# Variables, Types, and I/O

8/29/2007

## Opening Discussion

- Let's look at solutions to the interclass problems from three people.
- How can command line be superior to GUIs?
- What is a bus?
- Will we learn how to make computer games?
- Why is C called C?
- What do my Saturn sims look like?

## Binary Numbers

- We like to use the decimal (base 10) number system, but numbers can be done in any positive integer base.
- Because of the simplicity in making the electronics, computers typically use a binary (base 2) system.
- In binary each digit is a power of two and the will have either a 0 or a 1 in it.

## Converting Decimal to Binary

- One way to convert binary to decimal is to find the largest power of two smaller than the number and subtract that out. That will be a one bit. Each power of two you skip is a zero bit.
- Many people prefer the “divide by two” method. I'll write it here as an algorithm.

```
while(n>0)
    if(n is odd) write a 1
    else write a 0
    Divide n by 2 throwing away fraction
```

- This second method is related to bit shifting which we will talk about next week.

## Hexadecimal and Octal Numbers

- Computers might work in binary, but writing numbers with only 1s and 0s can be a real pain.
- Hexadecimal (base 16) and octal (base 8) are common substitutes. They are closer to binary, but don't take nearly so many digits.
- Hex numbers have 0-9 and A-F. Octal contains only 0-7.
- Converting from binary to hex or octal is as simple as grouping together bit.
- Starting at the ones bit, group bits in groups of 4 for hex and groups of 3 for octal.

## Identifiers

- Many of the things you put in your C programs will need names. These names are called identifiers and they have certain rules.
- Identifiers can contain letters, numbers, and underscores. The first character can't be a number.
- C is case sensitive so IDENT and ident are two different identifiers in C.
- Identifiers beginning with an underscore are typically used by the system so we won't name anything that way.
- I will typically follow the “camel” naming scheme.

## Types

- One of my favorite aspects of programming languages is the typing systems in them.
- C has a fairly boring monomorphic typing system. That's a good place to start though because of its simplicity.
- Whenever I try to explain what types are the expression, “you're comparing apples to oranges” comes to mind. In that case apples and oranges are two classes of objects that you shouldn't mix up.
- Types specify what something can be used for. We will start off using the built in types of C. Later we will create our own types.

## Primitive Types

- Integer types
  - int
  - short int/short
  - long int/long
  - char
- Floating point types
  - float
  - double
- Integer types can optionally be specified as signed or unsigned. The default is signed.
- void is a type that you can't do anything with. It is often used as a place holder.
- Your book includes others. They aren't all valid unless you use the -std=c99 compile option.

## Variables

- In order to do anything significant in C we need to be able to declare names (these are identifiers) that can store values for us so that we can easily access them.
- These are called variables. You can picture a variable as being like a box that holds a value in it.
- We can declare variables in C at the beginning of any block of code (right after a {).
- A variable declaration is a statement that begins with the type of the variable and is followed by a variable name and an optional initialization.
- Multiple variables can be declared on a single line though some people frown on that practice.

## Output with printf

- We have seen printf, but we haven't really used the f part of it.
- Let's look at the man page for printf and see if we can figure out what it is saying.
- Let's play with printf some. We can also use the sizeof operator to learn a bit more about the primitive types on our system.

## Input with scanf

- The opposite of printf is scanf. Use this function when you want to read input from the user.
- Let's look at the man page for scanf.
- At this point, all the variables you pass to scanf will need to have & in front of them. This is because scanf needs to know where the variable is in memory and that is what & does. We'll go into more detail about this when we cover pointers.

## Minute Essay

- You should never use floating point numbers to represent money because many values you'd like to have for your money can't be perfectly represented in binary. For example, 1/10 is a fraction that can't be represented perfectly in binary with a finite number of bits. Can you think of an example fraction you can't write in decimal with a finite number of digits? What is it?
- Interclass Problem – Do problem 38 on page 90 of the text book.