Bitwise Operations

Bitwise operations are operations that directly manipulate the bits of a number. Bitwise operations are useful for various purposes, such as optimizing algorithms, performing certain calculations, and manipulating memory in lower-level programming languages like C and C++.

Here is a quick summary of common bitwise operations in C++:

Bitwise AND (`&`)

The bitwise AND operation (`&`) is a binary operation that takes two numbers, compares them bit by bit, and returns a new number where each bit is set (1) if the corresponding bits in both input numbers are set (1); otherwise, the bit is unset (0).

Example:

```
int result = 5 & 3; // result will be 1 (0000 0101 & 0000 0011 = 0000 0001)
```

Bitwise OR (`|`)

The bitwise OR operation (`|`) is a binary operation that takes two numbers, compares them bit by bit, and returns a new number where each bit is set (1) if at least one of the corresponding bits in either input number is set (1); otherwise, the bit is unset (0).

Example:

```
int result = 5 | 3; // result will be 7 (0000 0101 | 0000 0011 = 0000 0111)
```

Bitwise XOR (`^`)

The bitwise XOR (exclusive OR) operation (`^`) is a binary operation that takes two numbers, compares them bit by bit, and returns a new number where each bit is set

(1) if the corresponding bits in the input numbers are different; otherwise, the bit is unset (0).

Example:

```
int result = 5 ^ 3; // result will be 6 (0000 0101 ^ 0000 0011 = 0000 0110)
```

Bitwise NOT (`~`)

The bitwise NOT operation (`~`) is a unary operation that takes a single number, and returns a new number where each bit is inverted (1 becomes 0, and 0 becomes 1).

Example:

```
int result = ~5; // result will be -6 (1111 1010)
```

Bitwise Left Shift (`<<`)

The bitwise left shift operation (`<<`) is a binary operation that takes two numbers, a value and a shift amount, and returns a new number by shifting the bits of the value to the left by the specified shift amount. The vacated bits are filled with zeros.

Example:

```
int result = 5 << 1; // result will be 10 (0000 0101 << 1 = 0000 1010)
```

Bitwise Right Shift (`>>`)

The bitwise right shift operation (`>>`) is a binary operation that takes two numbers, a value and a shift amount, and returns a new number by shifting the bits of the value to the right by the specified shift amount. The vacated bits are filled with zeros or sign bit depending on the input value being signed or unsigned.

Example:

```
int result = 5 >> 1; // result will be 2 (0000 0101 >> 1 = 0000 0010)
```

These were the most common bitwise operations in C++. Remember to use them carefully and understand their behavior when applied to specific data types and scenarios.

Learn more from the following resources: