University of Anbar

College of Computer Science and Information Technology

Computer Network Systems Depts.

Visual Programming (C#)

First Semester (2022-2023)

Third Stage

Dr. Ismail Taha Ahmed

Dr. Baraa Tareq Hammad

Chapter One

C# Overview

1.1 Introduction

C# (C-Sharp) is a programming language developed by Microsoft that runs on the .NET Framework. C# is used to develop web apps, desktop apps, mobile apps, games and much more. C# is pronounced "C-Sharp". It is an object-oriented programming language created by Microsoft. C# has roots from the C family, and the language is close to other popular languages like C++ and Java.

The first version was released in year 2002. The latest version, **C# 10**, was released in November 2021.

C# is used for:

- Mobile applications
- Desktop applications
- Web applications
- Web services
- Web sites
- Games
- VR
- Database applications
- And much, much more!

Why Use C#?

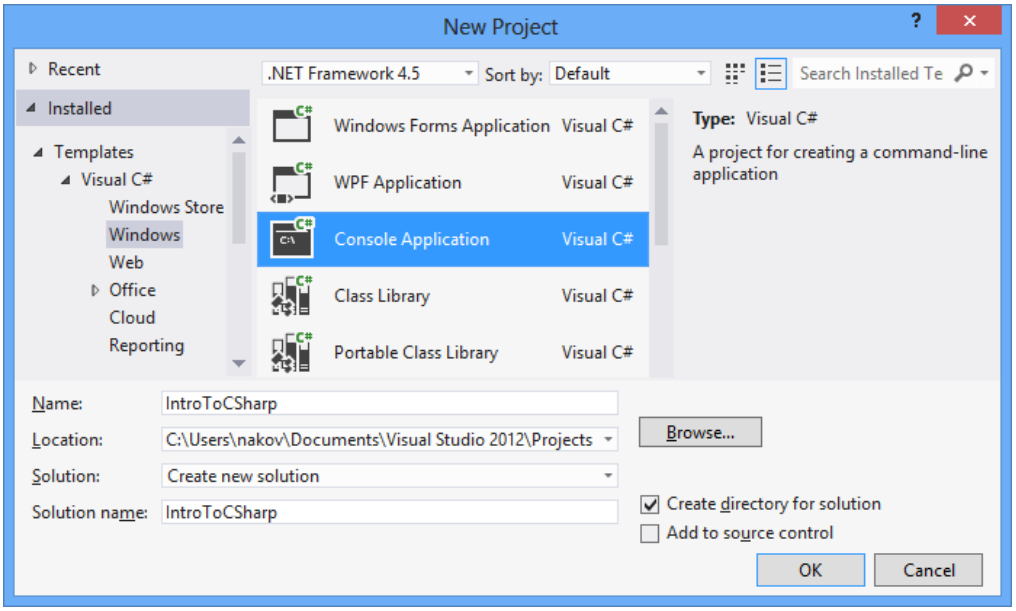
- It is one of the most popular programming language in the world
- It is easy to learn and simple to use
- It has a huge community support
- C# is an object oriented language which gives a clear structure to programs and allows code to be reused, lowering development costs
- As C# is close to C, C++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

Console Application

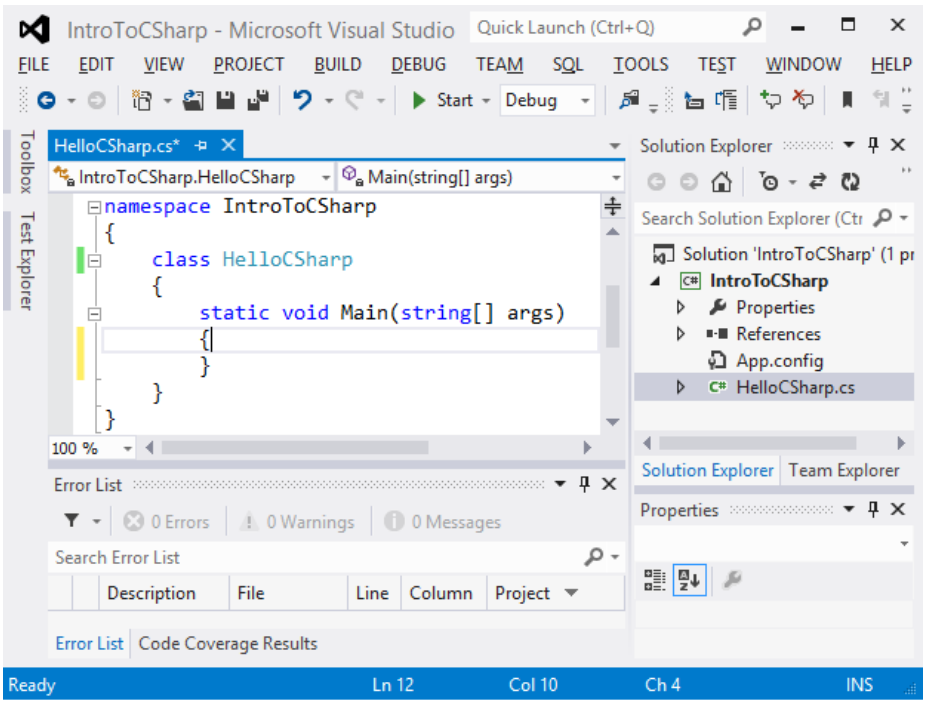
This introduction introduces console applications.

Creating the Console Application

1- First open Visual C# studio
2- Select File > New Project… to display the New Project dialog

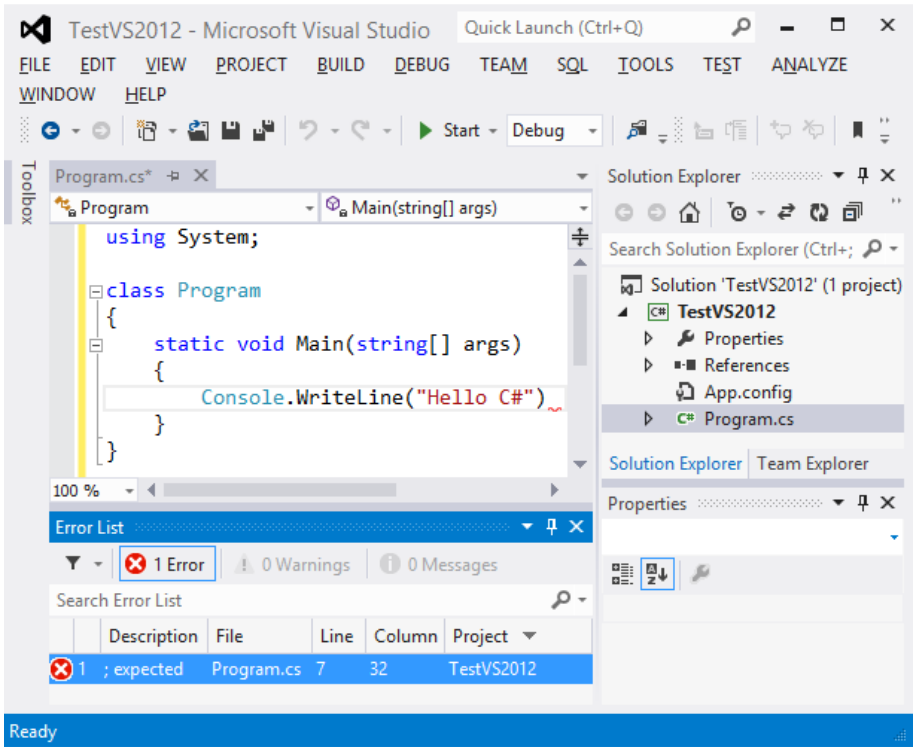


3- Then select the Console Application template. In the dialog's Name field, type Welcome1.
4- Click OK to create the project.
5- You can customize the colors shown in the code editor by selecting Tools > Options…. This displays the Options dialog. Then expand the Environment node and select Fonts and Colors. Here you can change the colors for various code elements.

