.Net

Ch. 01

.NET Framework and Visual Studio IDE

Syllabus C#

- .NET Framework and Visual Studio IDE, Language Basics
  - Introduction to .NET Framework
  - Features / Advantages
  - CLR, CTS and CLS
  - BCL / FCL / Namespace
  - Assembly and MetaData
  - JIT and types
  - Managed Code and Unmanaged Code
  - Introduction to .NET Framework and IDE versions.

Syllabus C#

- Different components (windows) of IDE.
- Types of Projects in IDE (Console, Windows, Web, Setup, etc.)
- Data Types (Value Type & Reference Type)
- Boxing and UnBoxing
- Operators (Arithmetic, Relational, Bitwise, etc.)
- Array (One Dimensional, Rectangular, Jagged)
- Decisions (If types and swith case)
- Loop (for, while, do…while, foreach)

Introduction .NET Framework

- Before starting the .NET framework we should know the definition of framework.
- A framework is a collection of classes, and applications which are libraries of SDKs (Software Development Kit) and APIs (Application Programming Interface) to help the different components all work together.

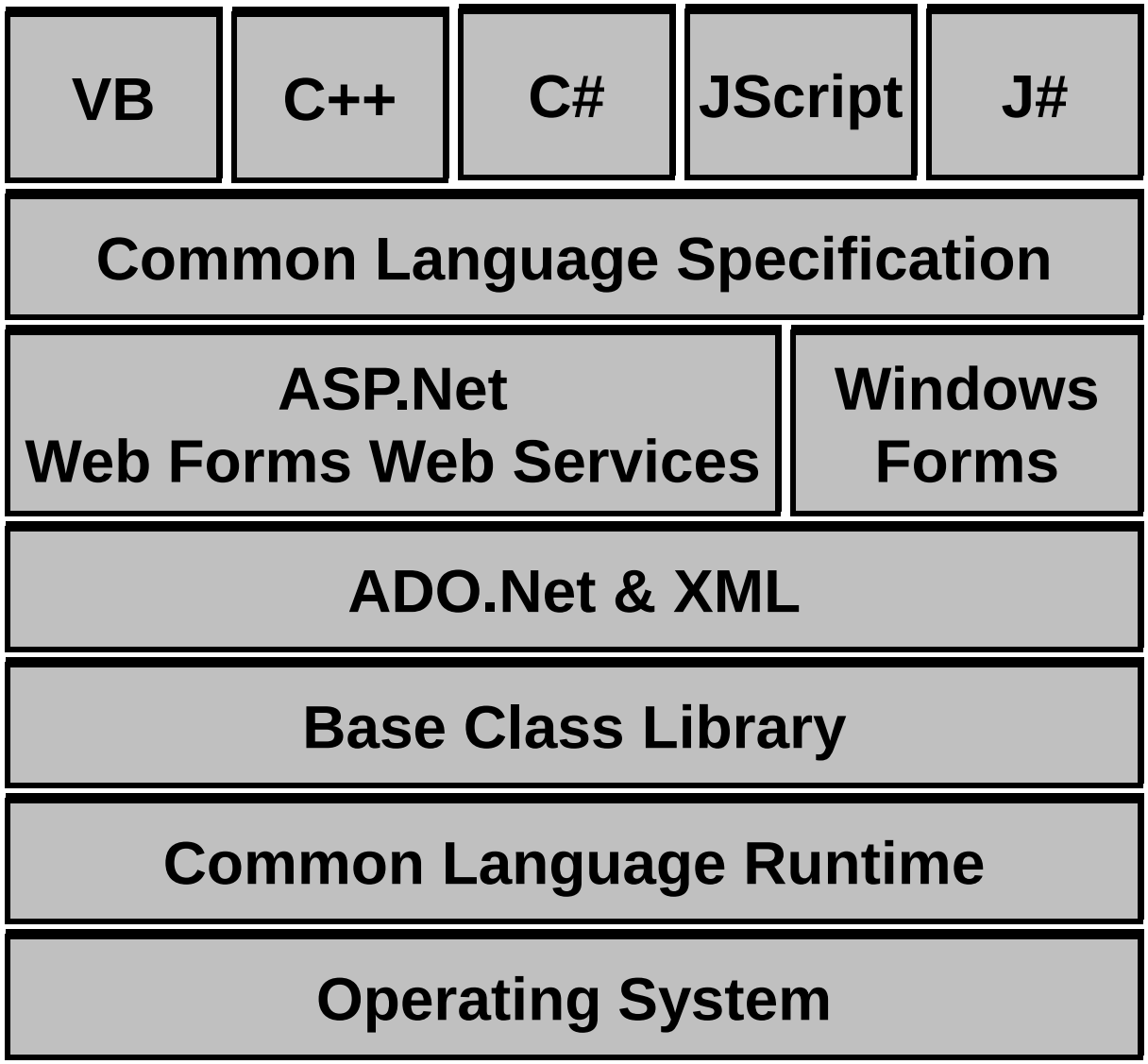
.NET Framework

- .NET is a framework which is used for developing web based and window based application within the ***Microsoft environment.***
- It provides a large library and supports ***all the programming languages available in the market.***
- It also allows language interoperability (ક્રિયાશીલતા) which is one of the most important features of .NET.

.NET Framework

- Programs that are written in .NET framework are executed in a software environment, known as **C**ommon **L**anguage **R**untime (CLR).
- CLR is a virtual machine that provides services such as security, memory management and exception handling.
- The Class library and the CLR together constitute (રચના) the .NET framework.

.NET Framework



.NET Framework

- The .NET Framework's Base Class Library provides user interface, data access, database connectivity, cryptography, web application development, numeric algorithms and network communications.
- Programmers produce software by combining their own source code with the .NET Framework and other libraries.
- Microsoft also produces an integrated development environment largely for .NET software called Visual Studio.

Features Of .Net Platform

- .Net is very famous because of its features.
  - Multilanguage Development
  - Platform and process independence
  - Automatic memory management
  - Easy Deployment
  - Distributed Architecture
  - Interoperability with Unmanaged code
  - Security
  - Performance and Scalability (સુગમતા) etc.

Features Of .Net Platform :

- Multilanguage Development
  - In .NET user can create application in many language.
  - We can create our application in VB, C#, C++ and many more programming languages.
  - .NET platform allows languages to be integrated (સંકલિત) with one another through the use of MSIL (**Microsoft Intermediate Language**).

Features Of .Net Platform :

- Platform and Processor independence :
  - When we compile any application in .Net platform first it is converted into MSIL code.
  - This MSIL code is CPU-Independent and is higher than any machine language.
  - MSIL code can be run on any platform and processor which supports CLR (Common Language Runtime). Because of this .NET framework is platform and processor independence.

Features Of .Net Platform :

- Automatic Memory Management
  - We have already used programs like C and C++, at that time we were allocation memory to particular object using different function and we released that memory when program is terminated.
  - In .NET environment Microsoft is providing facility to make developing easier.
  - It provides a component called **Garbage Collector** which handles this memory management task.
  - When any object required memory it automatically allocates and when it is no longer valid then garbage collector will automatically free up those memory to reuse.

Features Of .Net Platform :

- Easy deploying (Installing)
  - Once any type of application is developed, programmers always worried about how they deploy(install) their application.
  - Most of the companies use third party software to build their installation.
  - .NET application doesn’t require any extra activity than copying their files to a directory.
  - For uninstalling an application will be easy as deleting those files.

Features Of .Net Platform :

- Distributed Architecture
  - Advanced applications or websites which are presented to user come from different sources like servers located at different place.
  - Different application running on this server which fetches data from many database.
  - This is called as **distributed architecture.**
  - Application with distributed architecture are very complex to build and maintain.
  - .NET provides architecture for developing this type of complex application using different concept like XML, SOAP and UDDI.

Features Of .Net Platform :

- Interoperability with Unmanaged Code
  - The code which is executed by CLR is known as **Managed Code** and the code which is not executed by CLR is known as **Unmanaged Code**.
  - However, this code is still run by the CLR but we can't take the advantage like CTS and automatic memory management.
  - There are many situations in that you must work with code that is outside from .NET platform.

Features Of .Net Platform :

- Security :
  - To write secure code and make administrators to customize the permissions granted to code the CLR and the .NET framework provide many useful classes and services so that it can access protected resources.
  - Moreover, the CLR and the .NET framework provide useful classes and services that facilitate the use of cryptography (સંકેતલિપી) and role-based security.

Features Of .Net Platform :

- Performance and Scalability (સુગમતા) :
  - There is no magic tool that will allow a poorly designed application to scale and perform well.
  - The .NET framework gives you tools that make it easier to design performing software.

Advantages of .NET

- .NET is new generation programming, following are some of advantages of .NET
- Object Oriented Programming :
  - .NET framework and C# are entirely based on object oriented principles.
- Language independence :
  - All of the languages in .NET like VB.NET, C#, J# and managed C++ are complied in to a common intermediate language.
  - Any language code further compiled in uniform language.

