2- Statement :-

- A statement is a complete instruction asking the computer to carry out some tasks.
- Normally written one per line, although some statements span multiple lines.
- Always end with a semicolon (;).
- For example:

```
x = 2 + 3;
```

An **expression** is any computation which yields a value. When discussing expressions, we often use the term **evaluation**. For example, we say that an expression evaluates to a certain value.

2-1 operators:-

C++ is very rich in built-in operators. An ***operator*** is a symbol that tells the compiler to perform specific mathematical or logical manipulations. There are four general classes of operators in C: ***arithmetic, relational, logical,*** **and** ***bitwise***. In addition, there are some special operators for particular tasks and provides the following types of operators:

- **Arithmetic Operators**
- **Relational Operators**
- **Logical Operators**
- **Assignment Operators**
- **Increment & Decrement Operators**
- **Bitwise Operators**
- **Misc Operators**

2-1-1 Arithmetic in C++

Arithmetic expressions can be made up of constants, variables, operators and parentheses. The *arithmetic operators* in C++ are as follows:-

Operator	Name	Example	Result
+	Addition	5+5.5	10.5
-	Subtraction	5.67-10	4.33
*	Multiplication	2*2.4	4.8
/	Division	8/5	1
%	Remainder	8%5	3

NOTE: The % operator may appear ONLY with integer values.

When expressions are evaluated, they are evaluated left to right according to the following *precedence rules*:

1) ( ) 2) * / % 3) + -

The expression 4 + 8 / 2 is evaluated as follows:

4 + 8 / 2 = (division has the highest precedence)

4 + 4 = 8

The expression (4 + 8) / 2 is evaluated as follows:

(4 + 8) / 2 = (parentheses have the highest precedence)

12 / 2 = 6.

The expression 8 % 3 is evaluated as follows:

8 % 3 = 2 (the whole number remainder)

The expression 8 % 3.0 would be invalid (3.0 is a floating point value and therefore not valid), as would the expression 8.0 % 3.

Example:- write C++ program to print the result of z=(a+b*b-3)/4*a using static cast <type>() operator?

```
# include <iostream.h>

int main()

{

int a,b;  \\ a and b declaration

float z;  \\ z declaration

cout<< "pls enter a,b values"<<endl;

cin>> a>>b; \\ enter a and b values.

z= (float)(a+b*b-3)/4*a;  \\ calculate z value as float.

Cout<<"z="<<z<<endl;

cout<<"z="<< static_cast<int>(z); \\ print the integer values of z

return 0;

}
```

2-1-2 Relational Operators:-

The relational operators are used to test the relation between two values. All relational operators are binary operators and therefore require two operands. A relational expression returns zero when the relation is false and a non-zero when it is true. The following table shows the relational operators:-

Operator	Name	Example	Result
==	Equality	6==6	1(true)
!=	Inequality	6!=6	0(false)
<	Less than	6<6.25	1(true)
<=	Less than and equal	6<=6	1(true)
>	Greater than	6>6	0(false)
>=	Greater than and equal	6.5>=6	1(true)

The operands of relational operator must evaluate to a number, not char actors since it represent a numeric value, and the relational operators should not be used for comparing strings because this will result in the string addresses being compared.

Example: Write an expression to test if a number (n) is even.
Solution:- n%2==0

2-1-3 Logical Operators:-

In the term *relational operator* the word *relational* refers to the relationships values can have with one another. In the term *logical operator* the word *logical* refers to the ways these relationships can be connected together using the rules of formal logic. Then the logical operators are used to combine one or more relational expression. The following table shows the logical operators.

Operator	Name	Example	Result
&&	Logical And	4<6&& 5<=5	1
||	Logical Or	3>2 || 2<4	1
!	Logical Negation or not	!(1==1)	0

Logical ***And*** (&&) will execute the statement if and only if **both** of the simple conditions are true. If either or both of the simple conditions are false, then the program will skip the statements and proceeds to the statements followed.

Logical ***Or*** (||) will execute the statement if either or both of two conditions are true. If both of the simple conditions are false, then the program will skip the statements and proceeds to the statements followed.
Logical *negation* ! is a unary operator, and it will execute the statement if the condition is false and will skip the statement if the condition is true.

