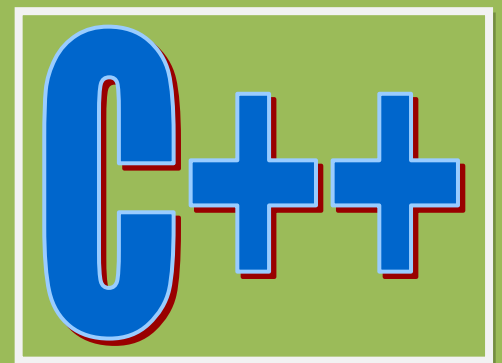


**Ministry of Higher Education &
Scientific Research
Al- Mustansiriya University
College of Engineering**

First Year

Computer Programming



Lect. Emad A. Hussien

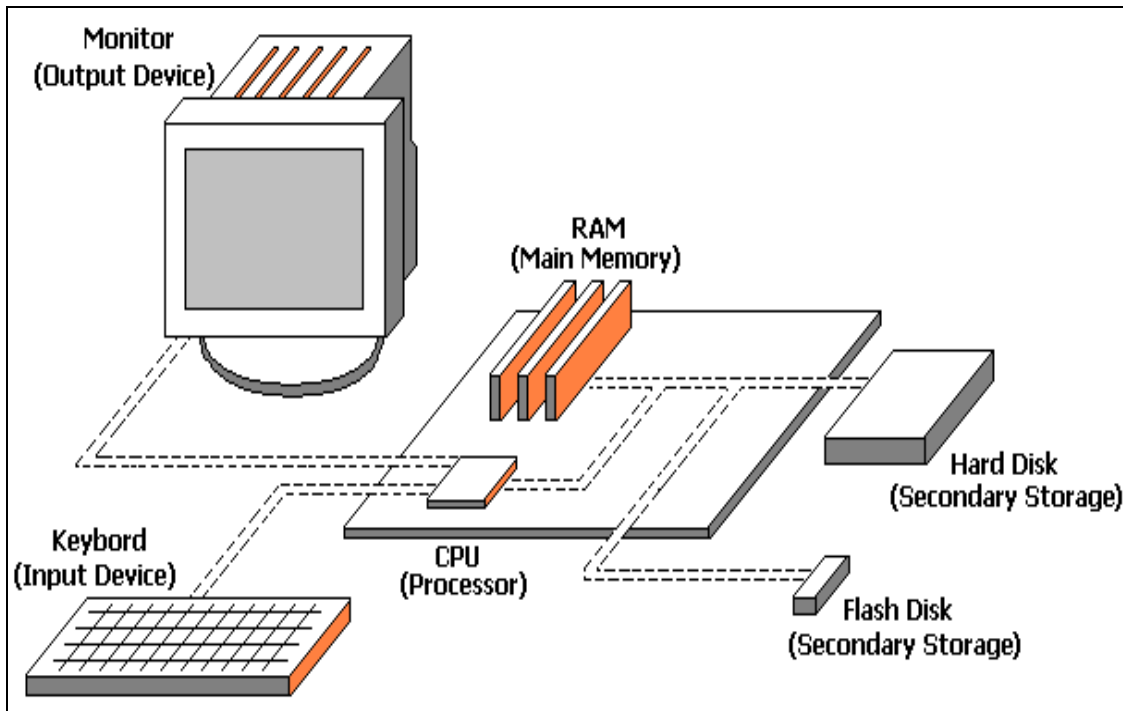
Lect. Gregor A. Armise

Asst.Lect.Majid E. Dobakh

2018 - 2019

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&T Labs

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:

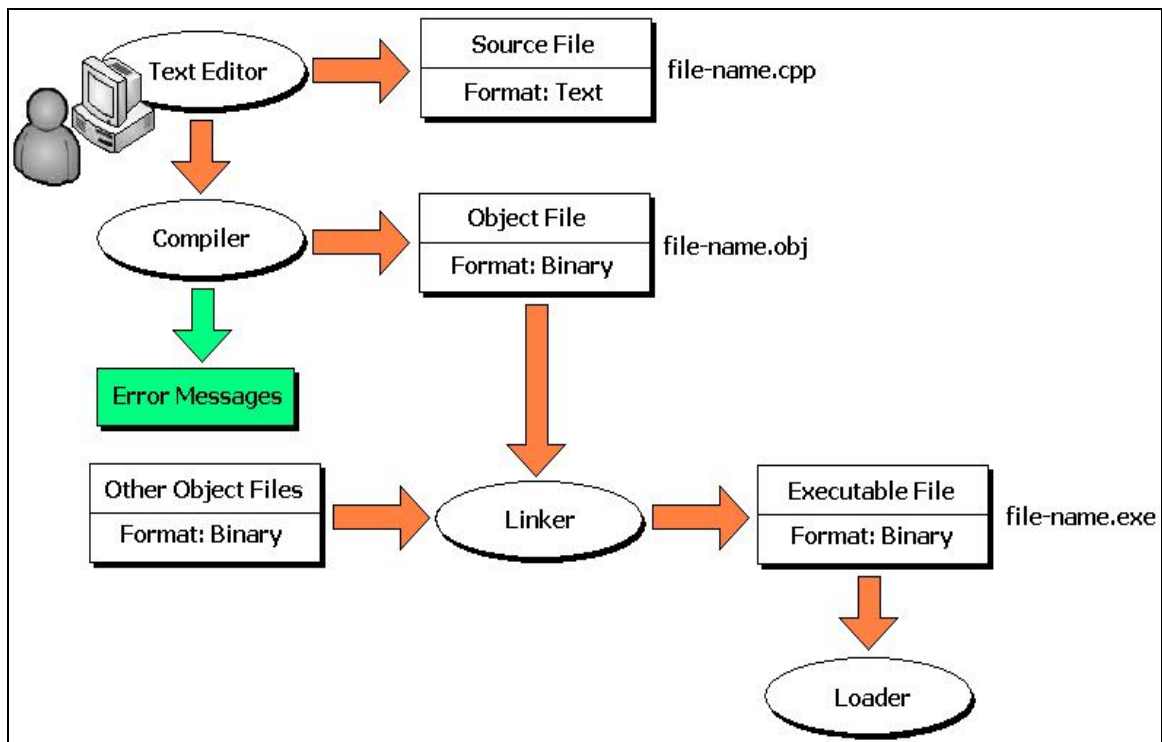
1. It supports all features of both structured programming and OOP.
2. C++ focuses on function and class templates for handling data types.

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. Algorithms

It is a combination of phrases and events that can be arranged as steps to solve a specific problem. That can be done by understanding this problem whether it mathematic or logic before convert it to flow chart.

Example-1: Write an algorithm to read five numbers then find and print their summation.

Solution: A computer algorithm can only carry out simple instruction like:

- ☐ "Read a numbers".
- ☐ "Add a number to anther number".
- ☐ "Output the summation".

Thus an algorithm is:

1. Start
2. Read L, M, N, O, P
3. Find Sum (S) , $S = L + M + N + O + P$
4. Print S
5. End

Example-2: Write an algorithm to read find the area of rectangle. Enter the length and width then print the result.

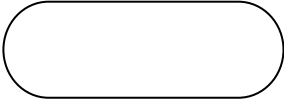
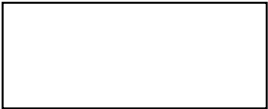
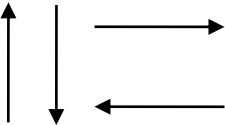
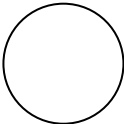
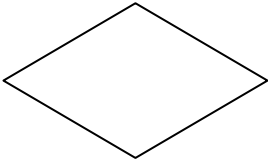
Solution: A computer algorithm can only carry out simple instruction like:

- ☐ "Enter length and width".
- ☐ "Fin the Area of the rectangle".
- ☐ "Output the Area".

1. Start
2. Input L, W
3. Find Area (A) , $A = L * W$
4. Print A
5. End

2: Flowcharts

Flowcharts are graphs that represent the formal view used to solve any problem. Flowcharts help the programmer to write his program. Flowcharts consist of a shapes connected by a straight lines. The following table shows these shapes and their operations.

Shape	Operation
	Start or End chart
	Input / Output data • Read or Input • Print
	Processing / Storing • Add(+) , Sub(-) , Mul(*) , Div(/) , Power(^), ... - Store a value (Put)
	Flow Lines
	Connector
	Decision • If statement • Question (?)
	Looping / Counters

The advantage of using algorithms and flowcharts are:

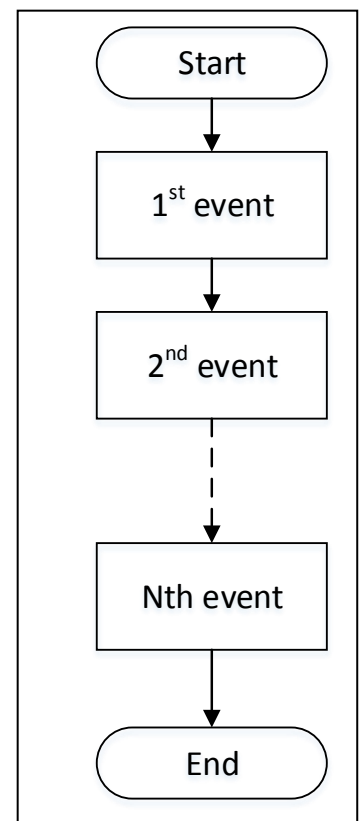
1. To show the mathematical logic used to solve problems.
2. To show how the data processing is done.
3. Helps the programmer to write his program.
4. Divides the big problem to smaller parts.
5. To avoid the errors that occurred during writing the program.
6. It is a middle step between problem difficulty and its conversion to suitable program.
7. Easy to convert it to any programming language.

In general, we can divide flowcharts to a four shapes (charts):

1. **Simple sequence charts**
2. **Branched charts.**
3. **Single loop charts.**
4. **Multi-loop charts.**

2-1: Simple sequence charts

The events arrangement of this type is as straight sequence from the beginning of the program to the end without any branches or repetition (see figure beside).



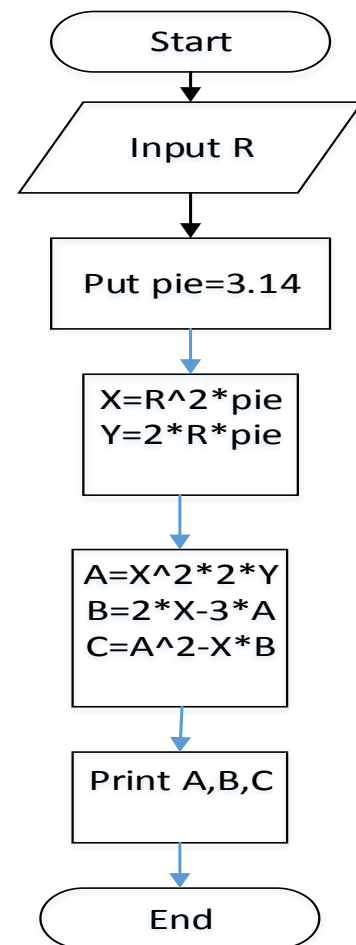
Example-3: Write an algorithm and draw a flow chart to find the value of A, B, C from the following equations:

$$A = X^2 + 2Y, \quad B = 2X - 3A, \quad C = A^2 - XB$$

Where X is the circle area and Y is the circumference. Input the radius (R) and print the value of A, B and C.