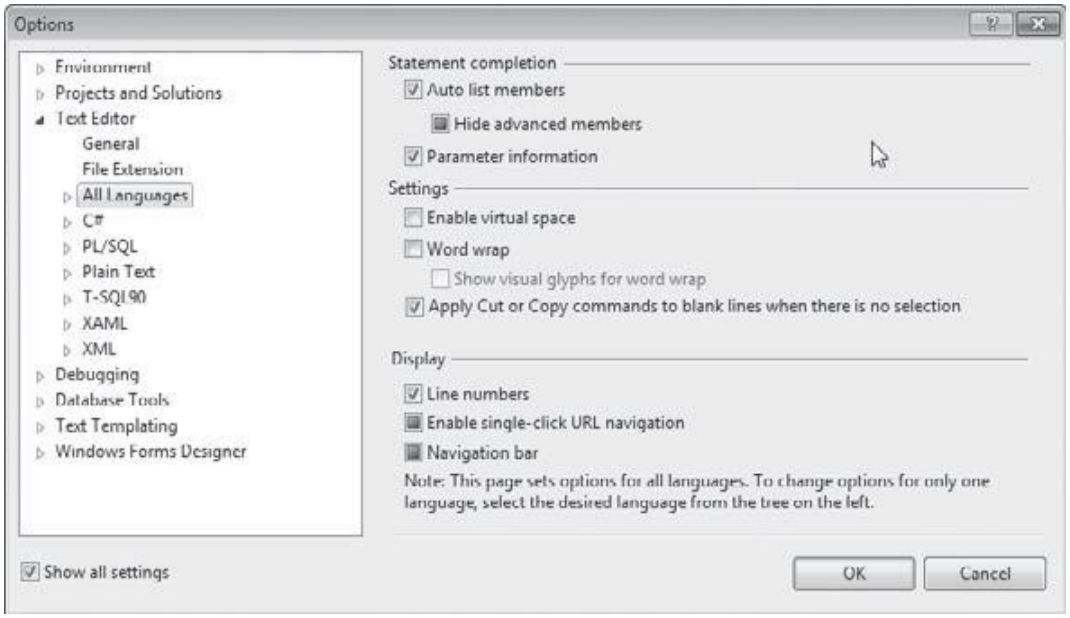


Modifying the Editor Settings to Display Line Numbers

1- Select Tools > Options….

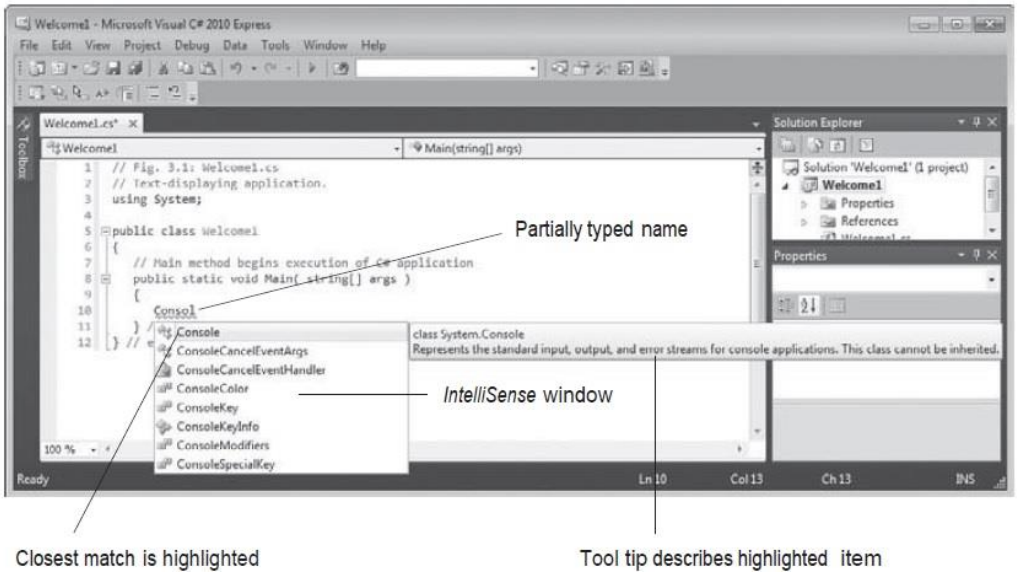


2- Click the Show all settings checkbox on the lower left of the dialog, then expand the Text Editor node in the left pane and select All Languages. On the right, check the Line numbers check-box. Keep the Options dialog open.



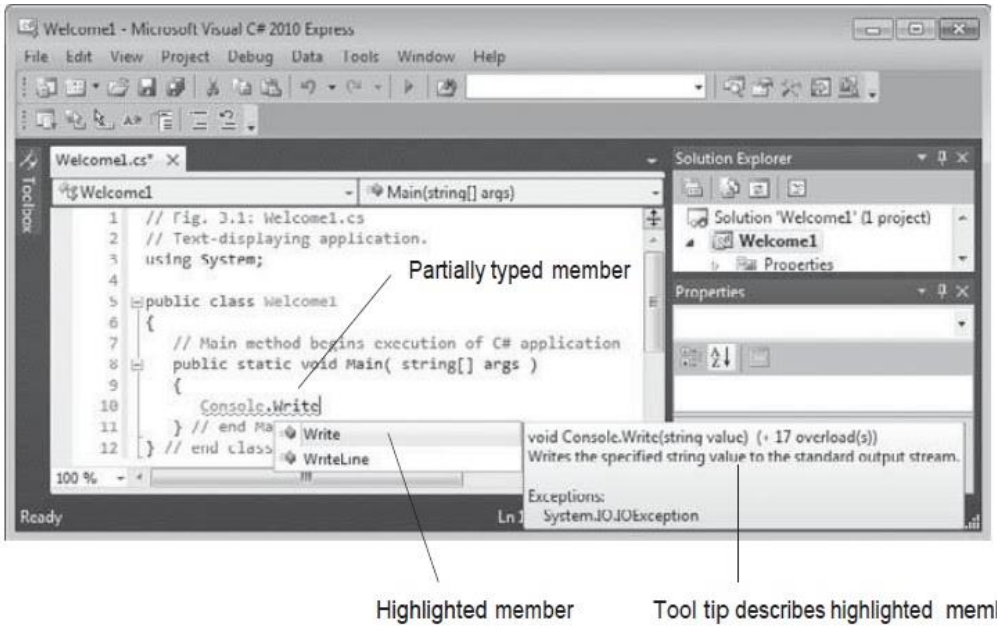
Writing Code and Using IntelliSense

1- In the editor window, type the code an IntelliSense window containing a scrollbar is displayed.



2- This IDE feature lists a class's members, which include method names. When you type the dot (.) after Console, the IntelliSense window reappears and shows only

the members of class Console that can be used on the right side of the dot. When you type the open parenthesis character, (, after Console.WriteLine, the Parameter Info window is displayed. This window contains information about the method's parameters.

