C++ Operators

Operators are used to perform operations on variables and values.

C++ divides the operators into the following groups:

- **Arithmetic operators**
- **Assignment operators**
- **Comparison operators**
- **Logical operators**
- **Bitwise operators**

Arithmetic Operators

Arithmetic operators are used to perform common mathematical operations.

Operator	Name	Description	Example
+	Addition	Adds together two values	x + y
-	Subtraction	Subtracts one value from another	x - y
*	Multiplication	Multiplies two values	x * y
/	Division	Divides one value by another	x / y
%	Modulus	Returns the division remainder	x % y
++	Increment	Increases the value of a variable by 1	++x
--	Decrement	Decreases the value of a variable by 1	--x

Example of Arithmetic Operators

```
#include <iostream>
using namespace std;
int main(){
  int x = 240;
  int y = 40;
  cout<<"x + y: "<<(x + y)<<endl;
  cout<<"x – y: "<<(x - y)<<endl;
  cout<<"x * y: "<<(x * y)<<endl;
  cout<<"x / y: "<<(x / y)<<endl;
  cout<<"x % y: "<<(x % y)<<endl;
  return 0;
}
```

Output:

```
x + y: 280
x - y: 200
x * y: 9600
x / y: 6
x % y: 0
```

Example of Increment and decrement operators

```
#include <iostream>
using namespace std;
int main(){
  int x = 240;
  int y = 40;
   x++; y--;
  cout<<"x++ is: "<<x<<endl;
  cout<<"y-- is: "<<y;
  return 0;
}
```

Output:

```
x++ is: 241
y-- is: 39
```

C++ Assignment Operators

Assignment operators are used to assign values to variables.

In the example below, we use the **assignment** operator (=) to assign the value **5** to a variable called **x**:

A list of all assignment operators:

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
&=	x &= 3	x = x & 3
|=	x |= 3	x = x | 3
^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3
<<=	x <<= 3	x = x << 3

Operator	Description
=	Simple assignment operator, Assigns values from right side operands to left side operand.
+=	Add AND assignment operator, It adds right operand to the left operand and assign the result to left operand.
-=	Subtract AND assignment operator, It subtracts right operand from the left operand and assign the result to left operand.
*=	Multiply AND assignment operator, It multiplies right operand with the left operand and assign the result to left operand.
/=	Divide AND assignment operator, It divides left operand with the right operand and assign the result to left operand.
%=	Modulus AND assignment operator, It takes modulus using two operands and assign the result to left operand.
<<=	Left shift AND assignment operator.
>>=	Right shift AND assignment operator.
&=	Bitwise AND assignment operator.
^=	Bitwise exclusive OR and assignment operator.
|=	Bitwise inclusive OR and assignment operator.

Example of Assignment Operators

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

int main() {
  int x = 5;
  x += 3;
  cout << x;
return 0;
}
```

Output:

```
8
```

Example

```
#include <iostream>

using namespace std;

int main() {

  int x = 5;

  x -= 3;

  cout << x;

  return 0;

}
```

Output

```
2
```

Example

```
#include <iostream>

using namespace std;
```

```
int main() {
  int x = 5;
  x *= 3;
  cout << x;
  return 0;
}
```

Output

15

Example

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

int main() {
  int x = 5;
  x %= 3;
  cout << x;
  return 0;
}
```

Output

2

Example

```
int main() {
```

```
        int x = 20;
        int y = 10;
        int result = x%y;
        cout<<"My result is:" <<result;
        return 0;
}

Output
0
```

Example

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

int main() {
  int x = 5;
  x &= 3;
  cout << x;
  return 0;
}

Output
1
```

Example

```
#include <iostream>

using namespace std;


int main() {

  int x = 5;

  x |= 3;

  cout << x;

  return 0;

}


Output

7
```

Example

```
#include <iostream>

using namespace std;


int main() {

  int x = 5;

  x ^= 3;

  cout << x;

  return 0;

}


Output

6
```

Example

```
#include <iostream>

using namespace std;


int main() {

  int x = 5;

  x >>= 3;

  cout << x;

  return 0;

}


Output

0
```

Example

```
#include <iostream>

using namespace std;


int main() {

  int x = 32;

  x >>= 3;

  cout << x;

  return 0;

}


Output

4
```

Example

```
#include <iostream>

using namespace std;


int main() {

  int x = 5;

  x <<= 3;

  cout << x;

  return 0;

}
```

Output

40

Example

```
#include <iostream>

using namespace std;


int main() {

  int x = 10;

  x <<= 4;

  cout << x;

  return 0;

}
```

Output

160

C++ Comparison Operators

Comparison operators are used to compare two values.

Note: The return value of a comparison is either true (1) or false (0).

A list of all comparison operators:

Operator	Name	Example
==	Equal to	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

Example

```
#include <iostream>

using namespace std;


int main() {

  int x = 5;

  int y = 3;

  cout << (x == y); // returns 0 (false) because 5 is not equal to 3
```

```
  return 0;
}


Output
0
```

Example

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


int main() {
  int x = 5;
  int y = 3;
  cout << (x != y); // returns 1 (true) because 5 is not equal to 3
  return 0;
}


Output
1
```

Example

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


int main() {
  int x = 5;
```

```
  int y = 3;
  cout << (x > y); // returns 1 (true) because 5 is greater than 3
  return 0;
}

Output
1
```

Example

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

int main() {
  int x = 5;
  int y = 3;
  cout << (x < y); // returns 0 (false) because 5 is not less than 3
  return 0;
}

Output
0
```

Example

```
#include <iostream>

using namespace std;


int main() {

  int x = 5;

  int y = 3;

  cout << (x >= y); // returns 1 (true) because five is greater than, or equal, to 3

  return 0;

}

Output

1
```

Example

```
#include <iostream>

using namespace std;


int main() {

  int x = 5;

  int y = 3;

  cout << (x <= y); // returns 0 (false) because 5 is neither less than or equal to 3

  return 0;

}


Output

0
```

C++ Logical Operators

Logical operators are used to determine the logic between variables or values:

Operator	Name	Description	Example
&&	Logical and	Returns true if both statements are true	x < 5 && x < 10
||	Logical or	Returns true if one of the statements is true	x < 5 || x < 4
!	Logical not	Reverse the result, returns false if the result is true	!(x < 5 && x < 10)

Example

```
#include <iostream>

using namespace std;


int main() {

  int x = 5;

  int y = 3;

  cout << (x > 3 && x < 10); // returns true (1) because 5 is greater than 3 AND 5 is less than
10

  return 0;

}


Output

1
```

Example

```
#include <iostream>

using namespace std;


int main() {

  int x = 5;

  int y = 3;

  cout << (x > 3 || x < 4); // returns true (1) because one of the conditions are true (5 is greater than 3, but 5 is not less than 4)

  return 0;

}

Output

1
```

Example

```
#include <iostream>

using namespace std;


int main() {

  int x = 5;

  int y = 3;

  cout << (!(x > 3 && x < 10)); // returns false (0) because ! (not) is used to reverse the result

  return 0;

}

Output

0
```