GOLDEN BOY

UNIT-1

.Net Framework and C# Programming

Introduction To C# .Net Framework :-

.Net →

1→ .Net Stands for Network Enable Technology which is developed by Microsoft Corporation.

2→ .Net is a Collection of languages like VC#.Net, VC++.Net, VB.Net and So on.

3→ .Net is a framework which provides a common platform to execute or Run the applications developed in various programming languages.

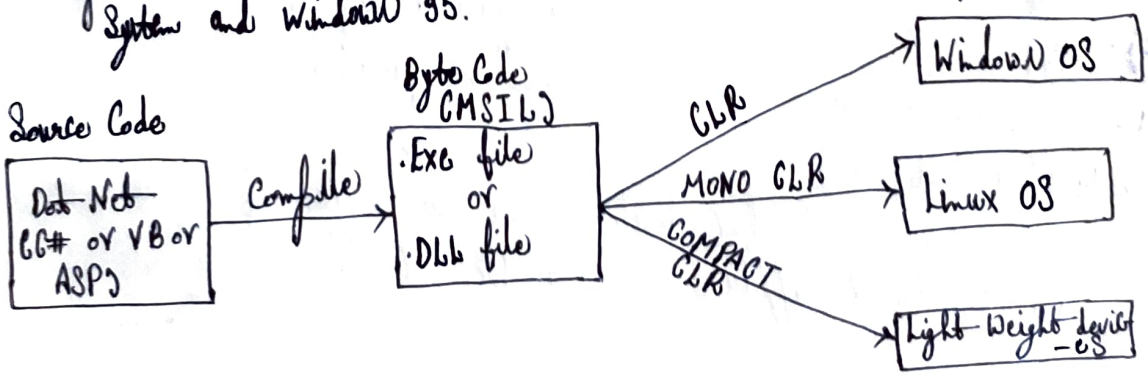
.Net objective → The .Net Framework is designed to fulfil the following objectives :—

1 → Platform Independent → All .exe and .dll files work in any operating System with the help of CLR, hence .Net is called as platform independent.

2 → (.exe) is executable file, its Consists of executable Code, and (.dll) is dynamic link library file, its Consists of reusable Code. .exe and .dll files Contains the Code in the format of byte Code is also called as MSIL (Microsoft intermediate language) Code.

3 → Machine language is also called as native Code. CLR is Common language runtime, CLR Software Converts byte Code into native Code. .Net is platform independent but CLR Software is platform dependent.

One question arises if we go in detail, that is either Dot Net is pure platform independent or not? Answer is Dot Net is partially platform dependent, as of now CLR Softwares are not available for DOS operating System and Windows 95.



4 → Language Independent → As Dot Net programming logic can be developed in any dot net framework compatible languages, hence dot net is called as language Independent.

Microsoft is introducing approximately 40 languages into Dot Net framework, out of which as of now approximately 24 languages and one Specification are released. But in 4.8 .Net framework upto 32 languages are supported.

Ex → VC#.Net, VB.Net, VC++, VJ#, VF#, PHP, COBOL, PERL, PYTHON, SMALLTALK, JSCRIPT--- etc, one Specification is ASP.Net.

5 → Language interoperability → It is a concept of developing an application with the help of more than one .Net programming language. After an application is compiled, the source code will be converted into byte code. Byte code follows a standard instruction set provided by CLS (Common Language Specification). CLS provides common rules and Syntaxes for all the language and CLS also provides common data types for all the languages and these common data types are called as CTS (Common type System).

C++	C#	VB.Net
int (2)	Short (2)	Short (2)
Long (4)	int (4)	Integer (2)
	long (8)	long (8)
	SYSTEM.INT16	
	SYSTEM.INT32	
	SYSTEM.INT64	

6 → It supports object-oriented programming environment.

7 → It supports to work with database programming with the help of ADO.Net.

8 → It supports to work with WPF (Window presentation foundation) for developing animations.

9 → It provides environment for developing various types of applications, such as Window-based applications and Web-based applications.

10→ It Supports to develop 3-tier architecture with the help of Dot Net Remoting.

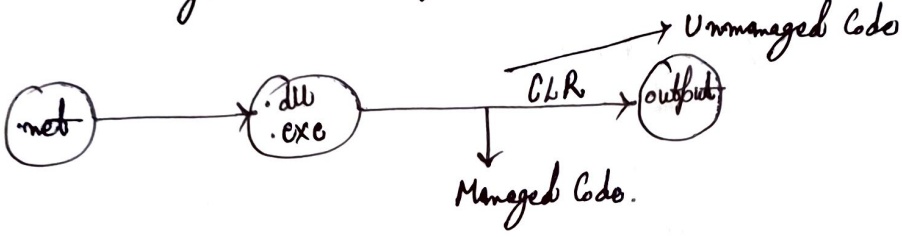
11→ Supports to develop game programming with the help of multi-threading.

12→ Supports to work with link programming.

Managed Code → The Code is developed and running under the control of the CLR is often termed managed Code.

Unmanaged Code → The Code which takes OS help while execution is called as Unmanaged Code.

Note → Managed Code is faster in execution.



.Net Framework →

1→ The .Net framework is a revolutionary platform that helps you to write the following types of applications:

→ Windows applications
→ Web applications
→ Web Services

2→ The .Net framework applications are multi-platform applications.

3→ The .Net framework has been designed in such a way that it can be used by .Net framework Compatible languages (C#, C++, Visual Basic, JScript, COBOL, etc.). All .Net framework Compatible languages can access the framework as well as communicate with each other.

4→ The .Net Framework Contain three major things that is

→ Languages → VB, C#, Pascal, Cobol

→ Technology → ASP.Net, ADO.Net

GOLDEN BOY

→ Servers → SQL Server, Share Point Server.

5 → In platform independent languages, the code gets executed under the control of special software called framework.

Example: java → JVM
.Net → CLR

Framework → A framework is a software which makes the functionalities of a OS and makes the code to execute under its control. Framework provides the following basic features like
→ Platform independency
→ Security
→ Memory Management

→ Microsoft has started the development of .Net framework in the late 90's originally under the name of NGWS (Next generation Windows Services).

→ To develop the framework first the Microsoft has prepared a set of specifications known as CLI (Common language infrastructure).

→ The .Net specifications are open to all (everyone can develop) which is standardized under ISO (International Standards organization) And ECMA (European Computer manufacturer association).