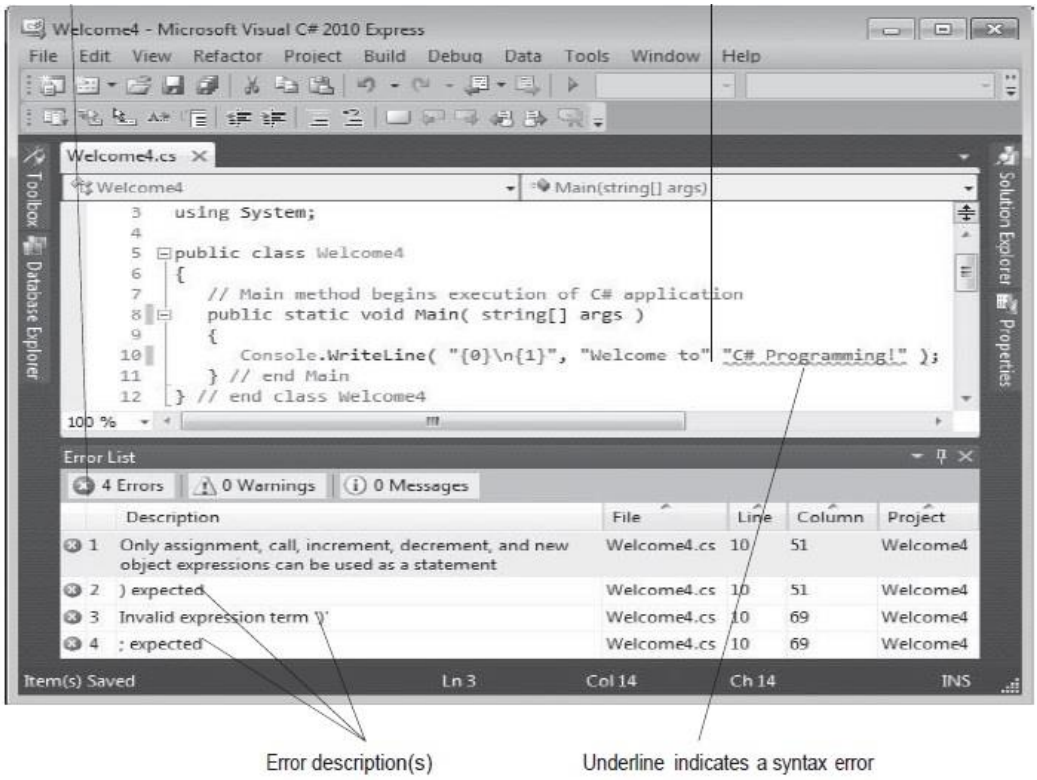


Compiling and Running the Application

1- Select Debug > Build Solution. If the application contains no syntax errors, this will compile your application and build it into an executable file (named Welcome1.exe, in one of the project's subdirectories). To execute it, type Ctrl + F5, which invokes the Main method.

2- (If you attempt to run the application before building it, the IDE will build the application first, then run it only if there are no compilation errors.) The statement in line 10 of Main displays Welcome to C# Programming!. The results of executing this application, displayed in a console (Command Prompt) window.

Syntax Errors, Error Messages and the Error List Window



1.2 C# Program Beginning

When we consider a C# program, it can be defined as a collection of objects that communicate via invoking each other's methods. Let us now briefly look into what a class, object, methods, and instant variables mean.

- **Object** − Objects have states and behaviors. Example: A dog has states - color, name, breed as well as behaviors - wagging, barking, eating. An object is an instance of a class.
- **Class** − A class can be defined as a template/blueprint that describes the behaviors/states that object of its type support.
- **Methods** − A method is basically a behavior. A class can contain many methods. It is in methods where the logics are written, data is manipulated and all the actions are executed.
- **Instance Variables** − each object has its unique set of instance variables. An object's state is created by the values assigned to these instance variables.

1.3 C# Program Structure

Let us look at a simple code that would print the words **Welcome to C# Programming!**.

Example 1.1:	**Output**
1 // welcome1.cs 2 // Text-displaying application. 3 using System; 4 5 public class Welcome1 6 { 7 // Main method begins execution of C# application 8 public static void Main(string[] args) 9 { 10 Console.WriteLine("Welcome to C# Programming!"); 11 } // end Main 12 } // end class Welcome1	Welcome to C# Programming!

Let us look at the various parts of the above program: -

Line 1 &2: Single-line comments start with two forward slashes (//).Any text between // and the end of the line is ignored by C# (will not be executed). Multi-line comments start with /* and ends with */. Any text between /* and */ will be ignored by C#.

Line 3: using System means that we can use classes from the System namespace.

Line 4: A blank line. C# ignores white space. However, multiple lines makes the code more readable.

Line 5: class is a container for data and methods, which brings functionality to your program. Every line of code that runs in C# must be inside a class. In our example, we named the class Welcome1.

Line 6 & 9: The curly braces {} marks the beginning and the end of a block of code.

Line 8: Another thing that always appear in a C# program, is the Main method. Any code inside its curly brackets {} will be executed. You don't have to understand the keywords before and after Main. You will get to know them bit by bit while reading this tutorial.

Line 10: Console is a class of the System namespace, which has a WriteLine() method that is used to output/print text. In our example it will output **"Welcome to C# Programming!"**.

Example 1.2: Displaying a Single Line of Text with Multiple Statements	**Output**
1 // Welcome2.cs 2 // Displaying one line of text with multiple statements. 3 using System; 4 5 public class Welcome2 6 { 7 // Main method begins execution of C# application 8 public static void Main(string[] args) 9 { 10 Console.Write("Welcome to "); 11 Console.WriteLine("C# Programming!"); 12 } // end Main 13 } // end class Welcome2	Welcome to C# Programming!

Example 1.3: Displaying Multiple Lines of Text with a Single Statement	**Output**
1 // Welcome1.cs 2 // Displaying multiple lines with a single statement. 3 using System; 4 5 public class Welcome3 6 { 7 // Main method begins execution of C# application 8 public static void Main(string[] args) 9 { 10 Console.WriteLine("Welcome\nto\nC#\nProgramming!"); 11 } // end Main 12 } // end class Welcome3	Welcome to C# Programming!

There are certain characters in C# when they are preceded by a backslash, they will have special meaning and they are used 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 (six spaces)
\v	Vertical tab

Example 1.4: Formatting Text with Console.Write and Console.WriteLine	**Output**
`1 // Welcome4.cs` `2 // Displaying multiple lines of text with string formatting.` `3 using System;` `4` `5 public class Welcome4` `6 {` `7 // Main method begins execution of C# application` `8 public static void Main( string[] args )` `9 {` `10 Console.WriteLine( "{0}\n{1}", "Welcome to", "C# Programming!" );` `11 } // end Main` `12 } // end class Welcome4`	`Welcome to` `C# Programming!`

1.4 Get User Input

You have already learned that Console.WriteLine() is used to output (print) values. Now we will use Console.ReadLine() to get user input.

In the following example, the user can input his or hers username, which is stored in the variable userName. Then we print the value of userName:

Example

```
// Type your username and press enter
Console.WriteLine("Enter username:");

// Create a string variable and get user input from the
keyboard and store it in the variable
string userName = Console.ReadLine();

// Print the value of the variable (userName), which will
display the input value
Console.WriteLine("Username is: " + userName);
```

The Console.ReadLine() method returns a string. Therefore, you cannot get information from another data type, such as int. The following program will cause an error:

Example

```
Console.WriteLine("Enter your age:");
int age = Console.ReadLine();
Console.WriteLine("Your age is: " + age);
```

The error message will be something like this:

Cannot implicitly convert type 'string' to 'int'

It is also possible to convert data types explicitly by using built-in methods, such as **Convert.ToBoolean, Convert.ToDouble, Convert.ToString, Convert.ToInt32 (int) and Convert.ToInt64 (long):**

Example

```
int myInt = 10;
double myDouble = 5.25;
bool myBool = true;

Console.WriteLine(Convert.ToString(myInt));    // convert int
to string
Console.WriteLine(Convert.ToDouble(myInt));    // convert int
to double
Console.WriteLine(Convert.ToInt32(myDouble));  // convert
double to int
Console.WriteLine(Convert.ToString(myBool));// convert bool
to string
```

Example

```
Console.WriteLine("Enter your age:");
int age = Convert.ToInt32(Console.ReadLine());
Console.WriteLine("Your age is: " + age);
```

1.5 C# Character Set

C# has the letters and digits, as show below:

Uppercase: **A, B, C ….. Z.**

Lowercase: **a, b, c… … z.**

Digit**:** **0….9.**

1.6 C# Identifiers

All C# **variables** must be **identified** with **unique names**.

These unique names are called **identifiers**.

Identifiers can be short names (like x and y) or more descriptive names (age, sum, totalVolume).