Advantages of .NET

- Better support for dynamic Web pages
  - ASP offered a lot of flexibility for web pages but still it was also inefficient because of lack of object oriented design and its use of interpreted scripting languages.
  - .NET offers an integrated support for Web pages, using a new Technology ASP.NET.
  - With ASP.NET code in your pages can be written in .NET aware high level language such as C# or Visual Basic.

Advantages of .NET

- Efficient data access :
  - ADO.NET is a component of .NET which provides efficient access to relational database and a variety of data sources.
  - Components are also available to allow access to the file system, and to directories.
  - In particular, XML support is built into .NET, which allows us to manipulate data.

Advantages of .NET

- Code Sharing :
  - Code sharing is possible between applications in .NET using assembly.
  - Assembly replaces the traditional the traditional DLL and it has formal facilities for versioning, and different versions of assemblies can exist side by side.
- Improved Security :
  - Each assembly can also contain built in security information that can indicate precisely what is the category of user or process is allowed to call which methods on which classes.

Advantages of .NET

- Zero impact installation :
  - There are two types of assemblies : Shared and Private.
  - Shared assemblies are common libraries available to all software whereas private assemblies are intended only for use with particular software.
  - A private assembly is entirely self-contained, so the process of installing it is simple.
  - There are no registry entries; the appropriate files are simply placed in the appropriate folder in the file system.

Advantages of .NET

- Support for Web Services :
  - .NET has fully integrated support for developing Web Services as easily as you would develop any other type of application.
- Visual Studio 2008
  - .NET comes with a developer environment, Visual Studio 2008.
  - Visual Studio 2008 will provides IDE as advanced Development tool.
- C#
  - C# is a new object oriented language intended for use with .NET

Components of .NET Architecture

- .NET Runtime (CLR)
- Managed and Unmanaged Code
- Intermediate Language
- Common Type System
- .Net Framework Class Library and Base Class Library
- Assemblies
- Metadata
- Assembly Cache
- Reflection
- JIT Compilation
- Garbage Collection

Components of .NET Architecture

- .Net Runtime (**C**ommon **L**anguage **R**untime)
  - The heart of the .NET framework is its runtime execution environment which is known as **Common Language Runtime (CLR).**
  - CLR works same as JAVA Virtual Machine.
  - CLR is an environment that executes MSIL (**Microsoft Intermediate Language**) code.
  - In java, the JVM is the concept to one language for all purpose while .NET platform supports multiple programming languages through the use of CLS **(Common Language Specification)**.

Components of .NET Architecture

- .Net Runtime (Common Language Runtime)
  - In this environment it also referred to as a managed environment, one in which common services, such as garbage collection and security are automatically provided.
- Features of CLR :
  - CLR has following features :
    - Base Class Library (BCL)
      - Supports Integrates code with the runtime that support the BCL.

Components of .NET Architecture

- Features of CLR :
  - Thread Support :
    - Provides classes and interfaces that enable multithreaded programming.
  - COM marshaller :
    - Provides marshalling to and form COM component.
  - Type checker :
    - Will not allow unsafe casts or uninitialized (અનઇનિસીયલાઇજ- ન ધારેલ) variables. MSIL can be verified to guarantee type safety.

Components of .NET Architecture

- Features of CLR :
  - Exception manager :
    - Provides structured exception handling.
  - Security engine :
    - Provides evidence-based security, based on the origin of the code in addition to the user.
  - Debug engine :
    - Allows you to debug your application on the origin of the code in addition to the user.
  - MSIL to Native Compiler :
    - Converts MSIL to native code (Just-In-Time)

Components of .NET Architecture

- Features of CLR :
  - Code manager :
    - Manages code execution.
  - Garbage Collector (GC)
    - Provides automatic lifetime management of all of your objects.
    - This is a multiprocessor, scalable garbage collector.
  - Class loader :
    - Manages metadata, as well as the loading and layout of classes.

Components of .NET Architecture

- Managed and Unmanaged Code :
  - Managed Code :
    - The code which can executed by CLR is known as Managed Code.
  - Unmanaged Code:
    - The code which is not executed by CLR is known as Unmanaged Code.
- Intermediate Language :
  - MSIL is the CPU-independent instruction set into which .NET framework programs are complied. It contains instructions for loading, storing, initializing and calling method of different objects.

Components of .NET Architecture

- Intermediate Language :
  - MSIL contains code and Metadata which is true cross language integration.
  - It also known as CIL (Common Intermediate Language) or IL (Intermediate Language).
- Common Type System (**CTS**) :
  - As Microsoft .NET provide application development using different programming languages.

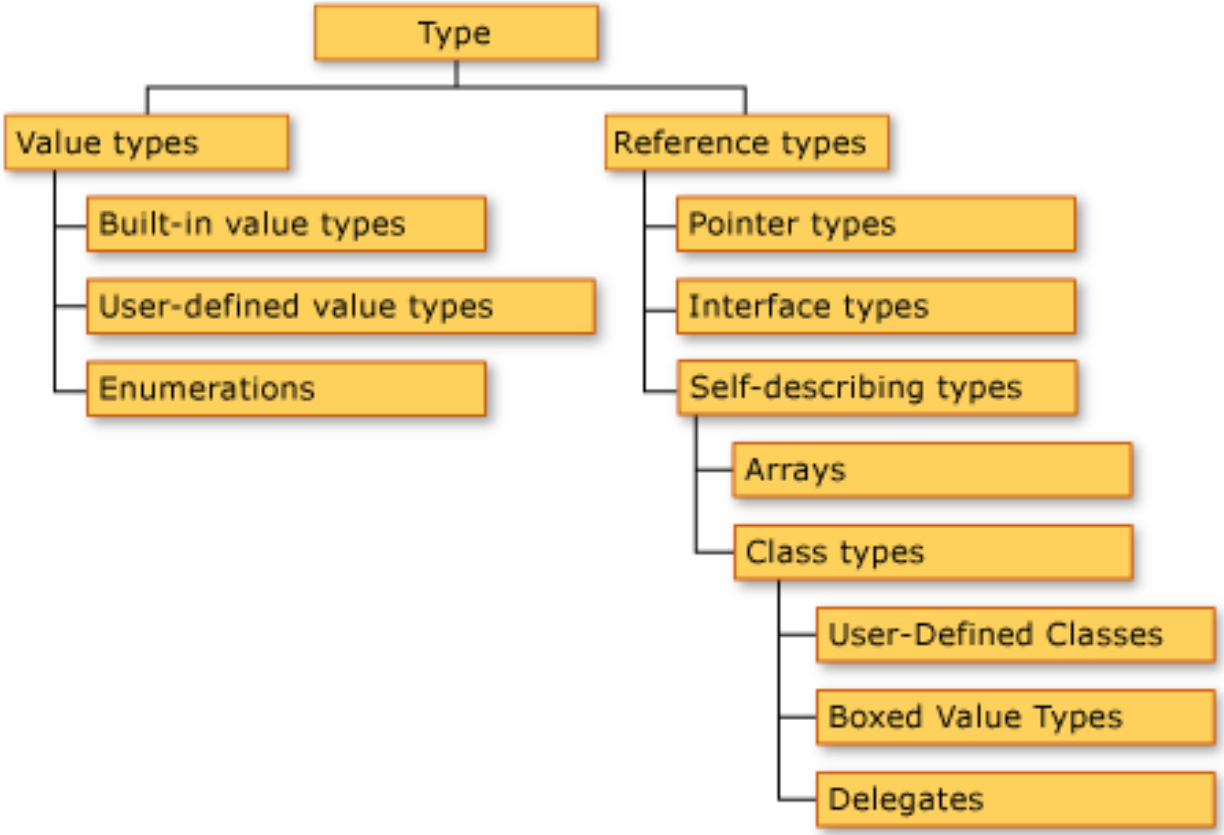
Components of .NET Architecture

- Common Type System (**CTS**) :
  - Microsoft has provided **C**ommon **T**ype **S**ystem which means we don't have to worry when we are developing multiple languages about how a data types declared in one language needs to be declared in another.
  - CTS don't specify primitive data types but it provides rich hierarchy of types.
  - The common type system supports two general categories of types, each of which is future divided into subcategories.

Components of .NET Architecture

- Common Type System (**CTS**) :
  - Common Type System Categories :

    (1) Value Types (2) Reference Types



Components of .NET Architecture

- Common Type System (**CTS**) :
  - Here interoperability (ક્રિયાશીલતા) is nothing without common data type support.
  - Which means is that an int should mean the same in VB, VC++, C# and all other .NET compliant languages.
  - CTS is much like Java, define every data type as a Class.

