Bitwise Operators

By:

Dr. Hakimzadeh
Computer Science and Informatics
IU South Bend
How is data represented in a computer?

- Using an Encoding!
How is Information Represented?

- **Encoding** is the process of transforming information from one format into another. The opposite operation is called **decoding**. This is often used in many digital devices. (http://en.wikipedia.org/wiki/Encoding)

Chinese Calligraphy

UPC (Universal Product Code)
How is Data represented/stored inside the Computer?

- Bit (Binary digit)
  - (A bit is a unit of storage in a computer)
  - (A bit is a single binary digit. 0 or 1)
How is Data represented?

- Encoding
How is Data represented?

- Encoding

http://www.unicode.org/charts/PDF/U2800.pdf
How is Data represented?

- Encoding

http://en.wikipedia.org/wiki/DNA
http://en.wikipedia.org/wiki/DNA_sequencing
How is Data represented?

- Encoding

http://en.wikipedia.org/wiki/Musical_notation
C++ Bitwise facilities

- C++ provide the following operators for bit manipulation.

  Bitwise AND &
  Bitwise OR |
  Bitwise NOT ~ (Complement)
  Bitwise XOR ^
  Shift Left <
  Shift Right >
Bitwise AND (&)

00101  00101  00101
11111  00010  00001
====  ==== ====
00101  00000  00001
Bitwise OR (|)

00101 | 00101 | 00101
11111 | 00010 | 00001
====  ====  ====
11111 | 00111 | 00101
Bitwise NOT (\( \sim \))

\[
\begin{array}{ccc}
00101 & 00010 & 00001 \\
\quad & \quad & \quad \\
11010 & 11101 & 11110 \\
\end{array}
\]
Bitwise XOR (^)

00101  00101  00101
11111  00010  00001
====  ====  ====
11010  00111  00100
Shift Left ($\ll$)

\[
X = 00101 \ll 2;
\]

\[
X = 10100
\]
Shift Right (>>)

\[ X = \text{00101} \quad >> \quad 2; \]

\[ X = \text{00001} \]
Understanding Data Encoding

```cpp
#include <iostream>
#include <string>
#include <iomanip>
using namespace std;

void main()
{

    unsigned char clearChar;
    cout << "====Understanding your Compiler and Datatypes====" << endl;
    cout << "Size of char:             " << sizeof(char) << endl;
    cout << "Size of unsigned char:    " << sizeof(unsigned char) << endl;
    cout << "Size of unicode widechar: " << sizeof(wchar_t) << endl;
    cout << "Size of int:              " << sizeof(int) << endl;
    cout << "Size of float:            " << sizeof(float) << endl;
    cout << "Size of double:           " << sizeof(double) << endl;

    getchar();
}
```

(c) Copyright 2007-2018, H. Hakimzadeh
cout << "============Displaying Information in Different Formats=============" << endl;
unsigned int aNum;
aNum = 65;
cout << "---------- aNum (int) ----------" << endl;
//std::cout.flags (  std::ios::showbase );
cout << "Char format(C++) = " << char(aNum) << endl;
cout << "Dec format(C++) = " << std::dec << (aNum) << endl;
cout << "Oct format(C++) = " << std::oct << (aNum) << endl;
cout << "HEX format(C++) = " << std::hex << (aNum) << endl;
cout << "BIN format(C++) = " << std::bitset<32>(aNum) << endl; // Need to #include <bitset>
Making our own binary print()

```cpp
string char_to_binary(unsigned char value)
{
    char theResult[128];
    unsigned char mask = 0x80;   // 0x80 = 1000 0000
    int i;

    for (i = 0; i < sizeof(value) * 8; i++) {
        if ((value & mask) == 0)
            theResult[i] = '0';
        else
            theResult[i] = '1';

        mask >>= 1;
    }

    theResult[i] = '\0';
    string result = theResult;
    return(result);
}
```
Making our own binary print()

```cpp
string int_to_binary(unsigned int value) {
    char theResult[128];
    unsigned int mask = 0x80000000;
    int i;

    for (i = 0; i < sizeof(value) * 8; i++) {
        if ((value & mask) == 0)
            theResult[i] = '0';
        else
            theResult[i] = '1';
        mask >>= 1;
    }
    theResult[i] = '\0';
    string result = theResult;
    return(result);
}
```
Bitwise OR operator

```cpp
cout << "==================== OR Operation=========================" << endl;

clearChar = 0x01;

cout << "0000 0001 | 00001100 = " << (clearChar | 0x0C ) << endl;
cout << "Hex: " << std::hex << (clearChar | 0x0C ) << endl;
cout << "Binary Result: " << char_to_binary(clearChar | 0x0C ) << endl;
```
cout << "=====================AND Operation========================" << endl;

clearChar = 0xff;

cout << "1111 1111 & 00001100 = " << (clearChar & 0x0C) << endl;
cout << "Hex: " << std::hex << (clearChar & 0x0C) << endl;
cout << "Binary Result: " << char_to_binary(clearChar & 0x0C) << endl;

(c) Copyright 2007, H. Hakimzadeh
Bitwise XOR operator

cout << "=====================XOR Operation========================" << endl;
clearChar = 0x09;
cout << "0000 1001 ^ 0000 1100 = " << char_to_binary(clearChar ^ 0x0C ) << endl;


cout << "=================Double XOR Operation=====================
Back to the original char: " << endl;
cout << char_to_binary(clearChar) << " ^ 0000 1100 ^ 0000 1100= " <<
char_to_binary(clearChar ^ 0x0C ^ 0x0C) << endl;

(c) Copyright 2007, H. Hakimzadeh
Bitwise Complement operator

cout << "============Complement Operation============" << endl;

clearChar = 0x09;

cout << "~0000 1001 = " << char_to_binary(~clearChar) << endl;
cout << "===============Understanding Bitwise Shift Operations====================" << endl;
unsigned char clearChar;
clearChar = 0x01;

clearChar = (clearChar << 2);
cout << "0x01 in Binary format = " << char_to_binary(clearChar) << endl;
clearChar = (clearChar >> 1);
cout << "Shift left 1 bit = " << char_to_binary(clearChar) << endl;
clearChar = (clearChar << 6);
cout << "Shift left 6 bits = " << char_to_binary(clearChar) << endl;