

Object Oriented
Programming with C# 6.0
Language

Object Oriented Programming with C# 6.0 Language



Linguistic Review
Hind M. Sami AL_Janabi
BSc. English Language/Linguistics

English Addition
August 2019

دار
النفيس
دار ابن النفيس
للنشر والتوزيع

The Hashemite Kingdom Of Jordan
The Deposit Number at The National
Library
(2018/8/4226)

005,133

ALkhouzaai, Wissam Ali

Object Oriented Programming With C#6.0

Language/ Wissam Ali ALkhouzaai.-Amman: Dar Ibn AlNafees for Publishing
and Distribution· 2018

() P.

Deposit No:2018/8/4226

Descriptors:/ computer programming lanuages// computer/


يتحمل المؤلف كامل المسؤولية القانونية عن محتوى مصنفه ولا يعبر هذا المصنف
عن رأي دائرة المكتبة الوطنية أو أي جهة حكومية أخرى

ISBN - 978-9923-718-62-9

البنفيس
دار ابن النفيس
للنشر والتوزيع

+962797135504

+962780080648

 Dar ibn alnafees

 dar_ibnalnafees@yahoo.com  alnafees02@gmail.com

Dedication

This book is dedicated to the martyrdom of justice and righteousness: Imam Hussain bin Ali, who led the battle of Taff (61) in Karbala.

Copying and reproducing this book without permission of the author is not accepted.

Introduction

This book discusses C#, one of the most famous and commonly used programming languages of the present.

The reason behind its preference lies in its simple yet strong structure and efficiency in programming the required tasks. Moreover, it is characterized by a unique flexibility that is retrieved from its object oriented programming (oop) supportive design.

This programming language is used in building software like (Applied, databases), dealing with the physical components of the computer and lastly to build Web pages).

All of these unique characteristics helped distinguish this language and made professional and beginner programmers abandon their old language once they have been acquainted with this language.

One point worth mentioning is that Microsoft, An American Corporation, designed this programming language and distributed it as a bundle known as (Visual studio .Net).

The book is divided into four sections: the first section is an introduction to the known programming languages and their types, then a brief history from the first languages up to the contemporary ones.

The second section discusses the central tokens for this language C#, and its structure, in addition to commands of reading, printing, conditional, repetitions, matrixes, structure, list and blocks.

The third section comprises the object –oriented programming and its details (objects &class), then follows a presentation of the features (inheritance, abstract, polymorphism and overloading).

At the end, dealing with errors and mistakes in C# were discussed.

The fourth section, the last, presents a detailed explanation of the terminology, how to initiate reading and writing.

At the end, a number of solved and unsolved examples were annexed for the reader to exercise and practice the theoretical information provided within this book.

Contents

Subject	Page Number
Chapter 1: Introduction	7
Computer Programming Language.	8
Introduction to C# Language	13
The history of the C# language	15
Algorithms.	18
Flow chart.	21
Chapter 2: The basics of the C# language	28
Tokens.	29
Variables Definition in C#.	37
How to open C# window.	48
Literals in C#.	50
C# Program Structure.	52
Read Statement in the C#.	55
Write statement in the C#.	56
Conditional Statements.	61
If Statement.	61
Switch Case Statement.	67
Loop Statements.	69
For Loop.	69
While Loop.	73
Do While Loop Statement.	80
Array.	83
Single Dimensional Array.	83
Double Dimensional Array.	87
Structure in C#.	94
List in C#.	97
Blocks in C#.	102
Procedure.	102
Function.	106
Passing parameter to the function.	112
Recursive Function.	120
Chapter 3: Object Oriented Programming	123

Subject	Page Number
Access Modifier.	131
Methods Types.	132
Constructors Methods.	138
OOP characteristic (Concepts).	143
Inheritance.	143
Abstract & interface Classes.	160
Polymorphism & Overloading.	164
Encapsulation.	168
Exception Handling.	170
Chapter 4: File Management in C#	175
I/O Class Types.	176
Create File Object.	178
Stream Reader Class.	176
Stream Writer Class.	181
File Access Class.	184
Home Work Questions.	191

Chapter 1

Introduction

Languages types.

1. Machine and assembly languages.

A machine language consists of the numeric codes for the operations that a particular computer can execute directly.

The codes are strings of 0s and 1s, or binary digits (“bits”), which are frequently converted both from and to hexadecimal (base 16) for human viewing and modification.

Machine language instructions typically use some bits to represent operations, such as addition, and some to represent operands, or perhaps the location of the next instruction.

Machine language is difficult to read and write, since it does not resemble conventional mathematical notation or human language, and its codes vary from computer to computer.

Assembly language is one level above machine language.

It uses short mnemonic codes for instructions and allows the programmer to introduce names for blocks of memory that hold data. Thus, one might write “add pay, total” instead of “0110101100101000” for an instruction that adds two numbers.

Assembly language is designed to be easily translated into machine language, although blocks of data may be referred to by names instead of their machine addresses, assembly language does not provide more sophisticated means of organizing complex information, like machine language, assembly language requires detailed knowledge of internal computer architecture, it is useful when such details are important, as in programming a computer to interact with input/output devices (printers, scanners, storage devices, and so forth).



4. Education-oriented languages.

- A. BASIC.
- B. Pascal.
- C. Logo.
- D. Hyper talk.



5. Object-oriented languages.

Object-oriented languages help to manage complexity in large programs. Objects package data and the operations on them, so only the operations are publicly accessible and internal details of the data structures are hidden.

This hiding information made large-scale programming easier by allowing a programmer to think about each part of the program in isolation.

In addition, objects may be derived from more general ones, “inheriting” their capabilities, such an object hierarchy made it possible to define specialized objects without repeating all that is in the more general ones.

Object-oriented programming began with the Simulate language (1967), which added hiding information to ALGOL.

Another influential object-oriented language was Smalltalk (1980), in which a program was a set of objects that interacted by sending messages to one another.

- A. C++& C#.
- B. Ada.
- C. Java.
- D. Visual Basic.



They are intended to solve relatively small programming problems that do not require the overhead of data declarations and other features needed to make large programs manageable.

Scripting languages are using for writing operating system utilities, for special-purpose file-manipulation programs, and, because they are easy to learn, sometimes for considerably larger programs.

A- PERL (practical extraction and report language) was developed in the late 1980s, originally for using with the UNIX operating system.

It was intended to have all the capabilities of earlier scripting languages.

PERL provided many ways to state common operations and thereby allowed a programmer to adopt any convenient style.

In the 1990s it became popular as a system-programming tool, both for small utility programs and for prototypes of larger ones.

Together with other languages discussed below, it also became popular for programming computer Web “servers.”



Introduction to C# Language.

In June (2000), Microsoft Corp. announced about new environment called (.Net) and a new language called (C#),C-Sharp, as the language of the (C#) is a powerful representation of the language (strongly-typed) and with object-oriented, designed in order to provide the best combination of simplicity , expressive and performance.

The environment (.Net) is grounded on the run time, general language and Common Language are similar to machine Java Virtual (JVM) and a set of

libraries that can be exploited by a large number of languages that can work together when converted compiling to an intermediate language (IL). That each of the (C#) and the environment of the (.Net) mutually reinforcing as much as, and that some of the characteristics of the (C#) put to work better with the (.Net) and some properties (.Net) put to work better with the (C#) although the target of (.Net) that works well with multiple languages.

(C#) Was constructed languages, is derived from a number of previous languages, but the most obvious is a Java (Java Language) and (C++) were written language before (Andres Heyzberg Anders Hejlsberg) involvement with (Scott Altmuth Scott Wiltamuth) as that.

(Heyzberg) was also designed a (.Net) and lead the work.

The (C#) language is the simple language contains only 80 keywords and keyword (12) of templates or data types, data types planted there.

But the (C#) has a high graphical capability when building modern concepts code.

Including (C#) totally in support of the programming structure and structural components based on component-based and object-oriented to object-oriented, and that one would expect from any modern language.

So (C#) language was inspired by the best previous languages, the most important characteristics of Small Talk, C++, Java, Delphi, also moved away from many of the disadvantages, and it is very similar in form to (C++, Java) languages.

The development of language began by a small team at Microsoft, led by two senior engineers, named: (Andres Heyzberg Anders Hejlsberg) and (Scott Altmuth Scott Wiltamuth) Famously (Heyzberg) invented (Turbo Pascal) one of the first integrated development environments, as well as for his leadership of the team work designed development (Delphi) and Component Library group environment in which (VCL) from (Borland).

So the (C#) as a language of object-oriented supports the definition of classes and deals with them, classes define types of new types allow an expansion of the language in order to better address the issues to be solved.

C# has the keywords to define new classes, curriculum, their properties and to carry out packaging encapsulation (inheritance, polymorphism and conformation) are the three pillars of programming oriented object.

In (C#) all is about class declaration as in the same definition (like in the language of Delphi).

Classes defined in (C#) did not require a separate header files or the Interface Definition Language (IDL), more than that (C#) supports pattern (XML) new for internal documentation in the synthesis, which facilitates the extraction of the documentation processes reference and assistance to the application files.

The (C#) also Supports interfaces, and is a way contract with classes for limited services only in the parameters required by the interface.

In (C#) classes cannot genetics inherit only one out, but classes can implement multiple interfaces.

When you execute the facade classes (C#) to provide all functions specified by the interface.

The (C#) also Supports structs, and is a concept that completely changed from what it was in C ++.

In C# building is to be somewhat restricted, and is thin that does not constitute a burden when it is created on the operating system or memory as in the case of the classes.

Building structs cannot inherit from classes or inheritance from it, but classes could implement Interface.

The (C#) provides the elements of component-oriented, such as properties, events, and builders definition are called attributes.

Oriented programming component supported in partial (CLR and in .Net) environment where you store your profile data synthesis classes. Metadata describing the classes and components of the curriculum and properties, as well as the security of their needs, and other features such

as if it can be serialized to pass through the borders of networks, or synthesis container to how to deal with jobs.

Therefore, the classes are compiled and integrated unit of information, so the environment can host her know them alone and how to read the definition of data classes, which does not need further information from separate sources.

Using C# and CLR can add specific metadata classes create their own attributes.

On the other hand it is possible to read the special definition data using CLR types are showing this data through reflection characteristics reflection in it.

Also assembly is a collection of files that appear to the programmer as if they were critical link library (DLL) file or an executable file and it is complex in (.Net) main unit in order to do reuse , version control and security.

CLR provides a set of classes for handling complexes.

The (C#) also provides support direct access to memory using pointers pattern (C++) and keywords in order to restrict such operations as the (unsafe) and to alert (garbage collector) in the CLR that does not pick objects up to which they refer indicators until they are released by the program.

The history of C# language.

Development of (.Net) platform began that had a set of class libraries, which named caretaker round Simple Managed C system or Acronym SMC using to do so.

Later, specifically in January 1999 form (Anders Hejlsberg) team of developers in order to build a new language named (Cool), and is a name of acronym for the phrase object-language orientation is similar to language (C).

Any (C-like Object Oriented Language), Microsoft decided to maintain this name but he abandoned it later for legal reasons related to the rights of registered marks.

In parallel with that (.Net) project was officially announced at a conference for professional developers (PDC) in July 2000 and was renamed to (C#) language was also exported the private execution time language (ASP.net) In addition to classroom libraries to this language.

Mark designed the Java language (James Gosling) and (Bill Joy), one of the founders of the company (Sun Microsystems) that brought in a language (Java).

That language (C#) is not only a traditional Java language; (Gosling) said ("It is as java somewhat, but after giving up reliability, productivity and safety").

Then wrote all of (Klaus Krefl) and (Ongelka Langer) in an article for them in a blog (The Java Lucy # of GTA almost identical programming.

This is a repeat tediously lacks creativity), it is very difficult to argue that the (Java) or (C#) is a programming language revolutionary who changed the way we write software, we have borrowed (C#) a lot of (Java) and vice versa.

Where the supports (C#) canning feature dismantling canning now and soon you will find a similar feature in the (Java).

(Anders Hejlsberg) in July 2000 said that the (C#) is not a "version of Java" but it is "much closer to the language of (C++) in terms of design.

In November 2005 announced a version (2.0) of (C#) It began here (C#) and Java development in a growing divergence trends.

The first and most important of these differences was to add the public styles Generics to both languages where he achieved these very stereotypes difference where the (C#) deals with the public styles real rows and generate their own code execution time, while Java with these styles feature had added treat to manage language the developer of a

public writing code as the interpreter was able to ensure that only healthy patterns, while these patterns are not turning into a real-time implementation patterns are not generating its own code along the lines of (C#).

In addition, a set of important features have been added to the (C#) in order to enable the use of the functional programming Grace note Link added in version (3.0) and frame programmatic support for expressions (Lambda) and ancillary roads and styles are called.

These features enable the developer to use functional programming techniques when it is advisable to do so.

Evolution of C#

C# Version	.NET Framework	Features	http://www.webdevelopmenthelp.net/
C# 1.0 (2002)	.NET Framework 1.0	Managed Code	
C# 2.0 (2005)	.NET Framework 2.0	Generics, Anonymous Methods, Nullable Types	
C# 3.0 (2008)	.NET Framework 3.5	Lambda Expressions, Anonymous Types, LINQ, Extension Methods, Expression Tree, Implicit Typing (var)	http://www.webdevelopmenthelp.net/
C# 4.0 (2010)	.NET Framework 4.0	Late Binding (dynamic), Optional parameters, Named Arguments, Extended COM Support	
C# 5.0 (2012)	.NET Framework 4.5	Async Features, Caller Information	
C# 6.0 (2014)	.NET Framework 5.0	Auto-Property Initializers, Exception Filters, await in catch and finally block, Primary Constructors, Declaration Expressions, Dictionary Initializer	

Design of the C# language goals.

The most important goals that led to the design of this language are summarized in the following points:

1. C# language should be simple and modern, general-use object-oriented.
2. The language should provide the investigation in support of the principles of software engineering such as strong verification of patterns or what is known as the (static verification) , verification of the limits of

the matrices and the discovery of attempts to use variables are configured and automatic garbage collection.

As well as emphasizing the importance of durability, sustainability and productivity software programmer.

3. The use of language enables you to develop software components usable in distributed environments.

4. The load source code with the goal of high importance, as well as programmed load, especially for an experienced C++ language and the language C.

5. Support for localization and globalization with a high importance of the goal.

6. C# programming language should be suitable for private embedded systems and host systems, whether large applications use complex operating systems, or simple applications have specific functional applications.

7. Although it should be on applications written in C# language to economize on memory use and processing power, but the language is not intended to compete directly with the performance and size of applications written in C or assembly language.



Naming.

The name of "C sharp" was inspired by the music symbol (#) indicates where the symbol (#) that the note have written elevated more by half a musical note, similar to the label with the language of the name (C++) which indicates the ("++") to the need to unstable increase (1).

Looks like the symbol (#) form of four signs (+) in the network (2 X 2) to imply that language is increasing of (1) the language (C++).

Has been chosen as the net symbol (#) Unicode (U code +0023) to represent Sharp icon in the scripting language named instead of the symbol (#) Unicode (U code + 266F) because of technical limitations prevent the show, such as lack of support for standard lines and some browsers to code (#) as well as the lack of it on the keyboard.

This tradition also dependence on the standard language descriptions (ECMA 334).

In any case, use Microsoft's true musical icon when they can practically do so, for example, in advertising campaigns or on the product casing. Subsequent used "mustache" in a number of .NET languages based on other languages, including languages (J #).

Where the language of dot Net designed by Microsoft through the derivation of the language (Java) and the language (A Sharp A#) derived from the language of (Ada) and the language of functional programming (F Sharp F #).

Algorithms.

It is a several of arguments instructions to execute arithmetic operations and logical operations and other wise with form sequential and argument.

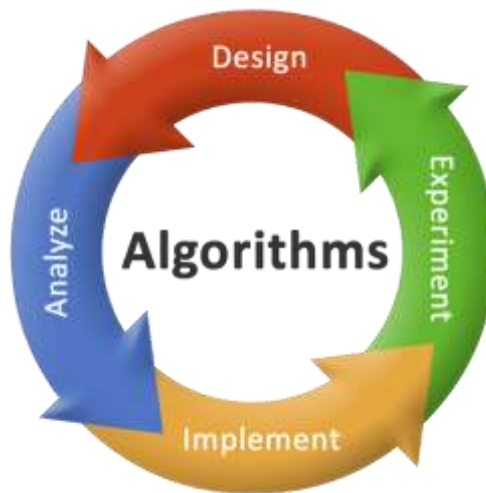
For Writing a successful program, first it requires to analyze and design a successful program Analysis process leading up to write the program, thus must be expressed in general terms, and as we mentioned in the definition of the algorithm is a set of public steps to resolve a particular issue and therefore, must be expressed in general terms independent of the programming language.

There are many ways to express the algorithm like; charts Stream, charts and semi-encryption schemes and all of these methods show the program works in different methods.

The algorithm formation process begins defining issue followed by the analysis process to develop and establish general rules of the issue and turn it into a simple procedure to resolve the matter, in order to be successful algorithm should have certain characteristics, such as.

- Finitude.
- Cohesion.
- Reliability.

The algorithm is a label that name is an ascription of the Arabic scientist, Abu Ja'far Muhammad ibn Musa al-Khwarizmi, a mathematician.



What are the terms of writing the algorithm?

1. Every algorithm have a beginning and an end.
2. The algorithm can be written in any language. (Arabic, English ...)
3. Taken into account when writing the algorithm to be shortened as much as possible.
4. Each step must be numbered in the algorithm, where the figure is one that represents the first step in the solution which is the beginning.
5. Advisable writing variables in the algorithm in English if possible.
6. Prefers writing arithmetic and logical operations the same way as written in the program if possible.

7. Whenever solution contains fewer steps, the more the better.
8. Use the minimum number of steps and the lowest number of variables to arrive at a solution.
9. Last step in the algorithm is the end, represents the end to resolve the matter.
10. There are an infinite number of solutions for any issue, but whenever the number of steps the solution is less, the better.

Algorithms Structure.

There are three structures to build programs and writing algorithms, they are.

1. Sequence.

The algorithm is a set of sequential instructions, these instructions may be either simple or from the following two types.

2. Selection.

That some problems can't be solved by a simple sequence of instructions, and you may need to test some of the conditions and look at the result of the test, if the result is correct track Includes sequential instructions, and if the wrong follow another path different from the instructions.

This method is the self-styled decision or choice.

3. Repetition.

When solving some of the problems it is necessary to reproduce the same sequence of steps a number of times.

This is so-called redundancy.

Example 1.

Write algorithm to find the marks average of three subjects?

$$AV = (D1 + D2 + D3) / 3$$

- 1- Begin.
- 2- Read Three Degree (D1, D2, D3).
- 3- Execute the equivalence $AV = (D1 + D2 + D3) / 3$.
- 4- Write (AV).
- 5- End.

Example 2.

Write algorithm to find the area of the circle, so we can use the following equation to find this area?

$$A=R^2 * P$$

- 1- Begin.
- 2- Read (R, P).
- 3- Execute the equivalence $A=R*R*P$.
- 4- Write (A).
- 5- End.

Example 3.

Write algorithm to find the value of (Y) So:

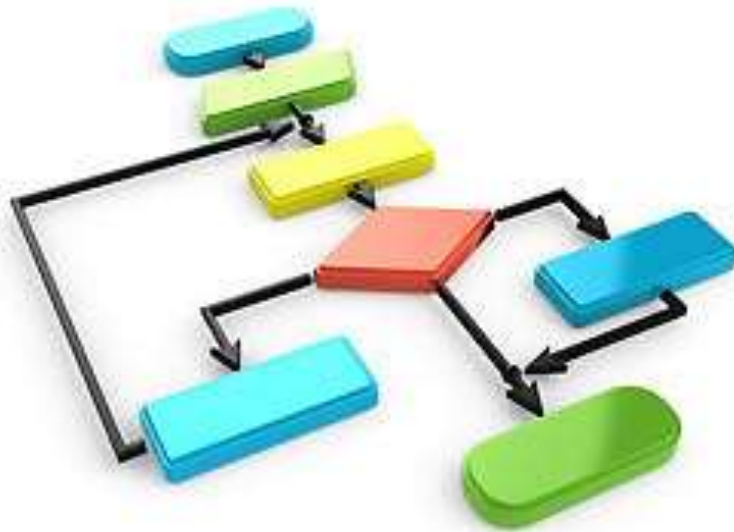
$$Y = \left\{ \begin{array}{ll} X & \text{If } X > 0 \\ -X & \text{If } X < 0 \end{array} \right\}$$

- 1- Begin.
- 2- Read (X).
- 3- If $X \geq 0$ Then $Y=X$ And Go To Step 5.
- 4- $Y=-X$.
- 5- Write (Y).
- 6- End.

Flow chart.




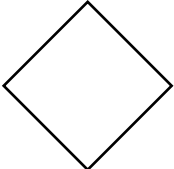
It is a photographer for the algorithm shows the steps to solve the problem from beginning to end which hide details to give a general picture of the solution representation.

It describes the flow of operations in the program including loops, control structures and decision-making.



Flow chart Symbols.

The following figures representing the symbols using to represent the problem solution by flow chart.

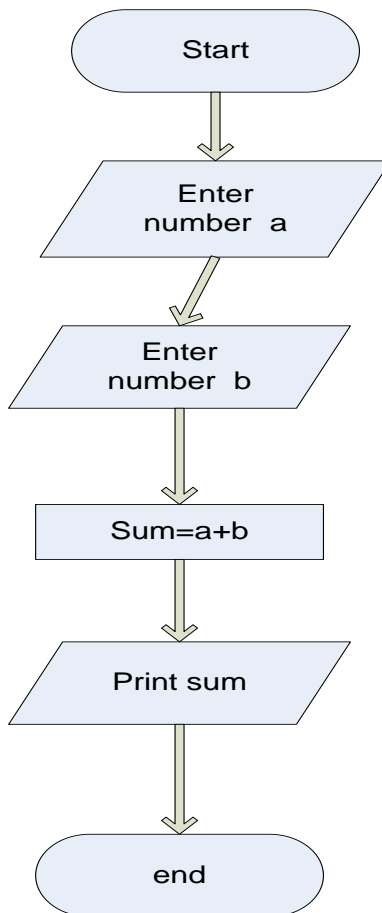
Symbol	Operation
	Start And End
	Read And Write
	Process
	Make decision

Symbol	Operation
↓	Direction of Operation

There are three types of flow charts, they are.

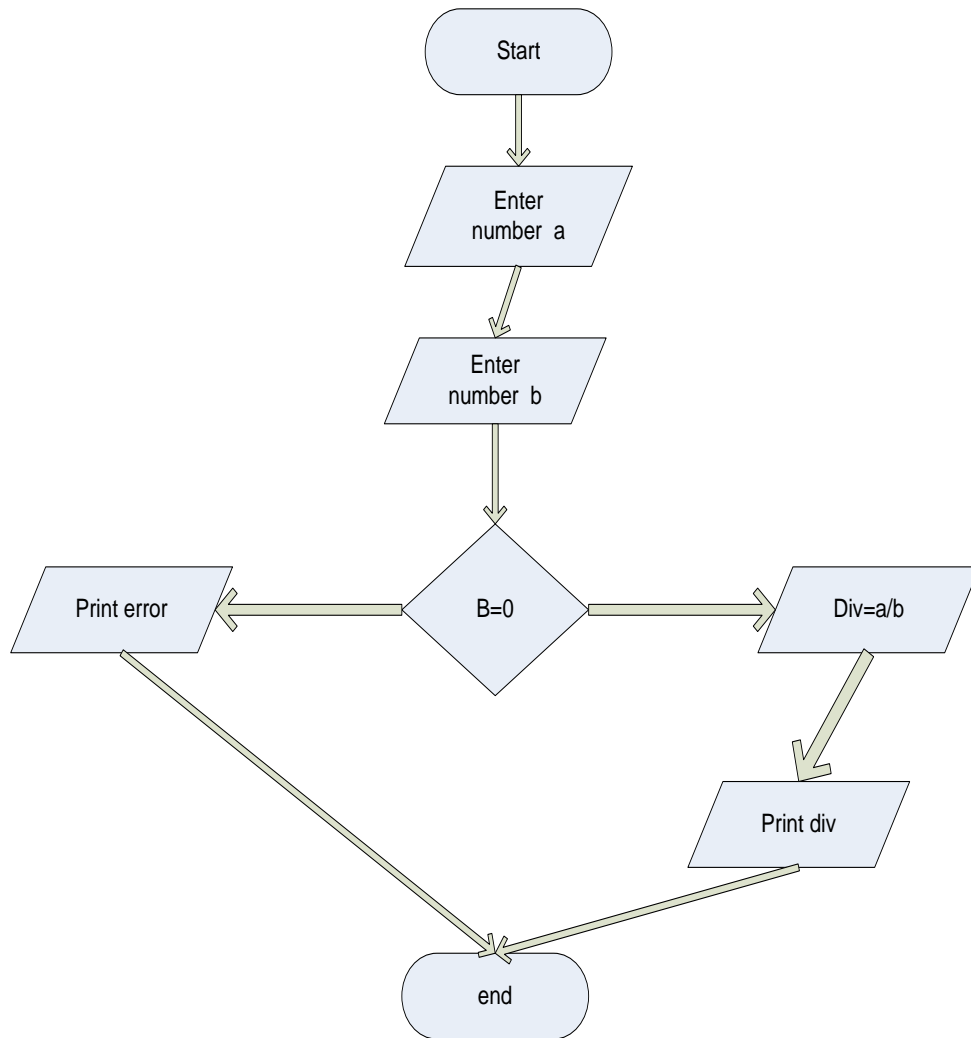
1- Sequential Flowcharts.

They are sequential steps to do specific job, they are shown in the following flowchart.



