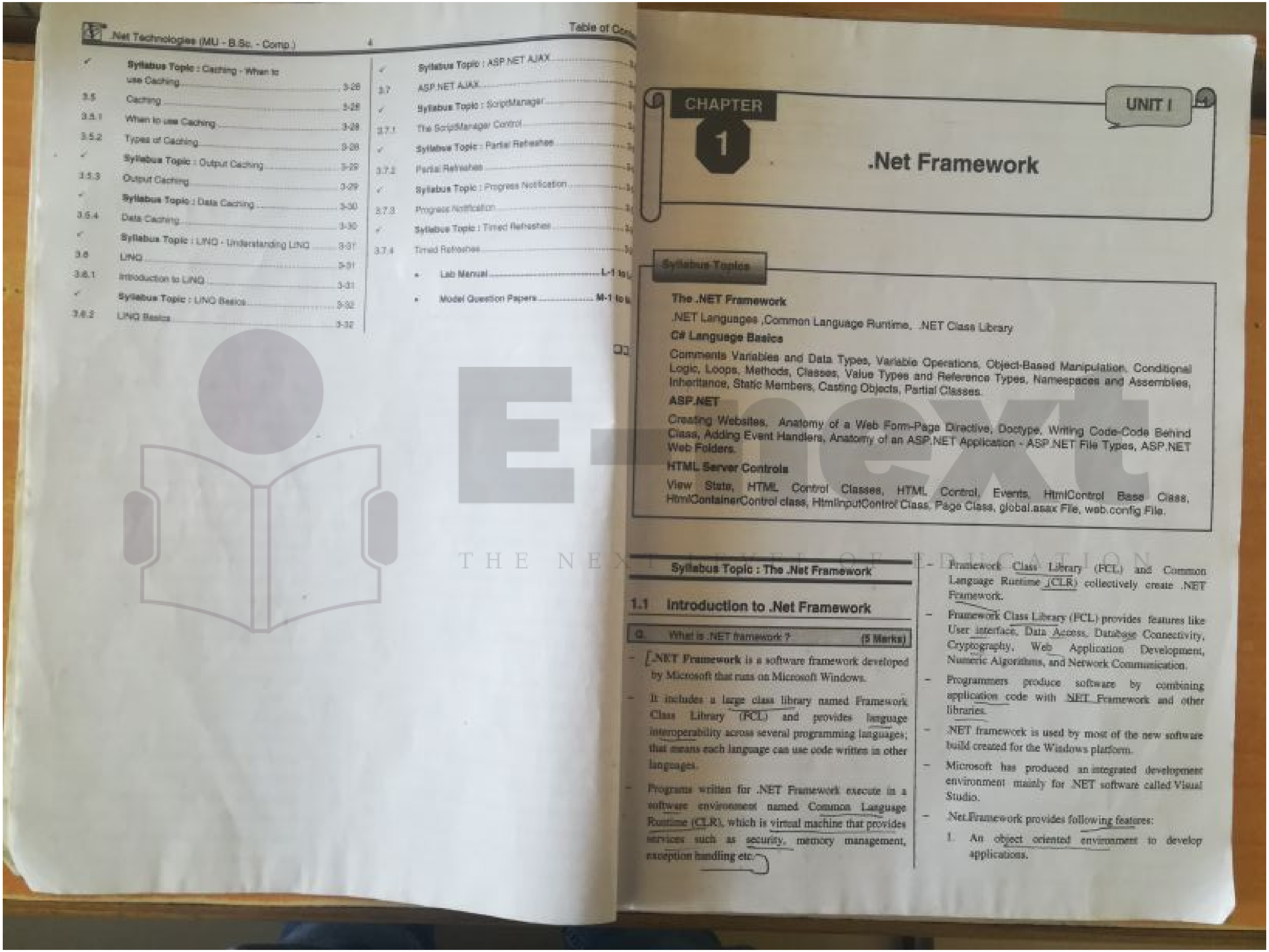


□□

CHAPTER 1

UNIT I

.Net Framework

Syllabus Topics

The .NET Framework

.NET Languages ,Common Language Runtime, .NET Class Library

C# Language Basics

Comments Variables and Data Types, Variable Operations, Object-Based Manipulation, Conditional Logic, Loops, Methods, Classes, Value Types and Reference Types, Namespaces and Assemblies, Inheritance, Static Members, Casting Objects, Partial Classes.

ASP.NET

Creating Websites, Anatomy of a Web Form-Page Directive, Doctype, Writing Code-Code Behind Class, Adding Event Handlers, Anatomy of an ASP.NET Application - ASP.NET File Types, ASP.NET Web Folders.

HTML Server Controls

View State, HTML Control Classes, HTML Control, Events, HtmlControl Base Class, HtmlContainerControl class, HtmlInputControl Class, Page Class, global.asax File, web.config File.

Syllabus Topic : The .Net Framework

1.1 Introduction to .Net Framework

Q. What is .NET framework ? (5 Marks)

- [.NET Framework is a software framework developed by Microsoft that runs on Microsoft Windows.
- It includes a large class library named Framework Class Library (FCL) and provides language interoperability across several programming languages; that means each language can use code written in other languages.
- Programs written for .NET Framework execute in a software environment named Common Language Runtime (CLR), which is virtual machine that provides services such as security, memory management, exception handling etc.
- Framework Class Library (FCL) and Common Language Runtime (CLR) collectively create .NET Framework.
- Framework Class Library (FCL) provides features like User interface, Data Access, Database Connectivity, Cryptography, Web Application Development, Numeric Algorithms, and Network Communication.
- Programmers produce software by combining application code with NET Framework and other libraries.
- .NET framework is used by most of the new software build created for the Windows platform.
- Microsoft has produced an integrated development environment mainly for .NET software called Visual Studio.
- Net.Framework provides following features:
  1. An object oriented environment to develop applications.

2. Code execution environment which helps in deployment and versioning.
3. Security to the code which is executing.
4. Language compatibility.

- In .Net user has choice to select the language in which he/she wants to develop the software.
- .NET is platform independent. It also supports language integration.

Advantages of .Net Framework

1. Good Design
2. It supports Object Oriented Programming features like class, object inheritance etc.
3. It has language compatibility.
4. Robust and Secure.
5. Supports window based application and web services.

1.1.1 .Net Framework Architecture

Q. What is .NET framework architecture ? (4 Marks)

- .Net Framework is a combination of CLR, FCL, ADO.Net and XML classes, Web/Window applications and Web services.

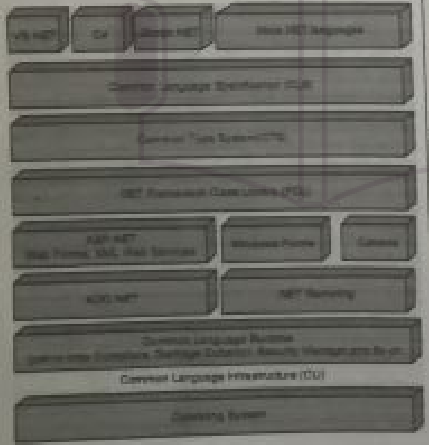


Fig. 1.1.1 : .NET Framework Architecture

- .Net Framework is a platform that provides tools and technologies to built Window based as well as Web based applications.
- The .Net architecture is divided into various modules. Each module has its own roles and responsibilities.

Syllabus Topic : .NET Languages

1. .Net Supported Languages

Microsoft itself supports most of the differe languages. .NET is the software development platfor provided by Microsoft. .NET supports more than 1 languages like VB, C#, C++, Jscript, J# etc.

2. Common Language Specification (CLS)

- A **Common Language Specification** (CLS) is document that tells how .NET language programs ca be converted into **Microsoft Intermediate Languag** (MSIL) code.
- CLS is a sub set of **Common Type System** (CTS) an it specifies a set of standards that required to b satisfied by all language compilers targeting CLR. helps in cross language inheritance and cross languag debugging.
- To fully interact with other objects without consideri their language, objects must expose to callers o those features that are common to all the languages.
- CLS specifies the rules and regulations for th languages so as to interact with other .Net languages.
- The CLS was created sufficiently big to conta the language constructs that are commonly required b developers.

3. Common Type System (CTS)

- **Common Type System** (CTS) defines some basic da types that Intermediate Language code ca understand.
- It explains set of data types that can be used in differe .Net languages in common, that means CTS giv guarantee that objects written in different .Ne languages can communicate with each other.
- Each .Net supporting language should match its da types to these standard data types.
- This makes it possible for two .Net supportin languages to communicate by sending/receivin parameters to and from each other.
- For Communicating the programs written in any .NE supporting language, the data types have to be we suitable to each other on the basic level.
- The common type system supports two gener categories of types :

(i) Value types

- Value types directly hold their data, and instances of value types are either allocated on the stack or allocated inline in a structure.
- Value types can be built-in, user-defined, or enumerations.

(ii) Reference types

- Reference types are used to store a reference to the memory address of value, and reference types are allocated on the heap.
- Reference types may in the form of self-describing types, pointer types, or interface types. The reference type is usually decided by the values of self-describing types.
- Self-describing types are further divided into two part ; arrays and class. The class types are user-defined classes, boxed value types, and delegates.

Syllabus Topic : .NET Class Library

4. .Net Framework Class Library (FCL)

- The .Net Framework Class Library (FCL) provides important functionality of .Net Framework. It is also called as Base Class Library. The .Net Framework Class Library (FCL) includes a huge group of reusable classes, interfaces, and value types that speed up the development process and provide access to in-built functionality. It is the foundation on which .NET Framework applications, components, and controls are built. It is common for all types of applications ,that means the way you access the Library Classes and Methods in VB.NET will be the same in C#.NET and it is common for all other languages in .NET
- The following are different types of applications that can make use of .net class library.
  1. Window based Application.
  2. Console based Application.
  3. Web Application.
  4. XML Web Services.
  5. Windows Services.
- Developers only need to include the FCL in their language code to use predefined functions and properties of FCL to implement frequently used and complicated functions like reading and writing data to file, graphic rendering, data manipulation and XML document manipulation.

Syllabus Topic : Common Language Runtime

5. Common Language Runtime (CLR)

- .Net Framework provides software called **Common Language Runtime** (CLR) to run programs which are written in any .NET language.
- **Common Language Runtime** (CLR) is heart of the .Net Framework.
- CLR resides above the operating system and executes all .Net programs. It handles Garbage Collection, Code Access Security (CAS), Thread Management etc.
- The source code which executes under the **Common Language Runtime** (CLR) is called as Managed Code.
- Programmers need not to worry about managing the memory if the programs are executing under the CLR because CLR provides Memory Management and Thread Management features. When our program needs memory, CLR allocates the memory to that program and de-allocates the memory when program execution is completed.
- The main function of Common Language Runtime (CLR) is to modify the Managed Code into native code and then execute the Program.
- Language Compilers like C#, VB.Net, J# compiler will translate the Program to **Microsoft Intermediate Language** (MSIL)code and then CLR translate that MSIL code into native code.
- The **Microsoft Intermediate Language** (MSIL) code or Managed Code compiled only when it needed. The Common Language Runtime's Just In Time (JIT) compiler converts Intermediate Language (MSIL) to native code on demand at application run time.

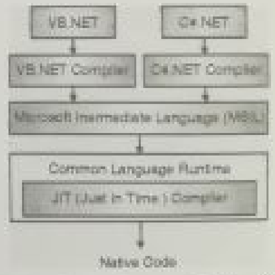


Fig. 1.1.2 : Common language Runtime(CLR)

- **Microsoft Intermediate Language (MSIL) Code :** When we compile our .Net code, it is not directly transformed to native or binary code; it is first transformed into intermediate code known as **Microsoft Intermediate Language (MSIL) code** which is then interpreted by the CLR. MSIL is not dependent on hardware and the operating system. Cross language compatibility i.e. program from one language can convert easily into another language is possible because MSIL is the same for all .Net languages. MSIL is further converted into native code by CLR.
- **Just in Time Compilers (JIT) :** It compiles Intermediate Language (IL) code into native executable code (.exe or .dll). Once code is converted to Intermediate Language (IL) then it can be called again by JIT instead of recompiling that code.

1.2 C# Language Basics

- C# is an advanced, object-oriented as well as general purpose programming language developed by Microsoft.
- C# is approved by ECMA (European Computer Manufacturers association) and ISO (International Standards Organization).
- Anders Hejlsberg and his team developed C# in the phases of .Net Framework development. C# is considered as an important language among the .Net Framework languages.

☞ Features of C#

(i) It is a latest programming language providing advanced features.

(ii) It is general-purpose.

(iii) It is object oriented.

(iv) It is component based.

(v) It is simple to learn.

(vi) It is a structured language.

(vii) It provides efficiency to programs.

(viii) It can be compiled as well as executed on a various types of computer platforms.

(ix) It is a part of .Net Framework.

☞ Strong Programming Features of C#

C# follows number of constructs from traditional high-level languages such as C and C++ and supports object-oriented paradigm. Maximum of C# concepts strongly resembles to Java. C# provides numerous strong programming features because of which it is widely used.

☞ Important features of C#

(i) Automatic Garbage Collection

(ii) Standard Library

(iii) Assembly Versioning

(iv) Properties and Events

(v) Simple Multithreading

(vi) Integration with Windows

(vii) Delegates and Events Management

(viii) Easy-to-use Generics

(ix) Indexers

(x) Conditional Compilation

Syllabus Topic : Comments

1.2.1 Comments

Q.	Explain Comments in C#.	(4 Marks

Comments can be used in document to state t functionality of the program and purpose of specific blo or lines of code. As the compiler ignores the comments, v can include them anywhere in a program without affect the source code. There are 3 types of comments in C#.

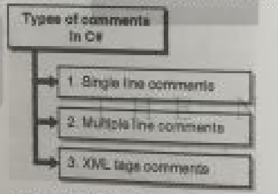


Fig. C1.1 : Types of comments in C#

➔ 1. Single line comments

Double slash (//) is used to represent single li comment.

☞ Example

```
// this is single line comment
```

➔ 2. Multiple line comments

It is used to write more than single line as commer Multiline comment start with /* and ends with */

☞ Example

```
/* This is multiline comment in c# which is simply
ignored by compiler.
It contain more than one line...*/
```

➔ 3. XML tags comments

In C#, you can create documentation for source code by inserting XML elements as special comment fields using triple slash before the code block to which the comments refer in the source code.

☞ Example

```
/// <Information>
/// This class performs important function.
/// Information>
```

Syllabus Topic : Variables and Data Types

1.2.2 Variables and Data Types

Q.	What is variable?	(2 Marks)
Q.	Explain data types in C#.	(4 Marks)

1.2.2(A) Variables

- A variable is nothing but a name given to a memory location on which our program works.
- Variable is a way to represent memory location using symbol so that it can be easily identified.
- Every variable in C# has a particular data type, that states the amount of memory required to represent that variable, the range of values which can be stored in that memory and the set of operations that can be applied on that variable.
- The Fundamental variable types available in C# can be categorized as :

Sr. No.	Variable Type	Example
1.	Decimal types	Decimal
2.	Boolean types	True or false value, as assigned
3.	Integral types	int, char, byte, short, long
4.	Floating point types	float and double
5.	Nullable types	Nullable data types

Defining Variables

☞ Syntax

```
Data-type   variable-list;
```

Here,

data_type: data_type contains any valid data type in C#, like Int, Float, Char,Double, or any user-defined data type.

variable_list: This variable_list includes one or more variable names which are separated using commas.

☞ Rules for defining variables

1. A variable can have alphabets, digits and underscore.
2. A variable name can start with alphabet and underscore only. It can't start with digit.
3. No white space and comma is allowed within variable name.
4. A variable name must not be any reserved word or keyword e.g. char, float ,if, else, switch ,goto, break, while, for etc.

☞ Example of declaring variable

1. int p, q;   2. double d;
3. float f;   4. char ch;

☞ Invalid variable names

1. int 4;   2. int x y;
3. int double;

☞ Initializing Variables

- We can initialize variable i.e. assign a value to that variable using equal to (=) sign.
- Syntax for initialization of variable is :

  variable_name=value;

☞ Example

```
//declaration
int a,b;
/* initialization */
a = 10;
b = 20;
```

- Variables can be initialized in their declaration also. The initialization of variable consists of an equal sign followed by a constant expression as follows:

```
Data-type variable_name = value;
```

- Some examples are :

  1. int d =3, f =5;   2. double pi =3.14159;
  3. char x='x';

1.2.2(B) Data Types

C# is a strongly typed language, it means that we cannot use variable without data types. Data types tell the compiler that which type of data is used for processing. The variables in C# are categorized into the following types :

1) Value types 2) Reference types

3) Pointer types

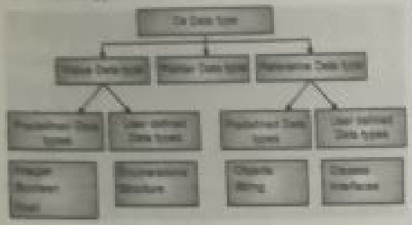


Fig. 1.2.1 : Data Types of C#

Types	Data Types
Value Data Type	int, char, float, Boolean, etc
Reference Data Type	String, Class, Object and Interface
Pointer Data Type	Pointers

1. Value Data Type

- Value type variables can be assigned a value directly. They are derived from the class **System.ValueType**.
- The value data types are integer-based and floating-point based. The value types directly contain data. C# language supports both signed and unsigned literals.
- There are 2 types of value data type in C# language :
  - (i) **Predefined Data Types** : It includes Integer, Boolean, Float, etc.
  - (ii) **User defined Data Types** : It includes Structure, Enumerations, etc.
- The memory size of data types may change according to 32 or 64 bit operating system.
- Following are the value data types. Size is given according to 32 bit Operating System.

Sr. No.	Data Types	Memory Size	Range
1.	Char	1 byte	– 128 to 127
2.	signed char	1 byte	– 128 to 127
3.	unsigned char	1 byte	0 to 127
4.	Short	2 byte	– 32,768 to 32,767
5.	signed short	2 byte	– 32,768 to 32,767
6.	unsigned short	2 byte	0 to 32,767
7.	Int	2 byte	– 32,768 to 32,767
8.	signed int	2 byte	– 32,768 to 32,767
9.	unsigned int	2 byte	0 to 32,767
10.	short int	2 byte	– 32,768 to 32,767
11.	signed short int	2 byte	– 32,768 to 32,767
12.	unsigned short int	2 byte	0 to 32,767
13.	long int	4 byte	
14.	signed long int	4 byte	
15.	unsigned long int	4 byte	
16.	float	4 byte	
17.	double	8 byte	
18.	long double	10 byte	

2. Reference Type

- The reference types do not contain the actual stored in a variable, but they contain a reference t variables i.e. address of variable. If the data is cha by one of the variables, the other variable automati reflects this change in value.
- There are 2 types of reference data type in C# langu
  - (i) **Predefined Types** : such as Objects, String.
  - (ii) **User defined Types** : user defined data types Classes, Interface which we see later.

i) Predefined Types

a) Object Type

- The **Object Type** is the ultimate base class fo data types in C# Common Type System (C i.e.in the unified type system of C#, all ty predefined and user-defined, reference types a value types are inherited directly or indir from Object.
- The object types can be used to assign valu any other types such as value types, refer types, predefined or user-defined types.

- Before assigning values, we must required to do type conversion.
- The process of converting value type to object type is known as **boxing**.
- And the process of converting object type to a value type is known as **unboxing**.
- In the following example, the integer variable i is boxed and assigned to object o.

Example

```
int i = 123;
// The following line boxes i.
object o = i;
```

- The object o can then be unboxed and assigned to integer variable i :

```
i = (int)o; // unboxing
```

b) String Type

- The **String Type** permits us to assign any string type value to a variable. The string type is an alternative for the System.String class. It is inherited from object type.
- for a string type ,we can provide value through string literals using two ways: quoted and @quoted.

Example

```
String name="C# Example";
```

- Example of @quoted way

```
@" C# Example"
```

ii) user-defined data types

The user-defined data types are: class, interface, or delegate. We will see them later.

iii) Pointer Data Type

- Pointer type variable store the memory address of variable.
- Pointer in C# is same as pointer in C or C++.

