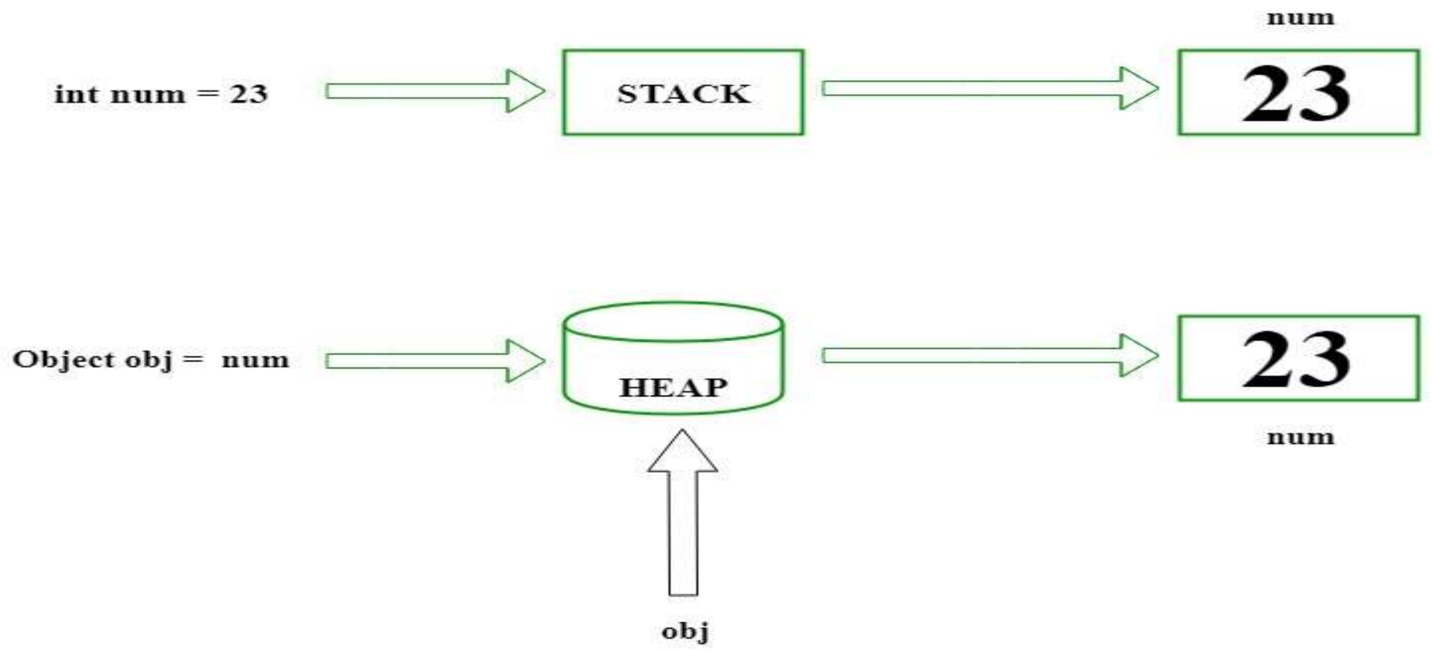
```
using System;
namespace StringType
{
        class Program
        {
                static void Main()
                {
                         string message = "Hello World";
                        Console.WriteLine(message);
                message += " hii";
                Console.WriteLine(message);
                Console.ReadLine();
                }
        }
}
```

## Pointer Data type :

- The Pointer Data Types will contain a memory address of the variable value.
- To get the pointer details we have a two symbols **ampersand (&)** and **asterisk (*)**.
- ***ampersand (&):*** It is Known as Address Operator. It is used to determine the address of a variable.
- ***asterisk (*):*** It also known as Indirection Operator. It is used to access the value of an address.
- **Syntax :**
- type* identifier;

## ❖ Boxing :

- The process of converting a **<u>Value Type</u> variable (char, int etc.) to a <u>Reference Type</u> variable (object)** is called **Boxing**.
- Boxing is an **implicit conversion** process in which object type (super type) is used.
- Value Type variables are always stored in Stack memory, while Reference Type variables are stored in Heap memory.

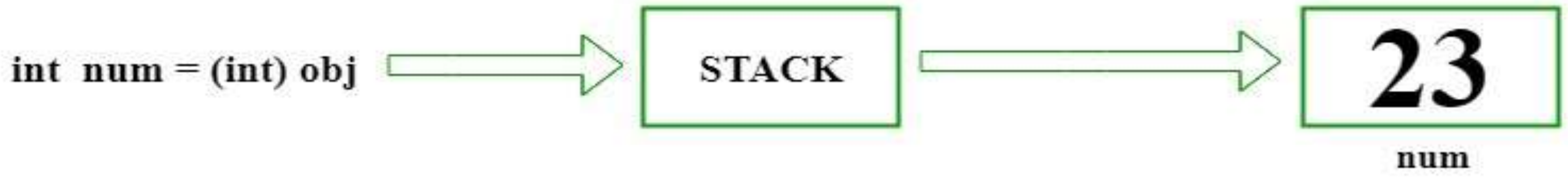


- **Example :**

```
using System;
namespace Boxing
{
   class Program
   {
       static void Main(String[ ] args)
       {
                int num = 23; // value type is int and assigned value 23
                Object Obj = num; // Boxing  (implicit)
                Console.WriteLine(num);
                Console.WriteLine(Obj);
                Console.ReadLine();
       }
   }
}
```

## ❖ Unboxing :

- The process of converting a **<u>Reference Type</u> variable into a <u>Value Type</u> variable** is known as **Unboxing**.
- It is an explicit conversion process.



- **Example :**

```
using System;
namespace Boxing
{
    class Program
    {
        static void Main(String[ ] args)
        {
                //int num = 23; // value type is int and assigned value 23
                //Object Obj = num; // Boxing  (implicit)
                Object Obj = 23;
                int i = (int)Obj; // Unboxing (Explicit)
                //Console.WriteLine(num);
                Console.WriteLine(Obj);
                Console.WriteLine(i);
                Console.ReadLine();
        }
    }
}
```

- In cases such as unboxing of a **null** object or casting the object as an incompatible data type, the program throws exceptions.

## Operators in C#:

- An operator is a symbol that tells the compiler to perform specific mathematical or logical manipulations. C# has rich set of built-in operators and provides the following type of operators −
- Arithmetic Operators
- Relational Operators
- Logical Operators
- Assignment Operators
- Misc Operators
- Bitwise Operators

# Arithmetic Operators :

- Arithmetic operators are used to perform arithmetic operations such as addition, subtraction, multiplication, division, etc.
- Assume variable **A** holds 10 and variable **B** holds 20 then −

| Operator | Description | Example |
|---|---|---|
| + | Adds two operands | A + B = 30 |
| - | Subtracts second operand from the first | A - B = -10 |
| * | Multiplies both operands | A * B = 200 |
| / | Divides numerator by de-numerator | B / A = 2 |
| % | Modulus Operator and remainder of after an integer division | B % A = 0 |
| ++ | Increment operator increases integer value by one | A++ = 11 |
| -- | Decrement operator decreases integer value by one | A-- = 9 |