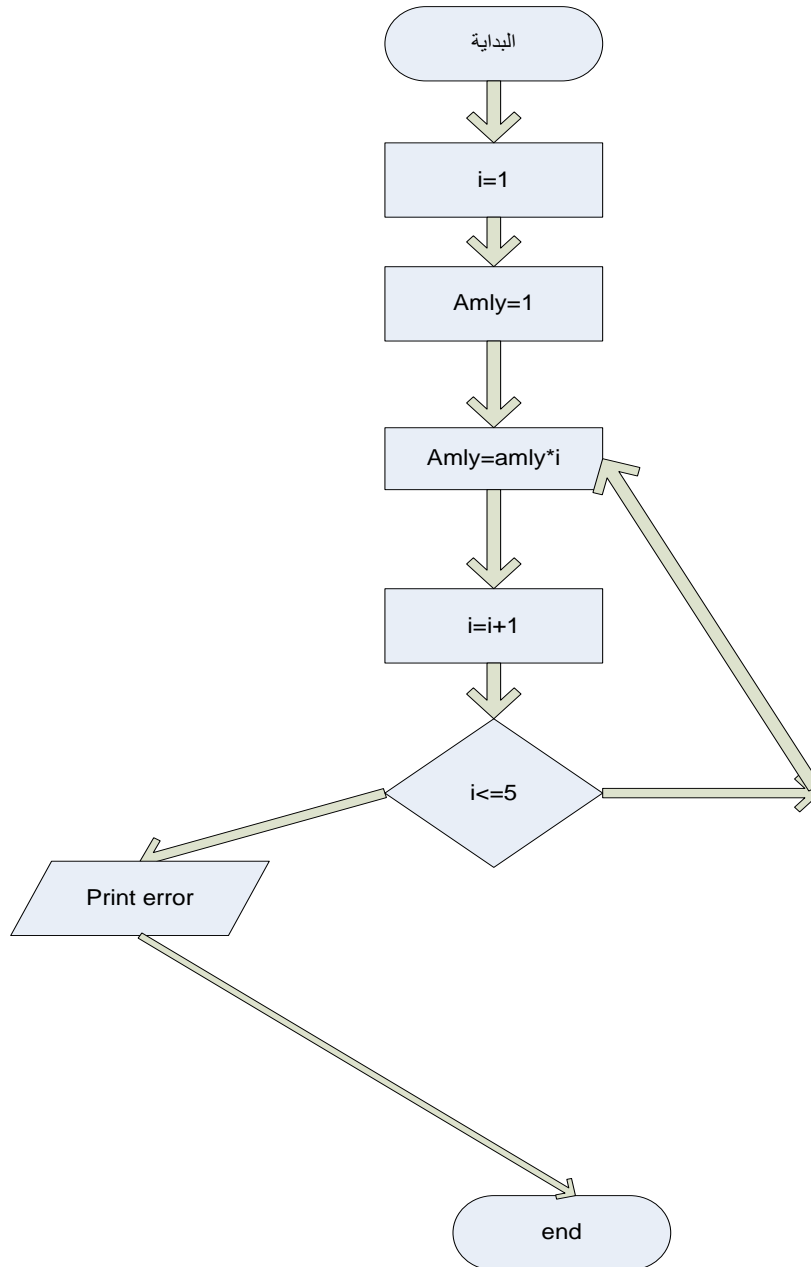
2- Branched Flowcharts.

These types will not be sequential, but will be like branches, they are shown in the following flowchart.



3- Loop Flowcharts.

These types will have specific loop to perform a specific job, they are show in the following flowchart.

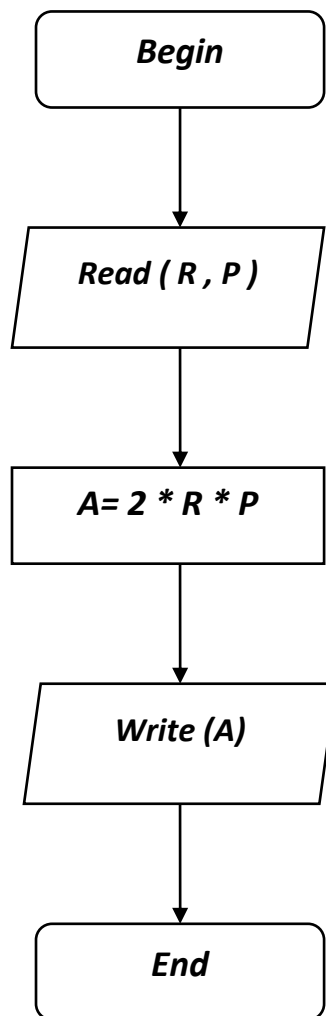


Example 1.

Write algorithm and draw the flowchart to find ambulant circle?

$$A=2*R*P$$

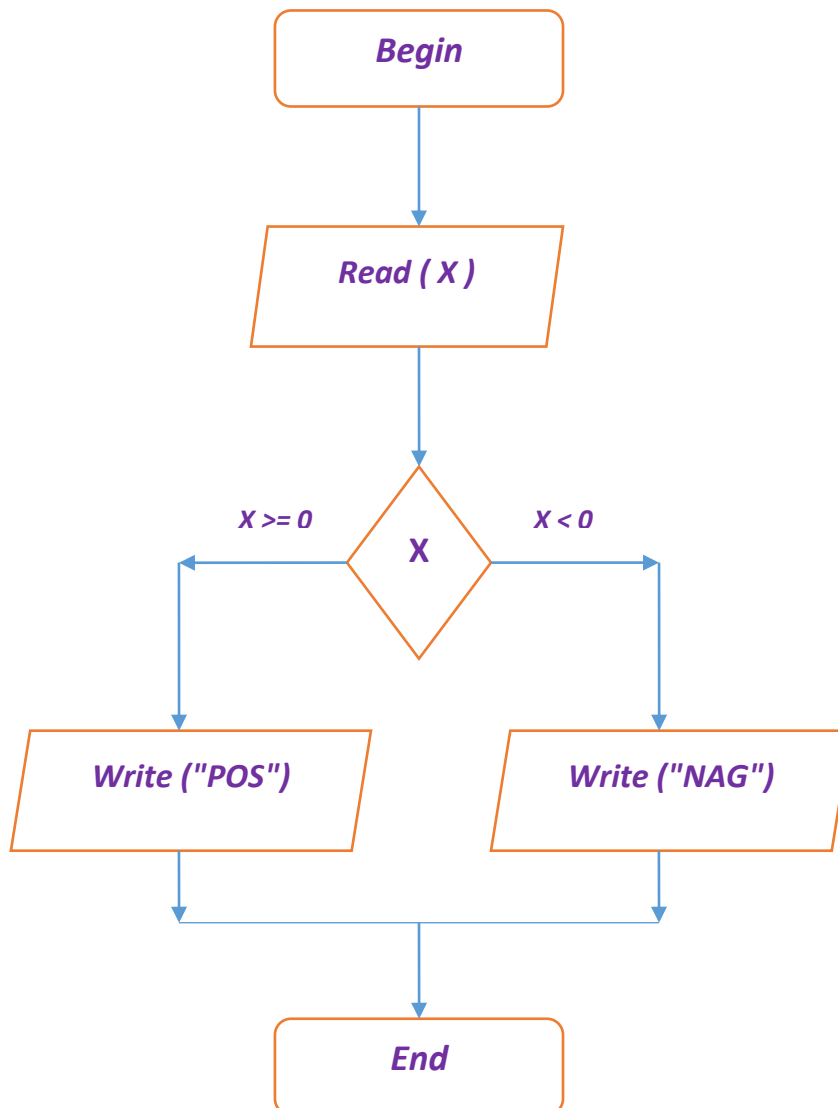
- 1- Begin.
- 2- Read (R, P).
- 3- Execute $A=2*R*P$.
- 4- Write (A).
- 5- End.



Example 2.

Write algorithm and draw the flow chart to read the value of (x). If (x) positive write ("pos") or if (x) negative write ("nag")?

- 1- Begin.
- 2- Read (X).
- 3- If ($X \geq 0$) Then Write ("Pos") and Go To Step 5.
- 4- Write ("Nag").
- 5- End.

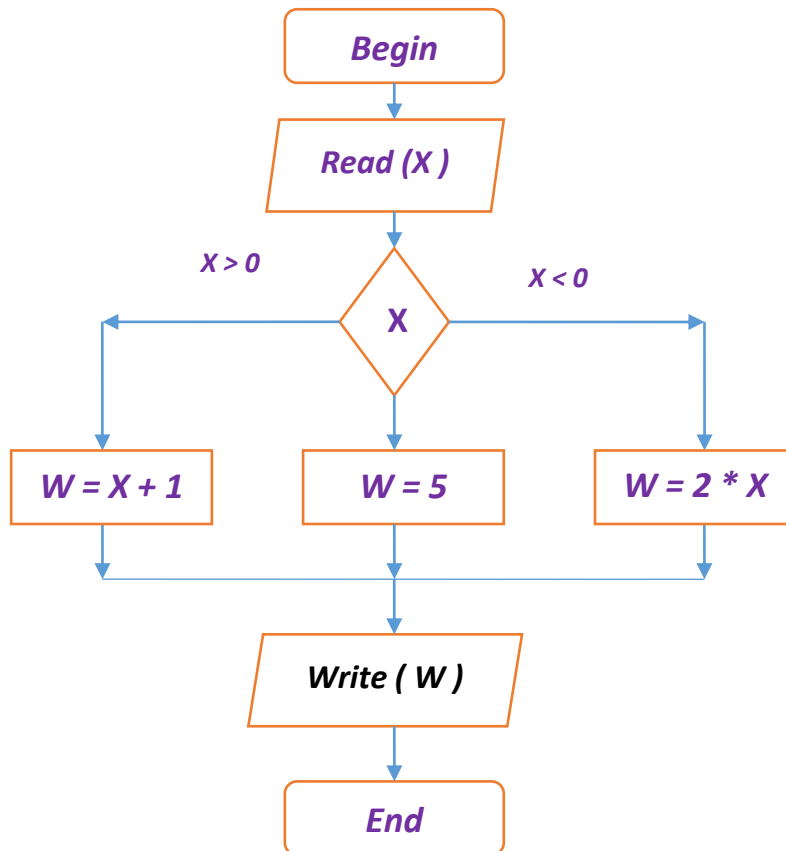


Example 3.

Write algorithm and draw the flow chart to find the value of (w) value so:

$$W = \left\{ \begin{array}{lll} X+1 & \text{if} & X > 0 \\ 5 & \text{if} & X = 0 \\ 2X+1 & \text{if} & X < 0 \end{array} \right\}$$

- 1- Begin.
- 2- Read (X).
- 3- If (X>0) Then W=X+1 And Go To Step 6.
- 4- If (X=0) Then W=5 and Go To Step6.
- 5- If (X<0) Then W=2*X+1.
- 6- Write (W).
- 7- End.



Chapter 2

**The basics of the C#
language**

Introduction.

C# is a programming language which it using for writing a computer programs like, programming the server side code in a web application developed by ASP.NET.

As well as we can using this language to programming data base and applications, finally we can use this language to programming operating systems.

It is like java. C# is intended to be the premier language for writing NGWS (Next Generation Windows Services) applications in the enterprise computing space.

The programming language C# derives from C and C++; however, it is modern, simple, support for object-oriented, and type-safe.

Contributing to the ease of usage is the elimination for certain features of C++, no more macros, no templates, and no multiple inheritances.

The aforementioned features create more problems than they provide benefit especially for enterprise developers.

New features for added convenience are strict type safety, versioning, garbage collection, and many more.

All these features are targeted at developing component-oriented software.

Although you don't have the sheer power of C++, you become more productive faster.

C# Features.

The C# programming language has some properties, like:

1. It is a case of sensitive language.

2. All the built in methods of classes must be written in Capital letter case, for example.

Console.ReadLine ()

3. The extension of a C# program must be (.cs).

4. It is using for designing both kind of application (web based and windows based).

Tokens.

The smallest individual units of a program are known as tokens. They contain the following elements:

1. Identifiers.

It means the names of the variables, functions and arrays ...etc. the C# language has the following rules for naming:

1. It allows to use the letters and numbers and () in the naming.
2. It doesn't allow starting the name with a number.
3. It is a case of sensitive language (it is difference between the small letter and capital letter).
4. It doesn't allow using the reserve word to name identifiers.

The following example shows the correct and incorrect identifier naming.

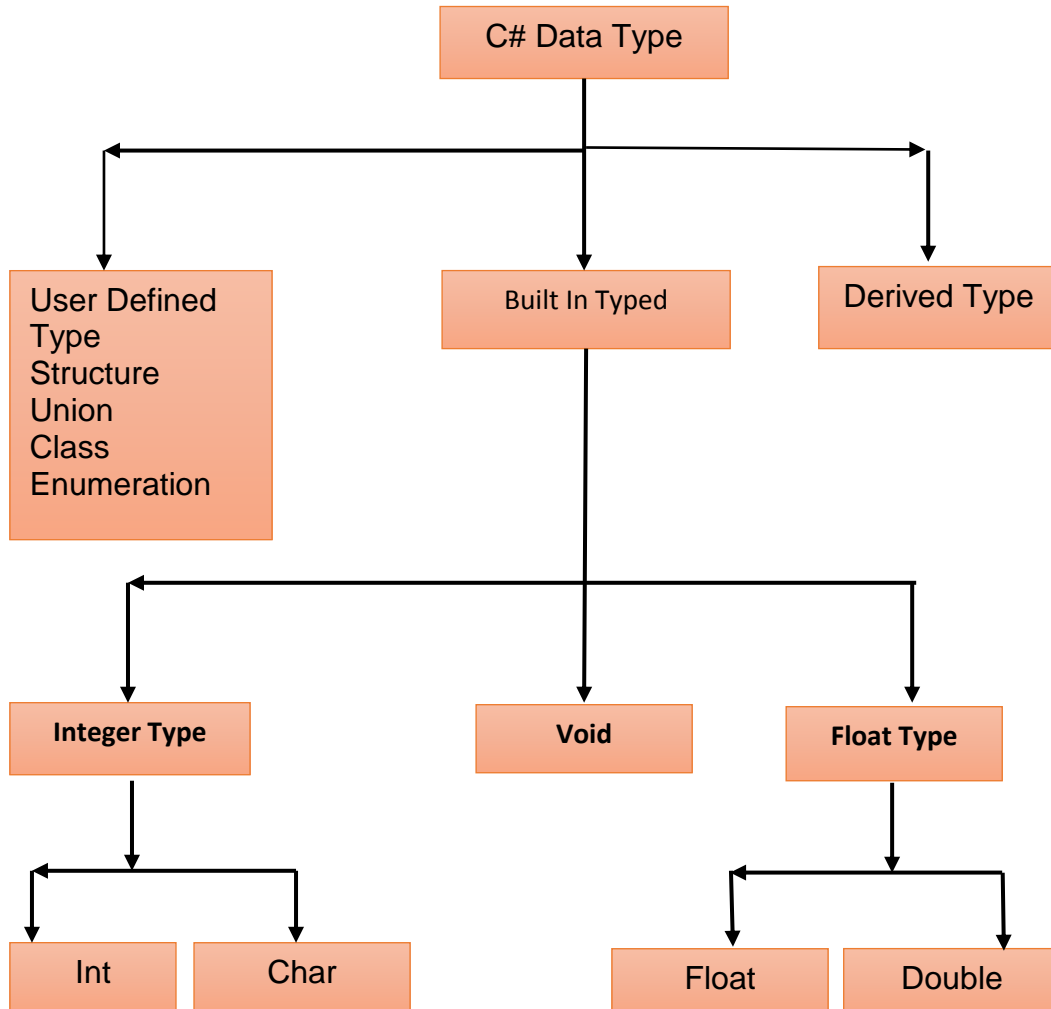
Correct	Incorrect
Count	1count
Test23	Hi\$thier
High _ balance	High-balance

Not.

The C# language don't put any constraint at the identifier name length, therefore all the letters of the name are important.

General Data Type.

The following figure shown the basic data types in C# language.



Identifiers types.

There are two types of the identifiers using in C# language, they are.

A. Variables.

A variable is nothing but a name given to a storage area that our programs can manipulate.

Each variable in C# has a specific type, which determines the size and layout of the variable's memory, the range of values that can be stored

within that memory, and the set of operations that can be applied to the variable.

C# language also allows defining other value types of variable like enum and reference types of variables like class, which we will cover in subsequent chapters.

For this chapter, let us study only basic variable types.

C# Data Types.

There are three major categories of the data types in C#, they are:

- ✓ Value types
- ✓ Reference types
- ✓ Pointer types

A. Values Types.

Values type variables can be assigned a value directly.

They are derived from the class System.Value Type.

The value types directly contain data.

Some examples are int, char, float, which stores numbers, alphabets, and floating point numbers, respectively.

When you declare an int type, the system allocates memory to store the value.

The following table lists the available value types in C# 2010:

Type	Represents	Range	Default Value
bool	Boolean value	True or False	False
byte	8-bit unsigned integer	0 to 255	0
char	16-bit Unicode character	U +0000 to U +ffff	'\0'
decimal	128-bit precise decimal values with 28-29 significant digits	$(-7.9 \times 10^{28} \text{ to } 7.9 \times 10^{28}) / 10^0$ to 28	0.0M
double	64-bit double-precision floating point type	$(+/-)5.0 \times 10^{-324} \text{ to } (+/-)1.7 \times 10^{308}$	0.0D
float	32-bit single-precision floating point type	$-3.4 \times 10^{38} \text{ to } + 3.4 \times 10^{38}$	0.0F
int	32-bit signed integer type	-2,147,483,648 to 2,147,483,647	0
long	64-bit signed integer type	-923,372,036,854,775,808 to 9,223,372,036,854,775,807	0L
sbyte	8-bit signed integer type	-128 to 127	0
short	16-bit signed integer type	-32,768 to 32,767	0
uint	32-bit unsigned integer type	0 to 4,294,967,295	0
ulong	64-bit unsigned integer type	0 to 18,446,744,073,709,551,615	0
ushort	16-bit unsigned integer type	0 to 65,535	0

To get the exact size of a type or a variable on a particular platform, you can use the `sizeof` method.

The expression `sizeof (type)` yields the storage size of the object or type in bytes. The following is an example to get the size of (int) type in any machine:

```
Namespace DataTypeApplication
{
Class Program
{
Static void Main (string [] args)
{
Console.WriteLine ("Size of int: {0}",sizeof (int));
Console.ReadLine ();
}
}
}
```

When the above code is compiled and executed, it produces the following result:

Size of int: 4

B. Reference Types.

The reference types do not contain the actual data stored in a variable, but they contain a reference to the variables.

In other words, they refer to a memory location.

Using more than one variable, the reference types can refer to a memory location.

If the data in the memory location is changed by one of the variables, the other variable automatically reflects this change in value.

Example of built-in reference types are: object, dynamic and string. It is illustrate in below:

C. Object Type.

The Object Type is the ultimate base class for all data types in C# Common Type System (CTS).

Object is an alias for System.Object class.

So object types can be assigned values of any other types, value types, and reference types, predefined or when a value type is converted to object type, it is called boxing and on the other hand, when an object type is converted to a value type, it is called unboxing.

```
object obj;  
obj =100;    // this is boxing
```

D. Dynamic Type.

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

Syntax for declaring a dynamic type is:

```
dynamic <variable_name> = value;
```

Example.

```
dynamic d =20;
```

Dynamic types are similar to object types except that type checking for object type variables takes place at compile time, whereas that for the dynamic type variables take place at run time.

E. String Type

The String Type allows you to assign any string values to a variable.

It is an alias for the System.String class.

It is derived from object type.

The value for a string type can be assigned using string literals in two forms: quoted and @quoted.

Example.

```
String str ="Tutorials Point";
```

A @quoted string literal looks like:

```
@ "Tutorials Point";
```

The user-defined reference types are: class, interface, or delegate. We will discuss these types in later chapter.

F. Pointer Types

Pointer type variables store the memory address of another type. Pointers in C# have the same capabilities as in C or C++.

Syntax for declaring a pointer type is:

```
Type* identifier;
```

Example,

```
char* cptr;
```

```
int* iptr;
```

C# Type Conversion.

Type conversion is basically a type casting or converting one type of data to another type. In C#, type casting has two forms:

- ✓ Implicit type conversion: these conversions are performed by C# in a type-safe manner. Examples are conversions from smaller to larger integral types and conversions from derived classes to base classes.

- ✓ Explicit type conversion: these conversions are done explicitly by users using the pre-defined functions. Explicit conversions require a cast operator.

The following example shows an explicit type conversion:

```
namespace TypeConversionApplication
{
    class ExplicitConversion
    {
        static void Main(string[] args)
        {
            double d = 5673.74;
            int i;
            // cast double to int.
            i = (int)d;
            Console.WriteLine (i);
            Console.ReadKey ();
        } } }
```

When the above code is compiled and executed, it produces the following result:

5673

C# Type Conversion Methods.

C# provides the following built-in type conversion methods:

No.	Methods & Description
1	ToBoolean Converts a type to a Boolean value, where possible.
2	ToByte Converts a type to a byte.
3	ToChar Converts a type to a single Unicode character, where possible.
4	ToDateTime Converts a type (integer or string type) to date-time structures.

5	ToDecimal Converts a floating point or integer type to a decimal type.
6	ToDouble Converts a type to a double type.
7	ToInt16 Converts a type to a 16-bit integer.
8	ToInt32 Converts a type to a 32-bit integer.
9	ToInt64 Converts a type to a 64-bit integer.
10	ToSbyte Converts a type to a signed byte type.
11	ToSingle Converts a type to a small floating point number.
12	ToString Converts a type to a string.
13	ToType Converts a type to a specified type.
14	ToUInt16 Converts a type to an unsigned int type.
15	ToUInt32 Converts a type to an unsigned long type.
16	ToUInt64 Converts a type to an unsigned big integer.

Variables Definition in C#.

The general form of variables definition in C# is shown in below.

```
<data_type> <variable_list>;
```

Here, the data type must be a valid C# data type including char, int, float, double, or any user-defined data type, etc., and variables list may consist of one or more identifier names separated by commas.

Some valid variables definitions are shown here:

```
int i;
```



```
char ch;  
float salary;  
double d;  
int x , y;
```

Note.

Note that we can initialize a variables at the definition time as:

```
Int i = 100;  
char ch = 'a';
```

Variables Initialization in C#.

Variables are initialized (assigned a value) with an equal sign followed by a constant expression. The general form of initialization is:

variable_name = value;

Variables can be initialized (assigned an initial value) in their declaration. The initializer consists of an equal sign followed by a constant expression as:

<data_type> <variable_name> = value;

Some examples are:

```
int d = 3;                /*initializing d*/  
float z = 22.4;          /*initializes z.*/  
double pi = 3.14159;     /*declares an approximation of pi.*/  
char x = 'x';           /* the variable x has the value 'x'. */
```

It is a good programming practice to initialize variables properly, otherwise program would produce unexpected result.

Try the following example which makes use of various types of variables:

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

When the above code is compiled and executed, it produces the following result:

C=30

Accepting Values from User.

The Console class in the System namespace provides a function ReadLine () for accepting input from the user and store it into a variable. For example.

```
int num;
num = Convert.ToInt32(Console.ReadLine());
```

Note.

The function Convert.ToInt32 () converts the data entered by the user to int data type, because Console.ReadLine () accepts the data in string format.

Expression in C#:

It is a syntax expression that has two sides left side and right side, so the left side must has one variable and doesn't allow to be constant or functional, while the right one may be constant or a mix from variables and operations.

Example1.

```
a = 20;
```

In the above example the left side is variable and the right one is constant.

Example2.

```
x = y + z;
```

In the above example the left side is variable and the right on is a mix of variables and operations.

Example3.

```
Ch = Console.ReadLine ( );
```

In the above example the left side is variable and the right one is built in function.

Note.

In C# language we can sign many variables to the same value in one statement, for example.

```
x = y = z = 20;
```

Note.

The following expression is not a valid statement and would generate compile-time error because we equal constant value to other constant value.

```
10 = 20;
```

B. Constants.

They are fixed values which didn't change during execution of the program.

The constants refer to fixed values that the program may not alter during its execution.

These 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. There are also enumeration constants as well.

The constants are treated just like regular variables except that their values cannot be modified after their definition.

Defining Constants.

Constants are defined by using the const keyword.

Syntax for defining a constant is:

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

The following program demonstrates defining and using a constant in your program:

```
using System;
namespace Declaring Constants
{
    class Program
    {
        static void Main(string[] args)
        {
            const double pi = 3.14159; // constant declaration
        }
    }
}
```

```
double r;  
Console.WriteLine ("Enter Radius: ");  
r = Convert.ToDouble (Console.ReadLine ());  
double areaCircle = pi * r * r;  
Console.WriteLine ("Radius: {0}, Area: {1}", r, areaCircle);  
Console.ReadLine ();  
}  
}  
}
```

Note.

When the above code is compiled and executed, it produces the following result:

Enter Radius:

3

Radius: 3, Area: 28.27431

2. Keywords.

They are reserve words for language, which implements any particular meaning in your program, and they can't use by the user as identifiers in program.

So every language has some reserve words. In C# language we have many words like.

(Using – system – namespace – console – for – while – ifetc.)

3. Operators.

An operator is a symbol that tells the compiler to perform specific mathematical or logical manipulations.

C# is rich in built-of operators and provides the following type of operators:

- ✓ Arithmetic Operators.
- ✓ Relational Operators.
- ✓ Logical Operators.
- ✓ Bitwise Operators.
- ✓ Assignment Operators.
- ✓ Misc. Operators.

This tutorial will explain the arithmetic, relational, logical, bitwise, assignment and other operators one by one.

1. Arithmetic Operators

The following table shows all the arithmetic operators supported by C#. Assume variable A holds 10 and variable B holds 20 then:

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

2. Relational Operators.

The following table shows all the relational operators supported by C#. Assume variable A holds 10 and variable B holds 20, then:

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

3. Logical Operators.

The following table shows all the logical operators supported by C#. Assume variable A holds Boolean value= True and variable B holds Boolean value= False, then:

Operator	Description	Example
&&	Called Logical AND operator. If both the operands are non-zero then condition becomes	(A && B) is false.

	true.	
	Called Logical OR Operator. If any of the two operands is none zero then condition becomes true.	(A B) is true.
!	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.	!(A && B) is true.

4. Bitwise Operators.

Bitwise operators works on bits and performs bit by bit operation. The truth table for &, | and ^ are as follows:

P	q	p & q	p q	p ^ q
0	0	0	0	0
0	1	0	1	1
1	1	1	1	0
1	0	0	1	1

Example.