P	Q	p && q	p || q	!p
False	**False**	**False**	**False**	**True**
False	**True**	**False**	**True**	**True**
True	**True**	**True**	**True**	**False**
True	**False**	**False**	**True**	**False**

Both the relational and logical operators are lower in precedence than the arithmetic operators. This means that an expression like **10 > 1 + 12** is evaluated as if it were written **10 > (1 + 12)**. The result is, of course, false. Several operations can be combined in one expression, as shown here:
10>5 && !(10<9) || 3<=4 (which will evaluate true).

Example: Write an expression to test if a character (c) is a digit.
Solution: c >='0' && c <= '9'
Example: Write an expression to test if a character c is a letter.
Solution: c >= 'a' && c<= 'z' || c >='A' && c <='Z'
Example: Write an expression to test if n is odd and positive or n is even and negative.
Solution: n%2 !=0 && n>= 0 || n%2 == 0 && n<0

2-1-4 Assignment operators:-

The assignment operator '=' stores the value of the expression on the right hand side of the equal sign to the operand on the left hand side. Examples a=5;
This statement assigns the integer value 5 to the variable a. the part at the left of the assignment operators (=) is known as the *lvalue* (left value) and the right one as the *rvalue* (right value). The most important rule when assigning is the right-to-left rule. The assignment operation always takes place from right to left, *and never the other way*. The Assignment operator are summarized in table below:-

Operator	Example	Equivalent to
=	n = 25	
+=	n += 25	n = n + 25
-=	n -= 25	n = n - 25
*=	n *= 25	n = n * 25
/=	n /= 25	n = n / 25
%=	n %= 25	n = n % 25
&=	n &= OxF2F2	n = n & OxF2F2
|=	n |= OxF2F2	n = n| OxF2F2
^=	n ^= OxF2F2	n = n^ OxF2F2
<<=	n<<=4	n=n<<4
>>=	n>>=4	n=n>>4

Example: Illustrate use of the assignment operators?
Solution:

```
# include <iostream.h>
int main ()
{
int i=5;
int k=2;
i /= k;
cout << " i is" << i << endl;
int j=20;
j *=k;
cout << " j is" << j << endl;
int m=15;
m%=4;
cout << " m is" << m << endl;
return 0;
}
```

Example:

```
#include <iostream.h>
int main(){
int a,b;
a=10;
```

```
b=4;
a=b;
b=7;
cout<<"a="<<a<<endl;
cout<<"b="<<b<<endl;
return 0;}
```

2-1-5 Increment & Decrement Operators:-

C++ allows two very useful operators not generally found in other computer languages. These are the increment and decrement operators, ++ and --. The operation ++ adds 1 to its operand, and -- subtracts 1. Therefore, the following are equivalent operations:-

int k=5; k=k+1

Operator	Name	Example	Result
++	Auto increment (prefix)	++k	6
++	Auto increment (prefix)	++k + 10	16
++	Auto increment (postfix)	K++	6
++	Auto increment (postfix)	K++ +10	15
--	Auto decrement (prefix)	--k	4
--	Auto decrement (prefix)	--k + 10	14
--	Auto decrement (postfix)	k--	4
--	Auto decrement (postfix)	k-- +10	15

Example: write a C++ program to show the increment and decrement of the integer variable a=5;

```
# include <iostream.h>
int main()
{
int a=5;
cout<<++a<<endl; // print 6
cout<<a++<<"\n"; // print 6 the icrement by 1 to become 7 at memory
```

```
cout<<++a<<"\n"; // here print 8
cout<<a++<<"\n"; // print also 8 but at the end store a=9 at memory
cout<<a<< "\n";   \\ to check a=9
return 0;
}
```

2-1-6 Bitwise Operators:-

Unlike many other languages, C++ supports a complete complement of *bitwise operators.* Since C++ was designed to take the place of assembly language for most programming tasks, it needed the capability to support many operations that can be done in assembler. *Bitwise operations* are the testing, setting, or shifting of the actual bits in a byte or word, which correspond to the standard **char** and **int** data types and variants. Bitwise operators cannot be used on type **float**, **double**, **long double**, **void**, or other more complex types. The bitwise AND, OR, and NOT (one's complement) are governed by the same truth table as were their logical equivalents except that they work on a bit-by-bit level. The exclusive OR ^ has the truth table shown here:

Q	**p**	**q^p**
0	**0**	**0**
0	**1**	**1**
1	**0**	**1**
1	**1**	**0**

The Bitwise operator:-

Operator	**Action**
&	**AND**
|	**OR**
^	**Exclusive OR (X-OR)**
~	**One's complement**
>>	**Shift Right**
<<	**Shift lift**

The shift operators, >> and <<, move all bits in a variable to the right or left as specified. The general form of the shift right statement is
variable >> number of bit positions
and the shift left statement is
variable << number of bit positions.

