Lesson 2: Variables, Types, and Expressions

3D Game Programming With C++

Digital Media Academy (Summer 2011)

Written by: Andrew Uzilov (andrew.uzilov@gmail.com)
Feel free to contact me with any questions.

A variable is named piece of memory that holds data. In this example from the previous lesson:

```
int damage = modifier * 35;  // what do you think this does?
```

the words `damage` and `modifier` are variables. Variables in C++ always have a type. Some commonly used C++ types are:

`int`	An integer, which is a round number (no decimal point) such as: -186, 0, 7, 419
`float`	A number with a decimal point, such as: -199.99, -0.0001, 12.0, 5692.34
`double`	Same as `float`, but can represent a lot more numbers with higher precision, although uses twice as much memory.
`bool`	Something that is either `true` or `false`
`char`	A single character – a symbol like `9` or `q` or `;` or `#` or even something unprintable
`string`	A bunch of characters in a certain order. Basically, a phrase of any length. Can even have zero characters in it (an "empty string").

Variables have to be declared before they can be used, which tells the compiler the variable's type and then its name, like this:

```
int health;
```

After that, you can define them (assign a value to them; sometimes called initializing). This puts the value "100" on the right into the piece of memory described by the variable "health" on the left:

```
health = 100;
```

You can declare and define a variable in the same statement:

```
int health = 100;
```

You can also copy a variable; this way, you can save the original value, and mark up the current one:

```
int origHealth = health;
/* now, feel free to change value of "health" – we just made
   a backup copy! */
```

After a variable was declared and defined, you can use it in statements, including to create other variables:

```
int newHealth = health - 15;  // took a hit
```

We can also change the value of the variable (it's called assigning a new value to a variable), like this:

```
int health = 100;
health = 50;   // change the value of health to 50
health -= 10;  // decrement by 10; health is now 40
health += 20;  // increment by 20; health is now 60
```

Caution: Never use a variable before you assign it a value!

You can convert variables of one type into another type. Normally, the compiler can do this implicitly, so you don't need to do extra typing:

```
double speed = 55.7;
double time  = 10.9;
int distance = speed * time;  // distance is 607
```

Variables can combined with other things and printed to the console. This is especially useful for debugging. Here is a basic program that stores your name in a variable of type `string`. When using string values, always put them in double quotes.

```
#include <iostream>
using namespace std;

int main() {
      string myName = "Andrew";
      cout << "My name is: " << myName << endl;
      return 0;
}
```

EXPRESSIONS

Expressions involve an operator. Expressions evaluate to a value (an answer to a question asked by the operator).

Some binary operators (they need two variables/values, one on each side):

`+` `-` `*` `/` `%` `==` `!=` `&&` `||` `<<` `>>`

Some unary operators (they need just one variable/value):

`-` `!` `++` `--`

Usually, you can combine a bunch of expressions together. When it doubt about operator precedence, use parentheses! (Remember the expression "Please Excuse My Dear Aunt Sally" ?)

```
double tempC = 13.5;
double tempF = (9.0 / 5.0) * tempC + 32.0;
```

With unary operators `++` and `--`, you can change the value of a variable in-place, but it is tricky – use with caution! Try these code examples in Eclipse and use `cout` to print the value of the variable `position`. What do you get in each case?

```
int timer = 0;
int position = 5 * timer++;  // what do you think "position" holds?
```

versus

```
int timer = 0;
int position = 5 * ++timer;  // what about now?
```

Exercise 2.1: Your first variable

Start a new Eclipse project and name it: `Exercise_2_1`

Put in the following code from the example above:

```
#include <iostream>
using namespace std;

int main() {
      string myName = "Andrew";
      cout << "My name is: " << myName << endl;
      return 0;
}
```

Change this program to output your own name, but make sure to keep your name in the variable `myName`.

Exercise 2.2: Combining variables

Copy your Eclipse project from Exercise 2.1 to a new project like this:

- Click on the icon  in the bottom left corner and choose "Navigator"
- In the Navigator tab on the left side of the screen, right-click on your `Exercise_2_1` project and choose "Copy"
- Right-click anywhere in the Navigator tab and choose "Paste"
- In the resulting dialog, enter `Exercise_2_2` as the new project name
- Close the Navigator tab

Now you made a copy of the project! In this copy, add a new `int` variable called `myAge` and use it to print your name and age to the console on a single line.

Exercise 2.3: Writing expressions

Copy your program from Exercise 2.2 to a new project called: `Exercise_2_3`

Change this copy to output useful facts about you, such as:

- How old will you be in 10 years?
- How many years until you turn voting age (18 years)?
- How old are you when measured in decades? In centuries? (Hint: use the `double` type to compute these.)

Use an expression to compute each of those values, and save them into variables, then print to console using those variables.