Assume that, if A = 60; and B = 13; now in binary format they will be as follows:

A = 0011 1100

B = 0000 1101

A & B = 0000 1100

A | B = 0011 1101

A ^ B = 0011 0001

~ A = 1100 0011

The Bitwise operators supported by C# are listed in the following table. Assume variable A holds 60 and variable B holds 13 then:

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

5. Assignment Operators.

The following assignments are operators supported by C#:

Operator	Description	Example
=	Simple assignment operator, Assigns values from right side operands to left side operand	C = A + B will assign value of A + B into C
+=	Add AND assignment operator, It adds right operand to the left operand and assign the result to left operand	C += A is equivalent to C = C + A

<code>-=</code>	Subtract AND assignment operator, It subtracts right operand from the left operand and assign the result to left operand	$C -= A$ is equivalent to $C = C - A$
<code>*=</code>	Multiply AND assignment operator, It multiplies right operand with the left operand and assign the result to left operand	$C *= A$ is equivalent to $C = C * A$
<code>/=</code>	Divide AND assignment operator, It divides left operand with the right operand and assign the result to left operand	$C /= A$ is equivalent to $C = C / A$
<code>%=</code>	Modulus AND assignment operator, It takes modulus using two operands and assign the result to left operand	$C %= A$ is equivalent to $C = C \% A$
<code><<=</code>	Left shift AND assignment operator	$C <<= 2$ is same as $C = C << 2$
<code>>>=</code>	Right shift AND assignment operator	$C >>= 2$ is same as $C = C >> 2$
<code>&=</code>	Bitwise AND assignment operator	$C \&= 2$ is same as $C = C \& 2$
<code>^=</code>	bitwise exclusive OR and assignment operator	$C \wedge= 2$ is same as $C = C \wedge 2$
<code> =</code>	bitwise inclusive OR and assignment operator	$C = 2$ is same as $C = C 2$

6. Misc Operators.

There are few other important operators including size of, type of and (? :) supported by C#. they are:

Operator	Description	Example
<code>sizeof()</code>	Returns the size of a data type.	<code>sizeof (int)</code> , will return 4.
<code>typeof()</code>	Returns the type of a class.	<code>typeof (StreamReader)</code> ;
<code>&</code>	Returns the address of a variable.	<code>&a</code> ; will give actual address of the variable.

*	Pointer to a variable.	*a; will pointer to a variable.
? :	Conditional Expression	If Condition is true? Then value X : Otherwise value Y
is	Determines whether an object is of a certain type.	If(Ford is Car) // checks if Ford is an object of the Car class.
as	Cast without raising an exception if the cast fails.	Object obj = new StreamReader("Hello"); StreamReader r = obj as StreamReader;

Operators Precedence in C#.

Operator's precedence determines the grouping of terms in an expression. This affects how an expression is evaluated.

Certain operators have higher precedence than others; for example, the multiplication operator has higher precedence than the addition operator:

For example $x = 7 + 3 * 2$; here, x is assigned 13, not 20 because operator * has higher precedence than +, so it firstly gets multiplied with $3*2$ and then adds into 7.

Here, operators with the highest precedence appear at the top of the table, those with the lowest appear at the bottom.

Within an expression, higher precedence operators will be evaluated first.

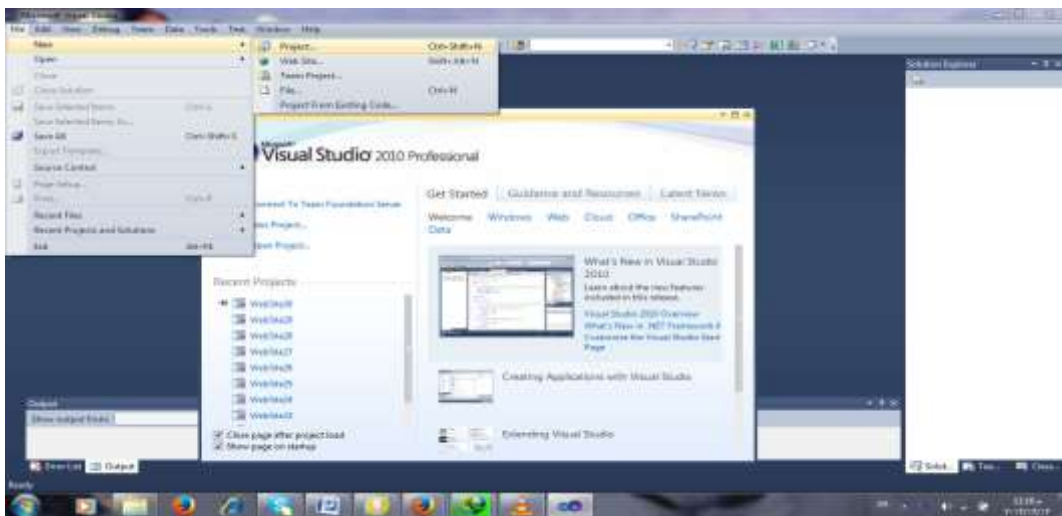
Category	Operator	Associativity
Postfix	() [] -> . ++ --	Left to right
Unary	+ - ! ~ ++ -- (type)* & size of	Right to left
Multiplicative	* / % s	Left to right
Additive	+ -	Left to right

Shift	<<>>	Left to right
Relational	<<= >>=	Left to right
Equality	== !=	Left to right
Bitwise AND	&	Left to right
Bitwise XOR	^	Left to right
Bitwise OR		Left to right
Logical AND	&&	Left to right
Logical OR		Left to right
Conditional	?:	Right to left
Assignment	= += -= *= /= %= >>= <<= &= ^= =	Right to left
Comma	,	Left to right

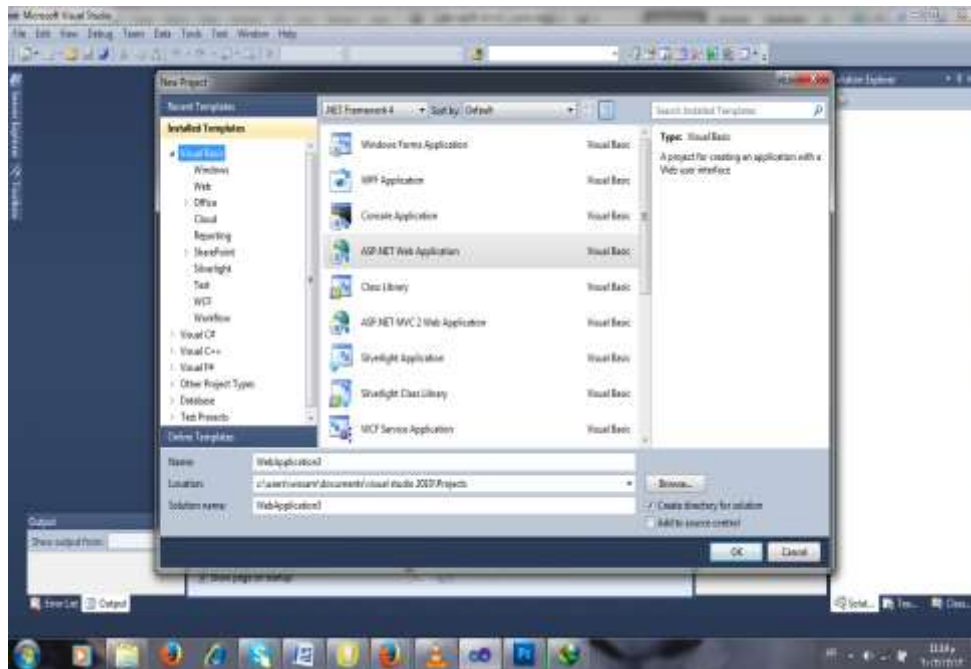
How to open C# window.

We can open the C# window in .Net program by doing the following steps:

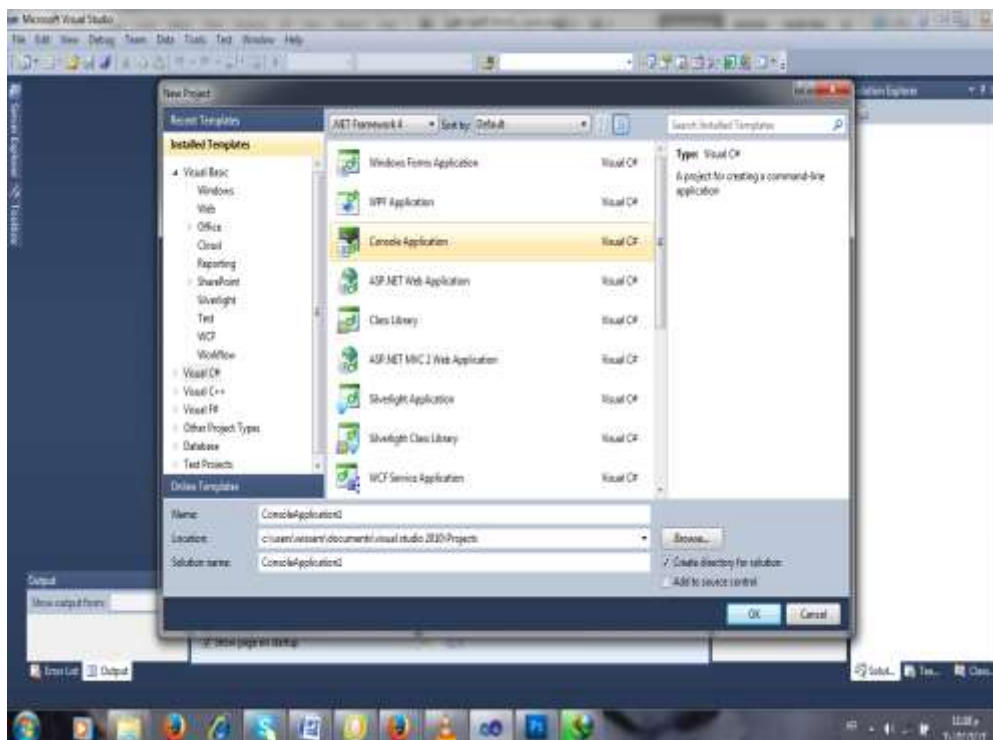
- 1- Go to the (File) menu then (New) after that (Project), it is shown in below.



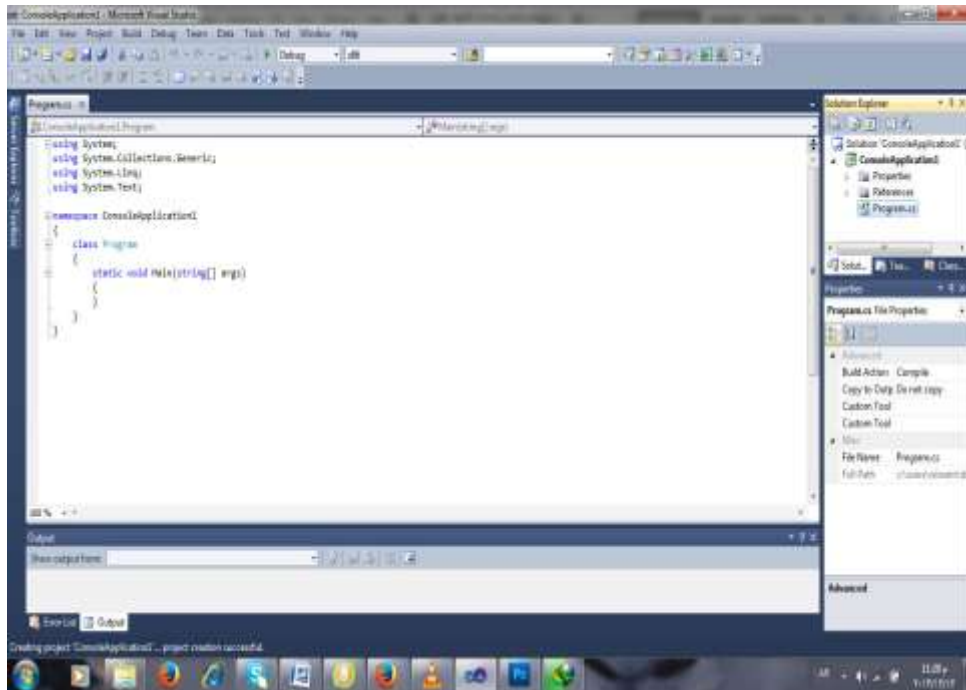
- 2- After pressing at (Project) it will appear the following window.



- 3- In the left of the above window, go to the (Resent Template) and press at the (Visual C#), it will appear the following window.



- 4- Change the name of program and the location of saving then Press (Console Application), it will appear the following window.



5- In the above window, we can write the C# code.

Literals in C#.

1. Integer Literals.

An integer literal can be a decimal, octal, or hexadecimal constant.

A prefix specifies the base or radix: 0x or 0X for hexadecimal, 0 for octal, and no prefix is required for decimal.

An integer literal can also have a suffix that is a combination of U and L, for unsigned and long, respectively.

The suffix can be uppercase or lowercase and can be in any order. Here are some examples of integer literals:

```
212      /* Legal */
215u     /* Legal */
0xFeeL   /* Legal */
078      /* Illegal: 8 is not an octal digit */
032UU    /* Illegal: cannot repeat a suffix */
```

These are other examples of various types of Integer literals:

```
85          /* decimal */
0213       /* octal */
0x4b       /* hexadecimal */
30         /* int */
30u        /* unsigned int */
30l        /* long */
30ul       /* unsigned long */
```

2. Floating-point Literals.

A floating-point literal has an integer part, a decimal point, a fractional part, and an exponent part. You can represent floating point literals either in decimal form or exponential form. Here are some examples of floating-point literals:

```
3.14159     /* Legal */
314159E-5L  /* Legal */
510E        /* Illegal: incomplete exponent */
210f        /* Illegal: no decimal or exponent */
.e55        /* Illegal: missing integer or fraction */
```

While representing using decimal form, you must include the decimal point, the exponent, or both and while representing using exponential form you must include the integer part, the fractional part, or both. The signed exponent is introduced by e or E.

3. Character Constants.

Character literals are enclosed in single quotes, e.g., 'x' and can be stored in a simple variable of char type.

A character literal can be a plain character (e.g., 'x'), an escape sequence (e.g., '\t'), or a universal character (e.g., '\u02C0').

There are certain characters in C# when they are preceded by a backslash they will have special meaning and they are using to represent like newline (\n) or tab (\t). Here, you have a list of some of such escape sequence codes:

Escape sequence	Meaning
\\	\ character
\'	' character
\"	" character
\?	? character
\a	Alert or bell
\b	Backspace
\f	Form feed
\n	Newline
\r	Carriage return
\t	Horizontal tab
\v	Vertical tab
\ooo	Octal number of one to three digits
\xhh . . .	Hexadecimal number of one or more digits

Example.

This is an example to show few escape sequence characters:

```

Namespace EscapeChar
{
    Class Program
    {
        Static void Main (string [] args)
        {
            Console.WriteLine ("Hello\tWorld\n\n");
            Console.ReadLine ();
        }
    }
}

```


When the above code is compiled and executed, it produces the following result:

Hello World

4. String Literals

String literals or constants are enclosed in double quotes "" or with @"". A string contains characters that are similar to character literals: plain characters, escape sequences, and universal characters.

You can break a long line into multiple lines using string literals and separating the parts using whitespaces.

Here are some examples of string literals. All the three forms are identical strings.

```
"hello, dear"
```

```
"hello, \  
dear"
```

```
"hello, " "d" "ear"
```

```
@"hello dear"
```

C# Program Structure.

A C# program basically consists of seven major parts, it is illustrated in the following:

- ✓ Namespace declaration.
- ✓ A class.
- ✓ Class methods.
- ✓ Class attributes.
- ✓ A Main method.
- ✓ Statements & Expressions.
- ✓ Comments.

Let us look at a simple code that would print the words "Hello World":

Example.

```
Using System;
Namespace HelloWorldApplication
{
Class HelloWorld
{
Static void Main (string [] args)
{
/* my first program in C# */
Console.WriteLine("Hello World");
Console.ReadKey();
}
}
}
```

When the above code is compiled and executed, it produces the following result:

Hello World

Let us look at various parts of the above program:

- ✓ The first line of the program using System: the using of keyword is to include the System namespace in the program.

A program generally has multiple using statements.

- ✓ The next line has the namespace declaration: A namespace is a collection of classes.

The *HelloWorldApplication* namespace contains the class *HelloWorld*.

- ✓ The next line has a class declaration: the class *HelloWorld* contains the data and method definitions that your program uses.

Classes generally would contain more than one method.

Methods define the behavior of the class.

However, the *HelloWorld* class has only one main method.

- ✓ The next line defines the main method: which is the entry point for all C# programs. The Main method states what the class will do when executed
- ✓ The next line */*...*/* will be ignored by the compiler and it has been put to add additional comments in the program.
- ✓ The main method specifies its behavior with the statement `Console.WriteLine("Hello World");`

WriteLine is a method of the *Console* class defined in the *System* namespace. This statement causes the message "Hello, World!" to be displayed on the screen.

- ✓ The last line `Console.ReadKey ();` it is for the VS.NET Users.

This makes the program wait for a key press and it prevents the screen from running and closing quickly when the program is launched from Visual Studio .NET.

Compile & Execute a C# Program.

If you are using Visual Studio.Net for compiling and executing C# program, take the following steps:

- Start Visual Studio.
- On the menu bar, choose File, New, and Project.
- Choose Visual C# from templates, and then choose Windows.
- Choose Console Application.
- Specify a name for your project, and then choose the OK button.
- The new project appears in Solution Explorer.
- Write code in the Code Editor.
- Click the Run button or the F5 key to run the project. A Command Prompt window appears that contains the line Hello World.

You can compile a C# program by using the command-line instead of the Visual Studio IDE:

- Open a text editor and add the above-mentioned code.
- Save the file as helloworld.cs
- Open the command prompt tool and go to the directory where you saved the file.
- Type cshelloworld.cs and press enter to compile your code.
- If there are no errors in your code, the command prompt will take you to the next line and would generate helloworld.exe executable file.
- Next, type helloworld to execute your program.
- You will be able to see "Hello World" printed on the screen.

Read Statement in the C#.

To read any value from the keyboard in the C#, we can use (Read_Statement). It is follow to Console class, the general form of this statement is shown in below.

```
variable_name = variable_type (Console.Read ( ));
```

- Variable type is optional, i.e. when we use int values, and then we will use the following form.

```
Int a;  
a = int.Parse (Console.Read ( ));
```

When we use float values, then we will use the following form.

```
Float a;  
a = float.Parse (Console.Read ( ));
```

When we use character values, then we will use the following form.

```
Char ch;  
Ch = Console.Read ( );
```

○ There are two types of the Read statements, they are:

1. **Read_Statement:** in this type, when we enter the values to the program we should write all values in the same line, the general form of this type is.

```
Variable_name = variable_type (Console.Read ( ));
```

Example.

```
Int a;
```

```
a = int.Parse (Console.Read ( ));
```

2. **ReadLine_Statement:** when we use this type we should insert every value to the program at one line, the general form of using this statement is.

```
variable_name = variable_type (Console.ReadLine ( ));
```

Example.

```
Int a;
```

```
a = int.Parse (Console.ReadLine ( ));
```

Write statement in C# language.

To write any value at the computer screen in the C# language, we can use (Write Statement).

It is followed by the Console class, and the general form of this statement is shown in below.

```
Console. Write (variable_name);
```

Note1.

If we want to write any text at the computer screen, we should put the text between double quotation (" "), for example.

Console. Write ("Karbala University");

Note2.

If we want to write any variable at the computer screen, we don't need to put this variable between double quotation (" "), for example.

Console. Write (a);

Note3.

If we want to combine the variable with the text, we will use the following form.

Console. Write ("Text" + variable);

Example.

Console. Write (" Sum= " + s);

Note4.

If we want to print out two variables or more than by one write statement, we will use the following form.

Console. Write (variable1, variable2);

Example.

Console. Write (x, y);

Write Statement Types.

There are two types of write statement, they are:

1- **Write_Statement**: when we use this statement, then all values that written at the computer screen will write at the same line, the general form of using this statement is:

```
Console. Write (Variable_name);
```

Example.

```
Int a = 5;  
Int b = 6;  
Console. Write (a);  
Console. Write (b);  
5 6
```

2- **WriteLine_Statement**: when we use this statement, then all values that written at the computer screen, will write every value at one line, the general form of using this statement is:

```
Console.WriteLine (Variable_name);
```

Example 1.

```
Int a=5;  
Int b=6;  
Console.WriteLine (a);  
Console.WriteLine (b);  
5  
6
```

Example 2.

Write program in C# language to read student marks in physics, chemistry & math and calculate the total and average of the student?

```
Using System;
```

```
Using System.Collections.Generic;
```

```
Using System.Linq;
```

```
Using System.Text;
```

```
Namespace ConsoleApplication1
```

```
{
```

```
Class Program
```

```
{
```

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

```
{
```

```
Int a, b, c, s;
```

```
double av;
```

```
Console.WriteLine ("Enter the Mark of the Physic");
```

```
    a= int.Parse (Console.ReadLine ());
```

```
Console.WriteLine ("Enter the Mark of the Chemistry");
```

```
    b = int.Parse (Console.ReadLine ());
```

```
Console.WriteLine ("Enter the Mark of the Mathematic");
```

```
    c = int.Parse (Console.ReadLine ());
```

```
    s = a + b + c;
```

```
    av = (a + b + c) / 3;
```

```
Console.WriteLine ("the sum of this marks is: " + s);
```

```
Console.WriteLine ("the average of this marks is: " + av);
```

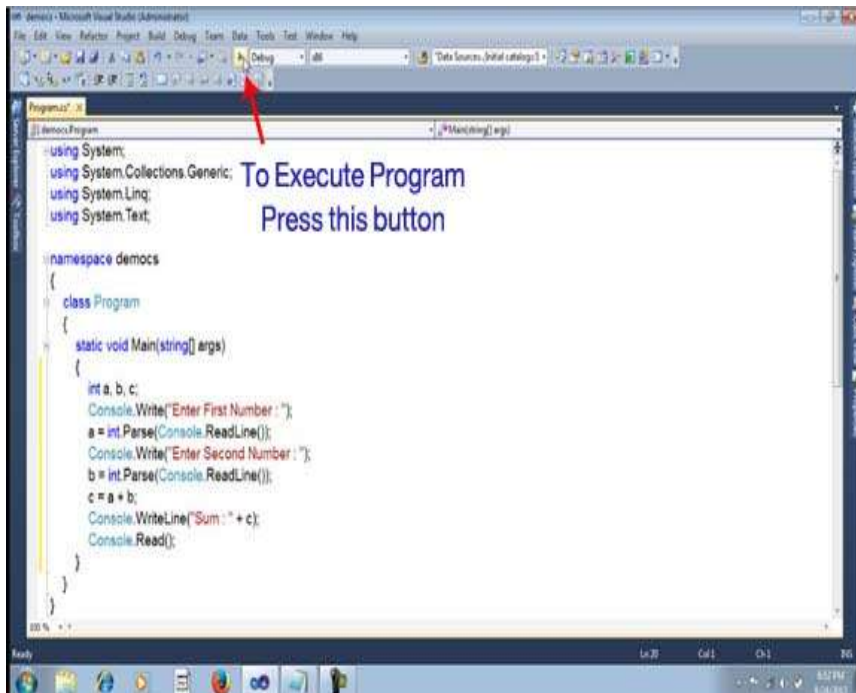
```
Console.Read ();
```

```
    }
```

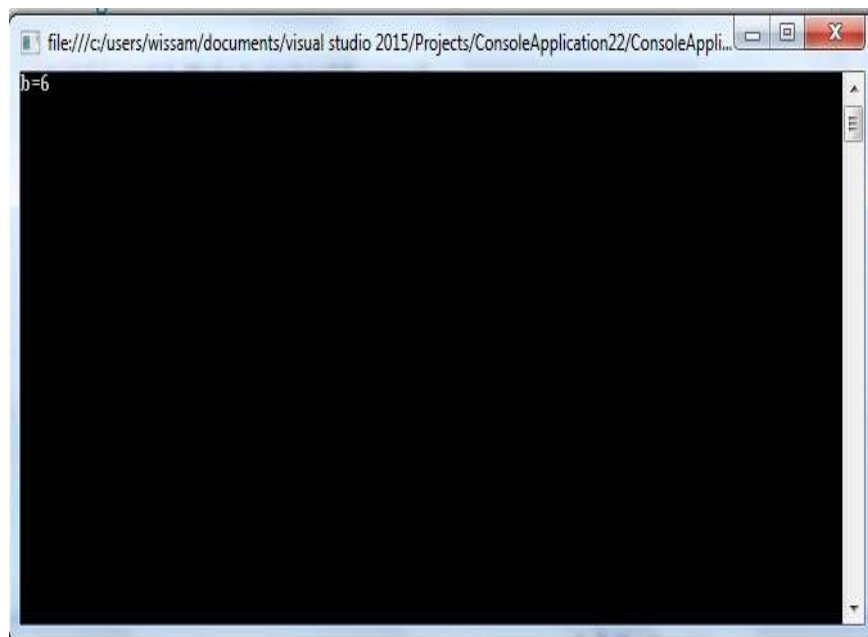
```
    }
```

```
}
```


To execute this program we should go to the (Start Debugging) from Tools bar or press (F5) from keyboard, like it is shown in below.



- The end statement in the above program was (Console.Read ());. It is using to showing the execution screen.



Example 3.

Write program in C# language to read length and breadth and calculate the area of rectangle? So we can calculate the rectangle area from the following equation.

$$\text{Area} = L * B$$

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApplication1
{
    Class Program
    {
        Static void Main (string[] args)
        {
            int a, b, area;
            Console.WriteLine ("Enter the Length of Rectangle");
            a= int.Parse (Console.ReadLine ());
            Console.WriteLine ("Enter the Width of Rectangle");
            b = int.Parse (Console.ReadLine ());
            area = a * b ;
            Console.WriteLine ("the area of the rectangle is: " + area);
            Console.Read ();
        }
    }
}
```

Conditional Statements.

They are using for making the decision within your program and sometimes move the execution to other place within program.

There are two types of conditional statement in C# language they are:

1. If statement.
2. Switch statement.

1. IF Statement.

This statement is using in C# language to execute statement or several statements when materialize specific condition or many conditions are written by programmer.

We can use different variations of if statement, so there are three types of this statement, these are.

- a) If else statement.
- b) If else if ladder statement.
- c) Nested if statement.

A. IF_Else Statement.

If we have a condition & there are two aspects of this condition, first for True and second for False, then we can use the type of (If-Else) statement.

The general form of this statement is.

```
If (condition)
{
    True statement Oblock;
}
Else
{
    False statement block;
}
```

Example1.

Write program in C# language to check any given number is (even) or (odd)?

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

```
namespace ConsoleApplication1
{

Class Program
    {
Static void Main(string[] args)
    {
int a;
Console.WriteLine ("Please Insert Your Number");
a= int.Parse (Console.ReadLine ());
if ( a % 2==0)
Console.WriteLine ("The Number is Even");
else
Console.WriteLine ("The Number is odd");
Console.Read ();
}
}
}
```

Note1.

If we have just one execution statement we don't need to put this statement in brackets, for example:

```
If (x >=0)
Console.WriteLine ("The Number is Positive");
```

Note2.

If we have more than one execution statement, we will need to put this statement in brackets, for example.

```
If (x >=0)
{
    Console.WriteLine ("The Number is Positive");
    X = x + 23;
}
```

Example2.

Write a program in C# language to read age of a person & print whether he / she can vote or not?

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
class Program
{
static void Main(string[] args)
{
int a;
Console.WriteLine ("Please Insert the Age of Person");
a= int.Parse (Console.ReadLine ());
if ( a >= 18 )
Console.WriteLine ("He / She Can Vote");
else
Console.WriteLine ("He / She Can't Vote");
Console.Read ();
}
}
}
```

Home Work1.

Write a program in C# language to read student mark in oop subject and check whether the student is Pass or Fail?

Home work2.

Write a program in C# language to read specific number from keyboard and check whether the number is Positive or Negative?

B. IF_Else_IF Statement.

If we have more than one condition then we can use If–Else-If ladder statement. The general form of this statement is:

```
If (condition-1)
    Execution Statement-1;
Else if (condition-2)
    Execution Statement-2;
    _____
    _____
Else if (condition-n)
    Execution Statement-n;
Else
    Default statement;
```

Example 1.

Write a program in C# language to read 3 values from keyboard & print out the largest one at computer screen?

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
class Program
{
static void Main(string[] args)
{
int a,b,c;
Console.WriteLine ("Please Insert the First Number");
a= int.Parse (Console.ReadLine ());
Console.WriteLine ("Please Insert the Second Number");
```

```
b = int.Parse (Console.ReadLine ());
Console.WriteLine ("Please Insert the Third Number");
c = int.Parse (Console.ReadLine ());
if ((a > b) && (a>c))
Console.WriteLine ("The a Value is creator than b & c");
else if ((b > a) && (b > c))
Console.WriteLine ("The b Value is creator than a & c");
else
Console.WriteLine ("The c Value is creator than a & b");
Console.Read ();
}
}
}
```

Home Work.

Write a program in C# language to read a student marks in physics, chemistry & math then calculate and print their total average of these marks?

C. Nested IF Statement.

If we have in our program more than two conditions, then we will use nested if statement. The general form of this statement is:

```
If (condition)
{
    _____
    _____
    If (condition)
    {
        -----
        -----
    }
}
```

Example.

Write program in C# language to read 3 values from keyboard & print out the largest one at computer screen?

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
class Program
    {
static void Main(string[] args)
    {
int a,b,c;
Console.WriteLine ("Please Insert the First Number");
    a= int.Parse (Console.ReadLine ());
Console.WriteLine ("Please Insert the Second Number");
    b = int.Parse (Console.ReadLine ());
Console.WriteLine ("Please Insert the Third Number");
    c = int.Parse (Console.ReadLine ());
if ((a > b) && (a>c))
    {
Console.WriteLine ("The a Value is creator than b & c");
if ((b > c) && (b > c))
    {
Console.WriteLine ("The b Value is creator than a & c");
if ((c > a) && (c > b))
    {
Console.WriteLine ("The c Value is creator than a & b");
    }
    }
    }
    }
}
```



```
Console.Read ();  
    }  
}  
}
```

2. Switch Case Statement.

It is also a decision making statement.

It works faster than if statement and more elegance, but the disadvantage is that it doesn't support logical and relational operators.

The general form of this statement is:

```
Switch (expression)  
{  
    Case value-1:  
        Statement-1;  
        Break;  
    Case value-2:  
        Statement-2;  
        Break;  
  
    _____  
    _____  
    Case value-n:  
        Statement-n;  
        Break;  
    Default:  
        Default statement;  
        Break;  
}
```

Example.

Write program in C# language to read two values & character, then perform operation depending on the reading character, it is shown in the following menu:

1. Addition operation, if the character is (+).
2. Subtraction operation, if the character is (-).
3. Multiplication operation, if the character is (*).
4. Division operation, if the character is (/).

If the user doesn't enter any choice of the above fourth choices, print out the following sentence ("Please Insert Truth symbol")?

Solution.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            int a, b;
            float c;
            char ch;
            Console.WriteLine ("Please Insert the First Number");
            a = int.Parse (Console.ReadLine ());
            Console.WriteLine ("Please Insert the Second Number");
            b = int.Parse (Console.ReadLine ());
            Console.WriteLine ("Please Insert the operation");
```

```
        ch = Console.ReadKey ().KeyChar ;
Console.WriteLine ();
Switch (ch)
    {
case'+':
        c = a + b;
Console.WriteLine (" C=" + c);
break;
case'-':
        c = a - b;
Console.WriteLine (" C=" + c);
break;
case'*':
        c = a * b;
Console.WriteLine (" C=" + c);
break;
case'/':
        c = a / b;
Console.WriteLine (" C=" + c);
break;
default:
Console.WriteLine ("Please Insert Truth symbol ");
break;
    }
Console.Read ();
    } } }
```

Loop Statements.

Loop statements are using for repeating a block of program during finite or infinite times.

It is also known as iterative or repetitive statement. It could be classified into two categories:

- 1- Entry Controlled Loop.
- 2- Exit Controlled Loop.

1- Entry Controlled Loop.

This type of loop checks the condition, and if the condition is satisfied then executes the body of the loop. “For Loop” & “While Loop” are types of this category.

2- Exit Controlled Loop.

This type of loop executes at least one time either condition is satisfied or not. “Do While Loop” is type of this category.

1- For Loop.

It is an entry controlled loop, it is executed single statement or several statement at many times, and the general form of this loop will be as following:

```
For (initialization value; test condition; increment/decrement)
{
    Body of the loop;
}
```

Example 1.

Write program in C# language to print out the values from 1 to 10 at the computer screen, by using For Statement?

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApplication2
```

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

Example 2.

Write program in C# language to print the list of even values between (1) to (50), by using For Statement?

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

```
namespace ConsoleApplication2
{
class Program
{
static void Main(string[] args)
{
int x ;
for (x = 1 ; x <= 50 ; x++ )
{
if ( x % 2 ==0)
Console.WriteLine(x);
}
}
}
```

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

Example 3.

Write program in C# language to print out the values from 10 to 1, by using For Statement?

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;
```

```
namespace ConsoleApplication2  
{  
    class Program  
    {  
        static void Main(string[] args)  
        {  
            int x ;  
            for (x = 10 ; x >= 1 ; x-- )  
            {  
                Console.WriteLine(x);  
            }  
            Console.ReadLine ();  
        }  
    }  
}
```

Example 4.

Write program in C# language to print out the table of given number, by using For Statement?

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication2
{
    class Program
    {
        static void Main(string[] args)
        {
            int x, no ;
            Console.WriteLine (" Please Insert Your Number");
            no = int.Parse(Console.ReadLine());
            for (x = 1; x <= 10; x++)
            {
                Console.WriteLine(x * no);
            }
            Console.ReadLine ();
        }
    }
}
```

Example 5.

Write program in C# language to print out the following series:

```
1
0
1
0
.
.
```

n times

By using For Statement?

Home Work1.

Write program in C# language to print out factorial of any given number, by using For Statement?

Home Work 2.

Write program in C# language to calculate x to the power n, by using For Statement?

2- While Loop.

It is also an entry controlled loop to iterative statement or many statements many times specifically by a programmer, the general form of this loop will be:

```
While (condition)
{
    Body of the loop;
}
```

Or

```
Initialization;
While (condition)
{
    Body of the loop;
    Increment/decrement;
}
```


Example 1.

Write program in C# language to print out the numbers between (1...50) by using While Statement?

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

```
namespace ConsoleApplication4
{
class Program
    {
static void Main(string[] args)
    {
int i;
i=1;
While (i<=50)
{
Console.WriteLine (i);
i = i + 1;
Console.ReadLine ();
} } } }
```

Example 2.

Write program in C# language to print out the numbers between (50...1) by using While Statement?

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

```
namespace ConsoleApplication4
{
```

```
class Program
{
static void Main(string[] args)
{
int i;
i=50;
While (i>=1)
{
Console.WriteLine (i);
I = I - 1 ;
Console.ReadLine ();
} } } }
```

Example 3.

Write program in C# language to find the summation of the even numbers between (1...100) by using While Statement?

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApplication4
{
class Program
{
static void Main(string[] args)
{
int i;
int sum=0;
i=1;
While (i>=100)
{
sum = sum + i;
Console.WriteLine (i);
```

```
i=i+2;
Console.ReadLine ();
}
}
}
}
```

Example 4.

Write program in C# language to reverse the digits of a given number (123) and converting to (321) by using While Statement?

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApplication3
{
class Program
{
static void Main(string[] args)
{
int i, num, rev = 0, r;
Console.WriteLine (" Enter Your Number");
num = int.Parse(Console.ReadLine());
while (num > 0)
{
r = num % 10;
rev = (rev * 10) + r;
num = num / 10;
}
Console.WriteLine ("Reverse of Digit = " + rev);
Console.ReadLine ();
}
}
}
```

Example 5.

Write program in C# language to check whether the given number is palindrome or not? For example (121) is palindrome by using While Statement?

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication3
{
    class Program
    {
        static void Main(string[] args)
        {
            int i, num, rev = 0, r, num1;
            Console.WriteLine (" Enter Your Number");
            num = int.Parse(Console.ReadLine());
            num1 = num;
            while (num > 0)
            {
                r = num % 10;
                rev = (rev * 10) + r;
                num = num / 10;
            }
            Console.WriteLine ("num = " + num1);
            Console.WriteLine ("rev = " + rev);
            if (num1 == rev)
                Console.WriteLine ("the number is palindrome");
            else
                Console.WriteLine ("the number is not palindrome");
            Console.ReadLine ();
        }
    }
}
```

```
}  
}
```

Example 6.

Write program in C# language to print the Binary Equivalent of given integer value?

```
class demo  
{  
    public static void main(String [] args)  
    {  
        InputStreamReader in=new InputStreamReader (System.in);  
        BufferedReader br=new BufferedReader (in);  
        int bin[]=new int[50];  
        int num , l , loc = 0 , r;  
        Console.ReadLine ("Enter Number :");  
        num = Int.Parse(Console.ReadLine());  
        While (num > 0)  
        {  
            r=num%2;  
            bin [loc]=r;  
            num = num / 2;  
            loc ++;  
        }  
        Console.WriteLine ("The Binary Equivalent is :");  
        for (i=loc-1;i>=0;i--)  
        {  
            Console.WriteLine (bin[i]);  
        }  
    }  
}
```

Example 7.

Write program in C# language to print the sum & average of digits of a given number. For example?

123

Sum of digits = 6

Average of digits =2

By using While Statement?

Example 8.

Write program in C# language to print the largest digits of a given number. For example.

Input: 56976.

Output: 9.

By using While Statement?**Example 9.**

Write program in C# language to check whether a given number is Armstrong or not, for example.

$153=1^3+5^3+3^3=153$.

By using While Statement?

Break Statement.

It is using for terminating the flow of loop, it is always used along with the decision making statements.

The general form for using this statement is shown in the following example:

Class demo

```
{
    Public static void main (String [] args)
    {
        int i;
        for ( i=1 ; i<=10 ; i++ )
        {
            Console.WriteLine (i);
            If (i==5)
                Break;
        }
    }
}
```

Example 1.

Write program in C# language to check whether a given number is prime or not?

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication4
{
    class Program
    {
        static void Main(string[] args)
        {
            int i, num;
            bool isprime = false;
            Console.WriteLine (" Enter Your Number :");
            num = int.Parse(Console.ReadLine());
            for (i = 2; i <= num - 1; i++)
            {
```

```
if (num % i != 0)
    isprime = true;
else
    {
        isprime = false ;
    }
break;
    }
}
if (isprime == true)
    Console.WriteLine (" Number is Prime ");
else
    Console.WriteLine (" Number is Not Prime");
Console.ReadLine ();
    }
}
}
```

Example 2.

Write program in C# language to read three values & print the lcm of these values. For example.

a ,b,c

$m=a*b*c$

Solution.

```
for l = 1 to m
    if ( i %a == 0 and i % b == 0 and i %c == 0 ) then
        print i ;
        break ;
```


Continue Statement.

It is using for ignoring the sequence of loop.

The general form for using this statement is shown in the following example:

```
class demo
{
    public static void main(String args[])
    {
        int i;
        for ( I = 1 ; I <= 10 ; i++ )
        {
            If ( I >= 5 && I <= 8 )
                continue;
            Console.WriteLine (i);
        }
    }
}
```

3- Do While Loop Statement.

It is an exit controlled loop, means it will execute at least one time either condition is satisfied or not. The general form will be:

```
Do
{
    Body of the loop;
}
While (condition);
```

Example.

Write program in C# language to read two values from keyboard in every time and character, add these two numbers together , if the user enters

(1) the program will be terminated and print out ("Program Terminate") at the monitor screen?

Class demo

```
{
    Public static void main (String [] args)
    {
        Int a, b, c, ch;
        do
        {
            Console.WriteLine ("Enter Two Number: ");
            a = int.Parse (Console.ReadLine ());
            b = int.Parse (Console.ReadLine ());
            c = a + b;
            Console.WriteLine ("Sum: " + c);
            Console.Write ("Press 1 for Continue Your program: ");
            Ch = int.Parse (Console.ReadLine ());
        }
        While (ch == 1);
        Console.WriteLine ("Program Terminate");
    }
}
```

Nesting of Loop Statement.

If loop statement contains one or more than one another loop statement into its body then this term is known as nesting of loop statement.

Example 1.

Write program in C# language to print the multiply table from 2 to 10?

```
namespace ConsoleApplication4
{
    class Program
    {
```

```

static void Main(string[] args)
{
    Int i , j, res;
    For (i=2 ; i<=20 ; i++)
    {
        For (j=1; j<=10 ; j++)
        {
            res=i * j ;
            Console.WriteLine (res + "\t");
        }
        Console.ReadLine ();
    }
}
}
}

```

Example 2.

Write program in C# language to print the list of prime numbers between (1) to (100) by using for statement?

Class demo

```

{
    Static void Main (String [] args)
    {
        Int num i;
        Bool isprime=false;
        Console.Write ("Enter Number :");
        Num=int.parse (Console.ReadLine ());
        For (i=2;i<=num-1;i++)
        {
            If (num%i!=0)
                Isprime=true;
            Else
            {
                Isprime=false;
                Break;
            }
        }
    }
}

```

```
        }  
    }  
    If (isprime==true)  
        Console.WriteLine ("Given Number is Prime");  
    Else  
        Console.WriteLine ("Given Number is Not Prime");  
} } }
```

Example 3.

Write program in C# language to print the list of Palindrome numbers between (1) to (1000)?

Array.

It is a collection of memory location which stored in data and has the same type and shares in a common name.

It is also known as subscripted variable.

It could be classified into two categories, they are:

A) Single Dimensional Array.

B) Double Dimensional Array.

a)

b)

c)

d) Single Dimensional Array.

It is a collection of memory location which stored in data and has the same type and shares in a common name, this type in C# will contain a single row with multiple columns.

The general form of define this type in C# will be as following.

```
Datatype [ ] arrayname = new datatype [size];
```

For using this array in the body of program, it will be as following.

Array_name [array_index];

Example 1.

Write program in C# language to read 5 values using one dimensional array & print them at screen?

```
class demo
{
    public static void main(String [] args)
    {
        int i ;
        int [] arr = new int[5];
        Console.WriteLine ("Enter Five Values :");
        for (i = 0 ; i < 5 ; i++)
        {
            arr[i] = int.Parse(Console.ReadLine());
        }
        Console.WriteLine ("The Given Values are :");
        for (i = 0 ; i < 5 ; i++)
        {
            Console.WriteLine (arr[i]);
        }
        Console.ReadLine ();
    }
}
```

Example 2.

Write program in C# language to read 5 values using one dimensional array & print out the reverse of this array at computer screen?

```
class demo
{
    public static void main(String [] args)
```

```
{
    int i ;
    int [] arr = new int[5];
    Console.WriteLine ("Enter Five Values :");
    for (i = 0 ; i < 5 ; i++)
    {
        arr[i] = int.Parse(Console.ReadLine());
    }
    Console.WriteLine ("The Given Values are :");
    for (i = 5 ; i > 0 ; i--)
    {
        Console.WriteLine (arr[i]);
    }
    Console.ReadLine ();
}
}
```

Example 3.

Write program in C# language to read one dimensional array has [5] elements & find the summation of these elements?

```
class demo
{
    public static void main(String [] args)
    {
        int l, sum=0 ;
        int [] arr = new int[5];
        Console.WriteLine ("Enter Five Values :");
        for (i = 0 ; i < 5 ; i++)
        {
            arr[i] = int.Parse(Console.ReadLine());
        }
        Console.WriteLine ("The Given Values are :");
        for (i = 0 ; i < 5 ; i++)
        {
```

```
        Sum = sum + arr[i];
    }
    Console.WriteLine ("Sum=" + sum);
    Console.ReadLine ();
}
}
```

Example 4.

Write program in C# language to search about specific element in one dimensional array that has [10] elements?

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication24
{
    class Program
    {
        static void Main(string[] args)
        {
            int i,x;
            int f =0;
            int [] arr = new int[5];
            Console.WriteLine ("Enter Five Values :");
            for (i = 0; i < arr.Length; i++)
            {
                arr[i] = int.Parse(Console.ReadLine());
            }
            Console.WriteLine ("Please Insert Search Value :");
            x = int.Parse (Console.ReadLine ());
            for (i = 0; i < arr.Length; i++)
            {
                if (arr[i] == x)
```

```
{
    Console.WriteLine ("The value is exist");
    f = 1;
    break;
}
}
if (f == 0 )
Console.WriteLine ("The Value is not exist");
Console.ReadLine ();
}
}
}
```

Home Work 1.

Write program in C# language to print the largest element from an array that has one dimensional array [10] elements?

Home Work 2.

Write program in C# to sort the elements of an array that has one dimensional array [10] elements. (Selection Sort)?

Home Work 3.

Write program in C# language to count how many even & odd elements in an array that has one dimensional array [10] elements?

Home Work 4.

Write program in C# language to print the prime elements from a one dimensional array that has [10] elements?

e) Double Dimensional Array.

It is a collection of memory locations which stored in data and has the same type, this type of array contains multiple rows with multiple columns.

The general form of define this type of array will be.

```
Datatype [ , ] ArrayName =new DataType [rows, cols];
```

The general form of using this type in a body of program is.

```
Array_name [index of row, index of column];
```

Example1.

Write program in C# language to read 9 values for a matrix of [3*3] and print out the elements in matrix form?

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApplication5
{
class Program
    {
static void Main(string[] args)
    {
int[ , ] mat = new int[3,3];
int i, j;
Console.WriteLine (" Enter 9 Digit for the Matrix: ");
for (i = 0; i <= 2; i++)
    {
for (j = 0; j <= 2 ; j++)
    {
```

```

        mat [ i , j ] = int.Parse(Console.ReadLine());
    }
}
Console.WriteLine (" The Matrix is: ");
for (i = 0; i <= 2; i++)
{
for (j = 0; j <= 2 ; j++)
    {
        Console. Write (mat [ i , j ] + "\t");
    }
    Console.WriteLine ();
}
Console.ReadLine ();
} } }

```

Example 2.

Write program in C# language to read two dimensional array [3*3] and print out the reverse matrix at computer screen?

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApplication5
{
class Program
    {
static void Main(string[] args)
    {
int[ , ] mat = new int[3,3];
int i, j;
Console.WriteLine (" Enter 9 Digit for the Matrix: ");
for (i = 0; i <= 2; i++)
    {
for (j = 0; j <= 2 ; j++)

```

```

        {
            mat [ i , j ] = int.Parse(Console.ReadLine());
        }
    }
Console.WriteLine (" The Reverse Matrix is: ");
for (i = 2; i >= 0; i--)
    {
for (j = 2; j >= 0 ; j--)
    {
        Console. Write (mat [ i , j ] + "\t");
    }
    Console.WriteLine ();
}
Console.ReadLine ();
} } }

```

Example 3.

Write program in C# language to read two dimensional array [3*3] and find the elements summation of this matrix?

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApplication5
{
class Program
    {
static void Main(string[] args)
    {
int[ , ] mat = new int[3,3];
int i, j, sum=0;
Console.WriteLine (" Enter 9 Digit for the Matrix: ");
for (i = 0; i <= 2; i++)
    {
for (j = 0; j <= 2 ; j++)

```

```
        {
            mat [ i , j ] = int.Parse(Console.ReadLine());
        }
    }
for (i = 0; i <= 2; i++)
    {
for (j = 0; j <= 2 ; j++)
    {
        Sum=sum + mat [ i , j ];
    }
}
Console. Write ("Sum=" + sum);
Console.ReadLine ();
    }
}
}
```

Example 4.

Write program in C# language to add two matrixes, everyone has two dimensional array [2, 2]?

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

```
namespace ConsoleApplication5
{
class Program
    {
static void Main(string[] args)
    {
int [ , ] x = new int[2, 2];
int [ , ] y = new int[2, 2];
int [ , ] z = new int[2, 2];
Int l , j;
```

```
Console.WriteLine (" Enter 4 Digit for the Matrix X : ");
for (i = 0; i <= 1; i++)
    {
for (j = 0; j <= 1; j++)
    {
        x [ i , j ] = int.Parse(Console.ReadLine());
    }
    }
Console.WriteLine (" Enter 4 Digit for the Matrix Y: ");
for (i = 0; i <= 1; i++)
    {
for (j = 0; j <= 1; j++)
    {
        y[ i , j ] = int.Parse(Console.ReadLine());
    }
    }
for (i = 0; i <= 1; i++)
    {
for (j = 0; j <= 1; j++)
    {
        z[ i , j ] = x[ i , j ] + y[ i , j ];
    }
    }
Console.WriteLine ("The Result of Adding Matrix is :");
for (i = 0; i <= 1; i++)
    {
for (j = 0; j <= 1; j++)
    {
Console .WriteLine (z [ i , j]);
    }
    }
Console.ReadLine ();
    }
}
}
```

Example 5.

Write program in C# to transpose a matrix has two dimensional array [3, 3]?

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication5
{
class Program
{
static void Main(string[] args)
{
int [, ] mat = new int[3,3];
int i, j;
Console.WriteLine (" Enter 9 Digit for the Matrix: ");
for (i = 0; i <= 2; i++)
{
for (j = 0; j <= 2 ; j++)
{
mat [ i , j ] = int.Parse(Console.ReadLine());
}
}
Console.WriteLine (" The Matrix is: ");
for (i = 0; i <= 2; i++)
{
for (j = 0; j <= 2 ; j++)
{
Console. Write (mat [ i , j ] + "\t");
}
}
Console.WriteLine ();
}
}
```

```
Console.WriteLine (" The Transpose Matrix is: ");
for (i = 0; i <= 2; i++)
    {
for (j = 0; j <= 2; j++)
    {
Console. Write (mat [ j , l ] + "\t");
    }
Console.WriteLine ();
    }
}
}
```

Example 6.

Write program in C# language to add two matrixes, everyone has two dimensional array [2, 2]?

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication5
{
class Program
    {
static void Main(string[] args)
    {
int [ , ] x = new int[2, 2];
int [ , ] y = new int[2, 2];
int [ , ] z = new int[2, 2];
int l , j;
Console.WriteLine (" Enter 4 Digit for the Matrix X: ");
for (i = 0; i <= 1; i++)
```

```
{
for (j = 0; j <= 1; j++)
    {
        x[ i , j ] = int.Parse(Console.ReadLine());
    }
}
Console.WriteLine (" Enter 4 Digit for the Matrix Y: ");
for (i = 0; i <= 1; i++)
    {
for (j = 0; j <= 1; j++)
    {
        y[ i , j ] = int.Parse(Console.ReadLine());
    }
}
for (i = 0; i <= 1; i++)
    {
for (j = 0; j <= 1; j++)
    {
        z[ i , j ] = x[ i , j ]+y[ i , j ];
    }
}
Console.WriteLine ("The Result of Adding Matrix is :");
for (i = 0; i <= 1; i++)
    {
for (j = 0; j <= 1; j++)
    {
Console.WriteLine (z[ i , j ]);
    }
}
Console.ReadLine ();
    }
}
}
```


Structure in C#.

It is using to store a group of data which has different types in neighboring memory location according to one name, so it is different from array, in the array should have the same type, but the structure possible has a different type.

To define the structure in C# language, we can use the following general form.

```
Struct Structure_Name
{
    Public String var1;
    Public int var2;
}
```

Example.

```
Struct students
{
    Public string name;
    Public int age;
    Public int mark;
}
```

Note.

The definition of the structure always written out side of the main function, it will be after (Class Program).

It is shown in the following example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace ConsoleApplication37
{
class Program
    {
Struct students
    {
Public string name;
Public int age;
public int mark;
    }
static void Main(string[] args)
    {
-----
        -----
    }
}
}
```

The second step is to create object for the defining structure, to use it for reading and writing the structure data, and you should know that object created will be inside the main function. The general form for creating object is.

```
Structure_Name Object_Name = new Structure_Name ( );
```

Example.

```
students s = new students();
```

To set value to any structure field, it will be as following.

```
Object_Name.Structure_field = Value;
```

Example.

```
s.name = "Wissam";  
s.age = 36;  
s.mark = int.Parse(Console.ReadLine());
```

To print out the contains of structure fields at screen, it will as follows.

```
Console.WriteLine(Object_Name.Structure_Filed_Name);
```

Example.

```
Console.WriteLine(s.name);  
Console.WriteLine(s.age);  
Console.WriteLine(s.mark);
```

Example.

Write program in C# language to build structure for student information, so it has on record (name, age, mark) fill this record by information then print out at the screen?

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;
```

```
namespace ConsoleApplication37  
{  
class Program  
{  
struct students  
{  
public string name;  
public int age;  
public int mark;
```

```
    }  
static void Main(string[] args)  
    {  
students s = new students();  
    s.name = "Wissam";  
    s.age = 36;  
    s.mark = 68;  
    Console.WriteLine ("the student information is :");  
    Console.WriteLine ("Full Name = " + s.name);  
    Console.WriteLine ("Age = " + s.age);  
    Console.WriteLine ("Mark = " + s.mark);  
    Console.ReadLine ();  
    }  
}  
}
```

Example.