# Relational Operators :

- Relational operators are used to check the relationship between two operands. If the relationship is true the result will be true, otherwise it will result in false.
- Assume variable **A** holds 10 and variable **B** holds 20, then −

| Operator | Description | Example |
|---|---|---|
| == | Checks if the values of two operands are equal or not, if yes then condition becomes true. | (A == B) is not true. |
| != | Checks if the values of two operands are equal or not, if values are not equal then condition becomes true. | (A != B) is true. |
| > | Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true. | (A > B) is not true. |
| < | Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true. | (A < B) is true. |
| >= | Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true. | (A >= B) is not true. |
| <= | Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true. | (A <= B) is true. |

## Logical Operators :

- Logical operators are used to perform logical operation such as and, or.
- Logical operators operates on boolean expressions (true and false) and returns boolean values.
- Logical operators are used in decision making and loops.
- Assume variable **A** holds Boolean value true and variable **B** holds Boolean value false, then −

| Operator | Description | Example |
|---|---|---|
| && | Called Logical AND operator. If both the operands are true then condition becomes true. | (A && B) is false. |
| \|\| | Called Logical OR Operator. If any of the two operands is True then condition becomes true. | (A \|\| B) is true. |
| ! | Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false. | !(A && B) is true. |

## Assignment Operators :

- Used to assigning a value to a variable.

| Operator | Description | Example |
|---|---|---|
| = | Simple assignment operator, Assigns values from right side operands to left side operand | C = A + B assigns value of A + B into C |
| += | Add AND assignment operator, It adds right operand to the left operand and assign the result to left operand | C += A is equivalent to C = C + A |
| -= | Subtract AND assignment operator, It subtracts right operand from the left operand and assign the result to left operand | C -= A is equivalent to C = C - A |
| *= | Multiply AND assignment operator, It multiplies right operand with the left operand and assign the result to left operand | C *= A is equivalent to C = C * A |
| /= | Divide AND assignment operator, It divides left operand with the right operand and assign the result to left operand | C /= A is equivalent to C = C / A |
| %= | Modulus AND assignment operator, It takes modulus using two operands and assign the result to left operand | C %= A is equivalent to C = C % A |

# Miscellaneous Operators :

- There are few other important operators including **sizeof, typeof** and **? :** supported by C#.

| **sizeof()** | **Returns the size of a data type.** | **sizeof(int), returns 4.** |
|---|---|---|
| typeof() | Returns the type of a class. | typeof(StreamReader); |
| & | Returns the address of an variable. | &a; returns actual address of the variable. |
| * | Pointer to a variable. | *a; creates pointer named 'a' to a variable. |
| ? : | Conditional Expression | If Condition is true ? Then value X : Otherwise value Y |

## Ternary Operators :

- The ternary operator ? : operates on three operands. It is a shorthand for if-then-else statement. Ternary operator can be used as follows:
- **Syntax :**

  variable = Condition? Expression1 : Expression2;
- The ternary operator works as follows:
- If the expression stated by Condition is true, the result of Expression1 is assigned to variable.
- If it is false, the result of Expression2 is assigned to variable.
- **Example :**

  result = (number % 2 == 0)? "Even Number" : "Odd Number";

## Bitwise Operators :

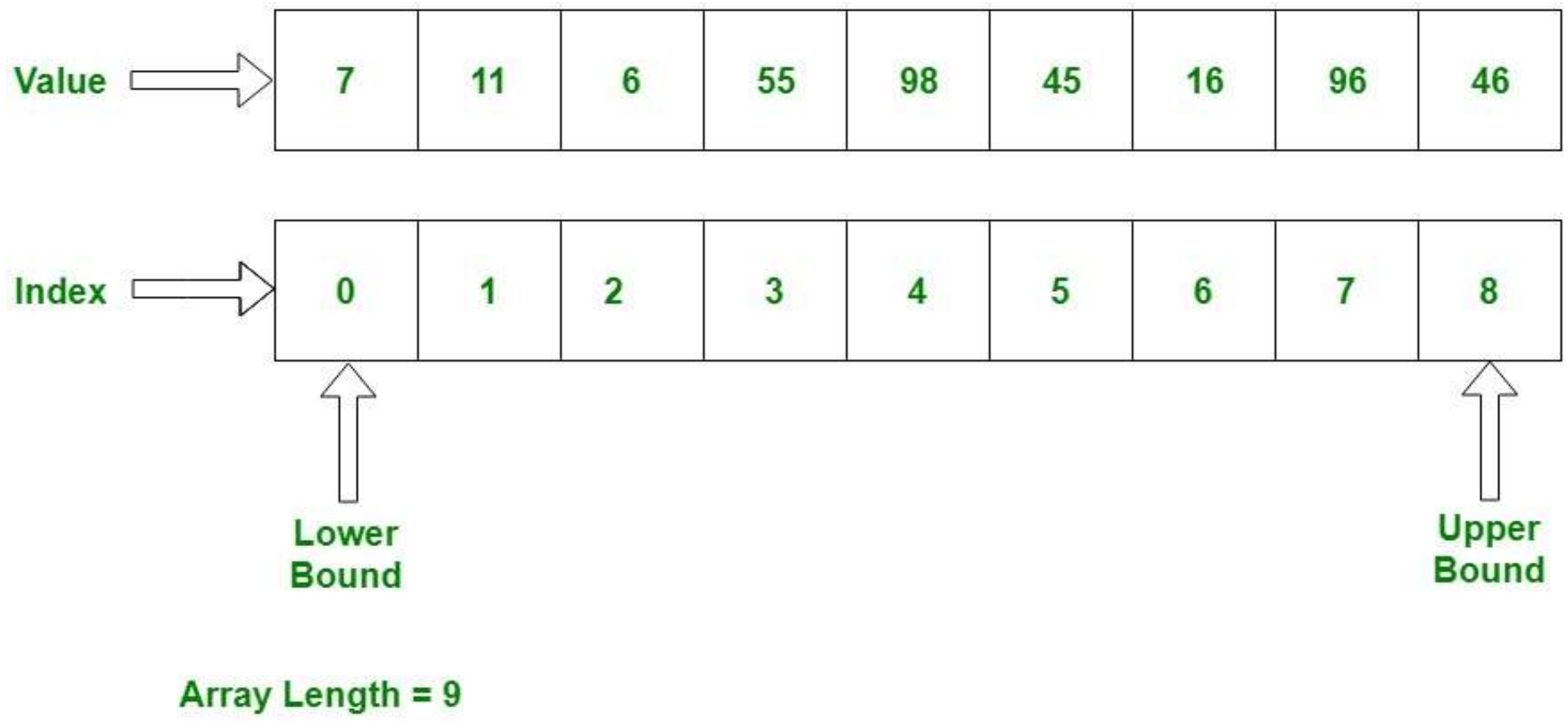
- Bitwise operator works on bits and perform bit by bit operation. The truth tables for &, |, and ^ are as follows −
- Assume if A = 60; and B = 13; then in the binary format they are as follows −
- A = 0011 1100
- B = 0000 1101
- -----------------
- A&B = 0000 1100
- A|B = 0011 1101
- A^B = 0011 0001
- ~A  = 1100 0011

| Operator | Description | Example |
|---|---|---|
| & | Binary AND Operator copies a bit to the result if it exists in both operands. | (A & B) = 12, which is 0000 1100 |
| \| | Binary OR Operator copies a bit if it exists in either operand. | (A \| B) = 61, which is 0011 1101 |
| ^ | Binary XOR Operator copies the bit if it is set in one operand but not both. | (A ^ B) = 49, which is 0011 0001 |
| ~ | Binary Ones Complement Operator is unary and has the effect of 'flipping' bits. | (~A ) = -61, which is 1100 0011 in 2's complement due to a signed binary number. |
| << | Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand. | A << 2 = 240, which is 1111 0000 |
| >> | Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand. | A >> 2 = 15, which is 0000 1111 |

## ❖ Array in C# :

- An array is a collection of similar types of data.
- An array is a group of like-typed variables that are referred to by a common name.
- And each data item is called an element of the array.
- The data types of the elements may be any valid data type like char, int, float, etc. and the elements are stored in a contiguous location.
- **Important Points to Remember About Arrays in C# :**
  - In C#, all arrays are dynamically allocated.
  - Since arrays are objects in C#, we can find their length using member length. This is different from C/C++ where we find length using sizeof operator.
  - A C# array variable can also be declared like other variables with [] after the data type.
  - The variables in the array are ordered and each has an index beginning from 0.
  - C# array is an object of base type **System.Array**.