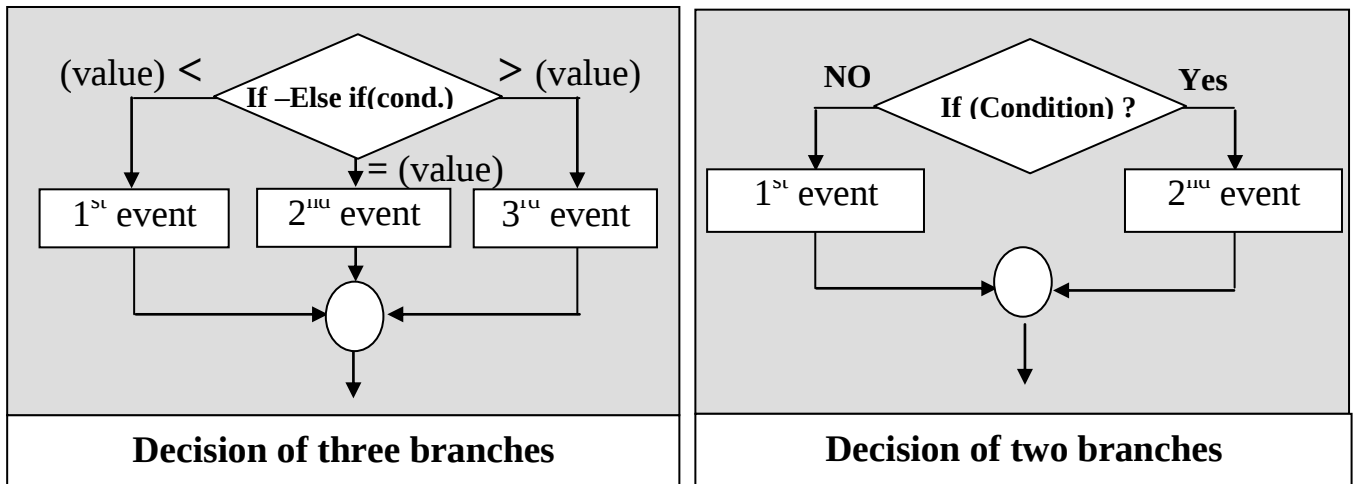
Solution:

1. Start
2. Input Radius (R)
3. Put pie = 3.14
4. Find Area (X), $X = R^2 * \text{pie}$
5. Find Circumference (Y), $Y = 2 * R * \text{pie}$
6. Find A, $A = X^2 + 2 * Y$
7. Find B, $B = 2 * X - 3 * A$
8. Find C, $C = A^2 - X * B$
9. Print A, B, C
10. End



2-2: Branched charts

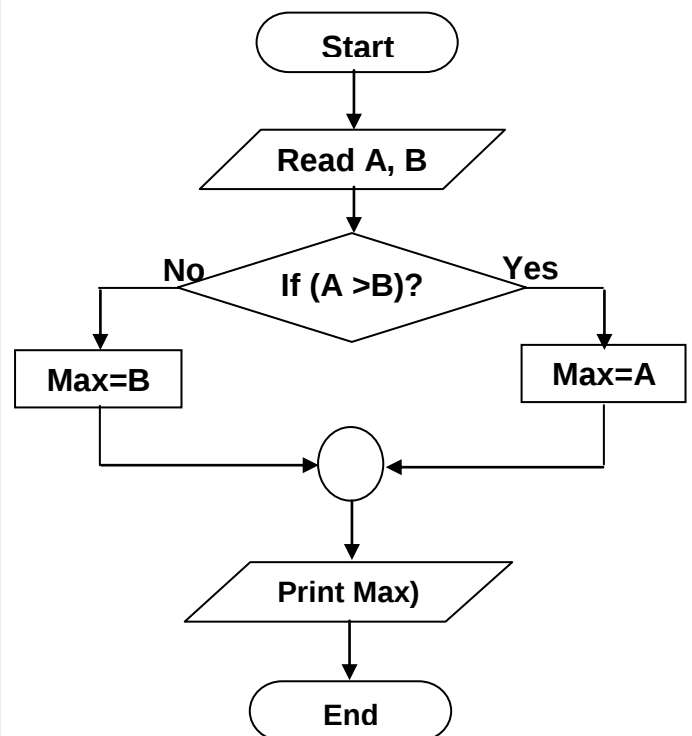
The need for the branching is to make decisions or comparison between two or more choices. Each choice will flow in different way (branch). Generally the branched charts may take one of the two forms below:



Example-4: Write an algorithm and draw a flow chart to find and print the maximum number between two numbers.

Solution

1. Start
2. Read A , B
3. If (A > B) ; continue.
Else: goto-5.
4. Find maximum (Max = A) : goto-6
5. Find maximum (Max = B).
6. Print Max
7. End

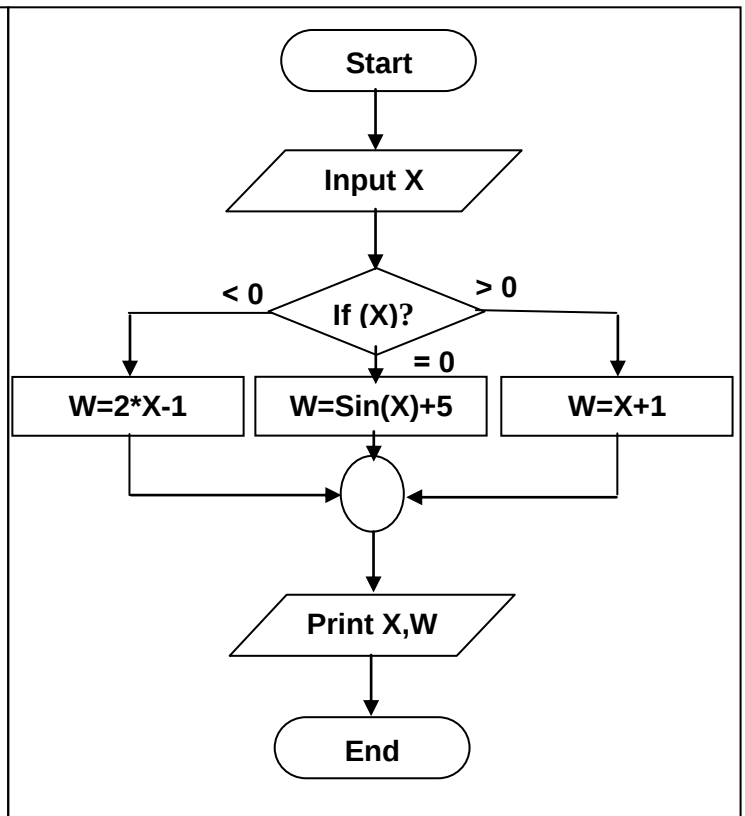


Example-5: Write an algorithm and draw a flow chart to evaluate W from the below equation. Input X and print the value of W for each value of X.

$$W = \begin{cases} X+1 & : X > 0 \\ \sin(X) + 5 & : X = 0 \\ 2X-1 & : X < 0 \end{cases}$$

Solution:

1. Start
2. Input x
3. If (x > 0) ; continue
 elseif (x = 0) ; goto-5
 elseif (x < 0) ; goto -6
4. Find W, W = X+1 : goto-7
5. Find W, W = Sin(X)+5: goto-7
6. Find W, W = 2*X-1
7. Print X , W
8. End



2-3: Single loop charts

These charts are used when we need to repeat an operation or group of operations to specific number of times. These types of charts are used to create the counters. Counters are used to repeat an operation or group of operations in specific number. Counters can be classified to two forms:

- General form.
- Conventional form.

1- General form

This type of counters can be represented according to the following form:

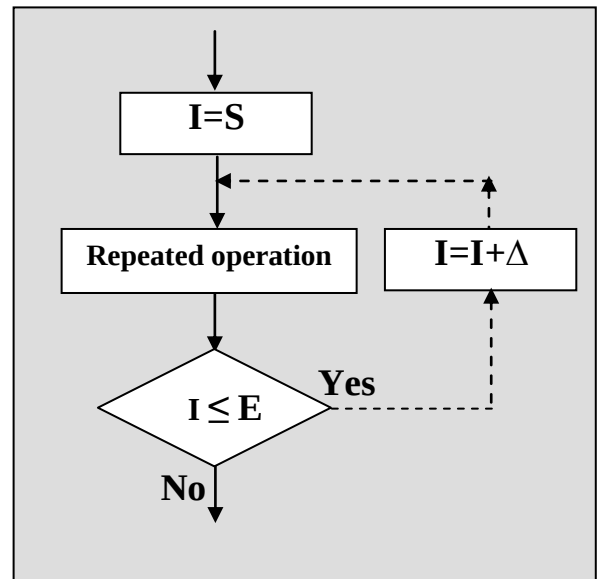
Where:

I : Counter name (may take any symbol)

S: Initial (Starting) value

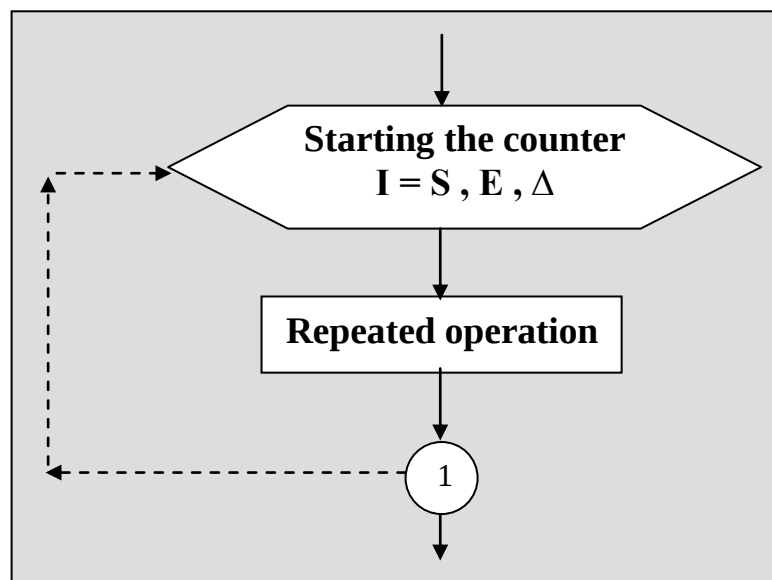
E: End (Final) value

Δ : Step size (if not indicated, its value will be 1)