Data Type	Description
System.Byte	1 Byte unsigned integer between 0-255

Components of .NET Architecture

Data Type	Description
System.Int16	2 byte signed integer in range : -32768 **to** 32767
System.Int32	4 byte signed integer : -2,14,74,83,648 **to** 2,14,74,83,647
System.Int64	8 byte signed integer : -92,23,37,20,36,85,47,75,808 to 92,23,37,20,36,85,47,75,807
System.Single	4 byte floating point. **Negative Values :** -3.402823E38 to -1.401298E-45 **Positive values :** 1.401298E-45 to 3.402823E38

Components of .NET Architecture

Data Type	Description
System.Double	8 Byte wide floating point : **Negative values :** -1.7976931348623 1E308 **to** -4.94065645841247E-324 **Positive Value :** 4.94065645841247E-324 **to** 1.7976931348623 1E308
System.Object	4 byte address reference to an object
System.Char	2 byte single Unicode character
System.String	String of us to 2 billion Unicode characters

Components of .NET Architecture

Data Type	Description
System.Decimal	12 byte signed integer that can have 28 digits on either side of decimal.
System.Boolean	4 Bytes number that contains true(1) or false(0)

Components of .NET Architecture

- .Net Framework Class Library (FCL) and Base Class Library (BCL) ::
  - In C and C++, we include header files like stdio.h, conio.h for using library functions.
  - In **.NET**, BCL is the collection of classes and namespace which we use in application for variety of inbuilt functionality.
  - The **.NET** framework Base Class Library provides an extensive collection of classes which are hierarchically organized via namespace.

Components of .NET Architecture

- .Net Framework Class Library (FCL) and Base Class Library (BCL) ::
  - The Base Class Library consists of classes related to Data, XML, Web Forms, Windows Form, Smart Device, Input output etc.
  - The namespace is logical container or partition which group the different classes related to same functionality.
  - It looks like drives or folder in our computer.
  - We organized our data into different folder or drives.

Introduction to Namespace

- To understand the concept of namespace, consider a situation in which you need to create to classes with same name.
- In any operating system we cannot create two files with same name in same folder.
- If we have to create two files with same name then we have to create those files in different folders.
- The namespace is the same concept to the folder.
- Namespace is used to resolve the name clash problem in your program.

Introduction to Namespace

- Namespace is just a collection of classes.
- We can add our classes, functions, variables and related code in a namespace to create a useful container of related code.
- Now we can create another namespace and add classes, functions or variables with same name of the old namespace without any clash
- To create a namespace we have to use **namespace** keyword

Introduction to Namespace

- Syntax of Namespace

```
namespace namespace_name
{
  // required statements…
}
```

- To use classes, functions or variable from name space we have to include a statement saying that you are using that namespace in your program.

Syntax : using namespace **namespace_name**

Components of .NET Architecture

- Some of the Common Namespace ::

Namespace	Contains
System	Fundamental and base classes that defines commonly used value and references data types, events, interface, attributes.
System.Data	Classes related to ADO.NET which useful for working with database.
System.Collections	Classes used for various objects like lists, queue, array etc.
System.Drawing	Classes related to GUI drawings.

Components of .NET Architecture

Namespace	Contains
System.IO	Classes related to reading and writing data streams and files.
System.Web.UI	Classes related to create controls and pages that will appear on Web applications as user interface on a web page.
System.Windows.Form	Classes for creating Windows-based applications that take full advantage of rich user interface
System.XML	Provide standard-based support for processing XML.

Components of .NET Architecture

- Assemblies :
  - Assemblies are the building blocks of .NET framework applications; they from the fundamental unit of deployment, reuse, activation scoping and security permissions.
  - An assembly is a collection of types and resources that are built to work together and form a logical unit of functionality.
  - An assembly provides the common language runtime with the information.

Components of .NET Architecture

- Assemblies :
  - Assemblies can be **static or dynamic.**
  - **Static assemblies** can include .NET framework types (interfaces and classes), as well as resources for the assembly (bitmaps, JPEG files, resource files etc.).
  - Static assemblies are stored on disk in portable executable (PE) files.
  - We can use .NET Framework to create **Dynamic Assembly.**
  - Dynamic Assembly are run directly from memory and are not saved to disk before execution.

Components of .NET Architecture

- Assemblies :
  - The .NET assembly is the standard for components developed with the Microsoft.NET. Assemblies can be divide in two types:
    - Private Assemblies (executable [.exe] file)
      - A private assembly is used by a single application and it store in that application's install directory.
      - By default, private assemblies are created. There is no restriction about giving name to assembly in private type.
    - Shared Assemblies (dynamic link library [.DLL] file)
      - A shared assembly is one that can be **referenced by** more than one application.

Components of .NET Architecture

- Shared Assemblies :
  - Shared assembly has several restrictions like it must have a strong name and globally unique.
  - Shared assembly should be placed inside the Global Assembly Cache (GAC).

Components of .NET Architecture

- Metadata :
  - The data about DATA is called METADATA.
  - Metadata is a feature that the CLR know the details about a particular component or object.

Components of .NET Architecture

- Metadata
  - The metadata for an object is persisted (ચાલુ) at compiled time and queried at runtime so that the CLR know how to initialize object, call their methods and access their properties.
  - An application can interrogate metadata and learn what an object exposes. This process of exposes is known as reflection.

Components of .NET Architecture

- This data is stored in component itself in a binary format inside an assembly.
- It contains a declaration for every type including names and its members like methods, fields, properties and event.
- When a class loader of CLR loads an assembly at that time it uses a metadata to locate the body of method.

Components of .NET Architecture

- Assembly Cache :
  - The assembly cache is a directory which contains the different assembly on the machine.
  - There are two types of assembly cache:
    - Global assembly cache
    - Transient (temporary) assembly cache.
  - When assemblies are downloaded to the local machine using browser, it is automatically installed in the transient (temporary) assembly cache.

Components of .NET Architecture

- Assembly Cache :
  - Each computer where the common language runtime is installed has a machine-wide code cache called the global assembly cache.
  - The global assembly cache stores assemblies specifically designated to be shared by several applications on the computer.
  - We can share assemblies by installing them into the global assembly cache only when you need to.

Components of .NET Architecture

- Reflection :
  - Reflection is the process by which .NET applications can access an assembly's metadata information and discover its methods and data types at runtime.
  - We can also dynamically invoke methods and use type information through late binding through Reflection API.
  - **System.type** class is the core (ಕ್ಲಾಸ್) of the reflection. It is used to represent a Common Type System which includes methods that allow to determined type's name, which module it contain and its namespace with it contains value or reference type.

Components of .NET Architecture

- Reflection :
  - Using **System.Reflection.Assembly** class we can retrieve all of the types in an assembly and all modules contained in the assembly.

Components of .NET Architecture

- Just-In-Time Compilation (JIT)
  - When any .NET application compile, it is not converted into machine code but it converted into MSIL (Microsoft Intermediate Language).
  - This code is a machine independent. So CLR provide **Just-In-Time Compilation** technology to convert the MSIL code into a platform/device-specific code so that it can be executed on current machine.

Components of .NET Architecture

- Just-In-Time Compilation (JIT) :
  - The .NET provides three type of JIT compilers
    - Pre-JIT
      - This JIT compiles an assembly's entire code into native code at one cycle.
      - Normally, it is used at installation time.
    - Econo-JIT
      - This compiler is used on devices with limited resources.
      - It compiles the IL code bit-by-bit freeing resources used by the cached native code when required.

Components of .NET Architecture

- Just-In-Time Compilation (JIT) :
  - Normal-JIT
    - The default JIT compiles code only as it called and places the resulting native code in the cache.
    - When, JIT compiles, the native code is placed into the cache, so that when the next call is made to the same method, the cached code is executed. So it increases the application speed.

Components of .NET Architecture

- Garbage Collection :
  - Memory management is one of the housekeeping duties that take a lot of programming time in developing application.
  - .NET provides an environment with the garbage collection system.
  - Garbage collection runs when application needs free memory.
  - There is no exact time of execution of garbage collection system.

Components of .NET Architecture

- Garbage Collection :
  - When application required more memory and the memory allocator reports that there is no free memory than garbage collection is called. It starts by assuming the everything should be deleted from the memory.
  - First it creates a graph of used memory by application. When it has complete graph of memory used by application then it copies this data and free up the whole memory and reallocate the memory to the application.

Components of .NET Architecture

- Garbage Collection :
  - Garbage collector also free up the memory for unused object at regular interval.
  - We do not have to write code to perform memory management task when developing managed applications.

Importance Of .NET

- Day by day modern approaches to software development is introduced so some times it is not possible to carry on old code with new code.
- Every time some technology evolves and adds new features, it ends up a bit more complicated then it was before.
- For that Microsoft has provide solution in terms of .NET.
- .NET was designed with ability to work with existing software. .NET is a framework or an API for programming on the windows platform.
- .NET framework supports multiple programming language which will help the programmer work without updation on software coding.