- The array has can contain primitive data types as well as objects of a class depending on the definition of an array. Whenever use primitives data types, the actual values have to be stored in contiguous memory locations. In the case of objects of a class, the actual objects are stored in the heap segment.



## Single Dimensional Array :

- To create single dimensional array, you need to use square brackets [] after the type.

  **int**[] arr = **new int**[5];//creating array

- You cannot place square brackets after the identifier.

  Like : **int** arr[] = **new int**[5];

  //It will throw compile time error

- **Other valid Declaration and Initialization at same time :**
- There are 3 ways to initialize array at the time of declaration.

  **int**[] arr = **new int**[5]{ 10, 20, 30, 40, 50 };

- We can omit the size of array.

  **int**[] arr = **new int**[]{ 10, 20, 30, 40, 50 };

- We can omit the new operator also.

  **int**[] arr = { 10, 20, 30, 40, 50 };

```
using  System;
namespace array
{
    class Program
    {
        static void Main(string[] args)
        {
            int[] a={14,21,33};
            //int []a=new int[3];
            for (int i = 0; i < a.Length; i++)
            {
                Console.WriteLine("a[{0}] : {1}",i,a[i]);
            }
            /*foreach(int i in a)
            {
                Console.WriteLine(i);
            }*/
            Console.ReadLine();
        }
    }
}
```

## Multi- Dimensional Array :

- The multidimensional array is also known as **rectangular** arrays in C#.
- It can be two dimensional or three dimensional.
- The data is stored in tabular form (row * column) which is also known as matrix.
- To create multidimensional array, we need to use comma inside the square brackets.
- It contains more than one comma (,) within single rectangular brackets (“[ , , ,]”). To storing and accessing the elements from a multidimensional array, you need to use a nested loop in the program.
- **Syntax :**

  Datatype [ , ] variable_name=new DataType[row,column];
- **int**[,] arr=**new int**[3,5];//declaration of 2D array
- **int**[,,] arr=**new int**[3,2,4];//declaration of 3D array

```
using System;
namespace array
{
 class Program
 {
    static void Main(string[] args)
    {
       int [,]a=new int[2,3];
       for(int i=0;i<a.GetLength(0);i++)
       {
           for (int j = 0; j < a.GetLength(1);j++ )
           {
           a[i, j] =
    Convert.ToInt32(Console.ReadLine());
           }
        }
for (int i = 0; i < a.GetLength(0); i++)
{
   for (int j = 0; j < a.GetLength(1); j++)
   {
   Console.WriteLine("a[{0},{1}]{2} ",i,j,a[i, j]);
   }
 }

            /*foreach(int data in a)
            {
                Console.WriteLine(data);
            }*/

            Console.ReadLine();
        }
    }
}
```

## Jagged Array :

- A jagged array is an array whose elements are arrays, possibly of different sizes.
- In other words, the length of each array index can differ.
- A jagged array is sometimes called an "array of arrays".
- Its elements are reference types and are initialized to null.
- **Syntax :**

  data_type[ ][ ] name_of_array = new data_type[rows][ ];
- **Example :**

  int[][] jagged_arr = new int[4][ ];
- In the above example, a single-dimensional array is declared that has 4 elements(rows), each of which is a 1-D array of integers.

```
using System;
namespace array
{
    class Program
    {
        static void Main(string[] args)
        {
            int [][]jag=new int[3][];
            jag[0] = new int[3]{1,2,3};
            jag[1] = new int[4]{4,5,6,7};
            jag[2] = new int[2]{8,9};
            for (int i = 0; i < jag.Length;i++)
            {
                for(int j=0;j<jag[i].Length;j++)
                {
                    Console.Write(jag[i][j]+" ");
                }
                Console.WriteLine();
            }
        }
    }
}
```

## Decision Making/ Conditional Structure /Flow Control Statements :

- C# allows us to perform actions based on some type of conditions that may be logical or comparative. Based on the result of these conditions i.e., either TRUE or FALSE, an action would be performed as asked by the user. It's just like a two- way path. If you want something then go this way or else turn that way. To use this feature, C# provides us with four conditional statements:
- **if** statement
- **if…else** statement
- **else if ladder**
- **switch** statement

## if Statement :

- This statement allows us to set a condition. On being TRUE, the following block of code enclosed within the if clause will be executed.
- **Syntax** :

```
        if (condition){ // if TRUE then execute this code }
```

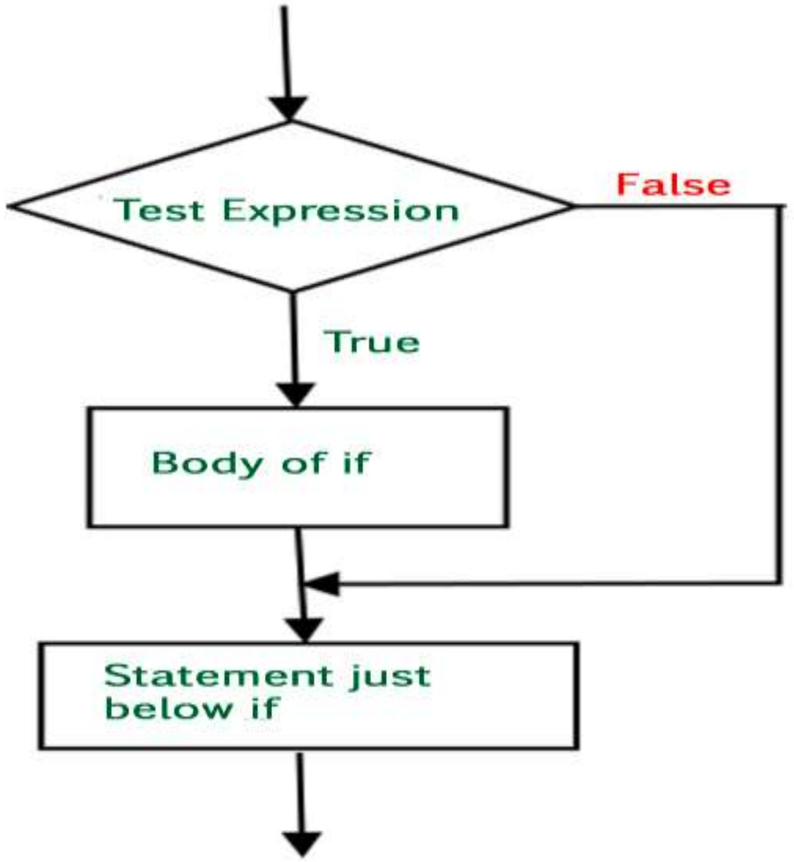
- **Example :**

```
   int x = 12;
if (x > 0) {
   Console.WriteLine( "The number is positive“);
}
```

