Bit shift operator

By Lups

Introduction

If you ever programmed in your life, you probably used operators such as `&, ^, |` and others. But have you ever seen the Bit Shift Operators?

The Bit Shift operators are represented by `<<`, `>>>` or `>>`, and they are divided in 2 different types, the left shift and the right shift. But what exactly is this shift? The bit shift operators shift a bit either to the right or to the left. Let's take a look at an example:

let's suppose I have a integer that it's value is equals to `0010`, that's equivalent to `2`. But what happens if I apply this statement:

```
0010 << 1
```

A single left shift would happen. What that means? It means that the most significant bit is shifted to the left alongside with the others and a 0 is inserted on the right. Example: 0101 << **2** = 0101**00**

`0010` will now become `0100`, and the `2` will become a `4`. Let's resume this:

```
0010 << 1 -> 0100
  0010 is equal to 2
  but, when a single left shift is apllied
  0100 is equal to 4
```

I hope you undestood the concept of left shift, cause we are moving onto right shift now. There are two types of right shifts, the **Logical Right Shift** and the **Arithmetic Right Shift**. We will began with the Logical Right Shift, so, the Logical Right Shift is when the least-significat bit is lost and a 0 is inserted on the other end. For example:

```
  0101 >>> 1 -> 0010
  0101 is equal to 5
  0010 is now equal to 2
```

For positive numbers, a single logical right shift divides the number by two, and throws it's remainders out. Now... Let's move on to the Arithmetic Right Shift (I'm going to abreviate to ARS), the ARS works by making the least-significant bit to get lost and the most significant bit is copied. For example:

```
        1011 >> 1 -> 1101
        1011 is now equal to -5
        1101 is now equal to -3
```

Bit Shift Operators in C++

Now that you understood all the theory behind the ***Bit Shift Operators***, let's take a look at a C++ example.

```
bit1 = 0x00AA;
  bit2 = 0x5500;

  z = (x << 8) + (y >> 8);
```

So, here x is shifted 8 positions to the left and y is shifted 8 positions to the right. The value of z is 0xAA55 (sum of x + y after shift).

Let's see just one more example:

```
#include <iostream>
  #include <bitset>

  using namespace std;

  int main() {
          unsigned short short1 = 1024;
          bitset<16> bitset1{short1};
          cout << bitset1 << endl;

          unsigned short short2 = short1 >> 1; // 512
          bitset<16> bitset2{short2};
          cout << bitset2 << endl;

          unsigned short short3 = short1 >> 10; // 1
          bitset<16> bitset3{short3};
          cout << bitset3 << endl;
          return 0;
  }
```

In the code above, we put in practice the Right Shift. The output is this:

```
  0000010000000000
  0000001000000000
  0000000000000001
```

See more

[Bitwise Operators]