Example:

```
1.  int x = 3 ;
2.  int z = 7 ;
3.
4.  cout <<  "z + x= " ;
5.  cout << z + x ;
```

الناتج هو "z + x= "
الناتج هو 10

ملاحظة : ما داخل الأقواس "المفردة' أو "المزدوجة" يعتبر نص.

3. Compound Assignment Operators

C++ allows combining the arithmetic operators with assignment operator as in the following table.

Operator	C++ expression	Equivalent expression	Explanation and use
+=	B + = 5;	B = B +5;	`int B= 4;` `B+=5 ;` // B=9
−=	C – = 6;	C = C – 6;	`int C = 10;` `C-= 6 ;` // C=4
*=	D * = 2;	D = D*2;	`int D = 10;` `D*=2;` // D=20
/=	E / = 3;	E = E/3;	`int E = 21;` `E/=3;` // E = 7
%=	F % = 4;	F = F % 4 ;	`int F=10;` `F%=4 ;` // F=2

4. Relational Operators:

The relational operators are explained in the following table:

Operator	Usage	Example	Explanation
<	Less than	A<B	A is less than B.
>	Greater than	A>B	A is greater than B.
<=	Less than or equal to	A<=B	A is less than or equal to B.
>=	Greater than or equal to	A>=B	A is greater than or equal to B.
==	Equality	A == B	A equal to B.
!=	Not equal to	A != B	A is not equal to B.

Example:

```
1. int x = 3 ;
2. int z = 7 ;
3.
4. if ( x != z)
5. {
6.    cout << "Not equal" ;
7. }
```

إذا كان x لا يساوي z

فاطبع الجملة "Not equal"
ناتج البرنامج "Not equal" لأن ناتج الشرط True "صواب"

5. Logical Operators

The logical operators are used to combine multiple conditions (logical statements). The following table describes the logical operators:

Operator	Usage	Example
&& (logical AND)	The compound condition is true, if both conditions are true.	((a>b) && (a>c))
|| (logical OR)	The compound statement is true, if any or both conditions are true.	((a>b) || (a>c))
! (logical NOT)	It negates the condition.	!(a>b)

Example:

```
1. int x = 3 ;
2. int z = 7 ;
3.
4. if ( x >0 && z>0)
5. {
6.    cout << "Both numbers positive" ;
7. }
```

إذا كان x أكبر من الصفر وأيضاً z أكبر من الصفر
(ناتج الشرط True)

ناتج البرنامج " Both numbers positive"

Equal and Assigned المساواة والإسناد

الإسناد: هو إعطاء المتغير قيمة:

Example:

```
1 int x ;
2 x = 7 ;
3 cout << x;
```

الناتج 7

المساواة: هو مقارنة قيمتين:

Example:

```
1. int x = 7 ;
2. int y = 7 ;
3.
4. cout << x==y;
5. cout << x > 3
6.
7. int z = x == 8
8. cout << z ;
```

الناتج صواب (True) "سيطبع 1 في الشاشة"
الناتج خطأ (False) "سيطبع 0 في الشاشة"

اسناد ناتج المقارنة للمتغير z
الناتج False لأن x تساوي 7 وليس 8
وستتم طباعة 0 على الشاشة.

ملاحظة:

x = = y تعني (هل أن x يساوي y) وهي عملية مقارنة ناتجها إما صواب أو خطأ.

Example: Write a C++ program to calculate the rectangle area. **برنامج لإيجاد مساحة المستطيل:**

من المهم :
- فهم فكرة البرنامج.
- تحويل الفكرة إلى خطوات منطقية، وأهمها معرفة معادلة مساحة المستطيل.

مساحة المستطيل = الطول × العرض

إذاً نحتاج إلى ٣ متغيرات ، متغيرين سيقوم المستخدم بإدخالهما (الطول والعرض) ومتغير سيحتوي على ناتج ضرب المتغيرين (المساحة).

خطوات الحل البرمجي:

١) التصريحات Declare (تعريف المتغيرات).
٢) المدخلات Input (إسناد قيم للمتغيرات "الطول والعرض").
٣) المعالجة Process (الضرب).
٤) المخرجات Output (مساحة المستطيل).

```
#include<iostream.h>
void main( )
{
    1. int height, width, area;
    2. cout << "Enter Height: ";
    3. cin >> height;
    4. cout << "Enter Width: ";
    5. cin >> width
    6. area = height * width
    7. cout << "---------\n";
    8. cout << "Resutl is: " <<area << endl;
}
```

طباعة نص يطلب إدخال الطول
استقبال الطول من المستخدم
طباعة نص يطلب إدخال العرض
استقبال العرض من المستخدم

طباعة خط أفقي وسطر جديد
طباعة "النتيجة هي" ثم ناتج الضرب ثم سطر جديد

الناتج:

```
Enter Height: 7
Enter Width: 8
---------
Result is: 56
Press any key to continue_
```

Examples

```
y= ((2 == 4) && (7>5));
z= !(1 > 4);
w= ((2 > 0) || (5<0));
```

the results are:

```
y=0 , z=1 , w=1
```

The following program illustrates the application of logical operators.

```
#include <iostream.h>
main()
{
     int p=1, q=0, r=1, s,t,x, y, z;
     s = p||q;
     t = !q;
     x = p&&q;
     y = (p || q && r||s);
     z = (!p || !q && !r || s);
     cout << "s = "<<s << ", t = " <<t <<", x = "<<x << endl;
     cout << "y = " <<y <<", z = "<<z<< endl;
     return 0;
}
```

The expected output are given below :

```
s = 1, t = 1, x = 0
y = 1, z = 1
```

6. Bitwise Operators

Bitwise operation means convert the number into binary and perform the operation on each bit individually. The bitwise operators are listed in the table below:

Operator	Description	Example of code
&	Bitwise AND	`A & B;`
|	Bitwise OR	`A | B;`
^	Bitwise Exclusive OR	`A ^ B;`
~	complement	`~A;`
<<	Shift Left	`A<<2; //shift left by 2 places`

>>	Shift Right	`A>>2;//shift right by 2 places`
&=	Bitwise AND assign	`A &= B;// A=A&B`
|=	Bitwise OR assign	`A |= B; // A=A|B`
^=	Bitwise XOR assign	`A ^= B; // A=A^B`
<<=	Bitwise shift left assign	`A <<= 2;//Shift left by 2 places and assign to A`
>>=	Bitwise shift right assign	`A >>= 1;//Shift right by 1 place and assign to A`

Example

```
#include<iostream.h>
void main()
{
   int A=42, B=12, C=24, D;
   D = A^B;
   C <<= 1;
   A <<=2;
   B >>=2 ;
   cout<<"A="<<A<<endl;
   cout<<"B="<<B<<endl;
   cout<<"C="<<C<<endl;
   cout<<"D="<<D;
}
```

The expected outputs are given below :

```
A = 168
B = 3
C = 48
D = 38
```

```
#include<iostream.h>
void main()
{
   int A =20, D , E, F;
   int B = 18, C = 30;
   D = C^B;
   E = A &B ;
   F = C|A ;
   cout <<"E="<<E<<endl;
   cout <<"F="<<F<<endl;
   cout <<"D="<<D<<endl;
}
```

The expected outputs are given below :

```
E = 16
F = 30
D = 12
```

7. Increment and Decrement Operators

They are used for increasing and decreasing a variable by unity. These operators may be placed before or after the variable name as shown in the following table.

Operator	Pre or post	Description
++k	Pre-increment	First increase the value of k by one then evaluate the current statement by taking incremented value.