

University of Technology  
الجامعة التكنولوجية



Computer Science Department  
قسم علوم الحاسوب

أساسيات البرمجة

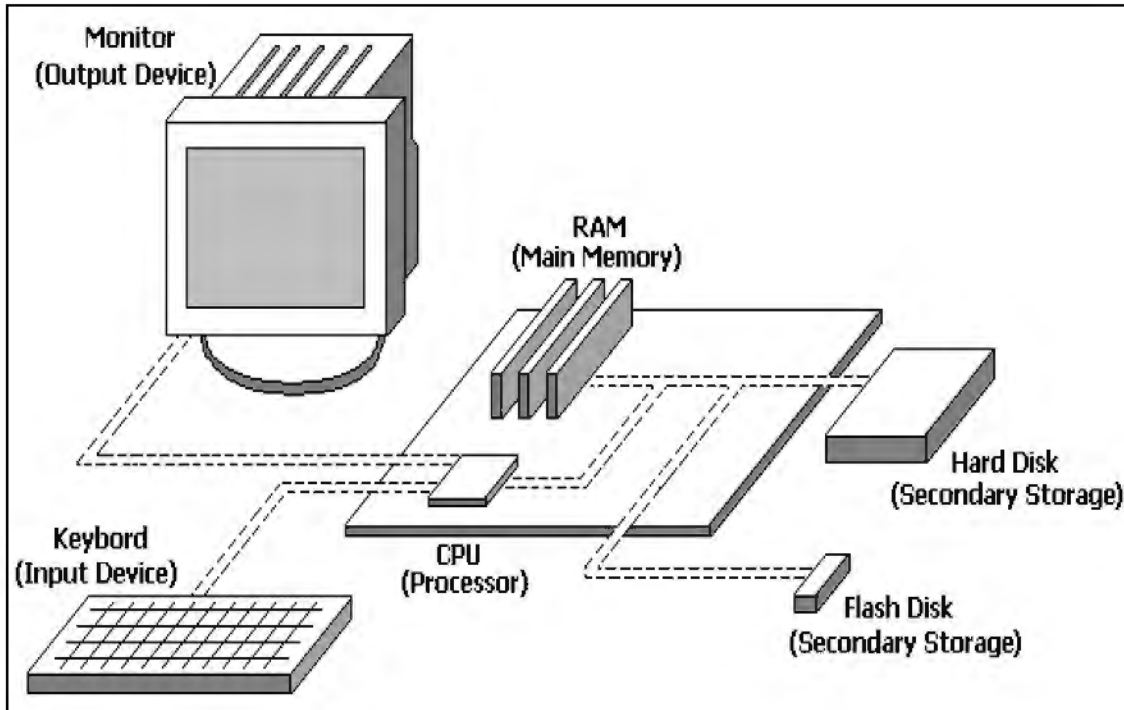
أ.م.د. بشار سعدون ، م. ياسر منذر  
م.د. انمار علي ، م. رشا اسماعيل



[cs.uotechnology.edu.iq](http://cs.uotechnology.edu.iq)

# LECTURE 1

## 1. Introduction:



*hardware components*

**Computer** is a device capable of performing computations and making logical decisions at speeds millions and even billions of times faster than human beings.

Computers process data under the control of sets of instructions called **computer programs**.

**Programming** is the process of writing instructions for a computer in a certain order to solve a problem.

The computer programs that run on a computer are referred to as **software (SW)**. While the hard component of it is called **hardware (HW)**.

Developing new software requires written lists of instructions for a computer to execute. Programmers rarely write in the **language** directly understood by a computer.

## 2. Short History:

The following is a short history, just for given a general view of how languages are arrived:

- 1954: Fortran.
- 1957: Cobol.
- 1958: Algol (*Base for Simula*).
- 1958: Lisp.
- 1961: B1000.
- 1962: Sketchpad.
- 1964: Basic.
- 1967: Simula67.
- 1968: FLEX.
- 1970: Pascal (*From Algol*).
- **1971: C** (*From a language called B*).
- 1972: Smalltalk72 (*Based on Simula67 and Lisp*).
- 1976: Smalltalk76.
- 1979: ADA (*From Pascal*).
- **1980: C with classes** (*experimental version*).
- **1983: C++ (by Bjarne Stroustrup)**.
- 1986: Objective-C (*from C and Smalltalk*).
- 1986: Eiffel (*from Simula*).
- 1991: Sather (*From Eiffel*).
- 1991: Java.
- 2000: C#.



**Bjarne Stroustrup**  
at: AT&TLabs

### 3. C++ Programming Language:

For the last couple of decades, the C programming language has been widely accepted for all applications, and is perhaps the most powerful of structured programming languages. Now, C++ has the status of a structured programming language with object oriented programming (OOP).

C++ has become quite popular due to the following reasons:

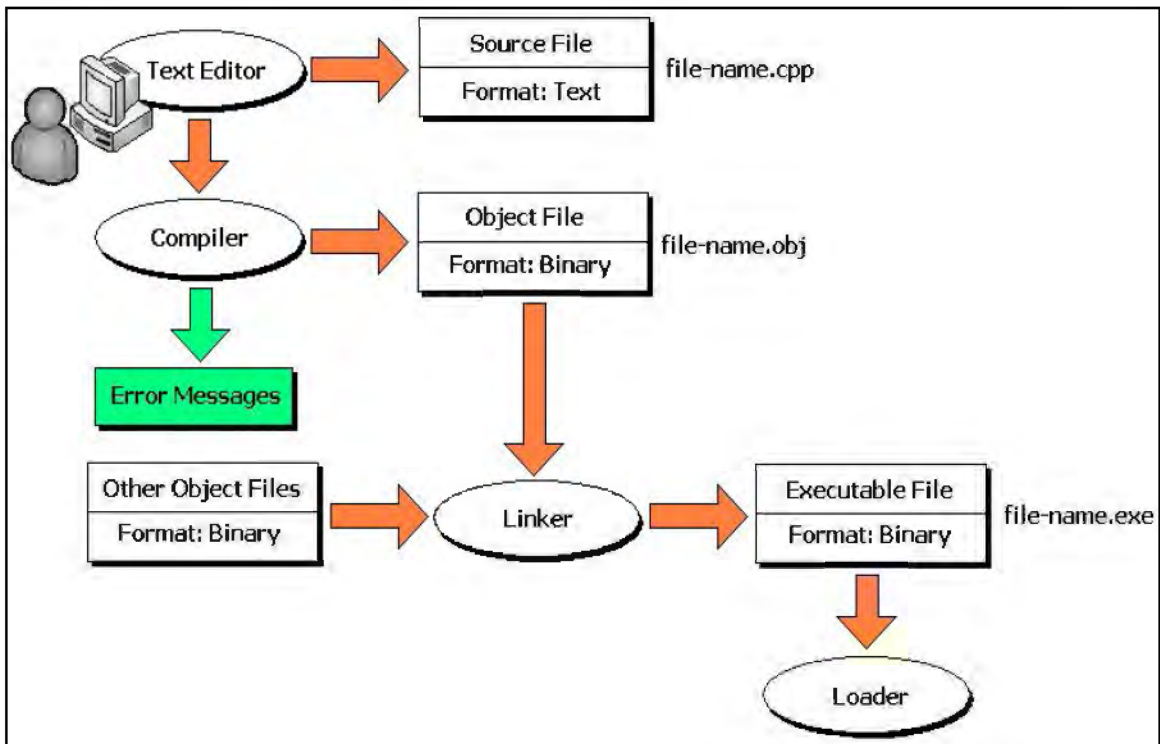
1. It supports all features of both structured programming and OOP.
2. C++ focuses on function and class templates for handling data types.

### 4. C++ Program Development Process (PDP):

C++ programs typically go through six phases before they can be executed. These phases are:

1. **Edit:** The programmer types a C++ source program, and makes correction, if necessary. Then file is stored in disk with extension (.cpp).
2. **Pre-Processor:** Pre-processing is accomplished by the pre-processor before compilation, which includes some substitution of files and other directories to be include with the source file.
3. **Compilation:** Converting the source program into object-code.
4. **Linking:** A linker combines the original code with library functions to produce an executable code.
5. **Loading:** The loader loads the program from the disk into memory.
6. **CPU:** Executes the program, residing in memory.

These steps are introduced in the figure below:



# LECTURE 2

## 1. Algorithm:

As stated earlier an algorithm can be defined as a finite sequence of effect statements to solve a problem. An effective statement is a clear, unambiguous instruction that can be carried out .Thus an algorithm should special the action to be executed and the order in which these actions are to be executed.

## Algorithm properties:

- **Finiteness**: the algorithm must terminate a finite number of steps.
- **Non-ambiguity**: each step must be precisely defined. At the completion of each step, the next step should be uniquely determined.
- **Effectiveness**: the algorithm should solve the problem in a reasonable amount of time.

**Example 1:** Develop an algorithm that inputs a series of number and output their average .

A computer algorithm can only carry out simple instruction like:

- "Read a number".
- "Add a number to anther number".
- "Output a number".

Thus an algorithm is:

1. Carry out initialization required.
2. Read first number.
3. While the number of numbers is not complete do
4. begin
5. Add the number to the accumulated sum.
6. increment the count of numbers entered.

