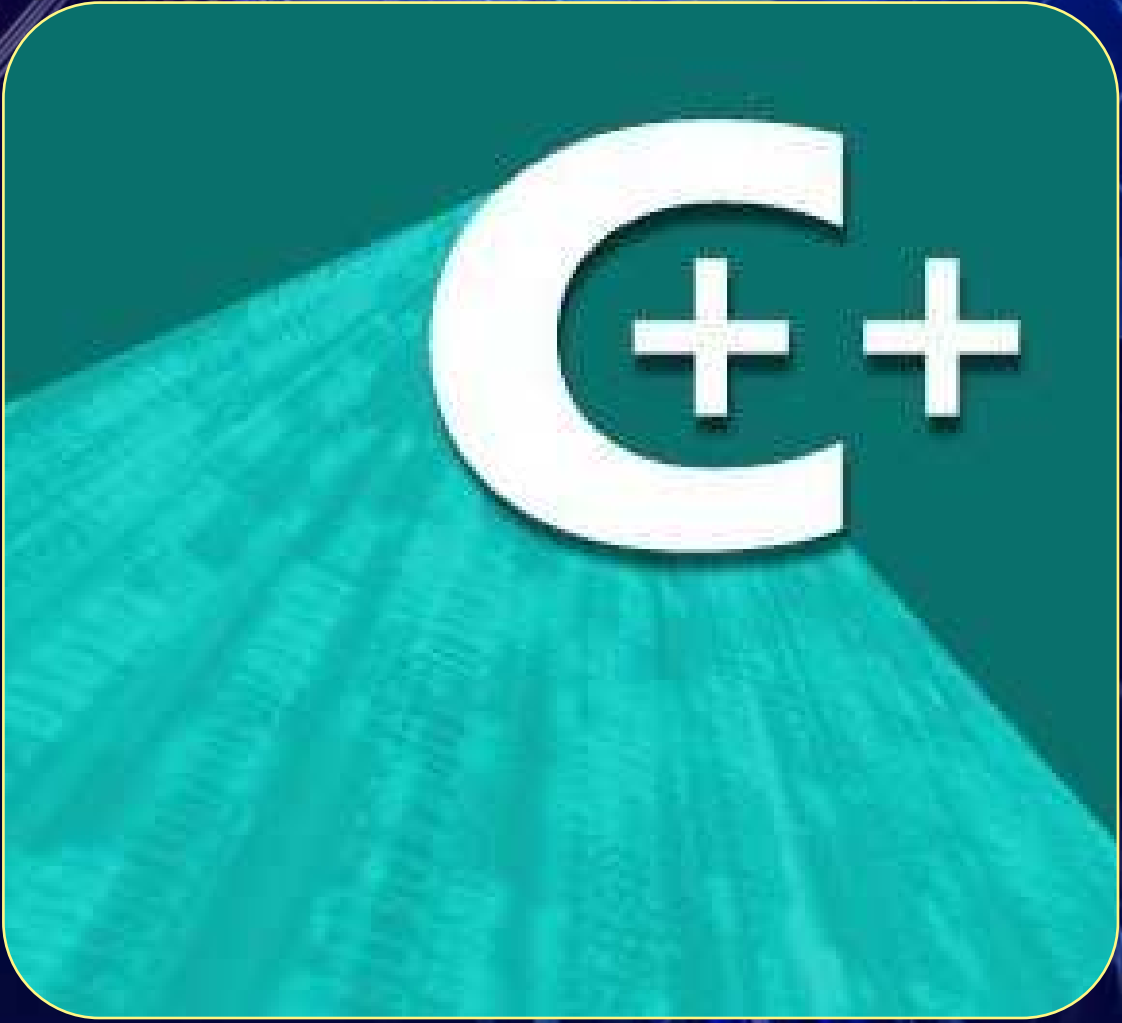


Unit

3

# INPUT/OUTPUT HANDLING IN C++

A large, stylized logo for C++ programming. The letters 'C++' are rendered in a bold, white, sans-serif font with a slight 3D effect. They are centered on a teal-colored rectangular background that has rounded corners and a subtle gradient. The background is set against a dark blue, futuristic digital landscape with glowing lines and dots, suggesting a network or data flow.