Note: It is recommended to use descriptive names in order to create understandable and maintainable code:

The general rules for naming variables are:

- Names can contain letters, digits and the underscore character (_)
- Names must begin with a letter
- Names should start with a lowercase letter and it cannot contain whitespace
- Names are case sensitive ("myVar" and "myvar" are different variables)
- Reserved words (like C# keywords, such as `int` or `double`) cannot be used as names

1.7 C# Keywords

The following list shows the reserved words in C#. These reserved words may not be used as constant or variable or any other identifier names.

asm	else	new	this
auto	enum	operator	throw
bool	explicit	private	true
break	export	protected	try
case	extern	public	typedef
catch	false	register	typeid
char	float	reinterpret_cast	typename
class	for	return	union
const	friend	short	unsigned
const_cast	goto	signed	using
continue	if	sizeof	virtual
default	inline	static	void
delete	int	static_cast	volatile
do	long	struct	wchar_t
double	mutable	switch	while
dynamic_cast	namespace	template	

1.8 C# Data Types

As explained in the variables chapter, a variable in C# must be a specified data type:

Example

```
int myNum = 5;               // Integer (whole number)
double myDoubleNum = 5.99D;  // Floating point number
char myLetter = 'D';         // Character
bool myBool = true;          // Boolean
string myText = "Hello";     // String
```

A data type specifies the size and type of variable values.

It is important to use the correct data type for the corresponding variable; to avoid errors, to save time and memory, but it will also make your code more maintainable and readable. The most common data types are:

Data Type	Size	Description
int	4 bytes	Stores whole numbers from -2,147,483,648 to 2,147,483,647
long	8 bytes	Stores whole numbers from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	4 bytes	Stores fractional numbers. Sufficient for storing 6 to 7 decimal digits
double	8 bytes	Stores fractional numbers. Sufficient for storing 15 decimal digits
bool	1 bit	Stores true or false values
char	2 bytes	Stores a single character/letter, surrounded by single quotes
string	2 bytes per character	Stores a sequence of characters, surrounded by double quotes

1.9 Variables

Variables are containers for storing data values. To create a variable, you must specify the type and assign it a value:

```
type variableName = value;
```

Where *type* is a C# type (such as `int` or `string`), and *variableName* is the name of the variable (such as **x** or **name**). The **equal sign** is used to assign values to the variable.

To create a variable that should store text, look at the following example:

Create a variable called **name** of type `string` and assign it the value "**John**":

```
string name = "John";
Console.WriteLine(name);
```

Create a variable called **myNum** of type `int` and assign it the value **15**:

```
int myNum = 15;
Console.WriteLine(myNum);
```

You can also declare a variable without assigning the value, and assign the value later:

```
int myNum;
myNum = 15;
Console.WriteLine(myNum);
```

Change the value of `myNum` to 20:

```
int myNum = 15;
myNum = 20; // myNum is now 20
Console.WriteLine(myNum);
```

1.10 Constants

If you don't want others (or yourself) to overwrite existing values, you can add the `const` keyword in front of the variable type.

This will declare the variable as "constant", which means unchangeable and read-only:

```
const int myNum = 15;
myNum = 20; // error
```

The `const` keyword is useful when you want a variable to always store the same value, so that others (or yourself) won't mess up your code. An example that is often referred to as a constant, is PI (3.14159...).

Note: You cannot declare a constant variable without assigning the value. If you do, an error will occur: A const field requires a value to be provided.

1.11 Arithmetic Operations

To perform arithmetic operations, C# language provides the binary operators:

Operator	**Description**	**Example** **Assume A=10 and B= 20**
+	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

Example 1.5: A C# program that add, subtract, multiply and divide two numbers.

Output

```
using System;

namespace Tutlane
{
    class Program
    {
        static void Main(string[] args)
        {
            int result;
            int x = 20, y = 10;
            result = (x + y);
            Console.WriteLine("Addition Operator:
" + result);
            result = (x - y);
            Console.WriteLine("Subtraction
Operator: " + result);
            result = (x * y);
            Console.WriteLine("Multiplication
Operator: "+ result);
            result = (x / y);
            Console.WriteLine("Division Operator:
" + result);
            result = (x % y);
            Console.WriteLine("Modulo Operator: "
+ result);
            Console.WriteLine("Press Enter Key to
Exit..");
            Console.ReadLine();
        }
    }
}
```

Example 1.6: different program to solve the same issue above.

Output

```
1.  using System;
2.
3.  namespace Operator
4.  {
5.      class ArithmeticOperator
6.      {
7.              public static void Main(string[]
    args)
8.              {
9.                      double firstNumber = 14.40,
    secondNumber = 4.60, result;
10.                     int num1 = 26, num2 = 4, rem;
```

```
11.
12.                 // Addition operator
13.                 result = firstNumber +
    secondNumber;
14.                 Console.WriteLine("{0} + {1} =
    {2}", firstNumber, secondNumber, result);
15.
16.                 // Subtraction operator
17.                 result = firstNumber -
    secondNumber;
18.                 Console.WriteLine("{0} - {1} =
    {2}", firstNumber, secondNumber, result);
19.
20.                 // Multiplication operator
21.                 result = firstNumber *
    secondNumber;
22.                 Console.WriteLine("{0} * {1} =
    {2}", firstNumber, secondNumber, result);
23.
24.                 // Division operator
25.                 result = firstNumber /
    secondNumber;
26.                 Console.WriteLine("{0} / {1} =
    {2}", firstNumber, secondNumber, result);
27.
28.                 // Modulo operator
29.                 rem = num1 % num2;
30.                 Console.WriteLine("{0} % {1} =
    {2}", num1, num2, rem);
31.             }
32.     }
33. }
```

Example 1.7: Write a C# Sharp program to print the output of multiplication of three numbers which will be entered by the user.	**Output**

```
using System;
public class Exercise6
{
public static void Main()
{
int num1, num2, num3;
Console.Write("Input the first number to multiply:
");
 num1 = Convert.ToInt32(Console.ReadLine());
Console.Write("Input the second number to
multiply: ");
num2 = Convert.ToInt32(Console.ReadLine());
Console.Write("Input the third number to multiply:
");
```

```
num3 = Convert.ToInt32(Console.ReadLine());
int result = num1 * num2 * num3;
Console.WriteLine("Output: {0} x {1} x {2} = {3}",
num1, num2, num3, result);
}
}
```

1.12 Decision Making: Equality and Relational Operators

There are following relational operators supported by C# language. 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.

Example 1.8: a C# program to explain the Relational operators.

Output

```
1. using System;
2.
3. namespace Operator
4. {
5.     class RelationalOperator
6.     {
7.             public static void Main(string[]
   args)
8.             {
9.                     bool result;
10.                    int firstNumber = 10,
   secondNumber = 20;
11.
12.                    result =
   (firstNumber==secondNumber);
13.                    Console.WriteLine("{0} == {1}
   returns {2}",firstNumber, secondNumber, result);
14.
15.                    result = (firstNumber >
   secondNumber);
16.                    Console.WriteLine("{0} > {1}
   returns {2}",firstNumber, secondNumber, result);
17.
18.                    result = (firstNumber <
   secondNumber);
19.                    Console.WriteLine("{0} < {1}
   returns {2}",firstNumber, secondNumber, result);
20.
21.                    result = (firstNumber >=
   secondNumber);
22.                    Console.WriteLine("{0} >= {1}
   returns {2}",firstNumber, secondNumber, result);
23.
24.                    result = (firstNumber <=
   secondNumber);
25.                    Console.WriteLine("{0} <= {1}
   returns {2}",firstNumber, secondNumber, result);
26.
27.                    result = (firstNumber !=
   secondNumber);
28.                    Console.WriteLine("{0} != {1}
   returns {2}",firstNumber, secondNumber, result);
29.             }
30.     }
31. }
```

1.13 Logical Operators

There are following logical operators supported by C# language.

Assume variable A holds 1 and variable B holds 0, then:

Operator	Description	Example
&&	Called Logical AND operator. If both the operands are non-zero, then condition becomes true.	(A && B) is false.
||	Called Logical OR Operator. If any of the two operands is non-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.