Write program in C# language to build structure for five students' information, so it has on record (name, age, mark) fill this record by information then print out at the screen?

List in C#.

It is a technique in which allows the programmer to store a several data that have same type in neighboring memory locations, we can say it is similar to array but it is a single dimensional array, so we can't represent as a double dimensional array which using in game programming applications and 3D, 4D dimensional applications.

List has some features, like:

1- Dynamic using.

2- We can easily add new elements to a list (this property not available in array).

3- We can easily delete exist elements from a list (this property not available in array).

Dealings with List in C#.

1. We can define any new list in C# language by using the following general form.

```
List <Type> List_Name = New List <Type> ( );
```

Example.

```
List <int> students = new List<int>();
```

2. We can add new element to the list by using the following general form.

```
List_Name.Add (Value);
```

Example.

```
students.Add (2);
```

Note.

The length of the list always start from (0) in C# language, and continuous to infinity.

Note2.

We can fill array elements automatically by using (for) statement, it is shown in the following example.

Example.

```
for (i=0; i<10;i++)  
{  
    students .Add (int.Parse (Console.ReadLine()));  
}
```

3- List count.

If we want to know the number of the elements in a list, we can use the command (count), it is shown in the following example:

```
Distination = List_Name.Count;
```

Example.

```
Console.WriteLine (students.Count);
```

4-Print list elements.

We can print out the elements of the list at screen by using for statement or for each statement, it is shown in the following example:

Example.

```
Foreach (int n in students)  
{  
    Console.WriteLine (n);  
}
```

5- Remove Item from a list.

We can remove any item from a list by using either the element value or by using the element index, the general form of removing any item from a list in C# is.

A. Remove item from list by using item value.

```
List_Name.Remove (Value);
```

Example.

```
students.Remove (7);
```

B. Remove item from list by using index.

```
List_Name.RemoveAt (index of the value);
```

Example.

```
students.RemoveAt (3);
```

Example 1.

Write program in C# language to define a list has five elements, then fill this array by elements print out these elements at screen, then remove two elements from array one by value, and one by index, then print out the final element list at screen?

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication38
{
    class Program
    {
        static void Main(string[] args)
        {
```

```
List<int> students = newList<int>();
    students.Add (2);
    students.Add (32);
    students.Add (12);
    students.Add (7);
    students.Add (17);
Console.WriteLine ("The Elements of the List before Remove Items are :");
    Foreach (int n in students)
        {
            Console.WriteLine (n);
        }
Console.WriteLine ("before Remove Items are:" + students.Count);
    students.Remove (7);
    students.RemoveAt (3);
Console.WriteLine ("*****");
Console.WriteLine ("The Elements of the List after Remove Items are :");
    Foreach (int n in students)
        {
            Console.WriteLine (n);
        }
Console.WriteLine ("after Remove Items are:" + students.Count);
Console.ReadLine ();
} } }
```

Example 2.

Write program in C# language to define list has (10) elements, then print out the elements which greater than (10) at the computer screen?

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```



```
namespace ConsoleApplication38
{
class Program
{
static void Main(string[] args)
{
List<int> students = newList<int>();
students.Add (2);
students.Add (32);
students.Add (12);
students.Add (7);
students.Add (17);
Console.WriteLine ("The Elements Which greater than 10 are :");
foreach (int n in students )
{
if (n > 10)
Console.WriteLine (n);
}
Console.ReadLine ();
} } }
```

Example 3.

Write program in C# language to define list has (10) elements, fill this list by value, print out the even value in this list at the computer screen?

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace ConsoleApplication38
{
class Program
{
```

```
static void Main(string[] args)
    {
    List<int> students = new List<int>();
    Console.WriteLine ("please Insert List Elements :");
    int i;
    for (i=0; i<10;i++)
        {
            students .Add (int.Parse (Console.ReadLine()));
        }
    Console.WriteLine ("The Elements Which greater than 10 are :");
    foreach (int n in students )
        {
        if (n % 2 == 0)
        Console.WriteLine (n);
        }
    Console.ReadLine ();
    } } }
```

Blocks in C#.

It is a division program into a set of parts each part contains a set of commands that represent the code in order to carry out a specific operation process, the benefits of this division process will lead to:

- 1-Ease of maintenance program.
- 2-Abbreviation for the program code, where I can use the same block in multiple places in the program.
- 3-Ease of understanding of the program by other programmers.
- 4- Reducing of written code will lead to reduce the space needed by the program from memory.
- 5-Ease to develop program.

The types of blocks in C#.

There are two types of blocks in C#, they are.

1- Procedure.

It is division program to a set of parts each part contains a set of commands that represents the code, you should know that we can call every procedure from any part of program as well as the procedure will be received from the main program many parameters and manipulate a specific job then it don't return any result to the main program.

To define any procedure in C# language, we can use the general form.

Access_modifiers void procedure_name (parameters)

```
{  
    Procedure_code;  
}
```

So:

- ✓ Access_modifiers: represent the access method to the procedure, in C# we have four types of the Access modifiers, they are:

1- Public type: in this type we can access to the procedure from any part of program, i.e. Access is not restricted.

2- Private type: in this type we can access to the procedure from the part which define procedure inside it.

3- Protected type: in this type the access is limited to the containing procedure from a driven procedure only.

4- Static type: in this type the access will be static from any part of program.

- ✓ Void: that means the procedure will not return any value to the program.
- ✓ Procedure name: For representing the procedure name in the program, we should follow the naming rules in the C# which we explained it in the beginning of this book.
- ✓ Parameters: represent the variables that pass to the procedure, so we can pass the parameters with their types as following:

```
Access_modifiers      void      procedure_name(parameter_type
parameter_name, ...etc.)
```

Example.

```
Private static void addtwo (int a, int b)
{
    Procedure body;
}
```

To call this procedure from any part of program, we can use the following formula.

```
Procedure_name ( );
```

Example1.

Write program in C# language to build procedure which print out ("oop with C#") at the computer screen?

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

```
using System.Threading.Tasks;

namespace ConsoleApplication26
{
class Program
    {
static void Main(string[] args)
    {
        print ();
        Console.ReadLine ();
    }
    Private static void print ( )
    {
        Console.WriteLine ("oop with C#");
    }
    }
    }
}
```

Example2.

Write program in C# language to build procedure which calculate addition two numbers (a, b) and put the result in (c) and write the result at the computer screen?

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication26
{
class Program
    {
static void Main(string[] args)
    {
        int a = 5;
```

```
        int b = 6;
        addtwo (a, b);
        Console.ReadLine ();
    }
Private static void addtwo (int a , int b)
    {
        int c = a + b;
        Console.WriteLine(c);
    }
}
}
```

Example3.

Write program in C# language to build procedure which read two values (x & y) and return the greater value from them?

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication26
{
    class Program
    {
        static void Main(string[] args)
        {
            int x, y;
            Console.WriteLine ("Please insert two numbers :");
            x = int.Parse (Console.ReadLine ());
            y = int.Parse (Console.ReadLine ());
            max ( x , y );
            Console.ReadLine ();
        }
    }
}
```

```
private static void max (int x , int y)
{
    if (x > y)
        Console.WriteLine ("the value of x is greater than y");
    if (x < y)
        Console.WriteLine ("the value of y is greater than x");
    if (x == y)
        Console.WriteLine ("the two values are equal");
}
}
```

2- Function.

It is a division program to a set of parts each part contains a set of commands that represents the code, you should know that we can call every function from any part of program as well as the function will be received from the main program many parameters and manipulate a specific job then it will return a result to the main program.

To define any function in C# language, we can use the general formula.

Access_modifiers return_value_type function_name (parameters)

```
{
function_body;
Return (value);
}
```

So:

- ✓ Access_modifiers: represent the access method to the function, in C# we have four types of the Access modifiers, they are:

1- Public type: in this type we can access to the function from any part of program, e.i Access is not restricted.

2- Private type: in this type we can access to the function from the part which define function inside it.

3- Protected type: in this type the access is limited to the containing function from a driven function only.

4- Static type: in this type the access will be static from any part of program.

- ✓ Return_value_type: it represents the type of the value that return from the function to the main program.
- ✓ Function name: it represents the function name in the program; we should follow the naming rules in the C# which we explained it in the beginning of this book.
- ✓ Parameters: represents the variables that pass to the function, so we can pass the parameters with their types as following:

```
Access_modifiers      void      function_name(parameter_type
parameter_name, ...etc.)
```

Example.

```
Private static int addtwo (int a, int b)
{
}
}
```

To call this function from any part of program, we can use the following formula.

Distination_variable_name=function_name (parameters will pass to the function);

Example 1.

Write program in C# language to read two values from keyboard, add together and print out the result at the screen, by using function technique?

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApplication27

{
class Program
    {
static void Main(string[] args)
    {
        int x;
        int y;
        int z;
        Console.WriteLine ("please insert two values X & Y");
        x = int.Parse (Console.ReadLine ());
        y = int.Parse (Console.ReadLine ());
        z = addtwo(x, y);
        Console.WriteLine ("z=" + z);
        Console.ReadLine ();
    }
private static int addtwo(int x, int y)
    {
        return (x + y);
    }
}
}
```

Example 2.

Write program in C# language to read two values from keyboard (x & y) and find the summation values between (x & y), then print out the result at computer screen by using function technique?

```
using System.Text;
```

```
using System.Threading.Tasks;
```

```
namespace ConsoleApplication28
```

```
{
```

```
class Program
```

```
{
```

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

```
{
```

```
    int x;
```

```
    int y;
```

```
    int z;
```

```
    Console.WriteLine ("please insert two values X & Y");
```

```
    x = int.Parse (Console.ReadLine ());
```

```
    y = int.Parse (Console.ReadLine ());
```

```
    z = sum(x, y);
```

```
    Console.WriteLine ("z=" + z);
```

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

```
}
```

```
private static int sum(int x, int y)
```

```
{
```

```
    int i,s=0;
```

```
    for (i=x; i<=y ;i++)
```

```
    {
```

```
        s = s + i;
```

```
    }
```

```
    return (s);
```

```
}
```

```
}  
}
```

Note.

The C# language allows us to call a specific function or procedure from inside other function or procedure, it is shown in the following example.

Example.

Write program in C# language to read two values from keyboard and multiply them, and then print out the result at the screen by using call function from inside other function technique?

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;  
namespace ConsoleApplication28  
{  
class Program  
{  
static void Main(string[] args)  
{  
    read ();  
    Console.ReadLine ();  
}  
private static void read()  
{  
    int x;  
    int y;  
    Console.WriteLine ("please insert two values X & Y");  
    x = int.Parse (Console.ReadLine ());  
    y = int.Parse (Console.ReadLine ());  
}
```

```
        mult ( x , y );
    }

private static void mult (int x, int y)
    {
int z;
        z = x * y;
        print (z);
    }
private static void print (int z)
    {
        Console.WriteLine ("z=" + z);
    }
}
```

Note.

Note that the variables types and their numbers must to be compatible between a headers of the function and call it, it is shown in the following example.

Example.

```
{
    Float x;
    Sum (x);
}
Private static void (int x, int y)
{
    .
    .
}
```

In the above example, there are two errors, the first error is representing in which we call the function with one parameter, it is (x) but when we

define the function we write in the header two parameters (x & y), this is the first error.

The second one represents in which we call the function with one parameter (x) and the type of (x) is float, but when we define function we write in a header(x) and we define (x) as integer type, therefore the above example is wrong because there is no compatible between the headers of the function and call it.

Note 1.

In C# language there are many ready built in functions written by a language designers, and we can directly use by the programmers, like.

```
Console.ReadLine ( );  
Console.WriteLine ( );  
Int.Parse ( );
```

Note 2.

We can pass a string as parameters to the function or procedure; it is shown in the following example.

Example.

We will pass string (str) to the procedure to print out at the screen with ("Helow") by the procedure?

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;
```

```
namespace ConsoleApplication31  
{  
class Program  
{  
static void Main(string[] args)
```

```
{
    string str;
    str = "Wissam Ali";
    show (str);
    Console.ReadLine ();
}
private static void show( string str)
{
    Console.WriteLine ("Helow" + str);
}
}
```

Passing parameter to the function.

There are three types of passing parameters to the function in C# language, they are:

1- Passing parameter by value.

In this type we will send value to the function/procedure, so the actual value of the parameter will not affect outside the function, but the value which sent to the function/procedure (inside function) will affect.

It is shown in the following example:

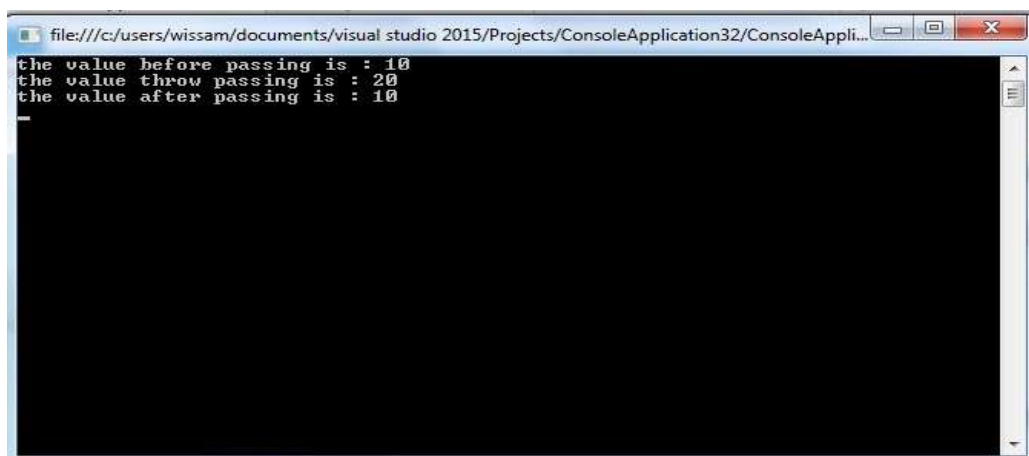
Example.

In the following example, we will pass parameter to the function by value, so we will use the parameter (x) in the main program and we will give specific value, then we will send to the function formal parameter to the function (x1) and we will print out the value of this parameter before, through and after passing to see the effect.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace ConsoleApplication32
{
class Program
{
static void Main(string[] args)
{
int x = 10;
Console.WriteLine ("the value before passing is: " + x);
incr (x);
Console.WriteLine ("the value after passing is: " + x);
Console.ReadLine ();
}
private static int incr(int x1)
{
x1 = x1 + 10;
Console.WriteLine ("the value throw passing is: " + x1);
Return (x1);
}
}
}
```

When we execute this program, the execution screen will be as following.



The screenshot shows a console window with the following output:

```
the value before passing is : 10
the value throw passing is : 20
the value after passing is : 10
```

2- Passing parameters by reference.

In this type we will send a reference of the parameter to the function/procedure, so the actual value of the parameter will affect inside and outside the function, we can apply this technique by using the reserve word (ref) before parameters in the call and header of the function.

It is shown in the following example:

Example.

We will apply the same previous example, but by using reference technique, and we will see the results?

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.Linq;
```

```
using System.Text;
```

```
using System.Threading.Tasks;
```

```
namespace ConsoleApplication32
```

```
{
```

```
class Program
```

```
{
```

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

```
{
```

```
int x = 10;
```

```
Console.WriteLine ("the value before passing is: " + x);
```

```
incr (ref x);
```

```
Console.WriteLine ("the value after passing is: " + x);
```

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

```
}
```

```
private static int incr(ref int x1)
```

```
{
```

```
x1 = x1 + 10;
```

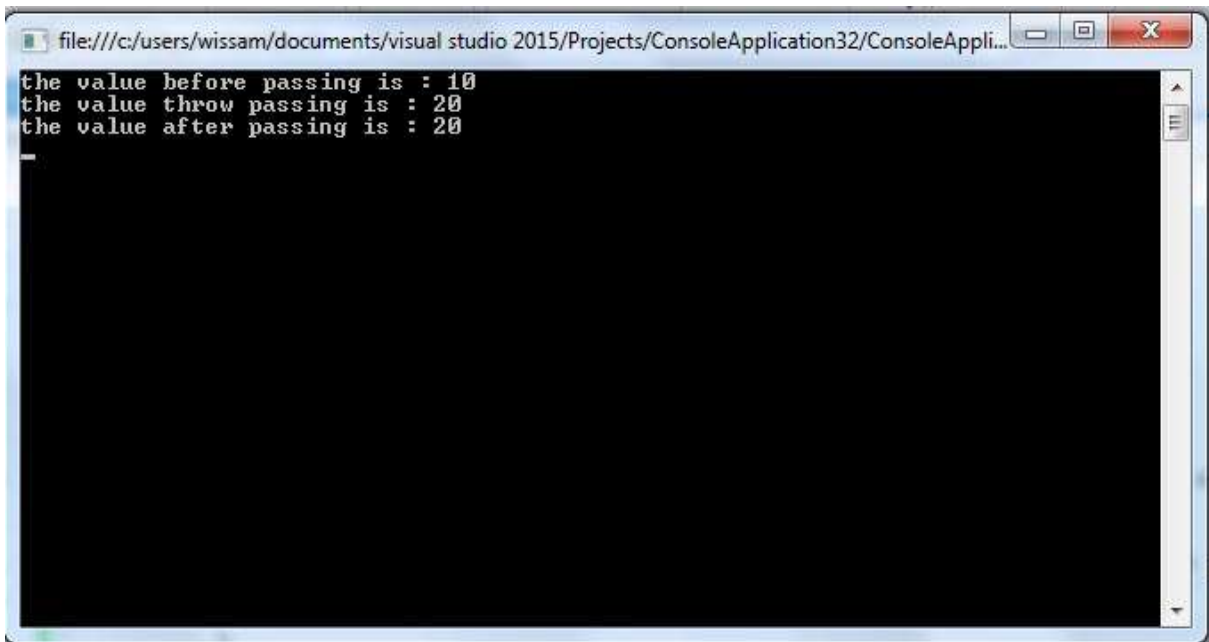
```
Console.WriteLine ("the value throw passing is: " + x1);
```

```
return (x1);
```

```
}
```

```
}
```

```
}
```

```
file:///c:/users/wissam/documents/visual studio 2015/Projects/ConsoleApplication32/ConsoleAppli...
the value before passing is : 10
the value throw passing is : 20
the value after passing is : 20
```

3- Passing Parameter by the output.

It is similar to passing parameter by reference, but the difference between them is in the first type we should give values to the parameters before passing them to the function, but in the second type we should not give values to the parameter before passing them to the function, to apply this technique we will use the reserve word (out) before the parameters in the call and header of the function.

Finally we should note that the actual value of the parameters will change in the memory after execute function/procedure.

It is shown in the following example:

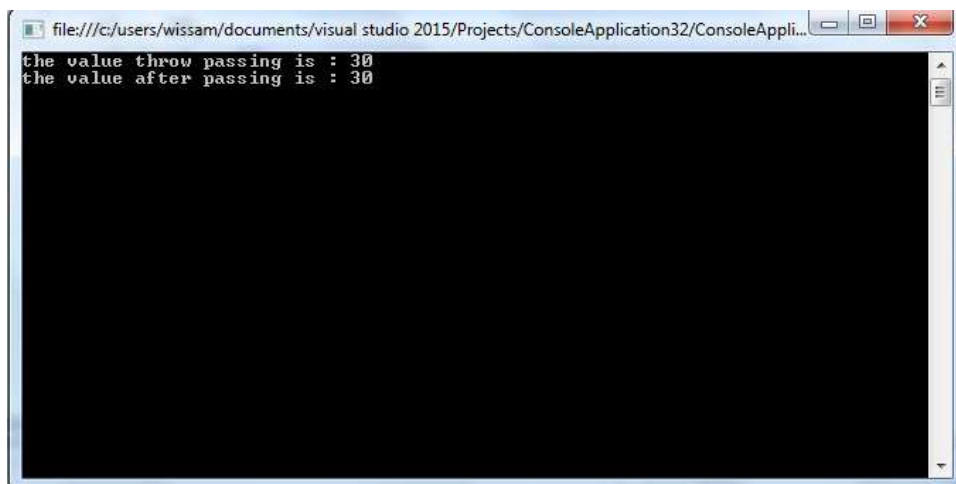
Example.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace ConsoleApplication32
{
class Program
```

```
{
static void Main(string[] args)
{
    int x ;
    incr (out x);
    Console.WriteLine ("the value after passing is: " + x);
    Console.ReadLine ();
}
private static int incr (out int x1)
{
    x1 = 20;
    x1 = x1 + 10;
    Console.WriteLine ("the value throw passing is: " + x1);
    return (x1);
}
}
```

When we execute this program we will get as the following screen.



Passing Array to the Function.

We can pass one dimensional array or two dimensional arrays to the function in C# language, it is shown in the following examples:

Example1.

Write program in C# language to read one dimensional array has [5] elements, then passing this array to specific function to print out at computer screen?

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication34
{
    class Program
    {
        static void Main(string[] args)
        {
            int i;
            int[] arr = new int[5];
            Console.WriteLine ("Enter Values of Five Elements :");
            for (i=0;i<5;i++)
            {
                arr[i] = int.Parse (Console.ReadLine());
            }
            print (arr);
            Console.ReadLine ();
        }
        private static void print (int[] arr1 )
        {
            int i;
            for (i = 0; i<5; i++)
            {
                Console .WriteLine (arr1 [i]);
            }
        }
    }
}
```

Example2.

Write program in C# language to read two dimensional array has [2, 2] elements, then passing this array to specific function to print out at computer screen?

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication34
{
    class Program
    {
        static void Main(string[] args)
        {
            int i,j;
            int [ , ] arr = new int[2,2];
            Console.WriteLine ("Enter Values of Four Elements :");
            for (i = 0; i < 2; i++)
            {
                for (j=0;j<2;j++)
                {
                    arr[ i , j ] = int.Parse(Console.ReadLine());
                }
            }
            print(arr);
            Console.ReadLine ();
        }
        private static void print(int [ , ] arr1)
        {
            int i,j;
            for (i= 0;i<2;i++)
            {
```

```

        for (j=0;j<2;j++)
            {
                Console.WriteLine (arr1 [ i , j ]);
            }
        }
    }
}

```

Example3.

Write program in C# to read two arrays everyone has two dimensional array [2, 2], then add these arrays together and print out at the screen by using functional technique?

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication34
{
    class Program
    {
        static void Main(string[] args)
        {
            int i,j;
            int [ , ] a = new int[2,2];
            int [ , ] b = new int[2, 2];
            Console.WriteLine ("Enter Values of array A :");
            for (i = 0; i < 2; i++)
                {
                    for (j=0;j<2;j++)
                        {
                            a[ i , j ] = int.Parse(Console.ReadLine());

```

```

    }

    }
    Console.WriteLine ("Enter Values of array B :");
    for (i = 0; i < 2; i++)
    {
        for (j = 0; j < 2; j++)
        {
            b[ i , j ] = int.Parse(Console.ReadLine());
        }

    }
    print (a,b);
    Console.ReadLine ();
}
private static void print(int [ , ] a, int [ , ] b)
{
    int i,j;
    int [ , ] c = new int[2, 2];
    for (i = 0; i < 2; i++)
    {
        for (j = 0; j < 2; j++)
        {
            c[ i , j ]=a[ i , j ]+b[ i , j ];
        }
    }
    Console.WriteLine ("The Result of Adding these arraye are :");
    for (i= 0;i<2;i++)
    {
        for (j=0;j<2;j++)
        {
            Console.WriteLine(c [ i , j ]);
        }
    }
}
}
}

```

```
}  
}
```

Example4.

Write program in C# to read two arrays everyone has one dimensional array [5], elements, then add these arrays together and print out at the screen by using functional technique?

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;  
  
namespace ConsoleApplication34  
{  
class Program  
{  
static void Main(string[] args)  
{  
int i;  
int[] a = new int[5];  
int[] b = new int[5];  
Console.WriteLine ("Enter Values of array A :");  
for (i = 0; i < 5; i++)  
{  
a[i] = int.Parse(Console.ReadLine());  
}  
Console.WriteLine ("Enter Values of array B :");  
for (i = 0; i < 5; i++)  
{  
  
b[i] = int.Parse(Console.ReadLine());  
}  
print(a,b);  
Console.ReadLine ();
```

```

    }
private static void print(int[] a, int[] b)
    {
        int i;
        int[] c = new int[5];
        for (i = 0; i < 5; i++)
            {
                c[i]=a[i]+b[i];
            }
        Console.WriteLine ("The Result of Adding these arraye are :");
        for (i= 0;i<5;i++)
            {
                Console.WriteLine(c[i]);
            }
    }
}
}
}

```

Recursive Function.

It is a function that call itself by itself, it is shown in the following example.

Example.

Write program in C# language to find factorial of any given number by using recursive function technology?

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

```

```

namespace ConsoleApplication36
{
class Program
    {
static void Main(string[] args)

```



```
{
int n,res;
Console.WriteLine ("Please Insert the Number of Factorial :");
    n = int.Parse (Console.ReadLine ());
    res= fact(n);
Console.WriteLine ("The Result of the Factorial Number is:" + res);
Console.ReadLine ();
}
private static int fact(int n)
{
    int f;
    if (n == 1)
        return (1);
        f = fact (n - 1) * n;
    return (f);
}
}
```

Example2.

Write program in C# language to find (X^n) by using recursive function technology?

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace ConsoleApplication36
{
class Program
{
```

```
static void Main(string[] args)
{
    int x,n,res;
    Console.WriteLine ("Please Insert the Number of X :");
        x = int.Parse (Console.ReadLine ());
    Console.WriteLine ("Please Insert the Number of N :");
        n = int.Parse (Console.ReadLine ());
        res = power(x,n);
    Console.WriteLine ("The Result of the Power is:" + res);
    Console.ReadLine ();
}
private static int power(int x, int n)
{
    int p;
    if (n == 0)
        return (1);
        p = x * power(x, n - 1);
    return (p);
}
}
```

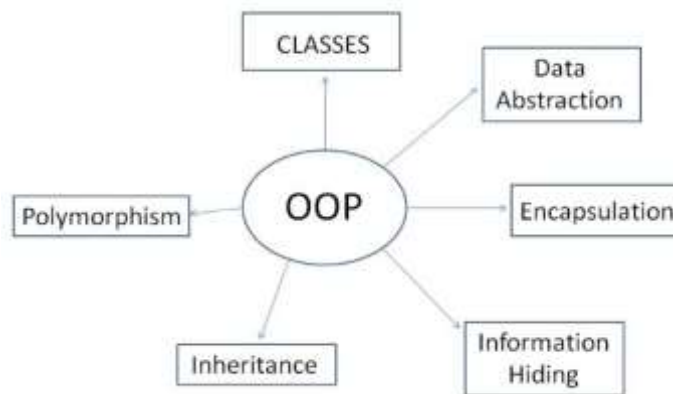
Chapter 3

Object Oriented Programming

Object Oriented Programming (OOP) with C#.