- ◆ Explain basic structure of C++ program.
- ◆ Introduce the use of Preprocessor directives in C++ program.
- ◆ Comment Statement in C++

### 3.1 BASIC STRUCTURE OF C++

In C++ program is divided into three parts:

1. Preprocessor Directives
2. Main Function Header
3. Body of Program / Function

The basic structure of the C++ program is given in Figure No. 2.1:

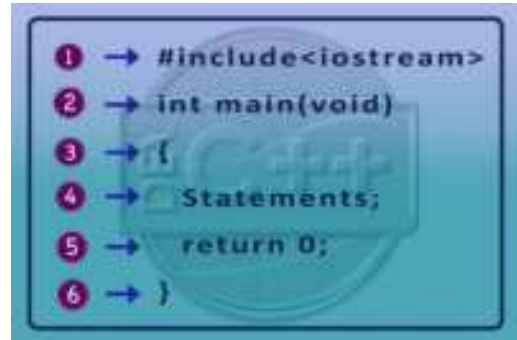


Fig. 2.1 Basic Structure of C++ Program

S.NO.	Codes and Symbols	Description
1.	<code>#include&lt;iostream&gt;</code>	The # symbol is called Preprocessor Directives. <b>#include</b> is used to link the external header files/libraries which may be required in program. <b>#define</b> is used define constants in program.
2.	<code>using namespace std;</code>	This instruction tells the compiler to use standard namespace. The Namespace is the collection of identifiers. It is used for variables, class, functions, and objects. All these elements of the Standard Library of C++ are declared within standard "std".
3.	<code>int main(void)</code>	<b>int main(void)</b> function is used for execution of C++ program. void main(void) nothing to return value.

4.	{	This symbol indicates the beginning of the main function. It is also known as <b>Opening Curly braces</b> .
5.	Statement;	Instructions that perform a particular task is called a statement. Statement terminator (;) used to the end of statements. This symbol also known as semi colon. For example: <code>cout &lt;&lt; "Pakistan Zindabad";</code> The output of the given example is that "Pakistan Zindabad" will print on the screen.
6.	return value;	The return value is the exit code of your program. By default, <code>main( )</code> in C++ returns an <b>int</b> integer data type value to the operating system.
7.	}	This symbol indicates the ending of the main function. It is also known as <b>Closing Curly braces</b> .



The body of the function is enclosed between curly braces. All instructions are executed within opening "{" and closing "}" curly braces.

### 3.2 COMMENT STATEMENT IN C++

The comment statement are those statements that are ignored by the compiler. These statements do not execute. Through comments, the programmers give special remarks to the statements for their convenience. In C++, there are two types of comment statements.

1. Single Line Comment
2. Multi Line Comment

#### 1. Single Line Comment:

It is used to a single-line explanation with the help of a double slash (//) symbol. If the programmer wants to use a single line comment on more

than one line, this may need to put a double slash on each line at the start. These comments are ignored by the compiler, which means comments are not executable.

**For example:**

```
// Single line comment
// This is my first program
#include<iostream>
int main( )
{
    Statements...;
return 0;
}
```

## 2. Multi Line Comment:

It is used for multiple-line explanations. Symbols (`/*` and `*/`) are needed at the start and end of the statements. These comments are ignored by the compiler, which means comments are not executable.

**For example:**

```
/* Multi line comment
   This is my first program */
#include<iostream>
int main( )
{
    Statements...
return 0;
}
```



- ◆ Differentiate between input and output functions.
- ◆ Use input and output functions in a program.
- ◆ Describe the use of statement terminator in a program
- ◆ Use escape sequences in any C++ program.

## 3.3 INPUT/OUTPUT Handling in C++

In C++ Input and Output streams perform Input/ Output (I/O) operation and these I/O stream are stored in header file. Such as `<iostream>`. These header files must be mentioned at the beginning of the program.