7. Read next number.
8. End
9. Evaluate the average.

**Example 2: Devolve an algorithm that allows the user to enter the count of numbers in a list followed by these numbers. The algorithm should find and output the minimum and the maximum numbers in the list.**

An algorithm for this might be:

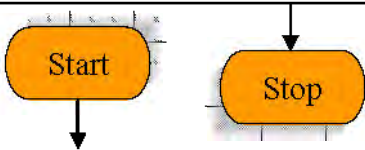

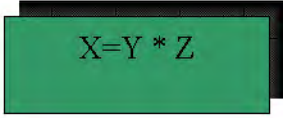
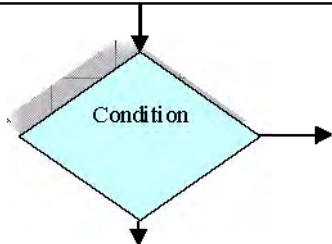
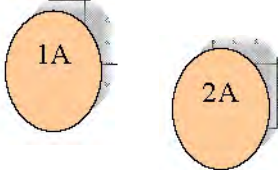
- Initialize.
- Get count of numbers.
- Enter numbers and find maximum and minimum .
- Output result.

The user might enter zero for the count. To deal with this case the above general case can be extended as follows to be an algorithm:

1. Initialize the require variables.
2. Get count of numbers.
3. If count is zero then exit.
4. Otherwise begin.
5. Enter numbers.
6. Find max and min.
7. Output result.
8. End.

## 2. Flowcharts

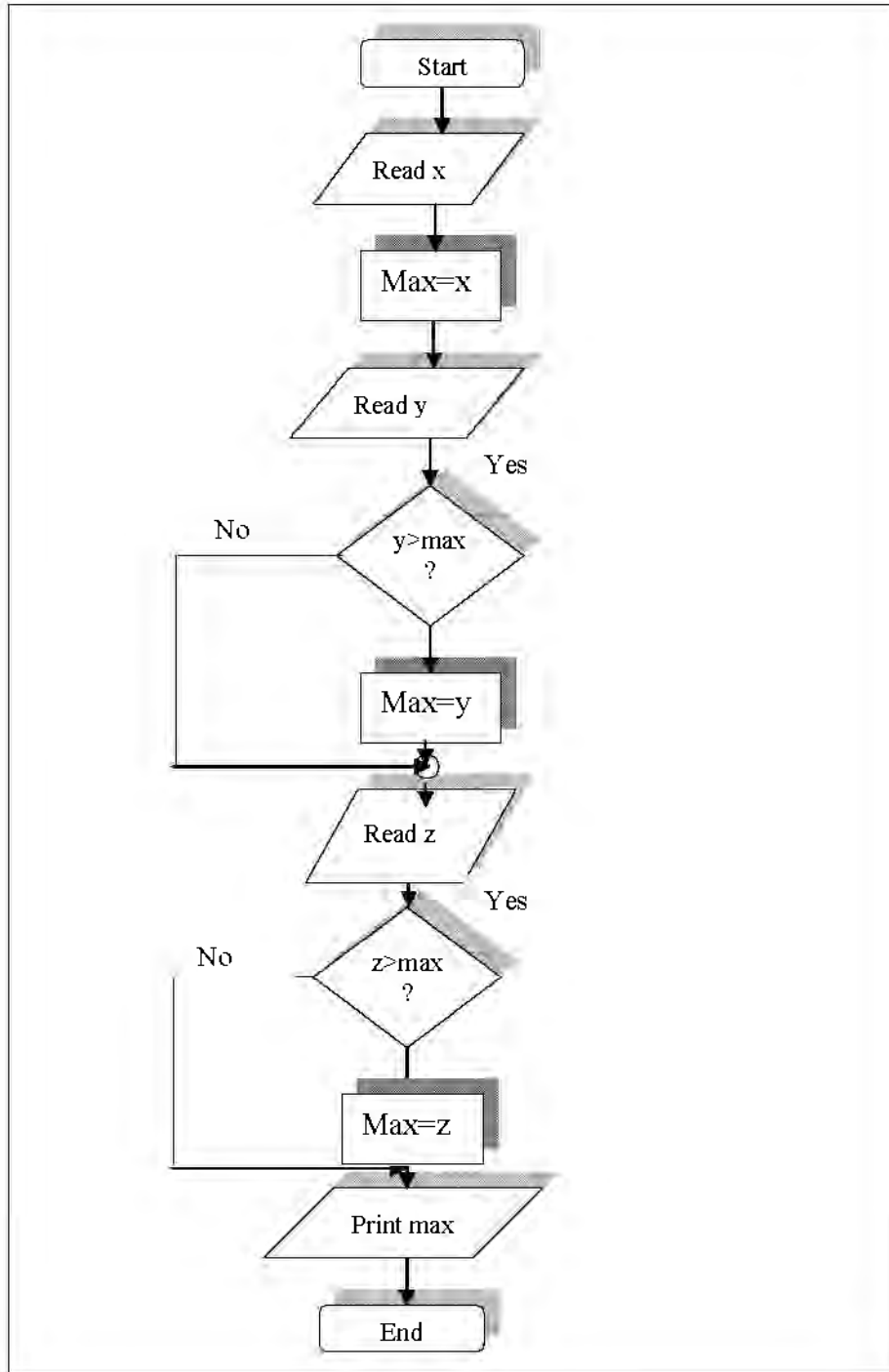
A flowchart is a graphical representation of an algorithm or of a portion of an algorithm .Flowcharts are drawn using symbols. The main symbols used to draw a flowchart are shown in following figure.

	Start and Stop Symbols
	Input and Output Symbols
	Mathematical and logical processing symbol
	Decision making symbol
	Connector symbols



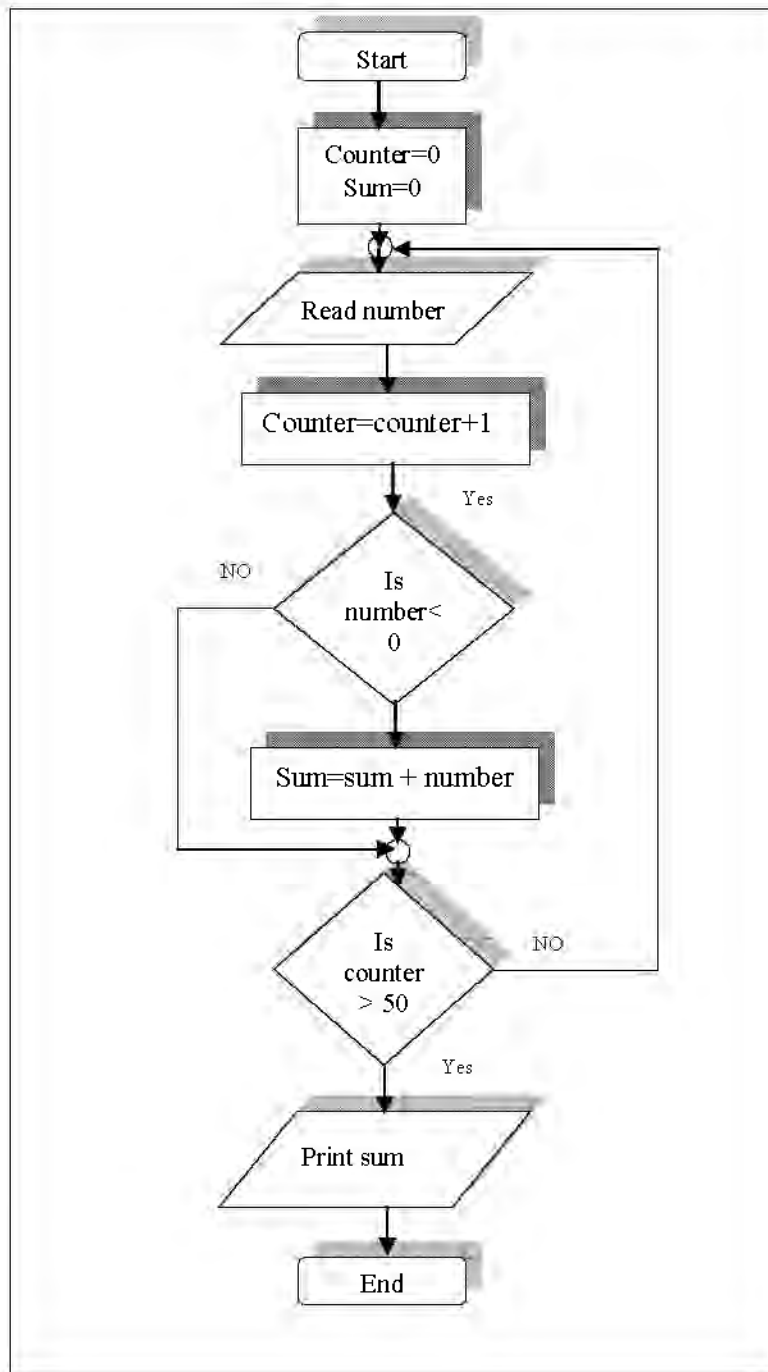
**Example 1:**

Draw a flowchart to read 3 numbers: x , y and z and print the largest number of them.



**Example 2:**

Draw the flowchart required to find the sum of negative numbers among 50 numbers entered by the user.



# WORK SHEET (1)

## AN INTRODUCTION

Q1: What do you mean by program?

Q2: Why C++ language becomes quite popular?

Q3: Talk briefly about C++ program development process?

Q4: Write an algorithm and flowcharts for the following:

- a. Sum the even numbers for n numbers.
- b. Display numbers from 0 to 10.
- c. The multiplication of 10 numbers.

