Structure of C Programs

9-5-2006

Opening Discussion

- Do you have any questions about the quiz?
- Did everyone get to enjoy their 3-day weekend?
- What did we talk about last class?
- Do you have any questions about the reading?

Floating Point Values

- Most modern machines have built in ability to handle fractional values with floating point calculations.
- Floating point can be contrasted to fixed point. The latter has a certain number of bits above and below a binary point (it's not a decimal point when you aren't working in decimal).
- Floating point is represented like scientific notation in binary. The main thing you should know about it is that these aren't truly real numbers. They aren't continuous and math with them is imprecise.

Testing Math Operations in C

- We can use C to test some of the things we have talked about. This is because C has ways to understand hexadecimal numbers and to print them out as well. We can also play tricks that I don't expect you to understand to print floating point values as hexidecimals.
- You book talks about printing integers by putting %d in format strings for printf. If you use %x you will get the value in hex. This and other things can be found in the man pages as well as in later chapters.

Pieces of C Programs

- Programs in most programming languages are made out of statements that are grouped in various different ways. C is no exception to this.
- The statements come in multiple forms, but can contain expressions that produce values.
- A compound statement is multiple statements inside of curly braces.
- Let's look at some of the code we wrote last class and pull apart the various statements and expressions. What are the operators and operands that we used?

Assignment

- In an imperative language, like C, one of the most significant operators is the assignment operator: =.
- This appears with an expression on the left that evaluates to a memory location (a variable for what we are doing now) and an expression of a “matching” type on the right.
- When a statement with an assignment is reached, the RHS is evaluated and its value is stored in the LHS.
- C also provides shortcuts to do binary operations and assignment in a compact form: +=, -=, *=, etc.
- Write a program to calculate the cost of gas for a trip in multiple steps.

++ and -- : Prefix vs. Postfix

- It turns out that adding and subtracting 1 are such common operations that they have their own operators in C.
- The ++ and -- operators can be used either in postfix or prefix (before or after the operand). The only difference is what the expression returns.
- When expressions change variables they are said to have side effects. This includes assignment and the various assignment operators as well as increment and decrement. Good programming form makes side effects explicit.
- Multiple alterations to a single variables in a statement produces undefined results.

Modulus

- C includes some operators you might not be familiar with. One of these is the modulus operator. It works on integers and returns the remainder after division. For example, num%2 would be 1 for odd numbers and 0 for even numbers.
- Modulus is a remarkably useful operator when doing math on computers.
- Let's put some code into our program so that we can enter our average speed and get the time it will take to make the trip in hours, minutes, and seconds.

Bitwise Operators

- There are some operators that appear in the front cover of the book that aren't covered in chapter 3, but that relate do our discussion of binary numbers and binary arithmetic.
- C has certain bitwise operators that allow you to play with the bits in numbers.
  - ~ : bitwise not or ones complement
  - << : left shift
  - >> : right shift
  - & : bitwise and
  - ^ : bitwise xor
  - | : bitwise or
- A common use of these is packing colors into an ARGB int.

Type Casting and Type Conversion

- One type of expression that might seem odd to you is the type casting operator. You use this when you want to force the type of an expression to be something.
- An example is when you want to divide two values and you have them stored in ints, but you want the fractional value.
- C also does implicit type conversions when you operate on two expressions with different types.

Questions on Chapter 3

- Do you have any questions on chapter 3 that weren't covered in class today?

Functions

- The primary focus of this class is on problem solving. It turns out the most important aspect in solving large problems is figuring out good ways to break them up.
- So far our code has all gone in a main function and it executes straight from the top to the bottom. Our ability to break things up is rather limited. Every time we want to do something we have to enter the commands to do it.
- Functions allow us to break our solutions into pieces and call on those pieces as needed. Not only does this break things up, it allows us to reuse common chunks of code.

Minute Essay

- What would the values of num1 and num2 be at the end of this code?
  - int num1=5,num2=3;
  - num1+=7;
  - num2=(num1++);
  - num1%=5;
- Remember that there is open lab in this room this afternoon. Also remember that assignment #1 is due on Thursday.