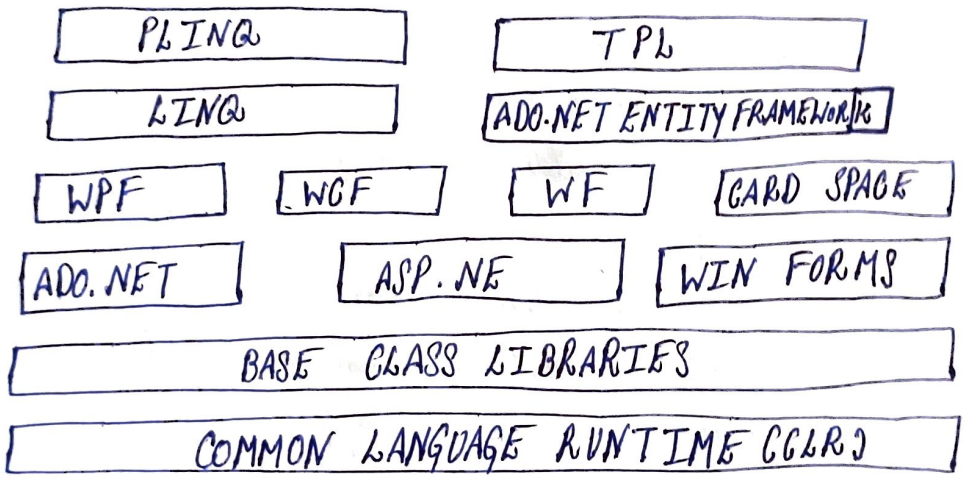
→ CLI specification talks about 4 important aspects that is
CLS (Common language specification)
CTS (Common type System)
BCL (Base class libraries)
VES (Virtual execution System) or CLR (Common language runtime)

→ .Net framework versions
1.0 (2002), 1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8. [CLR in version 4]

→ The .Net framework consists of an enormous library of codes used by the client languages such as C#. Following are some of the components of the .Net Framework :—

→ Common Language Runtime (CLR)
→ The .Net Framework class Library
→ Common Language Specification
→ Common Type System
→ Metadata and Assemblies
→ Window Forms
→ ASP.Net and ASP.Net AJAX
→ ADO.Net
→ Window Workflow Foundation (WF)
→ Window Presentation Foundation
→ Window Communication Foundation (WCF)
→ LINQ

.NET Framework architectural diagram →



The .Net framework is one of the tools provided by the .Net infrastructure and tools Component of the .Net platform. The .Net framework provides an environment for building, deploying and running Web Services and .Net applications.

Common Language Runtime (CLR) →

→ CLR is the heart of the .NET framework. It is the execution engine of .NET framework, where all .Net applications are running under the Supervision of CLR. The main function of CLR is to convert the

GOLDEN BOY

managed Code into native Code and then execute the program. CLR acts as a layer between OS and the applications written in the .NET languages. CLR provides various features to applications like

→ Security
→ Platform independency
→ Automatic memory management
→ Runtime error handling
→ Code Verification
→ Compilation
→ Thread management

→ The Components of CLR are →

Class Loader → It is used to load all the classes at runtime for the execution of an application.

JIT Compiler → Just In Time, It is responsible for the conversion of MSIL (Byte Code) into machine code (native code).

Code Manager → It is responsible for managing code at runtime.

Garbage Collector → The .Net Garbage Collector is responsible for ~~managing~~ automatic memory management, where memory management is a process of allocation and deallocation of memory that is required for a program execution. Memory management is of two types :—

1→ Manual / explicit memory Management → In this the programmers are responsible for allocation and deallocation of memory that is required for a program execution.

2→ Automatic / implicit memory Management → In this case the garbage Collector will take care of allocation and deallocation process of memory.

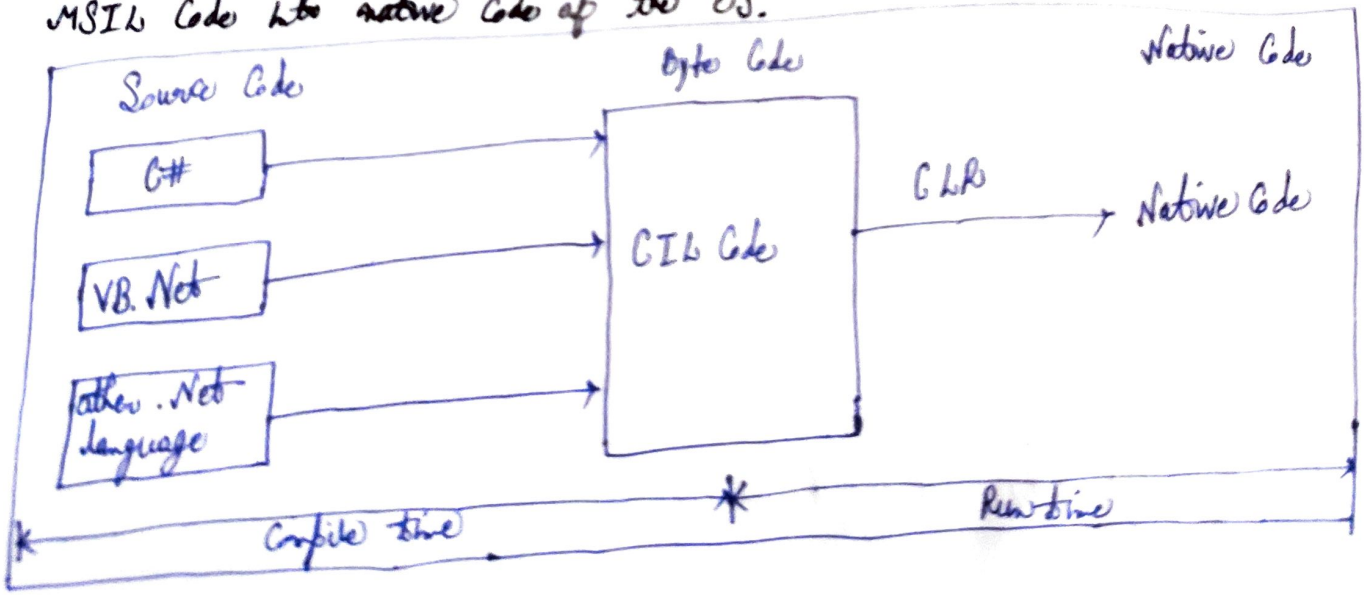
The Garbage Collector will allocate the memory for an object when and where they are required and also deallocate the memory of those objects once they become unused under the program. The unused objects of a program is called garbage and ~~is~~ deallocate immediately.

Security Engine → (manager) → It is responsible for taking care of the Security of the application that is the Security Engine will not allow the application to interact directly with Operating System or OS to interact with the application.

Type checker → It enforces Strictness in type checking that is type checker will verify the types used in the application with CTS(or) CLS Standards supported by the CLR.

Common Type System :→ Common type System (CTS) describes a set of types that can be used in different .Net languages in common. That is, the CTS ensures that object written in different .Net languages can interact with each other.

Microsoft Intermediate ~~Code~~ Language (MSIL) → MSIL stands for Microsoft Intermediate language. We call it Intermediate language (IL) or Common Intermediate language (CIL). During Compile time, the Compiler ~~Compiler~~ converts the source code into Microsoft Intermediate language, (MSIL). MSIL is CPU-Independent Set of instructions that can be efficiently converted to the native code. During the runtime the Common Language Runtime (CLR)'s Just in time (JIT) Compiler converts the MSIL code into native code of the OS.



Thread Support → It allows multithreading support to our application.

Debug Engine → It allows proper debugging of an application.

Base class library → It provides all the types that an application need at runtime.

Exception Manager → Handles all the exception for an application during runtime.

COM Marshaller → It provided interoperability to our applications.

Base class libraries →

→ A library is a set of functionalities where each and every programming language has built in library like header files in C/C++, packages in java. Similarly, .Net language have built in libraries known as base class libraries. The speciality of these library is that they can be consumed under any .Net language CBCL are the best example of language interoperability because these library are developed in C# language and can be consumed in any .Net framework compatible language.

→ The very popular namespace in base class library is System

→ We can use the base class library in the system namespace for many different tasks including.

→ Input / output operations
→ String Handling
→ Managing arrays, lists and maps
→ Accessing files and file systems
→ Security
→ Windowing
→ Window messages
→ Drawing
→ Database Management
→ Managing errors and exceptions

→ Win form technology is used to develop window based applications
→ Asp. Net is used to build rich internet based web applications
→ ADO. Net is used to create Data Access layer to query and manipulate data from underlying data Sources like SQL Server, Oracle etc.

