



Number System



Types of Number System

Conversion of Binary to Decimal

Conversion of Decimal to Binary/Octal/Hexadecimal

Conversion of Any number System to other Number System

Bit manipulation

What is bit manipulation?

AND operator

OR operator

OR operator is used to perform logical OR operation on two or more operands.

It is denoted by the symbol `&&` in C++.

It returns true if at least one of the operands is true, otherwise it returns false.

For example, `5 > 3 && 10 < 20` will return true because both conditions are true.

Similarly, `5 > 3 && 10 > 20` will return false because the second condition is false.

Here is a truth table for the OR operator:

A	B	A && B
True	True	True
True	False	False
False	True	False
False	False	False

As you can see, the OR operator returns true only when both operands are true.

Here is a C++ program that demonstrates the use of the OR operator:

```
#include <iostream>
using namespace std;
int main()
{
    int a = 5, b = 10, c = 20;
    if (a > 3 && b < 20)
        cout << "Both conditions are true." << endl;
    if (a > 3 && b > 20)
        cout << "Both conditions are false." << endl;
    return 0;
}
```

The output of the program will be:

```
Both conditions are true.
Both conditions are false.
```

As you can see, the OR operator works as expected.

Here is another example of the OR operator:

```
#include <iostream>
using namespace std;
int main()
{
    int a = 5, b = 10, c = 20;
    if (a > 3 || b < 20)
        cout << "At least one condition is true." << endl;
    if (a > 3 || b > 20)
        cout << "At least one condition is false." << endl;
    return 0;
}
```

The output of the program will be:

```
At least one condition is true.
At least one condition is false.
```

As you can see, the OR operator works as expected.

Here is a final example of the OR operator:

```
#include <iostream>
using namespace std;
int main()
{
    int a = 5, b = 10, c = 20;
    if (a > 3 && b < 20 || a < 3 && b > 20)
        cout << "Both conditions are true or both are false." << endl;
    if (a > 3 && b > 20 || a < 3 && b < 20)
        cout << "Both conditions are false or both are true." << endl;
    return 0;
}
```

The output of the program will be:

```
Both conditions are true or both are false.
Both conditions are false or both are true.
```

As you can see, the OR operator works as expected.

Here is a final example of the OR operator:

```
#include <iostream>
using namespace std;
int main()
{
    int a = 5, b = 10, c = 20;
    if (a > 3 && b < 20 || a < 3 || b > 20)
        cout << "At least one condition is true." << endl;
    if (a > 3 && b > 20 || a < 3 || b < 20)
        cout << "At least one condition is false." << endl;
    return 0;
}
```

The output of the program will be:

```
At least one condition is true.
At least one condition is false.
```

NOT operator

XOR operator

Left shift operator

Right shift operator

Right shift operator shifts the bits of a number to the right by a specified number of positions.

It is denoted by the symbol `>>` in C++.

For example, if we have a number 10 (binary 1010) and we shift it right by 2 positions, we get 2 (binary 10).

The right shift operator is used to divide a number by 2 for each shift position.

For example, if we have a number 10 (binary 1010) and we shift it right by 2 positions, we get 2 (binary 10).

The right shift operator is used to divide a number by 2 for each shift position.

For example, if we have a number 10 (binary 1010) and we shift it right by 2 positions, we get 2 (binary 10).

The right shift operator is used to divide a number by 2 for each shift position.

For example, if we have a number 10 (binary 1010) and we shift it right by 2 positions, we get 2 (binary 10).

The right shift operator is used to divide a number by 2 for each shift position.

For example, if we have a number 10 (binary 1010) and we shift it right by 2 positions, we get 2 (binary 10).

The right shift operator is used to divide a number by 2 for each shift position.

For example, if we have a number 10 (binary 1010) and we shift it right by 2 positions, we get 2 (binary 10).

The right shift operator is used to divide a number by 2 for each shift position.

For example, if we have a number 10 (binary 1010) and we shift it right by 2 positions, we get 2 (binary 10).

The right shift operator is used to divide a number by 2 for each shift position.

For example, if we have a number 10 (binary 1010) and we shift it right by 2 positions, we get 2 (binary 10).

The right shift operator is used to divide a number by 2 for each shift position.

For example, if we have a number 10 (binary 1010) and we shift it right by 2 positions, we get 2 (binary 10).

The right shift operator is used to divide a number by 2 for each shift position.

For example, if we have a number 10 (binary 1010) and we shift it right by 2 positions, we get 2 (binary 10).

The right shift operator is used to divide a number by 2 for each shift position.

Next class teaser

- Introduction to Recursion
- Factorial and Fibonacci problems using Recursion
- Power of number using bits
- Count number of stairs interview problem



▶ THANK YOU ◀