Lecture 3
CS 137
Fall 2014
by Chantelle Gellert

Announcments

- Readings: 1,2,3,4,5,6
- Quiz September 18th in class about 10 minutes, includes expressions
- Assignment 1 available now, due Sept. 24th 8:59pm

GCD Reminder

```
#include <stdio.h>

int main(void){
        int a = 0;
        int b = 0;
        int temp = 0;
        scanf("%d", &a);
        scanf("%d", &b);

        /*assume a > 0 and b > 0 */

        while(b !=0){
                temp = b;
                b = a%b;
                a = temp;
        }

        printf("%d\n", a);

 return 0;
}
```

Greatest Common Division what variables look like

a -> 0
b -> 0
temp -> 0

a -> 806
b -> 388
temp -> 388

a mod b = 130
b -> 130
a -> 338

What is an int

Think about computer memory and how it has a table of bytes. There are 8 bits in a byte. Each bit is either 0 or 1.

How to convert binary into decimal:

00011010

$$= 0 * 2^6 + 0 * 2^5 + 1 * 2^4 + 1 * 2^4 + 1 * 2^3 + 0 * 2^2 + 1 * 2^1 + 0 * 2^0$$
$$= 0 + 0 + 16 + 8 + 0 + 2 + 0$$
$$= 26$$

01001001

$$= 2^6 + 2^3 + 2^0$$
$$= 64 + 8 + 1$$
$$= 73$$

byte ”is called” char (usually in c)

Computer memory

Computer memory is a ”table of bytes”. Think of latter starting at 0 that increases by 1 every step of the latter. Each byte has an address.

A int holds 31 bits to create a value however the last bit is used to represent the sign of the integer. 1 being negative and 0 representing positive.

In C usually

char - 8 bits, 1 byte, from -128 to 127 or (-2^7 to 2^7 -1)
int - 32 bits, 4 bytes from -2,147,483,648 to 2,147,483,647 or (-2^{31} to 2^{31} -1)
if you have all bits 0 then that equals 0

What to do with a int

What can you do with an int?
int a = 806;
int b = 338;
add a + b
subtract a - b
divide a/b // (806/338 = 2.3) however c prints only 2 beccause it rounds
mod a percent b //(806 mod 338 = 130)

Printing:

```
printf("%d\n", a - b); //468
printf("%d + %d = %d \n", a,b, a + b);
```

Compound assignment

```
a = a + 2;
a += 2;
a = a - 2;
a -= 2;
a = a * 2;
a *= 2;
a = a/2;
a /= 2;
```

In general we have: variable operator = expression

Increment/decrement

```
a = a + 1;
a += 1;
++a;
a++;
a = -1;
a -= 1;
--a;
a--;
```

Example variable start: a = 806 , b = 338

```
a += ++b;
```

++b means do a pre-increment. Another words you add 1 to b and then add b to a.
Result: a = 1145 b = 339

```
a += b++;
```

b++ means do a post-increment. Another words you add b to a and then add 1 to b.
Result: a = 1144 b = 339

```
a += --b;
```

–b pre-decrement. Another words you minus 1 to b and then add b to a.
Result: a = 1143 b = 337

```
a += b--;
```

b– post-decrement. Another words you add b to a and then minus 1 from b.
Result: a = 1144 b = 337

Complicated expressions

```
a = 3*b - ++c
a = ((3*b) - (++c))
a += b += c
a+= (b+= c)
```

Table of operator precedence and associativity on page 6 of the book.

Negative int values: two complements

Converting positive to negative:
1) Flip all the bits (one's complement)
2) add one

Example: with the number 3
000...00011 (3 represented in binary)
Step 1) 111...11100 (flip the bits, 1 goes to 0 and 0 goes to 1)
Step 2) 111...11101 (add 1)

Example: with the number -3
111...11101
Step 1) 000...00010
Step 2) 000...00011

Only one zero value
Example: with the number 0
000...000000
Step 1) 111...11111
Step 2) 000...00000
Magic!