Variable declarations

Before you can receive inputs, you must have a place to store them. In programming, these storage locations are called *variables*. The three basic types of variables are **char**, **int**, and **float**. Each variable you use must be declared. The declaration allocates space in memory. To make this allocation, C++ needs to know what data type you want to use.

char for single characters (1 byte each)
int for integers (2 bytes usually, 4 bytes on some computers)
float for real (floating point) numbers (0.0 and 10^{-38} to 10^{38})
double (higher precision "float"). Typically 15 digits of precision (0.0 and 10^{-308} to 10^{308})

You can declare several variables at a time (separated by a comma). Variables can be initialized at the time of declaration.

Example: **int** count = 0, status.

Valid variable names

- Use letters, digits, and _ (underscore)
- Begin with a letter
- Up to 32 characters in length
- Must avoid reserved words (**int, float, for, if, return**, etc.)
- Should avoid function names

Style of choosing names

- Easily understood (example: use **int** velocity, rather than **int** v)
- Use lowercase to distinguish from the predefined constants (example: **#define** HEIGHT 7).
- Use common sense to avoid confusion. (example: don't have both v1 ("one") and vl ("el") in the same program).

Entering values

There are four ways to enter values:

1. Constants
   ```
   #define PI 3.14159
   #define DISTANCE 350
   ```
2. Variable declarations
   ```
   double side_1 = 7.5;
   ```
3. Assignment statements
   ```
   Takeoff_height = 5;
   ```
4. Interactively, by user inputs via the keyboard (**cin** command will prompt the user for these values).

Output with cout
cout is a function in the **iostream** library. Example statement:
cout << ”Height at fence is “<< height_at_fence << “ feet.”<< **endl**;

endl generates an “end of line”. Subsequent text will be written on the *next* line.

Get in the habit of including **endl** in your **cout** statements; otherwise everything will run on the same line.

For example, the two statements:

cout<<”GO RED SOX”;
cout<<”Enter angle:”;

will print the following on the screen:

GO RED SOXEnter angle:

“Literal Strings”
Whatever is within double quotes will be printed literally, character by character. The literal string cannot extend over more than one line of code. If you can’t fit all you want on one line, end the quotes and restart on the next line:

cout << ”Height is 10 feet, and velocity ”;
cout << ”is 15 ft/sec”;

Input with cin
cin >> number;
cin sets the value of the variable number to whatever the user types into the program when prompted. Only one value can be read in after each set of insertion operators >>. You can request multiple variables be read in at once by using >> for each new variable:

Correct: **cin** >> x >> y;
INCORRECT: **cin** >> x, y; or **cin** >> x y;

Arithmetic and assignment operators

Operation	Symbol	Syntax
Addition	+	x + y
Subtraction	-	x - y
Multiplication	*	x * y
Division	/	x / y
Modulus	%	x % y
Negative	-	-x

Modulus = remainder after division. Example: 35 % 7 yields 0. 8 % 3 outputs 2.

= is the assignment operator. This is *different* from algebra. Evaluate the statement on the right, and assign it to the variable on the left.

Valid statements

```
sum = sum + x;
x = -y;
area = PI * radius * radius;
```

Invalid statements

```
x + x = 10          // two variables on left – ERROR
-x = y;             // operator on left - ERROR
8 = x;              // cannot assign a value to '8' - ERROR
```