Importance of C#

- In .NET framework, C# is a language that has been designed from scratch to work with .NET, as well as to take advantage of all the progress in developer environments and in our understanding of object oriented programming principles.
- C# is advanced level programming language which will help to provide all features of best level object oriented programming.
- C# will helpful to develop Desktop application with advanced features.

Introduction to the Visual Studio Development SYSTEM

- Visual Studio 2008 provides advanced development tools, debugging features, database functionality, and innovating features for quickly creating all modern applications across a variety of platforms.
- Visual Studio 2008 includes enhancements such as visual designers for faster development with .NET framework.
- Visual Studio 2008 provides rich client side and server side frameworks to easily build client centric web applications that integrated with any data provider.

Introduction to the Visual Studio Development SYSTEM

- The web applications run within any modern browser and can have complete rights to access ASP.NET application services and Microsoft platform.
- Visual Studio 2008 also includes large improvements to web development tools and language enhancements that speed development with all types of data.
- Visual Studio 2008 provides facilities like tools and framework support to create AJAX-enabled web application.

The Microsoft .NET Framework

- .NET Framework 1.0
  - Work with Visual Studio .NET
  - It contained the first version of the CLR and the first version of Base Class Library.
- .NET Framework 1.1
  - Work with Visual Studio .NET 2003
  - Included updates to ASP.NET and ADO.NET.
  - This version was subsequently updated twice with Service Pack 1 and 2.

The Microsoft .NET Framework

- **.Net Framework 2.0**
  - Work with Visual Studio 2005
  - Introduced a new version of the CLR with additions to the base class libraries, including generics, generic collections, and significant additions to ASP.NET
- **.NET Framework 3.0**
  - Work with Visual Studio 2005
  - This version is essentially .Net framework 2.0 with the addition of windows presentation foundation (WPF), Windows communication foundation(WCF), Windows Workflow foundation (WF), and CardSpace(Updated SP1, and SP2).

The Microsoft .NET Framework

- **.Net Framework 3.5**
  - Work with Visual Studio 2008
  - Added new features such as AJAX-enabled websites and LINQ. The SP1 update added dynamic data, and small set of enhancements.
- **.Net Framework 4**
  - Work with Visual Studio 2010
  - Introduced a new version of the CLR, expanded Base Class Libraries, and new features such as the **M**anaged **E**xtensibility **Framework**(**MEF**), **D**ynamic **L**anguage **R**untime (DLR), and code contacts.

The Microsoft .NET Framework

- **.Net Framework 4.5**
  - Work with Visual Studio 2012
  - Included an updated version of CLR 4, support for building Windows store apps, and updates to WPF, WCF, WF and ASP.NET
- **.Net Framework 4.5.1 RC**
  - Work with Visual Studio 2013 RC
  - Includes performance and debugging improvements, support for automatic binding redirections, and expended support for windows store apps.

Rapid Application Development :

- Visual Studio 2008 provides developers with the ability to target multiple versions of the .NET Framework from within the same development environment.
- Developers will be able to build applications that target the .NET framework. They can got support a wide variety of projects in the same environment.
- Visual Studio 2008 also supports improved language and data features, such as Microsoft Language Integrated Query (LINQ) to help developers rapidly create modern software.

Breakthrough User Experience :

- Visual Studio 2008 provides new tools using that developers provides speed creation of their connected application on the latest platforms including the web, Windows Vista, Office 2007, SQL Server 2008 and Windows Server 2008.
- ASP.NET, AJAX and other new web technologies will enable developers to quickly create a new generation of more efficient, interactive and personalised web experiences.

Effective Team Collaboration :

- Visual Studio 2008 provides expanded and improved environment for developing applications.
- It helps improve collaboration in development teams, including tools that help integrate database professional and graphic designers into the development process.

Advantages of Visual Studio 2008

- We can create web applications more easily with an improved design surface and standard support.
- Use data from any data source more smoothly with LINQ (Language Integrated Query), a set of language extensions to Visual Basic and Visual C#.
- We can use multiple versions of .NET Framework to work with Visual Studio 2008.
- Enhance collaboration between developers and designers to create more compelling user understanding.

Advantages of Visual Studio 2008

- We can build applications that use the latest web technologies with improved support for AJAX and web controls and the Microsoft AJAX Library.
- We find out the full power of the .NET framework with integrated tools that simplify building great user experiences and connected systems.
- We can also create connected applications using new visual designer for Windows Communications Foundation and Windows Workflow Foundation.

Advantages of Visual Studio 2008

- We may use Visual Studio's professional development environment to build Microsoft Office-based solutions that are reliable, scalable and easy to maintain which available in Visual Studio 2008 Professional Edition only.
- In Short, Visual Studio 2008 will provide advanced features to work with GUI based advanced development environment and support of VISUAL STUDIO 2008 Integrated Development Environment (IDE).

Features of Visual Studio 2008

- It integrates Visual Basic, Visual C# and Visual C++ to support a wide verity of development styles.
- Build applications for Windows, Web, Microsoft Office System, .NET Framework, SQL server and Windows Mobile with integrated drag-and-drop designers.
- Editor features such as Edit and Continue and Microsoft IntelliSense simplify the cycle of designing, developing and debugging and application.
- Deploy client applications easily with Click once

Features of Visual Studio 2008

- Decrease development time by reducing the need for infrastructure code and helping to enhance application security.
- Used ASP.NET to speed the creation of interactive, highly appealing web applications and web services.
- Master Page in ASP.NET allow developers to easily manage a consistent site layout in one place.

Editions of Visual Studio :

- Visual Studio is available in several editions like,
  - Visual Studio Community
  - Visual Studio Professional
  - Visual Studio Enterprise
  - Visual Studio Test Professional
  - Visual Studio Express

Editions of Visual Studio :

- **Visual Studio Community**
  - On 12 November 2014, Microsoft announced Visual Studio Community
  - A new free version similar in functionality to Visual Studio Professional.
  - Unlike Express, Visual Studio Community supports multiple languages, and provides support for extensions.
  - Visual Studio Community is oriented towards individual developers and small teams.

Editions of Visual Studio :

- Visual Studio Professional
  - Visual Studio Professional Edition provides an IDE for all supported development languages.
  - As of Visual Studio 2010, the Standard edition was dropped.
  - MSDN support is available as MSDN Essentials or the full MSDN library depending on licensing.
  - It supports XML and XSLT editing, and can create deployment packages that only use ClickOnce and MSI.
  - It includes tools like Server Explorer and integration with Microsoft SQL Server also.

Editions of Visual Studio :

- Visual Studio Enterprise
  - Visual Studio Enterprise edition provides a set of
    - Software and database development
    - Collaboration
    - Metrics
    - Architecture
    - Testing and reporting tools in addition to the features provided by Visual Studio Professional.

Editions of Visual Studio :

- Visual Studio Test Professional
  - Visual Studio Test Professional is an edition which was introduced with Visual Studio 2010.
  - Its focus is aimed at the dedicated tester role and includes support for the management of test environments, the ability to start and report on tests and to connect to Team Foundation Server.
  - It does not include support for development or authoring of tests.

Editions of Visual Studio :

- Visual Studio Express
  - Visual Studio Express Editions are a set of free lightweight individual IDEs which are provided as stripped-down versions of the Visual Studio IDE on a per-platform basis or per-language basis.
  - It installs the development tools for the supported platforms (web, Windows, phone) or supported development languages onto individual Visual Studio Shell AppIds.

Different Components of IDE :

- The Visual C# IDE is a collection of development tools exposed through a common user interface.
- Some of the tools are shared with other Visual Studio Languages.
- Most of tools can be opened form View Menu
  - Code Editor :
    - For writing Source CODE
  - C# Compiler :
    - To converting C# source code into an executable program.

Different Components of IDE :

- Visual Studio Debugger :
  - To test program by execution.
- Error List Window :
  - To Display list of errors.
- Toolbox and Designer :
  - For rapid development by user interfaces using the mouse.
- Solution Explorer :
  - For viewing and managing project files and settings.
- Project Designer :
  - For configuring compiler options, deployment paths, resources etc.

Different Components of IDE :

- Class View :
  - For navigating through source code according to types, not files.
- Properties Window :
  - For configuring properties and events on controls.
- Object Browser :
  - For viewing the methods and classes available in dynamic link libraries including .NET Framework assemblies and COM object.
- Document Explorer :
  - For browsing and searching product documentation on your local machine and on the internet.

Introduction to C#

- C# is derived (તારવેલી) from C and C++.
- C# is a simple, modern, object oriented and type-safe programming language.
- It will be easy to learn for programmers who are familiar to C and C++.
- Visual Studio also supports Visual Basic, Visual C++, and the scripting languages like, VBScript and Java Script.
- All of these languages provide access to the Microsoft .NET platform, which includes a common execution engine and a rich class library.
- The Microsoft .NET platform defines a “Common Language Specification” (CLS).