1.14 Bitwise Operators

Bitwise operator works on bits and perform bit-by-bit operation. The truth tables 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

For Example:

Assume 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++ language 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 form 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

1.15 Assignment Operators

There are following assignment operators supported by C#language:

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

Chapter Three

Selection Statement

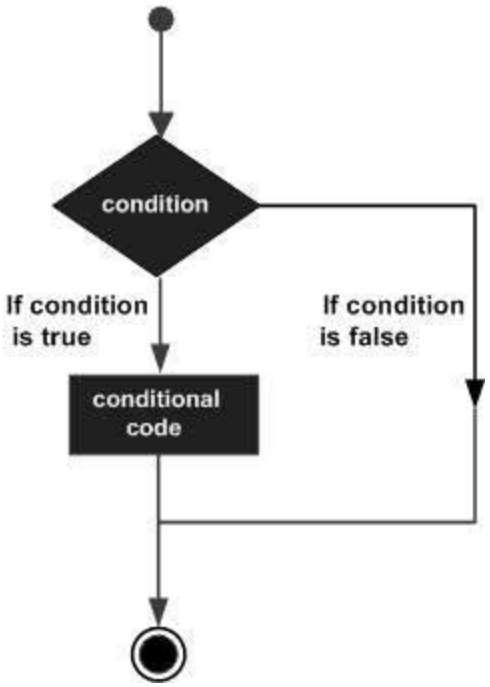
3.1 Selection Statement

C++ provides three selection structure**: if, if/else, switch.**

3.2 The Single if Statement Structure

The syntax of an if statement in C++ is

```
if (Boolean expression)
Statement1;
```



Example: if (avrq >= 3.5)

cout << "good";

Example: cin >> num;

If (num ==0)

zcount = zcount +1;

Example 3.1: Write a C++ program to print the value less than 20.	**Output**
`#include <iostream>` `using namespace std;` `int main ()` `{` `int a = 10;` `if( a < 20 )` `{` `cout << "a is less than 20" << endl;` `}` `cout << "value of a is : " << a << endl;` `return 0;` `}`	a is less than 20 value of a is : 10

3.3 The Single if Statement Structure (Blocks)

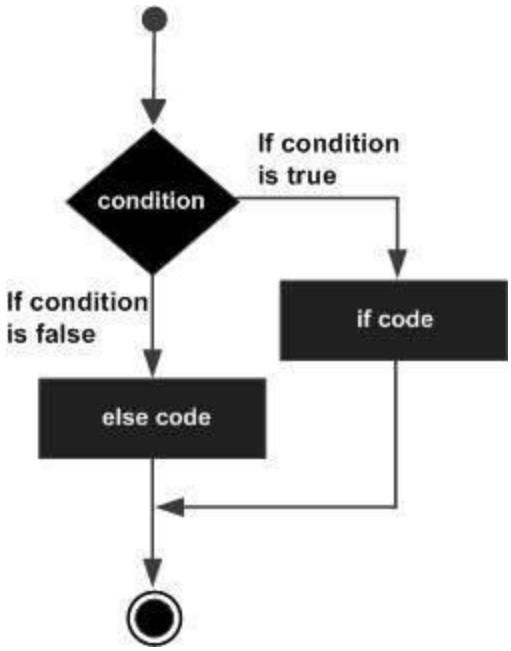
```
if (Boolean expression)
{
  Statement1;
  Statement2;
  Statement3;
}
```

Example 3.2: Write C++ program to read a number and check if it's positive, if it's so prints it, add it a total, and decrement by2.	**Output**
`#include <iostream>` `using namespace std;` `int main () {` `  int num, total;` `  cin >> num;` `  if (num>=0)` `  {` `  cout << num <<"is a positive";` `  total+=num;` `  num=num-2;` `  }` `  return 0;` `  }`	46 is a positive

3.4 The if/else Statement Structure

The syntax of an if...else statement in C++ is

```
if (Boolean expression)
{
 statement1;
}
else {statement2;
}
```



Example 3.3: Write C++ program to read a student degree, and check if it's degree greater than or equal to 50, then print pass, otherwise print fail.	**Output**
`#include <iostream>` `using namespace std;` `int main () {` `  int degree;` `  cin>>degree;` `  if( degree >= 50 )` `  cout << "pass";` `  else` `  cout << "fail";` `  }`	45 fail

Example 3.4: Write C++ program to read a number, and check if it's even or odd.	**Output**
`#include <iostream>` `using namespace std;` `int main () {` `  int num;` `  cin >> num;` `  if( num%2==0 )` `  cout << "even";` `  else` `  cout << "odd";`	26 Even

}	

3.5 Nested if and if/else Statements:

if /else statement:

```
if (expression or condition 1) Statements1 ;
else if (expression or condition 2) Statements2 ;
.
.
else if (expression or condition n) Statements n ;
else Statement;
```

Example 3.5: Write C++ program to print the grades.	**Output**
`#include <iostream>` `using namespace std;` `int main () {` `int value;` `cin>>value;` `if( value==0 ) cout<< "grade is A";` `else if( value==1 ) cout<< "grade is B";` `else if( value==2 ) cout<< "grade is C";` `else cout<<"grade is X";` `}`	2 grade is C
Example 3.6: Write a C++ program to print the days of week.	**Output**
`#include <iostream>` `using namespace std;` `int main () {` `int day;` `cin>>day;` `if( day==1 ) cout<< "Sunday";` `else if( day==2 ) cout<< "Monday";` `else if( day==3 ) cout<< "Tuesday";`	4 Wenesday

```
  else if( day==4 ) cout<< "Wednesday";
  else if( day==5 ) cout<< "Thursday";
  else if( day==6 ) cout<< "Friday";
  else if( day==7 ) cout<<"Saturday";
  else cout<<"unknown day";

}
```

Example 3.7: Write C++ program to compute the value of Z according to the following equations:

$$Z = \begin{cases} x+5 & : x < 0 \\ cos(x)+4 & : x = 0 \\ \sqrt{x} & x > 0 \end{cases}$$

Output

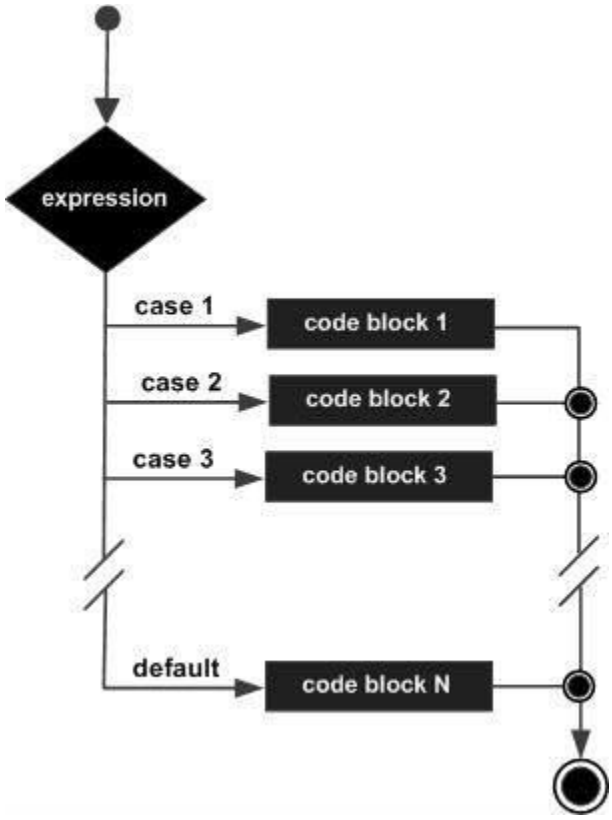
H. W

```
#include <iostream>
#include<math.h>
using namespace std;

int main () {
     int Z, x;
     cout <<"enter x value \n";
     cin >> x;
 if (x<0) Z= x+5;
 else if (x==0)
 Z=cos(x)+4;
 else Z=sqrt(x);
 cout <<" Z is "<< Z;

 }
```

3.6 The Switch Selection Statements (Selector)

The syntax for a **switch** statement in C++ is as follows

```
switch(selctor) {

   case constant-expression1:

      statement(s);

      break;

   case constant-expression:

      statement(s);

      break;

   default:
```



Example 3.8: Write a C++ program to print the evaluation and grade.	**Output**
`#include <iostream>` `using namespace std;` `int main () {` `  char grade = 'D';` `  switch(grade) {`	`You passed` `Your grade is D`

```
      case 'A' :
         cout << "Excellent!" << endl;
         break;
      case 'B' :
      case 'C' :
         cout << "Well done" << endl;
         break;
      case 'D' :
         cout << "You passed" << endl;
         break;
      case 'F' :
         cout << "Better try again" << endl;
         break;
      default :
         cout << "Invalid grade" << endl;
   }
   cout << "Your grade is " << grade << endl;

   return 0;

}
```

Example 3.9: Write the C++ program to implement the mathematical operation.