→ WPF (Window presentation foundation) → It is used to create application with rich user experience. It includes application user Interface, 2D graphics, 3D graphics and multimedia. It takes advantage of H/W acceleration of modern graphic cards. WPF makes the user Interface better, Scalable and resolution independent.

→ WCF (Window Communication foundation) → It is used for building and developing Services based on Web Services standards.

→ WF (WWF) stands Window Workflow foundation → Which is used to build process oriented business workflow and rules Engine.

→ LINQ (Language Integrated Query) → allows you to query (Communicates) the data from the various data Sources (like SQL database, XML documents, Ado. Net Datasets, various Web Services and any other objects Such as Collection, Generics etc.) using a SQL query like syntax with .Net framework languages like C# and VB.

→ ADO. Net Entity Framework → It provides added features under the traditional ADO. Net. This is used to query and store data into the relational databases (like SQL (like SQL Server, Oracle, DB2 etc.) in ORM (Object - relational mapping) fashion.

→ TPL (Task Parallel Library) → The purpose of TPL is to make the developer more productive by Simplifying the process of adding parallelism and Concurrency to application.

→ PLINQ (Parallel Language Integrated Query) → It is used to maximize processor utilization with increased throughput in a multicore architecture.

~~Benefits of the~~

The Origin of .NET Technology :→

→ OLE Technology (Object linking and Embedding) :→ OLE was developed by Microsoft in the early 1990's to enable inter process communication. OLE provided support to achieve the following:

a → To embed documents from one application into another application.

b → To enable one application to manipulate objects located in other application.

OLE Technology enabled users to develop applications which required interoperability between various modules such as MS-Word and MS-Excel.

COM Technology (Component object Model) :→ Till the advent of ~~COM~~ COM technology, the monolithic approach had been used for developing S/W. But when the programs became too large and complex, the monolithic approach leads to a number of problems in terms of maintainability and testing of S/W. To overcome these problems Microsoft introduced a Component based model for developing S/W. In this approach a program is divided into number of independent Components where one offers a particular task. Each Component can be developed and tested independently and then integrated into the main System. This technology is called Component object Model.

Benefits :→

→ It reduces the complexity of S/W

→ It enhances S/W maintainability

→ It enables distributed development among multiple departments and organizations.

.NET Technology (Networks Enable Technology) :→

→ It is a third-generation Component model which provides a new levels of interoperability compared to COM technology.

2→ It enabled Application level communication by Microsoft intermediate language CMSIL as CIL mechanism.

3→ .NET Technology allows true cross-language integration with MSIL.

4→ .NET Technology contains MSIL and also includes other technologies and tools that enable users to develop and implement applications easily.

.NET Languages → The .NET framework is a language neutral. So we can use the number of languages for developing .Net applications. They include

Native to .Net :
- → C#
- → Visual basic
- → C++
- → JScript

Third-Party languages :
- → COBOL
- → Eiffel
- → Perl
- → Python
- → Smalltalk
- → Mercury
- → Scheme

All .NET languages are not created equal. Some can use the components of other languages, some can use the classes produced in other languages to create, and some languages can extend the classes of other languages using the inheritance features of .NET.

Benefits of the .NET Framework →

→ Simple and faster System development
→ Enhanced built in functionality
→ It supports rich oop Concepts
→ Integration of different applications into one platform.
→ Ease of deployment and execution.

→ Interoperability with existing applications
→ Wide range of Scalability
→ Simple and easy to build Sophisticated development tools.
→ Fewer bugs
→ Potentially better performance.

Features of .NET →

→ Language Independency
→ Base class Library's
→ Portability
→ COM interoperability
→ CLR
→ Memory Management
→ Simplified development

Integrated Development Environment (IDE) for C# →

An IDE is a tool that helps you write your programs Microsoft provides the following development tools for C# Programming:

→ Visual Studio (VS)
→ Visual C# Express (VCE) / Visual Studio Express
→ Visual Web developer.

Using these tools, you can write all kinds of C# programs from Simple Command-line applications to more complex applications. You can also write C# ~~Code~~ Source Code files using a basic text editor like Notepad, and Compile the code into assemblies using the Command-line Compiler, which is again a part of the .NET Framework.

GOLDEN BOY

Introduction to C# (Pronounced "C Sharp") →

C# is a simple, modern, general purpose object-oriented and type-safe programming language developed by Microsoft, USA and Approved by ECMA (European Computer Manufacturers Association) and ISO (International Standards organization.

C# was developed by Andres Hejlsberg and his team during the development of .Net framework. C# is designed for Common language Infrastructure (CLI), which consists of the executable code and runtime environment that allows use of various high-level languages on different computer platforms and architecture.

History of C# language → During the development of .NET, the class libraries were originally written in a language called Simple managed code (SMC) and later the language had been renamed as C#.

The following reasons make C# a widely used professional language:

→ It is a modern, general-purpose programming language.
→ It is object-oriented
→ It is component oriented
→ It is easy to learn
→ It is a structured language
→ It produces efficient programs
→ It can be compiled on a variety of computer platforms
→ It is part of .Net framework

Overview of C# → C# can be used to develop two ~~types~~ categories of programs, that is

1 → Executable application programs - These programs are written to carry out certain tasks and require the main method in one of the classes.

2 → Component libraries → These do not require the main declaration because they are not stand-alone application programs. These are written for use by other application.

Characteristic of C# →

SIMPLE:

1→ C# is Simple by eliminating some operators such as "::" or "→" and pointers

2→ C# treats integer and Boolean data types as entirely two different data types. Boolean values are pure true or false values in C# so no more errors of "=" operator and "==" operator. "==" is used for comparison and "=" is used for assignment operation

MODERN: C# is modern language because of the following features:

1→ It provides Automatic garbage Collection.

2→ It provides Rich intrinsic model for error handling

3→ It provides Modern approach to debugging

4→ It provides robust Security model.

5→ It provides Decimal data types for financial applications.

Object oriented →

1→ C# truly object oriented. It Support Data Encapsulation, inheritance, polymorphism, interfaces.

2→ In C#, everything is an object. There are no more global functions, variables and Constants.

Type Safe →

1→ In C# we cannot perform unsafe Casts like Convert double to a Boolean.

2→ All dynamically allocated objects and arrays are initialized to Zero.

3→ Usage of any uninitialized variable produces an error message by the Compiler.

4→ Access to Arrays is range checked and warned if it goes out of boundary.

5→ C# checks the overflow in arithmetic operations.

6→ C# Supports automatic garbage Collection.

Interoperability → C# provides Support for using COM objects, No matter what language was used to develop them. C# also supports a Special feature that enable a program to Call out any native API.

GOLDEN BOY
ALOK

Consistant → C# Supported an ~~undefi~~ unified system, which eliminates the problem of varying integer types. All types are treated as objects and developers can extend the type system simply and easily.

Versionable → Making new versions of SW modules work with the existing applications is known as versioning. With the help of new and override keywords, with this support, a programmer can guarantee that his new class library will maintain binary compatibility with the existing client applications.

Compatible → C# enforces the .Net CLS (Common language specification) and therefore allows interoperation with other .NET language. C# provides support for transparent access to COM and OLE Applications.

Applications of C#

→ Console applications
→ Windows applications
→ Developing Windows Controls
→ Developing ASP.Net projects
→ Creating Web Controls
→ Providing Web Services
→ Developing .Net Component library

NOTES:→

→ C# is case sensitive.
→ All Statements and expression must end with a Semicolon (;)
→ The program execution starts at the Main Method.
→ Unlike Java, program file name could be different from the class name.