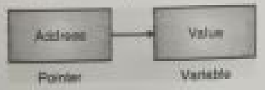


Fig. 1.2.2

- Ampersand sign(&) is used to assign address of variable to pointer which is called as Address of operator and asterisk sign(*) is used to access the value stored at that address which is called as Indirection operator.

Syntax

```
Data-type *identifier;
```

Example

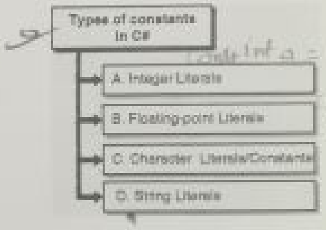
(i) char *cptr;

(ii) int *iptr;

1.2.3 C# Constants

> **Q.** What is constant ? Explain types of constats in c#. **(4 Marks)**

- The constant refers to variable whose value cannot be modified while throughout the execution of program.
- We cannot be assign value to constant variable at run time; we must assign value to constant at compile time i.e. at time of the declaration of constant variable.
- Constants can declared by using the const keyword.
- The fixed values are also called literals.
- Constants can be of any of the basic data types like an integer constant, a floating constant, a character constant, or a string literal.
- Following are the types of constants in C#.



constant pi = 3.14

Fig. C1.2 : Types of constants in C#

(A) Integer Literals

- Decimal or octal or hexadecimal constant are used to represent integer literal.
- Prefix is used to state the base for ineger literal. 0x or 0X is used as prefix for hexadecimal and we can not add decimal literal.

- An integer literal can have a suffix that is a combination of U or u and L or l. U or u is used to indicate unsigned integer and L or l for long integer.

Examples

(i) 55 (decimal integer constant)

(ii) 0x2a (hexadecimal integer constant)

(iii) 15u (unsigned integer constant)

(iv) 234l (long)

(v) 22ul (unsigned long)

(B) Floating-Point Literals

Integer part , decimal point, a fractional part, and an exponent part are included in floating-point literal . Sometimes floating-point literal consist of exponential part which is denoted by E or e.

Examples

(i) 3.14159

(ii) 314159E-5F

(C) Character Literals/Constants

- Character literals are written in single quotes.
- A character literal can be a plain character such as 'a' or an escape sequence such as '\t','\n' etc.
- There are some characters in C#. When they starts with backslash they have specific meaning and these characters are called as escape sequence character.

Examples

List of escape sequence character are as follows :

Sr. No.	Escape sequence	Meaning
1.	\\	Used to represent \ character
2.	\'	Used to represent ' character
3.	\"	Used to represent " character
4.	\?	Used to represent ? character
5.	\a	Used to represent Alert or bell
6.	\b	Used to represent Backspace
7.	\f	Used to represent Form feed
8.	\n	Used to represent Newline
9.	\r	Used to represent Carriage return
10.	\t	Used to represent Horizontal tab
11.	\v	Used to represent Vertical tab
12.	\xhh . . .	Used to represent Hexadecimal number of one or more digits

(D) String Literals

- String literals are written in double quotes " " or can written using @" ".
- A string is array characters. A string contains characters such as plain characters, escape sequences, universal characters.
- You can split a long line into multiple lines using string literals and this is done by separating the long line using whitespaces.

Example

1) "hello dear"

2) "hello dear"

3) @"hello dear"

Declaring the constant

Syntax

```
const<data_type><constant_name>= value;
```

Example

(i) int const a=10;

(ii) int const b=20;

Program 1.2.1

Write a program to demonstrate how to define and use a constant in program.

Solution :

Program that demonstrates how to define and use a constant in program

```
using System;
```

the using keyword is used to add the System namespace in our program. A program usually contains more than one using statement.

```
namespace ConstantDeclarationDemo
{
Class constantDemo
{
Static void Main(string[] args)
{
const double pi =3.14159;
double r;

Console.WriteLine("Enter Radius: ");


    r =Convert.ToDouble(Console.ReadLine());

double areaofCircle = pi * r * r;
Console.WriteLine("Radius of circle : {0}, Area of circle: {1}", r, areaofCircle);
Console.ReadLine();
}
}
}
```

This statement represent namespace declaration. namespace is a collection of classes.

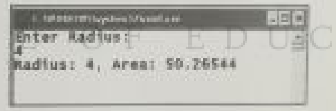
declaration of constant

This function is used to print message to console

Console.ReadLine() function is used to read input from user and Convert.ToDouble() convert that input into double datatype

Display zeroth and first argument as result

Output



```
Enter Radius:
4
Radius: 4, Area: 50.26544
```

Syllabus Topic : Variable Operations

1.2.4 Variable Operations

Q. Explain types of operators in C#. (4 Marks)

An operator is a symbol that tells the compiler to perform specific mathematical or logical operation or action on operand.

Operand is variable or literal on which operator can be applied to perform specific task.

C# has large set of built-in operators.

Following are types of operators which perform different types of operations on variable in C# language.

- Basically operators are classified in following three types on basis of number of operands they require to perform operation.

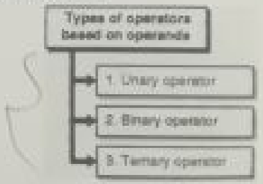


Fig. C1.3 : Types of operators based on operands

1. Unary operator

Operator which require only one operand to operate is called as Unary operator. For e.g. pre increment(++a), post increment(a++), predecrement(--a), postdecrement(a--) etc.

2. Binary operator

Operator which require two operands to operate is called as binary operator. For e.g. +,- etc.

3. Ternary operator

Operator which require three operands to operate is called as Ternary operator. e.g. conditional operator (?:).

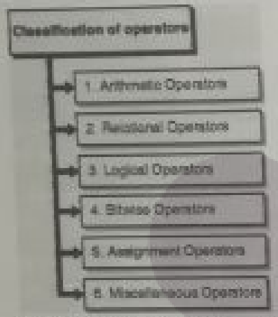


Fig. C1.4 : Classification of operators

1) Arithmetic Operators

Arithmetic operators are binary operators which require two operands to perform operation. Table 1.2.1 contain all the arithmetic operators available in C#. Assume x =10 and y=20 then

Table 1.2.1

Sr. No.	Operator	Description
1.	+	Used to adds two operands
2.	-	Used to Subtracts second operand from the first operand
3	*	Used to do multiplication of two operands
4	/	Used to divide numerator by de-numerator and find quotient
5.	%	This operator is called as Modulus operator and it is used to calculate remainder after division of two operands
6.	++	This is Increment operator which is unary operator used to increase value of integer by one
7.	--	This is Decrement operator which is unary operator which decreases value of integer by one

2) Relational Operators

Relational operators are binary operators i.e Rela operators requires two operators to perform task, t table display all the relational operators availa C#. Assume x =10 and y = 20, then

Table 1.2.2

Sr. No.	Operator	Description	Examp
1.	==	It is used to Check wheather the values of two operands are equal or not. if values of two operands are equal then condition evaluats to true.	(x == y) true.
2.	!=	It is used to Check wheather values of two operands are equal or not. if values of two operands are not equal then condition evaluats to true.	(x != y) is
3.	>	It is used to Check wheather value of left operand is greater than the value of right operand.if left operand is greater than the value of right operand then evaluated to true.	(x > y) true.
4.	<	It is used to Check wheather value of left operand is less than the value of right operand. if value of left operand is less than the value of right operand then condition evaluated to true.	(x < y) is
5.	>=	It is used to Check wheather value of left operand is greater than or equal to the value of right operand.if yes then condition evaluated to true.	(x >= y) true.
6.	<=	It is used to Check wheather value of left operand is less than or equal to the value of right operand. if yes then condition evaluated to true.	(x <= y) is

3) Logical Operators

Table 1.2.3 display all the logical operators present in C#. In logical operators includes operators such as logical And (&&), logical or (||) which are are binary operators and logical not(!) which is unary operator . Assume variable X contains Boolean value true and variable Y contains Boolean value false, then

Table 1.2.3

Sr. No.	Operator	Description	Example
1.	&&	This operator is known as Logical AND operator. If both the operands contains non zero value then condition evaluated to true.	(X && Y) is false.
2.	||	This operator is known as Logical OR operator. If any one oprand from two operands has non zero value then condition evaluated to true.	(X || Y) is true.
3.	!	This operator is known as Logical NOT operator. It is used to reverses the logical state of the operand. If a condition is true then Logical NOT operator will convert it to false.	

4) Bitwise Operators

Bitwise operator operate on single bit at a time and carry out bit by bit operation. The truth tables for &(AND), |(OR), and ^(XOR) 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

The Bitwise operators which are available in C# are given in the below Table 1.2.4. Assume variable X holds 14 (binary equivalent: 0000 1110) and variable Y holds 11 (binary equivalent: 0000 1011), then

Table 1.2.4

Sr. No.	Operator	Description	Example
1.	&	This Binary AND Operator copy single bit at a time to the result if that bit presents in both operands.	(X& Y) = 10, which is 00001010
2.	|	This Binary OR Operator copy single bit at a time to the result if that bit presents in either operand.	(X| Y) = 15, which is 00001111
3.	^	This Binary XOR Operator copy single bit at a time to the result if that bit is set in one operand but in not in both.	(X^ Y) = 5, which is 00000101
4.	~	This Ones Complement which is unary Operator has the effect of flipping bits i.e convert 1 to 0 and 0 to 1	(~X) = 241, which is 1111 0001
5.	<<	The Left Shift Operator is Binary operator. The value of left operand is moved to left by the number of bits stated using the right operand.	X << 2 = 56, which is 00111000
6.	>>	This is Right Shift Operator which is binary operator. The value of left operand is moved right by the number of bits specified by the right operand.	X >>2 = 3, which is 00000011

Assume if X = 14; and Y = 11; then in the binary format they are as follows :

X = 0000 1110

Y = 0000 1011

X&Y = 0000 1010

X/Y = 0000 1111

X^Y = 00000101

~X = 1111 0001

➜ 5) Assignment Operators

There are following assignment operators supported by C# are listed in Table 1.2.5.

Table 1.2.5

Operator	Description	Example
=	assignment operator which assigns value of right side operand to left side operand.	X= 20 assigns value 20 into X
-=	This is Subtract AND assignment operator which subtracts right operand from the left operand and store the result of subtraction in left operand.	X -= Y is same as X = X - Y
/=	This is Divide AND assignment operator which divides left operand with the right operand and store the result of division in left operand.	X /= Y is same as X = X / Y
+=	This operator is used to add right operand to the left operand and store the result in left operand.	X += Y is same as to X = X + Y
*=	This is Multiply AND assignment operator which perform multiplication of right operand with the left operand and store the result of multiplication in left operand.	X*= Y is same as X = X * Y
%=	This is Modulus AND assignment operator which calculate remainder using two operands and store the result in left operand.	X %= Y is same as to X = X % Y
<<=	This is Left shift AND assignment operator in which value of left operand is moved to left by the number of bits stated using the right operand and result is stored in left operand.	X<<= 3 is same as X = X << 3
>>=	Right shift AND assignment operator in which value left operand is moved to right by the number of bits specified by the right operand and result is stored in left operand.	X >>= 2 is same a X = X
&=	This is Bitwise AND assignment operator. This Operator copy single bit at a time to the result if that bit presents in both operands and this result is stored in left operand.	X &= 2 same a X= X &
^=	This bitwise exclusive OR assignment operator.	X ^= 2 same a X= X^2
|=	This is bitwise inclusive OR assignment operator which copy single bit at a time to the result if it that bit is set in one operand but in not in both and result is stored in left operand.	X |= 2 same a X = X |

➜ 6) Miscellaneous Operators

There are some important operators which i sizeof, typeof and **Conditional operator** (? : **are available in C#** are listed in Table 1.2.6.

Table 1.2.6

Sr. No.	Operator	Description	Example
1.	sizeof()	This operator is used to return the size of a data type.	sizeof(float) returns 4.
2.	typeof()	This operator is used to return the type of a class.	typeof(StringRea
3	&	This operator is used to return the address of variable.	&x; returns address of variable.
4.	*	This operator is used to represent the Pointer which store address of variable.	*x; creates pointer 'x'

5.	? :	This operator is known as Conditional Expression which i used to check condition.	x=10;
			y=20;
			x>y?x:y

6.	is	This operator is used to states whether an object is of a certain type or not.	String name;
			If(name is String)
			checks name is an object of the string class.

7.	as	This operator perform type casting without occourance of an exception if the type casting fails.	Object obj = new StringReader("Tech Max");
			StringReader r = obj as StringReader;

1.2.4(A) Precedence of Operators in C#

Q. What is mean by operator precedence ? (2 Marks)

- The precedence of operators specifies that which operator will be evaluated first and which next in an expression.
- The associativity specifies the operators direction to be operate, it may be left to right or right to left.
- The precedence and associativity of C# operators is given in Table 1.2.7.

Table 1.2.7

Sr. No.	Category (By Precedence)	Operator(s)	Associativity
1.	Unary	+ - ! ~ ++ -- (type)* & sizeof	Right to Left
2.	Additive	+ -	Left to Right
3.	Multiplicative	% *	Left to Right
4.	Relational	<><= >=	Left to Right
5.	Shift	<<>>	Left to Right
6.	Equality	== !=	Right to Left
7.	Logical AND	&	Left to Right
8.	Logical OR	|	Left to Right
9.	Logical XOR	^	Left to Right
10.	Conditional OR	||	Left to Right
11.	Conditional AND	&&	Left to Right
12	Null Coalescing	??	Left to Right
13.	Ternary	?:	Right to Left
14	Assignment	= *= /= %= += -= <<= >>= &= ^= |= =>	Right to Left

1.2.5 C# Keywords

Q. What is keyword and list some keyword in C# ? (2 Marks)

- Keywords are predefined sets of reserved words that have special meaning in a program.
- The meaning of keywords cannot be changed; neither can they be directly used as an identifier in a program.

- A list of Reserved Keywords available in C# programming language is given below :

abstract	base	as	bool	break	catch	case
byte	char	checked	class	const	continue	decimal
private	protected	public	return	readonly	ref	sbyte
explicit	extern	false	finally	fixed	float	for
foreach	goto	if	implicit	in	in (generic modifier)	int
ulong	ushort	unchecked	using	unsafe	virtual	void
null	object	operator	out	out (generic modifier)	override	params
default	delegate	do	double	else	enum	event
sealed	short	sizeof	stackalloc	static	string	struct
switch	this	throw	true	try	typeof	uint
abstract	base	as	bool	break	catch	case
volatile	While					

Syllabus Topic : Object-Based Manipulation

1.2.6 Object-Based Manipulation

C# is a pure object oriented programming languages. C# supports all the object oriented programming concepts like :

1. **Object :** Object is an entity having its own properties. We can take an example of flower. Flower is an object having properties like color, fragrance etc.
2. **Class :** Class is collection of data members (variables, arrays, etc) and member methods.
3. **Encapsulation :** Wrapping up of data members and member methods is known as encapsulation.
4. **Polymorphism :** It means to take out more than one forms. That means single entity can behave differently in different situation.

☞ **Example**

+ operator : for numerical operands, its gives addition while for string type of operands it gives concatenation.

3 + 5 = 8

a + b = ab

5. **Inheritance :** Creating a new class (child) from existing class (parent) is known as inheritance. In this case properties of parent class are accessible is child class.

Note : The important OOP concepts we are going to learn in next sections in detail.

Syllabus Topic : Conditional Logic

1.2.7 Conditional Logic

- Conditional logic is also called as Decision ma structures.
- Decision making structures need that progr states one or more conditions that are checked b program.
- In Decision making statements, statements wi executed if the condition is evaluated to true statements after if block are executed when condition is evaluated to false.
- Most control structures in C# are similar to those or C++, but there are some specific differences.
- C# provides following types of decision ma statements.

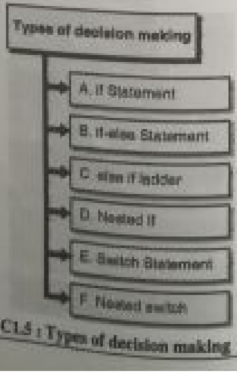


Fig. C1.5 : Types of decision making

1. **if statement** : An if statement consists of a expression which may evaluate to true or false, followed by one or more statements.
2. **if...else statement** : Statement followed by if block are executed when condition is true, if the condition is evaluated to false else part get executed.
3. **else if ladder :** The "else if" ladder is used to test set of conditions in a sequence. It is also considered as multi-way decision making statement.
4. **nested if statements** : One if is written in another if statement.
5. **switch statement** : It is multi-way decision making statement. It provides more alternatives to programmer than if or if-else.
6. **nested switch statements** : One switch statement appear inside another switch statement.

➜ 1.2.7(A) If Statement

Q. Explain working of if-else statement with example. **(4 Marks)**

The C# if statement tests the condition. It is executed if condition is true.

☞ **Syntax**

```
if(condition)
{
//code to be executed
}
```

Program 1.2.2

Write a program to check wheather given number is even or not.

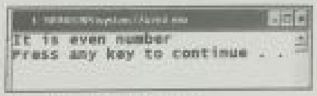
Solution :

Program to check even or not number

```
using System;
namespace EvenNo
{

public class IfDemo
  {
    public static void Main(string[] args)
    {
      int num = 10;
      if (num % 2 == 0)
      {
        Console.WriteLine("It is even number");
      }
    }
  }
```

Checks whether number is divisible by 2 or not. If remainder is 0 then num is even

Output



```
It is even number
Press any key to continue . . .
```

➜ 1.2.7(B) If-else Statement

In C# if-else statement, it executes the if block if condition is true otherwise executeelse block .

☞ **Syntax**

```
if(condition)
{
//code if condition is true
}
else
{
//code if condition is false
}
```

Program 1.2.3

Write a program to demonstating working of if-else loop.

Solution :

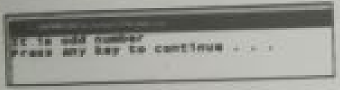
Program that demonstrating working of if-else loop

```
using System;
public class IfExample
  {

    public static void Main(string[] args)
    {
      int num = 11;
      if (num % 2 == 0)
      {
        Console.WriteLine("It is even number");
      }
      else
      {
        Console.WriteLine("It is odd number");
      }

    }
  }
```

Checks whether number is divisible by 2 or not. If remainder is 0 then num is even

Checks whether number is divisible by 2 or not. If remainder is not 0 then num is odd

Output



```
It is odd number
Press any key to continue . . .
```

1.2.7(C) else if Ladder

Q.	Explain working of else if ladder statement with example. **(4 Marks)**

- The "else if" ladder is used to test set of conditions in a sequence.
- It is also considered as multi-way decision making statement.
- Number of conditions is given in a sequence with subsequent statements.
- If any of the given condition is satisfied then the related statements are executed and the control exits from the else if ladder. That means further conditions are not going to be checked.
- But if the condition does not satisfy then the complier goes to next condition to check the condition.

Syntax

```
if(condition1)
{
   //code to be executed if condition1 is true
}
else if(condition2)
{
   //code to be executed if condition2 is true
}
else if(condition3)
{
   //code to be executed if condition3 is true
}
...
else
{
//code to be executed if all the conditions are false
}
```

Program 1.2.4

Write a Program to demonstrate working of else if ladder.

Solution :

Program to demonstrate working of else if ladder

```
using System;
namespace IfElseLadderDemo
{
public class IfExample
  {
     public static void Main(string[] args)
     {
       Console.WriteLine("Enter Marks : ");
       int marks = Convert.ToInt32(Console.ReadLine());

       if (marks  <0 || marks  >100)
       {
         Console.WriteLine("Invalid marks");
       }
         else if(marks  >= 0 && marks  < 50)
{
         Console.WriteLine("Fail");
       }
       else if (marks  >= 50 && marks  < 60)
```

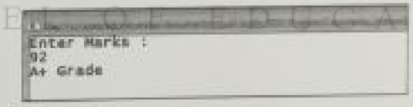
Accepts total marks of student

If num is less than zero or num is greater than 100,num entered by user is wrong

If num greater than or equal to 0 and num less than equal to 50 then grade is fail

```
       {
         Console.WriteLine("D Grade");
       }
       else if (marks >= 60 && marks < 70)
       {
         Console.WriteLine("C Grade");
       }
       else if (marks >= 70 && marks < 80)
       {
         Console.WriteLine("B Grade");
       }
       else if (marks >= 80 && marks < 90)
       {
         Console.WriteLine("A Grade");
       }
       else if (marks >= 90 && marks <= 100)
       {
         Console.WriteLine("A+ Grade");
       }
     Console.ReadLine();
     }
  }
}
```

Output



```
Enter Marks :
92
A+ Grade
```

1.2.7(D) Nested If

- Nested if means we can write one if inside another if statement. While checking number of conditions, we may need to write if statement inside another if statement.