Output

```
#include <iostream>
using namespace std;

int main () {
  int a,b;
  char x;
  cout <<"enter two numbers \n";
  cin>> a >> b;
  cout << "+ for adition \n ";
   cout << "- for subtraction \n";
    cout << "* for multiplication \n";
     cout << "/ for divition \n";
      cout << "enter your choice \n";
      cin >> x;
   switch(x) {
      case '+' :
         cout << a+b ;
         break;
      case '-' :
         cout << a-b;
         break;
      case '*' :
         cout << a*b;
         break;
```

```
enter two numbers
78 89
+ for adition
- for subtraction
* for multiplication
/ for divition
enter your choice
*
6942
```

```
      case '/' :
         cout << a/b;
         break;
      default :
      break;
   }

   return 0;

}
```

3.7 The Nested Switch Selection Statements

The syntax for a **switch** statement in C++ is as follows:

```
switch(selector)
{
   case label1: statement(s); break;
   case label2: statement(s); break;
   case label3: switch (selctore2);
                {
                 case label1: statement(s); break;
                 case label2: statement(s); break;
                   .
                   .
                }
   case label-n: statement(s); break;
   default: statement(s);break;
}
```

Example:

```
switch(ch1)
 {
   case 'A':
      cout << "This A is part of outer switch";
      switch(ch2)
 {
         case 'A':
            cout << "This A is part of inner switch";
```

```
            break;
         case 'B': // ...
      }
      break;
   case 'B': // ...

}
```

Example 3.10: Write a C++ program to implement the following:	**Output**
`#include <iostream>` `using namespace std;` `int main () {` `  int a = 100;` `  int b = 200;` `switch(a) {` `  case 100:` `  cout << "This is part of outer switch\n";` `switch(b) {` `  case 200:` ` cout << "This is part of inner switch" << endl;` `  }` `  }` ` cout << "Exact value of a is: " << a << endl;` ` cout << "Exact value of b is: " << b << endl;` `  return 0;` `}`	Exact value of a is : 100 Exact value of b is : 200

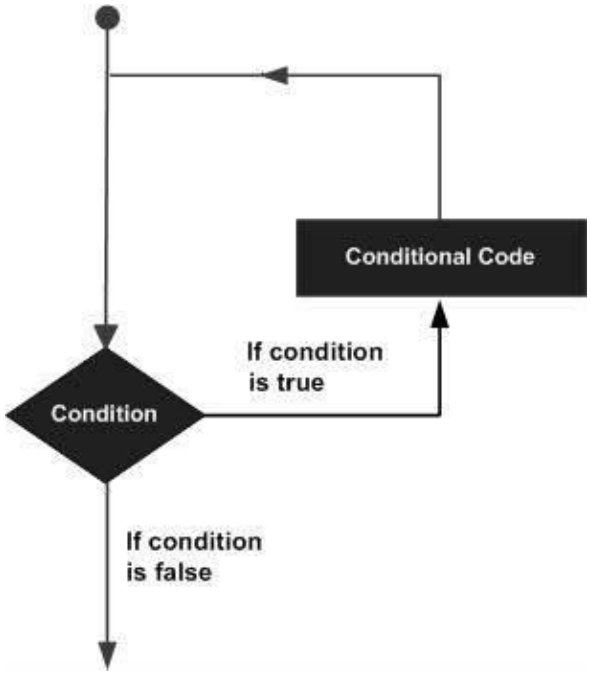
Chapter Four

Loop Statements

4.1 Selection Statement

C++ provides three iteration structure: ***While, do/while, and for.***

Loop statement allows us to execute a statement or group of statements multiple times and following is the general from of a loop statement in most of the programming languages:



A loop becomes ***infinite loop*** if a condition never becomes ***false***.

4.2 While Repetition Structure

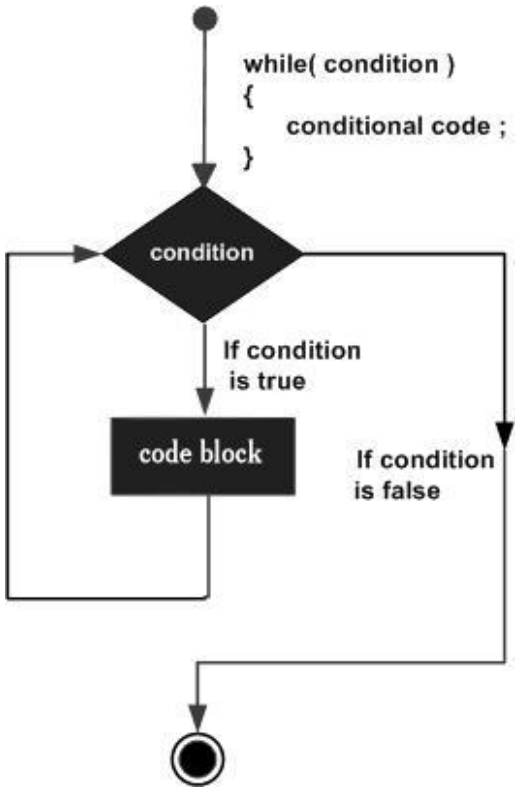
A **while** loop statement repeatedly executes a target statement as long as a given condition is ***true***. The syntax of a while loop in C++ is:

```
while(condition)
        statement1;
```

```
while (condition) {
        statement1;
        statement2;
        statement3;
            .
            .
        statement-n;
            }
```

Here, **statement(s)** may be a single statement or a block of statements. The **condition** may be any expression, and true is any non-zero value. The loop iterates while the condition is ***true***.

When the condition becomes ***false,*** program control passes to the line immediately following the loop.



Example 4.1: Write a C++ code to print the values between 1 and 5?	**Output**
`#include <iostream>` `using namespace std;` `int main ()` `{ int i=1;` `while (i<=5)` `Cout << i++;` `}`	1 2 3 4 5

Example 4.2: Write a C++ program to print the values between 0 and 9?	**Output**
`` #include <iostream> using namespace std; int main () { int i=0; while (i<10) { Cout << i; i++; } } ```	0123456789

Example 4.3: Write a C++ program to print the values between 0 and 9?	**Output**
`` #include <iostream> using namespace std; int main () { int i=0; while (i<10) { Cout << i; i+=2; } } ```	**H.W**

Example 4.4: Write a C++ program to find the summation of the following series: **Sum**= 1+3+5+7+…. +99 Note: find the summation of the odd numbers, between 0 and 100.	**Output**
`` #include <iostream> using namespace std; int main () { int count=1; int sum =0; while(count<=99) ```	sum is: 2500

```
    {
       sum = sum + count;
       count = count +2;
    }
    cout <<"sum is:" << sum;

}
```

Example 4.5: Write a C++ program to find the summation of the following series:

$$\sum_{i=1}^{n} i^2 = 1^2 + 2^2 + 3^2 + \cdots + n^2$$

Output

```
#include <iostream>
using namespace std;
int main ()
{
    int i=1, n, sum=0;
    cout <<"enter positive number:";
    cin >> n;
    while (i<=n)
    {
        sum+=i*i;
        i++;
    }
    cout << "Sum is:"<< sum<< endl;

}
```

```
enter positive number:5
Sum is:55
```

Example 4.6: Write a C++ program to inverse an integer number.