Example:- write C++ program to input two values and using AND, OR, XOR, shift to lift by 2(N1) between them, then print the results?

Solution:-

```
#include <iostream.h>
int main(){
const int N1 = 187;
const int N2 = 242;
cout<<N1<<"&"<<N2<<"="<< (N1 & N2)<<"\n";
cout<<N1<<"|"<<N2<<"="<< (N1 | N2)<<"\n";
cout<< N1<< "^" << N2<< "="<< N1 ^ N2 <<"\n";
cout << N1<<"<<2="<<(N1<<2)<<"\n\n";
return 0;
}
```

2-1-7 Misc Operators:-

There are few other operators supported by C++ Language

Operator	Description
sizeof	sizeof operator returns the size of a variable. For example, sizeof(a), where a is integer, will return 4.
Condition ? X : Y	Conditional operator. If Condition is true ? then it returns value X : otherwise value Y
,	Comma operator causes a sequence of operations to be performed. The value of the entire comma expression is the value of the last expression of the comma-separated list.
. (dot) and -> (arrow)	Member operators are used to reference individual members of classes, structures, and unions.
Cast	Casting operators convert one data type to another. For example, int(2.2000) would return 2.
&	Pointer operator & returns the address of an variable. For example &a; will give actual address of the variable.
*	Pointer operator * is pointer to a variable. For example *var; will pointer to a variable var.

2-1-7-1 Conditional operator

The conditional operator ?: is called ternary operator as it requires three operands.
The format of the conditional operator is:

Conditional expression ? expression1 : expression2;

If the value of conditional expression is true then the expression1 is evaluated, otherwise expression2 is evaluated.

```
int x = 10;
y = x>9 ? 100 : 200;
```

Example:

```
int m= 1, n=2;
int min = (m<n ? m : n);
cout << min; // result min =1
```

Example:

```
int min = (m<n ? ++m : n++); // result min =2
```

m is incremented because ++m is evaluated but n is not incremented because n++ is not evaluated.

Example:

```
int x = 10, y = 5, z = 42;
int t = y > x ? z : z + 5;
cout << t; // result t =47
```

Example: Write an expression to give the absolute value of a number n.

Solution: (n >=0 ? n: -n)

2-1-7-2 Comma operator

Multiple expressions can be combined into one expression using the comma operator. The comma operator takes two operands. It first evaluates the left operand and then the right operand, and returns the value of the latter as the final outcome,

Example:

```
int m, n, min;
int mCount = 0, nCount = 0;
min= (m < n ? mCount++, m : nCount++, n);
```

When m is less than n, mCount++ is evaluated and the value of m is stored in min. Otherwise, nCount++ is evaluated and the value of n is stored in min.

2-1-7-3 Sizeof operator:-

As we know that different types of variables, constant, etc. require different amounts of memory to store them. The sizeof operator can be used to find how many bytes are required for an object to store in memory.

Example:

```
# include <iostream.h>
int main (void)
{
cout << "char size = " << sizeof (char) << " bytes\n";
cout << "char* size = " << sizeof (char*) << " bytes\n";
cout << "short size = " << sizeof (short) << " bytes\n";
cout << "int size = " << sizeof (int) << " bytes\n";
cout << "long size = " << sizeof (long) << " bytes\n";
cout << "float size = " << sizeof (float) << " bytes\n";
cout << "double size = " << sizeof (double) << "bytes\n";
cout << "1.55 size = " << sizeof (1.55) << " bytes\n";
cout << "1.55L size = " << sizeof (1.55L) << " bytes\n";
cout << "HELLO size = " << sizeof ("HELLO") << "bytes\n";
}
```

When run, the program will produce the following output (some small different values can be found on author PC.)

```
char size = 1 bytes
char* size = 2 bytes
short size = 2 bytes
int size = 2 bytes
long size = 4 bytes
float size = 4 bytes
double size = 8 bytes
1.55 size = 8 bytes
1.55L size = 10 bytes
HELLO size = 6 bytes
```

2-1-7-5 Simple Type Conversion:-

The process in which one pre-defined type of expression is converted into another type is called conversion. There are two types of conversion in C++.

- Implicit conversion
- Explicit conversion

- **Implicit conversion**

Data type can be mixed in the expression.