- **Output :**

```
The number is positive
```

## Flowchart :


Test Expression
False
True
Body of if
Statement just below if

## if...else Statement :

- We understood that if a condition will hold i.e., TRUE, then the block of code within if will be executed. But what if the condition is not TRUE and we want to perform an action? This is where else comes into play. If a condition is TRUE then if block gets executed, otherwise else block gets executed.
- **Syntax**:

```
if (condition)
{
        // if TRUE then execute this code
}
else
{
        // if FALSE then execute this code
}
```

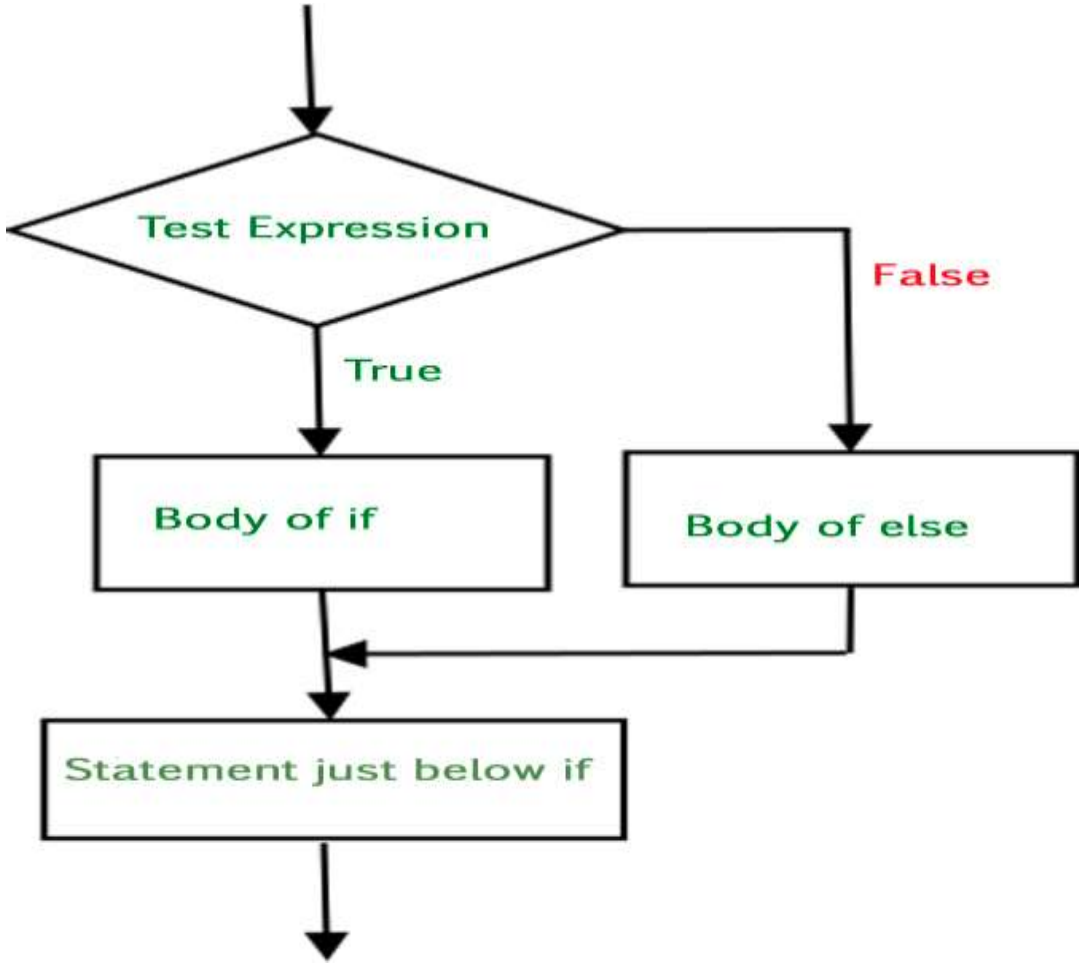
- **Example :**

```
int x = -12;
if (x > 0) {
    Console.WriteLine("The number is positive“);
}
else{
   Console.WriteLine("The number is negative“);
}
```

- **Output :**

```
The number is negative
```

## Flowchart :


Test Expression
False
True
Body of if
Body of else
Statement just below if

## else if ladder :

- This allows us to use multiple if…else statements. We use this when there are multiple conditions of TRUE cases.

**Syntax**:

```
if (condition) {
 // if TRUE then execute this code }
 else if(Condition){
 // if TRUE then execute this code }
 else if(Condition) {
 // if TRUE then execute this code }
 else {
 // if FALSE then execute this code }
```

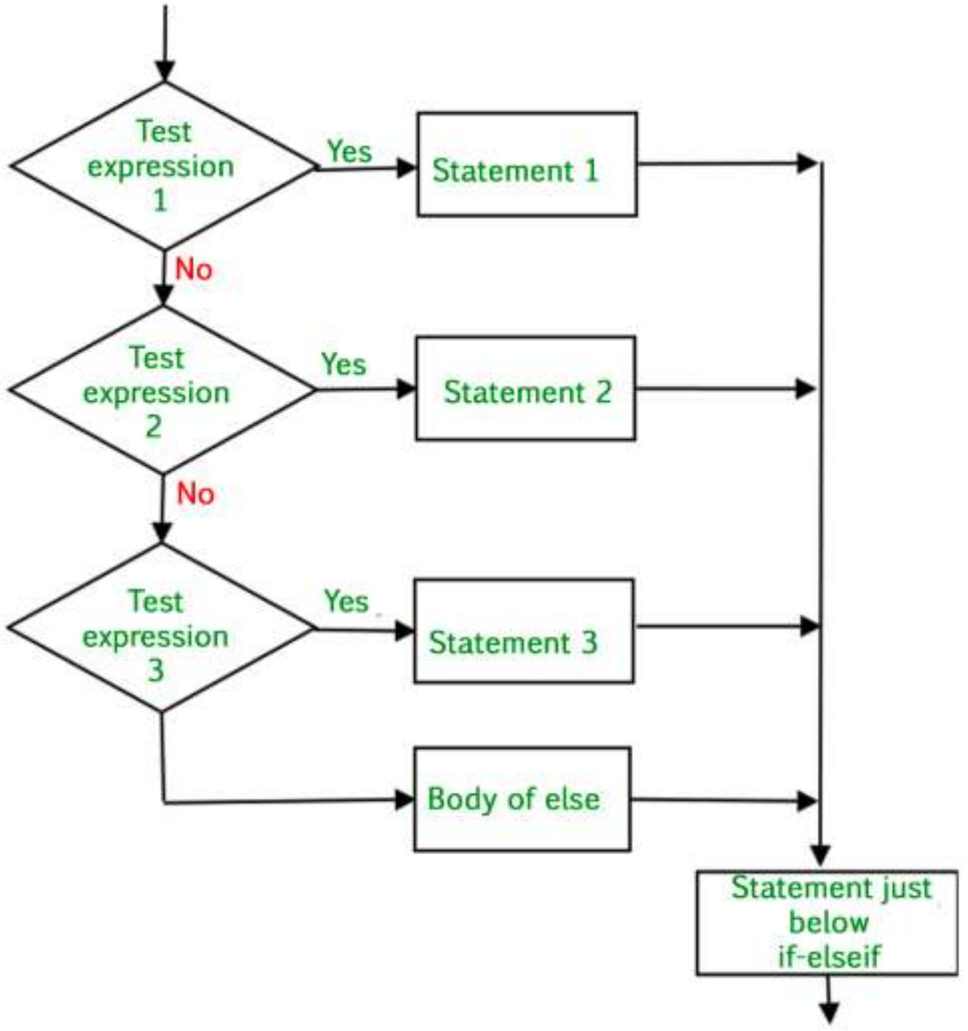
- **Example :**

```
string x = "August";
if (x == "January") {
    Console.WriteLine("Happy Republic Day“);
}
else if (x == "August") {
    Console.WriteLine("Happy Independence Day!!!“);
}
else{
    Console.WriteLine("Nothing to show“);
}
```

