Class Notes from 2010-08-26.

Read for next time TCS ch. 3.

Variable

Every location in the computer's memory has a number, called its address. You can think of them like post office boxes.

However, we don't want to have to remember the numbers (addresses) for all of these locations, so a high-level language allows us to give them names. Named memory locations are called variables.

So we can ask the compiler to set aside space in memory for different kinds of data and give that space a name.

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

"char" (which is short for "character") means set aside space big enough to hold one character (1 byte).
"fred" is the name we will use to refer to this space in memory.

int N; // declares an integer variable called "N"
"int" sets aside space to hold one integer (a number without a decimal point).

The words "char" and "int" are examples of data types.
A data type determines how much space to set aside and what sort of operations you can do on the data.

The named memory locations "fred", "N", etc. are variables (so-called because I can change the data stored in them).

The main built-in data types we will be using in class:
bool - holds one bit (0/1, false/true)
char - holds one character (1 byte)
int - holds one number without a decimal point
double - holds one "floating-point" number (a real number, i.e., a number possibly with a decimal point).
string - which refers to a sequence or string of characters.

Something like "int N;" is called a declaration, and the rule in C++ is you must always define something before you use it.

The declaration just sets aside the space in memory. How do you put something in it? This is done with the assignment statement.

```
int N;
N = 3; // stores 3 in N
```

You should read this not as "N equals three," but as "N gets three" or "N becomes three" or "N is assigned 3" …

```
int M;
M = 2;
N = M + 2; // now N has the value 4
```

The sequence
int M;

M = 2;
can be abbreviated:
int M = 2; // declare M with initial value 2

Consider this example (supposing the preceding declaration and assignment to N):

N = N + 1; // N has the value 5 after this statement

We can perform operations (such as +, -, *, /) on many data types including integers and doubles.
Operations have a precedence, which means we do */ before +-.

For example 3 + N / 2 mean 3 + (N / 2)
For operators of the same precedence, we do them in order from left to right. For example N / M / 2 means (N / M) / 2.

When the compiler sees an operation, it figures out what the machine should do from the types of the operands. So on the machine, integer operations are different from floating point operations. If the types are both integers, it does integer arithmetic.
If the types are both double, it does double arithmetic.
If the types are mixed, it converts the integer to a double and does double arithmetic.
3 / 2 both operands are ints, so it does integer arithmetic and gives you the value 1.
Likewise 9/10 has the value 0.
3.0 / 2.0 both operands are doubles, and it does double arithmetic and return the double result 1.5.
3 / 2.0 one operand is int and one is double, so it converts 3 to 3.0 and gives 1.5,

the same result as 3 / double(2).

double X = 2;

Then 3 / X would give 1.5, because one of the operands is double.
3 / 2 * X
It will compute (3/2), giving 1, and multiply by 2.0 giving 2.0.

C++ is an object-oriented language.
You can divide the data in programs into values and objects (a fuzzy distinction). Values are things (such as numbers and characters) that you do operations on (such as +) to get other values. They belong to types.

Objects are things that have behaviors. You tell them to do things and they do them. They belong classes.

robot.forward(0.5, 2); // this tells the object called "robot" to move forward at half speed for 2 seconds.