Introduction To Console

- The console programming is the method to perform CUI (Command User Interface) based programming in place of GUI (Graphical User Interface)
- Console programming is the best way to perform logical programming.
- To perform Console programming we have to following steps.

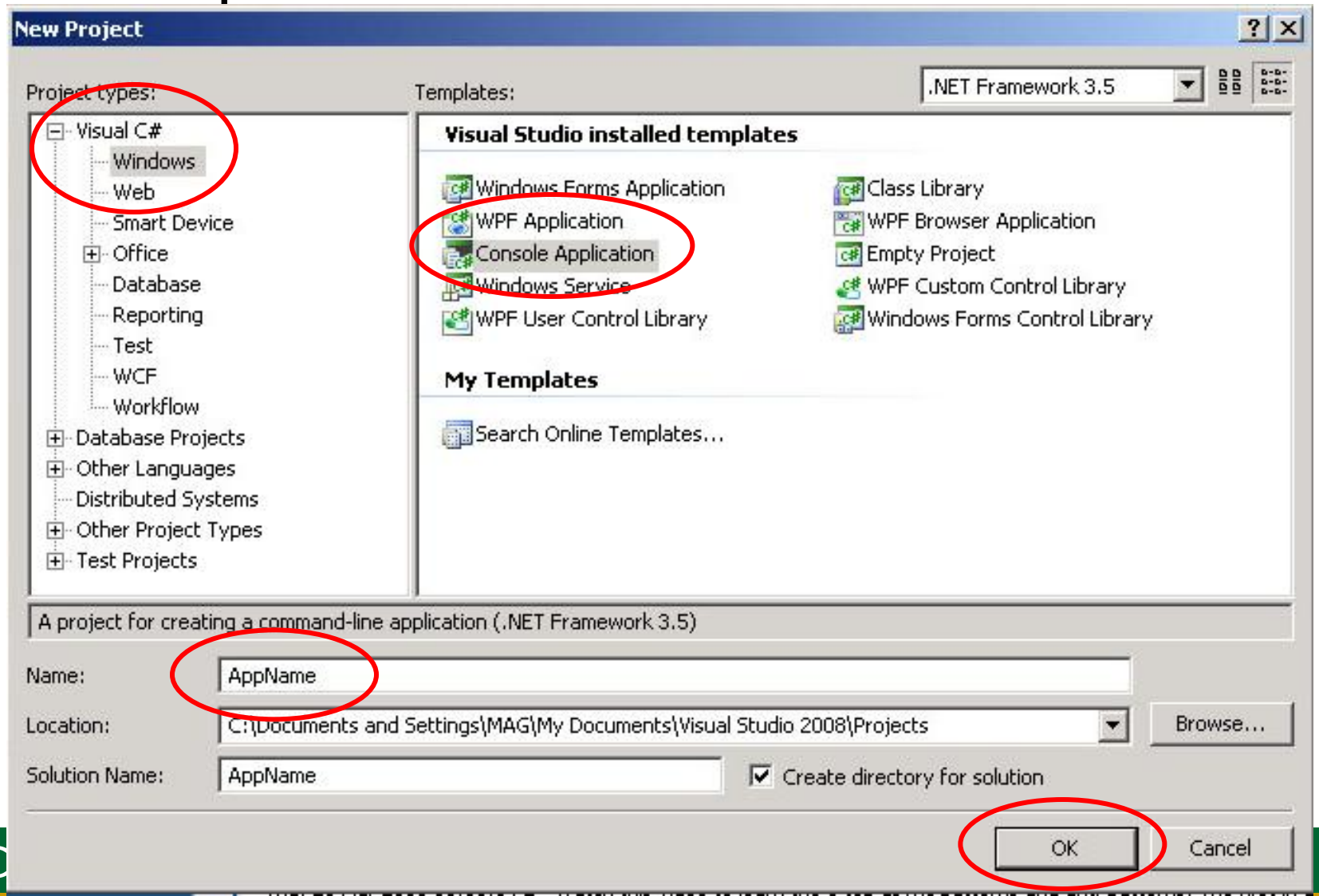
  Step-1 Install Visual Studio-2008

  Step-2 Start Visual Studio

  Step-3 Goto FILE > New > Project

Introduction To Console Programming:

Step-4 Select Console Application with Proper name.



Introduction To Console Programming:

Step-5 Write Coding in Code Window

```
// Namespace Declaration must in first
   using System;


   // Program start class
   class NamedWelcome
   {
      // Main begins program execution.
      static void Main(string[] args)
      {
         // Write to console
         Console.WriteLine("Monarch C# Programming");
        Console.ReadLine();
      }
   }
```

Def.1 : Write a console application to print your name and address :

```
class Program
{       static void Main(string[] args)
        {       System.Console.WriteLine("Gohil M.A.”);
                System.Console.WriteLine("Sanghavi Street");
                System.Console.WriteLine(“Lathi");
                System.Console.ReadLine();
        }
}
```

Note : We can also add **BackSlash Codes like “\n”**

Variable in C#

- As a C# developer, we need to understand how to store and process data in our C# application.
- Whenever application needs to store data temporarily for use during execution, we store that data in a variable with specified data type.
- How to declare variable in C#?
  - We can declare variable in C# as given.

    Syntax: <DataType> <identifier(ઓળખકર્તા)>;

    Example: int a; (int is DataType, a is Variable)

Initialization of Variables :

- C# compiler requires that any variable be initialized with some starting value before using that variable.
- C# has two methods for initialize variable before using it.
  - Variable that are members in a class or structure, if it is not initialized explicitly are by default zeroed out when they are created.
  - Variable that are local to a method must be explicitly initialized in code.

Naming Conventions of the variable

- As we learn rules to assign the variable name in other languages, most of the rules are same with that.
- Like,
  - Variable name should be start with letter or underscore.
  - We can’t use reserve keywords as a variable name.
  - It must not have special characters.
  - No spaces allowed in name.
- We have to care for above rules while naming.

Scope of the Variable :

- The scope of variable is the region (વિસ્તાર) of code from which the variable can be accessed.
  - A member variable of a class is in scope for as long as its containing class is in scope.
  - A local variable is in scope until a closing brace indicates the end of the block statement or method in which it was declared.
  - A local variable that is declared in a for, while, or similar statement is in scope in the body of that loop.

Def.2 Demonstrate Use Of Variable and print sum of two variable :

```
class Program
    {       static void Main(string[] args)
        {   int a = 10;
            int b = 20;
            int c = a+b;
            System.Console.WriteLine("A+B ="+c);
            System.Console.ReadLine();
        }
    }
```

Note : Here, “+” is concat operator.

Constants :

- As the name suggest, a constant is a variable whose value cannot be changed throughout its lifetime or we can say its value set at compile time and can never be changed.
- We can use **"const"** keyword when it is declared and initialized any variable as a constant.
- **For example :**

```
const float PI=3.14; //Cannot be changed
```

Constants :

- Characteristics Of The Constants :
  - They must be initialized when they are declared, and once a values has been assigned, it can never be overwritten.
  - The value of a constant must be computable at compile time.
  - Therefore, you can’t initialize a constant with a value taken from a variable.
  - Constants are always implicitly static. So, notice that you don’t have to use static modifier in the constant declaration.

Def.3 Demonstrate Use Of Double constant.

```
class Program
    {   static void Main(string[] args)
        {const double PI = 3.14;
          System.Console.WriteLine("A+B ="+c);
          System.Console.WriteLine("PI = " + PI);
          System.Console.ReadLine();
        }
    }
```

Def.4 WAP to check that given number is positive, Negative or Zero.

```
class Program
    {   static void Main(string[] args)
        {   int a = 0;
            if (a > 0)
                Console.WriteLine("Number is Positive");
            if (a < 0)
                Console.WriteLine("Number is Negative");
            if (a == 0)
                Console.WriteLine("Number is ZERO");
            Console.ReadLine();
        }
    }
```

Def.5 WAP to print your name 10times.

```
static void Main(string[] args)
{
        int a;
        for (a = 1; a < 10; a++)
        {
                Console.WriteLine("MONARCH");
        }
        Console.ReadLine();
}
```

Def.6 WAP to print 1 to 10 serial number.

```
static void Main(string[] args)
{
        int a;
        for (a = 1; a<10; a++)
        {
                Console.WriteLine(a);
        }
        Console.ReadLine();
}
```

Def.7 WAP to print odd numbers from 1 to 100 number

```
static void Main(string[] args)
{     int a;
          for (a = 1; a <= 100; a++)
              if (a % 2 != 0)
                  Console.WriteLine(a);
          Console.ReadLine();
}
```

Def.8 WAP to print factorial value for given number.

```
static void Main(string[] args)
{      int a, b=1;
          for (a = 5; a >= 1; a--)
             b = b * a;
          Console.WriteLine("5! is=" + b);
          Console.ReadLine();
}
```