Syntax

```
if(Boolean_expression 1)
{
       /* execute when Boolean expression 1 is true*/
        if(Boolean expression 2)
       {
            /* execute when Boolean expression 1 is true*/
       }
}
```

Program 1.2.5

Write a program for nested if-else.

Solution :

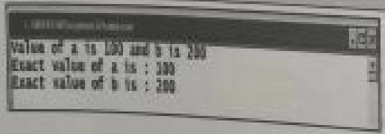
Program for nested If else

```
usingSystem;
namespaceDecisionMaking
{
classProgram
{
staticvoidMain(string[] args)
{
/* local variable definition */
int a =200;
int b =500;
if(a = =200)
{
if(b = =500)
{
Console.WriteLine("Value of a is 200 and b is 500");
}
}
Console.WriteLine("Exact value of a is : {0}", a);
Console.WriteLine("Exact value of b is : {0}", b);
Console.ReadLine();
}
}
}
```

A and b are local variables. Variables defined inside a function are called as local variables

If a equal to 100 is true then check next condition b equal to 500

Output



```
Value of a is 100 and b is 200
Exact value of a is : 100
Exact value of b is : 200
```

→ 1.2.7(E) switch Statement

Q. Explain working of switch statement with example. (4 Marks)

- The switch case is multi way decision making statement which helps to select a single option from multiple giv options.

☞ Syntax

```
switch(expression)
{
  case constant_expression 1:
```

```
  Statement 1;
  break;

  case constant_expression 2:
  Statement 2;
  break;
  ---------------
  case constant_expression m:
  Statement n;
  break;

  default:
  Statement m ;
}
```

☞ Expression

It is usually name of a variable value of which we want to check, or an expression.

☞ Constant expression

- These are the constant values with which the value of expression is compared. The statements in the case are executed whose value matches with the expression.
- If none of the constant expression is matched with the expression, then the statements written in default are executed.

Program 1.2.6

Write a program of switch statement.

Solution :

Program for switch-case statement

```
usingSystem;
namespaceDecisionMaking
{
  classProgram
  {
        staticvoidMain(string[] args)
     {
        /* local variable definition */

        char grade ='B';

        switch(grade)
        {
          case 'A':
          Console.WriteLine("Excellent!");
          break;
```

Passes grade to switch block to match it with different cases

Print "Excellent" if grade matched with A

```
case 'B':
case 'C':
Console.WriteLine("Well done");
break;
case 'D':
Console.WriteLine("Pass");
break;
case 'F':
Console.WriteLine("Better try again");
break;
default:
Console.WriteLine("Invalid grade");
break;
}
Console.WriteLine("Your grade is {0}", grade);
Console.ReadLine();
   }
  }
}
```

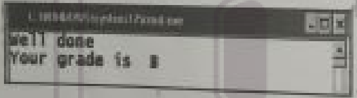
Print " well done" if grade matchead with B or C

Print "you passed" if grade matched with D

Print "better try again" if grade matched with F

If grade does not match with any case value then print "Invalid grade"

Output



```
well done
Your grade is B
```

→ 1.2.7(F) Nested switch

Q. Explain working of nested switch statement with example. **(4 Marks)**

- It is possible to have a switch as part of another switch. Even if the case values of the inner and outer switch are sa no conflicts will arise.

☞ Syntax

```
switch(expression)
{
  case constant_expression 1:
  Statement 1;
   switch(expression)
   {
        case constant_expression 1:
        Statement 1;
       break;
       case constant_expression 2:
       Statement 2;
       break;
       ___________
       case constant_expression n:
```

```
        Statement n;
        break;
        default:
        Statement m ;
  }
  break;
  case constant_expression 2:
  Statement 2;
  break;
  ------------
  case constant_expression n:
  Statement n;
  break;

  default:
  Statement m ;
}
```

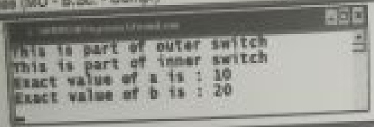
Program 1.2.7 : Write a program of nested switch case.

Solution : Program of nested switch case

```
usingSystem;
namespaceSwitchDemo
{
  classNestedSwitch
  {
    Static void Main(string[] args)
    {
      int a = 10;
      int b = 20;
      switch(a)
      {
      case 10:
      Console.WriteLine
      ("This is part of outer switch ");
      switch(b)
      {
         Case 20:
         Console.WriteLine("This is part of inner switch ");
         break;
         }
         break;
      }
      Console.WriteLine("Exact value of a is : {0}", a);
      Console.WriteLine("Exact value of b is : {0}", b);
      Console.ReadLine();
    }
  }
}
```

Checks; a is matched with 10,if yes then checks b is matched with 20. If both conditions are true then executes statements in those cases.

Output



```
This is part of outer switch
This is part of inner switch
Exact value of a is : 10
Exact value of b is : 20
```

Syllabus Topic : Loops

1.2.8 Loops

Q. What is use of loop and list various looping statment? (2 Ma

- Loops are used to execute specific task repeatedly in our program.
- Rather than writing the code again and again we can use the concept of loop. There are various situations when we may want to execute specific task multiple times.
- For example student mark sheet. Here we want to accept details from student and want to generate the marks sheet. This task is obviously repeated for number of students. In such situations we use the loops.
- Following are the looping statement in c#

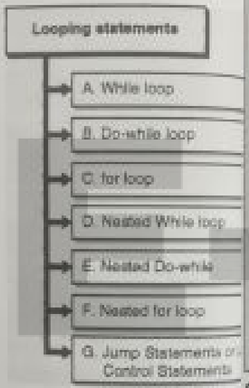


Fig. C1.6 : Looping statem

1.2.8(A) while Loop

Q. Explain working of while loop with example. (4 Marks)

- While loop is used when we want to execute the set of statements repeatedly until the given condition is satisfying.
- While loop is considered as an entry controlled loop. That means the condition is given at the beginning of loop. if the given condition does not satisfy, the loop statements never get executed.
- If the condition is satisfied, then loop statements are repeatedly executed until the condition is satisfying. O condition becomes false, the control exits the loop.

Syntax

```
while(Condition 1)
{
   Statement 1;
}
```

Program 1.2.8

Write a program to check given number is Armstrong or not.

Solution :

Program to checking givne number is Armstrong or not

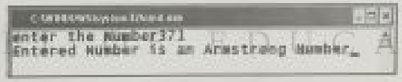
```
using System;
namespace ArmstrongDemo
{
  class ArmstrongProgram
  {
    static void Main(string[] args)
```

```
    {
      int num , sum = 0, rem = 0, temp;
      Console.WriteLine("Enter Number");
      num = Convert.ToInt32(Console.ReadLine());
      temp = num;
      while (temp != 0)
{
        rem = temp % 10; //find the reminder
        sum = sum + (rem * rem * rem);
        temp = temp / 10;
      }
    if (num == sum)
   {
Console.WriteLine("Entered Number is an Armstrong Number");
   }
      else
     {
        Console.WriteLine("Entered Number is not an
Armstrong Number");
       }
    }
  }
}
```

Until the temp becomes 0 it will repeat the loop statements To get digits from the temp we calculate the remainder by dividing the temp with 10

If given number is same as addition of cubes of digits of the number then it is Armstrong number

Output



```
enter the Number371
Entered Number is an Armstrong Number
```

Program 1.2.9

Write a program to accept a number from user and print factorial of it.

e.g. Factorial of 5 is calculated as

= 5*4*3*2*1 = 120

Solution : Program to accept a number from uses and print factorial of it

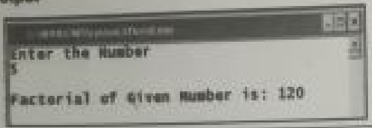
```
usingSystem;
namespace factorial
{
   class Program
   {
      staticvoid Main(string[] args)
      {
         int i, num, fact=1;
```

Declares num and fact ,i and initializes fact to 1

```
         Console.WriteLine("Enter the Number");
         num=int.Parse(Console.ReadLine());
         while( num>0)
         {
            fact = fact *num;
            num--;
         }
         Console.WriteLine("\nFactorial of Given Number
is: "+fact);
         Console.ReadLine();
      }
   }
}
```

Accepts n from user

Output



```
Enter the Number
5

Factorial of Given Number is: 120
```

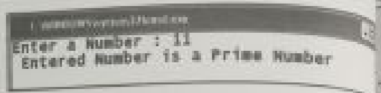
Program 1.2.10

Write a program to find whether the entered number is prime or not. (Prime number is the one which is divisible by 1 and itself only)

Solution : Program to finding entered number is prime or not

```
using System;
namespace example
{
  class prime
  {
    public static void Main()
    {
      Console.Write("Enter a Number : ");
      int num;
      num = Convert.ToInt32(Console.ReadLine());
      int k, i = 1;
      k = 0;
      while (i <= num / 2)
      {
        if (num % i == 0)
        {
          k++;
        }
        i++;
      }
      if (k == 2)
      {
        Console.WriteLine("Entered Number is not a
Prime Number");
      }
      else
      {
        Console.WriteLine(" Entered Number is a Prime
Number");
      }
      Console.ReadLine();
        }
  }
}
```

Output



```
Enter a Number : 11
 Entered Number is a Prime Number
```

→ 1.2.8(B) do-while Loop

Q. Explain working of do-while loop with examp (4 M

- The C# do-while loop is used to iterate a part program several times.
- If the number of iteration is not fixed then use d loop.
- Remember that the conditional expression app the end of the loop, so the statement(s) in th execute once before the condition is tested. Hen called as exit controlled loop.
- If the condition is true, the flow of control jum do, and the statement(s) in the loop execute again process repeats until the given condition become

☞ Syntax

```
do
{
  Statement 1;
} while(condition 1);
```

Program 1.2.11

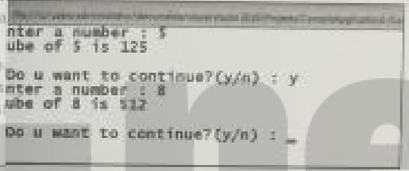
Write a program in C# to accept a number from and print it's cube. Ask user for continuity, if user yes, repeat the process.

Solution : Program to print cube of given num

```
using System;
namespace RepeatDemo
{
public class DoWhileExample
  {
        public static void Main(string[] args)
    {
          int num,cube;
          char ch;

          do
          {
            Console.Write("Enter a number : ");
            num =int.Parse(Console.ReadLine());
```

```
            cube=num*num*num;
            Console.WriteLine("Cube of "+num + " is
"+cube);
            Console.Write("\n Do u want to
continue?(y/n) : ");
            ch=char.Parse(Console.ReadLine());
          }while(ch=='y');
      }
  }
}
```

tput



```
nter a number : 5
ube of 5 is 125

Do u want to continue?(y/n) : y
nter a number : 8
ube of 8 is 512

Do u want to continue?(y/n) : _
```

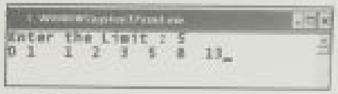
ogram 1.2.12

rite a program to print Fibonacci series.

lution : Program to print Fibonacci series

```
usingSystem;
namespace fibonaci
{
  class Program
  {
      staticvoid Main(string[] args)
      {
          int i=0, count, f1 =0, f2 =1, f3 =0;
          Console.Write("Enter the Limit : ");
          Count =int.Parse(Console.ReadLine());
          Console.Write(f1 + " ");
          Console.Write(f2+" ");
          do
          {
              f3 = f1 + f2;
              Console.Write(" "+f3);
              f1 = f2;
              f2 = f3;
              i++;
          } while(i <= count);
          Console.ReadLine();
      }
  }
}
```

Output



```
Enter the Limit : 5
0 1  1  2  3  5  8  13_
```

Program 1.2.13

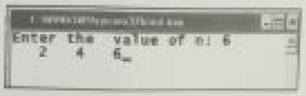
Write a program to display all even numbers from 1 to N.

Solution :

Program to display all even numbers from 1 to N.

```
usingSystem;
namespace EvenNumber
{
  class EvenProgram
  {
      staticvoid Main(string[] args)
      {
          int i=1,n;
          Console.Write("Enter the value of n:");
          n=int.Parse(Console.ReadLine());
          do
          {
              if(i%2==0)
              Console.Write(" "+i);
              i++;
          }while(i<=n);
      }
  }
}
```

Output



```
Enter the  value of n: 6
  2   4   6_
```

→ 1.2.8(C) for Loop

Q. Explain working of for loop with example (4 Marks)

- The C# for loop is used to iterate a part of the program number of times. If the number of iterations is fixed, it is recommended to use for loop instead of while or do-while loop.
- The C# for loop is same as C or C++. We can initialize variable, check condition and increment or decrement value.

Syntax

```
for(initialization; condition; increment/decrement)
{
    //code to be executed
}
```

Program 1.2.14

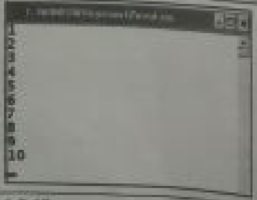
Write a program to print 1 to 10 numbers using for loop.

Solution :

Program to print 1 to 10 numbers using for loop.

```
usingSystem;
namespaceLoops
{
    classProgram
    {
        staticvoidMain(string[] args)
        {
            for(int a = 1; a < = 10; a ++)
            {
                Console.WriteLine(a);
            }
            Console.ReadLine();
        }
    }
}
```

Output



```
1
2
3
4
5
6
7
8
9
10
```

Program 1.2.15

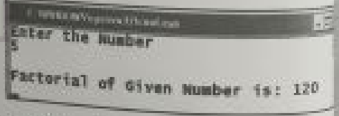
Write a program to accept a number from user and print factorial of it.

Solution :

Program to accept a number from user an factorial of it

```
usingSystem;
namespace factorial
{
    class Program
    {
        staticvoid Main(string[] args)
        {
            int i, number, fact;
            Console.WriteLine("Enter the Numbe
             number =int.Parse(Console.ReadLi
            fact = number;
            for(i = number -1; i > =1; i--)
            {
                fact = fact * i;
            }
            Console.WriteLine("\nFactorial of Gi
Number is: "+fact);
            Console.ReadLine();
        }
    }
}
```

Output



```
Enter the Number
5

Factorial of Given Number is: 120
```

→ 1.2.8(D) Nested while Loop

> **Q.** Explain working of nested while loop with ex (4

- We can write one while loop inside another wh Nested while loop in c# is same as nested whi C++.

Syntax

```
while (condition/expression)
{
    while (condition/expression)
    {
        Statement  ←── Body of inner for loop
    }
    Statement  ←── Body of outer while loop
}
```

rogram 1.2.16

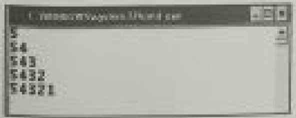
Vrite a C# program to print triangle pattern of first five atural numbers using nested while loop.

olution :

rogram to print triangle pattern of first five atural numbers using nested while loop.

```
using System;
namespace loop
{
    class Program
    {
        static void Main(string[] args)
        {
            int i = 5;
            while (i > = 1)
            {
                int j = 5;
                while (j >= i)
                {
                    Console.Write(j);
                    j--;
                }
                i--;
                Console.WriteLine();
            }
            Console.Read();
        }
    }
}
```

Output



```
5
54
543
5432
54321
```

→ 1.2.8(E) Nested do-while

> **Q.** Explain working of nested do-while loop with example. **(4 Marks)**

- We can write one do- while loop inside another while loop. Nested do while loop in c# is same as nested do while in C or C++.

Syntax

```
do
{
  do
  {
    statements;  ←── Body of inner do-while loop
  }while( condition/expression);
statements;  ←── Body of outer do-while loop
}while( condition );
```

Program 1.2.17

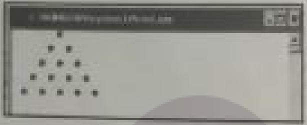
Write a C# program to print triangle pattern of character '*' using nested do while loop.

Solution :

Program to print triangle pattern of character '*' using nested do while loop

```
using System;
namespace loop
{
    class Program
    {
        static void Main(string[] args)
        {
            int i = 5;
            do
            {
                int space = i;
                do
                {
                    Console.Write(' ');
                    space--;
                }  while (space >= 1);
                int j = 5;
                do
            {
                Console.Write("* ");
                j--;
            }  while(j >= i);
            Console.WriteLine();
            i--;
        } while(i >= 1);
        Console.ReadLine();
    }
}
```

Output



```
    *
   * *
  * * *
 * * * *
* * * * *
```

1.2.8(F) Nested for Loop

Q. Explain working of Nested for loop with example. **(4 Marks)**

- We can write one for loop inside another for loop. Nested for loop in c# is same as nested while in C or C++.

Syntax

```
for ( init; condition; increment )
{
    for ( init; condition; increment )
    {
            statement(s);
    }
    statement(s);
}
```

Program 1.2.18

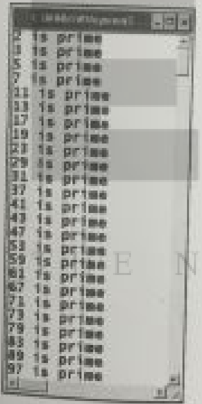
Write a program to a nested for loop to find the prime numbers from 2 to 100.

Solution :

Program to find prime numbers from 2 to 100 using nested for loop

```
usingSystem;
namespaceLoops
{
    classProgram
    {
        staticvoidMain(String[] args)
        {
            /* local variable definition */
            int p, q;
            for(p =2; p <100; p++)
            {
                for(q =2; q <=(p / q); p++)
                if((p% q)==0)break;// if (fa
found, not prime
                if(q >(p / q))
                Console.WriteLine("{0} is pri
            }
            Console.ReadLine();
        }
    }
}
```

Output



```
2 is prime
3 is prime
5 is prime
7 is prime
11 is prime
13 is prime
17 is prime
19 is prime
23 is prime
29 is prime
31 is prime
37 is prime
41 is prime
43 is prime
47 is prime
53 is prime
59 is prime
61 is prime
67 is prime
71 is prime
73 is prime
79 is prime
83 is prime
89 is prime
97 is prime
```

1.2.8(G) Jump Statements or Control Statements

Q. Explain working of loop control statements with example. (4

- Jump statements or Loop control statements execution from its normal sequence i.e Jump st is used to shift the control from one location to the program without checking any condition.
- When execution ends, all automatic objects created in that program are destroyed.
- C# provides the following loop control statements

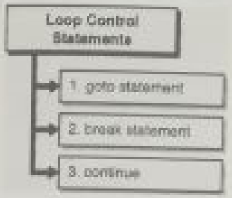


Fig. C1.7 : Loop Control Statements

1) goto statement

- The C# goto statement is also called as jump statement.
- It is used to move control from one part of program to the other part of the program.
- It transfer control without checking any condition.
- The goto needs a label to recognize the place where to transfer the execution control.
- A label is any valid identifier in C# and must be followed by a colon.
- The label is placed immediately before the statement where the control is to be transferred. The goto statement can move the execution control to any block of code in a program.

Syntax

```
          Statement 1;
          goto label 1
          ....
          Statement 4;
label 1 : Statement 5;
          Statement 6;
```

Program 1.2.19

Write a program to demonstrate use of goto statement.