# Lecture 3

## 1 Character set:

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

Uppercase: A, B, C, . . . , Z

Lowercase: a, b, c, . . . , z

Digits: 0, 1, 2, . . . ,9

**Special Characters:** All characters other than listed treated as special characters for example:

+	-	*	/	^
(	[	{	}	]
)	<	=	>	, (Comma)
" (Double Conations)	. (Dot)	: (Colon)	; (Semicolon)	␣ (Blank Space)

In C++ language, upper case and lower case letters are distinct and hence there are 52 letters in all. For example **bag** is different from **Bag** which is different from **BAG**.

## 2 Identifiers:

An **identifier** is a name given to some program entity, such as variable, constant, array, function, structure, or class. An identifier is a sequence of alphanumeric (alphabetic and numeric) characters, the first of which must be a letter, and can't contain spaces. The length of an identifier is machine dependent. C++ allows identifiers of up to **127 characters**.

A **variable** should not begin with a digit. C++ does not set a maximum length for an identifier. Some examples of valid identifiers are as follows:

My_name	(7 char.)
i	(1 char.)
B	(1 char.)

Examples of invalid identifiers are:

3ab a()test ros sal

### 3 Keywords:

The keywords are also identifiers but cannot be user defined, since they are reserved words. All the keywords should be in lower case letters. Reserved words cannot be used as variable names or constant. The following words are reserved for use as keywords:

Some of C++ Language Reserved Words:				
break	case	char	cin	cout
delete	double	else	enum	false
float	for	goto	if	int
long	main	private	public	short
sizeof	switch	true	union	void

### 4 Constants:

There are three types of constants: **string constants**, **numeric constants**, and **character constants**.

**1. String Constants:** A string constants are a sequence of alphanumeric characters enclosed in double quotation marks whose maximum length is 255 characters. In the following are examples of valid string constants: ("The result=", "RS 2000.00", "This is test program"). The invalid string constants are like: (Race, "My name, 'this').

**2. Numeric Constants:** Numeric constants are positive or negative numbers. There are four types of numeric constants: integer, floating point, hexadecimal, and octal.

Integer	Integer Short integer (short) Long integer (long)
Float	Single precision (float) Double precision (double) Long double
Hexa	Short hexadecimal Long hexadecimal
Unsigned	Unsigned char Unsigned integer Unsigned short integer Unsigned long integer
Octal	Short octal Long octal

**(a) Integer constants:** Do not contain decimal points: int x,y; shortint x,y; longint x,y;

- Integer data: size (16 or 32) fill in  $-2^{15}$  to  $2^{15}-1$  for 16 bit and  $-2^{31}$  to  $2^{31}-1$  for 32 bit.
- Short integer: fill in  $-2^{15}$  to  $2^{15}-1$ .
- Long integer: fill in  $-2^{31}$  to  $2^{31}-1$ .
- Unsigned: fill in (0 to 65635) for 16 bit and (0 to 4,294, 967, 295) for 32 bit.

**(b) Floating point constants:** Positive or negative numbers are represented in exponential form. The floating point constant consists of an optionally (signed) integer or fixed point number (the mantissa) followed by the letter E and e and an optionally signed integer (the exponent). Ex. (9010e10, 77.11E-11).

- Float 4 bytes.
- Double 8 bytes.
- Long double 12 or 16.

**(c) Hexadecimal constants:** Hexadecimal numbers are integer numbers of base 16 and their digits are 0 to 9 and A to F.

**(d) Octal constants:** Octal numbers are numbers of base 8 and their digits are 0 to 7.

**3. Character Constants:** A character represented within single quotes denotes a character constant, for example 'A', 'a', ':', '?', etc...

Its maximum size is 8 bit long, signed, and unsigned char are three distinct types.

Char x;                    char x,y,z;

The backslash (\) is used to denote non graphic characters and other special characters for a specific operations such as:

Special Escape Code:	
Escape Code	Description
<code>\n</code>	New line. Position the screen cursor to the beginning of the next line.
<code>\t</code>	Horizontal TAB (six spaces). Move the screen cursor to the next tab stop.
<code>\r</code>	Carriage return. Position the cursor to the beginning of the current line, do not advance to the next line.
<code>\a</code>	Alert. Produces the sound of the system bell.
<code>\b</code>	Back space
<code>\\</code>	Backslash. Prints a backslash character.
<code>\f</code>	Form feed
<code>\v</code>	Vertical tab
<code>\"</code>	Double quote. Prints a (") character.
<code>\o</code>	Null character
<code>\?</code>	question mark
<code>\ooo</code>	Octal value
<code>\xhhh</code>	Hexadecimal value

## 5. C++ operators:

<b>C++ operators</b>	Arithmetic operators	
	Assignment operators	
	Comparison and logical operators	Relational, equality, logical
	Bit wise logical operators	
	Special operators	Unary, ternary, comma Scope, new&delete, other

**1. Arithmetic operators:** These operators require two variables to be evaluated:

+ addition                      - subtraction                      \* multiplication  
/ division                      % modula (remainder of an integer division)

The division result are:

Integer / integer = integer                      ►  $39/7=5$   
Integer / float = float                      ►  $39/7.0 =5.57$   
float / integer = float                      ►  $39.0/7 =5.57$   
float / float = float                      ►  $39.0/7.0=5.57$

while  $39\%5=7$ , since  $39=7*5+4$

Arithmetic operators as per precedence:

( ) for grouping the variables.

- Unary for negative number.

\* / multiplication & division.

+ - addition and subtraction.



**Example:**  $X+y*X-Z$ , where  $X=5$ ,  $Y=6$ , and  $Z=8$ .

$$5 + (6*5)-8 \rightarrow (5+30)-8 \rightarrow 35-8 \rightarrow 27$$

**2. Assignment Operators:** The operational assignment operator has the form:

**Variable = variable operator expression;**

**Ex:**  $x=x+5;$        $y=y*10;$

The operational assignment operator can be written in the following form:

**Variable operator = expression**

**Ex:**  $x+=5;$        $y*=10;$

It is used to assign back to a variable, a modified value of the present holding:

<b>=</b>	Assign right hand side (RHS) value to the left hand side (LHS).
<b>+=</b>	Value of LHS var. will be added to the value of RHS and assign it back to the var. in LHS.
<b>-=</b>	Value of RHS var. will be subtracted to the value of LHS and assign it back to the var. in LHS.
<b>*=</b>	Value of LHS var. will be multiplied to the value of RHS and assign it back to the var. in LHS.
<b>/=</b>	Value of LHS var. will be divided to the value of RHS and assign it back to the var. in LHS.
<b>%=</b>	The remainder will be stored back to the LHS after integer division is carried out between the LHS var. and the RHS var.
<b>&gt;&gt;=</b>	Right shift and assign to the LHS.
<b>&lt;&lt;=</b>	Left shift and assign to the LHS.
<b>&amp;=</b>	Bitwise AND operation and assign to LHS
<b> =</b>	Bitwise OR operation and assign to LHS
<b>~=</b>	Bitwise complement operation and assign to LHS

This is a valid statements:

A=b=c+4;

C=3\*(d=12.0/x);

Exercise:

Rewrite the equivalent statements for the following examples, and find it results. Assume: X=2 , Y=3 , Z=4 , V=12 , C=8.

Example	Equivalent Statement	Result
X += 5	X = X + 5	X ← 7
Y -= 8	Y = Y - 8	Y ← -5
Z *= 5	Z = Z * 5	Z ←
V /= 4		V ←
C %= 3		C ←

**3. Comparison and logical operators:** It has three types relational operators, equality operators, and logical operators.

**(a) Relational operators:** < less than, > greater than, <= less than or equal, >= greater than or equal, an expression that use relational operators return the value of one if the relational is TRUE ZERO otherwise.

Ex: 3 > 4 → false, 6 <=2 → false, 10 > -32 → true, (23\*7) >= (-67+89) → true

**(b) Equality operators:** == equal to , != not equal to

Ex: a=4, b=6, c=8. A==b → false, (a\*b) != c → true, 's' == 'y' → false.

**(c) Logical operators:** The logical expression is constructed from relational expressions by the use of the logical operators not(!), and(&&), or(| |).

**AND (&&) Table:**

A	B	A && B
T	T	T
T	F	F
F	T	F
F	F	F

**AND (&&) Table:**

A	B	A && B
1	1	1
1	0	0
0	1	0
0	0	0

OR (  ) Table:		
A	B	A    B
T	T	T
T	F	T
F	T	T
F	F	F

OR (  ) Table:		
A	B	A    B
1	1	1
1	0	1
0	1	1
0	0	0

NOT (!) Table:	
A	!A
T	F
F	T

NOT (!) Table:	
A	!A
1	0
0	1

## Examples:

### Example 1:

a=4, b=5, c=6

(a<b)&&(b<c)	(a<b)    (b>c)	!(a<b)    (c>b)	(a<b)    (b>c) && (a>b)    (a>c)
T && T	T    T	!(T)    T	T    F && F    F
T	T	F    T	T    F    F
		T	T    F
			T

### Example 2:

Assume: X=0, Y=1, Z=1. Find the following expression:

M = ++X || ++Y && ++Z

M = ++X || ++Y && ++Z

= 1 || (2 && 2)

= T || (T && T)

= T || T

= T

= 1

### (d) Bitwise logical operator:

& bitwise AND, ^ bitwise exclusive OR(XOR), | bitwise inclusive OR,

>> bitwise left shift, << bitwise right shift, ~ bitwise complement.