ALOK

C# Program that prints the words "Hello World":

```
Using System;
namespace HelloWorldApplication
{
    class HelloWorld
    {
        public Static Void Main(String[] args)
        {
            Console.WriteLine("Hello World");
            Console.ReadKey();
        }
    }
}
```

Using System:

→ Using is the keyword, which is used to specify namespace.

→ The above line tells the compiler to look into the system namespace library for unresolved class names.

→ Because of the above line we have not used prefix to the Console class in the output line.

→ When the compiler parses the Console.WriteLine method, it will understand that the method is undefined, However it will search through the namespaces specified in using directives and upon finding the method in the system namespace, will compile the code without any complaint.

Namespace Declaration → A namespace is a collection of classes. The HelloWorld application namespace contains the class ~~HelloWor~~ HelloWorld.

Class Declaration → The next line has a class declaration, the class HelloWorld contains the data and method definitions that your program uses. Classes

→ Writeline(...) allo... generally contain multiple methods. Methods define the behavior of the class. However, the HelloWorld class has only one method Main.

Class HelloWorld :

→ The above line Declares a class, C# is true object oriented language and therefore everything must be placed inside a class.

→ class is Keyword, which declares new class definition.

→ HelloWorld is the name of the class. It should be a valid identifier.

Braces { } → C# is a block structured language and the code always enclosed by { and }, therefore every class definition in C# begins with opening brace { and end with closing brace }.

Public Static Void Main ( ) →

→ The above line defines a method named main. Every C# executable program must include the Main() method in one of the classes. This is the starting point for executing the program. A C# application can have any number of classes but only one class can have the Main() method to initiate the execution.

→ The above line contains public, Static and Void Keywords

public : It tells the C# compiler that Main() method is accessible by anyone.

Static : The Keyword Static declares the Main() method is a global one and can be called without creating an instance of the class.

Void → The keyword void is a type modifier that tells that the main method does not returns any value.

→ In contrast to other languages main has capital, not lower case M.

Output line : System.Console.WriteLine("Hello World");

→ The WriteLine is the Static method of the Console class, which is located in System namespace.

→ . is represented as the member access operator.

→ ; is called the terminator

ALOK

→ WriteLine always append a newline character to the end of the string. This means any subsequent output will start on a new line.

Comments → Comments play an important role in the maintenance of programs. They are used to enhance readability and understanding of code. C# supports two types of comments

→ Single line Comments → Single line comments begin with a double backslash (//) symbol and terminate at the end of the line.

example → // C# program

→ Multiline Comments → If we want to use multiple lines for a comment, this comment starts with /* and terminates with */.

Example → /* this is Example of C# CSharp program */

Console.ReadKey(); → The line Console.ReadKey(), makes the program wait for a key press and it prevents the screen from running and closing quickly when the program is launched from Visual Studio.Net.

C# language Fundamentals →

Tokens → The smallest, non-reducible, textual elements in a program are referred to as tokens. The Compiler recognizes them for building up expressions and Statements. A C# program is a collection of tokens, Comments and Whitespaces

The C# tokens are :

→ Keywords
→ Identifiers
→ Literals
→ Operators
→ Punctuators

Keywords → Keywords are reserved words predefined to the C# Compiler. They have fixed meanings and these meanings cannot be changed. Keywords serve as basic building blocks for program statements. All keywords must be written

in lowerCase. Keywords cannot be used as identifiers except when they are prefaced by the @ character.

Eg. ~~Reserved~~ Reserved Keywords → abstract, catch, interface, null, sealed, switch, volatile, public, true, using etc.

In C#, some identifiers have special meaning in context of code, such as get and set, these are called Contextual Keywords.

Example Contextual Keywords → add, global, alias, group, into, set, get, let, orderby etc.

~~Literals~~ →

Identifiers → An identifier is a name, which is used to identify a class, variable, functions, interface and label. The basic rules for framing identifiers in C# are as follows:

→ They can have alphabets, digits (0-9), or underscore.
→ They must not begin with a digit.
→ Upper case and lower case letters are distinct.
→ It should not be a C# Keyword.

Literals → Literals are value constants assigned to variables in a program. C# Supports several types of literals.

Numeric literals:
Integer Literals → An integer literal refers to a sequence of digits. There are three types of integers, namely decimal & hexadecimal integers.

Decimal Integer → Decimal integer contains a set of digits from 0 to 9 with an optional + or − Sign. Example :→ 123, −76657, +736, 0

Hexadecimal integer → A Sequence of digits preceded by 0x (or) 0X is considered as a hexadecimal integer. It may also include alphabets A through F (or) a through f. A letter A through F represents the numbers from 10 through 15.

Example → 0x2, 0xabc, 0X53674, etc.

GOLDEN BOY
ALOK

Real literals → The number containing fractional part is called real Const (11)
floating-point literals. We have two Notations for representing real literals
→ Decimal notation
→ ~~Specific~~ Scientific notation Const Exponential notation.

Decimal Notation → It represents number having a whole number followed by decimal point and fractional part. Example → 215., -.75, +.5, .95 etc

Scientific Notation → A real no. may also be expressed in exponential Const Scientific Notation. Example → 215.65 may be written as 2.15502 in exponential Notation. 02 means multiply by 10^2.

Syntax :→ mantissa e exponent
Example → 0.65e+4, 12e-2
Mantissa → mantissa is either a real no. expressed in decimal notation
Exponent → It is an integer number with an optional Sign.
E Const e → It separates the mantissa and the exponent can be written either in uppercase Const lower case.

Exponential notation ~~is~~ is useful for representing the number that are ~~either~~ either Very large or ~~very~~ Very small in magnitude.

Boolean literals → There are two boolean literal values.
true & False.

Character literals:—
SingleCharacter literals → A Single character Constant Contains a Single character enclosed by Single quote marks is called Single character Constants
Example → 'a', '#', '2',

String Constants → A String Constant is a sequence of characters enclosed by double quotes. The characters may be any letter, digits, special characters and blank space.
Example → "2001", "Well done", "Welcome to C# programming".

Backslash character Constants → C# supports some special backslash character Constants that are used in output methods. Each backslash character Constant Contains two characters to represent a single character. These combinations are

Called escape sequences.

1→ \a – alert, 2→ \b – Backspace, 3→ \f – Form feed, 4→ \n → New line, 5→ \\ Backslash
5→ \r → Carriage return, 6→ \t – Horizontal tab, 7→ \v → Vertical tab, 8→ \' – Single quote
9→ \" – Double quote, 10→ \0 – Null.

Data Types→ Every Variable in C# is associated with a data type. Data types specify the size and values that a variable can store. The data types in C# are primarily divided into two types:

→ Value types
→ Refrence Types

Value types and reference types differ in two characteristics
→ Where they are stored in memory
→ How they behave in the context of assignment statements.

Value Type→ Value Type (which are of fixed length) are stored on the stack (Stack is a place where the data stores is fixed length like float, int etc.), when a value of a variable is assigned to a variable the value is actually copied. This mean that two identical copies of the value are available in memory.

The Value types of C# can be grouped into two categories, namely,

1→ User defined types (or) Complex types
2→ Predefined types (or) Simple types

User defined types (or) Complex types→ Here we define our own complex types known as User defined value types, which includes

→ Structures
→ Enumerations

Predefined types→ These are also known as Simple types (or) primitive types.
Predefined types are divided into

→ Numeric types
→ Boolean types
→ Character types

ALOK

Numeric types → Numeric types include integral types, floating point types and decimal types.