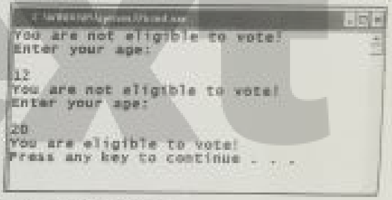
Solution :

Program to demonstrate use of goto statement

```
using System;
namespace JumpStatements
{
public class GotoExample1
    {
    public static void Main(string[] args)
    {
    noteligible:
        Console.WriteLine("You are not eligible to vote!");

    Console.WriteLine("Enter your age:\n");
    int age = Convert.ToInt32(Console.ReadLine());
    if (age < 18)
  {
        goto noteligible;
    }
    else
    {
        Console.WriteLine("You are eligible to vote!");
    }
   }
  }
}
```

If condition is true then control is transfer to label noteligible ,otherwise execute print the message 'you are eligible for vote'

Output



```
You are not eligible to vote!
Enter your age:

12
You are not eligible to vote!
Enter your age:

20
You are eligible to vote!
Press any key to continue . . .
```

2) break statement

- The **break** statement in C# has following two uses:
  1) As soon as the **break** statement is used inside a loop, the loop is instantly terminated and program control transfer to the next statement after the loop.
  2) It can be used to end the execution of a case in the **switch** statement.
- It breaks the current flow of the program at the given condition.
- In the nested loop, the break statement will be used to stop the execution of the innermost loop and begin execution of code after that block.

Syntax

```
break;
```

Program 1.2.20

Write a program to demonstrate use of break statement.

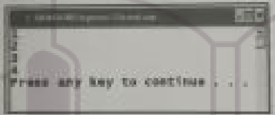
...ution :

...rogram to demonstrate use of break statement

```
using System;
namespace JumpStatements
{
public class BreakExample
  {
    public static void Main(string[] args)
    {
      for (int i = 1; i <= 10; i++)
      {
        if (i == 5)
        {
          break;
        }
        Console.WriteLine(i);
      }
    }
  }
}
```

If i is equal to 5 then Exit from loop

Output



```
1
2
3
4
Press any key to continue . . .
```

- Use C# Break Statement in nested Loop
- Break statement in c# is used to stop execution of loop.

Program 1.2.21

Write a program to demonstrate use of C# Break statement in nested Loop.

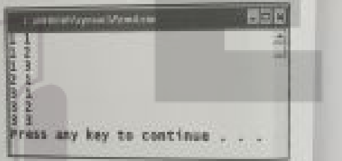
Solution :

Program to demonstrate use C# Break statement in nested Loop

```
using System;
namespace JumpStatements
{
public class BreakExample
  {
          public static void Main(string[] args)
      {
          for(int i=1;i<=3;i++)
          {
              for(int j=1;j<=3;j++)
              {
                  if(i==2&&j==2)
                  {
                  break;
                  }
Console.WriteLine(i+" "+j);
              }
          }
      }
  }
}
```

If i equal to 2 and j equal to 2 then Exit from inner for loop and execution control transfer to outer for loop. Increment the value of i

Output



```
1 1
1 2
1 3
2 1
3 1
3 2
3 3
Press any key to continue . . .
```

→ 2) C# Continue Statement

- The continue statement set the control at the beginning of loop for particular condition.
- That means for the given condition, the statements below the continue statement are skipped and then execute same block of code from beginning.
- We can use continue statement with any looping statement.

☞ Syntax

```
jump-statement;
continue;
```

Program 1.2.22

Write a program to demonstrate continue statement in C#.

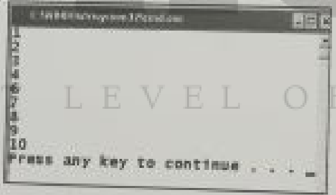
Solution :

Program to demonstrate continue statement in C#.

```
using System;
namespace JumpStatements
{
public class ContinueExample
  {
        public static void Main(string[] args)
        {
        for(int i=1;i<=10;i++)
{
        if(i==5)
{
        continue;
      }
      Console.WriteLine(i);
    }
  }
 }
}
```

When value of i is equal to 5 then console.writeLine() statement is skipped and execution control transfer to beginning of for loop

Output



```
1
2
3
4
6
7
8
9
10
Press any key to continue . . . _
```

Syllabus Topic : Methods

1.2.9 Methods

> **Q.** What is function? How to declare, define and call the function? **(4 Marks)**

- Method is a block of statements used to execute a task repeatedly.
- Every C# program has at least one method named main().
- Defining method in C#

☞ **Syntax**

```
<access-specifier><return-type><FunctionName>(<parameter list>)
{
   // method body
}
```

- Following are the various elements of a method :

1. **Access specifier** : Decides the scope of method.
2. **Return type** : A method may or may not return a value to calling function. Called function can send single value at a time to calling function. The return type is the data type of that value returned by the method. Void is used as return type, if the method is not returning any value.
3. **Method name:** Method name is a unique identifier which is case sensitive. It follows all variable naming rules i.e. start with alphabet, underscore, not start with number, method name should not keyword etc.
4. **Parameter list** : The parameter is used to give input to method and receive data from a method. Method may contain parameters or not. We can pass more than one parameters to function. The parameter list contains the data type and names of the parameters of the method.
5. **Method body** : Method body contains the group of statements needed to execute for performing a particular task.

☞ **Example**

Acess- specifier return-type function-name parameter-list

```
public   int   add (int a , int b)
{
   int c;
c=a+b;
Return c;
}
```

Return value c of integer type

1.2.9(A) Types of Function

- There are two types of functions in C# code :

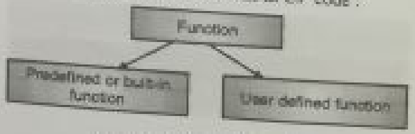


Fig. 1.2.3 : Types of functions

1. **Predefined or built in functions**

These are the functions which are already defined by C# such as Console. ReadLine() for reading, Console.WriteLine() for writing etc.

- C# has rich set of built-in functions.

Program 1.2.23 : Write a program to demonstrate built-in function.

Solution :

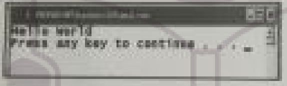
Program to demonstrate built-in function

```
usingSystem;
namespaceHelloWorldDemo
{
  Class FirstProgram
  {
    static void main(string[] args)
    {
      Console.WriteLine("Hello World");
    }
  }
}
```

Use namespace in your program which support predefined functions

Use built-in function Console.WriteLine() to display the output on screen

Output



```
Hello World
Press any key to continue . . .
```

2. **User-defined functions**

These are the functions which are created by the C # programmer to perform specific task. Programmer can use these functions many times.

Syntax

```
return_type function_name(data_type parameter...)
{
  Code to be executed
}
```

Program 1.2.24

Write a program to demonstrate user defined function.

Solution :

Program to demonstrate user defined function

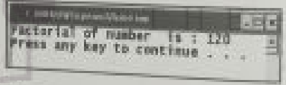
```
usingSystem;
namespaceCalculatorDemo
{
```

```
classFactorialExample
{
  publicint factorial(int n)
  {
    int result;
    if(n==1)
    {
      return1;
    }
    else
    {
      result = factorial(n-1)* n;
      return result;
    }
  }
  staticvoidMain(string[] args)
  {
    FactorialExample f=newFactorialExample();

    int fact=f. factorial(5);

    Console.WriteLine("Factorial of number is :
{0}", fact);
  }
 }
}
```

User defined function factorial () passing number whose factorial we want to calculate

Calling factorial() method

Output

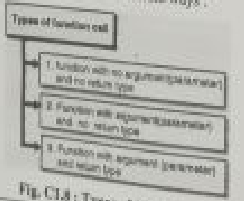


```
Factorial of number is : 120
Press any key to continue . . .
```

1.2.9(B) Type of Function Call

Q. Write note on types of function call. (4 Marks)

- Function can be called in different ways :



Types of function call

1. Function with no argument(parameter) and no return type

2. Function with argument(parameter) and no return type

3. Function with argument (parameter) and return type

Fig. C1.8 : Types of function call

1) Function with no argument(parameter) and no return type

- In this type of function call,there is no information transfer between calling function and called function.
- When function has no argument, it does not receive any information from calling function.
- Similarly, when it does not return any value ,calling function does not receive any information from called function.

Program 1.2.25

Write a program of calling function with no argument and no return value.

Solution :

Program of calling function with no argument and no return value

```
using System;
namespace FunctionDemo
{
  class FunctionCall

  {
    public void Display()
    {
Console.WriteLine("This is non parameterized function");

  }

static void Main(string[] args)
  {
    FunctionCall fun= new FunctionCall ();
    fun.Display();
  }
 }
}
```

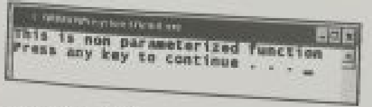
User defined function Display() without return type and does not have any parameter

There is no return statment

Create object of class

Function call without passing any parameter

Output



```
This is non parameterized function
Press any key to continue . . .
```

2) Function with argument(parameter) and no return type

- In this type of function call, there is information transfer between calling function and called function.

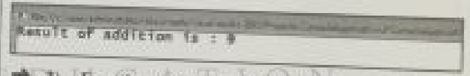
Program 1.2.26

Write a program of calling function with argument and no return value.

Solution : Program demonstrating function call

```
using System;
namespace FunctionDemo
{
  class FunctionCall
  {
    public void add(int n1, int n2)
    {
      int result;
      result = n1 + n2;
      Console.WriteLine("Result of addition is : " +
result);
    }
    static void Main(string[] args)
    {
      FunctionCall p = new FunctionCall();

      p.add(3, 6);

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

Output



```
Result of addition is : 9
```

3) Function with argument(parameter) and return type

The call function is executed until return statement is encountered. The return value is pass back to function call.

Program 1.2.27

Write a program to demonstrate working of return key word.

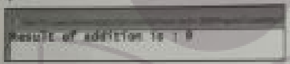
Solution :

Program to demonstrate working of return keyword

```
using System;
namespace returnstatementDemo
{
  class Program
  {
```

```
public int addition(int n1, int n2)
{
    return n1 + n2;  ← Return addition of n1 and n2
}
static void Main(string[] args)
{
    Program p = new Program();
    int result;

    result = p.add(3, 6);  ← Returned value is assigned to result

Console.WriteLine( "result of addition is :"+result);

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

Output



```
Result of addition is : 9
```

1.2.9(C) Methods of Passing Parameter

Q. Which methods are used to pass parameter to method? **(4 Marks)**

There are three ways to pass parameter to method

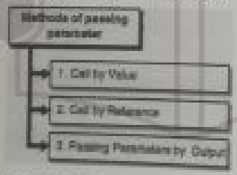


Fig. C1.9 : Methods of passing parameter

➜ 1) Call by Value

- In call by value, values of parameters are passed to the function at the time of function call.
- Formal parameters are replaced by actual parameter at the time of function call.
- In this method, we pass values of parameters so the changes in formal parameters do not reflect in actual parameters.
- In previous section, the values are passed by value for the parameters.

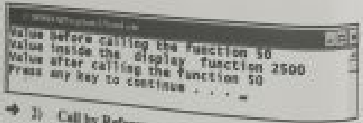
Program 1.2.28

Write a program to demonstrate call by value.

Solution : Program to demonstrate call by value

```
using System;
namespace FunctionDemo
{
  class CallByValueDemo
  {
    public void Display(int a)
    {
      a= a*a;
      Console.WriteLine("Value inside the display
function " +a);
    }
    static void Main(string[] args)
    {
      int a = 50;
      CallByValueDemo c1 = new CallByValueDemo
();
      Console.WriteLine("Value before calling the functio
n" +a);

      C1.Display(a);  ← Calling function by passing value

Console.WriteLine("Value after calling the function " + a)
;
    }
  }
}
```

Output



```
Value before calling the function 50
Value inside the display function 2500
Value after calling the function 50
Press any key to continue . . . _
```

➜ 2) Call by Reference

- C# provides a ref keyword to pass address of parameter to method.
- It passes reference(address) of argument to the function rather than copy of original value.

- The changes in formal parameters also reflect in actual parameters.

Program 1.2.29

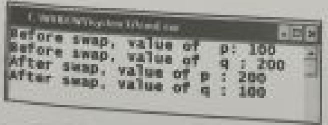
Write to a program to demonstrate call by reference.

Solution :

Program to demonstrate call by reference

```
usingSystem;
namespace FunctionDemo
{
class SwapDemo
{
void swap(ref int a ,ref int b)
{
    int temp;
      temp = a;  ← save the value of a

      a = b;  ← Put value in b in a

      b = temp;  ← Put value of b in temp
}
static void Main(string[] args)
{
SwapDemo n =new SwapDemo();
int p=100;  }
int q=200;  } ← Local variable declaration

Console.WriteLine("Before swap, value of p: {0}", p);  }
Console.WriteLine("Before swap, value of q : {0}", q);  } ← Value before swapping

n.swap(ref p, ref q);  ← calling a function to swap the values

  Console.WriteLine("After swap, value of p : {0}", p);
  Console.WriteLine("After swap, value of q : {0}", q);
Console.ReadLine();
```

Output



```
Before swap, value of  p: 100
Before swap, value of  q : 200
After swap, value of p : 200
After swap, value of q : 100
```

➜ 3) Passing Parameters by Output

- A return statement can be used to return only one value from a called function to calling function.
- But using output parameters, you can return multiple values from a called function to calling function.
- Output parameters are similar to call by reference (use ref keyword),but difference is that they transfer data out of the method rather not accept in method.

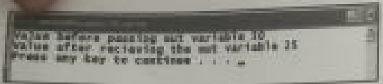
Program 1.2.30

Write a program to demonstrate passing parameter by output.

Solution :

Program to demonstrate passing parameter by output

```
using System;
namespace FunctionDemo
{
  class OutParameterDemo                Declare out parameter
  {
      public void Display(out int v)  ←
    {
      int square = 5;
      v = square;
      v*=v;  ← v = v*v
    }

    static void Main(string[] args)
    {                                  Creating object of class
      int v = 50;
OutParameterDemo o= new OutParameterDemo ();  ←

 Console.WriteLine("Value before passing out variable " +
v);
      o.Show(out v);  ← Passing out parameter

      Console.WriteLine("Value after recieving the out va
riable " + v);
    }
  }
}
```



```
Value before passing out variable 10
Value after reciesving the out variable 25
Press any key to continue . . .
```

1.2.9(D) Scope of variables

Q. Write note on scope of variable (4 Marks)

- Scope is the area of program in which the variable is accessible.
- There are two areas where variables can be declared in C# programming language.

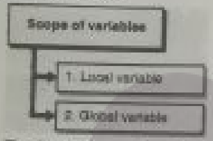


Fig. C1.10 : Scope of variables

1) Local Variables

- Variables that are defined inside a method are called local variables.
- They can be used only inside that function or block of code in which they are defined.
- There is no scope of local variable outside that method.
- Local variables are deleted once the working of function that created them is completed.
- They are recreated every time when that function is called.

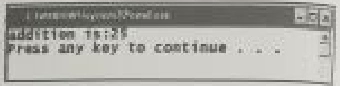
Program 1.2.31

Write a program to demonstrate local variable scope.

Solution : Program to demonstrate local variable scope

```
using System;
namespace FunctionDemo
{
  class LocalDemo
  {
    void add(int a,int b)
    {
      int c;   ← Local variable declaration
      c=a+b;
      Console.WriteLine("addition is:"+c);   ← Local variable can be accessed only inside function
    }
    static void Main(String[] args)
    {
      LocalDemo local=new LocalDemo();
      local.add(10,15);
    }
  }
}
```

Output



```
addition is:25
Press any key to continue . . .
```

2) Global Variables

- Global variables are defined outside the methods, at the top of the program.
- Global variables maintain their values throughout execution of program.
- Global variable is available throughout the program anywhere. They can be accessed inside any function defined in the program.

Program 1.2.32

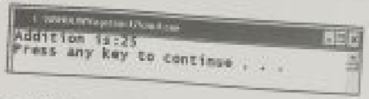
Write a program to demonstrate global variable scope.

Solution :

Program to demonstrate global variable scope

```
using System;
namespace FunctionDemo
{
  class LocalDemo
  {
    int c;   ← Global variable declaration
    void add(int a,int b)
    {
          c=a+b;
    }
  static void main(String[] args)
  {
      LocalDemo local=new LocalDemo();
      local.add(10,15);
      Console.ReadLine("Addition is:"+ local.c);   ← Access value of global variable c
    }
  }
}
```

Output



```
Addition is:25
Press any key to continue . . .
```

1.2.9(E) Recursion

Q. What is recursion ? Explain with example. (4 Marks)

- Method can call itself in its definition is called as **recursion**.
- To do same task repetitively, recursion is used.

Program 1.2.33

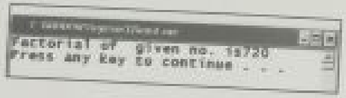
Write a program to find factorial of number using recursion.

Solution :

Program to find factorial of number using recursion

```
usingSystem;
namespaceFunctionDemo
{
  class FactorialDemo
  {
  publicint factorial(int num)
  {
      int result;   ← Local variable declaration
      if(num ==1)
      {
          return1;
      }
      else
      {
          result = factorial(num -1)* num;   ← Recursive call to function factorial()
          return result;
      }
  }
  staticvoidMain(string[] args)
  {
        FactorialDemo f=newFactorialDemo();
        Console.WriteLine("Factorial of 6 is : {0}",
n.factorial(6));
    }
  }
}
```

Output



```
Factorial of given no. is720
Press any key to continue . . .
```

1.2.10 C# Object

Q. Write note on object in C#. (4 Marks)

- In C#, Object is a real world thing such as car, person, chair etc.
- That means, object is an entity that has state and behavior. Here, state means data and behavior means functionality.
- Object is a runtime entity that means it is created at runtime.
- Object is an instance of a class.
- All the members of the class can be accessed through object.
- To access the members of class , dot (.) operator is used.
- The dot operator links the name of an object with the name of a member.

Syntax

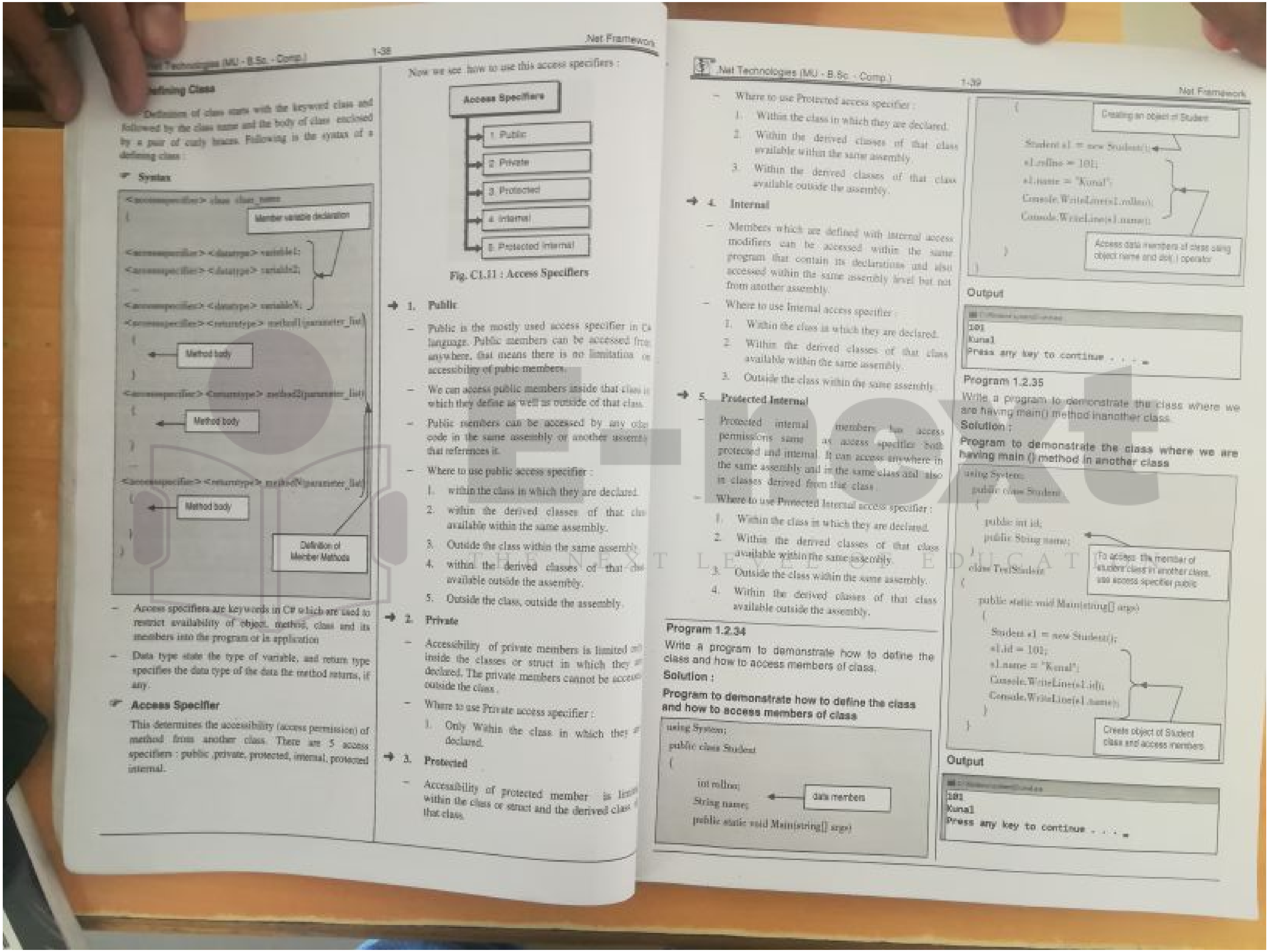
```
<Class-name> <Object-name> =new <Class-name>
e.g.
Student s1 = new Student();   ← creating an object of Student
```

Syllabus Topic : Classes

1.2.10(A) Classes

Q. What is mean by class? How to define class? (4 Marks)

- Class is a collections of data members and methods.
- We can create one or more objects of same class to access class members.
- The methods and variables that are included in class are called as members of that class.