Def.9 WAP to print Fibonacci series.

```
static void Main(string[] args)
{       int i, count=10, f1 = 0, f2 = 1, f3 = 0;
        Console.WriteLine(f1);
        Console.WriteLine(f2);
        for (i = 0; i <= count; i++)
        {   f3 = f1 + f2;
            Console.WriteLine(f3);
            f1 = f2;
            f2 = f3;
        }
        Console.ReadLine();
}
```

Def.10 Write a program to input a name and print a welcome message.

```
static void Main(string[] args)
{ string name;
  name = Console.ReadLine();
  Console.WriteLine("Welcome " + name);
  Console.ReadLine();
}
```

Def.11 Write a program to input to values and print its sum.

```
static void Main(string[ ] args)
{
        int a, b, c;
        Console.Write("Enter Value 1 :");
        a = int.Parse(Console.ReadLine());
        Console.Write("Enter Value 2 :");
        b = int.Parse(Console.ReadLine());
        c = a + b;
        Console.Write("Sum of Value 1+2 :"+c);
        Console.Read();
}
```

Data Types :

- In any programming language Data Types are most important thing.
- C# is a strongly typed language, therefore every variable and object must have a declared type.
- The C# type system contains two type categories of Data Types.
  - Value Types
  - Reference Types

Data Types :

- Value Types
  - Value type are the typical primitive (પ્રાથમિક કક્ષાનું) types available in most programming language and are allocated on the stack(થપ્પી).
    - It has sub types like, Integer Types, Floating Point types, Decimal types, Boolean types, Character Types.
- Reference Types
  - Reference types are typically class instances and are allocated on the heap(ઢગલો).
    - It has sub types like, The object types and the string types.

Value Type V/S Reference Type

Value Types	Reference Types
It is allocated on stack	It is allocated on heap
Value type variable contains the data itself	Reference type variable contains the address of memory location where data is actually stored.
Integer, float, boolean etc are value types.	String and object are reference types.
Struct is value type	Classes and interfaces are reference types.

Value Type V/S Reference Type

Value Types	Reference Types
When you copy a value type variable to another one, the actual data is copied and each variable can be independently manipulated.	When coping a reference type variable to another variable, only the memory address is copied. Both variables will still point to the same memory location, which means, if you change value of one variable, the value will be changed for the other variable too.

Data Types in C#

C# Type	.Net Framework (System) type	Signed?	Bytes Occupied	Possible Values
sbyte	System.Sbyte	Yes	1	-128 to 127
short	System.Int16	Yes	2	-32768 to 32767
int	System.Int32	Yes	4	-2147483648 to 2147483647
long	System.Int64	Yes	8	-9223372036854775808 to 9223372036854775807
byte	System.Byte	No	1	0 to 255
ushort	System.Uint16	No	2	0 to 65535
uint	System.UInt32	No	4	0 to 4294967295
ulong	System.Uint64	No	8	0 to 18446744073709551615
float	System.Single	Yes	4	Approximately $\pm 1.5 \times 10^{-45}$ to $\pm 3.4 \times 10^{38}$ with 7 significant figures
double	System.Double	Yes	8	Approximately $\pm 5.0 \times 10^{-324}$ to $\pm 1.7 \times 10^{308}$ with 15 or 16 significant figures
decimal	System.Decimal	Yes	12	Approximately $\pm 1.0 \times 10^{-28}$ to $\pm 7.9 \times 10^{28}$ with 28 or 29 significant figures
char	System.Char	N/A	2	Any Unicode character (16 bit)
bool	System.Boolean	N/A	1 / 2	true or false

Boxing and Unboxing :

- In C# it is possible to convert a value of one type into a value of another type.
  - The operation of Converting a Value type to a Reference type is called as **Boxing.**
  - The operation of Converting a Reference type to Value type is called as **Unboxing.**

Boxing V/S Unboxing

Boxing	Unboxing
Meaning: Boxing is the process of converting a value type to the reference type	**Meaning:** Unboxing is the process of converting a reference type to value type.
Type of Conversion: Implicit Conversion	**Type of Conversion:** Explicit Conversion

Example:
int i=1;
Object obj=i;
	Example:
	object obj=123;
	i=(int)obj;

Operators in C#

- In C# there are different types of operators available which are given below.
  - Arithmetic Operator
  - Relational Operator
  - Logical Operator
  - Bitwise Operator
  - Assignment Operator
  - Misc Operator

Arithmetic Operator

- Arithmetic operators are used when any mathematical operation is to be done.

Sign	Introduction
+	Used for addition
-	Used for subtraction
*	Used for multiplication
/	Used for division
%	Used for finding the remainder. It is known as modulus operator.
++	Increment increases value by one
--	Decrement decreases value by one.

Relational Operator

- When variables are related to some value at that the relational operator is used.

Operator	Description
==	This operator is used to compare the two variables or the values to check the equality.
!=	Checks if the value of two operands are equal or not, if values are not equal then condition becomes true.
<	Used for less than
>	Used for greater than
>=	Used for greater than equal to
<=	Used for less than equal to

Logical Operator

- When more than two conditions are to be checked at the same time, we can use logical.

Operator	Description
&&	And operator. Returns true if all the condition are true
||	Or operator. Returns true if any of the conditions is true.
!	Not operator. It reverses the condition. That means if the result of the condition is true then it converts into false.

Bitwise Operators :

- Bitwise operator works on bits and performs bit by bit operation.
- The truth tables for **&**, **|**, and **^** are as follows:

A	B	A&B	A|B	A^B
0	0	0	0	0
0	1	0	1	1
1	1	1	1	0
1	0	0	1	1

Assignment Operator

- There are different assignment operators as given.

Operator	Description
=	Assigning the value a=5
*=	a*=10 is similar to a=a*10
/=	a/=10 is similar to a=a/10
+=	a+=10 is similar to a=a+10
-=	a-=10 is similar to a=a-10
%=	a%=10 is similar to a=a%10

MISC Operators

Operator	Description
sizeof()	Returns the size of a data type.
typeof()	Returns the type of a class.
&	Returns the address of an variable.
*	Pointer to a variable.
?:	Conditional Expression
is	Determines whether an object is oa certain type.
as	Cast without raising an exception if the cast fails.

Operators Precedence in C#

- Operator precedence determines the grouping of terms in an expression.
- This affects how an expression is evaluated.
- Certain operators have higher precedence than others.
- Examples:
  - The multiplication operator has higher precedence than the addition operator and division operator has also higher precedence than subtraction.

Category	Operator	Associativity
Postfix	() [] -> . ++ - -	Left to right
Unary	+ - ! ~ ++ - - (type)* & sizeof	Right to left
Multiplicative	* / %	Left to right
Additive	+ -	Left to right
Shift	<< >>	Left to right
Relational	< <= > >=	Left to right
Equality	== !=	Left to right
Bitwise AND	&	Left to right
Bitwise XOR	^	Left to right
Bitwise OR	|	Left to right
Logical AND	&&	Left to right
Logical OR	||	Left to right
Conditional	?:	Right to left
Assignment	= += -= *= /= %=>>= <<= &= ^= |=	Right to left
Comma	,	Left to right

Conditional Statements

- Introduction:
  - While we working with JavaScript we can use such kind of control structure to get particular output with branching.
- If statement is divided into following categories.
  - Simple **if** Statement
  - if... else statement
  - if...else...if (ladder if) statement
  - Nested if statement.

If...(Simple)

Syntax : if(condition)

```
{     Statements
      When condition is true;
}
```

Purpose:

It will process only if the condition is true otherwise nothing.

If...else

Syntax :

```
if(condition)
{Statements when condition is true;
}
else
{Statements when condition is false;
}
```

Purpose :

It will process first section if the condition is true otherwise it executes the second section if the condition is false.

If...else if (Ladder)

Syntax :

```
if(condition)
    { Statements when condition is true; }
else if(condition)
    {Statements when condition is true; }
else
    {Statement when all condition is false;}
```

Purpose :

It will process first section if first condition is true, or it will process second section if second condition is true, otherwise it executes else section if all the condition is false.

Nested if Statement

Syntax :

```
if(condition)
{     if(condition)
      { When condition is true;  }
}
else
{Statement when all condition is false;}
```

Purpose :

Nested means one if inside the other if. If first condition will be true then it will check the other if, both condition will be true then it will execute given statement.

Switch Case

- The switch statement causes a particular group of statements to be chosen from several available groups.
- The selection is tasted upon the current value of an expression that is included within the switch statement.
- Switch case is a built in multi-way decision statement.
- It tests value of given variable (Expression) against a list of case values & block of statements associated with it is executed when a match is found.

- The general form of the switch statement is:

switch(expression)

Here expression result is an integer value or char type.