Ex: x=23 (0001 0111)      ~x=132 (1110 1000)

X=33 (0010 0001)

$X \ll 3$

0 01000010

0 10000100

1 00001000 the resultant bit pattern will be (0000 1000)

$X=5, y=2 \rightarrow x \& y$  (0000) ,  $x | y$  (0111) ,  $x \wedge y$  (0111)

**(e) special operators:**

**1. Unary operator:**

*	Contents of the storage field to which a pointer is pointing.
&	Address of a variable.
-	Negative value (minus sign).
!	Negative (0, if value $\neq$ 0, 1 if value =0).
~	Bitwise complement.
++	Increment.
--	Decrement.
Type	Forced type of conversion
Size of	Size of the subsequent data type or type in byte.

**2. Ternary operator:** It is called conditional operator, it is like if else construction:

Expression 1 ? expression 2 : expression 3

If (v%2 == 0)  
e = true  
Else  
e=false

}  $E = (v\%2 == 0) ? \text{True} : \text{false}$

**3. Comma operator: (,)**

Int a,b,c; or it is used in control statements

**4. Scope operator: (::)** It is used in a class member function definition.

**5. New and delete operators:** it is a method for carrying out memory allocations and deallocations.

6. **Other operators:** parentheses for grouping expressions, membership operators.

### 6. Type Conversion:

Some variables are declared as integers but sometimes it may be required to get the result as floating point numbers. It is carried out in two ways:

(A) Converting by assignment: int x; float y; x=y;	(B) Cast operator: Result =(int) (19.2/4); or Result = int (19.2/4);
---	--

# Lecture 4

## 1 Statements:

A statement in a computer carries out some action. There are three types of statements used in C++; they are expression statement, compound statement and control statement.

Expression statement	Compound statement	Control statement
<code>x=y; sum=x+y;</code>	<code>{ a=b+c; x=x*x; y=a+x; }</code>	<code>If (a&gt;b) { a=l; k=a+1; }</code>

## 2 Getting Started with C++:

The skeleton of a typical C++ program structure is given below:

*Program heading*  
*Begin*  
*Type or variable declaration*  
*Statements of operation*  
*Results*  
*end*

The keyboard and screen I/O instructions in C++ are:

(a): **COU**T/ display an object onto the video screen:

`Cout<<var.1<<var2<<...<<var.n;`

(b): **Cin**/ It is used to read an object from a standard input device (keyboard):

`Cin>>var.1>>var.2>>...>>var.n;`

To begin learning C++ lets examine our first C++ Program:

## Example 1

```
#include<iostream.h>
void main( )
{
    // A program to print welcome
    cout << "Welcome";
}
```

Output:

Welcome

- ✓ **#include<iostream.h>** this line is for pre-processor directive. Any begins with # is processed before the program is compiled. C++ programs must be start with #include.

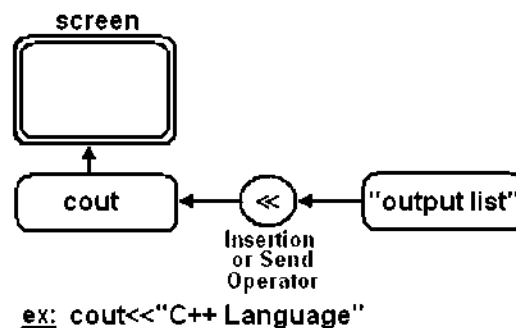
Every group of related functions is stored in a separate library called (header file).To use the *cin* and *cout*, must include the header file *iostream*.

- ✓ **main( )**, is the name of C++ function. Every C++ program must have a function called main.

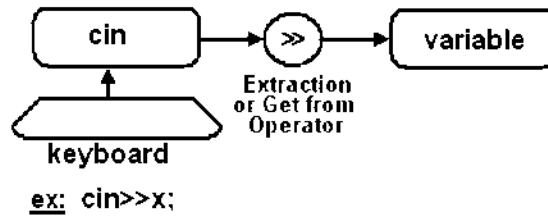
- ✓ **void**, is the return type of the main function. When the return type of a function is void, this function will not passes back any value to the calling function.

Some programmers use *int* as a return type for the main function, in this case a **return(0)** statement must be written as a last statement of the main function-body.

- ✓ **{**, introducing the statements that define the function.
- ✓ **}**, indicates the end of the statements in the function.
- ✓ **//**, text after these symbols is a comment. It does not affect the program code, and compilers normally ignore it.
- ✓ **cout**, the input stream object. It passes the characters quotes (") to the terminal screen.



- ✓ **cin**, the input stream object. It reads the input values from the keyboard.



- ✓ <<, the stream insertion operator (or send operator).
- ✓ >>, the stream extraction operator (or get from operator).
- ✓ ;, semicolon, the terminator of every C++ statement.

The **endl** is used in c++ to represent a new line, as shown in the following example:

<b>Example 2</b>	
<pre> #include&lt;iostream.h&gt; void main( ) {     cout &lt;&lt; "hallow" &lt;&lt; endl;     cout &lt;&lt; "students"; }           </pre>	<p><b>Output:</b></p> <div style="border: 1px solid black; padding: 10px; min-height: 80px;">           hallow students         </div>

The **\n** is a special escape code, also used in C++ to represent a new line, as shown in the following example:

<b>Example 3</b>	
<pre> #include&lt;iostream.h&gt; void main( ) {     cout &lt;&lt; "hallow \n";     cout &lt;&lt; "students"; }           </pre>	<p><b>Output:</b></p> <div style="border: 1px solid black; padding: 10px; min-height: 80px;">           hallow students         </div>





#### Example 4



The following program reads three different inputs and outputs it.

```
#include<iostream.h>
void main( )
{
    int num=3;
    cout << "number="<<num<<"\n";
    char ch='a';
    cout << "character="<<ch<<"\n";
    float fa=-34.45;
    cout<<"real number="<<fa<<"\n";
}
```

#### Output:

Number=3

Character=a

Real number=34.45

#### Example 5



The following program reads three different inputs and outputs it.

```
#include<iostream.h>
void main( )
{
    int n; float f; char c;
    cout << "input integer number:";
    cin>>n;
    cout<<endl;
    cout << "input decimal number:";
    cin>>f;
    cout<<endl;
    cout << "input character:";
    cin>>c;
}
```

#### Output:

input integer number: 5

input decimal number: 4.2

input character: A

### 4 Constants:

Like variables, constants are data storage locations. Unlike variables, and as the name implies, constants don't change.

```
const int myage=23;
const double pi=3.14;
const float salary=20.5;
```

### Example 6



Write a program that reads the radius of a circle, then computes and outputs its area.

```
#include<iostream.h>
void main( )
{
    const float pi = 3.14;
    int r; float c;
    cout << "enter the radius of circle:";
    cin>>r;
    cout<<endl;
    c = r * r * pi;
    cout << "the area of circle:" << c;
}
```

#### Output:

```
enter the radius of circle: 5
the area of circle: 78.5
```

### Example 7



The following program computes the arithmetic operators.

```
#include<iostream.h>
void main( )
{
    int a,b,sum,sub,mul,div;
    cout << "enter any two numbers<<endl;
    cin>> a>>b;
    sum=a+b;
    sub=a-b;
    mul=a*b;
    div=a/b;
    cout<<"a="<<a<<"b="<<b<<"sum="<<sum<<endl;
    cout<<"sub="<<sub<<endl;
    cout<<"mul="<<mul<<endl;
    cout<<"div="<<div<<endl;
}
```

#### Output:

```
Enter any two numbers
10 20
A=10 b=20 sum=30
Sub=-10
Mul=200
Div=0
```

### Example 8



The following program computes different division operators.

```
#include<iostream.h>
void main( )
{
    int x, y, z, r;
    x= 7 / 2;
    cout << "x=" << x <<endl;
    y=17/(-3);
    cout << "y="<< y <<endl;
    z=-17/3;
    cout << "z="<< z <<endl;
    r=-17/(-3);
    cout << "r="<< r <<endl;
}
```

Output:

```
x= 3
y= -5
z= -5
r= 5
```

The modulus operator "%" is used with integer operands (int, short, long, unsigned). It can't be used with float or double operands.

### Example 9

```
#include<iostream.h>
void main( )
{
    int y1, y2;
    y1 = 8 % 3;
    y2 = -17 % 3;
    cout << "y1="<< y1 <<endl;
    cout << "y2="<< y2 <<endl;
}
```

Output:

```
y1=2
y2=-2
```

# Lecture 5

## 1 Examples of order evaluation:

### Example 1:

Write the following equation as a C++ expression:

$$f = \frac{a + b + c + d + e}{10}$$

### Solution:

`f = (a + b + c + d + e) / 10;`

Note: the parentheses here are required because division has higher precedence than addition.