2. Conventional form.

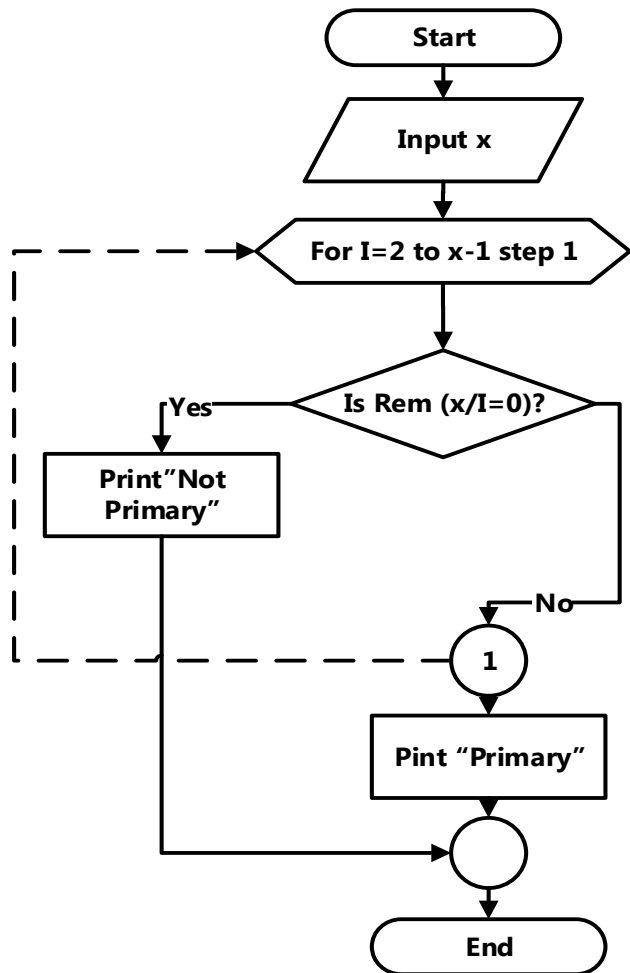
The form of this type of counters will be as shown:



Example-6: Write an algorithm and draw a flow chart to enter any number and check it whether its primary number or not. Use the conventional form.

Solution:

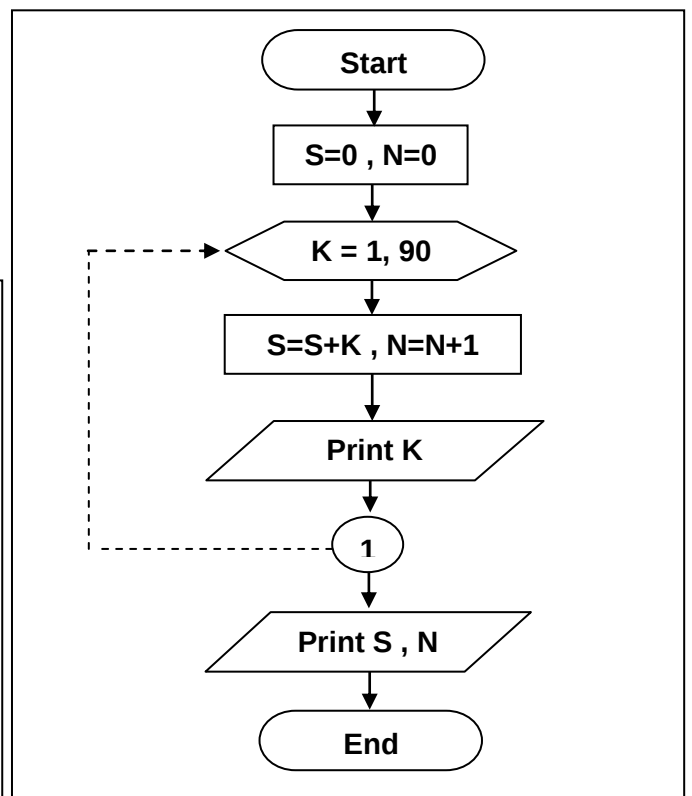
1. Start
2. Input x
3. For I=2 To x-1 step 1
4. If Remainder ((x / I) = 0); goto-7
Else; continue
5. Next I
6. Print "Primary number";goto-8
7. Print "Not a primary number"
8. End



Example-7: Write an algorithm and draw a flow chart to print the odd numbers from 1 to 90 then find and print its number and sum. Use the conventional form.

Solution:

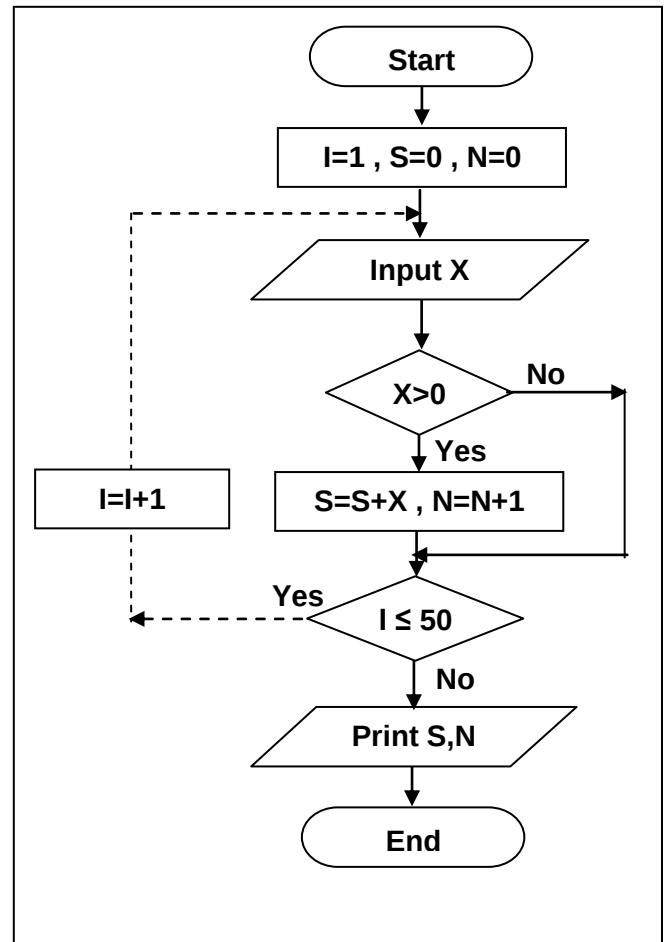
1. Start
2. Put S = 0 , N = 0
3. For K=1 To 90 step 1
4. Find sum of odds, S=S+K
6. Find number of odds, N=N+1
7. Print K
8. Next I
9. Print S , N
10. End



Example-8: Write an algorithm and draw a flow chart find the sum and number of positive numbers among 50 numbers entered by user. Use the general form.

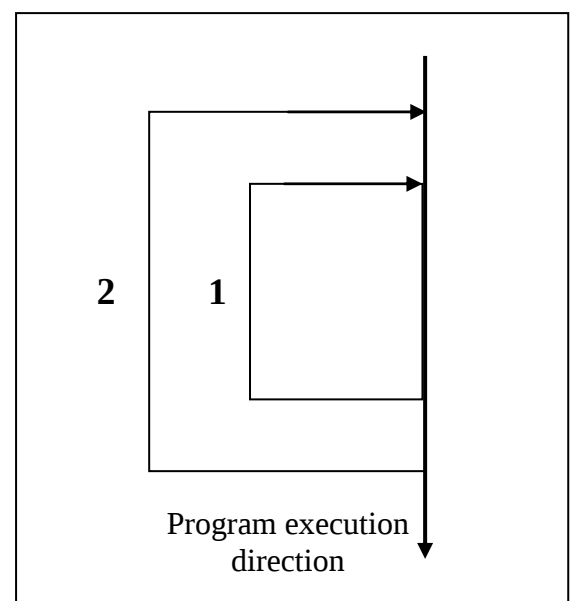
Solution:

1. Start
2. Put $I = 1, S=0, N=0$
3. Input x
4. If $(x \geq 0)$; continue
Else ; goto-7
5. Find Sum (S), $S=S+x$
6. Find Number (N), $N=N+1$
7. If $(I \leq 50)$; $I=I+1$; goto-3
Else ; continue
8. Print S , N
- 9.End



2-2-4: Multi-loops charts.

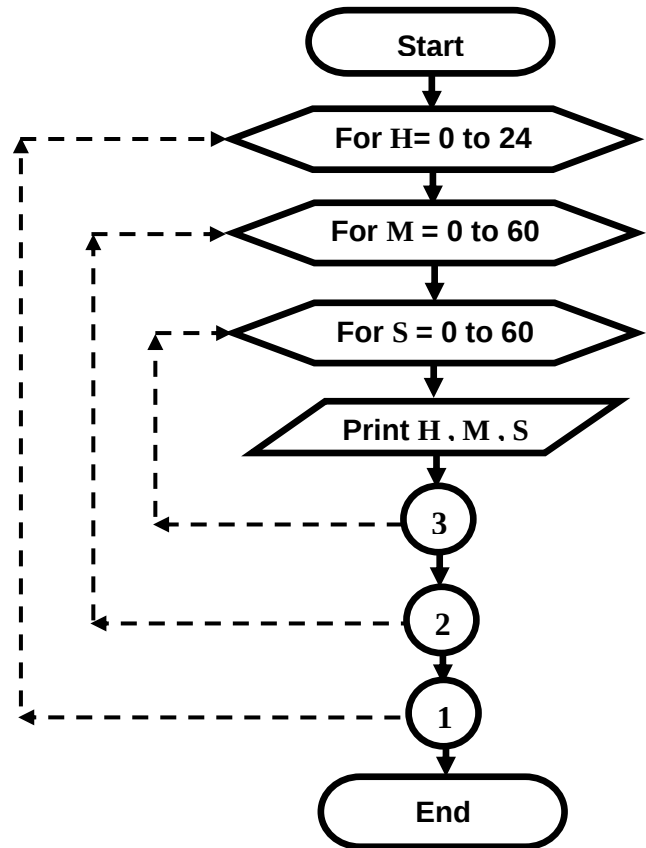
Its so called because it contains many loops. These loops are nested together without any intersection. Loop no.1 (shown in figure beside) is called "inner loop" and loop no.2 is the outer loop. The priority of execution will be to the inner loops then sequentially to the outer loops.



Example-8: Write an algorithm and draw a flowchart that shows how did the digital clock worked.