For Example: 123 ⟶ 321

Output

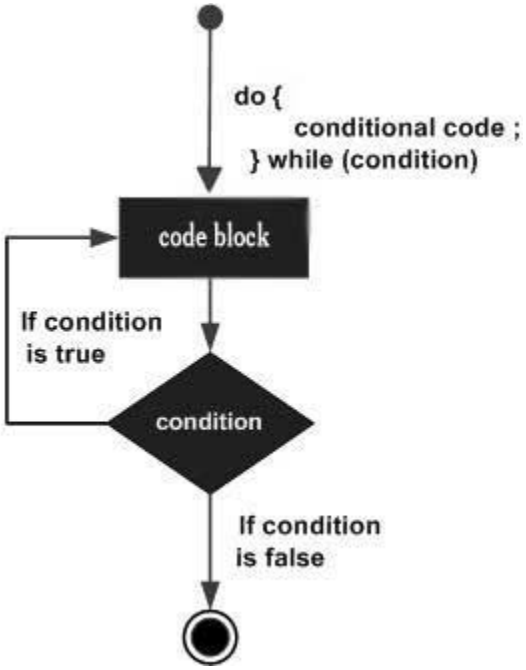
H.W

4.3 Do / While Repetition Structure

Unlike **while** loops, which test the loop condition at the top of the loop, the **do...while** loop checks its condition at the bottom of the loop. A **do...while** loop is similar to a while loop, except that a do...while loop is guaranteed to execute at least **one time**.

The syntax of a do...while loop in C++ is:

```
do {
   statement(s);
    }
while (condition);
```



Example 4.7: Write a C++ program to print the values between 0 and 9 using (do-while)?	**Output**
`` #include <iostream> using namespace std; int main () { int i=0; do { cout <<i; i++; } while (i<10); } ```	0 1 2 3 4 5 6 7 8 9

Example 4.8: Write a C++ program to print the values between 0 and 9 using (do-while)?	**Output**
`` #include <iostream> using namespace std; int main () { int i=0; do { cout <<i<<endl; i+=2; } while (i<10); } ```	0 2 4 6 8

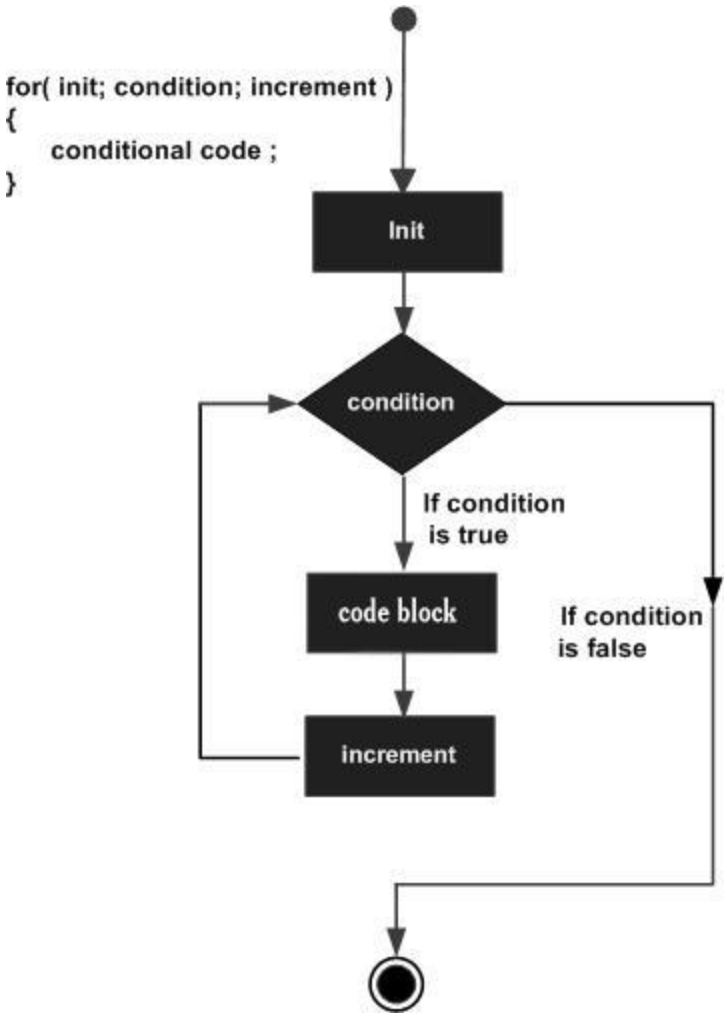
Example 4.9: Write a C++ program to find the factorial of n: **n!= n * n-1 * n-2 * n-3 * … * 2* 1**	**Output**
`` #include <iostream> using namespace std; int main() { int n, f=1; cout <<"enter positive number:"; cin >> n; do { f=f*n; n--; } while(n>1); cout << "factorial is:" <<f; } ```	enter positive number:4 factorial is:24

4.4 For Statement

A **for** loop is a repetition control structure that allows you to efficiently write a loop that needs to execute a specific number of times.

The syntax of for loop in C++ is:

```
for (initial; condition; increment)

 {
   statement(s);
}
```



Example 4.10: Write a C++ program to print the values between 0 and 9?	**Output**
`` #include <iostream> using namespace std; int main () { for(int i=0;i<10;i++) { cout << i; } } ```	0123456789

Example 4.11: Write a C++ program to print the values between 0 and 9?	**Output**
`` #include <iostream> using namespace std; int main () { for(int i=0;i<10;i+=2) { cout << i; } } ```	02468

Example 4.12: Write a C++ program to find the result of the following: $\sum_{i=1}^{20} a_i^2$	**Output**
`` #include <iostream> using namespace std; int main () { int sum=0; for(int i=1;i<=20;i++) sum=sum+(i*i); cout << "the sum is "<< sum; } ```	the sum is 2870

Example 4.13: Write a C++ program to print the following series: **1, 2, 4, 8, 16, 32, 64.**	**Output**
`` #include <iostream> using namespace std; int main () { int x; for (x=1; x<65; x*=2) cout << x<< " "; } ```	1 2 4 8 16 32 64

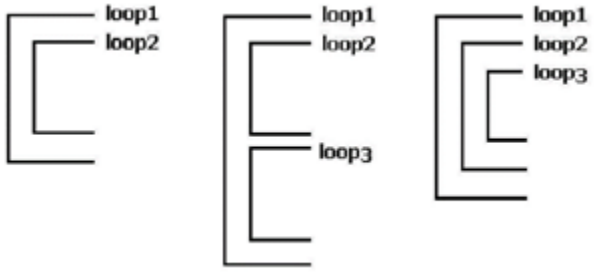
Example 4.14: Write a C++ program to print the following 1 10 2 9 3 8 4 7 5 6 6 5	**Output**
#include <iostream> ***using namespace std;*** ***int main ()*** ***{*** ***int x;*** ***for (x=1; x<7; ++x)*** ***cout << x<< " \t "<<11-x<< endl;*** ***}***	1 10 2 9 3 8 4 7 5 6 6 5

More about For Statement:

- We can use more than on control with for statement, as follow:
  for (int m=1, int n=8; m<n; m++, n--)
- We can create infinite loop, as follow:
  for (; ;)

4.5 Nested Loops

A loop can be nested inside of another loop.