I/O streaming uses multiple input/output channels.

### 3.3.1 Output Function

S.no.	Functions	Description with examples
1.	cout statement	<p><b>cout</b> is a predefined object in C++. It is used to display the output to the standard output device i.e. monitor. "cout" uses insertion operator (&lt;&lt;).</p> <p>Syntax: cout &lt;&lt; variable or cout &lt;&lt;exp./string &lt;&lt; variable</p> <p><b>For example:</b>  <b>cout &lt;&lt; "MY FIRST PROGRAM";</b></p>
2.	puts( )	<p>This function used to print the string to the output stream. The new line is automatically inserted after printing the string.</p> <p>Syntax: int puts(const char * str);</p> <p><b>For example:</b></p> <pre>#include&lt;iostream&gt; using namespace std; int main(void) { puts("MY FIRST PROGRAM"); return 0; }</pre> <div style="border: 1px solid black; padding: 5px; display: inline-block; margin-top: 10px;"> <p><b>OUTPUT:</b> MY FIRST PROGRAM</p> </div>



"**cout**" stands for "**Character Output**". Here C means character and Out means output. puts( ) is defined in <stdio> header file. This file must be included at the beginning of the program.

#### Teachers Note



Teacher are supposed to orient students also about the function of **putc( )** in C++ program.

### 3.3.2 Input Function

S.no.	Functions	Description with examples
1.	cin statement	<p><b>cin</b> is a predefined object that reads data from the keyboard with the extraction operator (&gt;&gt;). This operator allows you to accept data from standard input device.</p> <p>Syntax:        cin &gt;&gt; variable;</p> <p><b>For Example:</b></p> <pre>#include&lt;iostream&gt; using namespace std; int main(void) {     int a;     cin &gt;&gt; a;    //cin takes input in "a"     variable     return 0; }</pre>
2.	getch()	<ul style="list-style-type: none"> <li>• <b>getch( )</b> is predefined function. This function is defined in conio.h (Console Input and Output header file).</li> <li>• It is used to get a single character from keyboard during execution of program.</li> <li>• The entered character is not printed on the screen.</li> <li>• It is used to hold the output screen until the user press any key from the keyboard.</li> </ul> <p><b>For Example:</b></p> <pre>#include&lt;iostream&gt; #include&lt;conio.h&gt; using namespace std; int main(void) {     char ch = getch();     cout &lt;&lt;"X Class";     cout &lt;&lt; ch;      return 0; }</pre>

3.	getche()	<ul style="list-style-type: none"> <li>• The function of “getche( )” is similar to getch( ) function.</li> <li>• The “<b>getche( )</b>” stands for get character echo.</li> <li>• This function displays the character that entered by the user.</li> <li>• It is also predefined function in “conio.h” header file.</li> </ul> <p>Syntax: character variable = getche( );</p> <p><b>For example:</b></p> <pre>#include&lt;iostream&gt; #include&lt;conio.h&gt; using namespace std; int main(void) {     char ch;     int a=10,b=10;     cout &lt;&lt;"\n Do you want to continue (Y/N)...";     ch=getche();     cout &lt;&lt; "\n the addition is...." &lt;&lt;a+b;      return 0; }</pre> <div data-bbox="935 602 1247 762" style="border: 1px solid black; padding: 5px;"> <p><b>OUTPUT:</b> Do your want to continue(Y/N)..Y the addition is..... 20</p> </div>
4.	getchar()	<p>The getchar( ) function in C++ reads the character from standard input stream. It's defined in &lt;stdio.h&gt; header file. It needs to press Enter Key after entering the character.</p> <p><b>For example:</b></p> <pre>#include&lt;iostream.h&gt; #include&lt;stdio.h&gt; using namespace std; int main(void) {     char ch;     cout &lt;&lt; "\n Use of getchar</pre> <div data-bbox="925 1330 1243 1501" style="border: 1px solid black; padding: 5px;"> <p><b>OUTPUT:</b> Use of getchar function....a getchar is.....a</p> </div>