Integral types → Integral types can hold whole numbers such as 123, −96 and 5639. The size of the values that can be stored depends on the integral data type we choose. C# supports the concept of unsigned types and therefore it supports eight types of integers.

Signed integers → Signed integer types can hold both positive and negative numbers.

Type	Size	Minimum Value	Maximum Value
Sbyte	1 byte	−128	+127
Short	2 byte	−32768	+32767
int	4 byte	−2147483648	+2147483647
long	8 byte	−9223372036854775808	+9223372036854775807

Unsigned integers → unsigned integer types can hold only positive numbers.

Type	Size	Minimum Value	Maximum Value
byte	1 byte	0	255
ushort	2 byte	0	65535
uint	4 byte	0	4294967295
ulong	4 byte	0	18446744073709551615

* All integers are by default int type. In order to specify other integer types, we must append the characters U, L or UL
  → 123U (for uint type)
  → 123L (for long type)
  → 123UL (for ulong type)

Floating Point type → Floating point types holds num. containing fractional parts, such as 27.59 and −1.375. There are two kinds of floating point

Storage in C#

→ The floating point values are Single-precision nos. with a precision of Seven digits.

→ The double types represent double-precision nos with a precision of 15/16 digits.

Type	Size	Minimum Value	Maximum Value
float	4 byte	1.5×10^{-45}	3.4×10^{38}
double	8 byte	5.0×10^{-324}	1.7×10^{308}

→ Floating point nos. by default are double-precision quantities. To force them to be Single precision mode, we must append f or F to nos.

Example → 1.23f, 7.65925f.

→ Double precision types are used when we need greater precision in storage of floating point numbers.

→ Floating-point data types support a special value known as Not-a-Number (NaN). NaN is used to represent the results of operations such as dividing zero by zero, where an actual no. is not produced. Math operations NaN as an operand will produce NaN as result.

Decimal Type → The decimal type is a high precision, 128-bit data type that is designed for use in financial monetary calculations. It can store values in the range 1.0×10^{-28} to 7.9×10^{28} with 28 significant digits.

→ To specify a no. to be decimal type, we must append the character M or m to the value.

Example → 128.45M (if we omit M, the value will be treated as double).

Character Type → In order to store single characters in memory, C# provided a character data type called char. The char type assumes a size of two bytes but, in fact it can hold only a single character. Char data type has been designed to hold a 16-bit Unicode character, in which 8-bit ASCII

I GOLDEN BOY
ALOK

Code is a subset

Boolean Type → Boolean Condition can be used when we want to test a particular Condition during the execution of the program. There are two values that a boolean type can take true (or) false. Boolean type can be denoted by the keyword bool and uses only one bit of storage.

Reference type → Reference types (which are of variable length) are stored on the heap, and when an alignment b/w two reference variable occurs, only the reference is copied. The actual values remain in the same memory location. This means there are two references to a single value. The reference types can be divided into two groups.

→ User defined (or) Complex types
→ Predefined types (or) Simple types

User defined types → User defined reference types refer to those types which we defined using predefined types. They include —
- → Classes
- → Delegates
- → Interfaces
- → Arrays

Predefined types → Predefined types include two types
- → Object type
- → String type

Variables → A variable is an identifier that denotes a storage location used to store a data value. Unlike constants that remain unchanged during the execution of the program, a variable may take different values at different times during the execution of the program. Every variable has a type that determines what values can be stored in the variable.

GOLDEN BOY

Rules for forming a Variable →

→ They must not begin with a digit.

→ Uppercase and lowercase is distinct. This means that the Variable TOTAL is not same as total or Total.

→ It should not be a Keyword.

→ White Space is not allowed.

→ Variable names can be of any length.

Variable declaration → After designing Suitable variable names, we must declare the Variable before it is used in the program. Declaration does three things —

→ It tells the Compiler what the variable name is.

→ It specifies what type of data the variable will hold.

→ The place of declaration in the program decides the scope of the variable.

Syntax: data_type variable1, variable2, ---- variableN;

Here data_type must be a valid C# data type including char, int, float, double, or any user-defined data type, and variables are separated by commas. Example

```
int i, j, count;
float f, Salary;
double fii;
byte b;
Char C1, C2, C3;
decimal d1, d2;
~~uint no~~; uint m;
Ulong n;
```

Initializing Variables → The process of giving initial values to the variables is called initialization. Once the declaration has been done then initialization of the variables is done with assignment operator.

Syntax → Variable_name = Value;

before operation

ALOK

(29) (27)

It is also possible to assign a value at the time of its declaration.

Syntax → <data_type> <variable_name> = value;

Scope of Variable → The scope of the variable determines over what part(s) of the program a variable is actually available for use (active). This depends on the type of the variable and place of its declaration. C++ defines several categories of variable.

→ Static variable
→ Instance variable
→ Array elements
→ Value Parameter
→ Reference Parameter
→ Output Parameter
→ Local variable

Example →
```
class ABC
{ static int m;
  int n;
void fun(int x, ref int y, out int z, int[] a)
{ int j = 10;
  }
}
```

→ m = static variable
→ n = instance variable
→ x = value parameter
→ y = reference parameter
→ j = local variable
→ z = output parameter
→ a[0] = array element

Type Conversion → The process of converting one data type into another data type is known as type casting or type conversion. Type conversion are divided into two types:

1→ Implicit (or) up casting (or) Widening (or) automatic type conversion.
2→ Explicit (or) down casting (or) narrowing type conversion.

Implicit type Conversion (Automatic type Conversion) → Converting lower data type into higher data type is called Widening. In this one data type is automatically converted into another type as per rules described in C++ language. which means the lower level data type is converted automatically into higher level data type before operation proceeds. The result of the data type having higher level

data type.

Example → byte x=12;
int y=x;

In the above statement, the conversion of data from byte to int is done implicitly, in other words programmer's don't need to specify any type operators. Widening is safe because there will not be any loss of data. This is the reason even though the programmer does not use the cast operator the compiler does not complaint because of the lower data type is converting into higher data type. Here higher data type having the much more space to store the lower data type.

Explicit type Conversion → Converting higher data type into lower data type is called narrowing. Here we can place intended data type in front of the variable to be cast.

Syntax : data-type variablename1 = (cast-type) variablename2;

Example : double d=12.67853;
int n = (int) d;

Here we are converting higher level data type into lower level data type that means double type is converted into int type, the fractional part of the no. is lost and only 12 is stored in n. Here we are losing some digits this is the reason the compiler forces the programmer to use the cast operator when going for explicit casting.

Microsoft.Net provides three ways of type conversion :→

1→ Parsing
2→ Convert class
3→ Boxing and Unboxing

Parsing → Parsing is used to convert string type data to primitive value type. For this we use parse methods with value types.