Solution:

1. Start
2. For H(Hours) = 0 to 24
3. For M(Minutes) = 0 to 60
4. For S(Seconds) = 0 to 60
5. Print H, M, S
6. Next S
7. Next M
8. Next H
9. End

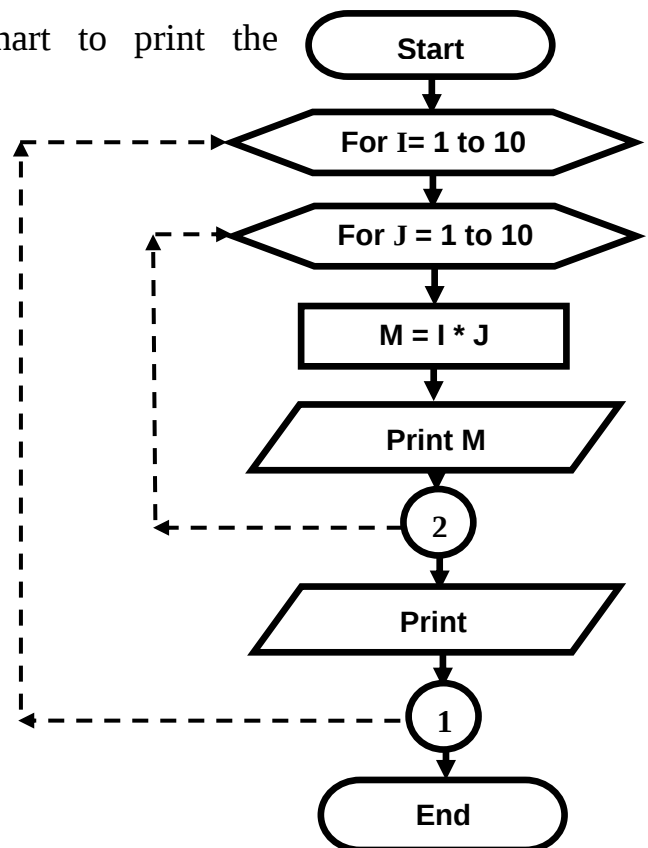


Example-9

Write an algorithm and draw a flowchart to print the multiplication table from 1 to 10.

Solution:

1. Start
2. For I = 1 to 10
3. For J =1 to 10
4. $M = I * J$
5. Print M
6. Next J
7. Print
8. Next I
9. End



WORK SHEET (1)

Q1: Write an algorithm and draw a flowchart to calculate A1, A2, A3 from the following equation:

$$A1 = X^2 + Y^2, \quad A2 = A1 - 3XY, \quad A3 = |A1| |A2|$$

Where X and Y are evaluated from the equations:

$$X, Y = \sqrt[3]{Z} \mp 4\sqrt{Z}$$

Print A1, A2, A3 for each input value of z.

Q2: Write an algorithm and draw a flowchart to calculate W from the following equations:

$$W = \begin{cases} X + Y & : X > 0, Y < 0 \\ X^2 Y^3 + 5X & : X = 0, Y > 0 \\ \frac{2X}{3Y} + 4X & : X < 0, Y = 0 \end{cases}$$

Print W for each input value of X and Y.

Q3: Draw a flowchart to enter a number represents a Celsius degree (C) then convert it to Fahrenheit degree (F) then print the grade of this degree using the following relations:

“Cold” when $F \leq 41$.

“Nice” when $41 < F \leq 77$.

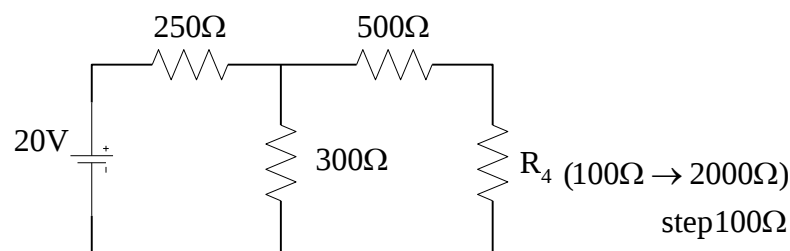
“Hot” when $F > 77$.

Hint: $F = (9C / 5) + 32$

Q4: Write an algorithm and draw a flowchart to find Y from the bellow equations for seven entering values of X. If you know that a=-8, print the value of Y for each value of X. Use the conventional form.

$$Y = \begin{cases} 3X^2 - |X| & : X \geq 0 \\ X + a & : X < 0 \end{cases}$$

- Q5:** A list contains N-degrees in computer course, draw a flowchart to calculate the percentage rate for the degrees limited between 60% and 70%. Use the conventional form.
- Q6:** Write an algorithm and draw a flowchart to enter a 40 numbers then find and print number and sum for the numbers greater than 0 and less than 25. Use the general form.
- Q7:** Write an algorithm and draw a flowchart to enter a 50 numbers then find and print number and sum for the positive and negative numbers. Use the conventional form.
- Q8:** Write an algorithm and draw a flowchart to find and print number and sum for the even and odd numbers for M-entered numbers. Use the conventional form.
- Q9:** By using the conventional form, draw a flowchart to find and print the power on the resistor R_4 .



- Q10:** Enter two numbers represents two readings for an electrical power consumption in the first and end of a specific month. Draw a flowchart to evaluate the consumption wage for 50 consumers according to the following table:

Consumption unit	Cost (in Dinar)
1 ← 300	8
301 ← 360	10
361 ← 900	20
≥ 901	30

Hint: $W = (R_2 - R_1) \times \text{Cost per unit}$

- Q11:** A list contains six degrees for 100 students, draw a flowchart to enter these degrees and find the average then print the grade of this average according to table below. Use the conventional form.

Average	Grade
0 - 49	Fail
50 - 59	Accept
60 - 69	Fluent
70 - 79	Good
80 - 89	Very Good
90 - 100	Excellent

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 65535) 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
\n	New line. Position the screen cursor to the beginning of the next line.
\t	Horizontal TAB (six spaces). Move the screen cursor to the next tab stop.
\r	Carriage return. Position the cursor to the beginning of the current line, do not advance to the next line.
\a	Alert. Produces the sound of the system bell.
\b	Back space
\\	Backslash. Prints a backslash character.
\f	Form feed
\v	Vertical tab
\"	Double quote. Prints a (") character.
\o	Null character
\?	question mark
\ooo	Octal value
\xhhh	Hexadecimal value

5. C++ operators:

C++ operators	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:

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

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

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

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 << 3

0 01000010

0 10000100

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

X=5, y=2 → **x&y** (0000) , **x|y** (0111) , **x^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 ≠ 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)? True : 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 bet 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>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): COUT/ 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++ let's 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.

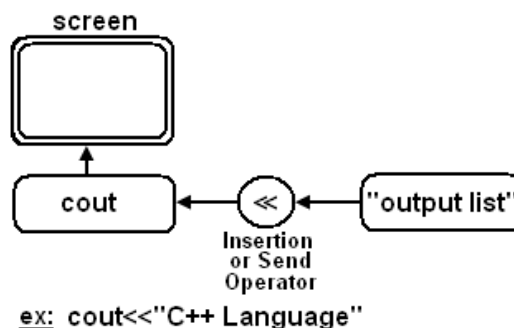
Every group of related functions is stored in a separate library called (header file).To use the *cin* and *cout*, must include the header file *iostream*.

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

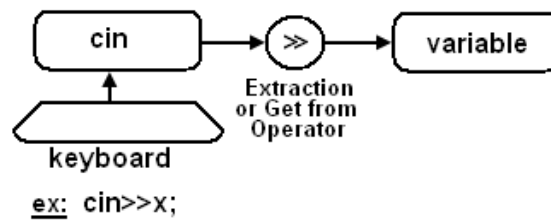
- ✓ **void**, is the return type of the main function. When the return type of a function is void, this function will not passes back any value to the calling function.

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:

Example 2

```
#include<iostream.h>
void main( )
{
    cout << "hallow" << endl;
    cout << "students";
}
```

Output:

```
hallow
students
```

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

Example 3

```
#include<iostream.h>
void main( )
{
    cout << "hallow \n";
    cout << "students";
}
```

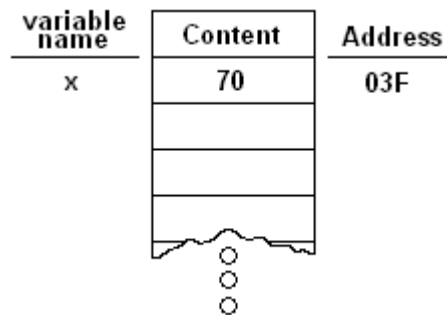
Output:

```
hallow
students
```

3 Variables Declaration:

A declaration is a process of naming the variables and their statements datatypes in C++. C++ allows declaration of the variables before and after executable statements. A variable is an object that may take on values of the specified type.

Also, a variable is a location in the computer's memory where a value can be stored for later use by the program. Variables are like buckets that hold data. These data buckets are really locations in the computer's memory.



The variable must be declared by specifying the datatype and the identifier.

datatype id.1, id2, ..., idn;

A variable is defined by stating its type, followed by one or more spaces, followed by the one or more variable names separated by commas, then followed by a semicolon. For example:

```
unsigned short int X;  
float Y;  
char A, a, c;
```

Note: C++ does distinguish between uppercase *A* and lowercase *a* variables (C++ is case-sensitive).

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.

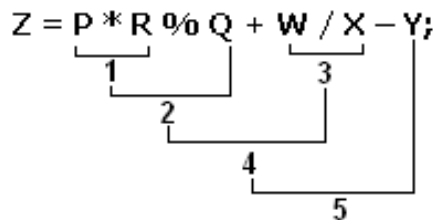
Example 2:

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:

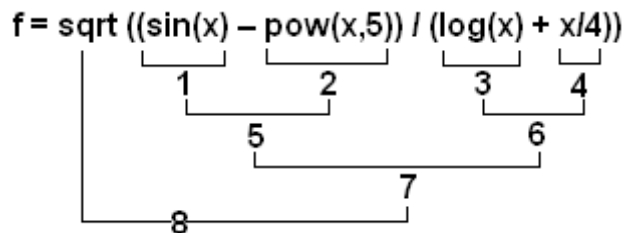
Write the following equation as a C++ expression and state the order of evaluation of the binary operators:

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

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:

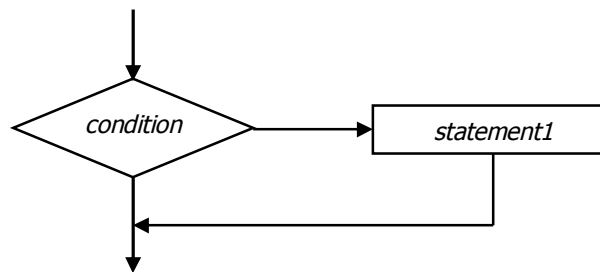
Conditional expressions are mainly used for decision making. C++ provides multiple selection structures: **if**, **if/else**, **else if**, **nested if** and **switch**.