Example:-

```
double a;
int b = 5;
float c = 8.5;
a = b * c;
```

When two operands of different type are encountered in the same expression, the lower type variable is converted to the higher type variable. The following table shows the order of data types:-

Order of data types	
Data type	**order**
long double	(highest)
double	↑
float	
long	
int	
char	(lowest)

The int value of b is converted to type float and stored in a temporary variable before being multiplied by the float variable c. The result is then converted to double so that it can be assigned to the double variable a.

- **Explicit conversion:-**

It is also called type casting. It temporarily changes a variable data type from its declared data type to a new one. It may be noted here that type casting can only be done on the right hand side of the assignment statement.

```
T_Pay = double (salary) + bonus;
```

Initially variable salary is defined as float but for the above calculation it is first converted to double data type and then added to the variable bonus.

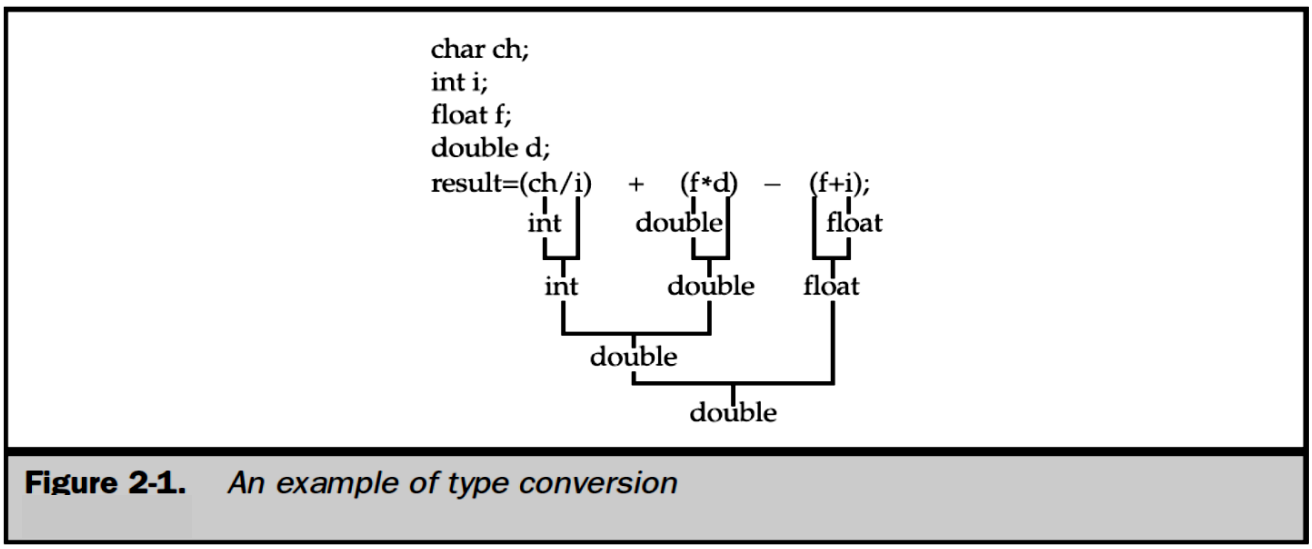

Figure 2-1. *An example of type conversion*

When the result of an arithmetic operation exceeds the range of the variable's data type, an error called overflow occurs.

Example:-

```
int m, n, min;
int mCount = 0, nCount = 0;
min= (m < n ? mCount++, m : nCount++, n);
```

When m is less than n, mCount++ is evaluated and the value of m is stored in min. Otherwise, nCount++ is evaluated and the value of n is stored in min

2.2 Operator Precedence

The order in which operators are evaluated in an expression is significant and is determined by precedence rules. These rules divided the C++ operators into a number of precedence levels (as table below). Operators in higher levels take precedence over operators in lower levels.

level	Operator						Kind	Order	
Highest	: :						Unary	Both	
	( )	[ ]	->	.			Binary	Left to Right	
	±	++ --	! ~	* &	New Delete	Sizeof ( )	Unary	Right to Left	
	-> *	. *					Binary	Left to Right	
	*	/	%				Binary	Left to Right	
	+	-					Binary	Left to Right	
	<<	>>					Binary	Left to Right	
	<	< =	>	> =			Binary	Left to Right	
	= =	!=					Binary	Left to Right	
	&						Binary	Left to Right	
	^						Binary	Left to Right	
	|						Binary	Left to Right	
	&&						Binary	Left to Right	
	||						Binary	Left to Right	
	? :						Binary	Left to Right	
	=	+= - =	*= /=	^= %=	&= |=	<<= >>=	Binary	Right to left	
Lowest	,						Binary	Left to Right	