Defining Class

Definition of class starts with the keyword class and followed by the class name and the body of class enclosed by a pair of curly braces. Following is the syntax of a defining class :

☞ **Syntax**

```
<accessspecifier> class class_name
{

<accessspecifier> <datatype> variable1;
<accessspecifier> <datatype> variable2;
...
<accessspecifier> <datatype> variableN;
<accessspecifier> <returntype> method1(parameter_list)
{

}
<accessspecifier> <returntype> method2(parameter_list)
{

}
...
<accessspecifier> <returntype> methodN(parameter_list)
{

}
}
```

Member variable declaration

Method body

Method body

Method body

Definition of Member Methods

- Access specifiers are keywords in C# which are used to restrict availability of object, method, class and its members into the program or in application
- Data type state the type of variable, and return type specifies the data type of the data the method returns, if any.

☞ **Access Specifier**

This determines the accessibility (access permission) of method from another class. There are 5 access specifiers : public ,private, protected, internal, protected internal.

Now we see how to use this access specifiers :

Access Specifiers

1. Public

2. Private

3. Protected

4. Internal

5. Protected Internal

Fig. C1.11 : Access Specifiers

➜ 1. **Public**

- Public is the mostly used access specifier in C# language. Public members can be accessed from anywhere, that means there is no limitation on accessibility of public members.
- We can access public members inside that class in which they define as well as outside of that class.
- Public members can be accessed by any other code in the same assembly or another assembly that references it.
- Where to use public access specifier :
  1. within the class in which they are declared.
  2. within the derived classes of that class available within the same assembly.
  3. Outside the class within the same assembly.
  4. within the derived classes of that class available outside the assembly.
  5. Outside the class, outside the assembly.

➜ 2. **Private**

- Accessibility of private members is limited only inside the classes or struct in which they are declared. The private members cannot be accessed outside the class.
- Where to use Private access specifier :
  1. Only Within the class in which they are declared.

➜ 3. **Protected**

- Accessibility of protected member is limited within the class or struct and the derived class of that class.

- Where to use Protected access specifier :
  1. Within the class in which they are declared.
  2. Within the derived classes of that class available within the same assembly.
  3. Within the derived classes of that class available outside the assembly.

➜ 4. **Internal**

- Members which are defined with internal access modifiers can be accessed within the same program that contain its declarations and also accessed within the same assembly level but not from another assembly.
- Where to use Internal access specifier :
  1. Within the class in which they are declared.
  2. Within the derived classes of that class available within the same assembly.
  3. Outside the class within the same assembly.

➜ 5. **Protected Internal**

- Protected internal members has access permissions same as access specifier both protected and internal. It can access anywhere in the same assembly and in the same class and also in classes derived from that class.
- Where to use Protected Internal access specifier :
  1. Within the class in which they are declared.
  2. Within the derived classes of that class available within the same assembly.
  3. Outside the class within the same assembly.
  4. Within the derived classes of that class available outside the assembly.

Program 1.2.34

Write a program to demonstrate how to define the class and how to access members of class.

Solution :

Program to demonstrate how to define the class and how to access members of class

```
using System;
public class Student
{

    int rollno;
    String name;
    public static void Main(string[] args)
    {

        Student s1 = new Student();
        s1.rollno = 101;
        s1.name = "Kunal";
        Console.WriteLine(s1.rollno);
        Console.WriteLine(s1.name);

    }
}
```

data members

Creating an object of Student

Access data members of class using object name and dot (.) operator

Output

```
101
Kunal
Press any key to continue . . . _
```

Program 1.2.35

Write a program to demonstrate the class where we are having main() method inanother class.

Solution :

Program to demonstrate the class where we are having main () method in another class

```
using System;
  public class Student
  {

    public int id;
    public String name;
  }
  class TestStudent
  {

    public static void Main(string[] args)
    {

      Student s1 = new Student();
      s1.id = 101;
      s1.name = "Kunal";
      Console.WriteLine(s1.id);
      Console.WriteLine(s1.name);
    }
  }
```

To access the member of student class in another class, use access specifier public

Create object of Student class and access members

Output

```
101
Kunal
Press any key to continue . . . _
```

Syllabus Topic : Value Types and Reference Types

1.2.11 Value Type and Reference Type

It is explained in previous section 1.2.9(C) Methods of passing parameter.

Syllabus Topic : Namespaces and Assemblies

1.2.12 Namespaces and Assemblies

1.2.12(A) C# Namespaces

Q. Write note on namespaces. (4 Marks)

- Namespaces in C# are used to store multiple classes into single unit so that we can manage and reuse them whenever needed.
- In C# program, we use System.Console where System is the namespace and Console is the class.
- To access the classes from namespace, we need to use namespacename.classname.
- We can use **using** keyword to use specific namespace in our program.

☞ Defining Namespace

We can define namespace with the keyword **namespace** followed by the namespace name.

☞ Syntax

```
namespace namespace_name
{

}
```

Program 1.2.36

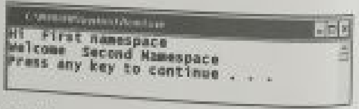
Write a program to demonstrate use of namespace.

Solution :

Program to demonstrate use of namespace

```
using First;
using Second;
namespace First          ←——  Namespace
{
  public class NamespaceDemo1
  {
        public void sayHi()
        {
              Console.WriteLine("Hi  First namespace");
        }
    }
 }
namespace Second
{
  public class NamespaceDemo2
  {
            public void sayWelcome()
            {
         Console.WriteLine("Welcome
Second Namespace");
            }
      }
}
public class TestNamespace
{
  public static void Main()
  {
        NamespaceDemo1
h1 = new NamespaceDemo1();

  NamespaceDemo2w1 = new NamespaceDemo2();
        h1.sayHi();
        w1.sayWelcome();
  }
}
```

Output



```
Hi  First namespace
Welcome  Second Namespace
Press any key to continue . . .
```

1.2.12(B) Nested Namespaces

- You can define one namespace inside another namespace in such a way that we define one if inside another if.

☞ Syntax

```
namespace namespace_name1
{     ←——  Code declaration
namespace namespace_name2
  {
     ←——  Code declaration
  }
}
```

- You can access members of nested namespace by using dot (.) operator.

Program 1.2.37

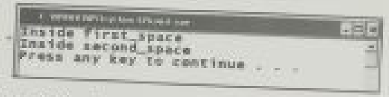
Write a program to show how to access members of nested namespace.

Solution :

Program to show how to access members of nested namespace

```
using System;
using FirstSpace;
using FirstSpace.SecondSpace;
namespace FirstSpace
{
  class NameSpaceExample1
  {
    public void Display()
    {
      Console.WriteLine("Inside first_space");
    }
  }
 namespace SecondSpace
 {
class NameSpaceExample2
   {
    public void Display()
    {
      Console.WriteLine("Inside second_space");
    }
   }
  }
}
class TestDemo
{
  static void Main(string[] args)
  {
      NameSpaceExample1 fe = new NameSpaceExample1();
        NameSpaceExample2 sc = new
NameSpaceExample2();
        fe.Display();
        sc.Display();
    }
  }
```

Output



```
Inside first_space
Inside second_space
Press any key to continue . . .
```

1.2.12(C) Assemblies

Q. Write note on assemblies. (4 Marks)

- When a .Net application is successfully compiled, assembly file is automatically generated.
- An assembly may be an executable file(.exe) or a Dynamic Link Library (DLL). At only first compilation this file is generated, and after that for each subsequent compilation it gets updated.
- The assembly generation is background process of an application.
- An Assembly contains IL (Intermediate Language) code. This IL is same as of the Java byte code.
- In the .Net language, assembly contains metadata. Features of every "type" is provided in the metadata.
- Assembly contains a special file called as Manifest. This manifest file contains the information regarding the current version of the assembly as well as other related information.
- .Net provides two types of assemblies. Single file and Multi file. A single file assembly holds all the necessary information like IL, Metadata, and Manifest in a single package. Most of the assemblies in .Net are of the type single file assemblies.
- Multi file assemblies are made up of various types of .Net binaries or modules and are usually created for larger applications.
- One of the assemblies contains the most important special manifest file and others assemblies may contain IL and Metadata instructions.

Syllabus Topic : Inheritance

1.2.13 Inheritance

Q. What is mean by inheritance? Write note on types of inheritance. (4 Marks)

- Inheritance is one of the characteristic of Object-oriented programming language.
- The mechanism of creating new class from existing class is called as inheritance.
- In inheritance, old class from which we derive new class is called as parent class, base class or super class.
- New class is called as child class, derived class or sub class.
- The derived class has its own attributes (variables) and behavior (functions) and it also access the variables and functions defined in its parent class.
- The thought behind inheritance is to implements the IS-A relationship.
- For example, dog IS-A animal. Here animal is parent class and dog is child class.

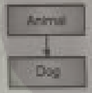


Fig. 1.2.4 : Inheritance Example

Advantages of Inheritance

1. It reduces code repetition.
2. It provides code reuse feature.
3. It reduces size of source code and improves code readability.
4. We can easily divide and manage large source code in parent and child class.
5. It helps to extend the code by overriding the base class functions in its child classes.

Types of Inheritance

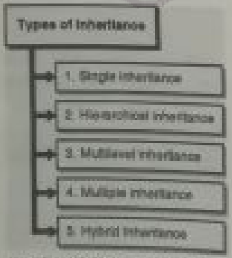


Fig. C1.12 : Types of Inheritance

1) Single inheritance

When one child class is derived from one parent class it is called as single inheritance.

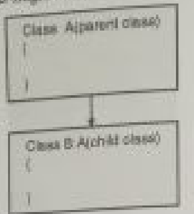


Fig. 1.2.4(a) : Example of single inheritance

Program 1.2.38

Write a program to demonstrate single level inheritance.

Solution :

Program to demonstrate single level inheritance

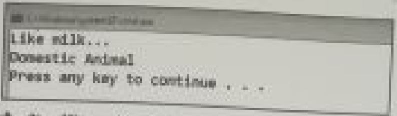
```
using System;
  public class Animal
  {
    public void like()
{
   Console.WriteLine("Like milk...");
}

  }
 public class Cat: Animal
 {
          public void type()
{
Console.WriteLine("Domestic Animal");
   }
  }
  class InheritanceDemo2
 {
     public static void Main(string[] args)
      {
        Cat c1 = new Cat();
        c1.like();
        c1.type();
      }
}
```

Parent class

Create child class Cat from parent class Anim

Call to function of parent class like() using object of child class and also to its own function

Output



```
Like milk...
Domestic Animal
Press any key to continue . . .
```

2) Hierarchical inheritance

- This is the type of inheritance in which there are multiple child classes created from one parent class. This type of inheritance is used when there is a requirement accessing one class features in multiple classes.
- In this type of inheritance, there is no single class which get properties of all the classes, hence we have to create objects for all the child classes.

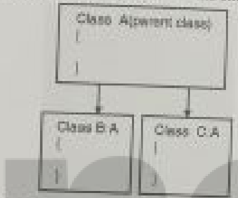


Fig. 1.2.4(b) : Example of Hierarchical inheritance

Program 1.2.39

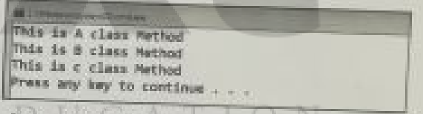
Write a program to demonstrate Hierarchical inheritance.

Solution :

Program to demonstrate Hierarchical inheritance.

```
using System;
using System.Text;

namespace ConsoleApplication52
{

  class A
  {
    public void msg()
    {
      Console.WriteLine("This is A class Method");
    }
  }
  class B : A
  {
    public void info()
    {
                  Console.WriteLine("This is B class
Method");
      }
   }
   class C : A
   {
     public void getinfo()
     {
                   Console.WriteLine("This is c class
Method");
     }
   }

  class D
  {
    public static void Main(String[] args)
    {
      B b1 = new B();
      C c1 = new C();
      b1.msg();      // can be called by both b1 and c1
      b1.info();
      c1.getinfo();
    }
  }
}
```

Output



```
This is A class Method
This is B class Method
This is c class Method
Press any key to continue . . .
```

3) Multilevel inheritance

- In this type of inheritance a class is inherited from another child class.

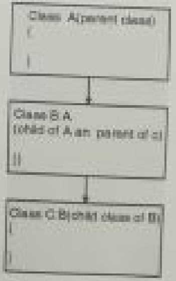


Fig. 1.2.4(c) : Example of multilevel inheritance

gram 1.2.40

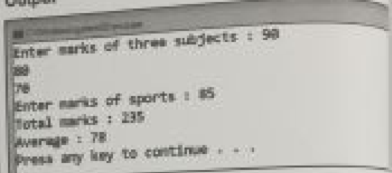
e a program to demonstate multilevel inheritance.

lution :

Program to demonstrate multilevel inheritance

```
using System;
using System.Text;

namespace ConsoleApplication52
{

  class student
  {
    public int hin, mar, eng, sports, tot, avg;
    public void getmarks()
    {
      Console.Write("Enter marks of three subjects : ");
      hin = Convert.ToInt32(Console.ReadLine());
      mar = Convert.ToInt32(Console.ReadLine());
      eng = Convert.ToInt32(Console.ReadLine());
    }
  }

  class sportsclass : student
  {
    public void getsports()
    {
      Console.Write("Enter marks of sports : ");
      hin = Convert.ToInt32(Console.ReadLine());
    }
  }

  class result : sportsclass
  {
    public void cal()
    {
      tot = hin + mar + eng + sports;
      avg = tot / 3;
      Console.WriteLine("Total marks : " + tot);
      Console.WriteLine("Average : " + avg);
    }
  }

  class D
  {
    public static void Main(String[] args)
    {
      result r = new result();
      r.getmarks();
      r.getsports();
      r.cal();
    }
  }
}
```

Output



```
Enter marks of three subjects : 90
80
70
Enter marks of sports : 85
Total marks : 235
Average : 78
Press any key to continue . . .
```

➜ 4) **Multiple inheritance**

- C# does not support multiple inheritances onc using only class. To overcome this problem can use interfaces.

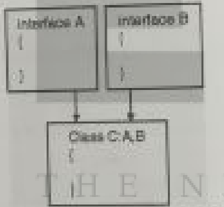


Fig. 1.2.4(d) : Eexample of multiple inheritance

☞ **Interface**

- Interface is just like a class but in which declarations of methods is allowed, we cannot any method.
- All the methods declared in an interface mu declared in the child classes.
- We cannot create an object of interface.

Program 1.2.41

Write a program to demonstrate multiple inher using interface.

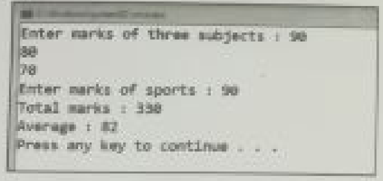
Solution :

Program to demonstrate multiple inheritance Interface

```
using System;
using System.Text;
```

```
namespace ConsoleApplication52
{
 interface student
 {
    void getmarks();
 }
 interface sportsinf
 {
    void getsports();
 }
class result : student, sportsinf
{
   public int hin, mar, eng, sports, tot, avg;
   public void getmarks()
   {
     Console.Write("Enter marks of three subjects : ");
     hin = Convert.ToInt32(Console.ReadLine());
     mar = Convert.ToInt32(Console.ReadLine());
     eng = Convert.ToInt32(Console.ReadLine());
   }
   public void getsports()
   {
     Console.Write("Enter marks of sports : ");
     sports = Convert.ToInt32(Console.ReadLine());
   }
   public void cal()
   {
     tot = hin + mar + eng + sports;
     avg = tot / 4;
     Console.WriteLine("Total marks : " + tot);
     Console.WriteLine("Average : " + avg);
   }
}

class D
{
  public static void Main(String[] args)
  {
    result r = new result();
    r.getmarks();
    r.getsports();
    r.cal();
  }
 }
}
```

Output



```
Enter marks of three subjects : 90
80
70
Enter marks of sports : 90
Total marks : 330
Average : 82
Press any key to continue . . .
```

➜ 5) **Hybrid Inheritance**

Is the combination of more than one types of inheritances.

Program 1.2.42

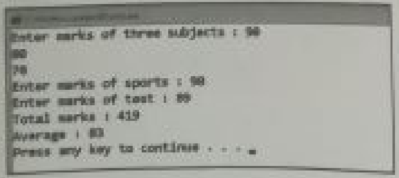
Write a program to demonstrate Hybrid inheritance.

Solution :

Program to demonstrate hybrid inheritance

```
using System;
using System.Text;

namespace ConsoleApplication52
{

  interface student
  {
    void getmarks();
  }

 interface sportsinf : student
 {
    void getsports();
 }
 interface testinf
 {
    void gettest();
 }
 class result : sportsinf,testinf
 {
   public int hin, mar, eng, sports, test, tot, avg;
   public void getmarks()
   {
     Console.Write("Enter marks of three subjects : ");
     hin = Convert.ToInt32(Console.ReadLine());
     mar = Convert.ToInt32(Console.ReadLine());
    avg = Convert.ToInt32(Console.ReadLine());
  }
  public void getsports()
  {
    Console.Write("Enter marks of sports : ");
    sports = Convert.ToInt32(Console.ReadLine());
  }
  public void gettest()
  {
    Console.Write("Enter marks of test : ");
    test = Convert.ToInt32(Console.ReadLine());
  }
  public void call()
  {
    tot = hin + mar + eng + sports + test;
    avg = tot / 5;
    Console.WriteLine("Total marks : " + tot);
    Console.WriteLine("Average : " + avg);
  }
}
class D
{
  public static void Main(String[] args)
  {
    result r = new result();
    r.getmarks();
    r.getsports();
    r.gettest();
    r.call();
  }
}
}
```

Output



```
Enter marks of three subjects : 90
80
70
Enter marks of sports : 90
Enter marks of test : 89
Total marks : 419
Average : 83
Press any key to continue . . .
```

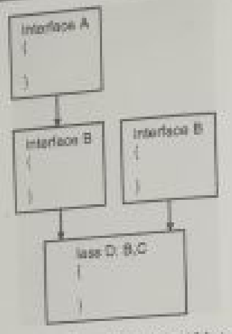


Fig. 1.2.4(e) : Example of hybrid interface

Syllabus Topic : Static Members

1.2.14 Static Members

Q. What is mean by static member? List the stat members in class. (2 Mark

- To create functions and variables of a cla static keyword can be used.
- When we create members of a class as static, then the is no matter how many objects of that class are creat There is only one copy of each static member which used by all the objects of that class.
- When the first object of class is created, all static d is initialized to zero if no, other initialization for th static data is present.
- We can declare following members of class as stat

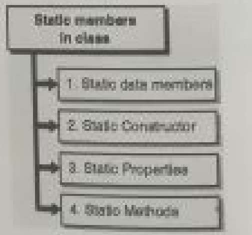


Fig. C1.13 : Static members in class

➜ 1) **Static Data Members**

- Class can have both static as well as non members .
- Static data members of class are also calle class members and non-static members of are called as instance members.