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;
    }
}
```

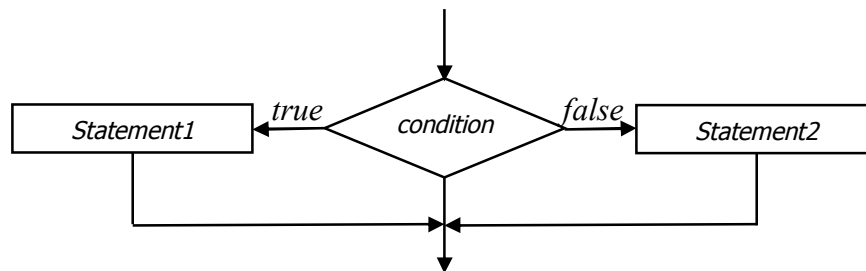
General Form of If/else statement:

```
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:

If (exp.) { Statements } Else { Statements }	If (exp.) { If (exp.) {Statements} Else { Statements } } Else {Statements}	If (exp.) { If (exp.) {Statements} Else { Statements } } Else {If (exp) {Statements} Else {Statement} }
---	---	---

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:  coucout << "grade is B";
            break;
    case 2:  coucout << "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 $r * r * \pi$,

π 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:

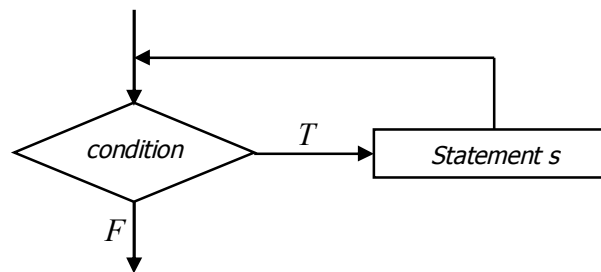
The loop statements are essential to construct systematic block styled programming. C++ provides three iteration structures: **while**, **do/while**, and **for**.

2. While Repetition Structure:

General Form of While statement:

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

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



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

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

Example `i = 1;`
3: `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:

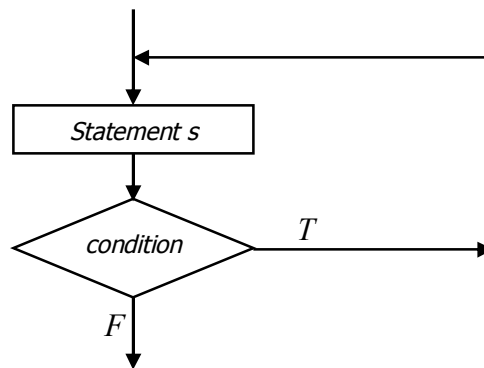
```
#include<iostream.h>
void main( )
{
    Char ch;
    Cout<<"enter a line \n";
    Ch=cin.get();
    While (ch!='\n' && ch!='\t')
    { cout.put(ch);
      Ch=cin.get(); }
}
```

3. Do / While Statement:

General Form of Do / While statement:

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

```
do
{
    statement1 ;
    statement2 ;
    :
    statement-n ;
}
while ( condition );
```



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:

even numbers only

0 2 4 6 8
i=10

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 ++ )    ←————→ for ( int i = n; i > 2; i -- )
        f = f * i;
    cout << "factorial is: " << f;
}
```

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:

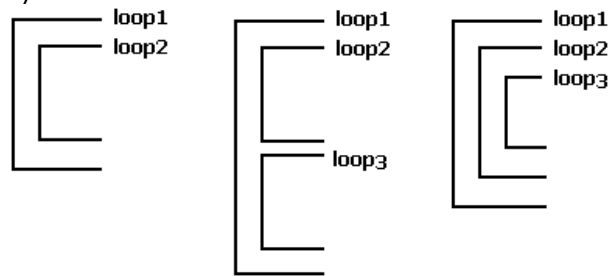
```
for ( int m = 1, int n = 8 ; m < n ; m ++ , n -- )
```

- ☑ We can create infinite loop, as follow:

```
for ( ; ; )
```

3. Nested Loops:

We can put loops one inside another to solve a certain programming problems. Loops may be nested as follows:

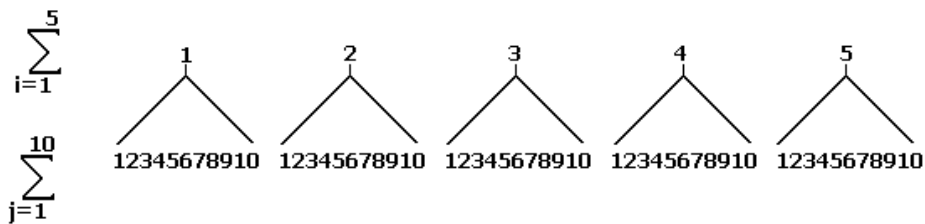


Example 8



Write C++ program to evaluate the following series:

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



```
#include<iostream.h>
```

```
void 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;
```

```
}
```

Example 9



Write C++ program to print the following figure:

```

+
+ +
+ + +
+ + + +
+ + + + +
+ + + + + +
+ + + + + + +
+ + + + + + +
+ + + + + + +
+ + + + + + +
```

```
#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";
    }
}
```

LECTURE 10

1. Breaking Control Statements:

For effective handling of the loop statements, C++ allows the use of the following types of control break statements:

(a) Break Control Statement:

The break statement is used to terminate the control from the loop statements of the case-switch structure. The break statement is normally used in the switch-case loop and in each case condition; the break statement must be used. If not, the control will be transferred to the subsequent case condition also. The general format of the break statement is : **(Break;)**

Example 1:

```
for ( i = 1; i < 100; i ++ )
{
    cout << i;
    if ( i == 10 ) break;
}
```

Output:

1 2 3 4 5 6 7 8 9 10

Example 2:

```
for ( i = 1; i < 10; ++ i )
    for ( j = 1; j < 20; ++ j )
    {
        cout << i * j << endl;
        if ( j == 10 ) break;
    }
```

Example 3:

```
Switch(day) {
Case '1':cout<<"Monday\n";
        Break;
Case '2': .....
}
```

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 < 1);</pre>	<pre>if (n < 1) continue; cout << 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.

<pre>for(i=1 ; i<=100 ; i++) s = s + i;</pre>	<pre>i = 1; while (i <= 100) { s = s + i; i++; }</pre>	<pre>i = 1; do { s = s + i; i++; } while (i <= 100);</pre>
--	--	---

Q2: Find the factorial of n.

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

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

<pre>for(i=1 ; i<=20 ; i++) s = s + (i*i);</pre>	<pre>i = 1; while (i <= 20) { s = s + (i*i); i++; }</pre>	<pre>i = 1; do { s = s + (i*i); i++; } while (i <= 20);</pre>
---	---	---

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

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

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 (≥ 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:

```

      1
    3 3 3
  5 5 5 5 5
7 7 7 7 7 7 7
9 9 9 9 9 9 9 9
  7 7 7 7 7 7 7
    5 5 5 5 5
      3 3 3
        1

```

```

+ + + + + + + + +
+ + + + + + + +
+ + + + + + +
+ + + + + +
+ + + + +
+ + + +
+ + +
+ +
+

```

- 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 searies:

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

# LECTURE 11

## 1. Functions:

A function is a set of statements designed to accomplish a particular task. Experience has shown that the best way to develop and maintain a large program is to construct it from smaller pieces or (modules). Modules in C++ are called functions.

Functions are very useful to read, write, debug and modify complex programs. They can also be easily incorporated in the main program. In C++, the `main()` itself is a function that means the main function is invoking the other functions to perform various tasks. The main advantages of using a function are:

- ❖ Easy to write a correct small function.
- ❖ Easy to read, write, and debug a function.
- ❖ Easier to maintain or modify such a function.
- ❖ Small functions tend to be self documenting and highly readable.
- ❖ It can be called any number of times in any place with different parameters.

## 2. Defining a Function

A function definition has a name, parentheses pair containing zero or more parameters and a body. For each parameter, there should be a corresponding declaration that occurs before the body. Any parameter not declared is taken to be an integer by default. The general format of the function definition is :

### General Form of Function:

```
return-type function-name (parameters-list)
{
 (body of function)
 statement1 ;
 statement2 ;
 :
 statement-n ;
 (return something)
}
```

The type of the function may be int, float, char, etc. It may be declared as type (void), which informs the compiler not to the calling program. For example:

**Void function\_name (---)**

**Int function\_name (---)**

Any variable declared in the body of a function is said to be local to that function. Other variables which are not declared either as arguments or in the function body are considered “global” to the function and must be defined externally. For example

**Void square (int a, int b) → a,b are the formal arguments.**

**Float output (void) → function without formal arguments**

#### Example 1:

```
void printmessage ()
{
 cout << "University of Technology";
}

void main ()
{
 printmessage();
}
```

#### Example 2:

```
int max (int a, int b)
{
 int c;
 if (a > b) c = a;
 else c = b;
 return (c);
}

void main ()
{
 cout << max (5, 6);
}
```

### 3. Return Statement:

The keyword return is used to terminate function and return a value to its caller. The return statement may also be used to exit a function without returning a value. The return statement may or may not include an expression. Its general syntax is:

**Return;**

**Return (expression);**

The return statements terminate the execution of the function and pass the control back to the calling environment.

#### **Example 1**



Write C++ program to calculate the squared value of a number passed from main function. Use this function in a program to calculate the squares of numbers from 1 to 10:

```
#include<iostream.h>

int square (int y)
{
 int z;
 z = y * y;
 return (z);
}

void main()
{
 int x;
 for (x=1; x <= 10; x++)
 cout << square (x) << endl;
}
```

### Example 2



Write C++ program using function to calculate the average of two numbers entered by the user in the main program:

```
#include<iostream.h>

float aver (int x1, int x2)
{
 float z;
 z = (x1 + x2) / 2.0;
 return (z);
}

void main()
{
 float x;
 int num1,num2;
 cout << "Enter 2 positive number \n";
 cin >> num1 >> num2;
 x = aver (num1, num2);
 cout << x;
}
```

### Example 3



Write C++ program, using function, 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>
int summation (int x)
{
 int i = 1, sum = 0;
 while (i <= x)
 {
 sum += i * i;
 i++;
 }
 return (sum);
}

void main ()
```

```

{
 int n ,s;
 cout << "enter positive number";
 cin >> n;
 s = summation (n);
 cout << "sum is: " << s << endl;
}

```

#### Example 4

 Write a function to find the largest integer among three integers entered by the user in the main function.

```

#include <iostream.h>
int max(int y1, int y2, int y3)
{
 int big;
 big=y1;
 if (y2>big) big=y2;
 if (y3>big) big=y3;
 return (big);
}

void main()
{
 int largest,x1,x2,x3;
 cout<<"Enter 3 integer numbers:";
 cin>>x1>>x2>>x3;
 largest=max(x1,x2,x3);
 cout<<largest;
}

```

#### Example 5

 Write program in C++, using function, presentation for logic gates (AND, OR ,NAND, X-OR,NOT) by in A,B enter from user.

```

#include<iostream.h>
void ANDF(int,int);
void ORF(int,int);
void XORF(int,int);
void NOTF(int);

void main()
{ char s;int a,b;
 cout<<"Enter the value A,B :";
 cin>>a>>b;

```

```

cout<<"Enter the select value \n";
cout<<"\ta--(AND gate)\n\to--(OR gate)\n\tx--(X-OR)\n\tn--(NOT
gate)\n\te--(<<EXIT>> :";
cin>>s;
switch(s)
{
 case 'a':ANDF(a,b);break;
 case 'o':ORF(a,b);break;
 case 'x':XORF(a,b);break;
 case 'n':NOTF(a);cout<<" ";NOTF(b);break;
 case 'e':break;
 default:cout<<"bad choose";
}
}
void ANDF(int a,int b)
{
 cout<<(a&b);
}
void ORF(int a,int b)
{
 cout<<(a | b);
}
void XORF(int a,int b)
{
 cout<<(a^b);
}
void NOTF(int a)
{
 cout<<(!a);
}

```

#### 4. Passing Parameters:

There are two main methods for passing parameters to a program:

1) **passing by value**, and 2) **passing by reference**.

