CS 237 Meeting 2 — 9/10/12

Announcements

1. First lab on Tuesday.

Introduction to C

1. Many elements of C are very similar to Java.
   (a) Scalar Data types
   (b) Scalar operations
   (c) Basic control structures
2. C Program structure is very different from Java.
   - In Java, a program is a collection of classes and a class is a collection of method and instance variable declarations. For the most part, each class is described in a separate source file.
   - A C program is a collection of functions and global variable definitions.
     – Global variable declarations and local variable declarations look the same
       * There are no qualifiers like private or public.
       * All global variables are accessible to all functions
     – Function definitions look just like Java method definitions:

```
<return type> function-name(<parameter declarations>) {
          <list of statement and local decls>
}
```

     – Somewhere in the list of functions definitions, there must be a declaration of a main function
       * It should return an int (0 means the program ended happily)
       * It should take an int and an array of "strings" as parameters providing a count of the number of arguments and the arguments
     – To illustrate these facts, consider the following two simple programs:

```
#include <stdio.h>

int main( int argc, char * argv[] ) {
     printf( "Hello World\n" );
}
```

and

```
#include <stdio.h>

void sayIt() {
     printf( "Hello World\n" );
}

int main( int argc, char * argv[] ) {
    sayIt();
}
```

     – One interesting thing about the way the C compiler processes source code is that it sometimes gets upset if a method is used earlier in a file than where its definition appears.
       * If you define sayIt after main in the last example, the program no longer compiles cleanly.
     – This can be a serious issue when one tries to define two or more mutually recursive functions. One will always be used before it is defined.
     – To solve this C allows the program to *declare* a function before it is *define*d. A declaration looks like a function header with a semi-colon after it. For example, we could define sayIt after main by adding a declaration as shown below:

```
#include <stdio.h>

void sayIt();

int main( int argc, char * argv[] ) {
    sayIt();
}

void sayIt(); {
    printf( "Hello World\n" );
}
```

- Functions and variables are not grouped into classes as they are in Java.
- Function and variable definitions can be kept together in a single file or organized in some way into a collection of separate source files.
  * The source file names used end in .c.
- Even if there are several files, there is no notion that each file corresponds to something like a class. The program is just the collection of functions and variables found in all of its files.
- To illustrate this, we can divide our silly little program into two files:

  file1.c:

```
#include <stdio.h>

int main( int argc, char * argv[] ) {
    sayIt();
}
```

  file2.c:

```
#include <stdio.h>

void sayIt(); {
    printf( "Hello World\n" );
}
```

3. Simple Output

(a) For the output of fixed text, a method named printf takes the place of System.out.println (or or more accurately, System.out.print).

For example:

```
#include <stdio.h>

int main( int argc, char * argv[] ) {
      int c;
      for ( c = 0; c < 10; c++ ) {
            printf("Hello World\n" );
      }
}
```

(b) To produce output that depends on variable values, you provide the extra values as additional parameters to the printf method and include special "formal specifiers" in the first argument that are replaced with the values of the additional parameters.

For example:

```
#include <stdio.h>

int main( int argc, char * argv[] ) {
      int c;
      for ( c = 0; c < 10; c++ ) {
            printf("The square of %d is %d\n", c, c*c );
      }
}
```

(c) The format specifier used depends on the type of the value being displayed:

%d is for int values

%s is for strings

%f is for double

(d) There are lots more details to these specifiers.

4. Input using scanf

- Just as the library function printf provides the ability to produce output, scanf can be used to read input from the keyboard (or other sources).
- Using scanf requires learning a tiny bit about one of C's most interesting feature, pointers.
  - Consider the program

```
#include <stdio.h>

void swap( int x, int y ) {
     int temp;

     temp = x;
     x = y;
     y = temp;
}

int main( int argc, char * argv[] ) {
     int a, b;

     a = 10;
     b = 100;

     swap( a, b );

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

}
```

  - The output of this program will be "a = 10, b = 100" rather than "a = 100, b = 10" as it might appear.
  - The swap function does not work as intended because when we use variables like a and b as parameters to a function, C passes the values of the variables rather than the variables themselves. swap has access to the values, but cannot change the variables.
  - In C, the address-of operator, & can be used to get a value that refers to a variable itself rather than its value.
  - The scanf function depends on the user to use address-of to pass it variables and it assigns values from the input to these variables.
  - Format specifiers like those used in printf control how the input is interpreted.
  - For example

```
scanf( "%d", &someIntVariable );
```

    can be used to read a single integer value from the input stream.
  - Let us use this to modify our powers printer to take input specifying the range of values it covers.

5. Preprocessor Stuff

   (a) #includes

   (b) #defines