Lesson:

Bit-wise Operator

PW SKILLS

Topics

- Introduction
- Bitwise AND
- Bitwise OR (|)
- Bitwise NOT (~)
- Bitwise XOR (^)
- Bitwise Left Shift (<<) and Right Shift (>>)

Introduction

Bitwise operators are used in programming to perform operations on individual bits of binary numbers. These operators are commonly used in tasks that require low-level manipulation of data, such as working with hardware, optimizing code, and cryptography. In this documentation, we will cover the most commonly used bitwise operators in programming.

Bitwise AND (&)

The bitwise AND operator (`&`) compares each bit of two integers and returns a new integer where each bit is set to 1 only if both corresponding bits in the original integers are 1. Otherwise, it sets the bit to 0.

Syntax

```
result = operand1 & operand2;
```

Example

```
const a = 12;    // binary: 1100
const b = 25;    // binary: 11001

const result = a & b;  // binary result: 1000 (decimal 8)
```

Bitwise OR (|)

The bitwise OR operator **(|)** compares each bit of two integers and returns a new integer. In the resulting integer, each bit is set to 1 if at least one of the corresponding bits in the original integers is 1.

Syntax

```
result = operand1 | operand2;
```

Example

```
const a = 12;    // binary: 1100
const b = 25;    // binary: 11001

const result = a | b;  // binary result: 11101 (decimal 29)
```

Bitwise XOR (^)

The bitwise XOR operator (`^`) compares each bit of two integers and returns a new integer where each bit is set to 1 if only one of the corresponding bits in the original integers is 1.

Syntax

```
result = operand1 ^ operand2;
```

Example

```
const a = 12;    // binary: 1100
const b = 25;    // binary: 11001

const result = a ^ b;  // binary result: 10101 (decimal 21)
```

Bitwise NOT (~)

The bitwise NOT operator (`~`) is a unary operator that flips the bits of an integer, changing 1s to 0s and 0s to 1s.

Syntax

```
result = ~operand;
```

Example

```
const a = 12;    // binary: 1100

const result = ~a;   // binary result: 11110011 (decimal -13)
```

Bitwise Left Shift (<<) and Right Shift (>>)

Bitwise left shift (`<<`) and right shift (`>>`) operators are used to shift the bits of an integer to the left or right by a specified number of positions.

Syntax

```
result = operand << num_bits;  // Left shift
result = operand >> num_bits;  // Right shift
```

Example

```
const a = 12;    // binary: 1100

const result1 = a << 2;  // Left shift by 2: binary result:
110000 (decimal 48)
const result2 = a >> 1;  // Right shift by 1: binary result: 110
(decimal 6)
```

Conclusion

Bitwise operators are essential tools for performing low-level bit manipulation in programming. They are commonly used in scenarios where you need to optimize code, work with binary data, or interface with hardware. Understanding how these operators work and when to use them is an important skill for any programmer.