- **Output :**

```
Happy Independence Day!!!
```

## Flowchart :


Test expression 1
Yes
Statement 1
No
Test expression 2
Yes
Statement 2
No
Test expression 3
Yes
Statement 3
Body of else
Statement just below if-elseif

## switch Statement :

- The “switch” performs in various cases i.e., it has various cases to which it matches the condition and appropriately executes a particular case block. It first evaluates an expression and then compares with the values of each case. If a case matches then the same case is executed.
- To use switch, we need to get familiar with two different keywords
  namely, **break** and **default**. The **break** statement is used to stop the automatic control flow into the next cases and exit from the switch case.
- The **default** statement contains the code that would execute if none of the cases match.

## Syntax :

```
switch(n) {
 case statement1:
            code to be executed if n==statement1;
            break;
 case statement2:
            code to be executed if n==statement2;
            break;
 case statement3:
            code to be executed if n==statement3;
           break;
 ......
 default:
           code to be executed if n != any case;
           break;
```

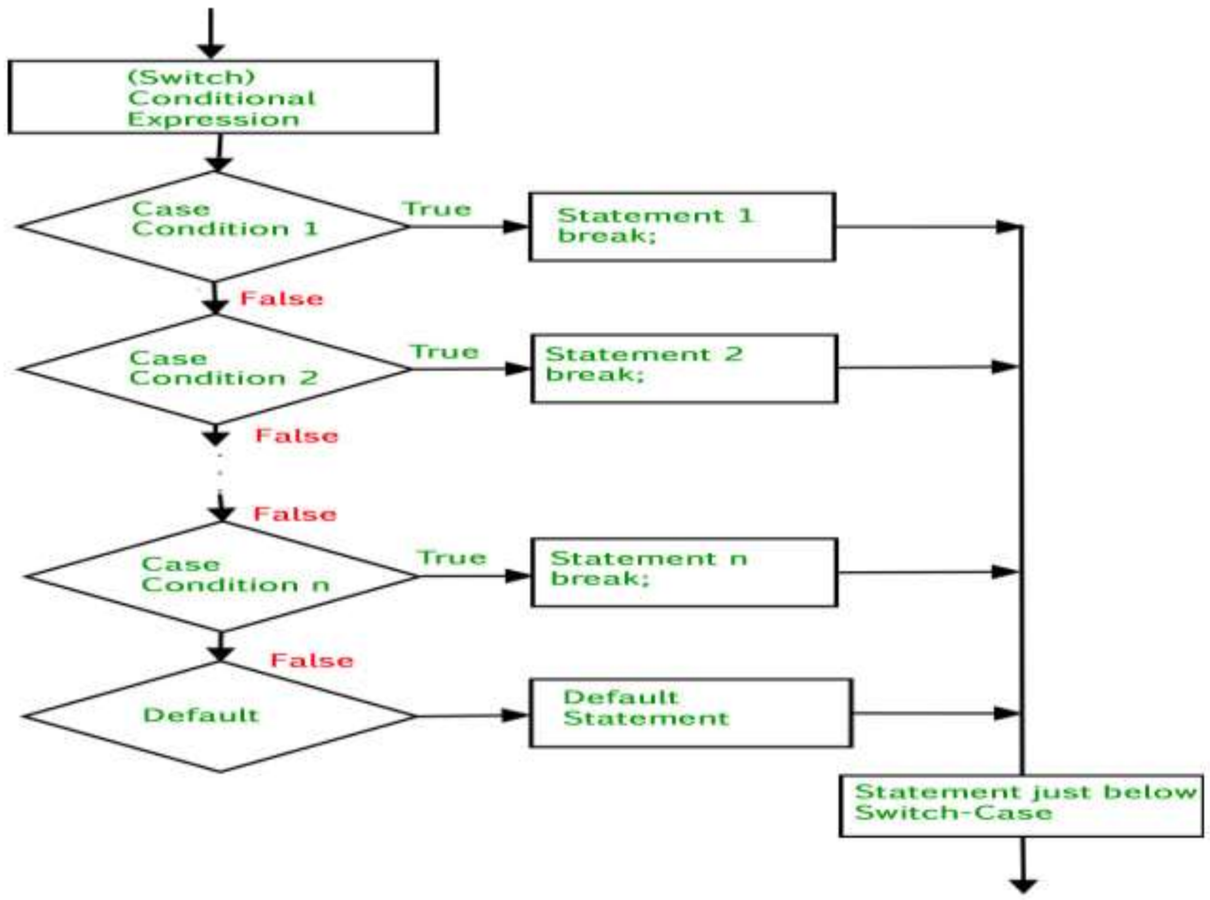
- **Example :**

```
char n= ‘A’;
switch(n) {
    case ‘B’:
              Console.WriteLine("Its B“);
             break;
    case  ‘C’:
              Console.WriteLine("Its C“);
             break;
    case  ‘A’:
              Console.WriteLine("Its A“);
             break;
    default:
              Console.WriteLine("Doesn't exist“);
             break;
}
```

**Output :**

```
Its A
```

## Flowchart :


(Switch) Conditional Expression
Case Condition 1
True
Statement 1 break;
False
Case Condition 2
True
Statement 2 break;
False
False
Case Condition n
True
Statement n break;
False
Default
Default Statement
Statement just below Switch-Case

## Looping Structure in C# :

- Loops in C# are used to execute the same block of code a specified number of times. C# supports following four loop types.
- **for** − loops through a block of code a specified number of times.
- **while** − loops through a block of code if and as long as a specified condition is true.
- **do...while** − loops through a block of code once, and then repeats the loop as long as a special condition is true.
- **foreach** − loops through a block of code for each element in an array.