It is an advanced programming paradigm, in which the program is division into units called objects, each object is a package of data, variables, constants, functions and organization units and interfaces to use.

The program is built by using the objects and linking them with each other and the external interface program is using the program structure and interfaces to use for each object.



Object-oriented programming is a new way to design and write software, the main idea in which you convert the program into different parts and each part represents a goal or a specific job.

The oop technique consist of two concepts, they are:

- 1- Object.
- 2- Class.

Object.

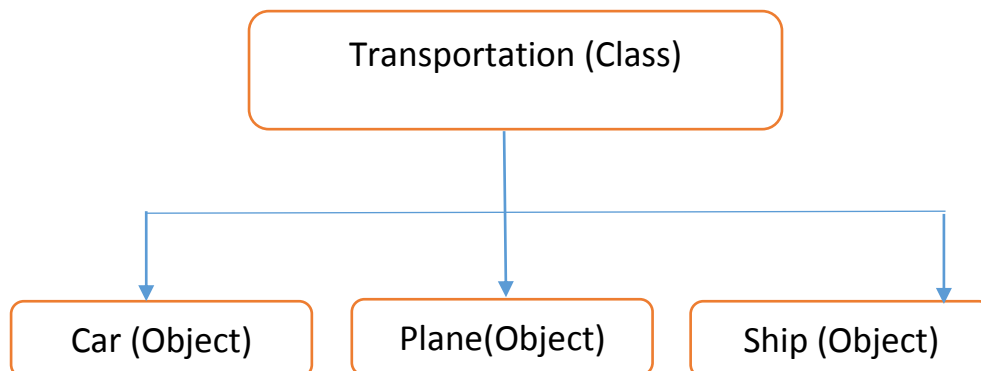
Perhaps all that we see in our daily life of humans, animals and fruit ... etc. It is an object, if we look at the category of animals, for example, lion , tiger , deer and rabbit each representing an independent creature itself, and has the characteristics that distinguish it from the other, and the behaviors and functions.

So each object is characterized by the properties and behaviors performed by these behaviors and produce events, and these three factors, each object is characterized by what others:

- 1- properties: they are called in programming language (Data).
- 2-behavior: they are functions performed by the object and called in programming language (Method or Functions).
- 3- Actions: they are performance that result from the behavior of the object is called in programming language (Events).

Class.

Each object belongs to a class higher than that ,for example, orange is an object which belongs to the category of fruit, the lion is an object belongs to the category of animals, car , plane and ship are objects which belong to the category of transportation.



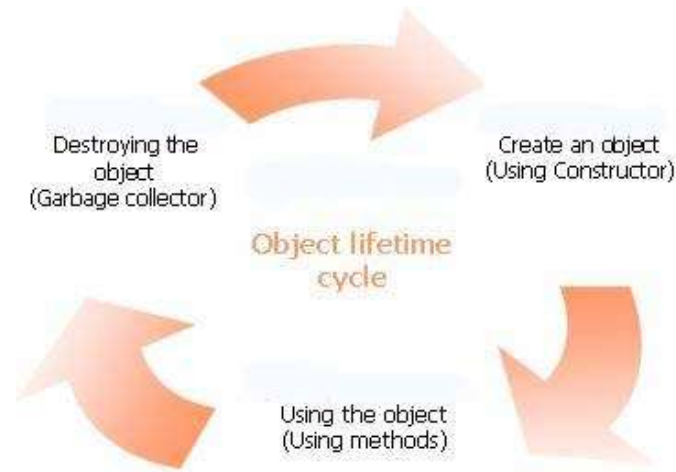
All the programs that written in C language are collection of (functions) that work together, either in relation to the language of C# programs, it is a set of classes, whether they made the programmer (User define classes) or classes ready in the language (Built in classes).

These classes contain inside a set of functions that are called (Method) and variables reservation data also contain a set of events.

And known these three (functions - variables - events) as Members.

Object Life Cycle.

The following figure is shown the object life cycle.



Create Class in C# Language.

We can create class in C# language by using the reserve word (Class) and using the following Structure.

```
Class class_name
{
    // define variables (fields)
    Var1;
    Var2,
    .....
    Var n;
    // Define Methods
    Public void method1_name (parameters)
    {
        Method_code;
    }
}
```

```
Public void method2_name (parameters)
{
Method_code;
}
.....
}
```

Example.

```
class human
{
    mind,hands,legs,eyes.....; }Attributes
    _____
    thinking ()
    working ()          => Behaviors
    walking ()
    vision ()
}
human frah, zaid, ali
```

Create object.

To use the definition class in program we should define object which using to call the member function in definition class, the general form to create object is:

```
Class_Name object_Name = new Class_Name ( );
```


Example.

```
Car mycar = new car ( );
```

Call member functions.

To call the member functions founded within class, we should use the created object, and the general form to call functions is.

```
Object_Name.Membrer_Function_Name (Parameters);
```

Example.

```
mycar.initilize (n, c, m);
```

Example.

In the following example we will built class for specific car have the fields (name, color, model). As well as it has two methods, the first method calls (Initialize) to set values to these fields, and other is (Show) to print these features at monitor screen?

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;
```

```
namespace ConsoleApplication40  
{  
class car  
{  
    string name;  
    string color;
```

```
string model;

public void initialize (string n, string c, string m)
    {
        name = n;
        color = c;
        model = m;
    }

public void show ()
    {
        Console.WriteLine ("Name: " + name);
        Console.WriteLine ("Color: " + color);
        Console.WriteLine ("Model: " + model);
    }
}

class Program
{
static void Main(string[] args)
    {
        car mycar = new car();
        mycar.initialize ("Kia", "Red", "2017");
        mycar.show ();
        Console.ReadLine ();
    }
}
}
```

Note.

We can create more than one object in the program, and using these objects to call member functions.

It is show in the following example:

Example.

We will rewrite the previous example to create three objects for three cars.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication40
{
    class car
    {
        string name;
        string color;
        string model;

        public void initialize (string n, string c, string m)
        {
            name = n;
            color = c;
            model = m;
        }

        public void show ()
        {
            Console.WriteLine ("Name: " + name);
            Console.WriteLine ("Color: " + color);
            Console.WriteLine ("Model: " + model);
        }
    }
}

class Program
{
    static void Main(string[] args)
```

```
{
    car mycar1 = new car();
    mycar1.initialize ("Kia", "Red", "2017");
    Console.WriteLine ("The Feature of The First Car is :");
    mycar1.show ();
    car mycar2 = new car();
    mycar2.initialize ("BMW", "White", "2015");
    Console.WriteLine ("The Feature of The Second Car is :");
    mycar2.show ();
    car mycar3 = new car();
    mycar3.initialize ("Mercedes", "Blue", "2013");
    Console.WriteLine ("The Feature of The Third Car is :");
    mycar3.show ();
    Console.ReadLine ();
}
}
```

Access Modifier.

It is a declaration operation for the variables, Functions, Classes, Methods and objects.

This technology will specify of the element visibility within program and how we can access to it, in C# language there are four types of elements access modifiers, they are.

1- Normal Access Modifier.

In this type, the elements definition will be visible only within the part which belongs to it, for example.

Example.

```
class Program
{
    static void Main(string[] args)
    {
        int x;
        string name;
    }
}
```

In the above example, the variables (x & name) will be visible only within the class (program) which defines in it, and we cannot use them in other class.

2- Public Access Modifier.

In this type, the elements definition will be visible by all classes follow to the program but we should note that these variables must define at class level (not Member level), it is show in the following example:

Example.

```
class Program
{
    public int x;
    public string name;
    static void Main(string[] args)
    {

    }
}
```

3- Private Access Modifier.

In this type the elements definition will be visible only in the class define within it, it is similar to first type but the difference between them is that this type will define at class level only (similar to public) but in the first type we can define it in both kinds (class level & method level), for example:

Example.

```
class Program
{
    Private int z;
    private int x;
    private string name;
    static void Main(string[] args)
    {

    }
}
```

4- Protected Access Modifier.

In this type the elements definition will be visible within the class which defines in it, and within the class which inherits from definition class, it is only defined at class level too. For example:

Example.

```
class Program {
    protected int z;
    protected int x;
    protected string name;
    static void Main(string[] args)
    {
```

```
    }  
  }  
}
```

Methods Types.

In C# language a user defined methods and we can classify them into four categories, they are:

1. Method with no arguments & not return values.
2. Method with arguments but not return values.
3. Method with arguments and return values.
4. Method with not arguments but return values.

1. Method with no arguments & not return values.

In this type we don't send any parameters to the method and in the same time we don't receive any results from this method, for example for this type messages printing method.

It is shown in the following example:

Example.

Write program in C# language to generate the following output?

```
*****  
                Baghdad University  
*****  
*****
```

Solution:

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;
```

```
using System.Threading.Tasks;
```

```
namespace ConsoleApplication43
```

```
{
```

```
Class logic
```

```
{
```

```
public void line()
```

```
{
```

```
int i;
```

```
for (i = 1; i <= 70; i++)
```

```
{
```

```
Console. Write ("*");
```

```
}
```

```
Console.WriteLine ();
```

```
}
```

```
}
```

```
class Program
```

```
{
```

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

```
{
```

```
logic obj = new logic();
```

```
obj.line ();
```

```
Console.WriteLine ("\t\t\t Baghdad University");
```

```
obj.line ();
```

```
obj.line();
```

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

```
}
```

```
}
```

```
}
```



```
        {
        Console. Write (ch);
        }
        Console.WriteLine ();
    }
}

class Program
{
    static void Main(string[] args)
    {
        logic obj = new logic();
        obj.line ("*");
        Console.WriteLine ("\t\t\t Baghdad University");
        obj.line ("&");
        obj.line ("^");
        Console.ReadLine ();
    }
}
}
```

Home Work.

Write program in C# language to create a Method named as table () which contains a number as argument & print the table of given number, by using Method with arguments but not return values technology?

.....

2. Method with arguments and return values.

If we want to perform any dynamic operation by a given value & result is an aggregate value, i.e. in this type we will send parameters to the method and in the same time we will return results to the main program or other method in program.

It is shown in the following example:

Example.

Write program in C# language to create method named max () which contains three numbers as arguments & return the largest one, by using Method with arguments and return values technology?

Solution.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace ConsoleApplication43
{
class logic
{
    public int max(int a, int b, int c)
    {
        int m;
        if (a > b && a > c)
            m = a;
        else if (b > a && b > c)
            m = b;
        else
            m = c;
        return m;
    }
}
class Program
{
    static void Main(string[] args)
    {
        logic obj = new logic();
        int res = obj.max(100, 20, 4);
        Console.WriteLine ("Largest: " + res);
    }
}
```

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

Example 2.

Write program in C# language to create method named fact () which contains a number as argument & return the factorial of that number, by using Method with arguments and return values technology?

Example 3.

Create method named as power () which contains base number(x) and power number (n) as argument & return x to the power n, by using Method with arguments and return values technology?

3. Method with no argument but return value.

In this type we will not pass any parameters to the method, and in the same time we will receive results after the method finished its work, it is shown in the following example.

Example.

Rewrite the previous example by using the method with no argument but return value, by using Method with no arguments but return values technology?

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;
```

```
namespace ConsoleApplication43
{
class logic
{
    int a, b, c;
    public void get(int x, int y, int z)
    {
        a = x;
        b = y;
        c = z;
    }
    public int max()
    {
        int m;
        if (a > b && a > c)
            m = a;
        else if (b > a && b > c)
            m = b;
        else
            m = c;
        return m;
    }
    public void show()
    {
        Console.WriteLine ("Largest: " + max ());
    }
}
class Program
{
    static void Main(string[] args)
    {
        logic obj = new logic();
        obj.get (100, 20, 4);
        obj.show ();
    }
}
```

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

Constructors Methods.

It is a method which writes to do specific job, it has the same name of class and it doesn't return any value to main program.

It is using to initialize a values to the variables, we can summarize the characteristics in the following points.

1. The name of constructor method must be same as the name of class.
2. They don't return any value to the main program.
3. They called automatically when the object of the class is created.

Types of Constructors Methods.

1. Default Constructors Methods.
2. Parameterized Constructors Methods.
3. Overloaded Constructors Methods.

1. Default Constructors Methods.

If a constructors Method don't contain any argument then it is known as default constructors Methods.

It is shown in the following example:

Example.

In the following example we will define two arguments (name & age) we will give values to these arguments, by using default constructor method?

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace ConsoleApplication43
```

```
{
    class cons
    {
        String name;
        int age;
        public cons()
        {
            name = "Wissam";
            age = 36;
        }
        public void show()
        {
            Console.WriteLine ("Name: " + name);
            Console.WriteLine ("Age: " + age);
        }
    }
}
```

```
class Program
```

```
{
    static void Main(string[] args)
    {
        cons obj = new cons();
        obj.show ();
        Console.Read ();
    }
}
```

```
}  
}  
}
```

2. Parameterized Constructors Methods.

The constructors Methods which take the arguments are known as parameterized constructors Methods.

The values of arguments must be passed at the time of object creation.

It is shown in the following example:

Example.

In the following example we will define to arguments (name & age) we will give values to these arguments, by using Parameterized constructor method?

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;  
  
namespace ConsoleApplication43  
{  
    Class cons  
    {  
        String name;  
        int age;  
        public cons(string n , int a)  
        {  
            name = n;  
            age = a;  
        }  
    }  
}
```



```

public void show()
    {
        Console.WriteLine ("Name: " + name);
        Console.WriteLine ("Age: " + age);
    }
}
class Program
{
    static void Main(string[] args)
    {
        cons obj = new cons("Wissam Ali", 36);
        obj.show ();
        Console.Read ();
    } } }

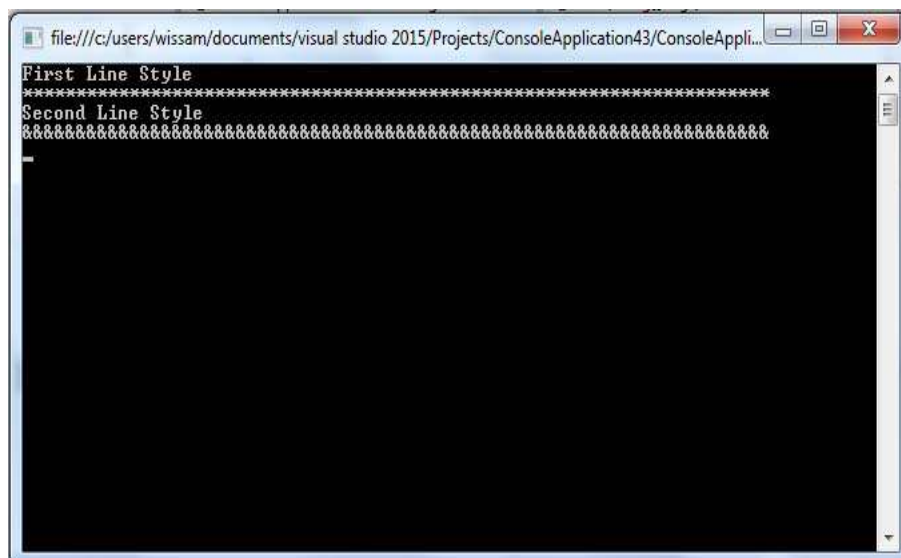
```

3. Overloaded Constructors.

If a class contains more than one constructor then this term is known as overloaded constructor. For example:

Example.

Write program in C# language to print out the following figure at the monitor screen?



```

file:///c:/users/wissam/documents/visual studio 2015/Projects/ConsoleApplication43/ConsoleAppli...
First Line Style
*****
Second Line Style
#####

```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication43
{
    Class cons
    {
        Class Line
        {
            int i;
            public Line()
            {
                for (i = 1; i <= 70; i++)
                {
                    Console. Write ("*");
                }
                Console.WriteLine ();
            }
            public Line(char ch)
            {
                for (i = 1; i <= 70; i++)
                {
                    Console. Write (ch);
                }
                Console.WriteLine ();
            }
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            Line i;
        }
    }
}
```

```
Console.WriteLine ("First Line Style");
i = new Line ();
Console.WriteLine ("Second Line Style");
i= new Line ('&');
Console.Read ();
    }
  }
}
```

OOP characteristics (Concepts).

There are four characteristics for the object oriented programming language, we will discuss everyone in below.

1) Inheritance.

It is a basic concept of the oop, we can explain this concept briefly through saying it is a derivation operation of the found class characteristic (methods & variables) in program to new class created by programmer, so the basic class will call the main class or parent class, and new class will call derived class or child class.

The purpose of using this technique is to abbreviation code written by programmer and doesn't need to rewrite the common codes between the derivation classes.

If we want to derive new class from exist class in C# language, we will use the following form.

```
Class derived_class_name: base_class_name
{
    Body of the class;
}
```

Note.

Remember that only public members of base class inherited in derived class.

Example.

The following example shows the general form of deriving of child class from parent class, such that we will derive child class called ("sec") from base class called ("first")?

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace ConsoleApplication43
{
    Class first
    {
        public void one()
        {
            Console.WriteLine ("Function of Base Class");
        }
    }
    Class sec : first
    {
        public void two()
        {
            Console.WriteLine ("Function of Derived Class");
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
```

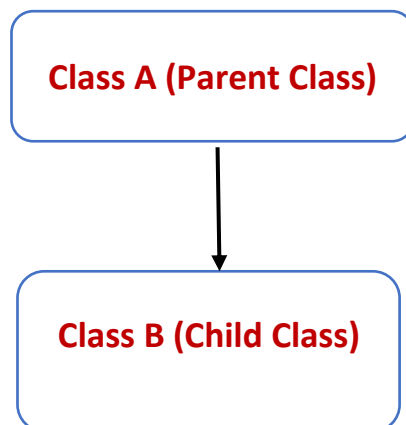
```
    sec obj = new sec();  
    obj.one ();  
    obj.two ();  
    Console.Read ();  
  }  
}  
}
```

Types of Inheritance.

In C# language we have four types of inheritance, they are.

A. Single Level Inheritance.

In this type we will inherit the characteristics (Methods & variables) of specific class to other class, for example if we have Class A and we want to inherit its characteristic to new class named class B, then this operation will be single level inherits, so the class A will call parent class and class B will call child class, it is shown in the following figure:



Example.

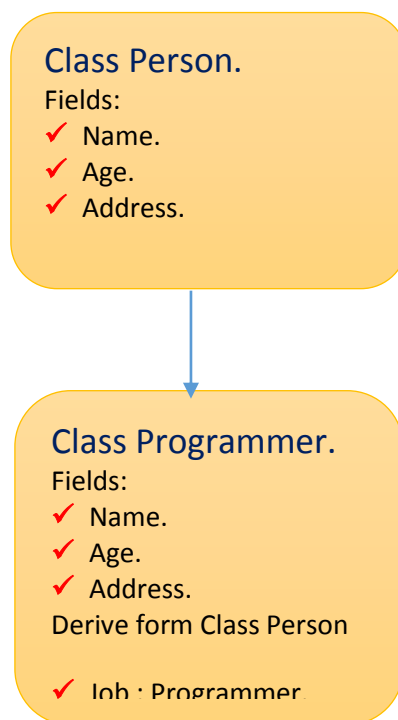
The following example shows the derivation of the above figure, such that in the following code we will inherit class called child from other class called parent.

```
Public Class parent  
{  
    Statement1  
    .
```

```
        .  
        Statement n  
    }  
Public Class child: Inherits parent  
{  
    Statement1  
    .  
    .  
    Statement n  
}
```

Example.

Write program in C# to apply the derivation of the following figure?



```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;
```

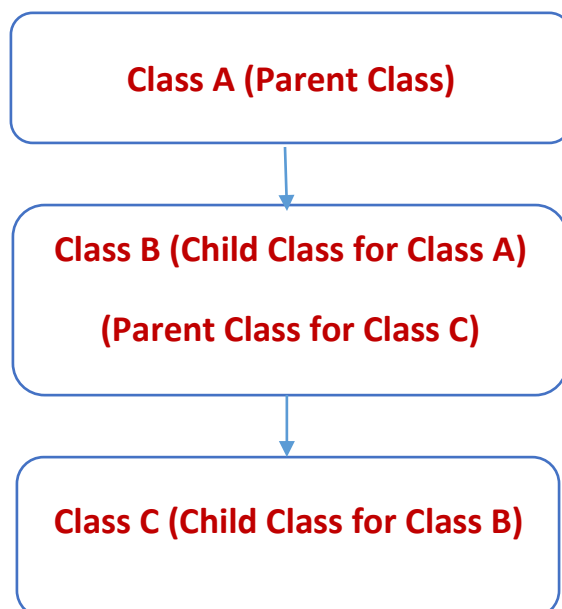
```
namespace ConsoleApplication44
{
    Public class person
    {
        public string name;
        public int age;
        public string address;
        public void set1 (string n, int a, string add)
        {
            name = n;
            age = a;
            address = add;
        }
    }
    class Programmer : person
    {
        public string job;
        public void set2 (string j)
        {
            job = j;
        }
        public void show()
        {
            Console.WriteLine ("Name = " + name);
            Console.WriteLine ("Age = " + age);
            Console.WriteLine ("Address = " + address);
            Console.WriteLine ("job = " + job);
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            programmer call = new programmer ();
            call .set1("Ali", 35, "Baghdad");
        }
    }
}
```

```
call.set2 ("programmer");  
call.show ();  
Console.ReadLine ();  
}  
}  
}
```

B. Multi-Level Inheritance.

It is a derivation operation of characteristics and features of specific class to other class (new class) in program, then the new class will inherit its features to other class, for example if we have class A and give its features to new class names class B, then create new class names class C and give its the features of class B, in that case this type of derivation will call Multi Level Inheritance.

It is shown in the following figure:



Example.

The following example will show the inheritance of above figure, such that in the following code we will inherits class call father from other class call grand_father, and in the same time we will derive class call child from base class called father.

```
Public Class grand_father
{
    Statement1
    .
    .
    Statement n
}
```

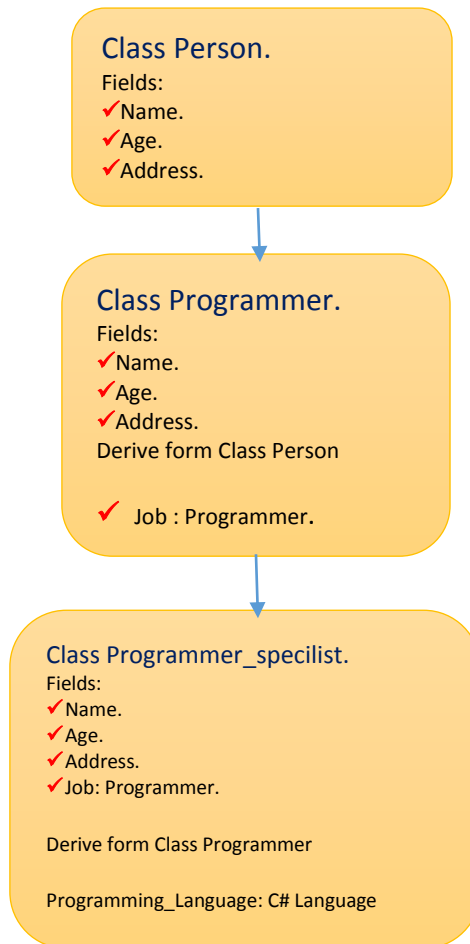
```
Public Class father: Inherits grand_father
{
    Statement1
    .
    .
    Statement n
}
```

```
Public Class child: Inherits father
{
    Statement1
    .
    .
    Statement n
```

```
End Class
}
```

Example.

Write program in C# language to apply the following figure derivation?