Example →

```
using System;
class Program
{
    Static void Main()
    { String text = "500"; // Convert string to number.
      int num = int.Parse(text);
      Console.WriteLine(num);
      int num2 = int.Parse(Console.ReadLine());
      Console.WriteLine(num2);
    }
}
```

Convert class → It is used to convert one primitive type to another primitive type. Convert is the predefined class. C# provides the following built-in type conversion methods as described:

S.No	Method	Description
1	ToBoolean	Converts a type to a Boolean value, where possible
2	ToByte	Converts a type to byte.
3	ToChar	Converts a type to single Unicode character, where possible.
4	ToDateTime	Converts a type to date time structures.
5	ToDecimal	Converts a floating point or integer type to decimal type.
6	ToDouble	Converts a type to double type.
7	ToInt16	Converts a type to 16-bit integer
8	ToInt32	Converts a type to a 32-bit integer
9	ToInt64	Converts a type to a 64-bit integer
10	ToSbyte	Converts a type to a Signed byte type.
11	ToSingle	Converts a type to a Small floating point number.
12	ToString	Converts a type to a String
13	ToType	Converts a type to a Specified type
14	ToUInt16	Converts a type to a unsigned int type
15	ToUInt32	Converts a type to unsigned long type.
16	ToUInt64	Converts a type to unsigned big integer.

GOLDEN BOY

Boxing and Unboxing → In object oriented programming, methods are invoked using objects. Since the value types such as int and long are not objects, we cannot use them to call methods. C# enables us to achieve this through a technique known as boxing.

Boxing → Boxing is the process of converting a value type to the refrence type. Boxing is an implicit type casting. To work with boxing, we required predefined data type called object, object data type is capable to hold any type of data.

Example →
```
int m=10;
object om = m;
```

In the first line we created a value type m and assigned a value to m. The second line, we created an instance of object om and assign the value of m to om. From the above operation, we saw converting a value of a value type into a value of a corresponding Refrence Type. This type of operation is called Boxing.

Unboxing → Unboxing is the process of converting a refrence type to value type. Remember that we can only unbox a variable that has previously been boxed. Unboxing is an explicit type conversion.

When unboxing a value, we have to ensure that the value type is large enough to hold the value of the object (refrence type value). Otherwise, the operation may result in runtime error.

Example →
```
int m=10;
object om=m;
byte n=(byte)om;
```

The above code will produce a runtime error.
Notice that when unboxing, we need to use explicit cast. This is because in case of unboxing, an object could be cast to any type. Therefore, the cast operator is necessary for the compiler to verify that it is valid as per the specified value type.

→ break , → continue ,

ALOK

Example → int m = 10;
object ow = m;
int i = (int)ow;

The first two lines show how to box a value type. The next line (int i = (int) ow) show the extraction of value type from the object (reference type). That is converting a value of a Reference Type into a value of a Value Type. This operation is called Unboxing.

Operators → An operator is a symbol that tells the computer to perform specific mathematical or logical manipulations. Operators are used in programs to manipulate data and variables. C# has rich set of operators. C# operators can be classified into the following types:

→ Arithmetic operators → (+, −, *, /, %)
→ Relational operators → (>, <, <=, >=, !=, ==)
→ Logical operators → (&&, ||, !)
→ Assignment operators → (=, +=, −=, *=, /=, %=)
→ Increment & decrement operators → (++, −−)
→ Conditional operators → (Condition ? true_value : false_value);
→ Bitwise operators → Works on each bit of data, used in bit level programing (&, |, ~, ^, <<, >>)
→ Special operators → (sizeof, typeof, is, as, *, ? :)

Conditional Statement → A block of code that is executed based on a condition is called conditional statement. There are divided into 2 types.

→ Conditional Branching
→ Conditional looping

When a program breaks the sequential flow and jumps to another part of the code, it is called branching. Branching statements are used to execute a statement or a group of statements.

Conditional Branching → When branching takes place based on certain condition is called conditional branching. The conditional branching statements are:

→ if → Single if, if-else, if-else if-else, nested if
→ Switch

Unconditional branching → When branching takes place without any condition is called unconditional branching. The unconditional branching statements are:

→ break, → Continue, → goto, → return.

Conditional Looping → Loop Control statements are used to execute a block of code several times until the given condition is True.

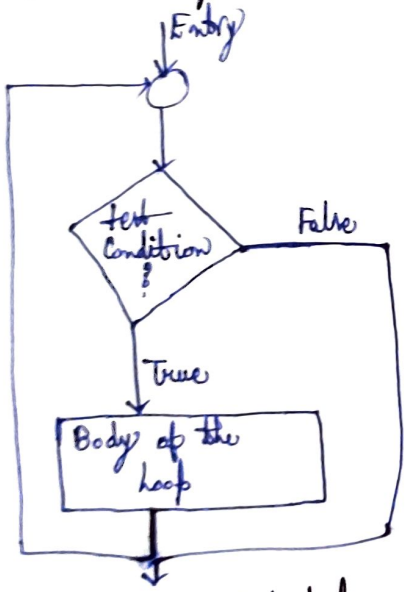
OR

The looping control statements that enables the programmer to execute a set of statements repeatedly till the required activity is completed are called looping control statements.

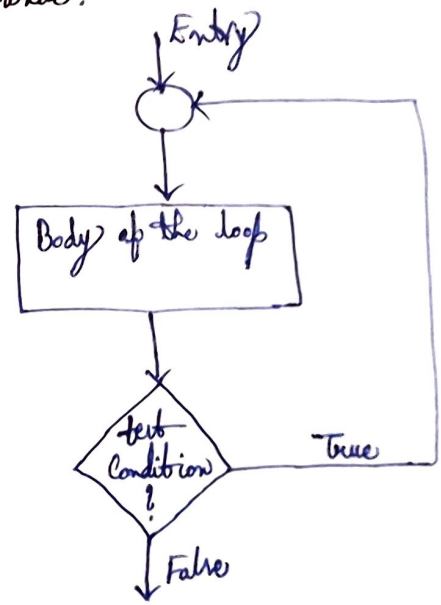
Types of Looping Control Statements → The looping control statements are divided into two types. They are —

Entry Controlled loop → In such type of loop, the test condition is checked first before the loop is executed. Examples of Entry Controlled loop are: for & while.

Exit Controlled loop → In such type of loop, the loop is executed first, then condition is checked, the loop is executed at least one time even the condition is false. Example of exit controlled loop is do-while.



Entry Controlled loop



Exit Controlled loop

For Loop :→ Repeats a block of code multiple times until a given condition is true. Initialization, looping condition and update expression (increment/decrement) is part of for loop.

Syntax: — for (initialization ; Condition ; increment)

{

Statement(s);

}

GOLDEN BOY
ALOK

While Loop → A While loop repeats a statement or group of statements until a given condition is true. It tests the condition before executing the loop body.

Syntax :→ While (Condition)
{
Statement(s);
}

Do-While Loop → Similar to while loop, but it tests the condition at the end of the loop body. The block of code inside do while loop will execute at least once even though the condition is false.

Syntax : do
{ Statement(s);
} While (Condition);

Nested Loops → C# allows to use one loop inside another loop.

Syntax → for (init; Condition; increment)
{ for (init; Condition; increment)
{ Statement(s);
}
Statement(s);
}

Syntax → while (Condition)
{ while (Condition)
{ Statement(s);
}
Statement(s);
}

Syntax → do
{ Statement(s);
do
{ Statement(s);
} while (Condition);
} while (Condition);

* → In loop nesting you can put any type of loop inside of any other type of loop. For example a for loop can be inside a while loop or vice versa.

Loop Control Statements → A Loop Control statement allows the normal execution path of a program. Loop Control statements are used when we want to skip some statements inside loop or terminate the loop immediately when some condition becomes true.

Break statement → When the break statement is encountered inside a loop, the loop is immediately terminated and program control resumes at the next statement following the loop. It can be used to terminate a case in the switch statement. If you are using break statement in nested loop (i.e., one loop inside another loop), the break statement will stop the execution of the innermost loop and start executing the next line of code after the block.