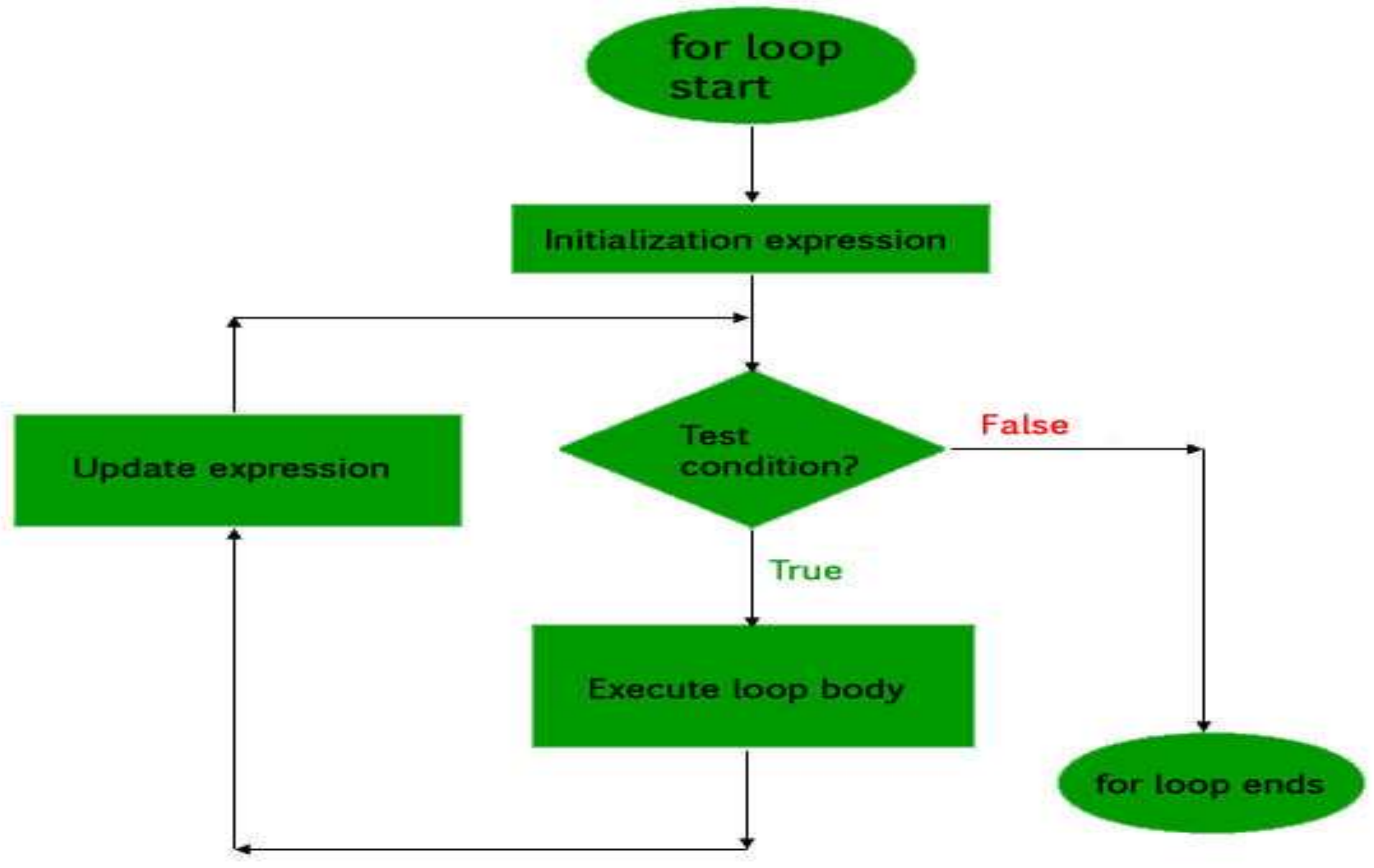
## The for loop statement :

- The for statement is used when you know how many times you want to execute a statement or a block of statements.
- **Syntax :**

```
for (initialization expression; test condition; update expression){
        code to be executed;
}
```

- In for loop, a loop variable is used to control the loop. First initialize this loop variable to some value, then check whether this variable is less than or greater than counter value. If statement is true, then loop body is executed and loop variable gets updated . Steps are repeated till exit condition comes.
- **Initialization Expression**: In this expression we have to initialize the loop counter to some value. for example: num = 1;
- **Test Expression**: In this expression we have to test the condition. If the condition evaluates to true then we will execute the body of loop and go to update expression otherwise we will exit from the for loop. For example: num <= 10;
- **Update Expression**: After executing loop body this expression increments/decrements the loop variable by some value. for example: num += 2;

## Flowchart :


for loop start
Initialization expression
Test condition?
False
True
Update expression
Execute loop body
for loop ends

- **Example :**

```
// code to illustrate for loop
for (int num = 1;  num <= 10;  num += 2) {
    Console.WriteLine( num);
}
```

- **Output :**

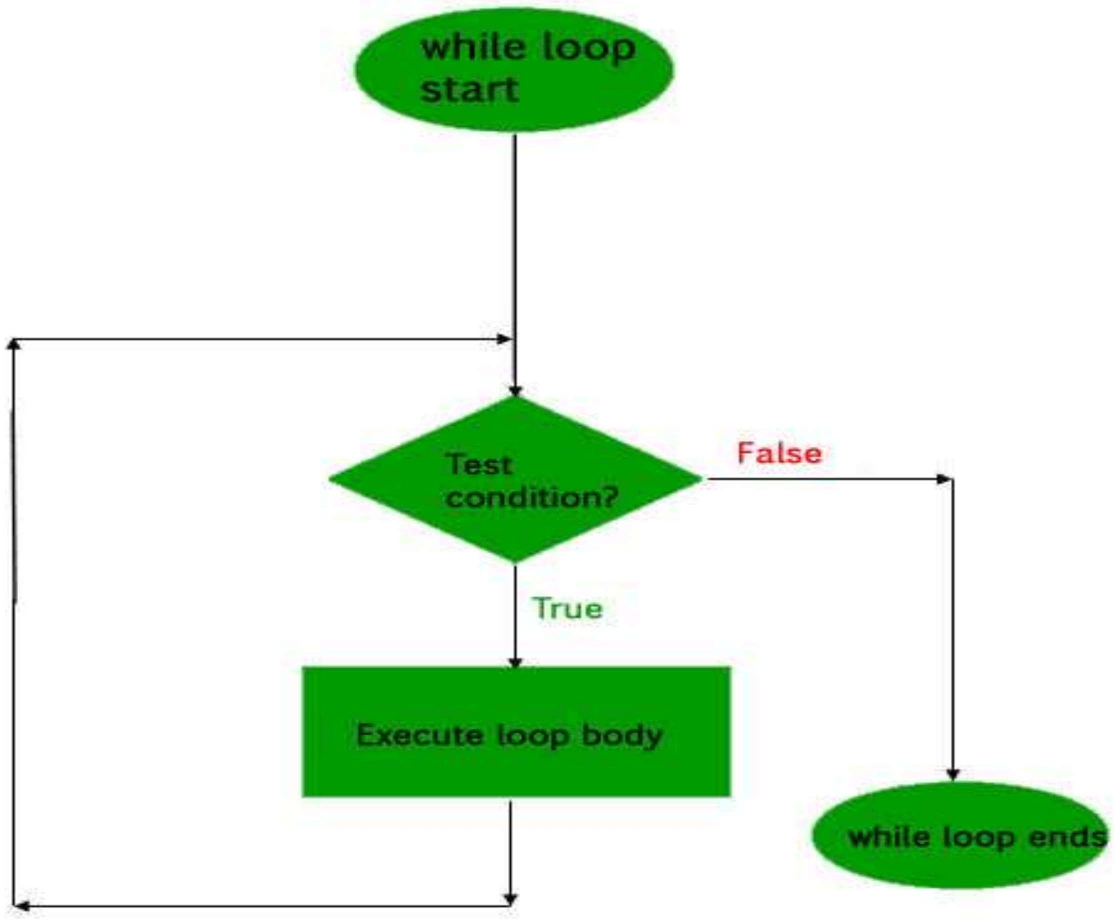
```
1
 3
 5
 7
 9
```

## while loop :

- The while loop is also an entry control loop like for loops i.e., it first checks the condition at the start of the loop and if its true then it enters the loop and executes the block of statements, and goes on executing it as long as the condition holds true.
- **Syntax**:

```
while (if the condition is true)
{
        // code is executed
}
```

## Flowchart :


while loop start
Test condition?
False
True
Execute loop body
while loop ends

**Example :**

```
// C# code to illustrate while loops
int num = 2;
while ( num < 12) {
     num += 2;
    Console.WriteLine(num);
}
```