		<pre>function"; ch = getchar(); cout &lt;&lt; "\n getchar is" &lt;&lt; ch;  return 0; }</pre>
5.	gets()	<p>This is a predefined function in C++ and it reads characters from stdin and stores them until a newline character found. It's defined in &lt;stdio&gt; header file. Program's code shows how to apply this function in C++.</p> <p>Syntax:      gets(variable);</p> <p><b>For example:</b></p> <pre>#include&lt;iostream&gt; #include&lt;stdio.h&gt; using namespace std; int main(void) {     char ch[20];     cout &lt;&lt; "\n Enter the message..";     gets(ch);     cout &lt;&lt; " Your message is...." &lt;&lt; ch;     return 0; }</pre> <div data-bbox="944 693 1246 899" style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <p><b>OUTPUT:</b></p> <p>Enter the message.... Pakistan Your message is..... Pakistan</p> </div>

**Teachers Note**

It is good if teacher informs students about the function of `getc()` in C++

### 3.3.3 Statement Terminator (;):

Every statement in C++ must be terminated with semi colon (;). It indicates the end of the statement and it is also called Statement terminator. If the terminator is missing an error message will occur.

### 3.3.4 Escape Sequences:

The escape sequences are special non-printing characters. They can be used with the “cout” in C++. An escape sequence starts with a backslash (\) and a code character.

The commonly used escape sequences are given below:

Escape Sequence	Explanation with example
\a	“a” means Alert or alarm. It causes a beep sound in the computer. <b>Example:</b> <code>cout &lt;&lt;“\a”;</code>
\b	“b” stands for backspace. It moves the cursor backspace. <b>Example:</b> <code>cout &lt;&lt; “\b”;</code>
\t	“t” stands for Horizontal tab. It is used to shift the cursor to a couple of spaces to the right in the same line. <b>Example:</b> <code>cout &lt;&lt; “\t”;</code>
\n	“n” stands for New line or line feed. It inserts a new line and cursor moves to the beginning of the next line. <b>Example:</b> <code>cout &lt;&lt; “\n”;</code>
\r	Carriage Return “r”. It is used to position the cursor to the beginning of the current line. <b>Example:</b> <code>cout &lt;&lt; “\r”;</code>
\\	“\ backslash” It is used to print the backslash character. <b>Example:</b> <code>cout &lt;&lt; “\\t”;</code>
\'	“Single quotation” It is used to print the “apostrophe (') sign or character. <b>Example:</b> <code>cout &lt;&lt; “\’”;</code>
\”	“Double quotation” It is used to print the quotation mark. <b>Example:</b> <code>cout &lt;&lt; \”;</code>

#### Teachers Note



Teachers should ask students to develop program using escape sequences.



## 3.4 OPERATORS:

Operators are special symbols used for specific purposes. Operators perform mathematical operations on Operands. For example:  $x + y$ . Here “x” and “y” are operand and “+” operator. There are seven types of operators that are used in C++ programming.

**3.4.1** Arithmetic Operators

**3.4.2** Increment Operators

**3.4.3** Decrement Operators

**3.4.4** Relational Operators

**3.4.5** Logical Operators

**3.4.6** Assignment Operators

**3.4.7** Arithmetic Assignment Operators

### 3.4.1 Arithmetic Operators:

In Arithmetic operators, five different operators are used to perform an arithmetic operation. All operators except Remainder or Modulus operator can be used in integer and float data type.

- **Addition (+):** It is used to perform arithmetic addition.

**Example:**  $a + b$ ;

- **Subtraction (-):** It's used to perform arithmetic subtraction.

**Example:**  $a - b$ ;

- **Multiplication (\*):** It used to perform arithmetic multiplication.

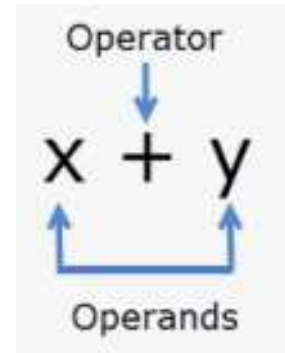
**Example:**  $a * b$ ;

- **Division (/):** It perform the arithmetic division of two numbers.