Continue Statement → The continue statement is used to skip some part of loop's body. It means we can use continue statement inside any loop (for, while & do-while). It skips the remaining statements of loop's body and starts next iteration.

goto →

1→ C# supports the "goto" statement to jump unconditionally from one point to another in the program.

2→ The goto requires a label in order to identify the place where the jump is to be made.

3→ A label is any valid variable name and must be followed by a colon (:).

4→ The label is placed immediately before the statement where the control is to be transferred.

5→ The label can be anywhere in the program either before or after the goto label statement.

6→ During execution (running) of a program, when a statement like "goto" "goto begin;" is met, the flow of control will jump to the statement immediately following the label "begin:" this happens unconditionally.

7→ goto breaks the normal sequential execution of the program.

8→ If the "label:" is before the statement "goto label;" a loop will be formed and some statements will be executed repeatedly. Such a jump is known as a, "backward jump".

9→ If the "label:" is placed after the "goto label;" some statements will be skipped and the jump is known as "forward jump".

Infinite Loop→ A loop becomes infinite loop if the condition never becomes false. We can make an endless or infinite loop by leaving conditional expression empty.

ARRAYS → An array is a set of similar type values that are stored in sequential order.

→ Array is a group of related data items that shares a common name.

→ Array is collection of homogenous data elements.

In C# the array index starts at zero. That mean the first item of an array starts at 0th position. The position of last item of an array will be total number of items − 1. In C#, arrays can be declared as fixed length or dynamic.

Fixed length array→ A fixed length array can store a predefined number of items.

Dynamic array→ A dynamic array does not have a predefined size. The size of a dynamic array increases as you add new items to the array.

→ To access an element we can use array name and index value of the particular element.

→ In C# arrays are reference types. Hence it stores default values based on data types.

Types of arrays→ C#.Net supports three types of arrays.

One-dimensional array→ Arranging collection of elements in a single row can be called one dimensional array. All items in a single dimension array are stored contiguously starting from 0 to the size of the array − 1.

Declaration Syntax →

```
datatype [ ] arrayName = new data type [Size];
                OR
int [ ] arrayName = { list of values };
```

Where, → data type → It is used to specify the type of elements in the array.

→ [ ] → Specifies Subscriptional operator

→ Size → Specifies the size of array.

→ array name → Specifies the name of the array
→ new → memory allocation operator.

foreach loop → It is specially designed for accessing the value of an array (or) Collection (Lists, hash-table, Stack, queue and linked list) where for each iteration of the loop one value of the array is assigned to the loop variable and the return to it

Syntax → foreach (type variable_name in expression)
{ body of the loop;
}

where → type – Specifies the data type
→ variable_name – Specifies the array variable name
→ in → it is a keyword
→ expression → Specifies array type or Collection type.

Two-Dimensional arrays → The Simplest form of the multidimensional array is the 2-dimensional array. Arranging a list of values in rows and columns is called 2-dimensional array. In this all rows must have equal no. of elements.

A 2-dimensional array can be thought of a table, which has X number of rows and Y number of columns. Following is a 2-dimensional array, which contains 3 rows and 4 columns.

	Column 0	Column 1	Column 2	Column 3
Row 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

Thus, every element in the array a is identified by an element name of the form a[i,j], where a is the name of the array, and i and j are the subscripts that uniquely identify each element in array a.

Declaration Syntax → datatype[,] arrayName = new datatype[rows, Columns];

OR

int[,] arrayname = {list of values};

Where,
→ data type → It is used to specify the type of elements in the array.
→ [ ] → Specifies Subscriptional operator
→ rows → Specifies the row size.
→ Columns → Specifies the column size.
→ array name → Specifies the name of the array
→ new → memory allocation operator.

Example → int[,] a = new int[4,5];

Jagged Array (or) array of arrays (or) dynamic array → Jagged array is Collection of rows which contain distinct number of elements in each row, that means all the rows may not contain same number of elements.

OR

Jagged array is Collection of one-dimensional arrays of varying size.

Declaration → Declaration of a jagged array involves two brackets.

type[][] array_name = new type[3][];

OR

type[][] array_name = { list of values };

Array Class → The Array class is the base class for all the arrays in C#, Array class is defined in the System namespace. In C# every array is automatically derived from the System.Array class. The Array class provides various properties and methods to work with arrays.

Properties of the Array class → The following table describes some of the most commonly used properties of the Array class.

S.No	Property Name	Description
1	Length	Gets a 32-bit integer that represents the total no. of elements in all the dimensions of the array.
2	Rank	Gets the Rank (no. of dimensions) of the array.
3	IsFixedSize	Gets a value indicating whether the array has fixed size.
4	IsReadOnly	Gets a value indicating whether the array is read only
5	LongLength	Gets a 64 bit integer that represents the total no. of elements in all the dimensions of the array.

Methods of array class →

1 → Clear → Sets the range of elements in the array to zero, to false, or to null, depending on the element type.

2 → Copy() → Copies the range of elements from an array starting at the first element and pastes them into another array.

3 → CopyTo() → Copies all the elements of the current one dimensional array to the specified one dimensional array starting at the specified destination array index.

4 → GetLength → Gets a 32-bit integer that represents the no. of elements in the specified dimension of the array.

5 → GetLongLength → Gets a 64-bit integer that represents the no. of elements in the specified dimension of the array.

Differences between for & foreach loop →

S.No	For	Foreach
1.	Loop variable refers to the index of an array.	Loop variable refers to the values of an array.
2.	Loop variable is integer type.	The type of the loop variable is similar to the type of values inside an array
3.	It is used for both accessing and assigning values to an array.	It is used only for accessing.

Strings → Strings represents sequence of characters. C# supports two types of Strings.

→ Mutable Strings (dynamic Strings)

→ Immutable Strings.

Immutable Strings → String objects are immutable, meaning that we cannot modify the characters contained in them. String is alias for the predefined System.String class in the CLR, there are many built in operations available that work with Strings.

C# also supports regular expressions that can be used for complex string manipulations and pattern matching.

Creating a String object → C# supports a predefined reference type known as string. We can use string to declare string type objects. You can create String

object using one of the following methods:
→ By assigning a String literal to a String variable
→ Copying from one object to another.
→ Concatenating two objects.
→ Reading from the keyword.
→ Using ToString method.

By assigning a String literal to a String variable → The most common way to create a String is to assign a quoted String of characters known as String.

Example → String S1;
S1 = "abc";
or String S1 = "abc";

Copying Strings → We can also create new copies of existing strings. This can be done in two ways.
→ Using the overloaded operator (=)
→ Using the static Copy method.

Example → String S2 = S1;
String S2 = String.Copy(S1);

both these statements would accomplish the same thing, namely, copying the contents of S1 into S2.

Concatenating Strings → We may also create new strings by concatenating existing strings. This can be done in two ways.
→ Using the overloaded operator (+)
→ Using the static Concat method

Example → String S3 = S1+S2;
String S3 = String.Concat(S1,S2);

If S1 = "abc" and S2 = "xyz", then both the statements will store the the String "abcxyz" in S3.

Reading from the Keyword → It is possible to read a string value interactively from the keyboard and assign it to a String object.

String S = Console.ReadLine();

On reaching this statement the Computer will wait for a string of characters to be entered from the keyboard. When the return key is pressed, the string will be read and assigned to the String objects.