The syntax for a **nested for loop** statement in C++ is as follows:

```
for (initial; condition; increment)
{
   for ( initial; condition; increment) {
      statement(s);
   }
   statement(s); // you can put more statements.
}
```

The syntax for a **nested while loop** statement in C++ is as follows:

```
while(condition)
{
   while(condition) {
      statement(s);
   }
   statement(s); // you can put more statements.
}
```

The syntax for a **nested do...while loop** statement in C++ is as follows:

```
do
{
   statement(s);
   do
   {
      statement(s);
   }
     while( condition );

}
 while( condition );
```

Example 4.15: Write a C++ program to evaluate the following series:

$$\sum_{i=1}^{5}\sum_{j=1}^{10} i + 2j$$

Output

```
#include <iostream>
using namespace std;
int main ()

{
    int i, j, sum=0;
    for (i=1; i<=5; i++)
    for (j=1; j<=10; j++)
    sum = sum + (i+2 *j);
    cout << "sum is:" << sum;

}
```

sum is:700

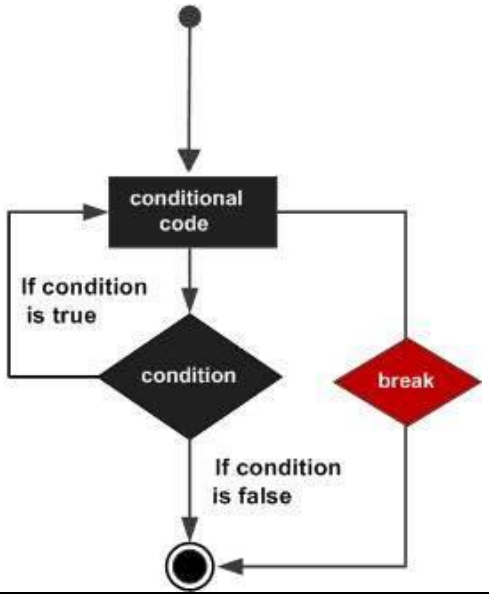
Example 4.16: what is the output for the following program:

Output

```
#include <iostream>
using namespace std;
int main ()
{
    int i, j, k;
    for (i=1; i<=2; i++)
    {
        for (j=1; j<=3; j++)
        {
            for (k=1; k<=4; k++)
            cout << "+";
            cout << "\n";
        }
     }
}
```

4.6 Break and Continue Control Statement

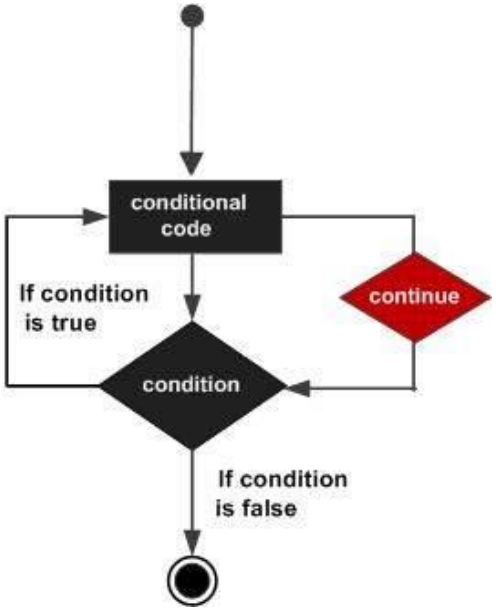
4.6.1 Break Control Statement



Example 4.17: write C++ program to print the values between 1 and 100?	**Output**
`` #include <iostream> using namespace std; int main () { int i; for (i=1; i<=100; i++) { cout << i; if (i==10) break; } } ```	1 2 3 4 5 6 7 8 9 10
Example 4.18: The following program uses a nested for loop to print the values between 10 and 20 and stop when arrive 15.	**Output**
`` #include <iostream> using namespace std; int main () { int a = 10; do { cout << "value of a: " << a << endl; ```	value of a: 10 value of a: 11 value of a: 12 value of a: 13 value of a: 14 value of a: 15

`a = a + 1;` `if( a > 15)` `{` `break;` `}` `}` `while( a < 20 );` `return 0;` `}`	

4.6.2 Continue Control Statement



Example 4.19: The following program uses a nested for loop to print the values between 10 and 20 and avoid the value 15?	**Output**
`#include <iostream>` `using namespace std;` `int main ()` `{` `int a = 10;` `do {` `if( a == 15)` `{` `a = a + 1;`	value of a: 10 value of a: 11 value of a: 12 value of a: 13 value of a: 14 value of a: 16 value of a: 17 value of a: 18 value of a: 19

```
          continue;
         }
       cout << "value of a: " << a << endl;
       a = a + 1;
       }
        while( a < 20 );

   return 0;
}
```

Example 4.20: The following program uses a nested for loop to find the prime numbers from 2 to 100.

Output

H.W

```
#include <iostream>
using namespace std;
int main ()
{
   int i, j;

   for(i = 2; i<100; i++)
    {
      for(j = 2; j <= (i/j); j++)
         if(!(i%j)) break;
         if(j > (i/j))
         cout << i << " is prime\n";
    }

   return 0;
}
```

Questions

Q1// Write C++ program to read three marks of student and compute average, if the average greater than or equal 90 then print the “excellent average” in the middle of second line and otherwise print "average value" in the end of third line?

Q2// Write C++ program to read 50 numbers then find the sum of all positive integer number which are divisible by 3 using (do-while) statement?

Q3// Write C++ program to find the summation of the following series:
sum = 1,1,2,3,5,8,13,21

Q4// Write C++ program to read float number and Rounding the float number to the nearest integer?

Q5// Write C++ program to reads a character and print if it is digit (0..9),capital letter (A,B, … ,Z), small letter (a, b, … ,z), special character (+, !,@, #, , {, >, …).

Q6// Write C++ program to find the summation of the following series:

sum = 1,3,9,27,81

Q7// Write C++ program to Display multiplication table up to a given range?

Q8// Write C++ program to print the summation of the following series:

$$S= x^2 + x^4 + x^8 + \ldots . x^n$$

Q9// Write C++ program to show the numbers in the figure below:

1

333

55555

333

1

Q10// Write C++ program to find the summation of the following series:

$$1 + (1/2)^2 + (1/3)^3 + (1/4)^4 \ldots.+ (1/n)^n$$

Q11// Write C++ program to find the perfect number of a positive number? For example: 6 is Perfect Number since divisor of 6 are 1, 2 and 3. Sum of its divisor is 1+2+3 = 6 But 15 is not a Perfect Number since divisor of 15 are 1, 3 and 5. Sum of its divisor is $1 + 3+ 5 \neq 15$.

Q12// What the Output of Following

```
#include <iostream>
using namespace std;
int main()
{
int n=8;
while ( n != 0 )
{
cout << "n= " << n <<
endl ;
n--;
}
cout << "n= "<<"zero" ;
   return 0;
}
```

```
#include <iostream>
#include <math.h>
using namespace std;
int main()
{ int s=0;
int x=2;
for(int i=2;i<=8;i*=2)
{
   s+=pow(x,i);
}
cout<<s;
return 0;
}
```

```
#include <iostream>
using namespace
std;
int main()
{ int i,j,n=6;
for(i=1;i<=n;i++)
{
for(j=i;j>=1;j--)
cout<<j<<" ";
cout<<endl;
}
return 0;
}
```

```
#include <iostream>
using namespace std;
int main()
{
for (int I = 0; I < 8; I ++ )
{
if ( I % 2 == 0 )  cout <<
I + 1 << endl;
else  if  ( I % 3 == 0 )
cout << I + 2 << endl;
 else  if  ( I % 5 == 0 )
cout << I + 3 << endl;  }
cout << "end ";
 return 0;  }
```

References

1- Paul J. Deitel and Harvey Deitel. 2016. C# 6 for Programmers (6th Edition) (6th. ed.). Prentice Hall Press, USA.

2- Perkins, Benjamin, Jacob Vibe Hammer, and Jon D. Reid. beginning C# 6 programming with visual studio 2015. John Wiley & Sons, 2015.

3- Albahari, Joseph. C# 10 in a Nutshell. " O'Reilly Media, Inc.", 2022.

4- Rob Miles,# Programming Yellow Book , “Cheese” Edition 8.1 December 2019.

5- Almeida, Fernando. "Visual C# .NET: Console Applications and Windows Forms." no. July (2018): 63.

6- Perkins, Benjamin, Jacob Vibe Hammer, and Jon D. Reid. beginning C# 6 programming with visual studio 2015. John Wiley & Sons, 2015.