**Output :**

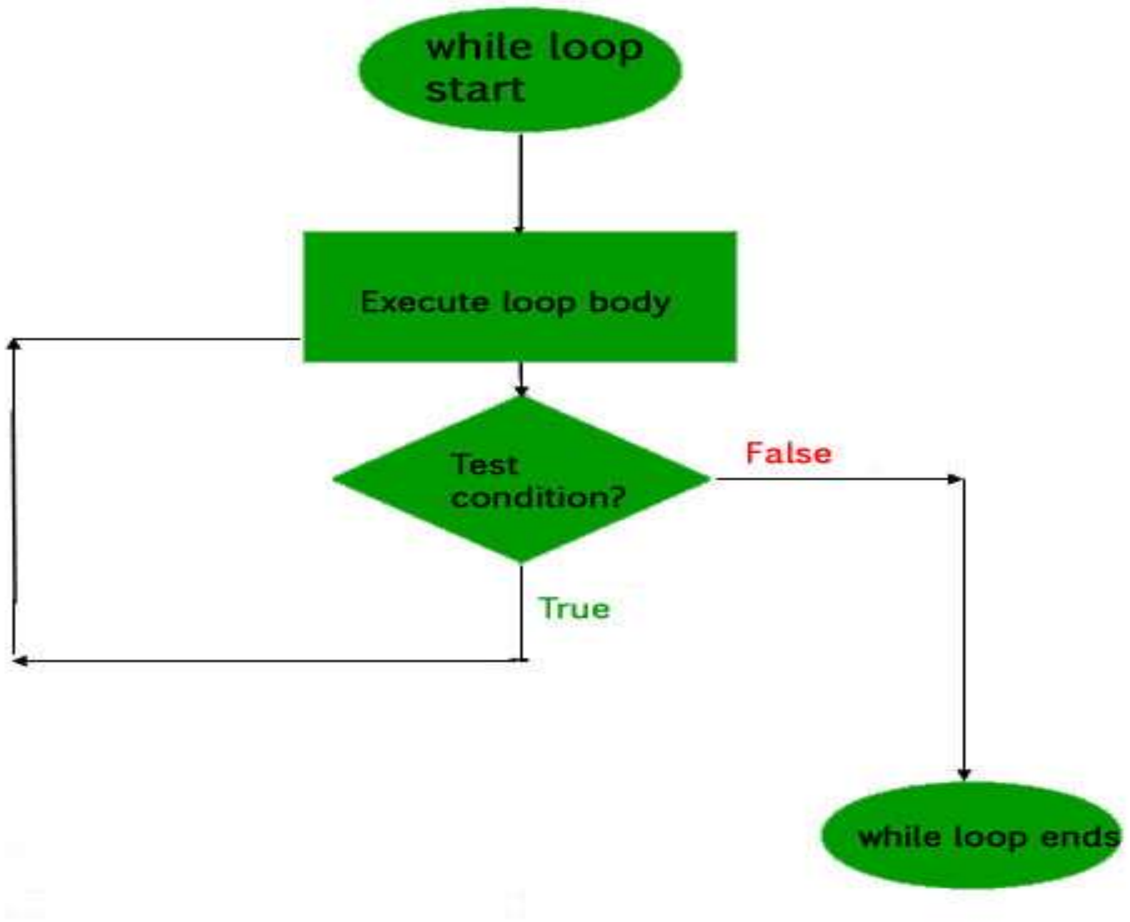
```
4
6
8
10
12
```

## do-while loop :

- This is an exit control loop which means that it first enters the loop, executes the statements, and then checks the condition. Therefore, a statement is executed at least once on using the do…while loop. After executing once, the program is executed as long as the condition holds true.
- **Syntax**:

```
do {
        //code is executed
} while (if condition is true);
```

## Flowchart :


while loop start
Execute loop body
Test condition?
False
True
while loop ends

- **Example :**

```
// C# code to illustrate do...while loops
 int num = 2;
do {
     num += 2;
    Console.WriteLine(num);
} while ( num < 12);
```

- **Output :**

```
4
6
8
10
12
```

## foreach loop :

- This loop is used to iterate over arrays. For every counter of loop, an array element is assigned and the next counter is shifted to the next element.

- **Syntax**:

```
foreach (array_element in value)
 {
        //code to be executed
  }
```

- **Example :**

```
 int [ ]val = {10, 20, 30, 40, 50, 60};
foreach (int a in  val) {
        Console.WriteLine(a);
}
 string [ ]arr = {"Ram", "Laxman", "Sita“};
foreach ( string val in arr) {
        Console.WriteLine( val);
}
```

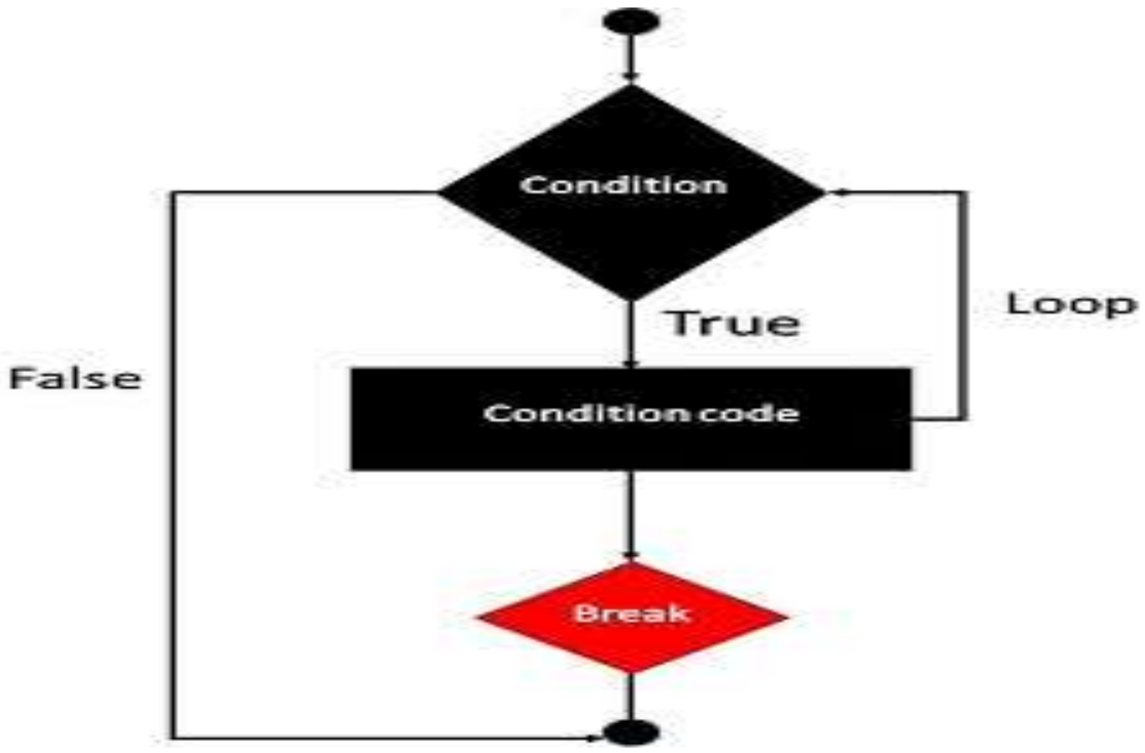
- **Output :**

```
10
 20
 30
 40
 50
 60
 Ram
 Laxman
Sita
```

## break Statement :

- The C# **break** keyword is used to terminate the execution of a loop prematurely.
- The **break** statement is situated inside the statement block. It gives you full control and whenever you want to exit from the loop you can come out. After coming out of a loop immediate statement to the loop will be executed.

