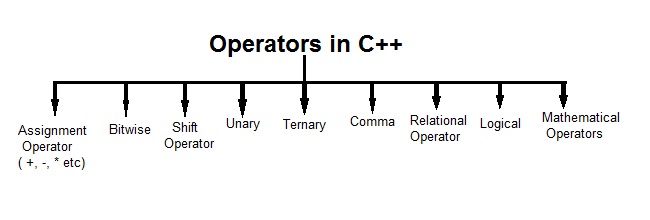
### Operators in C++

Operators are special type of functions, that takes one or more arguments and produces a new value. For example : addition (+), substraction (-), multiplication (\*) etc, are all operators. Operators are used to perform various operations on variables and constants.



#### Types of operators

1. Assignment Operator
2. Mathematical Operators
3. Relational Operators
4. Logical Operators
5. Bitwise Operators
6. Shift Operators
7. Unary Operators
8. Ternary Operator
9. Comma Operator

#### Assignment Operator ( = )

Operates '=' is used for assignment, it takes the right-hand side (called rvalue) and copy it into the left-hand side (called lvalue). Assignment operator is the only operator which can be overloaded but cannot be inherited.

#### Mathematical Operators

There are operators used to perform basic mathematical operations. Addition (+) , subtraction (-) , diversion (/) multiplication (\*) and modulus (%) are the basic mathematical operators. Modulus operator cannot be used with floating-point numbers.

C++ and C also use a shorthand notation to perform an operation and assignment at same type. *Example*,

int x=10;

x += 4 // will add 4 to 10, and hence assign 14 to X.

x -= 5 // will subtract 5 from 10 and assign 5 to x.

#### Relational Operators

These operators establish a relationship between operands. The relational operators are : less than (<) , grater thatn (>) , less than or equal to (<=), greater than equal to (>=), equivalent (==) and not equivalent (!=).

You must notice that assignment operator is (=) and there is a relational operator, for equivalent (==). These two are different from each other, the assignment operator assigns the value to any variable, whereas equivalent operator is used to compare values, like in if-else conditions, *Example*

int x = 10; //assignment operator

x=5; // again assignment operator

if(x == 5) // here we have used equivalent relational operator, for comparison

{

cout <<"Successfully compared";

}

#### Logical Operators

The logical operators are AND (&&) and OR (||). They are used to combine two different expressions together.

If two statement are connected using AND operator, the validity of both statements will be considered, but if they are connected using OR operator, then either one of them must be valid. These operators are mostly used in loops (especially while loop) and in Decision making.

#### Bitwise Operators

There are used to change individual bits into a number. They work with only integral data types like char,int and long and not with floating point values.

* Bitwise AND operators &
* Bitwise OR operator |
* And bitwise XOR operator ^
* And, bitwise NOT operator ~

They can be used as shorthand notation too, & = , |= , ^= , ~= etc.

#### Shift Operators

Shift Operators are used to shift Bits of any variable. It is of three types,

1. Left Shift Operator <<
2. Right Shift Operator >>
3. Unsigned Right Shift Operator >>>

#### Unary Operators

These are the operators which work on only one operand. There are many unary operators, but increment++ and decrement -- operators are most used.

**Other Unary Operators :** address of &, dereference \*, **new** and **delete**, bitwise not ~, logical not !, unary minus - and unary plus +.

#### Ternary Operator

The ternary if-else ? : is an operator which has three operands.

int a = 10;

a > 5 ? cout << "true" : cout << "false"

#### Comma Operator

This is used to separate variable names and to separate expressions. In case of expressions, the value of last expression is produced and used.

*Example* :

int a,b,c; // variables declaration using comma operator

a=b++, c++; // a = c++ will be done.

### sizeof operator in C++

sizeOf is also an operator not a function, it is used to get information about the amount of memory allocated for data types & Objects. It can be used to get size of user defined data types too.

sizeOf operator can be used with and without parentheses. If you apply it to a variable you can use it without parentheses.

cout << sizeOf(double); //Will print size of double

int x = 2;

int i = sizeOf x;

### typedef Operator

**typedef** is a keyword used in C language to assign alternative names to existing types. Its mostly used with user defined data types, when names of data types get slightly complicated. Following is the general syntax for using typedef,

**typedef** *existing\_name* *alias\_name*

Lets take an example and see how typedef actually works.

typedef unsigned long ulong;

The above statement define a term **ulong** for an unsigned long type. Now this **ulong** identifier can be used to define unsigned long type variables.

ulong i, j ;

#### typedef and Pointers

typedef can be used to give an alias name to pointers also. Here we have a case in which use of typedef is beneficial during pointer declaration.

In Pointers \* binds to the right and not the left.

int\* x, y ;

By this declaration statement, we are actually declaring **x** as a pointer of type int, whereas **y** will be declared as a plain integer.

typedef int\* **IntPtr** ;

**IntPtr** x, y, z;

But if we use **typedef** like in above example, we can declare any number of pointers in a single statement.