Operators

Operators are special symbols used for specific purposes. C++ provides six types of operators.
Arithmetical operators, Relational operators, Logical operators, Unary operators, Assignment operators, Conditional operators, Comma operator

Arithmetical operators

Arithmetical operators +, -, *, /, and % are used to performs an arithmetic (numeric) operation. You can use the operators +, -, *, and / with both integral and floating-point data types. Modulus or remainder % operator is used only with the integral data type.
Operators that have two operands are called binary operators.

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.

Relational Operators	Meaning
<	Less than
<=	Less than or equal to
==	Equal to
>	Greater than
>=	Greater than or equal to
! =	Not equal to

Logical operators

The logical operators are used to combine one or more relational expression. The logical operators are

Operators	Meaning
||	OR
&&	AND
!	NOT

Unary operators

C++ provides two unary operators for which only one variable is required.
For Example

```
a = - 50;
a = + 50;
```

Here plus sign (+) and minus sign (-) are unary because they are not used between two variables.

Assignment operator

The assignment operator '=' is used for assigning a variable to a value. This operator takes the expression on its right-hand-side and places it into the variable on its left-hand-side. For example:

```
m = 5;
```

The operator takes the expression on the right, 5, and stores it in the variable on the left, m.

```
x = y = z = 32;
```

This code stores the value 32 in each of the three variables x, y, and z.

in addition to standard assignment operator shown above, C++ also support compound assignment operators.

Compound Assignment Operators

Operator	Example	Equivalent to
+ =	A + = 2	A = A + 2
- =	A - = 2	A = A - 2
% =	A % = 2	A = A % 2
/=	A/ = 2	A = A / 2
*=	A * = 2	A = A * 2

Increment and Decrement Operators

C++ provides two special operators viz '++' and '--' for incrementing and decrementing the value of a variable by 1. The increment/decrement operator can be used with any type of variable but it cannot be used with any constant. Increment and decrement operators each have two forms, pre and post.

The syntax of the increment operator is:
Pre-increment: ++variable
Post-increment: variable++
The syntax of the decrement operator is:
Pre-decrement: --variable
Post-decrement: variable--

In Prefix form first variable is first incremented/decremented, then evaluated
In Postfix form first variable is first evaluated, then incremented/decremented

```
int x, y;
int i = 10, j = 10;
x = ++i; //add one to i, store the result back in x
y = j++; //store the value of j to y then add one to j
cout << x; //11
cout << y; //10
```

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 a = 5, b = 6;
big = (a > b) ? a : b;
```

The condition evaluates to false, therefore biggets the value from b and it becomes 6.

The comma operator

The comma operator gives left to right evaluation of expressions. When the set of expressions has to be evaluated for a value, only the rightmost expression is considered.

```
int a = 1, b = 2, c = 3, i; // comma acts as separator, not as an operator
i = (a, b); // stores b into i
```

Would first assign the value of a to i, and then assign value of b to variable i. So, at the end, variable i would contain the value 2.

The 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. For example

```
sizeof (char) returns 1
sizeof (float) returns 4
```

the sizeof operator determines the amount of memory required for an object at compile time rather than at run time.

The order of Precedence

The order in which the Arithmetic operators (+,-,*,/,%) are used in a. given expression is called the order of precedence. The following table shows the order of precedence.

Order	Operators
First	()
Second	*, /, %
Third	+, -

The following table shows the precedence of operators.

++, --(post increment/decrement)	
++ (Pre increment) -- (Pre decrement), sizeof ( ), !(not), -(unary), +(unary)	Highest
*,/, %	To
+, -	
<, <=, >, >=	Lowest
==,!=	
&&	
? :	
=	
Comma operator	