Declaring a Variable

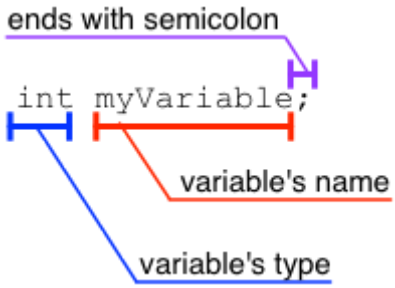
coursera.org/learn/programming-fundamentals/supplement/y7Yt3/declaring-a-variable

Variables

Programs track most of their state in *variables*—you can think of a variable as a box that stores a value. In order to use a variable, the programmer must declare it, specifying its type and name. The type specifies what kind of value can be held in a variable's box (for example, whether it is a number, a letter, or text). We will learn about types in the next module, but for now, we will use variables whose types are all **int**—meaning that the value in their box is a number.

Declaration

The name of a variable may be any valid *identifier*. An identifier is the formal programming term for a word that can be used to name something. In C, identifiers may be any length, and can contain letters, numbers, and underscores (_). They may only start with a letter or an underscore (not a number), and are case-sensitive (meaning that **abc** is different from **Abc** and **ABC** is different from both of them). The variable declaration ends with a semicolon—which is used to end many *statements* in C. A statement in a programming language is roughly analogous to a sentence in English—it is a complete line of code, which can be executed for an effect. This figure shows a variable declaration and identifies each of the pieces:



When executing code by hand, the effect of a variable declaration is to create a new box, labeled with the name of the variable. In C, a newly declared variable is *uninitialized*, meaning that its value is undefined. When the computer actually executes the program, it has a finite (but quite large) number of "boxes" (memory locations), and the variable will be given one that is currently not in use. The value of the variable will be whatever value happened to be in the location previously, until a new value is assigned to the variable (which we will see shortly). Correspondingly, when we execute a variable declaration by hand, we will draw a box and place a (?) in it for its value—indicating that it is unknown. If we ever use an unknown value as we execute our program, it indicates a problem with our program, since its behavior is undefined—its behavior will be changed based on whatever the value actually is, which we cannot predict.

The next video shows the execution of code containing two variable declarations—**x** and **y**. At the start, the execution arrow is at the beginning of the code. The area on the right—which represents the state of the program—is empty. As the execution arrow advances across these statements, we execute their effects: drawing a box for each variable, with a (?) in it, indicating that the variable is uninitialized.