```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;
```

```
namespace ConsoleApplication44  
{  
    Public class person  
    {  
        public string name;
```

```
public int age;
public string address;
public void set1 (string n, int a, string add)
    {
        name = n;
        age = a;
        address = add;
    }
}
class Programmer : person
{
    public string job;
    public void set2 (string j)
    {
        job = j;
    }
}
class Programmer_specilist : programmer
{
    public string programming_language;
    public void set3(string pr_In)
    {
        programming_language = pr_In;
    }
    public void show()
    {
        Console.WriteLine ("Name = " + name);
        Console.WriteLine ("Age = " + age);
        Console.WriteLine ("Address = " + address);
        Console.WriteLine ("job = " + job);
        Console.WriteLine ("Programming Language = " +
            programming_language);
    }
}
}
```

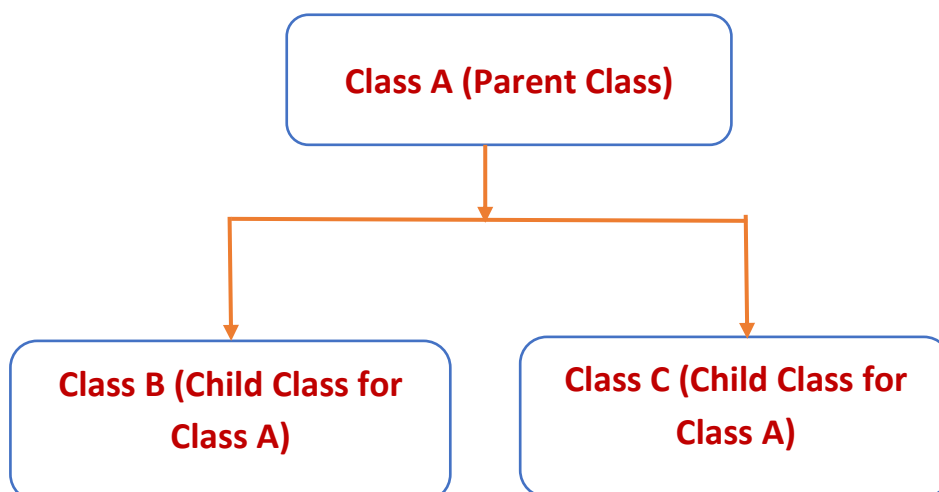
Class Program

```
{
static void Main(string[] args)
{
programmer_specilist call = newprogrammer_specilist ();
    call .set1("Ali", 35, "Baghdad");
    call.set2 ("programmer");
    call.set3 ("C# Language");
    call.show ();
    Console.ReadLine ();
}
}
```

C. Multiple Inherits.

It is a derivation operation of characteristics and features of specific class to more class, suppose we have class named (A) and we want to give its characteristics to two new classes, the first class named (B) and the second class named (C), this technique will be Multiple Inherits type.

So the class (A) will be call parent class, the Class (B) &(C) will be call child class. It is shown in the following figure.



Example.

The following example will show the inheritance of above figure, such that in the following code we will inherits class call child1 from other class call father, and in the same time we will derive class call child2 from base class called father.

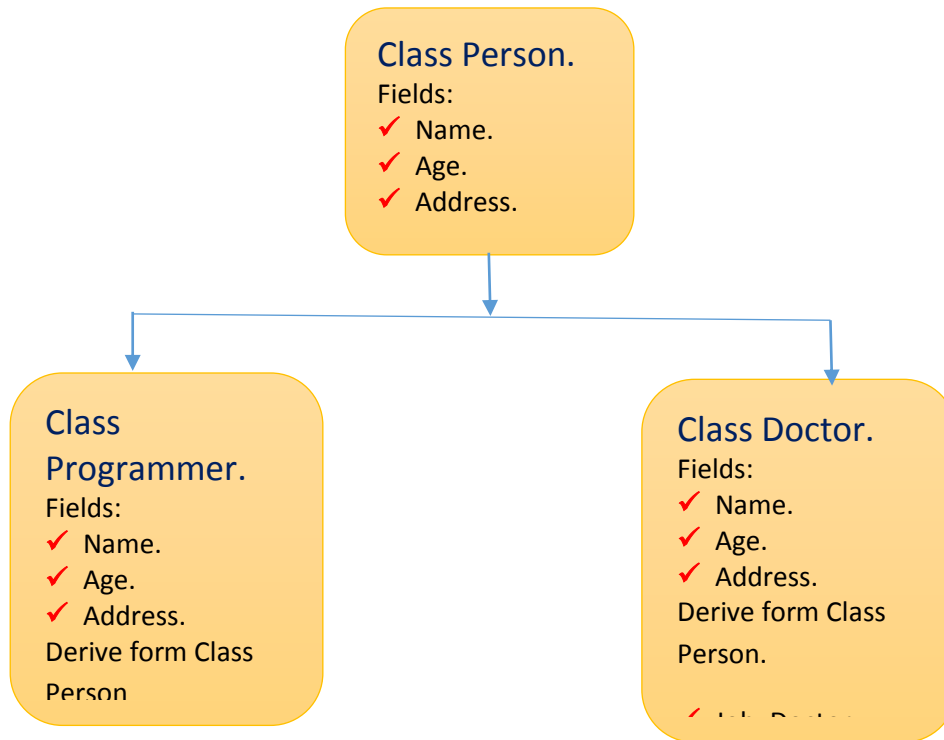
```
Public Class father
{
    Statement1
    .
    .
    Statement n
}
```

```
Public Class Child1: Inherits father
{
    Statement1
    .
    .
    Statement n
}
```

```
Public Class child2: Inherits father
{
    Statement1
    .
    .
    Statement n
}
```

Example.

Write program in C# language to apply the following inheritance?



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

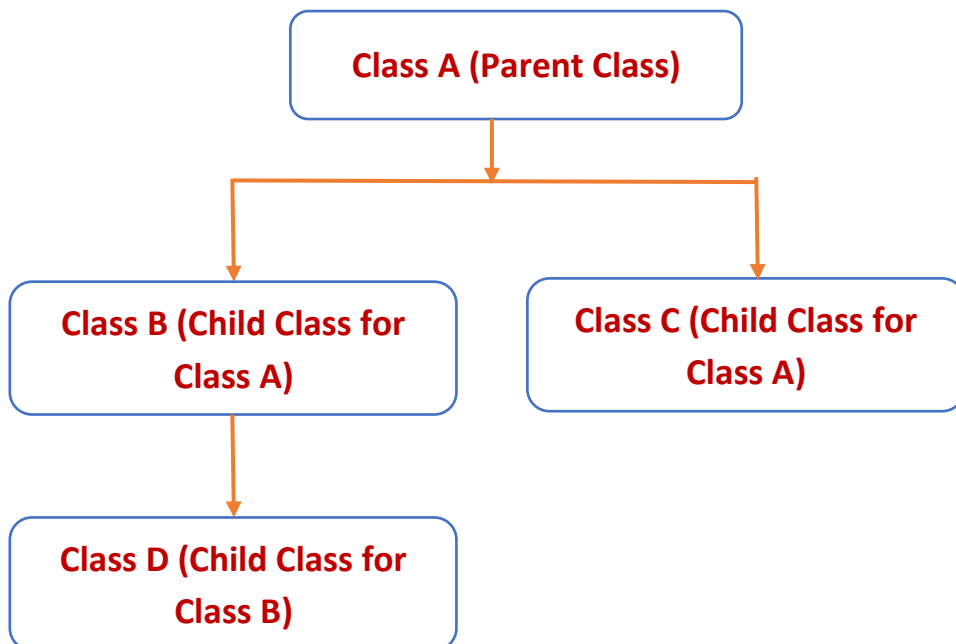
namespace ConsoleApplication44
{
    Public class person
    {
        public string name;
        public int age;
        public string address;
        public void set1 (string n, int a, string add)
        {
            name = n;
```

```
        age = a;
        address = add;
    }
}
class Programmer : person
{
public string job;
public void set2 (string j)
    {
        job = j;
    }
public void show()
    {
        Console.WriteLine ("Name = " + name);
        Console.WriteLine ("Age = " + age);
        Console.WriteLine ("Address = " + address);
        Console.WriteLine ("job = " + job);
    }
}
Class doctor : person
{
    public string job2;
    public void set3(string j2)
        {
            job2 = j2;
        }
    public void show()
        {
            Console.WriteLine ("Name = " + name);
            Console.WriteLine ("Age = " + age);
            Console.WriteLine ("Address = " + address);
            Console.WriteLine ("job = " + job2);
        }
}
```

```
class Program
{
static void Main(string[] args)
{
    programmer call1 = new programmer ();
    doctor call2 = new doctor();
    call1 .set1 ("Ali", 35, "Baghdad");
    call1.set2 ("programmer");
    call1.show ();
    call2.set1 ("Ahmed", 40, "kerbalaa");
    call2.set3 ("doctor");
    call2.show ();
    Console.ReadLine ();
}
}
```

D. Hybrid Inheritance.

It is a mix of previous types, so when we combine two types of inheritance in one type it will be called hybrid inheritance as, it is shown in the following figure:



In the above figure we have a basic class called class A and we can name grandfather class, this class will inherit its features to all other classes founded under it, and there is two classes class (B& class C) are derived from class A, this derivation will be of Multiple derivation.

In the above figure there is another derivation, it is derivation of class D from class B, this type will be as single level derivation.

Note that class D will take the features of class A& the features of class B. i.e. this derivation will be as hybrid derivation, it consists of single level derivation and multiple derivation.

The general structure form of this type is shown in the following example:

Example.

The following example will show the inheritance of above figure, such that in the following code we will inherits class call B from the base class call A, and in the same time we will derive class call C from base class called A, and we will derive class call D from base class called B .

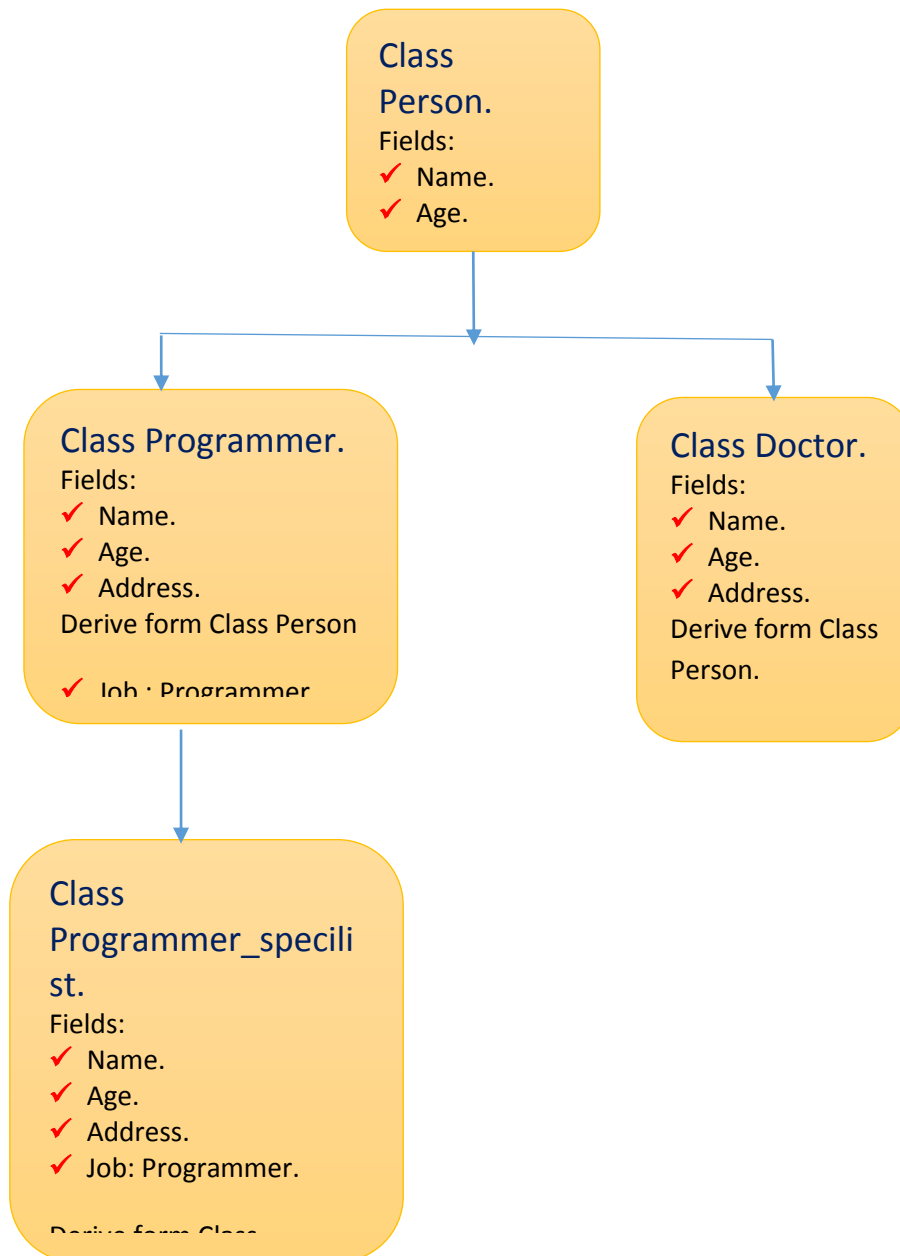
```
Public Class class A
{
    Statement1
    .
    .
    Statement n
}
```

```
Public Class Class B:Inherits Class A
{
    Statement1
    .
    .
    Statement n
}
```

```
}  
Public Class Class C:Inherits Class A  
  {  
    Statement1  
    .  
    .  
    Statement n  
  
  }  
Public Class Class D : Inherits Class B  
  {  
    Statement1  
    .  
    .  
    State mment n  
  
  }  
}
```

Example.

Write program in C# language to apply the inheritance shown in the following figure?



```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

```

```
using System.Threading.Tasks;
```

```
namespace ConsoleApplication44
```

```
{
```

```
Public class person
```

```
{
```

```
    public string name;
```

```
    public int age;
```

```
    public string address;
```

```
    public void set1 (string n, int a, string add)
```

```
    {
```

```
        name = n;
```

```
        age = a;
```

```
        address = add;
```

```
    }
```

```
}
```

```
class Programmer : person
```

```
{
```

```
    public string job;
```

```
    public void set2 (string j)
```

```
    {
```

```
        job = j;
```

```
    }
```

```
}
```

```
class Programmer_specilist : programmer
```

```
{
```

```
    public string programming_language;
```

```
    public void set3(string pr_In)
```

```
    {
```

```
        programming_language = pr_In;
```

```
    }
```

```
    public void show()
```

```
    {
```

```
        Console.WriteLine ("Name = " + name);
```

```
        Console.WriteLine ("Age = " + age);
```

```
        Console.WriteLine ("Address = " + address);
        Console.WriteLine ("job = " + job);
        Console.WriteLine ("Programming Language = " +
        programming_language);
    }
}
Class doctor : person
{
    public string job2;
    public void set4(string j2)
    {
        job2 = j2;
    }
    public void show()
    {
        Console.WriteLine ("Name = " + name);
        Console.WriteLine ("Age = " + age);
        Console.WriteLine ("Address = " + address);
        Console.WriteLine ("job = " + job2);
    }
}
}
class Program
{
    static void Main(string[] args)
    {
        programmer_specilist call1 = newprogrammer_specilist ();
        doctor call2 = new doctor();
        call1 .set1 ("Ali", 35, "Baghdad");
        call1.set2 ("programmer");
        call1.set3 ("C# Language");
        call1.show ();
        call2.set1 ("Ahmed", 40, "kerbalaa");
        call2.set4 ("doctor");
        call2.show ();
    }
}
```

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

Note 1.

We can set classes' un inheritance in C# language by using the keyword (sealed) before the name of class in declaration part, it is shown in the following example:

Sealed class Class_Name

```
{  
  
    Class_Body;  
  
}
```

Note 2.

The purpose of set class un inheritance is to keep at class elements (data & methods) from external using. This reduces errors resulting from the misuse of data.

Example.

Sealed class person

```
{  
public int id;  
public string name;  
public void set1 (int x, string n)  
    {  
        id = x;  
        name = n;  
    }  
}
```

class Program mer : person

Note 3.

You should know that we can create objects for the sealed classes in the main program.

2- Abstract & interface Classes.

It is a representation of specific type (Class) that to be unreal representation, It is known that the abstract classes in object-oriented programming do not allow them to reproduce objects and remain limited in its role in the process of defining the basic elements of the classes in order to inherit to the derived classes, i.e. (the abstract classes is classes which can't copy elements from it).

Abstract category can include functions abstract rechargeable definition at the level of derived classes, also can not include any function abstract, abstract and jobs are jobs that do not contain any software orders, they only display function definition.

Create abstract class.

To create abstract class in C# language we will use the key word (abstract), the general form to create abstract class is.

```
Abstract class Class_Name
```

```
{  
    Body of class;  
}
```

Example.

```
Abstract class person
```

```
{  
    public int id;  
    public string name;  
    public void initialize(int x, string n)
```

```
{
    id = x;
    name = n;
}
}
```

Note.

You should know we can't create objects from this class in the main program, for example.

```
class Program
{
    static void Main(string[] args)
    {
        person obj = new person();
    }
}
```

To deal with this type of class, we should derive its features from other class and treat with derived class; it is shown in the following example.

Example.

Write program in C# to create new class which have information of programmer (id, name, and specialty) and print out this information at screen by using Abstract Class technology?

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```



```
namespace ConsoleApplication45
{
    Abstract class person
    {
        public int id;
        public string name;
        public void set1 (int x, string n)
        {
            id = x;
            name = n;
        }
    }
    class Programmer : person
    {
        public string specility;
        public void set2 (string sp)
        {
            specility = sp;
        }
        public void show()
        {
            Console.WriteLine ("id = " + id);
            Console.WriteLine ("Name = " + name);
            Console.WriteLine ("specialty = " + specility);
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            programmer obj = newprogrammer ();
            obj.set1 (1, "Ali");
            obj.set2 ("Visual Basic");
            obj.show ();
            Console.ReadLine ();
        }
    }
}
```

```
}  
}  
}
```

Interface Classes.

They are using for designing the model of a system.

The interface contains the list of unimplemented methods & defines the responsibility of a class. The general form of interface is:

Interface interface name

```
{  
    Method prototype-1;  
    Method prototype-2;  
    _____  
    _____  
}
```

For implementing the interface on the class, it will be as following:

Class class name: interface name

```
{  
    Body of the class;  
}
```

Note.

If you implement an interface in your class then you must override all the methods whatever is declared inside the interface.

It is shown in the following example:

Example.

```
interface SatyaWork
{
    int max(int a, int b);
    int add(int a, int b);
}
Class Satya: SatyaWork
{
    Public int max (int a, int b)
    {
        Return a > b? a: b;
    }
    Public int add (int a, int b)
    {
        Return a + b;
    }
}
Class Program
{
    Static void Main (string [] args)
    {
        Satya obj = new Satya ();
        Console.WriteLine ("Largest: " + obj.max (10, 20));
        Console.WriteLine ("Sum: " + obj.add (10, 20));
        Console.Read ();
    }
}
```

3- Polymorphism &Overloading.

It is defined more than function in the same name, so we can use this technique to deal with the derived classes as if it's one.

Overloading technique is the practical implementation of Polymorphism.

If class contains more than one method with the same name but different argument, then this term is known as overloading method.

It is shown in the following example:

Example 1.

Write any structure program in C# to show the overloading technology?

Class employee

```
{
Private int id;
Private string name;

public void searchinfo ()
{

}
public void searchinfo (int x)
{

}

public void searchinfo(int x, string name)
{

}
}
```

In the above example, we create three overloading methods that have the same name (searchinfo), the first method with no arguments, the second method with one argument and the third method with two arguments.

If we create object for this class (employee) and we want to call these methods, they will appear as following.



Example 2.

Write program in C# to create an overloaded method named as max () which return the largest value among 2 or 3 given numbers.

Example 3.

Create an overloaded method named as LCM () which return the LCM of either 2, 3 or 4 given values.

Class logic

```

{
int i;
public int lcm(int a, int b)
{
    int m = a * b, res = 0;
    for (i = 1; i <= m; i++)
    {

```

```
        if (i % a == 0 && i % b == 0)
        {
            res = i;
            break;
        }
    }
    return res;
}

public int lcm(int a, int b, int c)
{
    int m = a * b * c, res = 0;
    for (i = 1; i <= m; i++)
    {
        if (i % a == 0 && i % b == 0 && i % c == 0)
        {
            res = i;
            break;
        }
    }
    return res;
}

public int lcm ( int a, int b, int c , int d)
{
    int m = a * b * c*d, res = 0;
    for (i = 1; i <= m; i++)
    {
        if (i % a == 0 && i % b == 0 && i % c == 0 && i % d == 0)
        {
            res = i;
            break;
        }
    }
    return res;
}
}
```

```
class Program
{
    static void Main(string[] args)
    {
        logic obj = new logic();
        Console.WriteLine ("Lcm of 2 Values: " + obj.lcm (10, 2));
        Console.WriteLine ("Lcm of 3 Values: " + obj.lcm (4, 2,6));
        Console.WriteLine ("Lcm of 4 Values: " + obj.lcm (2, 3, 5, 7));
        Console.Read ();
    }
}
```

Note.

We can use overloading technology for a contractor methods, it is shown in the following example.

Example.

In the following example we will built two contractor overloading methods, the first called (employee) does not have parameters, and the second has the same name but with parameters, all these methods follow to the employee class.

Class employee

```
{
    Private int id;
    Private string name;

    public employee ()
    {

    }

    public employee (int x, string n)
    {
```

```
this.id = x;  
this.name = n;  
}  
}
```

4- Encapsulation.

It is a technique in which allows the programmer to keep at privacy of the data and prevents the user from direct access to keep it from the mistakes of external intervention, so in this technique we will define this data as a private data, and in the same time we will access to this data through the member methods.

It is shown in the following example:

Example.

In the following program will shows the encapsulation technology, such that we will define two variables (name & id) and give them values then we will print out at monitor screen, by using this technology?

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;  
  
namespace ConsoleApplication47  
{  
    Class employee  
    {  
        Private int id;  
        Private string name;  
  
        public void set_data (int i , string n)  
        {  
            id = i;
```



```
        name = n;
    }

    public void show_data()
    {
        Console.WriteLine (id);
        Console.WriteLine (name);
    }

}

class Program
{
    static void Main(string[] args)
    {
        employee emp = new employee();
        emp.set_data (1, "Ali");
        emp.show_data ();
        Console.ReadLine ();
    } } }
```

Overriding Method.

If we define a method in a derived class which signature exactly matched with any other method of base class, then this term is known as overriding method.

It is shown in the following example:

Example.

In the following example we will show overriding method technology.

Class first

```
{
    Public virtual void msg ()
```

```
{
    Console.WriteLine ("This is The Method of First Class");
}
}
Class sec: first
{
    Public override void msg ()
    {
        base.msg ();
    }
}
Console.WriteLine ("This is The Method of Second Class");
}
}
Class Program
{
    Static void Main (string [] args)
    {
        Sec obj = new sec ();
        obj.msg ();
    }
}
Console. Read ();
} }
```

Exception Handling.

They are errors and exceptions management operations which appear in run time, so for manage these errors. C# language allows to deal with it through specific structure called Try...Catch. So the general form for using this structure is.

Maybe there are 3 types of errors In C# program could be generated, these are:

1. Compile time errors.
2. Logical Errors.
3. Run Time Errors.

- 1- Compile time errors are traced by the compiler and without debugging this, your program could not execute.
- 2- Logical errors are occurred due to invalid logic of the programmer and this could be traced by a programmer.
- 3- Run Time Errors- are known as exceptions and when they occurred they terminated the execution a program.

Exception Structure.

To apply exception In C# program, we will use the following structure.

Try

{

}

Catch Orders that we want to validate;

{

Orders that we want to be executed when the error appearance;

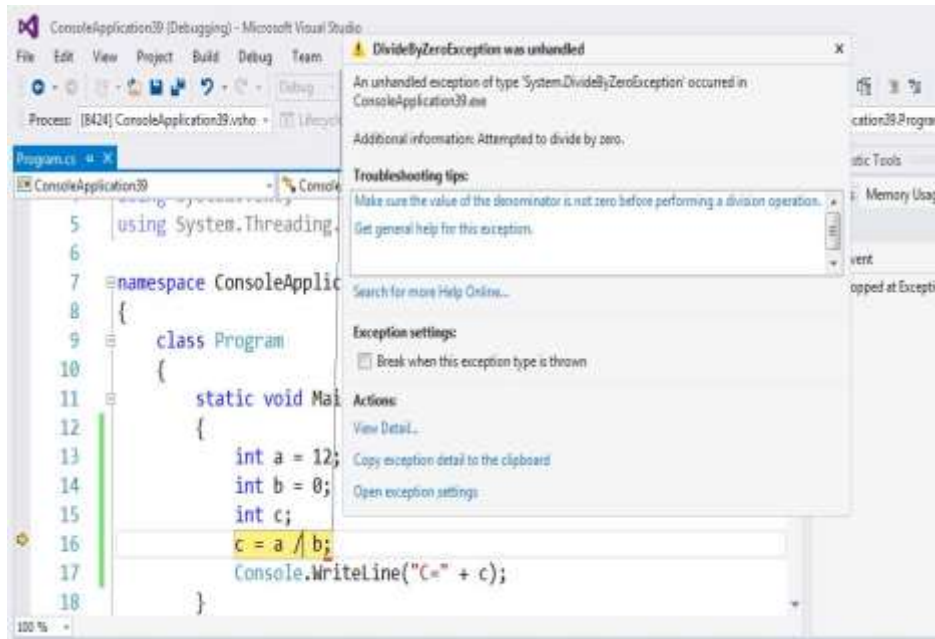
}

The feature of this technique is without this technique the program execution will stop when simplest error appears in run time, but when we use this technique the Implementation of the program will continue even after the error has appeared.

To understand this technique notes the following example:

Example.

Suppose we want to divide two numbers (a & b) and we want to put the result in (C) the variable (a) has specific value, but the variable (b) has (0), in the division operation we know that we can't divide by (0) therefore error will appear as in following.



This error called compiling run time error when the error appears, the execution of program will stop, to continue implementation of the program will use technology of Try ... Catch. It is shown in the following example.

Example.

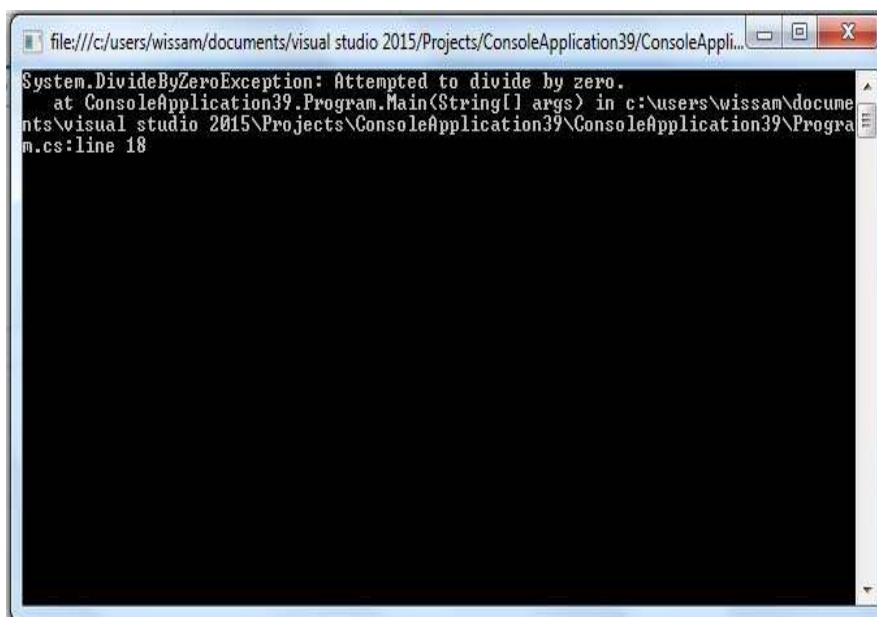
Suppose we want to implement of the same previous example but with the exception handling technology?

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace ConsoleApplication39
{
    class Program
    {
        static void Main(string[] args)
        {
```

```
try
{
    int a = 12;
    int b = 0;
    int c;
    c = a / b;
    Console.WriteLine ("C=" + c);
}
catch (Divide By Zero Exception ex)
{
    Console.WriteLine (ex.ToString ());
}
Console.ReadLine ();
}
}
```

When we execute this program, error will appear in the program execution window.