##### A- Passing by value:

When parameters are passed by value, a copy of the parameters value is taken from the calling function and passed to the called function. The original variables inside the calling function, regardless of changes made by

the function to it are parameters will not change. All the pervious examples used this method.

## B- Passing by Reference:

When parameters are passed by reference their addresses are copied to the corresponding arguments in the called function, instead of copying their values. Thus pointers are usually used in function arguments list to receive passed references.

This method is more efficient and provides higher execution speed than the call by value method, but call by value is more direct and easy to use.

### Example 6:



The following program illustrates passing parameter by reference.

```
#include <iostream.h>
void swap(int *a,int *b)
{
 int t;
 t=*a;
 *a=*b;
 *b=t;
}
void main()
{
 int x=10;
 int y=15;
 cout<<"x before swapping is:"<<x<<"\n";
 cout<<"y before swapping is:"<<y<<"\n";
 swap(&x,&y);
 cout<<"x after swapping is:"<<x<<"\n";
 cout<<"y after swapping is:"<<y<<"\n";
}
```

# LECTURE 12

## 1. Types of Functions:

The user defined functions may be classified in the following three ways based on the formal arguments passed and the usage of the return statement, and based on that, there are three of user defined functions:

1. A function is invoked without passing any formal argument from the calling portion of a program and also the function does not return back any value to the called function.
2. A function is invoked with formal arguments from the calling portion of a program but the function does not return back any value to the calling portion.
3. A function is invoked with formal arguments from the calling portion of a program which return back a value to the calling environment.

Here are two programs that find the square of the number using and not using a return statement.

### Example 1:

|                                                                                                                                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                          |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre># include &lt;iostream.h&gt; Void main() { Void square (int); Int max; Cout&lt;&lt;"Enter a value for n ?\n"; Cin&gt;&gt;max; For (int i=0;i&lt;=max-1;++i) Square (i) } Void square(int n) { Float value; Value=n*n; Cout&lt;&lt;"i="&lt;&lt;n&lt;&lt;"square="&lt;&lt;value&lt;&lt;endl; }</pre> | <pre># include &lt;iostream.h&gt; Void main() { float square (float); float l,max,value; max=1.5; i=-1.5; while (i&lt;=max) { value=square(i); Cout&lt;&lt;"i="&lt;&lt;i&lt;&lt;"square="&lt;&lt;value&lt;&lt;endl; i=i+0.5; } } Float square(float n) { Float value; Value=n*n; Return (value); }</pre> |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Example 2:



write C++ program, using function to find the summation of the given series:  $\text{Sum} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \dots \frac{x^n}{n!}$

```
#include <iostream.h>
Void main(void)
{
Long int fact (int);
Float power(float,int);
Float sum,temp,x,pow;
Int sign,i,n;
Longint factorial;

Cout<<"Enter a value for n?"<<endl;
Cin>>n;
Cout<<"Enter a value for x ?"<<endl;
Cin>>x;
I=3; sum=x; sign=1;
While (i<=n) {
Factval=fact(i);
Pow=power(x,i);
Sign=(-1)*sign;
Temp=sign*pow/factval;
Sum=sum+temp;
I=i+2;
}
Cout<<"sum of series ="<<sum;
}

Long int fact (int max)
{
Long intvalue;
Value=1;
For(int i=1;i<=max;++i)
Value=value*i;
Return (value);
}

Float power (float x, int n)
{
Float value2;
Value2=1;
For(int j=1;j<=n;++j)
Value2=value2*x;
Return(value2);
}
```

## 2. Actual and Formal Arguments:

The arguments may be classified under two groups, actual and formal arguments:

**(a) Actual arguments:** An actual argument is a variable or an expression contained in a function call that replaces the formal parameter which is a part of the function declaration. Sometimes, a function may be called by a portion of a program with some parameters and these parameters are known as the actual arguments.

**(b) Formal arguments:** are the parameters present in a function definition which may also be called as dummy arguments or the parametric variables. When the function is invoked, the formal parameters are replaced by the actual parameters. Formal arguments may be declared by the same name or by different names in calling a portion of the program or in a called function but the data types should be the same in both blocks

**For example:**

```
include <iostream.h>
Void main()
{
 Int x,y;
 Void output (int x, int y); // function declaration
 —
 —
 Output (x,y); // x and y are actual arguments
}
Void output (int a, int b) // formal or dummy arguments
{
 // body of function
}
```

## 3. Local and Global variables:

The variables in general may be classified as local or global variables.

**(a) Local variables:** Identifiers, variables and functions in a block are said to belong to a particular block or function and these identifiers are known

as the local parameters or variables. Local variables are defined inside a function block or a compound statement. For example,

```
Void func (int l, int j)
{
 Int k,m; // local variables
 // body of the function
}
```

Local variables are referred only to the particular part of a block or a function. Same variable name may be given to different parts of a function or a block and each variable will be treated as a different entity.

**(b) Global variables:** these are variables defined outside the main function block. These variables are referred by the same data type and by the same name throughout the program in both the calling portion of the program and in the function block.

### Example 3:



A program to find the sum of the given two numbers using the global variables.

```
#include <iostream.h>
Int x;
Int y=5;

void main()
{
 X=10;
 Void sum(void);
 Sum();
}

Void sum(void)
{
 Int sum;
 Sum=x+y;
 Cout<<"x="<<x<<endl;
 Cout<<"y="<<y<<endl;
 Cout<<"sum="<<sum<<endl;
}
```

#### 4. Recursive Functions:

A function which calls itself directly or indirectly again and again is known as the recursive function. Recursive functions are very useful while constructing the data structures like linked lists, double linked lists, and trees. There is a distinct difference between normal and recursive functions. A normal function will be invoked by the main function whenever the function name is used, where as the recursive function will be invoked by itself directly or indirectly as long as the given condition is satisfied. Forexample,

```
#include <iostream.h>
Void main(void)
{
Void func1(); //function declaration

Func1(); //function calling
}
Void func1() //function definition
{

Func1(); //function calls recursively
}
```

#### **Example 4:**



A program to find the sum of the given non negative integer numbers using a recursive function  $\text{sum} = 1 + 2 + 3 + 4 + \dots + n$

```
#include <iostream.h>
Void main(void)
{
Int sum(int);
Int n,temp;
Cout<<"Enter any integer number"<<endl;
Cin>>n;
Temp=sum(n);
Cout<<"value="<<n<<"and its sum="<<temp;
}
Int sum(int n) //recursive function
{
Int sum(int); //local function declaration
```

```

Int value=0;
If (n==0)
Return (value);
Else
Value=n+sum(n-1);
Return (value);
}

```

### ***The output of the above program:***

Enter any integer number

Value = 11 and its sum=66

The following illustrations will be helpful to understand the recursive function

|                                                              |                                                                              |                                                                                                        |
|--------------------------------------------------------------|------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------|
| <b>For value 1</b><br>$=1+\text{sum}(1-1)$<br>$=1+0$<br>$=1$ | <b>For value 2</b><br>$=2+\text{sum}(2-1)$<br>$=2+1+\text{sum}(1-1)$<br>$=3$ | <b>For value 3</b><br>$=3+\text{sum}(3-1)$<br>$=3+\text{sum}(2-1)$<br>$=3+2+1+\text{sum}(1-1)$<br>$=6$ |
|--------------------------------------------------------------|------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------|

### **Example 5:**

 A program to find the factorial (n!) of the given number using the recursive function. Its the product of all integers from 1 to n (n is non negative) (so  $n!=1$  if  $n=0$  and  $n!=n(n-1)$  if  $n>0$ )

```

#include <iostream.h>
Void main(void)
{
Long int fact (long int);
Int x,n;
Cout<<"Enter any integer number"<<endl;
Cin>>n;
X=fact(n);
Cout<<"value="<<n<<"and its factorial=";
Cout<<x<<endl;
}
Long int fact (long int n) //recursive function
{
Long int fact(long int); //local function declaration
Int value =1;
If (n==1)
Return(value)
Else
{value=n*fact(n-1);
Return(value);
} }

```

***The output of the above program:***

Enter any integer number

Value = 5 and its factorial=120

The following illustrations will be helpful to understand the recursive function

|                                                        |                                                                       |                                                                                                 |
|--------------------------------------------------------|-----------------------------------------------------------------------|-------------------------------------------------------------------------------------------------|
| <b>For value 1</b><br><b>=1*fact(1-1)</b><br><b>=1</b> | <b>For value 2</b><br><b>=2*fact(2-1)</b><br><b>=2*1</b><br><b>=2</b> | <b>For value 3</b><br><b>=3*fact(3-1)</b><br><b>=3*2*fact(2-1)</b><br><b>3*2*1</b><br><b>=6</b> |
|--------------------------------------------------------|-----------------------------------------------------------------------|-------------------------------------------------------------------------------------------------|

# WORK SHEET (5)

## Functions

Q1: Write a C++ program, using function, to counts uppercase letter in a 20 letters entered by the user in the main program.

Q2: Write a C++ program, using function, that reads two integers (feet and inches) representing distance, then converts this distance to meter.

Note: 1 foot = 12 inch

1 inch = 2.54 Cm

i.e.:

Input: feet: 8 or inches: 9

Q3: Write a C++ program, using function, which reads an integer value (T) representing time in seconds, and converts it to equivalent hours (hr), minutes (mn), and seconds (sec), in the following form:

**hr : mn : sec**

i.e.:

Input: 4000

Output: 1 : 6 : 40

Q4: Write a C++ program, using function, to see if a number is an integer (odd or even) or not an integer.

Q5: Write a C++ program, using function, to represent the permutation of n.

Q6: Write a C++ program, using function, to inputs a student's average and returns 4 if student's average is 90-100, 3 if the average is 80-89, 2 if the average is 70-79, 1 if the average is 60-69, and 0 if the average is lower than 60.

Q7: The Fibonacci Series is: 0, 1, 1, 2, 3, 5, 8, 13, 21, ... It begins with the terms 0 and 1 and has the property that each succeeding term is the sum of the two preceding terms. Write a C++ program, using function, to calculate the nth Fibonacci number.

Q8: Write a C++ program, using function, to calculate the factorial of an integer entered by the user at the main program.

Q9: Write a C++ program, using function, to evaluate the following equation:

$$z = \frac{x! - y!}{(x - y)!}$$

Q10: Write a C++ program, using function, to test the year if it's a leap or not.

**Note:** use `y % 4 == 0 && y % 100 != 0 :: y % 400 == 0`

Q11: Write a C++ program, using function, to find  $X^Y$ .