**Example:**  $a / b$ .

- **Remainder / Modulus (%):** It used to find remainder of a division. It returns the remainder of an integer value. Remainder operator is also known as Modulus operator. This operator is used only with integral data types.

**Example:**  $5 / 2 = 2$  and  $1$  is remainder. We can write in this form  $5 \% 2$



## Develop a simple calculator in C++ by using arithmetic operators.

```
#include<iostream>
#include<cstdio.h>
using namespace std;
int main(void)
{
    int a,b,add,sub,multi,remd;
    float div;
    cout <<"\n \t CALCULATOR";
    cout <<"\n \t =====";
    cout << "\n \t Enter the Value of a.....";
    cin >> a;
    cout << "\n \t Enter the value of b.....";
    cin >> b;
    add = a+b;
    cout << "\n \t Addition of "<< a <<"and.. "<< b << "is...."<<
add;
    sub=a-b;
    cout << "\n \t Subtraction of "<< a <<"and..."<< b <<
"is...."<< sub;
    multi=a*b;
    cout << "\n \t Multiplication of "<< a <<"and..."<< b <<
"is...."<< multi;
    div=a/b;
    cout << "\n \t Division of "<< a <<"and..."<< b << "is...."<<
div;
    remd=a%b;
    cout << "\n \t Remainder of Modulus division of "<< a
<<"and..."<< b << "is...."<< remd;
    return 0;
}
```

### OUTPUT

CALCULATOR

=====

Enter the Value of a..... 15

Enter the Value of b..... 10

Addition of 15 and 10 is .....25

Subtraction of 15 and 10 is..... 5

Multiplication of 15 and 10 is ..... 150

Division of 15 and 10 is..... 1

Remainder of Modulus division of 15 and 10 is.....5



In the division of integers, numbers show only whole numbers in result.

### Teachers Note



Teachers are supposed to explain the precedence of Arithmetic Operators.

### 3.4.2 Increment Operator (++):

The increment operator can be used with any type of variable. It is used to add 1 to the value of a variable. Increment operator represented by ++ (double plus sign). This operator can be applied only to a single variable. There are two ways to use increment operator:

- Prefix Increment Operator: You can apply this operator before the variable name. It can be written like ++a. i.e. x=++a;

#### Example:

```
#include<iostream>
using namespace std;
int main(void)
{
int a=10;
cout << "\n Value of a is...." << ++a; ➔ Prefix Increment Operator
return 0;
}
```

#### Output:

Value of a is .....11

In prefix increment operation value of a is printed as 11 because 1 is added in 'a' before printing.

- **Postfix Increment Operator:** If the increment operator is applied after the variable name, it is known as Postfix Increment operator. It is written like `a++`.

### Example:

```
#include<iostream>
using namespace std;
int main(void)
{
int a=10;
cout << "\n Value of a is...." << a++;
return 0;
}
```

#### Output:

Value of a is .....10

➡ Postfix Increment Operator

**In Postfix increment operation value of a is printed as 10 because 1 is added in 'a' after printing. So the value of 'a' will change to 11 after printing.**

### 3.4.3 Decrement Operator (--):

The decrement operator is same as increment operator but it subtracts 1 from the value of a variable. It is represented by - (double minus sign). It can also be used as prefix and postfix.

### 3.4.4 Relational Operator:

The relational operators are used to test the relation between two values. All relational operators are binary operators. These operators must require two operands. The result of the comparison is **True (1)** or **False (0)**. The relational operators are also known as Comparison Operators.

The following are the relational operators and their operations.

