### **Operators in C++**

Operators are symbols that perform operations on variables and values. For example, + is an operator used for addition, while - is an operator used for subtraction.

Operators in C++ can be classified into 6 types:

* Arithmetic Operators
* Assignment Operators
* Relational Operators
* Logical Operators
* Bitwise Operators
* Other Operators

1. Arithmetic Operators:

Arithmetic operators are used to perform arithmetic operations on variables and data.

For example, a + b;

Here, the + operator is used to add two variables a and b. Similarly, there are various other arithmetic operators in C++.

|  |  |
| --- | --- |
| **Operator** | **Operation** |
| + | Addition |
| - | Subtraction |
| \* | Multiplication |
| / | Division |
| % | Modulo Operation (Remainder after division) |

* Increment & Decrement Operators:

C++ also provides increment and decrement operators: ++ and -- respectively. ++ increases the value of the operand by **1**, while -- decreases it by **1**.

For example,

int num = 5;

// increasing num by 1

++num;

Here, the value of num gets increased to **6** from its initial value of **5**.

we used ++ and -- operator as **prefixes**. We can also use these operators as **postfix**.

There is a slight difference when these operators are used as a prefix versus when they are used as a postfix.

1. Assignment Operators:

In C++, assignment operators are used to assign values to variables.

| Operator | Example | Equivalent to |
| --- | --- | --- |
| = | a = b; | a = b; |
| += | a += b; | a = a + b; |
| -= | a -= b; | a = a - b; |
| \*= | a \*= b; | a = a \* b; |
| /= | a /= b; | a = a / b; |
| %= | a %= b; | a = a % b; |

1. Relational Operators:

A relational operator is used to check the relationship between two operands.

For example,

// checks if a is greater than b

a > b;

Here, > is a relational operator. It checks if a is greater than b or not.

If the relation is **true**, it returns **1** whereas if the relation is **false**, it returns **0**.

|  |  |  |
| --- | --- | --- |
| **Operator** | **Meaning** | **Example** |
| == | Is Equal To | 3 == 5 gives us **false** |
| != | Not Equal To | 3 != 5 gives us **true** |
| > | Greater Than | 3 > 5 gives us **false** |
| < | Less Than | 3 < 5 gives us **true** |
| >= | Greater Than or Equal To | 3 >= 5 give us **false** |
| <= | Less Than or Equal To | 3 <= 5 gives us **true** |

1. Logical Operators:

Logical operators are used to check whether an expression is **true** or **false**. If the expression is **true**, it returns **1** whereas if the expression is **false**, it returns **0**.

|  |  |  |
| --- | --- | --- |
| Operator | Example | Meaning |
| && | expression1 **&&** expression2 | Logical AND. True only if all the operands are true. |
| || | expression1 **||** expression2 | Logical OR. True if at least one of the operands is true. |
| ! | **!**expression | Logical NOT. True only if the operand is false. |

In C++, logical operators are commonly used in decision making.

1. Bitwise Operators:

In C++, bitwise operators are used to perform operations on individual bits. They can only be used alongside char and int data types.

|  |  |
| --- | --- |
| Operator | Description |
| & | Binary AND |
| | | Binary OR |
| ^ | Binary XOR |
| ~ | Binary One's Complement |
| << | Binary Shift Left |
| >> | Binary Shift Right |