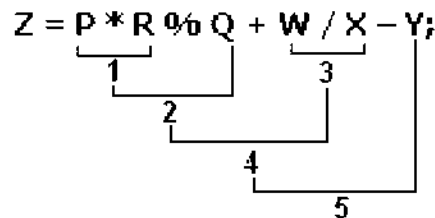
### Example 2:

State the order of evaluation for the following expression:

`Z = P * R % Q + W / X - Y;`

### Solution:

1. \*
2. %
3. /
4. +
5. -



### **Example 1**



Write C++ program to perform the above equation:

```
#include<iostream.h>
void main( )
{
    int Z, P, R, Q, W, X, Y;
    cout << "enter P: "; cin >> P;
    cout << "enter R: "; cin >> R;
    cout << "enter Q: "; cin >> Q;
    cout << "enter W: "; cin >> W;
    cout << "enter X: "; cin >> X;
    cout << "enter Y: "; cin >> Y;
    Z= P * R % Q + W / X - Y;
    cout << "the result=" << Z;
```

## 2 The "math.h" Library:

The "math.h" library contains the common mathematical function used in the scientific equations.

Common function from math.h library:	
Mathematical Expression	C++ Expression
$e^n$	Exp(x)
$\text{Log}(x)$	Log10(x)
$\text{Ln}(x)$	Log(x)
$\text{Sin}(x)$	Sin(x)
$x^n$	Pow(x,n)
$\sqrt{x}$	Sqrt(x)

Example:

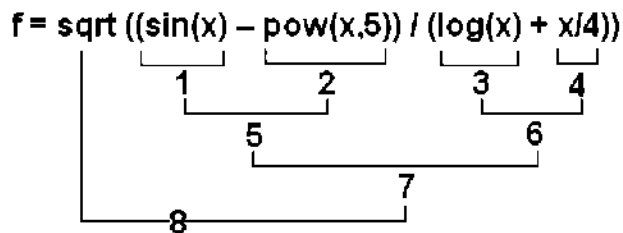
Write the following equation as a C++ expression and state the order of evaluation of the binary operators:

$$f = \sqrt{\frac{\sin(x) - x^5}{\ln(x) + \frac{x}{4}}}$$

Solution:

$$f = \text{sqrt} ((\sin(x) - \text{pow}(x,5)) / (\log(x) + x/4))$$

Order of evaluation:



Exercise:

Write the following equation as a C++ expression and state the order of evaluation of the binary operators:

$$z = \sqrt{\frac{x^2 y - 3 \sin(x)}{\tan x^3 + x^3 / y}}$$

Solution: ?

The ++ and - - operators can be written either before the variable (prefix notation) or after the variable (postfix notation) as in the following:

Prefix notation:	++ X	X is incremented before its value is taken or returned to current statement.
Postfix notation:	X ++	X is incremented after its value is taken or returned to current statement.

### The difference between the Prefix and Postfix notations:

#### Prefix notation

```
int y;
int x = 7;
cout << ++x << endl;
y=x;
cout << y;
```

#### Output:

8  
8

#### Postfix notation

```
int y;
int x = 7;
cout << x++ << endl;
y=x;
cout << y;
```

#### Output:

7  
8

### 3 Manipulator Functions:

They are special stream functions that change certain characteristics of the input and output.

(a) **Endl**: Generate a carriage return or line feed character.

```
Cout << "a" << endl;
```

(b) **Setbase**: It is used to convert the base of one numeric value into a nother base

**Dec(base 10), hex(base 16), oct(base 8)**

### Example 2

 Write C++ program to convert a base of a number:

```
#include<iostream.h>
void main( )
{
    int value;
    cout << "enter number:"; cin >> value;
    cout << "Decimal base=" << dec << value << endl;
    cout << "Hexadecimal base=" << hex << value << endl;
    cout << "Octa base=" << oct << value << endl;
}
```

```
Enter number
10
Decimal base=10
Hexadecimal base=a
Octal base=12
```

When using setbase the statement will be:

```
Cout<<"Decimal base="<<setbase(10);
```

```
Cout<<value<<endl;
```

(c) **setw**: It is used to specify the minimum number of character positions on the O/P field a variable will consume: **setw(int w)**

### Example 3

 Write C++ program to use tab:

```
#include<iostream.h>
#include<iomanip.h>
void main( void)
{
    int a,b;
    a=200;
    b=300;
    cout<<a<<'\t'<<b<<endl;
}
```

```
200 300
```

### Example 4

 Write C++ program to use setw:

```
#include<iostream.h>
#include<iomanip.h>
void main( void)
{
    int a,b;
    a=200;
    b=300;
    cout<<setw(5)<<a<<setw(5)<<b<<endl;
    cout<<setw(6)<<a<<setw(6)<<b<<endl;
}
```

```
200 300
200 300
```

(d) **setfill**: It is used to specify a different character to fill the unused field width of the value. **setfill(char f)**



### Example 5

 Write C++ program to use setfill:

```
#include<iostream.h>
#include<iomanip.h>
void main( void)
{
    int a,b;
    a=200;
    b=300;
    setfill('*');
    cout<<setw(5)<<a<<setw(5)<<b<<endl;
    cout<<setw(6)<<a<<setw(6)<<b<<endl;
}
```

```
**200**300
***200***300
```

(e) **Setfill**: It is used to control the number of digits of an output stream display of a floating point value. **Setprecision (int p)**

### Example 6

 Write C++ program to use setprecision:

```
#include<iostream.h>
#include<iomanip.h>
void main( void)
{
    float a,b,c;
    a=5; b=3; c=a/b;
    setfill('*');
    cout<<setprecision(1)<<c<< endl;
    cout<<setprecision(5)<<c<< endl;
}
```

```
1.7
1.66667
```

# WORK SHEET (2)

## First Elements of C++

- Q1: What do you mean by C++ character set?
- Q2: What do you mean by identifiers? What is the maximum length of identifiers?
- Q3: What do you mean by case-sensitive?
- Q4: What do you mean by reserved word?
- Q5: Write a general layout of C++ program. Comment on each part of it.
- Q6: What is the main purpose of endl and \n ?
- Q7: List and comment on the special escape codes.
- Q8: What are the main types of variables, their sizes, and their range of values?
- Q9: What do you mean by constants?
- Q10: List the priorities of the arithmetic operations.
- Q11: Find the value of A for the following:  
$$A = (5 + 2 * 3 + ((3 - 2) * 7) + -9) / 2.$$
- Q12: What are the main keywords included in iostream.h and math.h?
- Q13: What are the main differences between prefix and postfix notation?
- Q14: Find the value of B (true or false) for the following:  
i = 5;  
j = 9;

`B = ! ((i > 0) && (i >= j));`

Q15: Write C++ program to read x and compute sin, cos, and tan of x.

Q16: Rewrite the equivalent statements for the following examples, and find its results. Assume:  $X=2$ ,  $Y=3$ ,  $Z=4$ ,  $V=12$ ,  $C=8$ .

( $X+=5$ ,  $Y-=8$ ,  $Z*=5$ ,  $V/=4$ ,  $C\%=3$ )

Q17: Given that A and B are real variables with values 1.5, and 2.5 respectively, and C is integer variable with value 3, evaluate the following: NOT ( $A < 0$ ) AND ( $B/C \leq 0$ ).

Q18: Write a program in C++ to find the area of a circle.

Q19: Write a program to read a set of (5) real no.s and find out the sum and average of them.

# LECTURE 6

## 1. Selection Statements:

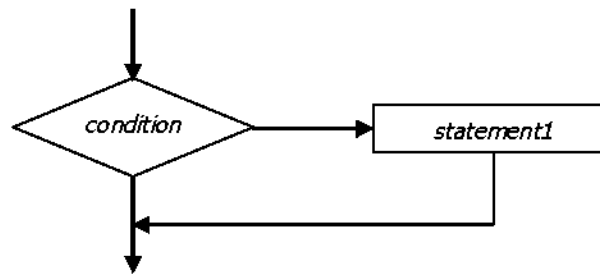
Conditional expressions are mainly used for decision making. C++ provides multiple selection structures: **if**, **if/else**, **else if**, **nested if** and **switch**.

## 2. The Single If Statement Structure:

The IF statement is used to express conditional expression. If the given condition is true then it will execute the statements; otherwise it will execute the optional statements.

**General Form of single-selection If statement:**

```
if ( expression or condition ) statement1 ;
```




Example 1:     if ( avrg >= 3.5 )  
                  cout << "good";

Example 2:     if ( x > 0.0 )  
                  sum += x;

Example 3:     cin >> num;  
                  if ( num == 0 )  
                  zcount = zcount + 1;

### Example 1

 Write a C++ program to read any two numbers and print the largest value of it:

```
#include<iostream.h>
void main( )
{
    Float x,y;
    Cout<<"Enter any two numbers\n";
    Cin>>x>>y;
    If (x>y)
    Cout << "largest value is"<<x<<endl;
}
```


### 3. The Single Block If Statement Structure :

The block IF statement are enclosed in ({} and {}) to group declaration and statements into a compound statement or a block. These blocks are always considered as a single statement. The structure is:

#### General Form of single block selection If statement:

```
if ( expression or condition )
{
    statement1 ;
    statement2 ;
    statement3 ;
}
```

### Example 2

 Write a C++ program to read a number and check if it's positive, if it's so print it, add it to a total, and decrement it by 2:

```
#include<iostream.h>
void main( )
{
    int num, total=0;
    cin >> num;
    if ( num >= 0 )
    {
        cout << num <<" is a positive";
        total += num;   num = num - 2;
    }
}
```