جدول (1.10): يبين اسبقيات العوامل

قواعـــــــد الأسبـــقيات		
The Unary Operators العوامل الاحادية	+ ،- ،++ ،-- ،!	الاسبقية العليا (تنفذ اولا)
The Binary Arithmetic Operations العوامل الرياضية الثنائية	* ،/ ،%	
The Binary Arithmetic operations العوامل الرياضية الثنائية	+ ،-	
The Boolean operations العوامل المنطقية	< ،> ،<= ،>=	
The Boolean operations العوامل المنطقية	== ،!=	
The Boolean Operations العوامل المنطقية	&&	
The Boolean Operations العوامل المنطقية	||	الاسبقية الدنيا (تنفذ اخيرا)

2.4 Mathematical library function (math.h):-

In case we want to use some of mathematical function like (e^x , x^y ,log x,.....). In this case we must include the header file <math.h> in our program because these functions are defined in that file, where the value of x is of type double.

Mathematical functions	C++ functions
x^y	pow(x,y)
e^x	exp(x)
$\log x$	log (x)
$\sqrt{x}$	sqrt (x)
$\sin x$	sin(x)
$\cos x$	cos(x)
$\tan x$	tan(x)
$\lvert x \rvert$	fabs(x)

Example: w.p.in C++ to compute and print the distance between two point (x1,y1) and (x2,y2) in Cartesian coordinates where:

***Dis* tance** $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$

```
# include <iostream.h>
# include <math.h>
int main(){
int x1,x2,y1,y2;
double distance;
cout <<"Input the value of x1,x2,y1,y2"<<endl;
cin >>x1>>x2>>y1>>y2;
distance=sqrt (pow ((x1-x2),2)+pow((y1,y2),2));
cout <<"The distance is "<<distance<<endl;
return 0;}
```

Example: w.p. in C++ to find the value of y from the expression below:

$$\frac{2x^2 + 5e^{3x} \sin 6b}{\sqrt{z}}$$

```
# include <iostream.h>
# include <math.h>
int main() {
int x,b,z;
double y;
cout <<"Input the value of x,b,z"<<endl;
cin >>x>>b>>z;
y=((2*pow(x,2))+5*exp(3*x)*sin(6*b))/sqrt(z);
cout <<"y= "<<y<<endl;
return 0;}
```

Example:- write C++ program to convert the (sec42200) to the corresponding in hours, minutes and seconds?

```
#include<iostream.h>
main() {
int sec =42200 % 60;
int temp =42200 / 60;
```

int min =temp % 60;

int hour = temp / 60;

cout<<"hour="<< hour<<",min="<< min<<",sec="<< sec;

return 0;}

Example: W.P. in C++ which inputs a temperature reading expressed in Fahrenheit and output its equivalent in Celsius, using the formula: $c = \frac{5(f-32)}{9}$

```
#include <iostream.h>
int main()
{
double f, c;
cout << "Temperture in Fahrenhait:";
cin >> f;
c=5*(f-32)/9;
```

H.W:-

1- write C++ program to find the value Y of the equation

a- $Y=[(x-z)^2-(x+z)]/32.$

b- $z = \frac{x^3-4x^2+x}{x^2+2x+2}$

c- $z = 1 - \frac{x^2}{2!} + \frac{x^4}{4!}$

2- what is the computation sequence of the following expression:-

$$((a+b)/(c-5))/(\frac{d+7}{\frac{e-37}{3}})$$

If a=10, b=20, c=14, d=8, and e=40.

3- for each the following algebraic expressions write an equivalent C++ arithmetic expression:-

1-) $\frac{a^3-b^2}{c^2+25}$

2-) $\frac{1}{x} + \frac{1}{x^2} + \frac{1}{x^3} + \frac{1}{x^4}$

3-) $x + y^2 + \frac{t}{z}$

f- w.p. C++ expression for the following formula use functions from the math library described?

1- $(\sin x)^2 \times (\cos)^2$

2- $e^{\frac{1}{2}\sqrt{tanx}}$

3- $\frac{\left(log\left(\frac{x^2}{1-x}\right)\right)}{(x^{5+x})}$

4- $y = sin\ x + e^{|x|+x} + x^{1.5}$