CSC 215

Math and Computer Science

SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

Bitwise Operators

- Work on each bit in the data type
- Allow us to change just a single bit or determine what a bit is
  - This is sometimes called bit twiddling
- Work on integer types

Survey

- Who has tried the following and not gotten what the expected.

```
if ( x > 10 & y < 20 )
            Or
if ( x > 190 | y < -20 )
            Or
x = y^3;
```

- It compiles and runs and doesn't give an error.
- It is doing operations on a bit level.

The Operators and Precedence

()	
~	bitwise negation or complement
<<, >>	shift left, shift right
&	bitwise and
^	bitwise exclusive or
|	bitwise or
<<=, >>=, &=, ^=, |=	bitwise shortcut assignments

SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

~ (tilde) – Bitwise Negation / Complement

- Unary operator
- Takes an integer expression and returns the 1's complement of it.
  - Converts the 1's to 0's and the 0's to 1's

Example:

```
int x = 83; // 1 byte works the same as 4 bytes (value 83)
            // pattern of 00000000 00000000 00000000 01010011
y = ~x;     // y is now -84
            // pattern of 11111111 11111111 11111111 10101100
```

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

<< – Left Shift Operator

- Binary operator
- The Operand on the left is the value to be shifted
- The Operand on the right is the number of places to shift
- Works the same on 1 byte or 8 bytes

Example:

```
// shift the bits in x 3 places to the left
// and store the result in shiftResult
shiftResult = x << 3;
```

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Example Explained

- For each place shifted, a zero will be brought in

```
char x = 10;
char shiftResult;
shiftResult = x << 3;
```

- Before shift: 0 0 0 0 1 0 1 0 (10)
- After shift: 0 1 0 1 0 0 0 0 (80)
- shiftResult contains the value 80;

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

>> – Right Shift Operator

- Binary operator
- The Operand on the left is the value to be shifted
- The Operand on the right is the number of places to shift
- Behaves differently on signed and unsigned types

Example:

```
// shift the bits in x 2 places to the right
// and store the result in shiftResult
shiftResult = x >> 2;
```

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Example Explained – Unsigned

- For each place shifted, a zero will be brought in

```
unsigned char x = 79;
unsigned char shiftResult;
shiftResult = x >> 2;
```

- Before shift: 0 1 0 0 1 1 1 1 (79)
- After shift: 0 0 0 1 0 0 1 1 (19)
- shiftResult contains the value 19;

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Example Explained – Unsigned

- For each place shifted, a zero will be brought

```
unsigned char x = 244;
unsigned char shiftResult;
shiftResult = x >> 2;
```

- Before shift: 1 1 1 1 0 1 0 0 (244)
- After shift: 0 0 1 1 1 1 0 1 (61)
- shiftResult contains the value 61;

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Example Explained – Signed positive

- For each place shifted, a zero will be brought in since it is positive

```
char x = 79;
char shiftResult;
shiftResult = x >> 2;
```

- Before shift: 0 1 0 0 1 1 1 1 (79)
- After shift: 0 0 0 1 0 0 1 1 (19)
- shiftResult contains the value 19;

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Example Explained – Signed Negative

- For each place shifted, a one will be brought in since it is negative

```
char x = -48;
char shiftResult;
shiftResult = x >> 2;
```

- Before shift: 1 1 0 1 0 0 0 0 (-48)
- After shift: 1 1 1 1 0 1 0 0 (-12)
- shiftResult contains the value -12;

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

& - Bitwise AND

- Binary operator with each operand being of integer type

  X & Y

- Uses the same logic as a logical AND (&&) (1 & 1 = 1 , 0 & ? = 0)
- Applied to every bit within the data type

Table

&(AND)	1	0
1	1	0
0	0	0

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

| - Bitwise OR

- Binary operator with each operand being of integer type

  X | Y

- Uses the same logic as a logical OR (||) (0 | 0 = 0 , 1 | ? = 1)
- Applied to every bit within the data type

Table

|(OR)	1	0
1	1	1
0	1	0

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

^ - Bitwise Exclusive OR

- Binary operator with each operand being of integer type

  X ^ Y

- To produce a 1, only 1 of the bits can contain a value of 1
- Applied to every bit within the data type