Operators	Meaning	Purpose	Expression
==	Equal to	Its check the equality of two operands values.	a==b
!=	Not equal to	It checks whether the value of the left operand is not equal to the value of the right operand.	a != b

>	Greater than	This operator checks the value of left operand is greater than the value of right operand.	a > b
<	Less than	Its check the value of left operand is less than the value of right operand.	a < b
>=	Greater than or equal to	It checks the value of the left operand is greater than or equal to the value of right operand.	a >= b
<=	Less than or equal to	It checks the value of the left operand is less than or equal to the value of right operand.	a <= b

### Use relational operators in C++ program:

```
// Relational Operators
#include<iostream>
using namespace std;
int main(void)
{
    int a = 10;
    int b = 20;
    cout << "\n \t Relational Operator";
    cout < "\n \t =====";
    cout << "\n \t" << "False \t"<< (a == b)<<"\t false
because 10 is not equal to 20";
    cout << "\n \t" << "True \t"<< (a < b) <<"\t true because
10 is less than 20";
    cout << "\n \t" << "False \t"<<(a > b) <<"\t false because
10 is not greater than 20";
    cout << "\n \t" << "False \t"<<(b <= a)<<"\t false because
20 is not less than or equal to 10";
    cout << "\n \t" <<"True \t"<< (a>=a)<<"\t true because 10
is not greater than 10 but equal to 10";
    cout << "\n \t" << "True \t"<<(b!=a)<<"\t true because 20
is not equal to 10.
return 0;
}
```

#### OUTPUT

```
0
1
0
0
1
1
```



### 3.4.5 Logical Operator:

Logical operators are used to determine two relational expression. These operators can be used in many conditional and relational expressions. There are three logical operators that are used in C++ programming.

Operators	Description	Expression
&&	This operator is called AND. The condition will be true if both expressions are true.	<code>x=10, y=5, z=12 x&gt;y &amp;&amp; x&lt;z</code>
	It's known as OR operator. The condition will be true if any one of the expressions is true.	<code>x=10, y=5, z=12 x&gt;y    x&gt;z</code>
!	This operator is called NOT. The condition will be inverted, false becomes true and true becomes false.	<code>x=10, y=5; !(x&lt;y);</code>

#### Differentiate between relational and logical operators

##### Relational Operator:

- These operators are used to perform logical operations on two given variables.
- The relational operators are used to compare any two values.

##### Logical Operator:

- These operators are used to compare the two relational statements.
- The logical operators are used to combine one and more than one relational expression.
- Like relational operators they also give **True (1)** or **False (0)** results.

### Use Logical and Relational operators in C++ program:

```
#include <iostream>
using namespace std;
int main(void)
{
int x = 10;
int y = 5;
int z = 12;
cout << "\n \t LOGICAL OPERATOR";
cout << "\n \t =====";
cout << "\n \t" << ((x > y) && (x < z)) << "\n \t ADD
OPERATOR"<< endl;
cout << "\n \t" << ((x > y) || (x > z)) << "\n \t OR
OPERATOR"<< endl;
cout << "\n \t" << !(x < y) << "\n \t NOT OPERATOR" <<
endl;
return 0;
}
```

#### OUTPUT

```
1
1
1
```

### 3.4.6 Assignment operator (=) vs Equal to operator (==)

Assignment operator (=)	Equal to operator (==)
The assignment operator (=) is used for assigning a variable to a value.	The equal to (==) operator is used to check the equality of two operands values.
This operator assigns the value of right-side expression to left-side variable. Such as x=10;	This operator compares value of the left side and right-side expression. Such as x=10 and y=10 than x==y If condition true otherwise false.

Use assignment operator in initialization of variable and equal to operator in order to compare two variables.

```
#include<iostream>
using namespace std;
int main(void)
{
    int x=20, y=10;
    cout << "\n \t Assignment vs Equal to Operator";

    cout << "\n \t =====";
    cout << "\n \t x = 20 assignment opt....." << x;
    cout << "\n \t y = 10 assignment opt....." << y;
    cout << "\n \t Equal to opt. result is....." <<
(x==y);
    return 0;
}
```

#### OUTPUT

```
x=20 assignment opt.....20
y=10 assignment opt.....10
Equal to opt is.....0
```

### Arithmetic Assignment Operators:

In arithmetic assignment operators, the arithmetic operator is combined with the assignment operator. The assignment operator comes to the right of an arithmetic operator. This operator is also known as Compound Assignment Operator.