Note.

We can write more than one Catch within program, it is shown in the following code.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace ConsoleApplication39
{
class Program
{
    static void Main(string[] args)
    {
        try
        {
            int a = 12;
            int b = 0;
            int c;
            c = a / b;
            Console.WriteLine ("C=" + c);
        }
        catch (Divide By Zero Exception ex)
        {
            Console.WriteLine (ex.ToString ());
        }
        catch (Arithmetic Exception ar)
        {
            Console.WriteLine (ar.ToString ());
        }
        catch (Index Out Of Range Exception ind)
        {
            Console.WriteLine (ind.ToString ());
        }
    }
}
```

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

Note.

There is an optional part in the Try...Catch structure technology, it called (Finally).

It is written after the Catch Part, so when the code has written inside them, there will be implemented in every way (If the error occurred or did not occur), it is shown in the following example:

Example.

Suppose we want to develop the previous example by adding (Finally) part to this code that has written command to print out specific message at screen.

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;  
  
namespace ConsoleApplication39  
{  
    class Program  
    {  
        static void Main(string[] args)  
        {  
            try  
            {  
                int a = 12;  
            }  
        }  
    }  
}
```

```
        int b = 0;
        int c;
        c = a / b;
        Console.WriteLine ("C=" + c);
    }
catch (Divide By Zero Exception ex)
    {
        Console.WriteLine (ex.ToString ());
    }
catch (Arithmetic Exception ar)
    {
        Console.WriteLine (ar.ToString ());
    }
catch (Index Out Of Range Exception ind)
    {
        Console.WriteLine (ind.ToString ());
    }
finally
    {
        Console.WriteLine ("this code will implementation always");
    }
Console.ReadLine ();
    }
}
```


Chapter 4

File Management in C#

File Management in C# Language.

What's File?

It is a collection of data stored in a disk with a specific name and specific directory path.

What is stream?

When a file opened for reading or writing, it becomes a stream, so what is a stream? It is a sequence of bytes passing through the communication path.

I/O Class Types.

In C# language there are three classes' types of I/O, they are.

1- Stream Reader Class.

It is using for receiving data from programmer/user and inserts it to program as pure data to process it, there are two types of reading statement in C# language, and they are.

- A. ReadLine Statement.
- B. ReadToEnd Statement.

2- Stream Writer Class.

It is using to read the results from C# Program and print out at screen to become execution results for programmer/user, so there are two types of writing statements in C# language, they are:

- A. Write Statement.
- B. WriteLine Statement.

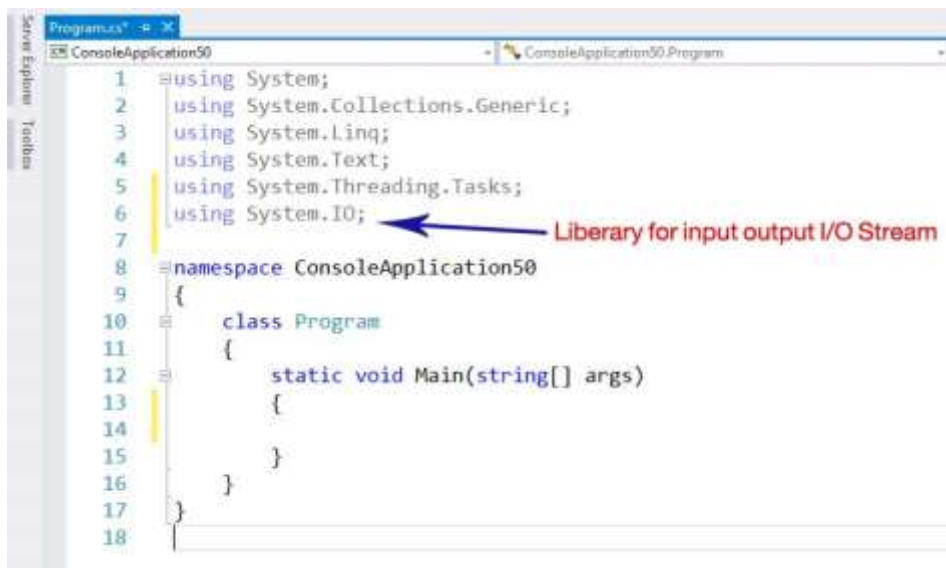
3- File Access Class.

These are several classes using to deal with files (Open File, Close File, Read from File, Write to File). Like.

- A. ReadAllLines.
- B. ReadAllText.
- C. ReadLines.
- D. WriteAllLines.
- E. WriteAllText.
- F. AppendAllText.
- G. AppendAllLines.

Note.

If we want to use these classes in C# language, we should use the library (System.IO) which is written in the start of program, it is shown in the following figure.



The image shows a screenshot of a C# code editor window. The code is as follows:

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6 using System.IO;
7
8 namespace ConsoleApplication50
9 {
10     class Program
11     {
12         static void Main(string[] args)
13         {
14
15         }
16     }
17 }
18
```

A red arrow points from the text "Library for input output I/O Stream" to the line `using System.IO;` on line 6.

Create File Object.

To read contains of a specific file.

We should firstly create object to deal with the reading file orders by using Stream Reader Class, the general form of create object is.

```
StreamReader Object_Name = New StreamReader ("File Full Path");
```

Example.

Suppose we have a text file on the desk top named ("wissam.txt"), we can create object to it as following.

```
StreamReader f = new StreamReader  
("C:/Users/Wissam/Desktop/wissam.txt");
```

StreamReader Class.

1- ReadLine statement.

This statement is using to read the first line of the text from file, if we want to read all the texts founded in this file with this statement (ReadLine), we should use loop with this statement, it is shown in the following program:

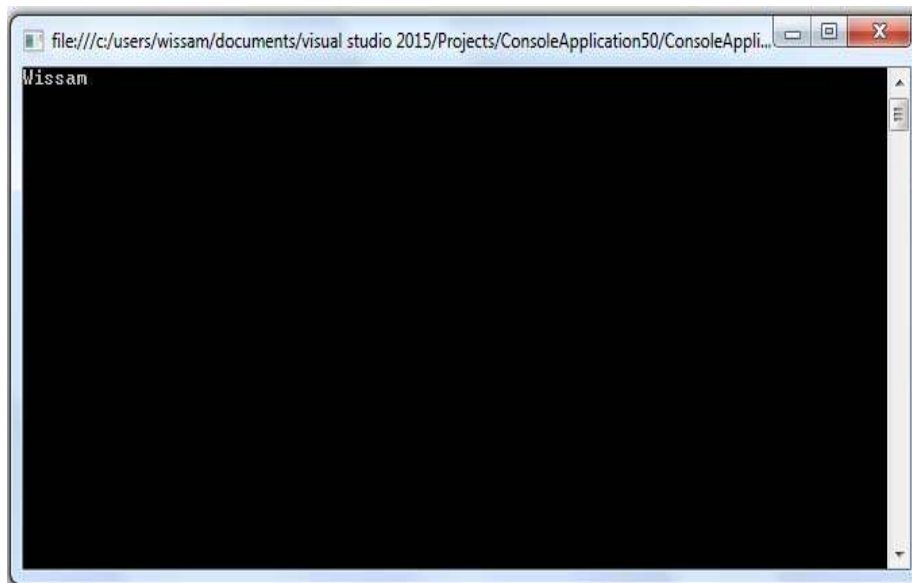
Example 1.

Suppose we have specific file in the desk top named ("Wissam.txt") and we want to read the first line in this file, write program in C# to do that?

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;  
using System.IO;
```

```
namespace ConsoleApplication50
{
class Program
    {
static void Main(string[] args)
    {
        StreamReader f = new StreamReader
("C:/Users/Wissam/Desktop/wissam.txt");
        Console.WriteLine (f.ReadLine ());
        Console.ReadLine ();
    }
}
}
```

When we execute this program, the following result will appear.



Example 2.

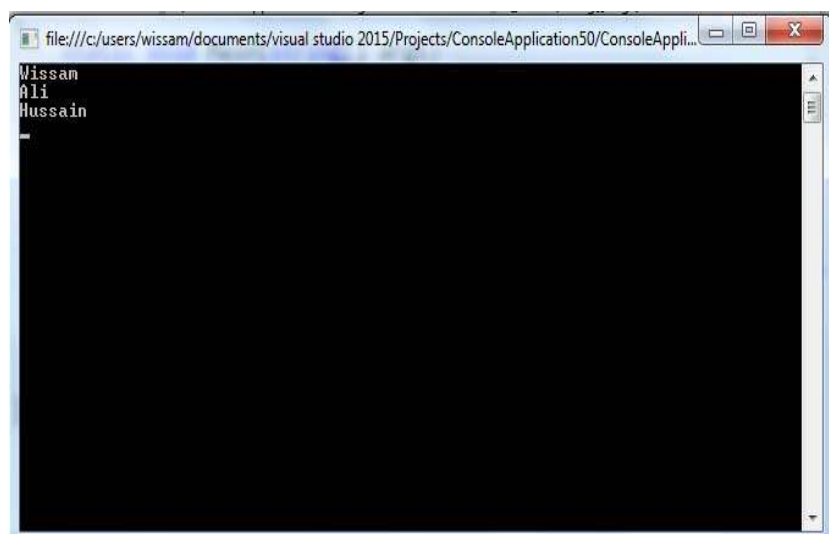
Rewrite the previous example to read all the texts within file by using (ReadLine) statement?

using System;

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;

namespace ConsoleApplication50
{
    class Program
    {
        static void Main(string[] args)
        {
            StreamReader f = new StreamReader
            ("C:/Users/Wissam/Desktop/wissam.txt");
            string line;
            while ((line=f.ReadLine()) != null )
            {
                Console.WriteLine (line);
            }
            Console.ReadLine ();
        }
    }
}
```

When we execute the above program, we will get the following results.

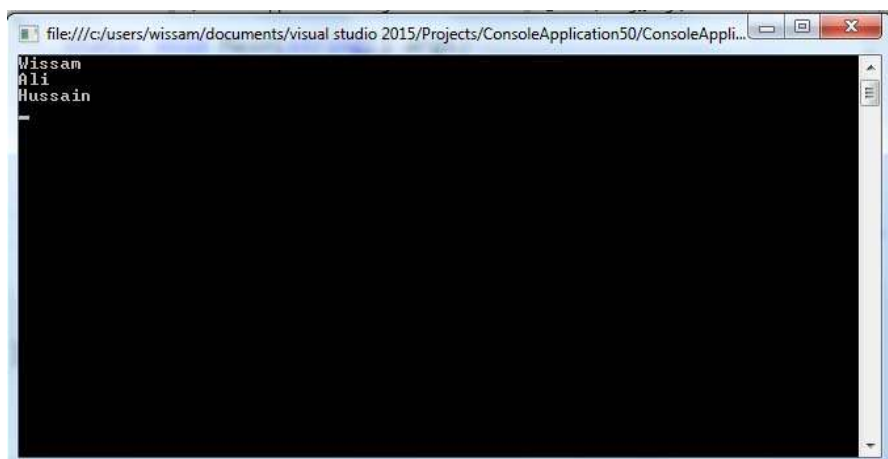


2- ReadToEnd Statement.

We use While Loop statement to read all texts within file, and we can replace this loop by ReadToEnd order, it is shown in the following example.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;

namespace ConsoleApplication50
{
class Program
{
static void Main(string[] args)
{
StreamReader f = new StreamReader
("C:/Users/Wissam/Desktop/wissam.txt");
Console.WriteLine (f.ReadToEnd ());
Console.ReadLine ();
}
}
}
```



StreamWriter Class.

It is using to insert texts within file with clear the old text, to use these classes we should create object to use write statements (by using StreamWriter Class), it is same object of read Statement, for example.

```
StreamReader Object_Name = New StreamReader ("File Full Path");
```

Example.

Suppose we have a text file on the desk top named ("wissam.txt"), we can create object to it like the following.

```
StreamWriter f = new StreamWriter  
("C:/Users/Wissam/Desktop/wissam.txt");
```

1- Write Statement.

It is using to write one line to the file, and if we want to write more than one line we should repeat this statement, so all the inserted texts will be at the same line, it is shown in the following example:

Note.

When we open any file for reading or writing we should close this file after we have finished from it and close the connection with it, in C# language we can do that through the Dispose order, so without this statement the file will be suspended. It is shown in the following example.

```
Object_Name.Dispose ( );
```

Example.

```
f2.Dispose ();
```

Example.

Suppose we have text file at a desktop named ("Wissam.txt") and we want to write within this file two statements ("Computer Science") & ("Computer Engineering") by using write Statement?

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;

namespace ConsoleApplication50
{
    class Program
    {
        static void Main(string[] args)
        {
            StreamWriter f2 = new StreamWriter
            ("C:/Users/Wissam/Desktop/wissam.txt");
            f2.Write ("computer Science");
            f2.Write ("computer engineering");
            f2.Dispose ();
            Console.ReadLine ();
        }
    }
}
```

2- WriteLine Statement.

It is using to write one line to the file, and if we want to write more than one line we should repeat this statement, so all the inserted texts will be at different lines, it is shown in the following example:

Example.

Suppose we have text file at a desktop named ("Wissam.txt") and we want to write within this file two statements ("Computer Science") &("Computer Engineering") by using WriteLine Statement?

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;

namespace ConsoleApplication50
{
    class Program
    {
        static void Main(string[] args)
        {
            StreamWriter f2 = new StreamWriter
            ("C:/Users/Wissam/Desktop/wissam.txt");
            f2.WriteLine ("computer Science");
            f2.WriteLine ("computer engineering");
            f2.Dispose ();
            Console.ReadLine ();
        }
    }
}
```

3- File Access Class.

They are several classes using to deal with files (Open File, Close File, Read from File, and Write to file...etc.). Like.

A. ReadAllLines Statement.

This order is using to read all lines within the file (Return results as lines), and store the results in string array defined by the user, it follows to specific class in C# called (File Class) , to use this order in C# language we will use the following general form.

```
String_Array_Distination = File.ReadAllLines ("Text File Full Path ");
```

Example.

Suppose we have text file at a desktop named ("Wissam.txt") and we want to read all the text within this file by using ReadAllLines Statement, write program in C# to do that?

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;  
using System.IO;
```

```
namespace ConsoleApplication50  
{  
class Program  
{  
static void Main(string[] args)  
{  
string [] str =File .ReadAllLines("C:/Users/Wissam/Desktop/wissam.txt");  
foreach (var item in str )  
{
```

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

Note.

We note that in above example we don't need to use (Dispose order) to close connection with the file, so in this type it will disconnect automatically.

B. ReadAllText Statement.

This order is using to read all texts within the file (Return results as text (one package)), and store the results in string variable defined by the user, it is followed to specific class in C# called (File Class) , to use this order in C# language we will use the following general form.

```
String_Variable_Distination =File.ReadAllLines ("Text File Full Path");
```

Example.

Suppose we have text file at a desktop named ("Wissam.txt") and we want to read all the text within this file by using ReadAllText Statement, write program in C# to do that?

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;
```

```
using System.IO;

namespace ConsoleApplication50
{
class Program
    {
static void Main(string[] args)
    {
        string str =File .ReadAllText
        ("C:/Users/Wissam/Desktop/wissam.txt");
        Console.WriteLine (str);
        Console.ReadLine ();
    }
}
}
```

C- ReadLines Statement.

It is using to return line after line as a list (not array) from specific text file, so it also called by using (File Class), the general form of using this statement is.

Destination as a list = File.ReadLines ("Text File Full Path").ToList ();

Example.

Suppose we have text file at a desktop named ("Wissam.txt") and we want to read all the text within this file by using ReadLines Statement, write program in C# to do that?

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;
```

```
namespace ConsoleApplication50
{
class Program
{
static void Main(string[] args)
{
List<string> lst =File. ReadLines
("C:/Users/Wissam/Desktop/wissam.txt").ToList ();
Console.WriteLine (lst);
Console.ReadLine ();
}
}
}
```

D- WriteAllLines Statement.

It is using to write many strings at many lines in specific text file, this statement is working by (File Class), the general form of using this statement in C# language is.

```
File.WriteAllLines ("Text File Full Path ", Array of string);
```

Example.

Write program in C# language to write full string to specific text file in desktop called ("Wissam.txt") by using WriteAllLines statement?

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;
```

```
namespace ConsoleApplication50
```

```
{
class Program
{
static void Main(string[] args)
{
string [] str = { "Baghdad", "Kerbalaa", "Babylon", "Najif" };
File.WriteAllLines ("C:/Users/Wissam/Desktop/wissam.txt", str);
Console.ReadLine ();
}
}
}
```

Note.

You must know that the use of this instruct will lead to delete old information in the file and print new information within it.

E- WriteAllText Statement.

It is using to write one string as one package at one line in specific text file, this statement is working by (File Class), so the using of this statement will lead to delete the old information in the text file and print the new information within this file.

The general form of using this statement in C# language is:

```
File.WriteAllLines ("Text File Full Path ", specific string as one package);
```

Example.

Write program in C# language to write string as a one package to specific text file in desktop called ("Wissam.txt") by using WriteAllText statement?

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```



```
using System.IO;

namespace ConsoleApplication50
{
class Program
    {
static void Main(string[] args)
    {
        File.WriteAllText ("C:/Users/Wissam/Desktop/wissam.txt", "how are you I hope you
are fine");
        Console.ReadLine ();
    }
}
}
```

F- AppendAllText Statement.

It is using to append specific strings as one package at the same line in specific text file, this statement is working by (File Class), so the using of this statement will lead to keep the old information in the text file and add the new information within this file.

The general form of using this statement in C# language is:

```
File.AppendAllText ("Text File Full Path ", specific string as one package);
```

Example.

Write program in C# language to add string as a one package to specific text file in desktop called ("Wissam.txt") by using AppendAllText statement?

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
using System.IO;

namespace ConsoleApplication50
{
class Program
    {
static void Main(string[] args)
    {
        File.AppendAllText ("C:/Users/Wissam/Desktop/wissam.txt"," Rely I
miss you");
        Console.ReadLine ();
    }
}
}
```

G- AppendAllLines Statement.

This statement is using to append array of string in the form of lines in specific text file, with keeping the old information in the old file, it is followed to the (File Class).

The general form of using this statement is:

```
File.AppendAllLines ("Text File Full Path", array of string);
```

Example.

Write program in C# language to add array of string to specific text file in desktop called ("Wissam.txt") by using AppendAllLines statement?

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;
```

```
namespace ConsoleApplication50
{
class Program
{
static void Main(string[] args)
{
string [] str = { "Ahmed", "Ali", "Hassa" };
File.AppendAllLines ("C:/Users/Wissam/Desktop/wissam.txt", str);
Console.ReadLine ();
}
}
}
```

Home Work Questions.

Question 1.

Write program in C# language to read two values (X & Y), then multiply these numbers without using the multiply operation?

Question2.

Write program in C# language to read two values (X & Y), then division these numbers without using the division operation?

Question3.

Write program in C# language to read (20) value from keyboard, find the summation of even numbers and odd numbers?

Question4.

Write program in C# language to read (20) value from keyboard, find the summation of positive numbers and negative numbers?

Question5.

Write program in C# language to sort elements of one dimensional array have (10) elements, then print out the sorted array (increasing sort) at computer screen?

Question6.

Write program in C# language to sort elements of one dimensional array have (10) elements, then print the sorted array (decreasing sort) at computer screen?

Question 7.

Write program in C# language to read two dimensional array have (3-row) and (3-column), then change the elements of the first row with the elements of third row?

Question8.

Write program in C# language to read two dimensional array have (3-row) and (3-column), sort the elements of these array columns increasing?

Question9.

Write program in C# language to read string from keyboard and print out every word at independent line?

Question 10.

Write program in C# language to read string from keyboard and print out the number of the vowels at the computer screen?

Question 11.

Write program in C# language to read string from keyboard and print out the central character in this string?

Question 12.

Write program in C# language to read string from keyboard and inverse the first character and latest character from every word?

Question 13.

Write program in C# language to read string from keyboard and print out the first character from every word?

Question 14.

Write program in C# language to read string from keyboard and print out the latest character from every word?

Question 15.

Write program in C# language to read two arrays everyone have (5) elements, contact them, then print out the result at computer screen?

Question 16.

Write program in C# language to read one dimensional array have (10) elements, print out the largest value and smallest value in this array?

Question 17.

Write program in C# language to read one dimensional array have (10) elements, fill these array by elements, then print out the repetition number it in this array?

Question 18.

Write program in C# language to compute (X^n) by using recursive function?

Question 19.

Write program in C# language to read several numbers from keyboard. Find the summation of latest two numbers?

Question20.

Write program in C# language to read two dimensional array have (3-row) and (3-column), convert to one dimensional array?

Question21.

write a program in C# language to read student mark from key board if the mark ≥ 90 and < 100 print out (A) , if the mark ≥ 80 and < 90 print out (B) , if the mark ≥ 70 and < 80 print out (C) , if the mark ≥ 60 and < 70 print out (D) , if the mark ≥ 50 and < 60 print out (E) , if the mark < 50 print out (FAILER) , by using nested if statement?

Question22.

Write program in C# language to read special character from key board and read tow variables (x & y) if the special character is (+) add (x & y) if (-) subtract (x & y) if (*) multiplication (x & y) if (/) division (x & y) otherwise print out "special character" by using case statement?

Question23.

Write program in C# language to enter two lengths (L1 & L2), find the average of the two lengths, and print out the largest length?

Question24.

Write program in C# language to read (20) values from keyboard and find the summation of them?

Question25.

Write program in C# language to read (10) values from keyboard and find the largest and smallest value?

Question26.

Write program in C# language to enter one dimensional array have (10) elements and print out the positive and negative numbers?

Question27.

Write program in C# language to read tow dimensional arrays have (3 row) and

(3 column) and find the average of each row?

Question28.

Write program with C# language to find the sum and average of (4) students have (3) degree and find the sum of the average?

Question29.

Write program in C# language to read tow dimensional array have 3-row and 3-culomn find the summation of the main diagonal?

Question30.

Write program in C# language to read string from key board and find the length of the string?

Question31.

Write program in C# language to read string and print the remained line after find the first space by using pointer?

Question32.

Write program in C# language to exchange the value of (x) by value of (y) and value of (y) by value of (x) by using the function technology, and using call by actual parameter?

Question33.

Write program in C# language to exchange the value of (x) by value of (y) and value of (y) by value of (x) by using the function technology, and using call by formal parameter?

Question34.

Write program in C# language to exchange the value of (x) by value of (y) and value of (y) by value of (x) by using the function technology, and using call by reference?

Question35.

Write program in C# language to solution the following equation by using function?

$$Z = \frac{a * b + d}{x + y^2 - b}$$

Question36.

Write program in C# language to found (N!) by using recursive function?

Question37.

write program in C# language to read tow array (A) and (B) everyone is one dimensional array has (10) elements, then add the elements of the tow Arrays and put the result in array name (C) , use for every step function ?

Question38.

Write program in C# language to read string capital latter and convert it to small latter?

Question39.

Write program in C# to create an overloaded method named as max () which return the largest value among 2 or 3 given numbers?

Question40.

Write program in C# to create new class which have information of programmer (id, name, and specialty) and print out these information at screen by using Abstract Class technology?

Question41.

Create a method named as fact () which contains a number as argument & return the factorial of that number?

Question42.

Create a method named as power () which contains base number(x) and power number (n) as argument & return x to the power n?

Question43.

Write program in C# language to create a Method named as table () which contains a number as argument and print the table of given number?

Question44.

Suppose we have specific file in the desk top name ("Wissam.txt") and we want to read the first line in this file, write program in C# to do that?

Question45.

Suppose we have text file at a desktop name ("Wissam.txt") and we want write within this file two statements ("Computer Science") & ("Computer Engineering") by using write Statement?



C.V

Personal Information.

Name: Wissam Ali Hussein Salman Al Kuzaey.

Birth day: Baghdad / 14-10-1977.

Gender: Male.

Marital: married.

Graduation: B.Sc. in computer Science / Baghdad University / 2003.

M.Sc. in computer Science / Sam Higgin Buttem Institute
of technology – India / 2013.

M.Sc. Topic: Design & Implementation of Student Information
Management System for Karbala University.

Specialization: Data Base & Cloud Computing.

Scientific Title: Assistant Lecturer.

Address: Republic of Iraq / Holly Karbala City / AL-hur district.

Phone No. : 00964 7724918820.

Email: wesali77@yahoo.com.

wissamali77@gmial.com.

Languages: Arabic Language + English Language.

Certificates.

1. B.Sc. in Computer Science \ Baghdad University \ graduation year 2002-2003.
2. M.Sc. In Computer Science \ Sam Higgin Buttem Institute of technology \ India \ graduation year 2013.
3. IC3 International certificate with estimation (very good).
4. TOEFL Certificate in English Language \ Karbala University.

5. Web Design Certificate from Microsoft Academy \ Qadiseya University branch.
6. Database Design Certificate from Microsoft Academy \ Qadiseya University branch.
7. Geographic Information System (GIS) Certificate from (RTI) Organization \ Holly Karbala.
8. Java Programming Language Certificate \ Crest Institute \ Hyderabad \ India.
9. SQL Server Certificate \ from IVAIS institute \ Hyderabad \ India.
10. Visual Basic.Net \ Crest Institute \ Hyderabad \ India.
11. English Certificate \ IELETES guru Institute \ Hyderabad \ India.
12. Database Administrator 1 Certificate \ Oracle University.
13. Database Administrator 2 Certificate \ Oracle University.
14. HTML + Java Script Certificate \ Banchab IT Collage \ India.
15. Web Design Certificate by using ASP.net \ Banchab IT Collage \ India.
16. Laptop maintenance \ SCS Indian Institute \ Hyderabad \ India.
17. I got certificate pass of teaching methods course from Ministry of higher education / university of Kerbela.
18. I got certificate for Web Application & Hacking Security from AL_Kafeel Institute of Information technology and developing skills.

Membership organizations.

1. I am Member of the Iraqi Teachers' Union.
2. I am Member of the Association of Iraqi programmers.
3. I am member of the teaching staff of educational association in Iraq.
4. I am member of the Scientific Forum for SHIATS University / India.
5. I am member of NIIT Computer Academy Education & Training Center / Hyderabad / India.

