

Re: Comparing with 0.

Source: <http://coding.derkeiler.com/Archive/C/ CPP/alt.comp.lang.learn.c-cpp/2005-02/0698.html>

From: Val (valmont_programming_at_hotmail.com)

Date: 02/21/05

Date: Mon, 21 Feb 2005 15:57:15 +0100

- > *If you want to compare whether a set of bits*
- > *within an unsigned integer are zero, then*
- > *look up "masking" in your text book.*

Not "unsigned", but signed int :)

And yes, I did read alright. Understanding of the operators one by one is not the problem.

- > *The steps are:*
- > *1. Create a "mask" containing the bit values*
- > *for each bit you want to test.*
- > *Example: 0xFF represents the "lowest" 8 bits.*
- >
- > *2. Arithmetic AND the unsigned integer with the*
- > *mask:*
- > *result = value & 0xFF;*
- >
- > *3. Test for zero.*
- > *if ((value & 0xff) == 0) /*... */*

This can't be right: Not even for unsigned. (compilable code below). "void binary()" from B. Stroustrup by the way.

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

```
using namespace std;
```

```
void binary (int i )
{
    // assume 8 bit byte
    bitset<8*sizeof(int)> b = i ;
    cout << b << endl;
}
```

```
int main(int argc, char *argv[])
```

```
{  
  int val = 256;  
  binary(val);  
  binary(val & 0xff);  
  cin.get();  
  return 0;  
}
```

So there's a more sophisticated algo needed. A one that also takes INT_MIN into account, since that is only zero's with the right most bit set to 1.

But it is not a "0".

The challenge is only to use "~ & ^ | + << >>".

NO "==" . :)

You see the problem that I have now?

That's why I asked where "==" was defined, hoping to get a good hint on how to do it :)