- Static members are loaded in memory at compile time while instance members of class are loaded in memory at execution time of program.
- 'this' keyword refers methods and variables of current class. We can't use **this** keyword with static members of class to access them.

```
class StaticExample

{
   public static int age;
   public static string name= "Kunal";
}
```

➜ 2) **Static Constructor**

- We cannot pass parameter to constructor which is created using static keyword.
- We cannot use any access modifier while defining static constructor.
- To initialize static data members of the class, Static Constructor is used.

➜ 3) **Static Properties**

- To get or set the values of static data members of class , Static properties are used.

Program 1.2.43

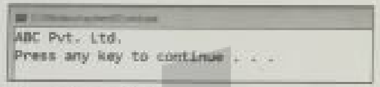
Write a program to demonstate working of static properties(functions) which contain get and set functions.

Solution :

Program to demonstate working of static properties (functions) which contain get and set functions.

```
using System;
namespace staticExample
{
  class StatDemo
  {
    static string company_name;
    //StaticProperty
    public static string _company_name
    {
      get
      {
        return company_name;
      }
      set
      {
        company_name = value;
      }
    }
    static void Main(string[] args)
    {
      StatDemo.company_name = "ABC Pvt. Ltd.";
      Console.WriteLine(StatDemo.company_name);
    }
  }
}
```

Output



```
ABC Pvt. Ltd.
Press any key to continue . . .
```

➜ 4) **Static Methods**

- While calling static method,we use class name and static method name only.
- No need to use object of class. Static methods use only static data members to perform calculation or processing.

Program 1.2.44

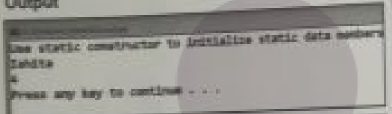
Write a program to demonstrate static constructor and static methods.

Solution :

Program to demonstrate static constructor and static methods

```
using System;
  namespace static_example
{
  class StaticDemo
  {
    public class test
    {
        static string tname;
        static int tage;
            static test()
        {
        Console.WriteLine("Use static constructor to
initialize static data members of class");
        tname= "Tehila";
        tage=4;
        }
        public static void display()
        {
            Console.WriteLine(name);
            Console.WriteLine(age);
        }
    }
    static void Main(string[]   args)
    {
        test.display();
        }
    }
}
```

Output



```
Use static constructor to initialize static data members
Zahida
[illegible]
Press any key to continue . . .
```

Syllabus Topic : Casting Objects

1.2.15 Casting Object

> **Q.** What is mean by casting object ? Explain types of casting object. **(4 Marks)**

- Casting object or Type conversion process is converting value from one data type to another type.
- Data type conversion is based on **type compatibility**.
- It is also called as Type Casting.
- In C#, type-casting has two forms :

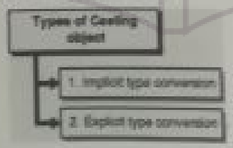


Fig. C1.14 : Types of casting object

➜ 1) **Implicit type conversion**

- In implicit conversion the compiler will make conversion of data from one data type to another without asking to programmer.
- The process of data conversion is performed by C# in a type-safe way i.e. convert data of one data type into other compatible data type. We can also say that we can convert source data from one type to targeting data type which require more memory size than type of source data.
- For example we can convert easily char value into integer and we can convert integer type value into float value easily.

➜ 2) **Explicit type conversion**

- These conversions are done explicitly by programmer using the built-in functions.
- Explicit conversions require a cast operator.

Program 1.2.45

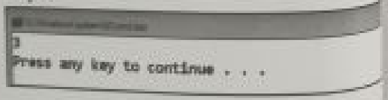
Write a program to demonstrate explicit type casting.

Solution :

Program to demonstrate explicit type casting

```
using System;
namespace TypeConversionDemo
{
  class ExplicitDatatypeConversion
  {
    static void Main(string[] args)
    {
      double a = 3.2;
      int j;

      j = (int) a;

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

Output



```
3
Press any key to continue . . .
```

- C# provides the following built-in type conversion methods :
  1. **ToBoolean()**- Converts a data type of data member to a boolean value(true or false).
  2. **ToByte()**-Convert data type of data member to byte.
  3. **ToChar()**-Converts a data type of data member into character type.
  4. **ToDateTime()**-Converts data type of data member which are either integer or string type into date-time format.
  5. **ToDecimal()**-Converts a floating point or integer type data member into a decimal type.
  6. **ToDouble()**-Converts data type of data member into a double type.
  7. **ToInt16()**-Converts a data type of data member into a 16-bit integer.
  8. **ToInt32()**-Converts a data type of data member into a 32-bit integer.

Program 1.2.46

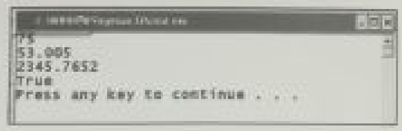
Write a program of built-in functions used in explicit type casting.

Solution :

Program of built-in functions used in explicit type casting

```
usingSystem;
namespaceTypeConversionDemo
{
  classExample
  {
        staticvoidMain(string[] args)
        {
            int a =75;
            float b =53.005f;
            double c =2345.7652;
            bool d =true;
                                  Convert integer value a into string
            Console.WriteLine(a.ToString()); ◄
                                  Convert float type value into string
            Console.WriteLine(b.ToString()); ◄

            Console.WriteLine(c.ToString()); ◄
                                  Convert double type value into string
            Console.WriteLine(d.ToString()); ◄
        }
  }                      Convert Boolean data type value into string
}
```

Output



```
75
53.005
2345.7652
True
Press any key to continue . . .
```

Syllabus Topic : Partial Classes

1.2.16 Partial Classes

- It is possible to split the definition of a class into two or more source files. Each source file contains a section of the class definition.
- All these parts are combined then into single class when the application is compiled.
- When working on large projects, spreading a class over separate files allows multiple programmers to work on it at the same time.
- Partial keyword can also use to split struct or interface over two or more files.

➜ **Benefit of partial classes**

1) More than one developer can write the code for the class at same time.
2) Main advantage is that visual studio uses partial classes to separate automatically generated system code from developer code.

1.3 ASP.NET

Syllabus Topic : Creating Websites

1.3.1 Creating Websites

- ASP.NET is a web application framework designed as well as developed by Microsoft to provide facility to programmers to create attractive web sites, web applications and web services.
- It allows you to use full featured various programming languages such as C#, VB.NET etc. to create web applications easily.
- It provides good combination of HTML, CSS and JavaScript to create dynamic featured application.
- First version of ASP.NET was released in January 2002.
- It is based on the Common Language Runtime (CLR).

ASP.NET is a web development language which provides a programming model and a variety of services which are essential to create robust web applications for personal computer as well as for mobile devices.

- ASP.NET use HTTP protocol for client and server communication over internet.
- The ASP.NET application codes can be written in any of the following languages which support .NET framework :

1. C#
2. Visual Basic.Net
3. Jscript
4. J#

- ASP.NET is used to create attractive, data-driven web applications over the internet. It provides a large set of controls such as text boxes, buttons, labels, checkboxes etc. to create attractive user interface.

Syllabus Topic : Anatomy of a Web Form

1.3.2 Anatomy of a Web Form-Page Directive

Q. Write note on page directives. (4 Marks)

- Web forms in ASP.NET expand the event-driven model of interaction between client and server.
- The web browser submits a web form as request to the web server and the web server send a HTML page as response to web browser.
- All user activities at client side are informed to the server for maintaining state i.e. to record all the actions done by user.
- The server processes the input provided by client and provide output to client.
- ASP.NET framework use Page state and Session state to maintain state of page.
- The state of page is the state of the client, that means the content of number of input fields provided by the client in the web form.
- The session contains the collective information received from number of pages that user visited and worked with.

Syllabus Topic : Page Directive

Page Directive

- Fundamentally Page Directives are commands which are used by the compiler when the page is compiled.
- Page directives in ASP.NET are commands which state the optional settings like recording customized controls and page language etc.
- These settings explain way of processing the web forms to the .Net framework.
- It is simple to include directives to an ASP.NET page.

Syntax

```
<%@ directive_name attribute=value [attribute=value]
%>
```

Following is the list of directives used in ASP.NET code :

1. **The Page Directive**

- When we want to state the attributes specific to an ASP.NET page then we need to use @Page Directive.

Syntax

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="Default.aspx.cs" Inherits="_Default"
Trace="true" %>
```

Example

```
<%@Page Language="C#" AutoEventWIreup="false"
CodeFile="Default.aspx.cs" Inherits="_Default"%>
```

The attributes of the Page directive are listed below :

i) **AutoEventWireup** : It is used to enable or disable events of methods . It takes value true or false.

ii) **Buffer** : It is used to enable or disable HTTP response buffering.

iii) **ClientTarget** : The name of the browser for which the server controls render the content.

iv) **ClassName** : It is used to set class name of the page.

v) **Description**: Informative text which is ignored by the parser.

vi) **Language** : It is used to set programming language in which code is written.

vii) **Src**: It is used to set file name of the code outside class.

viii) **ValidateRequest** : Used to check whether all input data is validated against a hardcoded.

ix) **CodeFile** : It is used to set name of the code behind file.

x) **Transaction** : It indicates whether the transactions are supported or not.

xi) **Debug** : The Boolean value that enables or disables compilation with debug symbols.

xii) **ErrorPage** : URL of the error page to transfer control if an unhandled exception occurs.

xiii) **EnableViewState** : The Boolean value that enables or disables view state across page requests.

xiv) **EnableSessionState** : The Boolean value that enables or disables session state across page requests.

Syllabus Topic : Doctype

1.3.3 Doctype

Q. What is use of Doctype in ASP.NET. ? (2 Marks)

- Use of doctype element ensures that output of page will be compliant to an XHTML standards.
- ASP.NET controls do not render font elements or attributes, such as bgcolor which would not conform to XHTML standards if the page includes an XHTML DOCTYPE element.
- So ASP.NET permits us to create Web pages that are conforming to XHTML standards.
- XHTML is a World Wide Web Consortium (W3C) standard.
- XHTML defines HTML as an XML document.
- Creating Web pages that are conforming to XHTML standards has several advantages :

  It ensure that the elements in the pages are well formed.
- As many browsers are supporting XHTML, creating pages that conform to XHTML standards support to guarantee that your pages render in same way in all browsers.
- Using XHTML helps to make pages conform more readily to accessibility standards.
- XHTML is extensible, permitting to define new elements.
- We can easily read XHTML page programmatically for situations in which the Web page is processed by a computer instead of being read by users and transformations can be used to manipulate the document.

Syllabus Topic : Writing Code - Code - Behind Class

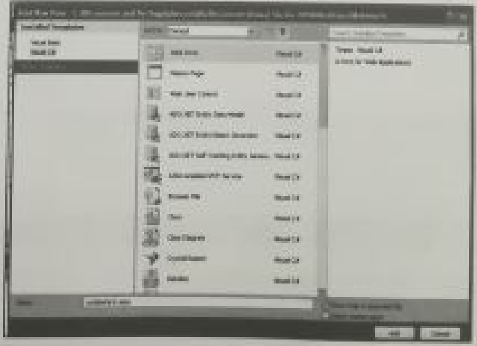
1.3.4 Writing Code-Code-Behind Class

Q. What is mean by code-behind class? How to write code behind class? (4 Marks)

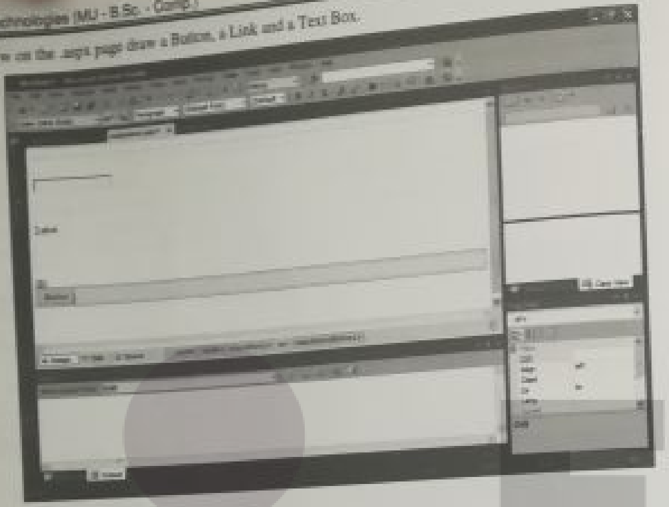
- Code Behind means code for an ASP.NET Web page that is written in a separate class file which can have the extension as .aspx.cs or .aspx.vb depending on the language used.
- We can write the code in a separate .cs or .vb code file for each .aspx page.
- One thing to remember in case of Code Behind is that the code for all the Web pages is compiled into a DLL file i.e. assembly that permits the web pages to be hosted free from any Inline Server Code.

Steps to write code behind the class

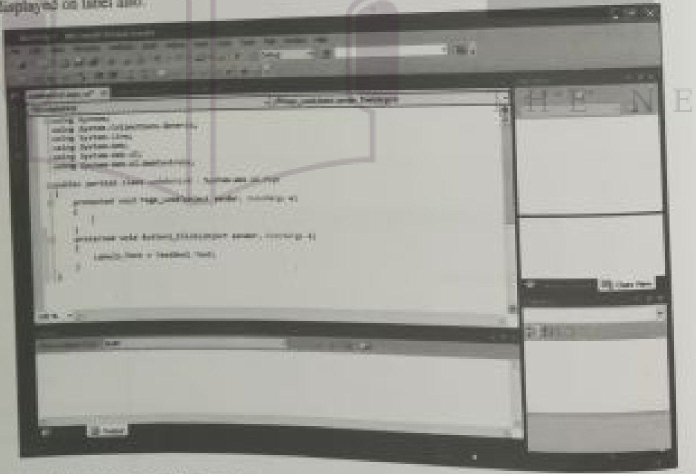
Step 1 : Create a new Blank Website in the Visual Studio and then add a Web page to it. Here we are creating a Web site for the Code Behind.



Step 2 : Now on the .aspx page draw a Button, a Link and a Text Box.



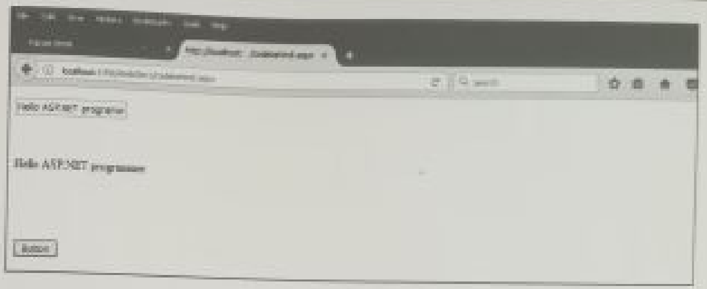
Step 3 : Now double-click on the Button, coding section will opened in a separate window whose extension is .aspx.cs. Write the code in this window. In this example we want that whenever text is written in textbox, it should get displayed on label also.



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class [illegible] : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        |
    }
    protected void Button1_Click(object sender, EventArgs e)
    {
        Label1.Text = TextBox1.Text;
    }
}
```

Step 4 : Now debug this page and verify that our program is running.

Output



Hello ASP.NET programer

Hello ASP.NET programmer

Syllabus Topic : Adding Event Handlers

1.3.5 Adding Event Handlers

1.3.5(A) Event

Q. What is mean by event ? (2 Marks)

- Change in the state of an object is called as event.
- Events are generated in response of user interaction with the Graphical User Interface (GUI) components like button, checkbox, radio button etc.
- For example, clicking on a button, moving the mouse, entering a character through keyboard, selecting an item from list, scrolling the page are the activities that are responsible for occurrence of an event.

Types of Events

There are two types of events:

1. **Foreground Events**

- These events need direct interaction of user. They are generated in response to a person interacting with the graphical components such as button, checkbox, radio button etc. in Graphical User Interface.
- Example of foreground events are clicking on a button, moving the mouse, entering a character through keyboard, selecting an item from list, scrolling the page etc.

2. **Background Events**

- These events does not need direct interaction of user are known as background events.
- Examples of background events are Operating system interrupts, hardware or software failure, timer expires, an operation completion.
- Event Handling is the process that controls the event and decides what action should happen when an event occurs.
- In ASP.NET Events occurred at the client machine and these events are handled at the server machine.
- The server has a program or lines of code explaining what to do when the event is occurred. This program is called event handler.
- When event occurrence message is transmitted to the server, it checks what type of event occurs and then execute event handler written for that event.

1.3.5(B) Event Arguments

Q. Give syntax to define event. (2 Marks)

- ASP.NET event handlers take two arguments and do not return any value. The first argument represents the object due to which event occurs and the second is event argument.

Syntax

```
Private void  EventName(object sender, EventArgs e);
```

1.3.5(C) Application and Session Events

Q. What are the application and session events? (2 Marks)

The important application events in ASP.NET are :

1. **Application_Start :** This event is occurred when the application is started.
2. **Application_End :** This event is occurred when the application stop its working.

The Session events are :

3. **Session_Start** : This event is occurred when a user first time requests a page from the application.
4. **Session_End** : This event is occurred when the session ends.

1.3.5(D) Page and Control Events

Q. What are the page and control events? (4 Marks)

- **Page and control events are :**

(i) **DataBinding** : This event is occurred when control is binds to a data source.

(ii) **Disposed** : This event is occurred when a page or the control is released.

(iii) **Error** : This event is occurred when an unhandled exception is thrown.

(iv) **Init** : This event is occurred when initialization of the page or the control is done.

(v) **Load** : This event is occurred when loading of the page or a control is done.

(vi) **PreRender** : This event is occurred when rendering of the page or the control is done.

(vii) **Unload** : This event is occurred when unloading of page or control from memory is done.

1.3.5(E) Event Handling Using Controls

Q. Write short note on event handling using controls? (4 Marks)

- All ASP.NET web controls i.e. Button, Checkbox etc. are implemented as classes and each control have events which are fired when user do specific actions on that control.
- For example, when a user clicks a button the 'Click' event is generated.
- There are predefined attributes and event handlers which are used for handling events.
- Event handler is program which execute in response to an event and take appropriate action on it.
- By default an event handler is created by visual studio by adding a Handles clause on the Sub procedure. This clause assign name to the control and event which is handled by the procedure.

The ASP.NET tag for a button

```
<asp:ButtonID="buttonCancel" runat="server
"Text="Cancel" Onclick="buttonCancel_Click"/>
```

The event handler for the Click event

```
Protected Sub buttonCancel_Click(ByVal sender AsObject,
ByVal e As System.EventArgs)
Handles btnCancel.Click
End Sub
```

The list of common events these attributes and controls on which they applied are given below.

Event	Attribute	Controls
Click	OnClick	Button, Image button, link button, image map
SelectedIndexChanged	OnSelectedIndexChanged	Drop-down list, list box, radio button list, check box list
TextChanged	OnTextChanged	Text box
Command	OnCommand	Button, image button, link button
CheckedChanged	OnCheckedChanged	Check box, radio button

- Due to some events form need to be send back to the server immediately. Such events are called as the postback events. For example the click event post back the form to server when we click on button.
- Some events do not send form back to the server immediately and these events are called as non-postback events. For example, the change events or selection events such as TextBox.TextChanged or CheckBox.CheckedChanged.
- The nonpostback events can be convert into post back immediately by setting value of their AutoPostBack property to true.

Program 1.3.1

Write a program of event handing by controls.

Solution :

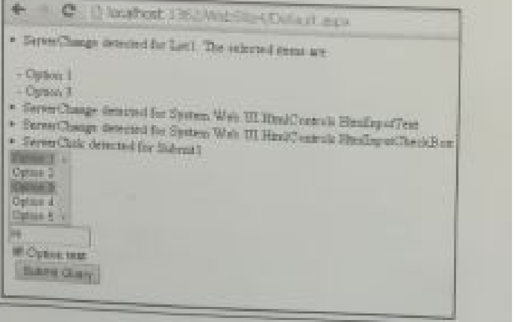
Program of event handling by controls

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="Default.aspx.cs" Inherits="ChangeEvents" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML
1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Change Events</title>
</head>
<body>
   <form id="form1" runat="server" >
   <div>
     <select runat="server"
          id="List1"
          size="5" multiple
          Name="List1"
          onserverchange="List1_ServerChange">
     <option>Option 1</option>
     <option>Option 2</option>
   </select>
   <br/>
   <input type="text"
        runat="server"
        ID="Textbox1"
        Size="10"
        Name="Textbox1"
        OnServerChange="Ctrl_ServerChange"> <br>
   <input type="checkbox"
        runat="server"
        ID="Checkbox1"
        Name="Checkbox1"
        OnServerChange="Ctrl_ServerChange">Option
text<br/>
      &nbsp;
   <input type="submit"
        runat="server"
        ID="Submit1"
        Name="cmdSubmit"
        value="Submit Query"
        onserverclick="Submit1_ServerClick">
     </div>
   </form>
 </body>
 </html>
```