## Flowchart :



- **Example :**

```
int i = 0;
   while(  i < 10)
   {
            i++;
                    if(  i == 3 )
                              break;
   }
 Console.WriteLine("Loop stopped at i = “,i );
```

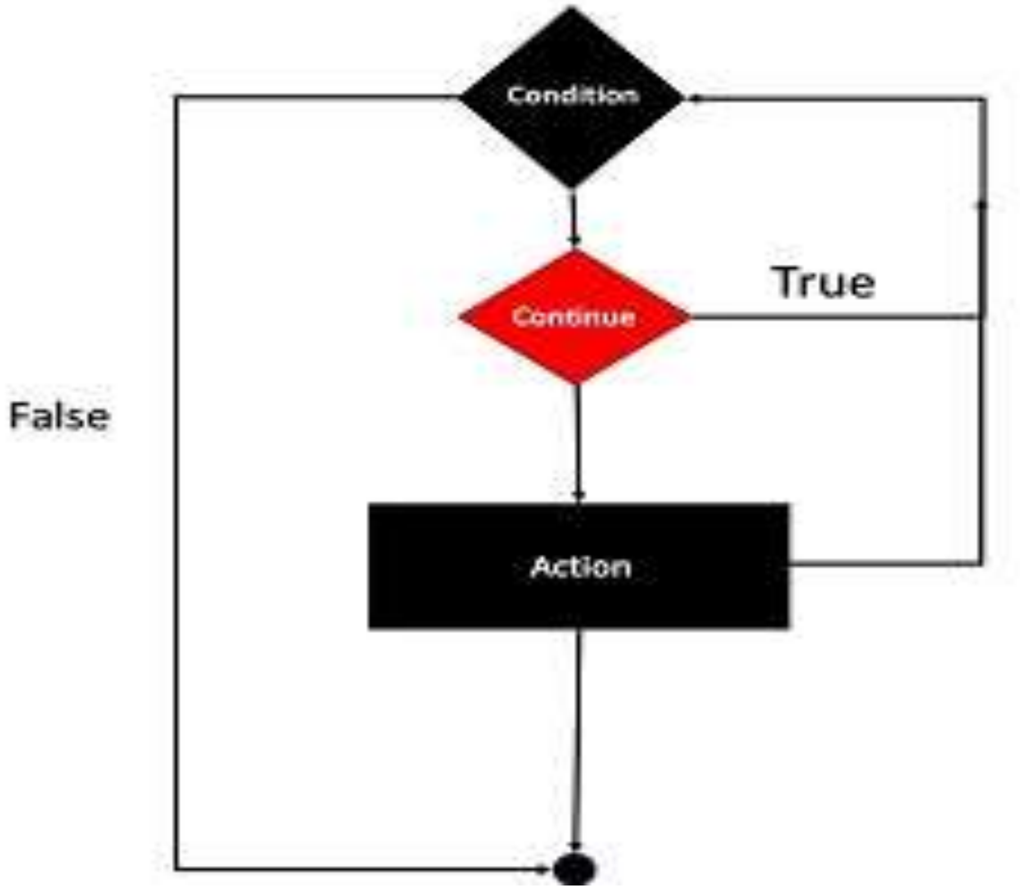
- **Output :**

```
Loop stopped at i = 3
```

## continue statement :

- The C# **continue** keyword is used to halt the current iteration of a loop but it does not terminate the loop.
- Just like the **break** statement the **continue** statement is situated inside the statement block containing the code that the loop executes, preceded by a conditional test. For the pass encountering **continue** statement, rest of the loop code is skipped and next pass starts.

## Flowchart :



- **Example :**

```
 int [ ]array = {1,2,3,4,5};
foreach(int value in  array )
{
        if(value == 3 )
                continue;
        Console.WriteLine(value);
}
```

- **Output :**

```
1
2
4
5
```

## Difference between break and continue :

| **Break** | **Continue** |
|---|---|
| The break statement is used to jump out of a loop. | The continue statement is used to skip an iteration from a loop |
| Its syntax is -:<br>**break;** | Its syntax is -:<br>**continue;** |
| The break is a keyword present in the language | continue is a keyword present in the language |
| It can be used with loops ex-: for loop, while loop. | It can be used with loops ex-: for loop, while loop. |
| The break statement is also used in switch statements | We can use **continue** with a switch statement to skip a case. |

## Some Common Namespaces :

| Namespace | What It Contains | Example Classes and Subnamespaces |
|---|---|---|
| `System.Collections` | Creation and management of various types of collections | `Arraylist`, `Hashtable`, `SortedList` |
| `System.Data` | Classes and types related to basic database management (see Chapter 11 for details) | `DataSet`, `DataTable`, `DataColumn`, |
| `System.Diagnostics` | Classes to debug an application and to trace the execution of code | `Debug`, `Trace` |
| `System.IO` | Types that allow reading and writing to and from files and other data streams | `File`, `FileStream`, `Path`, `StreamReader`, `StreamWriter` |
| `System.Math` | Members to calculate common mathematical quantities, such as trigonometric and logarithmic functions | `Sqrt` (square root), `Cos` (cosine), `Log` (logarithm), `Min` (minimum) |
| `System.Reflection` | Capability to inspect metadata | `Assembly`, `Module` |
| `System.Security` | Types that enable security capabilities (see Chapter 24 for details) | `Cryptography`, `Permissions`, `Policy` |

## 11 Programming Languages which are designed and developed by Microsoft are:

1. C#.NET
2. VB.NET
3. C++.NET
4. J#.NET
5. F#.NET
6. JSCRIPT.NET
7. WINDOWS POWERSHELL
8. IRON RUBY
9. IRON PYTHON
10. C OMEGA
11. ASML(Abstract State Machine Language)

## Links must visit :

More about Console Application :

https://www.c-sharpcorner.com/article/creating-console-application-in-c-sharp/

More about Windows Application :

https://www.guru99.com/c-sharp-windows-forms-application.html

More about Operators :

https://www.tutorialspoint.com/csharp/csharp_operators.htm

More about is-else :

https://www.tutorialspoint.com/csharp/csharp_decision_making.htm

https://www.geeksforgeeks.org/c-sharp-decision-making-else-else-ladder-nested-switch-nested-switch/?ref=lbp

More about Loops :

https://www.tutorialspoint.com/csharp/csharp_loops.htm

https://www.geeksforgeeks.org/loops-in-c-sharp/?ref=lbp

Foreach Loop :

https://www.geeksforgeeks.org/c-sharp-foreach-loop/

- More about Boxing and Unboxing :

https://www.geeksforgeeks.org/c-sharp-boxing-unboxing/

## Assignment Questions :

1. Write Down all full form of UNIT – 1.
2. History of c#
3. What is .net framework ? Write features of .net
4. Explain CLR.
5. Explain FCL.
6. Explain CTS.
7. Explain CLS.
8. Explain MSIL.
9. What is assembly ?
10. What is meta data?
11. Explain managed code and unmanage code in C#.
12. What is Boxing and Unboxing in C# ?
13. What is value type and reference type in C#?
14. Explain 2D array and jagged array in c#.
15. Explain foreach() loop.
16. Explain Types of IDE.
17. What is JIT compile and its types.