Operators

Operators in C++ are classified into: arithmetic, logical, relational, and bitwise operators.

Arithmetic operators

Operator	Action
-	Subtraction, also unary minus
+	Addition
*	Multiplication
/	Division
%	Modulus
--	Decrement
++	Increment

Examples:

```
z = x + y;
area_square = side * side;
area_triangle = (base*height)/2;
z = x % y;
```

Notes:

- The result of a binary operation with values of the same type is another value of the same type.
- If a binary operation is performed between values with different types, then the value with the lower type is converted to the higher type, and thus the operation is performed with values of the same type.
- When you apply / to an integer, any remainder will be truncated.
- The modulus operator % produces the remainder of an integer division. It cannot be used with floating-point types.

Cast operator

Example

Write a C++ program that reads two marks and prints the average.

```
#include <iostream>
using namespace std;

int main()
{
     int  mark1, mark2, sum, count = 2;
     float  average;
     cout << "Enter first mark: " << endl;
     cin >> mark1;
     cout << "Enter second mark: " << endl;
     cin >> mark2;
     sum = mark1 + mark2;
     average = sum/count;
     cout << "The average is: " << average << endl;
     return 0;
}
```

if `mark1` is 90 and `mark2` is 91 then average is 90.0 not 90.5.

To compute the `average` correctly, we use cast operator as follows:

```
average = (float)sum/(float)count;
```

Note that that cast operator affects only the value used in the computation, it does not change the type of the variables `sum` and `count`.

Priority of arithmetic operations

Precedence	Operator	Associativity
1	( )	innermost first
2	Unary + - cast	right to left
3	Binary * / %	left to right
4	Binary + -	left to right

Example:

Let us solve the following equation according to the priority of operations:

12*m + (m*n % 13 + m/n) * k/10

Assume m = 12 , n = 5 and k = 20

Sub expression	Result	Expression after each step
m * n	60	12*m+(60%13+m/n)*k/10
60 % 13	8	12*m+(8+m/n)*k/10
m / n	2	12*m +(8+2)*k/10
8 + 2	10	12*m+10*k/10
12 * m	144	144+10*k/10
10 * k	200	144+200/10
200 /10	20	144+20
144 + 20	164	164

Example:

Write a C++ program to compute the volume of a sphere.

$$v = 4 * \pi * r^3/3$$

```
#include <iostream>
using namespace std;
int main()
{
  const float PI = 3.141593;
  float  radius , volume;
  cout << "Enter the radius: " << endl;
  cin >> radius;
  volume = (4.0 * PI * radius * radius * radius)/3.0;
  cout << "The volume of sphere: " << volume << endl;
  return 0;
}
```

Example

Write a C++ program to compute the following equation:

$$f = \frac{x^3 - 2x^2 + x - 6.3}{x^2 + 0.505x - 3.14}$$

```
#include <iostream>
using namespace std;
int main()
{
float x, f;
cout << "Enter a value of x: " << endl;
cin >> x;
f = (x*x*x - 2*x*x + x - 6.3)/(x*x + 0.505*x - 3.14);
cout << "f = " << f << endl;
return 0;
}
```

The statement

```
f = (x*x*x - 2*x*x + x - 6.3)/(x*x + 0.505*x - 3.14);
```

can also be written as

```
f = (x*x*x - 2*x*x + x - 6.3)/
    (x*x + 0.505*x - 3.14);
```

Or

```
float  numerator, denominator;
numerator = x*x*x - 2*x*x + x - 6.3;
denominator = x*x + 0.505*x - 3.14;
f = numerator / denominator;
```

Overflow and Underflow

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

Example

```
float  x = 2.5e30;     //  x  = 2.5 × 10^30
float  y = 1.0e30;     //  y = 1.0 × 10^30
float  z;
z = x * y;
```

Here, the value of z will be 2.5e60, i.e. *overflow*. C++ generates an error message "Floating-point error: Overflow".

Similarly, when the result of an operation is too small to store in the memory allocated for the variable, an error called *underflow* occurs.

Example

```
float  x = 2.5e-30;     //  x  = 2.5 × 10^-30
float  y = 1.0e-30;     //  y = 1.0 × 10^-30
float  z;
z = x * y;
```

Here, the value of z will be 2.5e-60, i.e. *underflow*. C++ replaces this value by zero.

Increment / Decrement operators

are applied either in a **prefix position** (before the identifier) as in `++count`, or in a **postfix position** (after the identifier) as in `count++`.

The statement

```
x++;
```

is equal to the statement

```
x = x + 1;
```

and

```
--y;
```

is equal to the statement

```
y = y - 1;
```

However, there is a difference between the prefix and postfix forms when you use these operators in an **expression**.

The statement

```
w =  ++x  -  y;
```

is equivalent to the statements

```
x = x + 1;
w = x - y;
```

while the statement

```
w = x++  -  y;
```

is equivalent to the statements

```
w = x - y;
x = x + 1;
```

Example

```
#include <iostream>
using namespace std;

int main()
{
int  x , y , z;
x = 2;
y = 5;
z = x++ + y;
cout<<"x="<< x <<" y="<< y <<" z="<< z << endl;
z = ++x + y--;
cout<<"x="<< x <<" y="<< y <<" z="<< z << endl;
return 0;
}
```

Logical operators

Operator	Action
&&	AND
||	OR
!	NOT

Example:

```
(a || b) && !(a && b)
```

Relational operators

Operator	Action
>	Greater than
>=	Greater than or equal
<	Less than
<=	Less than or equal
==	Is equal
!=	Not equal

Result: 1 True
0 False

Example:

```
10>5 && !(10<9) || 3<=4
```

In this case the result is true.

Bitwise operators

Operator	Action
&	AND
|	OR
^	Exclusive OR (XOR)
~	One's complement (NOT)
>>	Shift right
<<	Shift left

Example:

```
int  x=10, y=2, r;
r = x & y;
r= x | y;
r = x ^ y;
r = x >> 1;
```

Assignment Operators

=
+=
-=
*=
/=
%=

Example:

```
x = x + 3;          sum = sum + x;      d = d / 4.5;
x += 3;             sum += x;           d /= 4.5;

r = r % 2;
r %= 2;
```

Exercises

1. Write a C++ program to perform the following equations:

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

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

c) $z = [(x - y)^2 - (x + y)]/32$

2. Consider the arithmetic expressions

1. a * b / (-c * 31 % 13) * d
2. a * (b * b) – (c * b) + d

Write the order in which the operations will be executed?

3. What is the computation sequence of the following expression

(a + b / (c - 5)) / ((d + 7) / (e - 37) / 3)

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

4. For each the following algebraic expressions write an equivalent C++ arithmetic expression.

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

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

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

5. Determine the values of the variables for each of the following C++ statements:

a) `z = x++*y;`

b) `z = 2*++x*y;`

c) `x += 4+--y/x---3;`

d) `y %= x;`

Assume that x=4 , y=6. Assume that all variables are integers.

6. What does this statement mean?

```
total += --n;
```