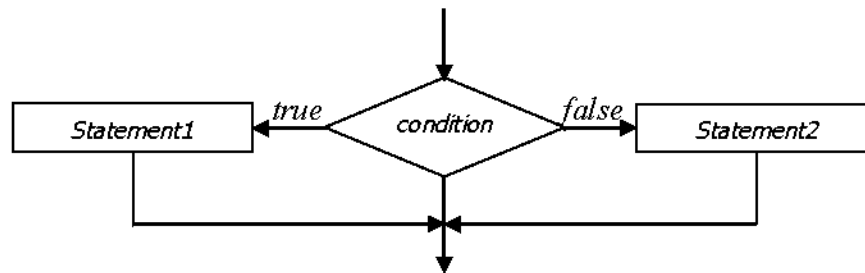
### General Form of If/else statement:

```
if ( expression )  
    statement1 ;  
else statement2 ;
```

```
if ( expression )  
    { statements }  
else { statements }
```

## 4. The If/else Statement Structure:

The IF structure is



In this case, either of the two statements are executed depending upon the value of the expression. Note that there is a semicolon after each of the statement but not after the IF expression. Note that the else statement without braces leads to **confusion** so:

```
If (i>j) { If (a>b)  
temp=a;  
    }  
Else  
temp=b;
```


Example 1:

```
cin >> value;  
if ( value >= 0 )  
    cout << "positive";  
else  
    cout << "negative";
```

Example 2:


```
cin >> num1 >> num2;
if ( num1 > num2 )
    cout << num1;
else
    cout << num2;
```

### Example 3

 Write a 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:

```
#include<iostream.h>
void main( )
{
    int degree;
    cin >> degree;
    if (degree >= 50 )
        cout << "pass";
    else
        cout << "fail";
}
```

### Example 4

 Write a C++ program to read a number, and check if it's even or odd:

```
#include<iostream.h>
void main( )
{
    int num;
    cin >> num;
    if ( num %2 == 0 )
        cout << "even";
    else
        cout << "odd";
}
```

## 5. Else if Statements:


### General Form of else if statement:

```
if ( expression or condition 1 ) statement1 ;  
else if ( expression or condition 2 ) statement2 ;  
else if ( expression or condition 3 ) statement3 ;  
:  
else if ( expression or condition n ) statement-n ;  
else statement-e ;
```

### Example 1:

```
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";
```


### Example 5

 Write a C++ program to read a number, and print the day of the week:

```
#include<iostream.h>  
void main( )  
{  
    int day;  
    cin >> day;  
    if ( day == 1 ) cout << "Sunday";  
    else if ( day == 2 ) cout << "Monday";  
    else if ( day == 3 ) cout << "Tuesday";  
    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 << "Invalid day number";  
}
```



### Example 6

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

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

```
#include<iostream.h>
void 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;
}
```

### 6. Nested If Statements:

Some of the samples of **NESTED if-else** constructions are shown below:

<pre>If (exp.) { Statements } Else     { Statements }</pre>	<pre>If (exp.) { If (exp.) {Statements} Else     { Statements} } Else     {Statements}</pre>	<pre>If (exp.) { If (exp.) {Statements} Else     { Statements} } Else     {If (exp) {Statements} Else {Statement} }</pre>
---	--	---

## Example 7



Write C++ program to find a largest value among three numbers:

```
#include<iostream.h>
void main( )
{
#include<iostream.h>
void main( )
{
Float x,y,z;
Cout<<"Enter any two numbers\n";
Cin>>x>>y,z;
If (x>y) {
If (x>z)
Cout << "largest value is"<<x<<endl;
Else
Cout << "largest value is"<<z<<endl;
}
Else If (y>z)
Cout << "largest value is"<<y<<endl;
Else
Cout << "largest value is"<<z<<endl;
}
```

# LECTURE 7

## 1. The Switch Selection Statement (Selector):

The switch statement is a special multi way decision maker that tests whether an expression matches one of the number of constant values, and braces accordingly.


### General Form of Switch Selection statement:

```
switch ( selector )
{
    case label1 : statement1 ; break;
    case label2 : statement2 ; break;
    case label3 : statement3 ; break;
    :
    case label-n : statement-n ; break;
    default : statement-e ; break;
}
```

### Example 1:


```
switch (value)
{
    case 0: cout << "grade is A";
           break;
    case 1: cout << "grade is B";
           break;
    case 2: cout << "grade is C";
           break;
    default: cout << "grade is X";
            break;
}
```

### Example 1

 Write C++ program to read integer number, and print the name of the day in a week:

```
#include<iostream.h>
void main( )
{
    int day;
    cout << "Enter the number of the day \n";
    cin >> day;
    switch (day)
    {
        case 1: cout << "Sunday";    break;
        case 2: cout << "Monday";   break;
        case 3: cout << "Tuesday";  break;
        case 4: cout << "Wednesday"; break;
        case 5: cout << "Thursday"; break;
        case 6: cout << "Friday";   break;
        case 7: cout << "Saturday";  break;
        default: cout << "Invalid day number"; break;
    }
}
```

### Example 2

 Write C++ program to read two integer numbers, and read the operation to perform on these numbers:

```
#include<iostream.h>
void main( )
{
    int a, b;
    char x;

    cout << "Enter two numbers \n";
    cin >> a >> b;

    cout << "+ for addition \n";
    cout << "- for subtraction \n";
    cout << "*" for multiplication \n";
    cout << "/" for division \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;
        case '/': cout << a / b;
                break;
        default: break;
    }
}

```

## 2. Nested Switch Selection Statement:


### General Form of Nested Switch Selection statement:

```

switch ( selector1 )
{
    case label1 : statement1 ; break;
    case label2 : statement2 ; break;
    case label3 : switch ( selector2 )
                    {
                        case label1 : statement1 ; break;
                        case label2 : statement2 ; break;
                        :
                    }
    case label-n : statement-n ; break;
    default : statement-e ; break;
}

```

### Example 3

 Write C++ program to read integer number, and print the name of the computerized department:

```

#include<iostream.h>
void main( )
{
    int i,j;
    cout << "Enter the number for the department name \n";
    cin >> i>>j;
    switch (i)
    {

```

```

case 1: cout << "Software Engineering Department"; break;
case 2: cout << "Control and computers Department"; break;
case 3: cout << "Computer Sciences Department";
        cout<<"Enter the no. of branch";
        switch(j)
{
case 1: cout << "Software"; break;
case 2: cout << "Information system"; break;
case 3: cout << "Security";
case 4: cout << "AI";
}
default: cout << "Invalid day number"; break;
}
}
}

```

### 3. Conditional Statement:

#### General Form of Conditional statement:

( *condition* ? True : False )

Example 1:     cin >> value;  
                   cout << (value >= 0 ? "positive" : "negative" );

Example 2:     cin >> x >> y;  
                   cout << ( x < y ? -1 : (x == y ? 0 : 1) );

#### Example 4

 Write C++ program to read integer number, and print if its even or odd:

```

#include<iostream.h>
void main( )
{
    int value;
    cout << "Enter the number \n";
    cin >> value;
    cout<<(value%2==0?"even":"odd");
}

```

# WORK SHEET (3)

## Selection Statements

Q1: Write C++ program to read two integer numbers then print "multiple" or "not" if one number is a multiple to another number.

Q2: Write C++ program to read integer number and print the equivalent string.

**e.g:**

0 → Zero  
1 → One  
2 → Two  
:

Q3: Write C++ program to read a score of student and print the estimation to refer it.

**e.g:**

100 - 90 → Exultant  
89 - 80 → Very good  
79 - 70 → Good  
69 - 60 → Middle  
59 - 50 → Accept  
49 - 0 → Fail

Q4: Write C++ program to represent a simple nested case (selector).

Q5: Write C++ program to compute the area of circle if the radius  $r=2.5$ .

**Note:** area of circle is  $r * r * pi$ ,  
 $pi$  is 3.14

Q6: Write C++ program to read an integer number and check if it is positive or negative, even or odd, and write a suitable messages in each case.

Q7: Write a program to read 3 numbers, and write the largest and smallest numbers.

Q8: Write C++ program to read an integer from 1 to 12, and print out the value of the corresponding month of the year.

Q9: 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 ( +, !, @, #, \\_ {, >, ... ).

Q10: Write C++ program to read x and compute the following:

$$Y = \begin{cases} \frac{x^2 + 5x - 20}{\sqrt{2x}} & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ x^2 + (5x)^2 - 10 & \text{if } x < 0 \end{cases}$$

Q11: Write C++ program to read 5 numbers and determine if the numbers sorted ascending or not.

Q12: Write C++ program to read two integer numbers, and read the operation to perform on these numbers.

Q13: Write a program to read X and print Sin X if X>0, square root X if X<0 and absolute X if X/2 is integer.