Syntax :

```
switch(Expression)
{     Case Constant1:
            Statements;
            break;
      Case Constant2:
            Statements;
            break;
      Case ConstantExpN:
            Statements;
            break;
      Default:
            Statements;
}
```

Purpose :

- Switch case statement will provide us easy features of working it will provide selection based switching in group of statements. After completion of group execution we must put a break; statement to stop execution in each group. We can use either integer number or character for switching from list of group.

- Loops
  - Loops are used when we want to execute a part for a block of statements for specified number of times or depending on some specified condition.
  - The main thing about any of the loop is that it executed for the number of times based on the logical expression (condition) that is specified to the particular loop.
  - When loop checks for the condition to execute at that time there are two types of loops available.
    - Entry Control Loop
    - Exit Control Loop

Looping Structure

Entry Control Loop

- The loop which is checked before executing the statements of loop is called as Entry Controlled loop.
- This loop is first checked and if it is true then only the statements under that loop are executed and if it is not true then it transfers the control at to the end of the loop. They are as follow.
  - for( ) loop **&** foreach( )
  - while( ) loop

For... Loop

Syntax :

```
for(initialization; condition; increment or decrement)
   {
       statements;
   }
```

Purpose :

- To provide a looping while condition will be true we can use for loop.

  initialization : Initialize the value of variable.

  test Condition : Test condition at here.

  Increment/decrement: To increase or decrease the value.

foreach( )... Loop Used for ARRAY

Syntax :

```
foreach(int <tempVarName> in arrayOfInts)
   {
      Console.WriteLine(tempVarName);
   }
```

Purpose :

- Foreach is a looping statement used with array.
- We can perform loop based on array element using foreach.

While... Loop

Syntax :

```
while(condition)
{     statements;
}
```

Purpose :

- To provide a looping till condition will be true we can use while loop.
  - While loop will check the condition first.
  - If the condition will be true then it will execute the statements given in the loop, after once complete the loop, it will again and again check the condition if condition will false then it will auto exit for the loop.

Looping Structure

Exit Control Loop

- The loop which is checked after executing the statements of loop is called as exit controlled loop.
- This loop is not checked in the began of loop but checks after execution of statement.
- While we want to execute any loop at least once at that time we can use this looping statement. They are as follow.
  - Do…while( )

Do...while Loop

Syntax :

```
do
{     statements;
}while(test condition);
```

Purpose :

- To provide a looping till condition will be true we can use do…while loop.
  - Do…while loop will check the condition after execution of statements provided in to loop.
  - If the condition will be true then it will again execute the statements given in the loop, if condition will false then it will auto exit for the loop.

Nesting of Loop

- When one looping statement is within another looping statement is called as nesting of loop.
- The nesting may continue up to any desired levels.

Jumping Statement : Break

Syntax : break;

Purpose : The break command will break the loop.

Example :

```
int j=0;
for(int i=0;i<5;i++)
{   Console.WriteLine("MONARCH");
    j++;
    if(j==2)
        break;
}
```

Jumping Statement : Continue

Syntax : continue;

Purpose : The continue command will break the current loop and continue with the next loop.

Example :

```
for(i=0;i<=10;i++)
 {   if(i==3)
            continue;
            document.write("Num. is "+i);
 }
```

return statement :

Syntax : return <Expression>;

Purpose : The return statement will return the given expression to the calling function.

goto label statement :

Syntax : goto <LabelName>;

Purpose : The goto statement will navigate the pointer to the specified label.

Example :

```
int i=1;
abc:   //Defined Label
Console.WriteLine(i);
i++;
if(i<=10)
        goto abc; //Pointer moves
                    to label...
```

String :

- In normal terms string is an array of characters but with the C# we can say that it is sequence of character object.
- Here we have used term object with the string definition because String is one of the class in the C# not the data type.
- Here, **System.String** is a class specifically designed to store a string and allow a large number of operations on the string.
- The **String** class provides many methods for creating, manipulating, and comparing strings.

String :

- Declaration and Initialization of String using **System.String** class :

  System.String name=“Monarch”;

  Or

  char[] ch={‘M’, ‘o’, ‘n’, ‘a’, ‘r’, ‘c’, ‘h’};

  string name=new string(ch);

- Methods Of String Class
  - String class contains many methods like,
    - Compre, Concat, Copy, Join, Intern, IsInterned, Clone, CompareTo, CopyTo

String : (Methods of the String Class)

String Functions	Definitions
Clone()	Make clone of string.
Compare (String1, String2)	This method is used to compare two specified string objects.
compareOrdinal(String1, String2)	This method is used to compare two specified String objects but here with compare method 'a' is less than a while with compareOrdinal method 'a' is greater than 'A'.

String : (Methods of the String Class)

String Functions	Definitions
CompareTo()	Compare two strings and returns integer value as output. It returns 0 for true and 1 for false.
Concat (String1, String2,... StringN)	This method is used to concatenate one or more Strings.
Contains()	The C# Contains method checks whether specified character or string is exists or not in the string value.

String : (Methods of the String Class)

String Functions	Definitions
Copy( )	This method is used to create a new instance of String with the same value as a specified String.
EndsWith()	This EndsWith Method checks whether specified character is the last character of string or not.
Equals()	The Equals Method in C# compares two string and returns Boolean value as output.
GetHashCode()	This method returns Hash Value of specified string.

String : (Methods of the String Class)

String Functions	Definitions
GetType()	It returns the System.Type of current instance.
GetTypeCode()	It returns the System.TypeCode for class System.String.
IndexOf()	Returns the index position of first occurrence of specified character.
IndexOfAny	This method reports the index of the first occurrence in this instance of any character in a specified array of Unicode characters.

String : (Methods of the String Class)

String Functions	Definitions
Insert()	Insert the string or character in the string at the specified position.
Intern	This method returns the system's reference to the specified String.
IsInterned	This method returns a reference to a specified String.
IsNormalized()	This method checks whether this string is in Unicode normalization form C.

String : (Methods of the String Class)

String Functions	Definitions
Join	This method is used to concatenate a specified separator String between each element of a specified String Array, yielding a single concatenated string.
LastIndexOf()	Returns the index position of last occurrence of specified character.
LastIndexOfAny()	Returns the index position of last occurrence of one on more characters in a string.
Length	It is a string property that returns length of string.

String : (Methods of the String Class)

String Functions	Definitions
PadLeft()	This method right-aligns the characters in a string on the right with spaces or a specified character, for a specified total length.
PadRight()	This method left-aligns the characters in a string on the right with spaces or a specified character, for a specified total length.
Remove()	This method deletes all the characters from beginning to specified index position.
Replace()	This method replaces the character.

String : (Methods of the String Class)

String Functions	Definitions
Split()	This method splits the string based on specified value.
StartsWith()	It checks whether the first character of string is same as specified character.
Substring()	This method returns substring.
ToCharArray()	Converts string into char array.

String : (Methods of the String Class)

String Functions	Definitions
ToLower()	Converts String into lower case based on rules of the current culture.
ToUpper()	Converts String into Upper case based on rules of the current culture.
Trim()	It removes extra whitespaces from beginning and ending of string.
TrimEnd()	This method deletes all occurrences of a set of characters specified in a character array from the end of a string.

Def.12 Write a program to convert given string in to UPPER CASE & lower case.

```
static void Main(string[ ] args)
{       string name="monarch computer Lathi";
        Console.WriteLine(name.ToUpper());
        Console.WriteLine(name.ToLower());
        Console.Read();
}
```

Def.13 Write a program to Enter a name and print it in reverse order.

```
static void Main(string[ ] args)
{       string name, result="";
        Console.WriteLine("Enter Any Name :");
        name=Console.ReadLine();
        for (int i = name.Length - 1; i >= 0; i--)
        {
            result += name[i];
        }
        Console.WriteLine("Reverse :" + result);
        Console.Read();
}
```

Def.14 WAP to Enter a string and check that given string is Palindrome or not.

```
static void Main(string[ ] args)
{string mname, newname = "";
  Console.WriteLine("Enter New Name :");
  mname = Console.ReadLine();
  for (int a = mname.Length - 1; a >= 0; a--)
  {   newname = newname + mname[a];
  }
  if (mname == newname)
      Console.WriteLine("String Is Palindrome");
  else
      Console.WriteLine("String is not Palindrome");
}
```

Array :

- Array is a variable which holds multiple values (elements) of similar data type.
- All the values are having their own index with an array.
- Index will starts at zero.
- In C# arrays are works as same as the other programming languages some of the differences are there which defines as follow
  - In **"C#"** When declaring an array, the square brackets **"[]"** must come after the data type, not after the identifier (variable) name.