Using ToString method → Another way of creating String is to call the ToString method on an object and assign the result to a String variable.

```
int number = 123;
String numStr = number.ToString();
```

The above statement Converts the number 123 to a String "123" and then assigns the String value to the String variable numStr.

Verbatim Strings → String can also be created using Verbatim Strings. Verbatim strings are those that start with the @ Symbol. This Symbol tells the Compiler that the string should be used Verbatim even if it includes escape characters.

```
String St = @"\EBG\CSharp\String.CS";
```

Methods for immutable Strings → Compare(), CompareTo(), Concat(), Copy(), Equals(), Insert(), Join(), Remove():

Inserting Strings → We can insert the string at specified position using the Insert() method, which is available in System.String class.

Mutable Strings → String objects are mutable, meaning strings are modifiable. Mutable Strings are created using StringBuilder class.

Example →
```
StringBuilder Str1 = new StringBuilder("abc"); // With initial size of three characters.
StringBuilder Str1 = new StringBuilder(); // empty String.
```

The String object Str1 is created with an initial size of three characters and Str2 is created as an empty String. They can grow dynamically as more characters are added to them.

The StringBuilder class Supports many methods that are useful for manipulating dynamic strings.

StringBuilder methods :→

1→ Append() → Appends a string
2→ AppendFormat() → Appends string using specific format.
3→ EnsureCapacity() → Ensure Sufficient Size
4→ Insert() → Inserts a string at a specified position
5→ Remove() → Removes the specified characters
6→ Replace() → Replaces all instances of a character with a specified one.

GOLDEN BOY
AIOK

C# also supports some special functions known as properties.

String Builder properties :→

1→ Capacity → To retrieve or set the no. of characters the object can hold.
2→ Length → To retrieve or set the length.
3→ MaxCapacity → To retrieve the maximum capacity of the object.
4→ [ ] → To get or set a character at a specified position.

The System.Text namespace contains the StringBuilder class and therefore we must include the using System.Text directive for creating and manipulating mutable strings.

Structures → Structure is collection of data items of different type.
or
A Structure is a collection of heterogeneous data elements.
→ Structures are similier to classes in C#
→ In C# Structure is a value type data type.

Defining a Structure → To define a Structure, you must use the struct statement.

Syntax →
```
struct Struct_name
{
    Data member1;
    Data member2;
    Data member3;
};
```

Example→
```
struct Books
{
    public String title;
    public String author;
    public String Subject;
}
```

Declaring Structure variables → Structure variables are declared using the Structure-tag in the program. A ~~Structures~~ Structure variable declaration is similar to the declaration of variables of any other data types. It includes the following elements.
→ The Structure tag name
→ List of variable names separated by commas
→ A terminating Semicolon.

Syntax→ Structure_tag Structure_Var1, Structure_Var2, ---- Structure_VarN;
Example→ Books book1, book2, book3;

Assigning values to structure members → Structure members can be accessed using the simple dot notation.

ALOK

Syntax → Structure Variable. Structuremember = Value;

Example → book1. title = "C#";
book1. author = "XYZ";
book1. Subject = "C Sharp";

Copying Structure Variable → Two variables of the same structure type can be copied the same way as ordinary variables. If person1 and person2 belongs to the same structure, then the following assignment operations are valid.

person1 = person2; — assign person1 to person2
person2 = person1; — assign person2 to person1

Note → 1→ We can also use the operator new to create Structure variables
Syntax → Struct-tag Structure-variable = new Struct-tag ();
Example → book1 b1 = new book1();
2→ Structure members are by default private and therefore cannot be accessed outside the Structure definition.

Structures with methods → Values to the data members are assigned using Structure-variable and the dot operator. We can also assign values to the data members by using Constructors.

A Constructor is a method which is used to set values of data members at the time of declaration.

Example →
```
Struct Number
{   int number; // data member
    public Number (int value)   // Constructor
    { number = value;
    }
}
```

The Constructor method has the same name as structure-tag and declared as public. The Constructor is invoked as follows.

Number n1 = new Number (100);

The above statement creates a structure object n1 and assigns the value of 100 to its only data member number. C# does not support default Constructor. created.

Nested Structures → Structure inside the structure is nested structure.

ALOK

43

Syntax →
```
Struct employee
  { public String name;
    public int Code;
    public Struct Salary
      { public double basic;
        public double allowance;
      };
```

We can also use Structure variables as members of another Structure

```
Struct M
  { public int x;
  }
Struct N
  { public int Y;
    public M m;
  };
```

~~Cluster~~ Class Versus Structure →

S.No	Class	Structure
1	Classes are reference types	Structs are value types
2	Classes are stored on heap	Structs are stored on stack
3	Classes Support inheritance	Structures do not Support inheritance
4	Permits initialization of instance fields	Do not permits initialization of instance fields
5	Classes Supports default Constructor	Structures do not Supports default Constructors
6	Classes Supports destructors	Structures do not Supports destructors
7	Default value is NULL	
8	It Copies the reference	It Copies the value

Features of C# Structures →

→ Structures can have methods, fields, indexers, properties, operator methods & events.

→ Structures can have defined Constructors, but don't have destructors. However you can not define a default Constructor for a Structure.

ALOK

→ Structures cannot inherit other structures or classes. Structures cannot be used as a base for other structures or classes.
→ A structure can implement one or more interfaces.
→ Structure members cannot be specified as abstract, virtual, or protected.
→ When you create a struct object using new operator, it gets created and the appropriate constructor is called. Structs can be instantiated without using the new operator.
→ If the new operator is not used, the fields remain unassigned and the object cannot be used until all the fields are initialized.

Enumerations → Enumerated types allow us to create our own symbolic names for a list of related items/items. It is defined as

Syntax → enum Identifier { Value1, Value2, --- ValueN };

Example → enum Shape
{ Circle;
Square;
Triangle;
};

enum → It is the keyword which allows the user to create our own symbolic names for a list of items.

Identifier → It is the name of enumerated data type

Example → enum day { mon, tue, wed, --- Sun };

The compiler automatically assign integer digits beginning with 0 to all enumeration constants, that is value1=0, value2=1, ---- however the programmer can change the default values.

enum day { mon=0, tue=10, wed=20, --- Sun=15 };

Enumerator Initialization → By default, the value of the first enum member is set to 0, and that of each subsequent member is incremented by one. However we may assign specific values for different members.

Example → enum Days {
Sun=1;
Mon=3;
Sat=5;
};

Enumerator base types → By default, the type of an enum is int. However, We, can declare explicitly a base type for each enum. The valid base types are : byte, sbyte, short, ushort, int, uint, long and ulong.

Example →
```
enum Position : byte
    { off;
      on ;
    };
```

The values assigned to the members must be within the range of values that can be represented by the base type.

Enumerator type Conversion → enum types can be converted to their base types and back again with an explicit conversion using a cast.

Example →
```
enum Values
    { Value 0;
      Value 1;
      Value 2;
    }
- - - - -
- - - - -
Values VL = (Values) 1;
int a = (int) VL;
```

C# Namespaces → Namespaces in C# are used to organize too many classes so that it can be easy to handle the application.

In a Simple C# program, we use System.Console where System is the namespace and Console is the class. To access the class of a namespace, we need to use namespace-name.class-name. We can use using keyword so that we don't have to use complete name all time.

In C#, global namespace is the root namespace. The global::System will always refer to the namespace "System" of .Net framework.

A Namespace is designed for providing a way to keep one set of names separate from another. The class names declared in one namespace does not conflict with the same class names declared in another.