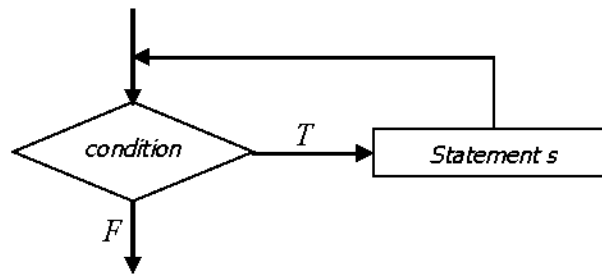
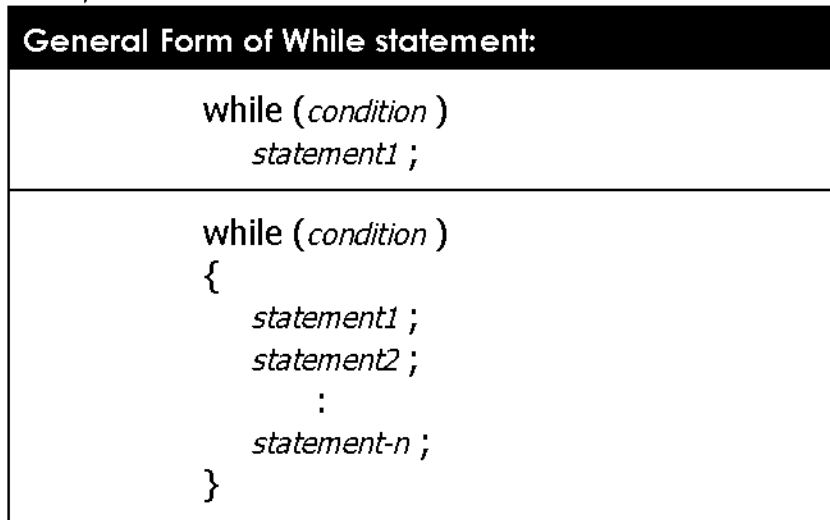


# LECTURE 8

## 1. Loop Statements:

The loop statements are essential to construct systematic block styled programming. C++ provides three iteration structures: **while**, **do/while**, and **for**.

## 2. While Repetition Structure:



The condition represents the value of a variable, unary or binary expression, and a value returned by a function.

Example    `i = 0;`  
1:        `while ( i < 10 )`  
          `{`  
          `cout << i;`  
          `i++;`  
          `}`

Output:  
0 1 2 3 4 5 6 7 8 9  
i=10

Example 2:

```

i = 0;
while ( i < 10 )
{
    cout << i;
    i += 2;
}

```

Output:      *even numbers only*  
0 2 4 6 8  
i=10

Example 3:


```

i = 1;
while ( i < 10 )
{
    cout << i;
    i += 2;
}

```

Output:      *odd numbers only*  
1 3 5 7 9  
i=11

### Example 1

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

$$\text{sum} = 1 + 3 + 5 + 7 + \dots + 99$$


*(in other words: find the summation of the odd numbers, between 0 and 100)*

```

#include<iostream.h>
void main( )
{
    int count = 1;
    int sum = 0;
    while ( count <= 99 )
    {
        sum = sum + count;
        count = count + 2;
    }
    cout << "sum is: " << sum << endl;
}

```

### Example 2

 Write C++ program to find the cub of a number, while it is positive:

```

#include<iostream.h>
void main( )
{
    int num, cubenum;
    cout << "Enter positive number \n";
    cin >> num;
    while ( num > 0 )
    {
        cubenum = num * num * num;
        cout << "cube number is :" << cubenum << endl;
    }
}

```

```
    cin >> num;
}
```

### Example 3



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

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

```
#include<iostream.h>
void 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;
}
```

### Example 4



Write C++ program to find the summation of student's marks, and it's average, assume the student have 8 marks:

```
#include<iostream.h>
void main( )
{
    int mark, i, sum = 0;
    float av = 0;
    i = 1;
    while ( i <= 8 )
    {
        cout << "enter mark: ";
        cin >> mark;
        sum = sum + mark;
        i++;
    }
    cout << "sum is: " << sum << endl;
    av = sum / 8;
    cout << "average is: " << av;
}
```

### Example 5



Write C++ program that display the following board pattern:

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

```
#include<iostream.h>
void main( )
{
    int row = 8, column;
    while ( row-- > 0 )
    {
        column = 8;
        if ( row % 2 == 0 )
            cout << " ";
        while ( column-- > 0 )
            cout << "*";
        cout << '\n';
    }
}
```

### Example 6

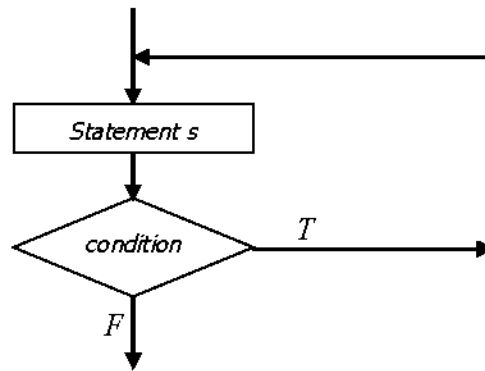


Write C++ program to check for a line feed and tab of a given character:

```
#include<iostream.h>
void main( )
{
    Char ch;
    Cout<<"enter a line \n";
    Ch=cin.get();
    While (ch!='\n' && ch!='\t')
    { cout.put(ch);
      Ch=cin.get(); }
}
```

### 3. Do / While Statement:

General Form of Do / While statement:
<pre>do     statement1 ; while (condition );</pre>
<pre>do {     statement1 ;     statement2 ;     :     statement-n ; } while (condition );</pre>



#### Example 1:

```
i = 0;
do
{
    cout << i;
    i++;
}
while ( i < 10 )
```

Output:

0 1 2 3 4 5 6 7 8 9  
i=10


#### Example 2:

```
i = 0;
do
{
    cout << i;
    i += 2;
}
while ( i < 10 )
```

Output:

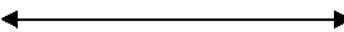
0 2 4 6 8  
i=10  
*even numbers only*

### Example 7


 Write C++ program to valid input checking, that accept the numbers between 50 ... 70 only:

```
#include<iostream.h>
void main( )
{
    int accept = 1;
    int x, low = 50, high = 70;
    do
    {
        cout << "enter number: ";
        cin >> x;
        if ( x >= low && x <= high )
            accept = 1;
        else
            accept = 0;
    }
    while ( ! accept );
```

*while (accept == 1) or  
while (accept != 0)*



### Example 8

 Write C++ program to find the summation of student's marks, and it's average, assume the student have 8 marks:

```
#include<iostream.h>
void main( )
{
    int mark, i, sum = 0;
    float av = 0;
    i = 1;
    do
    {
        cout << "enter mark: ";
        cin >> mark;
        sum = sum + mark;
        i++;
    }
    while ( i <= 8 )
    cout << "sum is: " << sum << endl;
    av = sum / 8;
    cout << "average is: " << av;
}
```

### Example 9



Write C++ program to find the factorial of n:

$$n! = n * n-1 * n-2 * n-3 * \dots * 2 * 1$$

```
#include<iostream.h>
void main( )
{
    int n, f = 1;
    cout << "enter positive number: ";
    cin >> n;
    do
    {
        f = f * n;
        n --;
    }
    while ( n > 1 );
    cout << "factorial is: " << f;
}
```

### Example 10



Write C++ program to find the summation of even numbers

```
#include<iostream.h>
void main( )
{
    int max,sum,digit;
    digit=2;
    cout << "enter a number: ";
    cin >> max;
    sum=0;
    do
    {
        Sum=sum+digit;
        Digit+=2;
    }
    while ( digit<=max );
    cout << "2+4+...="<<max<<"sum="<<sum<<endl; }
```

# LECTURE 9

## 1. For Statement:

### General Form of For statement:

```
for ( initialization ; continuation condition ; update )  
    statement1 ;
```

```
for ( initialization ; continuation condition ; update )  
{  
    statement1 ;  
    statement2 ;  
    :  
}
```

Example 1:     for ( i = 0; i < 10; i++ )  
                  cout << i;

Output:

0 1 2 3 4 5 6 7 8 9

Example 2:     for ( i = 0; i < 10; i += 2 )  
                  cout << i;

Output:

*even numbers only*

0 2 4 6 8

Example 3:     for ( i = 1; i < 10; i += 2 )  
                  cout << i;

Output:

*odd numbers only*

1 3 5 7 9

### Example 1



Write C++ program to add the numbers between 1 and 100:

```
#include<iostream.h>  
void main( )  
{  
    int sum = 0;  
    for ( int i = 1; i <= 100; i ++ )  
        sum = sum + i;  
    cout << "sum is: " << sum;  
}
```



## Example 2



Write C++ program to find the factorial of n (using for statement):

$$n! = n * n-1 * n-2 * n-3 * \dots * 2 * 1$$

```
#include<iostream.h>
void main( )
{
    int n, f = 1;
    cout << "enter positive number: ";
    cin >> n;
    for ( int i = 2; i <= n; i ++ )
        f = f * i;
    cout << "factorial is: " << f;
}
```

for ( int i = n; i > 2; i -- )

←

## Example 3



Write C++ program to the result of the following:

$$\sum_{i=1}^{20} a_i^2$$

```
#include<iostream.h>
void main( )
{
    int sum = 0;
    for ( int i = 1; i <= 20; i ++ )
        sum = sum + ( i * i );
    cout << "The sum is: " << sum;
}
```


## Example 4



Write C++ program to read 10 integer numbers, and find the sum of positive number only:


```
#include<iostream.h>
void main( )
{
    int num, sum = 0;
    for ( int i = 1; i <= 10; i ++ )
    {
        cout << "enter your number: ";
        cin >> num;
        if ( num > 0 )    sum = sum + num;
    }
    cout << "The sum is: " << sum;
}
```

### Example 5

 Write C++ program to print the following series: 1, 2, 4, 8, 16, 32, 64

```
#include<iostream.h>
void main( )
{
    int x;
    for ( x = 1; x < 65; x *= 2 )
        cout << x << " ";
}
```

### Example 6

 Write C++ program to print the following:

```
#include<iostream.h>
void 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

### Example 7

 Write C++ program to read a line using for loop