**Note:** use **pow** instruction with **math.h** library.

Q12: Write C++ program, using function, to inverse an integer number:

For example: 765432 → 234567

Q13: Write C++ program, using function, to find the summation of student's marks, and it's average, assume the student have 8 marks.

Q14: Write C++ program, using function, to convert any char. From capital to small or from small to capital.

Q15: Write C++ program using recursive function to find the power of n numbers.

# LECTURE 13

## 1. Arrays:

An array is a consecutive group of homogeneous memory locations. Each element (location) can be referred to using the array name along with an integer that denotes the relative position of that element within the array. The data items grouped in an array can be simple types like int or float, or can be user-defined types like structures and objects.

## 2. Array of One Dimension:

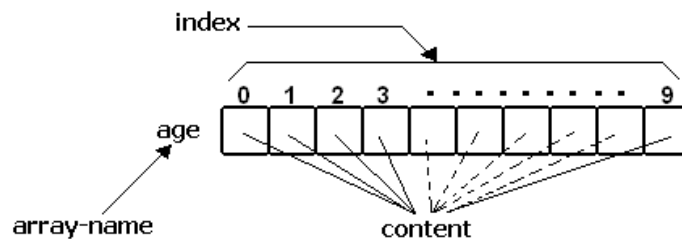
It is a single variable specifies each array element. The declaration of one dimensional arrays is:

### **General Form of 1D-Array:**

*data-type* Array-name [ *size* ];

Examples:

```
int age [10];
int num [30];
float degree[5];
char a [15];
```



The item in an array are called elements (in contrast to the items in a structure which are called members). The elements in an array are of the same type only the values vary.

### 3. Initializing Array Elements:

- The first element of array age:  
age [0] = 18;
- The last element of array age:  
age [9] = 19;
- All elements of array age:  
age [9] = { 18, 17, 18 ,18 ,19, 20 ,17, 18 ,19 };
- int x [ ] = { 12, 3, 5, 0, 11, 7, 30, 100, 22 };
- int y [10] = { 8, 10, 13, 15, 0, 1, 17, 22};

### 4. Accessing Array Elements:

We access each array element by written name of array, followed by brackets delimiting a variable (or constant) in the brackets which are called the array index.

- Accessing the first element of array num to variable x:  
x = num [0];
- Accessing the last element of array num to variable y:  
y = num [9];
- cout << num [0] + num [9];
- num [0] = num [1] + num[2];
- num [7] = num [7] + 3;                      ↔ num [7] += 3;

### 5. Read / Write / Process Array Elements:

- |                                                |                                                               |
|------------------------------------------------|---------------------------------------------------------------|
| - cout << num [4];                             | - for (int i=0; i<10; i++)<br>cout << num[ i ];               |
| - if ( num [5] > 5 )<br>cout << "greater";     | - for (int i=9; i>=0; i++)<br>cout << num[ i ];               |
| - for (int i=0; i<10; i++)<br>cin >> num[ i ]; | - sum=0;<br>for (int i=0; i<10; i++)<br>sum = sum + num[ i ]; |

### Example 1



Write C++ program to display 2<sup>nd</sup> and 5<sup>th</sup> elements of array distance:

```
#include<iostream.h>

void main()
{
 double distance[] = { 23.14, 70.52, 104.08, 468.78, 6.28};
 cout << "2nd element is: " << distance[1] << endl;
 cout << "5th element is: " << distance[4];
}
```

### Example 2



Write C++ program to read 5 numbers and print it in reverse order:

```
#include<iostream.h>

void main()
{
 int a [5];
 cout << "Enter 5 numbers \n";
 for (int i =0; i <5; i++)
 {
 cout << i << ": ";
 cin >> a [i];
 cout << "\n";
 }
 cout << "The reverse order is: \n";
 for (i =4; i >=0; i--)
 cout << i << ": " << a [i] << endl;
}
```

### Example 3



Write C++ program, to find the summation of array elements:

```
#include<iostream.h>

void main ()
{
 int const L = 10;
 int a [L];
 int sum = 0;
 cout << "enter 10 numbers \n";
 for (int i =0; i <L; i++)
 {
 cout << "enter value " << i << ": ";
 cin >> a [i];
 }
}
```

```

 sum += a [i];
 }
 cout << "sum is: " << sum << endl;
}

```

#### Example 4

 Write C++ program, to find the minimum value in array of 8 numbers:

```

#include<iostream.h>

void main ()
{
 int n = 8; int a [] = { 18, 25, 36, 44, 12, 60, 75, 89 };
 int min = a [0];
 for (int i = 0; i < n; i++)
 if (a [i] < min) min = a [i];
 cout << "The minimum number in array is: " << min; }

```

#### Example 5

 Write C++ program, to give the number of days in each month:

```

#include<iostream.h>
void main ()
{
 Int month, day, total_days;
 Int days_per_month[12]={31,28,31,30,31,30,31,31,30,31,30,31}
 Cout<<"\n Enter month(1 to 12):";
 Cin>>month;
 Cout<<"enter day(1 to 31):";
 Cin>>day;
 Total_days=day;
 For (int j=0;j<month-1;j++)
 Total_day+=day_per_month[j];
 Cout<<"Total days from start of year is:"<<total_days;
}

```

#### Example 6

 Write C++ program, using function, to find (search) X value in array, and return the index of it's location:

```

#include<iostream.h>

int search(int a[], int y)
{
 int i= 0;
 while (a [i] != y)
 i++;
}

```

```
 return (i);
}

void main ()
{
 int X, f;
 int a [10] = { 18, 25, 36, 44, 12, 60, 75, 89, 10, 50 };
 cout << "enter value to find it: ";
 cin >> X;
 f= search (a, X);
 cout << "the value " << X << " is found in location " << f;
}
```

## Example 7



Write C++ program, to split the odd numbers and even numbers of one array into two arrays:

$a = [1, 2, 3, 4, 5, 6, 7, 8, \dots, 20]$   
 $a_{\text{odd}} = [1, 3, 5, 7, \dots, 19]$   
 $a_{\text{even}} = [2, 4, 6, 8, \dots, 20]$

```
#include<iostream.h>
```

```
void main ()
```

```
{
 int a [20]= { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20 };
 int aodd[20], aeven [20];
 int i ,o=0, e=0;
 for (i=0 ; i<20; i++)
 if (a[i] % 2 !=0)
 {
 aodd[o]=a[i];
 o=o+1;
 }
 else
 {
 aeven[e]=a[i];
 e=e+1;
 }

 for (i=0 ; i<o; i++)
 cout<<aodd[i]<<" ";

 cout<<endl;

 for (i=0 ; i<e; i++)
 cout<<aeven[i]<<" ";
}
```

# LECTURE 14

## 1. Array of Two Dimension:

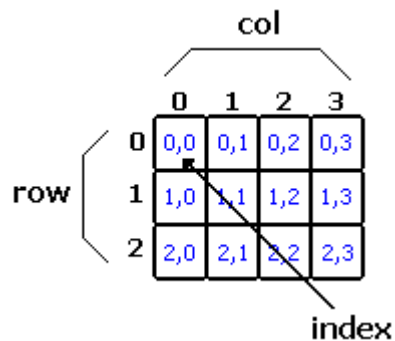
Arrays can have higher dimension. There can be arrays of two dimension which is array of arrays. It is accessed with two index. Also there can be arrays of dimension higher than two.

### General Form of 2D-Array:

```
data-type Array-name [Row-size] [Col-size];
```

Examples:

```
int a [10] [10];
int num [3] [4];
```



## 2. Initializing 2D-Array Elements:

- The first element of array age:

```
a [2] [3] = { {1, 2, 3} , {4, 5, 6} };
```

|   |   |   |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |

### 3. Read / Write / Process Array Elements

#### Example 1



Write C++ program, to read 15 numbers, 5 numbers per row, the print them:

```
#include<iostream.h>

void main ()
{
 int a [3] [5];
 int i , j;
 for (i = 0 ; i < 3; i++)
 for (j = 0 ; j < 5; j++)
 cin >> a [i] [j];

 for (i = 0 ; i < 3; i++)
 {
 for (j = 0 ; j < 5; j++)
 cout << a [i] [j];
 cout << endl;
 }
}
```

#### Example 2



Write C++ program, to read 4\*4 2D-array, then find the summation of the array elements, finally print these elements:

```
#include<iostream.h>

void main ()
{
 int a [4] [4];
 int i , j, sum = 0;
 for (i = 0 ; i < 4; i++)
 for (j = 0 ; j < 4; j++)
 cin >> a [i] [j];

 for (i = 0 ; i < 4; i++)
 for (j = 0 ; j < 4; j++)
 sum += a [i] [j];
 cout << "summation is: " << sum << endl;

 for (i = 0 ; i < 4; i++)
 {
 for (j = 0 ; j < 4; j++)
 cout << a [i] [j];
 cout << endl;
 }
}
```

### Example 3

 Write C++ program, to read 3\*4 2D-array, then find the summation of each row:

```
#include<iostream.h>
```

```
void main ()
```

```
{
 int a [3] [4];
 int i , j, sum = 0;
 for (i = 0 ; i < 3; i++)
 for (j = 0 ; j < 4; j++)
 cin >> a [i] [j];

 for (i = 0 ; i < 3; i++)
 {
 sum = 0;
 for (j = 0 ; j < 4; j++)
 sum += a [i] [j];
 cout << "summation of row " << i << " is: " << sum << endl;
 }
}
```

### Example 4

 Write C++ program, to read 3\*4 2D-array, then replace each value equal 5 with 0:

```
#include<iostream.h>
```

```
void main ()
```

```
{
 int a [3] [4];
 int i , j;
 for (i = 0 ; i < 3; i++)
 for (j = 0 ; j < 4; j++)
 cin >> a [i] [j];

 for (i = 0 ; i < 3; i++)
 for (j = 0 ; j < 4; j++)
 if (a [i] [j] == 5) a [i] [j] = 0;

 for (i = 0 ; i < 3; i++)
 {
 for (j = 0 ; j < 4; j++)
 cout << a [i] [j];
 cout << endl;
 }
}
```

### Example 5

 Write C++ program, to addition two 3\*4 arrays:

```
#include<iostream.h>

void main ()
{
 int a [3] [4], b [3] [4], c [3] [4];
 int i , j;
 cout << "enter element of array A: \n";
 for (i = 0 ; i < 3; i++)
 for (j = 0 ; j < 4; j++)
 cin >> a [i] [j];
 cout << "enter element of array B: \n";
 for (i = 0 ; i < 3; i++)
 for (j = 0 ; j < 4; j++)
 cin >> b [i] [j];
 for (i = 0 ; i < 3; i++)
 for (j = 0 ; j < 4; j++)
 c [i] [j] = a [i] [j] + b [i] [j];
 for (i = 0 ; i < 3; i++)
 {
 for (j = 0 ; j < 4; j++)
 cout << c [i] [j];
 cout << endl;
 }
}
```