ASP.NET file Demo.aspx.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
public partial class ChangeEvents : System.Web.UI.Page
{
    protected void Page_Load(object sender,
System.EventArgs e)
    {
            if (!Page.IsPostBack)
            {
                List1.Items.Add("Option 3");
                List1.Items.Add("Option 4");
                List1.Items.Add("Option 5");
            }
    }
    protected void Ctrl_ServerChange(object sender,
System.EventArgs e)
    {
          Response.Write("<li>ServerChange detected for
" + sender + "</li>");
    }
   protected void List1_ServerChange(object sender,
EventArgs e)
    {
          Response.Write("<li>ServerChange detected
for List1. " + "The selected items
          are:</li><br>");foreach (ListItem li in
List1.Items)
          {
              if (li.Selected)
                Response.Write("&nbsp;&nbsp;- " +
li.Value + "<br>");
          }
    }
   protected void Submit1_ServerClick(object sender,
EventArgs e)
    {
          Response.Write("<li>ServerClick detected for
Submit1.</li>");
    }
}
```

Output



localhost:1362/WebSite4/Default.aspx

- ServerChange detected for List1. The selected items are:

  - Option 1
  - Option 3
- ServerChange detected for System.Web.UI.HtmlControls.HtmlInputText
- ServerChange detected for System.Web.UI.HtmlControls.HtmlInputCheckBox
- ServerClick detected for Submit1

5(F) Default Events

Q. What is meant by default event and list some default event in C# ? (4 Marks)

- Every web control has a default event. for example, default event for the Page object is Load event and default event for the button control is the Click event.
- In visual studio, to create default event handler, we have to double click on the control in design view.
- List of default events for common controls :

Control	Default Event
RadioButton	CheckedChanged
RadioButtonList	SelectedIndexChanged
AdRotator	AdCreated
BulletedList	Click
Button	Click
CheckBox	CheckedChanged
Calender	SelectionChanged
CheckBoxList	SelectedIndexChanged
ImageButton	Click
ImageMap	Click
DataGrid	SelectedIndexChanged
DataList	SelectedIndexChanged
DropDownList	SelectedIndexChanged
LinkButton	Click
ListBox	SelectedIndexChanged
Menu	MenuItemClick
HyperLink	Click

Syllabus Topic : Anatomy of an ASP.NET Application-ASP.NET File Types

1.3.6 Anatomy of an ASP.NET Application-ASP.NET File Types

Q. Write note on ASP.NET file types. (4 Marks)

- Web site application can contains different types of files. Some file types are supported and managed by ASP.NET, and other file types are supported and managed by the Internet Information Server(IIS).
- Add New Item menu is used to create ASP.NET file types in Visual Studio. By using mapping, we can relate file type with application.

1.3.6(A) File Types Managed by ASP.NET

File type	Location	Description
.asax	Application root.	application class and optional methods i.e. event handlers , that run at various points in the application life cycle are present in Global.asax file.
.master	Application root or subdirectory.	Layout for other Web pages in the application. More information is defined by a master page.
.ascx	Application root or a subdirectory.	custom functionality that you can add to any ASP.NET Web Forms page is defined by this web user control file.
.config	Application root or a subdirectory.	XML elements that represent settings for ASP.NET features are present in the configuration file
.ashx	Application root or a subdirectory.	To give response to a Web request in order to generate dynamic content , this file handler is invoked .
.dll	Bin subdirectory.	A Dynamic Link library (DLL) is a library file which includes functions and source code that can be used by many programs at same time.
.asmx	Application root or a subdirectory.	classes and methods that can be invoked by other Web applications are included in this XML web services file
.browser	App_Browsers subdirectory.	features of an individual browser are identified in this browser definition file
.aspx	Application root or a subdirectory.	Web controls presentation and business logic are present in this ASP.NET Web Forms page.
.axd	Application root.	A handler file that is used to manage Web site administration requests, such as Trace.axd is managed by this handler file.
.skin	App_Themes subdirectory.	property settings to be applied to Web controls for consistent formatting is present in this skin file
.cd	Application root or a subdirectory.	This file type represent class diagram file.
.soap	Application root or a subdirectory.	This is a SOAP extension file.
.cs, .vb	App_Code subdirectory, or in the case of a code-behind file for an ASP.NET page, in the same directory as the Web page.	code that can be shared between pages, such as code for custom classes, business logic, HTTP modules, and HTTP handlers is defined in this source code file
.sln	Visual Studio project directory.	This is a solution file in Visual Studio
.csproj, .vbproj	Visual Studio project directory.	This is project file in Visual Studio Web-application project.
.disco, .vsdisco	App_WebReferences subdirectory.	Web services are located by using this XML web services discovery file
.sitemap	Application root or a subdirectory	logical structure of the Web application is defined by this sitemap file. ASP.NET includes a default sitemap provider that uses sitemap files to display a navigational control in a Web page.
.dsdgm, .dsprototype	Application root or a subdirectory.	A distributed service diagram (DSD) file that can be added to any Visual Studio solution that provides or consumes Web services to reverse-engineer an architectural view
.lic, .webinfo	Application root or a subdirectory.	To protect intellectual property by checking that a user is authorized to use the control is done by using licensing concept.
.sdm, .sdmDocument	Application root or a subdirectory.	This is a system definition model (SDM) file.
.mdb, .ldb	App_Data subdirectory.	This is an Access database file.
.resources, .resx	App_GlobalResources or App_LocalResources subdirectory.	A resource file that contains resource strings that refer to images, localizable text, or other data are stored in this resource file.
.mdf	App_Data subdirectory.	This is SQL Server Express database file.
.msgx, .svc	Application root or a subdirectory.	This is a Indigo Messaging Framework (MFx) service file.

Syllabus Topic : ASP.NET Web Folders

1.3.7 ASP.NET Web Folders

Q. Write note on ASP.NET web folders. (4 Marks)

Following are the ASP.NET web folders :

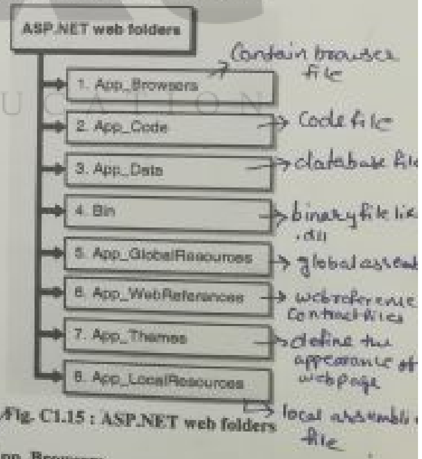


Fig. C1.15 : ASP.NET web folders

1. **App_Browsers**

- App_Browsers folder in a website includes web site related definitions of browser files that are used by ASP.NET to recognize browsers and states capabilities of those browsers.

→ 2. App_Code

- The App_Code folder is always available in the our project. This folder contains Source code for business objects and shared classes like cs and vb files etc. which we want to compile as element of our application.
- ASP.NET perform compilation of the source code which is present in the App_Code folder on the first request to our application in a dynamically compiled Web site project.
- For any changes in source code, recompilation of contents which are present in this folder is performed.

→ 3. App_Data

- Application data files such as .mdf database files, XML files, and another data store files are included in this folder.
- The App_Data folder is used by ASP.NET to store local database of application like the database for maintaining membership and role information.

→ 4. Bin

- Compiled assemblies i.e. .dll files for controls, components, or other code that we want to use in application are included in this folder bin.
- To automatically refer any classes in our application, code for that class must be present in Bin folder.

→ 5. App_GlobalResources

- Resources such as .resx and .resources files that are compiled into assemblies with global scope are included in this file.
- Resources in the App_GlobalResources folder are strongly typed and can be accessed programmatically.

→ 6. App_WebReferences

- Reference contract files such as .wsdl files, schemas such as .xsd files and discovery document files such as .disco and .discomap files that we create as Web reference for use in our application are included in this folder.

→ 7. App_Themes

- Group of files such as .skin files, .css files, image files and generic resources that define the appearance of ASP.NET Web pages and controls are included in this folder.

→ 8. App_LocalResources

- Resource files such as .resx and .resource files that are related with a particular page, user control, or master page in our application are included in this folder.

Syllabus Topic : HTML Server Controls

1.4 HTML Server Controls

- The HTML server controls are usually considered as the standard HTML controls which are created to enable server side processing.
- Number of HTML controls are not processed by the server but usually they are sent to browsers for display.
- When attribute runat="server" is added, they get converted into server control and when is attributed is added, they are available for server-side processing.
- For example, consider the HTML input control :

```
<input type="text" size="20">
```

- Add the runat and id attribute to convert into server control :

```
<input type="text" id="FirstText" size="20"
runat="server">
```

Advantages of using HTML Server Controls

- Even though All the functionalities of HTML server controls are available in ASP.NET server controls there are some advantages of HTML server controls.
- Layout can be done using static tables.
- HTML page can be converted so they can run under ASP.NET
- The Table 1.4.1 describes the HTML server controls and corresponding HTML tags :

Table 1.4.1

Name of Control	HTML tag
HtmlHead	<head>element
HtmlInputButton	<input type=button|submit|reset>
HtmlInputCheckbox	<input type=checkbox>
HtmlInputFile	<input type = file>
HtmlInputHidden	<input type = hidden>
HtmlInputImage	<input type = image>
HtmlInputPassword	<input type = password>
HtmlInputRadioButton	<input type = radio>
HtmlInputReset	<input type = reset>
HtmlText	<input type = text|password>
HtmlImage	<img> element
HtmlLink	<link> element
HtmlAnchor	<a> element
HtmlButton	<button> element
HtmlButton	<button> element
HtmlForm	<form> element
HtmlTable	<table> element
HtmlTableCell	<td> and <th>
HtmlTableRow	<tr> element
HtmlTitle	<title> element
HtmlSelect	<select&gt; element
HtmlGenericControl	All HTML controls not listed

Example

The following example uses a basic HTML table for layout. It uses some controls for accepting input from the users such as name, address, city, state etc. It also has a button control, which is clicked to get the user data displayed in the last row of the table.

Webform1.aspx

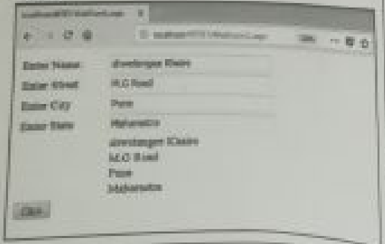
```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="WebForm1.aspx.cs"
Inherits="WebApplication11.WebForm1" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">


<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
   <title></title>
</head>
<body>
   <form id="form1" runat="server">
   <div>
   <table style="width: 54%;">
         <tr>
            <td >Enter Name:</td>
            <td>
               <asp:TextBox ID="name" runat="server"
style="width:230px">
               </asp:TextBox>
            </td>
         </tr>


         <tr>
            <td >Enter Street</td>
            <td>
               <asp:TextBox ID="street" runat="server"
style="width:230px">
               </asp:TextBox>
            </td>
         </tr>


         <tr>
            <td >Enter City</td>
            <td>
               <asp:TextBox ID="city" runat="server"
style="width:230px">
               </asp:TextBox>
            </td>
         </tr>


         <tr>
            <td >Enter State</td>

            <td>
               <asp:TextBox ID="state" runat="server"
style="width:230px">
               </asp:TextBox>
               </td>


         </tr>



         <tr>
            <td class="style1"> </td>
            <td ID="displayrow" runat ="server"
class="style2">
            </td>
         </tr>


      </table>
   </div>
   <asp:Button ID="Button1" runat="server"
onclick="Button1_Click" Text="Click" />
   </form>
</body>
</html>
```

Code behind file

Webform1.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace WebApplication11
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }

        protected void Button1_Click(object sender,
EventArgs e)
        {
            string str1 = "";
            str1 += name.Text + "<br />";
            str1 += street.Text + "<br />";
            str1 += city.Text + "<br />";
            str1 += state.Text + "<br />";
            displayrow.InnerHtml = str1;
        }
    }
}
```

Output



Enter Name [illegible] Khaire
Enter Street M.G Road
Enter City Pune
Enter State Maharashtra
[illegible] Khaire
M.G Road
Pune
Maharashtra

Syllabus Topic : View State

1.4.1 View State

Q.	What is mean by view state and how to create view state?	(4 Marks)

- We know that web application uses HTTP protocol as it is stateless.
- That means a new object of a page is created each time when we make a request to the server to get the page and after the execution of our page, page object has been lost immediately.
- It only happens because of one server on which all the controls of the Web Page are created and after use of those controls, the server destroys all the objects of that page. So to store the values of the controls, we use state management techniques.

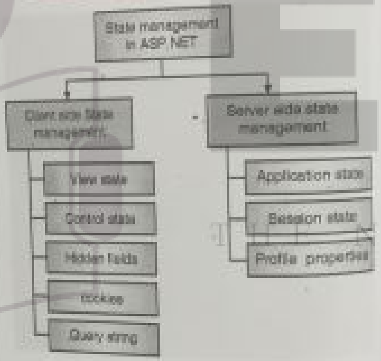


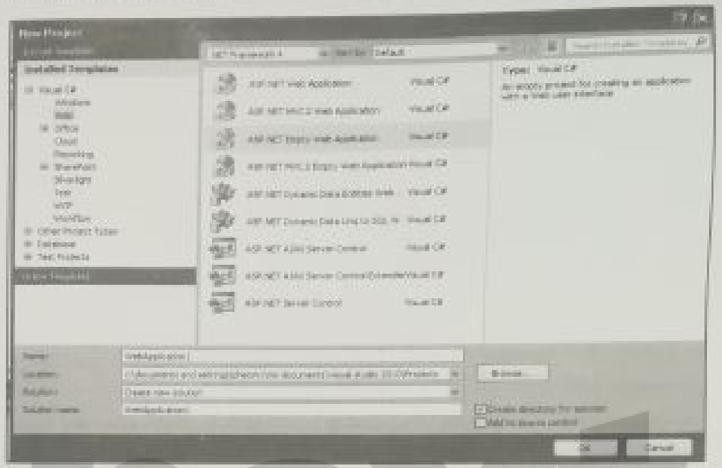
Fig 1.4.1 : State Management

- View State is the method to maintain the Value of the Page and Controls between round trips.
- View state is a Page-Level State Management technique. View State is turned on by default and normally serializes the data in every control on the page regardless of whether it is actually used at the time of a post-back.

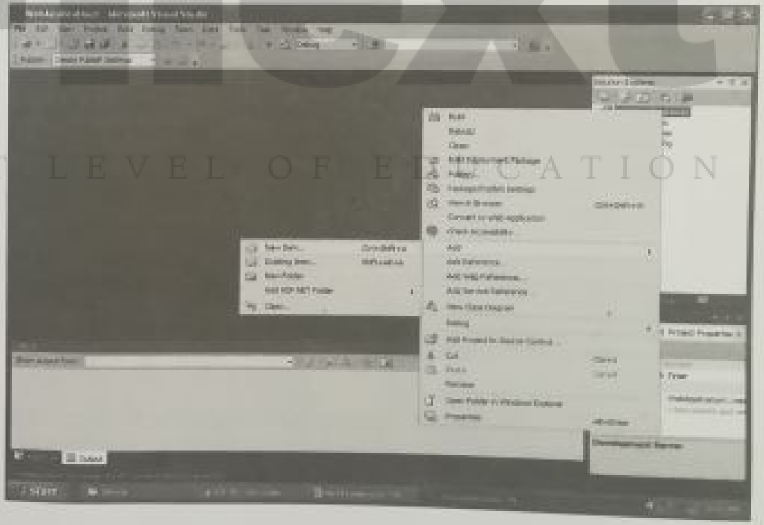
☞ **Steps to create view state**

Step 1 : Open Visual Studio 2010.

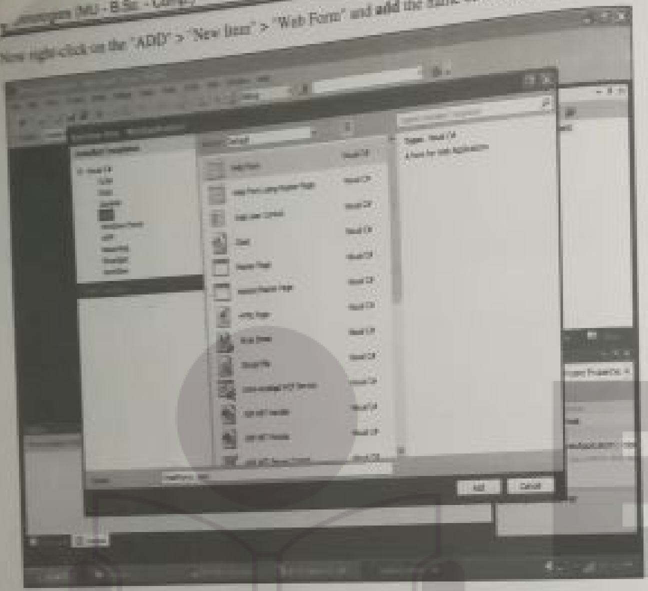
Step 2 : Then click on "New Project" > "Web" >"ASP.NET Empty Web Application".



Step 3 : Now click on Solution Explorer.



Now right-click on the "ADD" > "New Item" > "Web Form" and add the name of the Web Form.



Step 5 : Write following HTML and ASP.NET code and run web form.

Program 1.4.1 : Write a HTML code.

Solution :

HTML Code

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="WebForm1.aspx.cs"
Inherits="WebApplication1.WebForm1" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title></title>
</head>
<body>
<form id="form1" runat="server">
<div>
        user Name:-<asp:textbox id="TextBox1"
runat="server"></asp:textbox>
<br />
    Password :-<asp:TextBox id="TextBox2"
runat="server"></asp:textbox>
   <br />
<asp:button id="Button1" runat="server"
onclick="Button1_Click" text="Submit" />
<asp:button id="Button3" runat="server"
onclick="Button3_Click" text="Restore" />
    </div>
    </form>

</body>
</html>
```

Program 1.4.2

Write a program of view state ASP.NET page

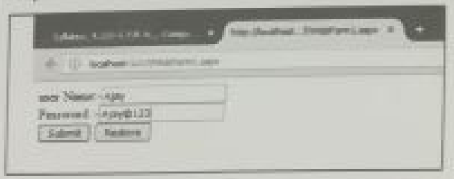
Solution : Program of view state ASP.NET page

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace WebApplication1
{
   public partial class WebForm1 : System.Web.UI.Page
   {
     protected void Page_Load(object sender, EventArgs e)
     {


     }
     public string a, b;
     protected void Button1_Click(object sender,
EventArgs e)
     {
       //
        ViewState["name"] = TextBox1.Text;
        ViewState["password"] = TextBox2.Text;


     TextBox1.Text = TextBox2.Text = string.Empty;

     }

     protected void Button3_Click(object sender,
EventArgs e)
     {


   if (ViewState["name"] != null)
   {
     TextBox1.Text = ViewState["name"].ToString();
   }
   if (ViewState["password"] != null)
   {
     TextBox2.Text = ViewState["password"].ToString();
   }


     }
   }
}
```

Value of TextBox1 and TextBox2 is assign on the ViewState

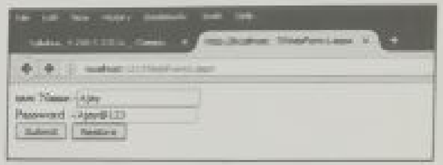
After clicking on submit Button TextBox value Will be Cleared

Output



user Name :-Ajay

Password :-Ajay@123



user Name :-Ajay

Password :-Ajay@123

- When we click on the Submit Button, the value of user name and password is submitted in View State and the View State stores the value of user name and password at the time of post-back.
- When we click on the Restore Button, we can get the value of user name and passwoed again.
- The Value of user name and password must be retained at the time of post-back and the values are stored into a base 64 encoded string and this information is then put into the View State Hidden Field.