Array : Single-dimensional array :

- To declare single-dimensional array in C#, we can use the following syntax:

  datatype[] arrayName;

- Initializing an Array
  - Declaring an array does not initialize the array in the memory.
  - When the array variable is initialized, you can assign values to the array.
  - Array is a reference type, so you need to use the **new** keyword to create an instance of the array. For example,

    DataType**[]** ArrayName = **new** DataType**[**Elements**];**

Array : Single-dimensional array :

- Assigning Values to an Array
  - You can assign values to individual array elements, by using the index number, like:
  - Example-1 :

```
int[] marks=new int[3];
marks[0] = 55;
marks[1] = 55;
marks[2] = 55;
```

  - Example-2 :

```
int[ ] marks = new int[5]{99,98,92,97,95};
```

Def.15 WAP to Assign MARKS for 4 subject to an Array variable and Print Total and percent.

```
int[] marks=new int[5];
   marks[0] = 55;
   marks[1] = 55;
   marks[2] = 55;
   marks[3] = 55;
   marks[4] = 55;
int tot = marks[0]+marks[1]+marks[2]+marks[3]
                                    +marks[4];
Console.WriteLine("Total Marks = " +tot);
float per = (float)tot/5;
Console.WriteLine("Percent     = " + per);
```

Def.16 WAP to Enter MARKS in 4 subject to an Array variable and Print Total and percent.

```
int [] marks = new int[5];
Console.Write("Enter Marks For Sub1 :");
marks[0]=int.Parse(Console.ReadLine());
.
.
Console.Write("Enter Marks For Sub5 :");
marks[4] = int.Parse(Console.ReadLine());
int tot = marks[0] + marks[1] + marks[2] +
                                marks[3] + marks[4];
float per = (float)tot/5;
Console.WriteLine("Total Marks :" + tot);
Console.WriteLine("Percentages :" + per);
```

Def.17 WAP to Enter MARKS in 4 subject to an Array variable using LOOP and Print Total and percent.

```
int [] marks = new int[5];
int tot = 0;
for (int a = 1; a <= 5; a++)
{ Console.Write("Enter Marks Sub " + a + " :");
  marks[a-1] = int.Parse(Console.ReadLine());
  tot =tot+ marks[a-1];
}
  Console.WriteLine("Total Marks : " + tot);
  float per = (float)tot / 5;
  Console.WriteLine("Percantages : " + per);
```

Array : Multi-dimensional array :

- C# allows multidimensional arrays.
- Multi-dimensional arrays are also called rectangular array.
- You can declare a 2-dimensional array as given:

  Syntax : DataType [,] VariableName;
- We have to put comma in square bracket to define it as multidimensional.
- Example :

```
int[,] marks = new int[2, 3];
int[,] marks = new int[2, 3]{{10,20,30},{40,50,60}};
```

Def.18 WAP to store MARKS of 2 students with 3 subjects using Multi-dimensional array

```
int[,] marks=new int[2,3] {{10,20,30}, {40,50,60}};
int[] tot = new int[2];
float[] per = new float[2];
tot[0] = marks[0,0]+ marks[0,1] + marks[0,2] ;
per[0] = (float)tot[0] / 3;
Console.WriteLine("Total For Student1:" + tot[0]);
Console.WriteLine("Percantages :" + per[0]);
tot[1] = marks[1,0]+ marks[1,1] + marks[1,2];
per[1] = (float)tot[1] / 3;
Console.WriteLine("Total For Student1:" + tot[1]);
Console.WriteLine("Percantages :" + per[1]);
```

Array : Jugged array :

- A Jagged array is an array of arrays.

DEF.19 : Demonstrate Use of Jagged Arrya

```
int[][] marks = new int[3][];
int[] tot = new int[3];
float[] per = new float[3];
marks[0] = new int[3];
marks[0][0] = 10;
marks[0][1] = 10;
marks[0][2] = 10;
tot[0] = marks[0][0]+marks[0][1]+marks[0][2];
per[0] = (float)tot[0] / 3;
Console.WriteLine("\nTotal :" + tot[0]);
```

Array : Jugged array :

```
Console.WriteLine("Per   :" + per[0]);
marks[1] = new int[4];
marks[1][0] = 20;
marks[1][1] = 20;
marks[1][2] = 20;
marks[1][3] = 20;
tot[1] = marks[1][0]+marks[1][1]+marks[1][2]
per[1] = (float)tot[1] / 4;              +marks[1][3];
Console.WriteLine("\nTotal :" + tot[1]);
Console.WriteLine("Per   :" + per[1]);
```

Array : Jugged array :

```
marks[2] = new int[3];

marks[2][0] = 30;

marks[2][1] = 30;

marks[2][2] = 30;

tot[2] = marks[2][0]+marks[2][1]+marks[2][2];

per[2] = (float)tot[2] / 3;

Console.WriteLine("\nTotal :" + tot[2]);

Console.WriteLine("Per   :" + per[2]);
```

Array : Array are Objects

- In C#, arrays are actually objects,
- System.Array is the abstract base type of all array types.
- We can use the properties, and other class members of Sytem.Array
- Example :
  - Use the Length property to get the length of an array.

Def.20 Count Length of Array and Print It

```
int[] number = { 1, 2, 3, 4, 5};

int len = number.Length;

Console.WriteLine("Elements : " + len);

Console.Read();
```

Array Class :

- The Array class, defined in the System namespace, is the base class for arrays in C#.
- Array class is an abstract base class but it provides CreateInstance method to construct an array.
- The Array Class provides methods for creating, manipulating, searching and sorting arrays.

Array Class : Properties

- **IsFixedSize**
  - Gets a value indicating whether the Array has a fixed size.
- **IsReadOnly**
  - Gets a value indicating whether the Array is read-only.
- **Length**
  - Gets a 32-bit integer that represents the total number of elements in all the dimensions of the Array.

Array Class : Properties

- **LongLength**
  - Gets a 64-bit integer that represents the total number of elements in all the dimensions of the Array.
- **Rank**
  - Gets the rank (number of dimensions) of the Array.
- **IsSynchronized**
  - Returns a value indicating if access to an array is thread-safe or not.
- **SyncRoot**
  - Returns an object that can be used to synchronize access to the array.

Array Class : Methods

- **Clear**
  - Sets a range of elements in the Array to zero, to false, or to null, depending on the element type.
- **Copy(Array, Array, Int32)**
  - Copies a range of elements from an Array starting at the first element and pastes them into another Array starting at the first element. The length is specified as a 32-bit integer.

Array Class : Methods

- **CopyTo(Array, Int32)**
  - Copies all the elements of the current one-dimensional Array to the specified one-dimensional Array starting at the specified destination Array index. The index is specified as a 32-bit integer.
- **GetLength**
  - Gets a 32-bit integer that represents the number of elements in the specified dimension of the Array.

Array Class : Methods

- **GetLongLength**
  - Gets a 64-bit integer that represents the number of elements in the specified dimension of the Array.
- **GetLowerBound**
  - Gets the lower bound of the specified dimension in the Array.
- **GetType**
  - Gets the Type of the current instance. (Inherited from Object.)
- **GetUpperBound**
  - Gets the upper bound of the specified dimension in the Array.

Array Class : Methods

- **GetValue(Int32)**
  - Gets the value at the specified position in the one-dimensional Array. The index is specified as a 32-bit integer.
- **IndexOf(Array, Object)**
  - Searches for the specified object and returns the index of the first occurrence within the entire one-dimensional Array.
- **Reverse(Array)**
  - Reverses the sequence of the elements in the entire one-dimensional Array.

Array Class : Methods

- **SetValue(Object, Int32)**
  - Sets a value to the element at the specified position in the one-dimensional Array. The index is specified as a 32-bit integer.
- **Sort(Array)**
  - Sorts the elements in an entire one-dimensional Array using the IComparable implementation of each element of the Array.
- **ToString**
  - Returns a string that represents the current object. (Inherited from Object.)

Array Class : Methods

- **BinarySearch**
  - This method searches a one-dimensional sorted Array for a value, using a binary search algorithm.
- **Clone**
  - This method creates a shallow copy of the Array.
- **CreateInstance**
  - This method initializes a new instance of the Array class.
- **GetEnumerator**
  - This method returns an IEnumerator for the Array.

Array Class : Methods

- **Initialize**
  - This method initializes every item of the value-type Array by calling the default constructor of the value type.
- **LastIndexOf**
  - This method returns the index of the last occurrence of a value in a one-dimensional Array or in a portion of the Array.

Def.21 WAP to store name list in array and print name using foreach.

```
string[] pets = { "dog", "cat", "bird" };

// ... Loop with the foreach keyword.

foreach (string value in pets)

{

   Console.WriteLine(value);

}

Console.Read();
```

Def.22 WAP to Enter 5 values in an array and print its sum & average using foreach loop.

```
int[ ] value = new int[5];
int tot = 0;
for (int a = 0; a <= 4; a++)
{ Console.Write("Enter Value "+(a+1)+": ");
  value[a]=int.Parse(Console.ReadLine( ));
  tot = tot + value[a];
}
  Console.WriteLine("Sum of Elements :”+tot);
  float per = (float)tot / 5;
  Console.WriteLine("Average :" + per);
  Console.Read();
```