```
#include<iostream.h>
void main( )
{
    Char ch;
    cout << "Enter a line\n";
    for (:(ch=cin.get())!='\n'); {
        cout<<"Your character is:"<<endl;
        cout.put(ch);
    }
}
```

## 2. More about For Statement:

We can use more than one 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 ( ; ; )`




```

#include<iostream.h>
void main( )
{
    int i, j;
    for ( i = 1; i <= 10; i ++ )
    {
        for ( j = 1; j <= i; j ++ )
            cout << " + ";
        cout << "\n";
    }
}

```

### Example 10

 Write C++ program to read a line using for loop

```

#include<iostream.h>
void main( )
{
    cout << "Explaining the nested for loop\n";
    for (int i=0;i<=2;i++) {
        cout<<i;
        for (int k=0;k<=2;k++) {
            cout<<"computer sciences department \n";
        }
    }
}

```

Exercise:

*What is the output of the following C++ program ?*

```

#include<iostream.h>
void 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";
        }
        cout << "\n";
    }
}

```



### Example 1

 Write C++ program to check if zero or negative value found:

```
#include<iostream.h>
void main( )
{
    int value,i;
    i=0;
    while (i<=10) {
        cout<<"enter a number \n";
        cin>>value;
        if (value<=0) {
            cout<<"Zero or negative value found \n";
            brek;
        }
        i++;
    }
}
```

### (b) Continue Control Statements:

The continue is used to repeat the same operations once again even if it checks the error. Its general syntax is: (**continue;**)

It is used for the inverse operation of the break statement. The following program segment will process only the positive integers. Whenever a zero or negative value is encountered, the computer will display the message "zero or negative value found" as an error and it continues the same loop as long as the given condition is satisfied.

```
Cin>>value;
While (value <=100) {
If (value <=0)
Cout<<"zero or negative value found\n";
Continue;
} }
```

#### Example 1:

```
do
{
    cin >> x;
    cin >> n;
    if ( n < 1 )    continue;
    cout << x;
```

#### Example 2:

```
n = 1;
for ( i = 1; i < 5; ++i )
{
    cin >> x;
    n = 5 * x++ * (-1) / n;
```

<pre>-- n; } while ( n &lt; 1 );</pre>	<pre>if ( n &lt; 1 ) continue; cout &lt;&lt; n; }</pre>
--	---

### (c) Goto Statement:

The goto statement is used to alter the program execution sequence by transferring the control to some other part of the program. Its general syntax is: (**goto label;**)

There are two ways of using this statement:

1. **Unconditional Goto:** It is used just to transfer the control from one part of the program to the other part without checking any condition. It is difficult in use.

#### Example 2

 Write C++ program to check if zero or negative value found:

```
#include<iostream.h>
void main( )
{
    Start: cout<<"***\n";
    Goto start;
}
```

2. **Conditional Goto:** It is used to transfer the control of the execution from one part of the program to the other in certain conditional cases.

#### Example 2

 Write C++ program to check if zero or negative value found:

```
#include<iostream.h>
void main( )
{
    Int value,i=0;
    While i<=10) {
    Cout<<"enter a number \n";
    Cin>>value;
    Cout<<"zero or negative value found \n";
    Goto error;
}
```

```

Error:
    Cout<<"input data error \n";
}

```

Using For Statement	Using While Statement	Using Do/While Statement
---------------------	-----------------------	--------------------------

Q1: Find the summation of the numbers between 1 and 100.

```

for( i=1 ; i<=100 ; i++ )
    s = s + i;

```

```

i = 1;
while ( i <= 100)
{
    s = s + i;
    i++;
}

```

```

i = 1;
do
{
    s = s + i;
    i++;
}
while ( i <= 100 );

```

Q2: Find the factorial of n.

```

cin >> n;
for( i=2 ; i<=n ; i++ )
    f = f * i;

```

```

cin >> n;
i = 2;
while ( i <= n)
{
    f = f * i;
    i++;
}

```

```

cin >> n;
i = 2;
do
{
    f = f * i;
    i++;
}
while ( i <= n);

```

Q3: To find the result of the following:  $\sum_{i=1}^{20} a_i^2$ .

```

for( i=1 ; i<=20 ; i++ )
    s = s + (i*i);

```

```

i = 1;
while ( i <= 20)
{
    s = s + (i*i);
    i++;
}

```

```

i = 1;
do
{
    s = s + (i*i);
    i++;
}
while ( i <= 20);

```

Q4: Read 10 numbers, and find the sum of the positive numbers only.

```

for( i=1 ; i<=10 ; i++ )
{
    cin >> x;
    if ( x>0 ) s = s + x;
}

```

```

i = 1;
while ( i <= 10)
{
    cin >> x;
    if ( x>0 ) s = s + x;
    i++;
}

```

```

i = 1;
do
{
    cin >> x;
    if ( x>0 ) s = s + x;
    i++;
}
while ( i <= 10);

```



Q5: Represent the following series: 1, 2, 4, 8, 16, 32, 64.

```
for( i=1 ; i<65 ; i*=2 )  
    cout << i;
```

```
i = 1;  
while ( i<65)  
{  
    cout << i;  
    i*=2;  
}
```

```
i = 1;  
do  
{  
    cout << i;  
    i*=2;  
}  
while ( i<65);
```

---

Q6: Find the sum of the following  $s = 1 + 3 + 5 + 7 + \dots + 99$ .

```
for( i=1 ; i<=99 ; i+=2 )  
    s = s + i;
```

```
i = 1;  
while ( i<=99)  
{  
    s = s + i;  
    i+=2;  
}
```

```
i = 1;  
do  
{  
    s = s + i;  
    i+=2;  
}  
while ( i<=99);
```

---

Q7: Find the sum and average of the 8 degrees of the student.

```
for( i=1 ; i<=8 ; i++ )  
{  
    cin >> d;  
    s = s + d;  
}  
av = s / 8;
```

```
i = 1;  
while ( i<=8)  
{  
    cin >> d;  
    s = s + d;  
    i++;  
}  
av = s / 8;
```

```
i = 1;  
do  
{  
    cin >> d;  
    s = s + d;  
    i++;  
}  
while ( i<=8);  
av = s / 8;
```

---

Q8: Find the cub of n numbers, while the entered number is a positive.

*Can't be solve this problem  
using For statement*

```
cin >> x;  
while ( x > 0 )  
{  
    c = x * x * x;  
    cin >> x;  
}
```

```
do  
{  
    cin >> x;  
    c = x * x * x;  
}  
while ( x > 0 );
```

# WORK SHEET (4)

## Iteration Statements

### Using While Statement:

- Q1: Write C++ program to find the summation of the odd numbers, between 0 and 100.
- Q2: Write C++ program to inverse an integer number.  
For example: 765432 → 234567
- Q3: Write C++ program to find G.C.D between m & n.
- Q4: Write C++ program to display the first 100 odd numbers.

### Using Do/While Statement:

- Q5: What are the output of the following segment of C++ code:

```
int i;
i = 12;
do
{
    cout << i << endl;
    i--;
}
while ( i > 0 );
```

- Q6: What are the output of the following segment of C++ code:

```
int count = 1;
do
{
    cout << ( count % 2 ? "*****" : "+++++" ) << endl;
    ++ count;
}
while ( count <= 10 );
```

- Q7: Write C++ program that utilize looping and the escape sequence `\t` to print the following table of value:

N	10 * N	100 * N	1000 * N
---	--------	---------	----------

1	┌──────────┐	10	100	1000
2		20	200	2000
3		30	300	3000
4		40	400	4000

Hint:\t to print six spaces.

**Using For Statement:**

- Q8: Write C++ program to read 7 marks, if pass in all marks ( $\geq 50$ ) print "pass" otherwise print "fail".
- Q11: Write C++ program to add the numbers between 1 and 100 and find its average.
- Q12: Write C++ program to print the following figures:



- Q13: Write C++ program to find e from the following series:  

$$e = 1 + (1/1!) + (1/2!) + (1/3!) + \dots + (1/n!)$$
- Q14: Write C++ program to find e from the following series:  

$$e = 1 + x + (x^2 / 2!) + (x^3 / 3!) + \dots (x^a / a!)$$
- Q15: Write C++ program to read 10 marks, suppose the student pass if all marks greater than or equal 50, and the average greater than or equal 50. If student fails in some lessons then print the number of these lessons, if student fails in average then print "fail in average".
- Q16: What is the output of the following C++ segment of code:  

```

for ( ; ; )
{
    cout << "enter your number: ";

```

```

cin >> x;
if ( x % 2 == 0 ) continue;
if ( x % 3 == 0 ) break;
cout << "Bottom of loop" << endl;
}

```

Q17: What is the output of the following C++ segment of code:

```

for ( l = 0; l < 8; l ++ )
{
    if ( l % 2 == 0 )    cout << l + 1 << endl;
    else if ( l % 3 == 0 ) continue;
    else if ( l % 5 == 0 ) break;
    cout << "end program \n";
}
cout << "end ...";

```

Q18: Write C++ program to print the following figure:

```

1
2 1
3 2 1
4 3 2 1
5 4 3 2 1

```

Q19: Write C++ program to print the following series:

1.  $Sum = 1 + 2^2 + 4^2 + \dots + n^2$
2.  $Sum = 1 - 3^x + 5^x - \dots + n^x$
3.  $Sum = 1 + 1/1! + 2/2! + 3/3! + \dots + n/n!$       where  $n! = 1 * 2 * 3 * \dots * n$