Features of View State

1. There is no need of session to retain the control values after post-back.
2. The values of all the Pages as well as Control Properties are stored.
3. Generates a custom View State Provider which helps to store View State Information in any data source like SQL Server.

Advantages of View State

1. Easy to use.
2. No server side resources are required to maintains view state.
3. View state enhances security features using Unicode implementation.

Disadvantages of View State

1. **Security Risk :** The Information of View State can be seen in the page output source directly. Using extra coding, you can manually do encryption and decryption of contents of a Hidden Field. If security is

portant then consider use of Server-Based state Mechanism so that no sensitive information is sent to the client.

2. **Performance :** If we use a large amount of data then performance is not good, because View State is stored in the page itself and storing a large value can cause the page to be slow.
3. **Device limitation :** We cannot store large amount of view data in Mobile Devices, because they may not have that much memory capacity to store.
4. View state able to store values for the same page only.

☞ When We Should Use View State

1. When amount of data to be stored is small.
2. Data is not sensitive.

Syllabus Topic : HTML Control Base Class

1.4.2 HTML Control Base Class

Q. Write note on HtmlControl base class. (4 Marks)

- System.Web.UI.Html Controls is a namespace which contain HtmlControl class which is parent class for other HTML control classes.
- The HtmlControl object is very important to HTML server controls, because every property and method in this class is derived by each HTML server control class.
- If we understand the methods and properties of the HtmlControl class then we can understood about 80% of the methods and properties of all the objects in the System.Web.UI.HtmlControls namespace.
- Using HtmlControl class, we can easily understand other control classes.
- The HTML server controls have their own methods and properties. But their properties and methods are included in the HtmlControl object.

☞ Properties of HtmlContole class

Property	Description
Visible	This property gives a Boolean value i.e. either true or false. We can get and set value of this property. Value of this property indicates whether a control is rendered to HTML for delivery to the browser of client or not.
Attribute	Group of attributes of HtmlControl object are returned by this property.
Disabled	This property returns boolean value i.e. either true or false. We can get and set value of this property. Value of this property indicates whether a control is disabled or not.
ID	Value of this property is string that you can get and set. This property defines the Identifier for the control.
EnableViewState	This property returns boolean value i.e. either true or false. We can get and set value of this property. Value of this property indicates whether a control should maintain its view state or not.
TagName	Tag name of an element such as input or div are returned by this property.
Style	CSS Style Collection for a control is returned by this property.

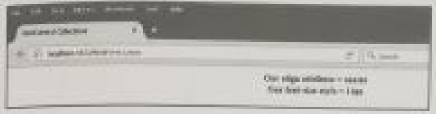
Program 1.4.3

Write a program of HtmlControl class.

Solution : Program of HtmlControl Class

```
<%@ page language="cs" runat="server"%>
<script runat=server>
void Page_Load()
{
OurDiv.InnerHtml = "Our align attribute = " +
OurDiv.Attributes["align"] + "<br>";
OurDiv.InnerHtml += "Our font-size style = " +
OurDiv.Style["font-size"];
}
</script>
<html>
<head>
<title>HtmlControl Collections</title>
<head>
<body>
<div id="OurDiv" align="center" style="font-size:14pt;
font-weight:bold" runat="server"/>
</body>
</html>
```

Output



```
Our align attribute = center
Our font-size style = 14pt
```

Syllabus Topic : HtmlContainerControl

1.4.3 HtmlContainerControl

Q. Write note on HtmlContainerControl class? (4 Marks)

- System.Web.UI.HtmlControls is a namespace which contains HtmlContainerControl class.
- HtmlContainerControl is used as the parent class for all HTML controls that requires a closing tag such as div, select, form etc.
- All properties and methods of HtmlContainerControl class are derived from the HtmlControl class and adds few properties and methods of its own.

☞ Syntax

```
public abstract class HtmlContainerControl : HtmlControl
```

☞ Constuctors of HtmlContainerControl class

1. **HtmlContainerControl() :** This is default constructor of HtmlContainerControl. This constructor Initializes a new instance of the HtmlContainerControl class using default values.
2. **HtmlContainerControl(String) :** This is parametrized constructor of HtmlContainerControl class. It initializes a new instance of the HtmlContainerControl class using the mentioned tag name.

☞ Properties of HtmlContainerControl class

Name	Description
Adapter	This property is derived from HtmlControl class which is used to get the browser-specific adapter for the control.
AppRelativeTemplateSourceDirectory	This property is derived from HtmlControl class which is used to get or set the application-related virtual directory of the Page or UserControl object that contains this control.
Events	This property is read-only and derived from HtmlControl class.This property is used to get a event handler delegates list for the control.
Attributes	This property is derived from HtmlControl which is used to get a group of all attribute name and value pairs mentioned on a server control tag within the ASP.NET page.
ID	This property is derived from HtmlControl class and it is used to get or set the programmatic identifier to the server control.
BindingContainer	This property is derived from HtmlControl class. This API supports the product infrastructure and is not intended to be used directly from our code. This property is used to get the control that contains this control's data binding.
ChildControlsCreated	This property is derived from HtmlControl class. This property is used to get a value that state whether the server controls and child controls have been created or not
InnerHtml	This property is used to get and set the content found between the opening and closing tags of the specified HTML server control.
ClientID	This property is inherited from HtmlControl class and used to get the control ID for HTML markup that is created by ASP.NET.
ClientIDMode	This property is inherited from HtmlControl class and used to get and set the algorithm which is used to generate the value of the ClientID property.
Visible	This property is derived from HtmlControl class which is used to get and set a value that state whether a server control is rendered as User Interface on the page or not.
Context	This property is derived from HtmlControl class which is used to get the HttpContext object related with the server control for the current Web request.

Name	Description
ViewState	This property is used to get a dictionary of state information that permit us to save and restore the view state of a server control across multiple requests for the same page.
SkinID	This property is derived from HtmlControl class which is used to get and set the skin to apply to the control.
Disabled	This property is derived from HtmlControl class which is used to get and set a value which state whether the HTML server control is disabled or not.
HasChildViewState	This property is derived from HtmlControl class which is used to get a value which state whether controls of current server child controls have any saved view-state settings.
Style	This property is derived from HtmlControl class which is used to get group of all cascading style sheet (CSS) properties which can be applied to a specified HTML server control in the ASP.NET file.
Page	This property is derived from HtmlControl class which is used to get a reference to the naming container of server control which creates a unique namespace for differentiating between server controls with the same ControlID property value.
IsChildControlStateCleared	This property is derived from HtmlControl class which is used to get a value stating whether controls contained within the control have control state or not.
IsTrackingViewState	This property is derived from HtmlControl class which is used to get a value that state whether the server control is saving changes to its view state or not.
IsViewStateEnabled	This property is derived from HtmlControl class which is used to get value indicating whether view state is enabled for the control.
NamingContainer	This property is derived from HtmlControl class which is used to get a reference to the naming container of server control which creates a unique namespace for differentiating between server controls with the same value of Control ID property.
Parent	This property is derived from HtmlControl class which is used to get reference to the parent control of server control in the page control hierarchy.
RenderingCompatibility	This property is derived from HtmlControl class which is used to get a value that states the ASP.NET version with which rendered HTML will be compatible with.
TagName	This property is derived from HtmlControl class which is used to get tag runat=server attribute and value pair.

Program 1.4.4

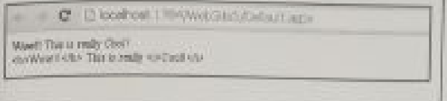
Write a program of HtmlContainerControl class.

Solution :

Program of HtmlContainerControl Class

```
<%@ page language="cs" runat="server"%>
<script runat="server">
 void Page_Load()
{
  string OurText = "<b>Wow!!</b> This is really
<i>Cool</i>";
   OurDiv1.InnerHtml = OurText;
   OurDiv2.InnerText = OurText;
}
</script>
<html>
<head>
<title>HtmlControl InnerStuff</title>
</head>
<body>
<div id="OurDiv1" style="font-size:14px"
runat="server"/>
<div id="OurDiv2" style="font-size:14px"
runat="server"/>
</body>
</html>
```

Output



Wow!! This is really Cool!
<b>Wow!!</b> This is really <i>Cool</i>

Syllabus Topic : HtmlInputControl Class

1.4.4 HtmlInput

Q. Write note on HtmlInput class. (4 Marks)

- The HtmlInput class is derived from the Html Control like HtmlContainerControl class and include some properties of its own .

Properties of HtmlInput class

Property	Description
Name	This property is used to get or set the unique name for the HtmlInput control.
Value	This property is used to get and set the value of the contents of the HtmlInput object.
Type	This property states the type of Input element HtmlInput control.

HtmlInput object types

Type	Html Server Control	Tag
Button	HtmlInputButton	<input type='button'runat='server'>
CheckBox	HtmlInputCheckBox	<input type='checkbox'runat='server'>
File	HtmlInputFile	<input type='file' runat='server'>
Hidden	HtmlInputHidden	<input type='hidden' runat='server'>
Image	HtmlInputImage	<input type='image' runat='server'>
Password	HtmlInputText	<input type='password' runat='server'>
Radio	HtmlInputRadioButton	<input type='radio' runat='server'>
Reset	HtmlInputButton	<input type='reset' runat='server'>
Submit	HtmlInputButton	<input type='submit' runat='server'>
Text	HtmlInputText	<input type='text' runat='server'>

Syllabus Topic : Page Class

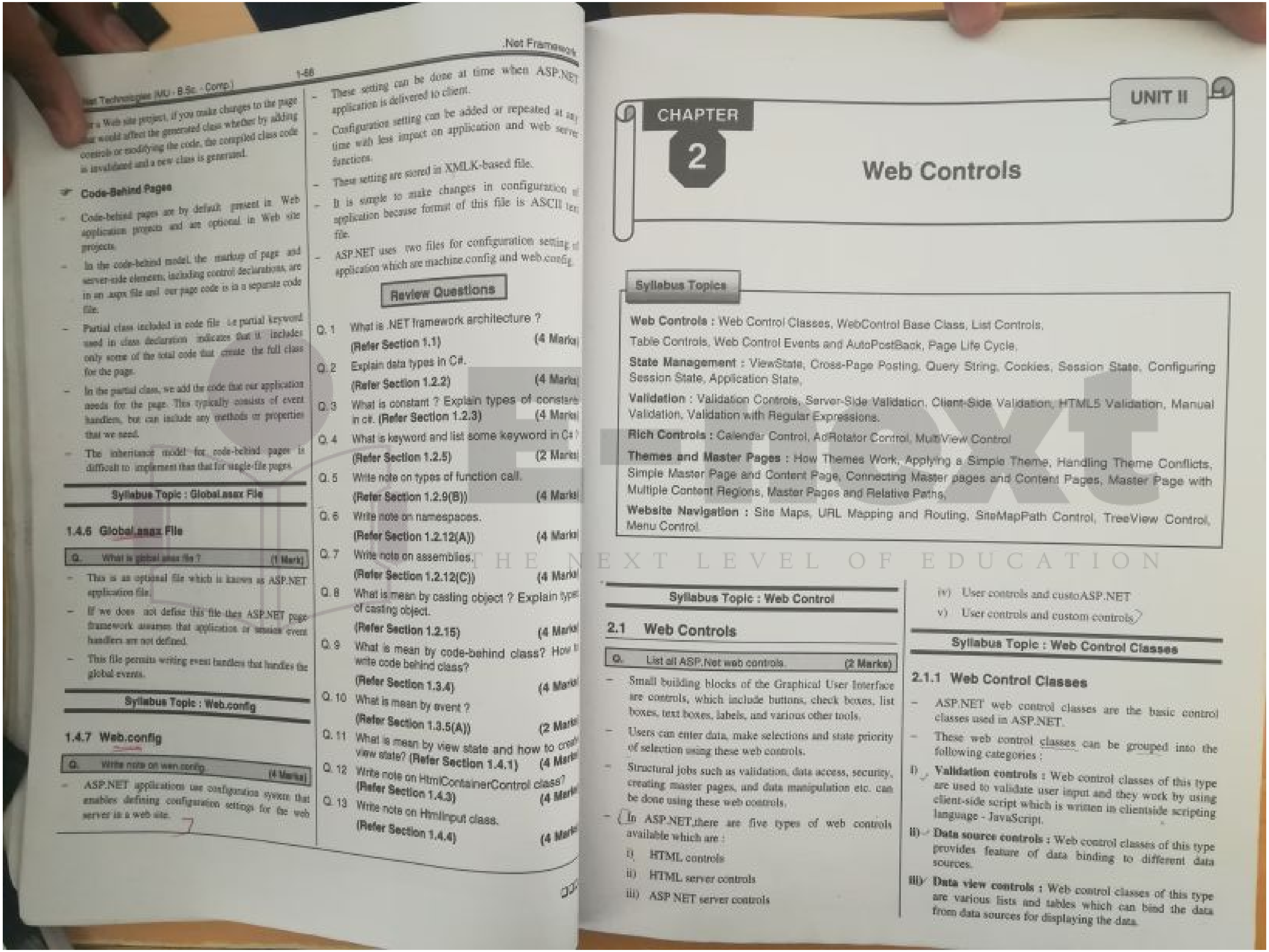
1.4.5 Page Class

Q. How to generate and run page class? (4 Marks)

- When an ASP.NET page is requested and HTML output is send to browser, ASP.NET creates an object of the class to represents the page.
- That class not only contains code that we have written for the page, but also the code that is automatically generated by ASP.NET.
- An ASP.NET page is executed as a single unit which includes server-side elements in a page such as web controls and event-handling code that we have written.
- We do not need to pre-compile pages into assemblies because we use a Web site project.
- Pages are dynamically complied by ASP.NET and executed when they are requested by a user at first time.
- When there is any change in page or resource , it needs to recompile the page.
- To enhance performance, pre-compilation of a Web project is done which is supported by web site project.
- ASP.NET Web application projects should be explicitly compiled before deployment at client side.
- The class or classes that the compiler creates depends on whether the page uses the single-file model or the code-behind model.

Single - file page

- We can create single-file pages in a Web site project, in which event-handling code ,the markup and server-side elements are all included in a single .aspx file.
- The ASP.NET generates and compiles a new class that is derived from the base Page class or from a custom base class defined with the derived attribute of the @ Page directive when page is compiled.
- For example, if you create a new ASP.NET Web page whose name is CodeSample1 in root directory of our application then a class named ASP.CodeSample1.aspx is generated that derives the Page class.
- The subfolder name is used as part of the generated class for pages in application subfolders.
- The generated class includes declarations for the controls in the .aspx page and the event handlers and other custom code are included in generated class.
- The generated class is compiled into an assembly, and when the page is requested, the assembly is loaded into the application domain, and then the page class is instantiated and executed to render output to the browser.



...a Web site project, if you make changes to the page that would affect the generated class whether by adding controls or modifying the code, the compiled class code is invalidated and a new class is generated.

Code-Behind Pages

- Code-behind pages are by default present in Web application projects and are optional in Web site projects.
- In the code-behind model, the markup of page and server-side elements, including control declarations, are in an .aspx file and our page code is in a separate code file.
- Partial class included in code file i.e partial keyword used in class declaration indicates that it includes only some of the total code that create the full class for the page.
- In the partial class, we add the code that our application needs for the page. This typically consists of event handlers, but can include any methods or properties that we need.
- The inheritance model for code-behind pages is difficult to implement than that for single-file pages.

Syllabus Topic : Global.asax File

1.4.6 Global.asax File

Q. What is global.asax file ? (1 Mark)

- This is an optional file which is known as ASP.NET application file.
- If we does not define this file then ASP.NET page framework assumes that application or session event handlers are not defined.
- This file permits writing event handlers that handles the global events.

Syllabus Topic : Web.config

1.4.7 Web.config

Q. Write note on web.config (4 Marks)

- ASP.NET applications use configuration system that enables defining configuration settings for the web server in a web site.
- These setting can be done at time when ASP.NET application is delivered to client.
- Configuration setting can be added or repeated at any time with less impact on application and web server functions.
- These setting are stored in XMLK-based file.
- It is simple to make changes in configuration of application because format of this file is ASCII text file.
- ASP.NET uses two files for configuration setting of application which are machine.config and web.config.

Review Questions

Q. 1 What is .NET framework architecture ? (Refer Section 1.1) (4 Marks)

Q. 2 Explain data types in C#. (Refer Section 1.2.2) (4 Marks)

Q. 3 What is constant ? Explain types of constants in c#. (Refer Section 1.2.3) (4 Marks)

Q. 4 What is keyword and list some keyword in C# ? (Refer Section 1.2.5) (2 Marks)

Q. 5 Write note on types of function call. (Refer Section 1.2.9(B)) (4 Marks)

Q. 6 Write note on namespaces. (Refer Section 1.2.12(A)) (4 Marks)

Q. 7 Write note on assemblies. (Refer Section 1.2.12(C)) (4 Marks)

Q. 8 What is mean by casting object ? Explain types of casting object. (Refer Section 1.2.15) (4 Marks)

Q. 9 What is mean by code-behind class? How to write code behind class? (Refer Section 1.3.4) (4 Marks)

Q. 10 What is mean by event ? (Refer Section 1.3.5(A)) (2 Marks)

Q. 11 What is mean by view state and how to create view state? (Refer Section 1.4.1) (4 Marks)

Q. 12 Write note on HtmlContainerControl class? (Refer Section 1.4.3) (4 Marks)

Q. 13 Write note on HtmlInput class. (Refer Section 1.4.4) (4 Marks)

□□□

CHAPTER 2 Web Controls

Syllabus Topics

Web Controls : Web Control Classes, WebControl Base Class, List Controls, Table Controls, Web Control Events and AutoPostBack, Page Life Cycle.

State Management : ViewState, Cross-Page Posting, Query String, Cookies, Session State, Configuring Session State, Application State.

Validation : Validation Controls, Server-Side Validation, Client-Side Validation, HTML5 Validation, Manual Validation, Validation with Regular Expressions.

Rich Controls : Calendar Control, AdRotator Control, MultiView Control

Themes and Master Pages : How Themes Work, Applying a Simple Theme, Handling Theme Conflicts, Simple Master Page and Content Page, Connecting Master pages and Content Pages, Master Page with Multiple Content Regions, Master Pages and Relative Paths,

Website Navigation : Site Maps, URL Mapping and Routing, SiteMapPath Control, TreeView Control, Menu Control.

Syllabus Topic : Web Control

2.1 Web Controls

Q. List all ASP.Net web controls. (2 Marks)

- Small building blocks of the Graphical User Interface are controls, which include buttons, check boxes, list boxes, text boxes, labels, and various other tools.
- Users can enter data, make selections and state priority of selection using these web controls.
- Structural jobs such as validation, data access, security, creating master pages, and data manipulation etc. can be done using these web controls.
- In ASP.NET,there are five types of web controls available which are :

i) HTML controls

ii) HTML server controls

iii) ASP.NET server controls

iv) User controls and custoASP.NET

v) User controls and custom controls

Syllabus Topic : Web Control Classes

2.1.1 Web Control Classes

- ASP.NET web control classes are the basic control classes used in ASP.NET.
- These web control classes can be grouped into the following categories :

I) **Validation controls :** Web control classes of this type are used to validate user input and they work by using client-side script which is written in clientside scripting language - JavaScript.

II) **Data source controls :** Web control classes of this type provides feature of data binding to different data sources.

III) **Data view controls :** Web control classes of this type are various lists and tables which can bind the data from data sources for displaying the data.