Table

^(XOR)	1	0
1	0	1
0	1	0

Setting bits in a number

unsigned char ch = 98; //01100010

Turn the 3 bit to a 1. Remember to start counting a zero

01100010

Set up a mask that has zeros in all spots but the one I wish to change

unsigned char mask = 0x08; // 00001000 (8)

ch = ch | mask;

```
ch      01100010
mask    00001000
ch      01101010
```

ch |= mask;

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Extracting a bit

unsigned char ch = 98; //01100010

Extract the sixth bit. Remember to start counting a zero

01100010

Set up a mask that has zeros in all spots but the one I wish to extract

unsigned char mask = 0x40; // 01000000 (64)

ch = ch & mask;

```
  01100010
& 01000000
  --------
  01000000
```

ch &= mask;

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Clearing a bit

unsigned char ch = 98; //01100010

Clear the sixth bit. Remember to start counting a zero

01100010

Set up a mask that has ones in all spots but the one I wish to extract

unsigned char mask = 0xBF; // 10111111 (191)

ch = ch & mask; 01100010 ch &= mask;

10111111

00100010

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Toggling bits

unsigned char ch = 98; //01100010

Complement the 4th and 5th bits.

01100010 // want 01010010

Set up a mask that has zeros in all spots but the ones I wish to toggle

unsigned char mask = 0x30; // 00110000 (48)

ch = ch ^ mask;

```
  01100010
  00110000
  01010010
```

ch ^= mask;

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Practical Examples

- VCU Image format 4 bit (16 values) grayscale image
- Each byte contains 2 grayscale pixel values
- aaaabbbb aaaa is one value, bbbb is another value
- To extract the 2 values, we need a mask of 00001111

```
unsigned char compressed, mask = 0x0F;
unsigned char pixel1, pixel2;
pixel1 = compressed & mask;
pixel2 = (compressed >> 4) & mask;
```

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Practical Examples

- Hardware Raid 5
- Use property of a^b^a = b
- Have a byte on different hard drives and a parity byte of all bytes exclusive or'd onto another drive.
- If one hard drive crashes, exclusive or the remaining bytes to recover the lost data

SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

Hardware Xor raid 5

- 4 drives
- drivea, driveb, drivec, drived
- Call drived the parity drive
- Drived = drivea ^ driveb ^ drivec
- Assume driveb crashes
  - lostdrive = drivea ^ drivec ^ drived;
  - lostdrive = drivea ^ drivec ^ drivea ^ driveb ^ drivec; // drived value
  - lostdrive = drivec ^ driveb ^ drivec; // cancel out drivea's
  - lostdrive = driveb; // cancel out driveb's

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Print Number Out in Binary

- cout << hex << x << endl; // print number in hex
- cout << oct << x << endl; // print number in octal
- cout << dec << x << endl; // print number in decimal

How do you print it in binary?

Print Number Out in Binary

Assume unsigned character (1 byte integer)

? ? ? ? ? ? ? ?	? Represent an unknown 1 or 0
1 0 0 0 0 0 0 0	set up mask with 1 in MSB spot (1 << 7)
? 0 0 0 0 0 0 0	do a bitwise and, if result is nonzero, print a 1, otherwise print a zero
0 1 0 0 0 0 0 0	shift mask 1 spot to the left and repeat
0 ? 0 0 0 0 0 0	(print a 1 or a zero)

Do this a total of 8 times.

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Exercise: Store a Date – do on board

- Day – values(1-31) only need 6 bits to represent 31
- Month – values (1-12) only need 4 bits to represent 12
- Year – values (0 – 3000) only need 12 bits to represent 3000

	Unused	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	M	M	M	M	D	D	D	D	D	D
Bit#	3 3 2 2 2 2 2 2 2 2 1 0 9 8 7 6 5 4 3 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

For the Integer Date

- int date = 0;
- int m = ?, d = ?, y = ?;

- date = y;
- date = date << 4;
- date = date | m;
- date = date << 6;
- date = date | d;

Your job is to extract them.

- The masks need
- To make sure nothing resides above the bits requested, create a mask of all zeros then 1s for the bits that represent each item.
  - Day – 6 bits total – 00000000 00000000 00000000 00111111
  - Month – 4 bits total – 00000000 00000000 00000000 00001111
  - Year – 12 bits total – 00000000 00000000 00001111 11111111

```
int daymask = 63;
int monthmask = 15;
int yearmask = 4095;
```

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY