Books available from lulu.com (if you want them); see links on course homepage.

Variables and Types:

Modern computer memory made out of bits (binary digits), which are On or Off. We think of these as 1 and 0.
Modern computers group the bits into groups of 8, called bytes. So you may have a 1 gigabyte main memory (RAM = random access memory) on you computer.

So you have lots of bytes into which you can store data of any sort (numbers, characters, music samples, pixels from video, etc.). You translate any kind of information into bits and therefore manipulate it on the computer.

Every memory location in the computer holds one byte of information.
Every memory location has an address (a number), and the computer can use these addresses to find out what is in a memory location, or to change it. Think of the addresses like P.O. box numbers.

But the computer has to know how to interpret the bits in a memory location. Should it interpret them as a number? as a character? or something else?

It would be very inconvenient (and very error-prone) to have to remember the numerical address of every location in the memory that you're using, and to remember what sort of data is stored in it.

Fortunately we don't have to do that. High-level languages like C++ allow us to give meaningful names to locations in the computer's memory and to state the type of data that will be stored in those locations. This also tells the compiler how much space in the memory to set aside, because different types of data take different amounts of space (different numbers of bytes).

For example:

char fred; // declares a character variable called "fred"

This tells the compiler to set aside enough space (1 byte) to hold one character, and to name that space "fred." So this is like putting on the mailbox a PostIt with "fred" on it, so you don't have to remember the address.

It also tells the compiler that the bits in location "fred" should be interpreted as a character, not as some other kind of data. For example, this tells the compiler how to print it out.

What about other data? If you declare:

int bob; // declares an integer variable called "bob"

An integer is a + or - number with no decimal point.
An "int" variable typically occupies 4 bytes, which limits the range of numbers that can be stored in an int variable.

For potentially fractional numbers and for very large and very small numbers, modern computers use "floating point." This is kind of like scientific notation, e.g., 6.02×10^{23}. The most common floating-point type in C++ is called "double" (typically 4 bytes).

double alice; // declares a double-size floating-point variable

Another type that we will use is "string", which can hold a variable number of characters.

How do you put something into a variable?
(1) There is already "something" in it, after it's declared, but it might not be what you want. So it's best to give it some initial value.

(2) To change its value you use an assignment statement:

```
bob = 2; // assigns the integer 2 to "bob"
```

(3) Don't read this "bob equals two"; read it "bob gets two," "bob becomes two," or "bob is assigned two," etc.

(4) It is a command to the computer to change the value stored in the memory location named "bob."

If you are counting things, you often need to do something like this:

```
int count;
count = 0; // initial value
…
    count = count + 1; // count one more
…
```

The assignment operator orders the computer to take the value on the right-hand side (RHS) and store it into the variable named on the LHS.

Example program: computing average grade.

Things to observe:

(1) We developed the program by stepwise refinement, first counting grades, then summing them, finally averaging them.

(2) We put in more output statements (cout <<) than we needed so that we could see what the program was doing as it ran.

(3) This program has several flaws. First, it will try to divide by zero if there are no grades to average. Second, there is no input validity checking (what is the range of legal grades?).

The program we developed in class is called average.cpp. Look also at the program called incorrect.cpp and see if you can see why it gives the wrong answer.