### Example 6

 Write C++ program, to convert 2D-array into 1D-array:

```
#include<iostream.h>

void main ()
{
 int a [3] [4];
 int b [12];
 int i , j, k = 0;

 for (i = 0 ; i < 3; i++)
 for (j = 0 ; j < 4; j++)
 cin >> a [i] [j];

 for (i = 0 ; i < 3; i++)
 for (j = 0 ; j < 4; j++)
 {
 b [k] = a [i] [j];
 }
}
```

```

 k++;
 }
 for (i = 0 ; i < k; i++)
 cout << b [i];
}

```

### Example 7

 Write C++ program, to replace each element in the main diameter (diagonal) with zero:

```
#include<iostream.h>
```

```
void main ()
```

```

{
 int a [3] [3];
 int i , j;

 for (i = 0 ; i < 3; i++)
 for (j = 0 ; j < 3; j++)
 cin >> a [i] [j];

 for (i = 0 ; i < 3; i++)
 for (j = 0 ; j < 3; j++)
 if (i == j) a [i] [j] = 0;

 for (i = 0 ; i < 3; i++)
 {
 for (j = 0 ; j < 3; j++)
 cout << a [i] [j];
 cout << endl;
 }
}

```

|     |     |     |
|-----|-----|-----|
| 0,0 |     |     |
|     | 1,1 |     |
|     |     | 2,2 |

i = j

|     |     |     |
|-----|-----|-----|
| 0,0 | 0,1 | 0,2 |
| 1,0 | 1,1 | 1,2 |
| 2,0 | 2,1 | 2,2 |

i = j

|     |     |     |
|-----|-----|-----|
| 0,0 | 0,1 | 0,2 |
| 1,0 | 1,1 | 1,2 |
| 2,0 | 2,1 | 2,2 |

i + j = n-1

|     |     |     |
|-----|-----|-----|
| 0,0 | 0,1 | 0,2 |
| 1,0 | 1,1 | 1,2 |
| 2,0 | 2,1 | 2,2 |

i > j

|     |     |     |
|-----|-----|-----|
| 0,0 | 0,1 | 0,2 |
| 1,0 | 1,1 | 1,2 |
| 2,0 | 2,1 | 2,2 |

i < j

### Example 8



Write C++ program, print the square root of an array:

```
#include<iostream.h>

void main ()
{
 int a [3] [3], b [3] [3];
 int i , j;
 for (i = 0 ; i < 3; i++) {
 for (j = 0 ; j < 3; j++) {
 b[i][j]= sqrt(a[i][j]);
 cout << b [i] [j];
 }
 }
}
```

### Example 9



Write C++ program, to read 3\*3 2D-array, then find the summation of the main diagonal and its secondary diagonal of the array elements, finally print these elements:

```
#include<iostream.h>

void main ()
{
 int a [3] [3];
 int i , j, x , y;
 for (i = 0 ; i < 3; i++) {
 for (j = 0 ; j < 3; j++) {
 cin >> a [i] [j];

 if (i == j)
 x=x+a[i][j];
 if (i + j =4)
 y=y+a[i][j];
 }
 }
 cout << "summation of diagonal is: " << x << endl;
 cout << "summation of inverse diagonal is: " << y << endl;
}
```

# WORK SHEET (6)

## Arrays

- Q1: Write a C++ program, using function, to find if the array's elements are in order or not.
- Q2: Write a C++ program, using function, to compute the number of zeros in the array.
- Q3: Write a C++ program, using function, to find the value of array C from add array A and array B.  $C[i] = A[i] + B[i];$
- Q4: Write a C++ program, using function, to multiply the array elements by 2.  $A[i] = A[i] * 2;$
- Q5: Write a C++ program, using function, to reads temperatures over the 30 days and calculate the average of them.
- Q6: Write a C++ program, using function, to merge two arrays in one array.
- Q7: Write C++ program, to read 3\*4 2D-array, then find the summation of each col.
- Q8: Write C++ program, to replace each element in the second diameter (diagonal) with zero.
- Q9: Write C++ program, to replace the elements of the main diameter with the elements of the second diameter.
- Q10: Write C++ program, to find the summation of odd numbers in 2D-array.
- Q11: Write C++ program, to find (search) X value in 2D-array, and return the index of it's location.
- Q12: Write C++ program, to convert 1D-array that size [16] to 2D-array that size of [4] [4].
- Q13: Write C++ program, to read A[ n, n ] of character, then find array B and array C, such that B contain only capital letters and C contain only small letters.

Q14: Write C++ program, to read A[ n, n ] of numbers, then put 10 instead each even positive number.

Q15: Write C++ program, to read A[ n, n ] of numbers, then put 10 instead each even positive number in the first diagonal.

Q16: Write C++ program, to read A[ n, n ] of numbers, then find the minimum number in array.

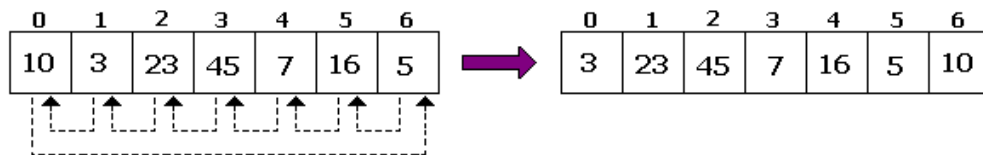
Q17: Write C++ program, to exchange row1 and row3 in 4\*3 array.

Q18: Write C++ program, to exchange row0 with col3 in 4\*4 array.

Q19: Write C++ program, to find the greatest number in the second diagonal, in 3\*3 array.

Q20: Write C++ program, to read X[ n ], and rotate the elements to the left by one position.

i.e:



Q21: Write C++ program, to read A[ n ] and a location Z then delete the number at location Z from the array, and print the new array after deletion.

Q22: Write C++ program to order the array in ascending and descending order.

Q23: Write C++ program to read (n) no.s and find the average of the even no. on it.

Q24: Create the array (b) from (a).

|   |   |   |    |
|---|---|---|----|
| 1 | 2 | 3 | 6  |
| 4 | 5 | 6 | 10 |
| 7 | 8 | 9 | 10 |

Q25: Create the arrays bellow.

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 |
| 2 | 2 | 2 | 2 | 1 | 2 | 1 | 1 |
| 3 | 3 | 3 | 3 | 1 | 1 | 2 | 1 |
| 4 | 4 | 4 | 4 | 1 | 1 | 1 | 2 |

# LECTURE 15

## 1. String:

In C++ strings of characters are implemented as an array of characters. In addition a special null character, represented by `\0`, is appended to the end of string to indicate the end of the string.

### General Form of String:

```
char String-name [size];
```

Examples: `char name [10] = "Mazin Alaa";`

→ `'M', 'a', 'z', 'i', 'n', ' ', ' ', 'A', 'l', 'a', 'a', '\0'`

`char str [ ] = "ABCD";`

→ `'A', 'B', 'C', 'D', '\0'`

```
str [0] : 'A'
str [1] : 'B'
str [2] : 'C'
str [3] : 'D'
str [4] : '\0' ↔ null
```

## 2. Read / Write / Process Array Elements:

### Example 1

 Write C++ program to print string, then print it character by character:

```
#include<iostream.h>
```

```
void main()
{
```

```
 char s [] = "ABCD";
```

```
 cout << "Your String is: " << s << endl;
```

```
 for (int i=0; i < 5; i++)
```

```
 cout << "S[" << i << "] is: " << s [i] << endl;
```

### Output is:

Your String is: ABCD

S[0] is: A

S[1] is: B

S[2] is: C

S[3] is: D

S[4] is:

## Example 2



Write C++ program to convert each lower case letter to upper case letter:

```
#include<iostream.h>
#include<ctype.h>

void main()
{
 char s [] = "abcd";
 cout << s << endl;

 for (int i =0; i < 4; i++)
 s [i] = char(toupper (s[i]));

 cout << s;
}
```

### Note:

There are several ways to read and write (there are several input/output function) like:

```
cin.getline (str, 10);
cin.get (ch);
cin.ignor (80, '\n');
cin.putback (ch);
cout.put (ch);
```

*Apply it ...*

### 3. Member Function of String:

The string library has many member functions of string like:

| Member Function                    | Functionality                                                                                                                                                                                                                                                                                      | Example                                                                                                                  |
|------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------|
| <b>strlen ( string )</b>           | Return the length of the string                                                                                                                                                                                                                                                                    | a [ ] = "abcd";<br>cout << strlen ( a );                                                                                 |
| <b>strcpy ( string2, string1 )</b> | Copy the content of the 1 <sup>st</sup> string into the 2 <sup>st</sup> string                                                                                                                                                                                                                     | char a[ ] = "abcd" , b[ ] = " ";<br>strcpy ( b , a );<br>cout << a << b;                                                 |
| <b>strcat ( string1, string2 )</b> | Append the content of the 2 <sup>nd</sup> string into the end of the 1 <sup>st</sup> string                                                                                                                                                                                                        | char a[ ] = "abcd" , b[ ] = "1234";<br>strcat ( a , b );<br>cout << a << b;<br>abcd1234 1234                             |
| <b>strcmp ( string1, string2 )</b> | Return 0 if the 1 <sup>st</sup> string is equal to the 2 <sup>nd</sup> string.<br><br>Return a Positive number if the 1 <sup>st</sup> string is greater than the 2 <sup>nd</sup> string.<br><br>Return a Negative number if the 1 <sup>st</sup> string is smaller than the 2 <sup>nd</sup> string. | char a[ ] = "abcd" , b[ ] = "abcd";<br>cout << strcmp ( a , b );<br><br>0    if a == b<br>+    if a > b<br>-    if a < b |

### 4. stdlib Library:

The stdlib library has many member functions of string like:

| Member Function            | Functionality                                              | Example                                                       |
|----------------------------|------------------------------------------------------------|---------------------------------------------------------------|
| <b>A atoi ( a )</b>        | Converts string to int type.                               | int i; char a [ ] = "1234";<br>i = atoi (a);                  |
| <b>A atof ( a )</b>        | Converts string to float type.                             | float f; char a [ ] = "12.34";<br>f = atof (a);               |
| <b>itoa ( i , a , 10);</b> | Converts integer number to alphabet (char or string type). | int i = 1234; char a [ ] = "";<br>cout << itoa ( i , a , 10); |

# WORK SHEET (7)

## String

Q1: Write C++ program to print a string, and then print it character by character *in reverses order*.

i.e:

abcd → a  
b  
c  
d

Q2: Write C++ program to check each character in the string to convert it to lower case letter if it's an upper case letter and convert it to upper case letter if it's a lower once.

Q3: Write C++ program to read a sentence and print its words separately.

Q4: Write C++ program to apply the following instructions:

- cin.getline ( str, 10 );
- cin.get ( ch );
- cin.ignor ( 80, '\n' );
- cin.putback ( ch );
- cout.put ( ch );

Q5: Write C++ program to apply the following instructions:

- strlen ( string )
- strcpy ( string2, string1 )
- strcat ( string1, string2 )
- strcmp ( string1, string2 )

Q5: Write C++ program to apply the following instructions:

- atoi ( a )
- atof ( a )
- itoa ( i , a , 10);