Operators	Description
$+=$ (Addition-assignment)	It adds the right operand to the left operand and assigns the result to the left operand. <b>Example:</b> $a+=2$ means $a=a+2$
$-=$ (Subtraction-assignment)	It subtracts the right operand from the left operand and assigns the result to the left operand. <b>Example:</b> $a-=3$ means $a=a-3$
$*=$ (Multiplication-assignment)	It multiplies right operand with the left operand and assigns the result to the left operand. <b>Example:</b> $a*=4$ means $a=a*4$
$/=$ (Division-assignment)	It divides left operand with the right operand and assigns the result to the left operand. <b>Example:</b> $a/=4$ means $a=a/4$

SUMMARY

- The C++ program consists of three parts.
  - Preprocessor Directives
  - main Function header
  - Body of program
- `#include<iostream>` is used to include header files like `iostream.h`, `conio.h`, etc.
- namespace is the collection of identifiers.
- The `main()` function is compulsory element of the C++ program.
- The comment statements are those statements that are ignored by the compiler. These statements are not executable.
- I/O Stream is a standard library file that contains definitions of Standard Input and Output functions.
- `cout` is an output object. It is used to display the output through output device like monitor.
- `puts()` is a string function and it is included in `<cstdio>` header file.
- `cin` works as an input object in C++.
- Statement terminator (`;`) is used for statement ending in C++ programming.
- Escape Sequences is a non - printable characters. It is used only with `cout` statement.
- Operators are special symbols used for specific purposes.
  - Arithmetic Operator are used for arithmetic operations or calculation.
  - Increment and Decrement Operator are used in two different ways in programming.
    - Prefix
    - Postfix
  - Relational Operators are used to test the relation between two values.
  - Logical Operators are used to determine two relational expression. Relational and Logical Operators works on a decision making and loops.





- ix) Which operator add the first operand to the second operand and gives the result to first operand.
- |       |       |
|-------|-------|
| a. *= | b. += |
| c. ++ | d. +  |
- x) `cout << 12-6/2;` What will be the result on screen?
- |      |       |
|------|-------|
| a. 3 | c. 6  |
| c. 9 | d. 12 |

## B. RESPOND THE FOLLOWING:

1. Use `\a` and `\r` both escape sequences in a program.
2. How many types of **comment** statements are used in C++?
3. Differentiate between Arithmetic operators and Relational operators.
4. Write a program in C++ and use all arithmetic assignment operators.
5. What is basic difference between Assignment operator and Equal to operator.
6. What is the basic difference between `\n` and `\t`?
7. Get the output of following program.

```
#include <iostream>
using namespace std;
int main(void)
{
    int a = 27;
    cout << "a is " << a << endl;
    cout << "a is now" << a++ << endl;
    cout << "a is now " << a << endl;
    cout << "a is now " << --a << endl;
    cout << "a is now " << a << endl;
    return 0;
}
```

## LAB ACTIVITIES

1. Develop programs for manipulating the following formulas.

Title	Formula	Description
Calculating Speed of an Object	$s=d/t$	Speed = distance / time
Newton's second law of motion	$F=ma$	
Calculating acceleration	$a=(v_f-v_i)/t$	Acceleration is equal to (final v - initial v) / time
Area of Triangle	$a= \frac{1}{2} bh$	area = (1/2) (base) (height)
Convert the Celsius to Fahrenheit	$F=(c*1.8)+32$	

2. Write a program to calculate the volume of a box.
3. Write a program of Marksheet takes input of five subjects, print its total and percentage also.
4. Write a code to calculate mathematical expression of  $a^2+2ab+b^2$ .
5. List out errors from the C++ program and remove those errors, write the output.

```
#include<iostream.h>
using namespace std
int main(void);
{
    int x ;
    cout << "\n Enter the value of x.....";
    cin >> x
    cout << "\n The square of x....." << a * a ;
    return 0;
}
```