Statements, Blocks, Lines & Symbols

In order to be as clear as possible when describing things, it's helpful to use a very specific and well defined set of terms.

The Appendix entry for this series of documents contains a reference defining key terms used throughout the class, but new ones will be introduced in most every section. Whenever possible the most common industry and/or academic terminology will be used, however many things may go by a variety of names; and likewise some terms may be used in a variety of ways. Wherever reasonable, an attempt will be made to note this in both the text and the lecture.

In addition to in-class communication, this focus on terminology is meant to help you extend your learning on your own. Knowing the correct terminology will help you use various search systems to find potential solutions and extend your working knowledge. It will likely save you some time in future classes as well.

Let's start with some of the simplest components common to most all programming languages. We've already looked at ***variables*** and their definitions. And as a part of that we also saw ***constants***. In order to actually do something with these variables and constants — say add some together, an ***operator*** will need to be employed. Just as in mathematics, there are symbolics for most common operators such as **+**, **-**, **/**, and ***** for addition, subtraction, division, and multiplication, respectively. There are also some basic comparison operators, primarily **>**, **<** and **==**, for greater than, less than, and equal to, respectively. Note that equal to is represented by **==** and not a single equal sign, this sets it apart from the assignment operator describe in the lesson on vaiables.

The next most basic element is the ***statement***. Although the exact definition of a statement may vary somewhat depending on the specific context in which its used, we will generally refer to a statement as a single command we want the computer to carry out. For example, adding two values together, comparing two values, and jumping to a different state are all examples of statements.

The variable definitions that were just introduced are, in some programming languages, also referred to as statements, and even in C++ they share some similar syntax. The primary difference is that that definitions don't actually _*do*_ anything. That difference will not be significant in this class, but it will be as you progress into more advanced programming.

Just like definitions, the syntax for statements implies that they be terminated with a semicolon. While it is possible to create several statements within a single line of text, it is generally discouraged, as it reduces readability, and can complicate finding errors, making future changes, etc.

A statement may contain one or more ***expressions***. For example the following are all valid statements:

```
foo = foo + 17;
foo = foo * 17 + bar * 9;
Serial.println( foo * foo );
```

A group of statements can be joined together form a ***statement block***. A block is defined used braces, also known as curly brackets. Although each individual statement must be terminated with a semicolon, a block does not need to be. Here's a sample statement block:

```
{
    foo = ( foo / 17 ) + bar * 9;
    Serial.println( foo * foo );
}
```

A block can be named and become a **function**. We will cover functions in detail soon. Blocks can be ***nested***, meaning that one or more blocks can be *inside* another block, as you will see in the experiment.

In the above examples you could also see the use of parenthesis. They are used in several ways. Just as in mathematics, they can be used to group components of an expression, typically to alter the order of operation. Parenthesis can also be nested as needed, consider the following statement:

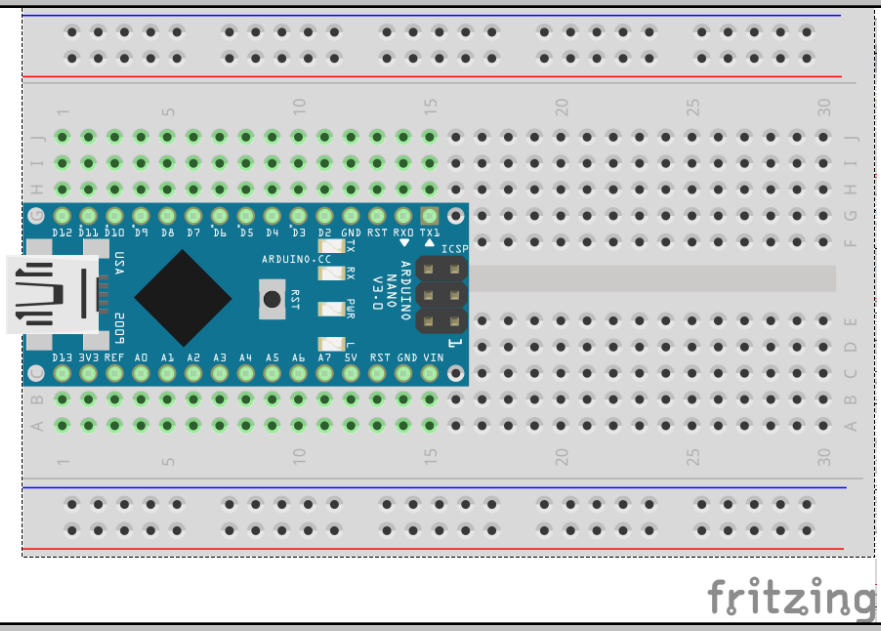
```
foo = ( ( foo + 4) * ( bar -1 ) ) / baz;
```

Here the addition and subtraction actions are preformed first, followed by the multiplication, and finally that result is used in the division operation and then the resulting value is assigned to the variable foo.

Additionally parenthesis are used with ***conditionals***, ***functions*** and when ***type casting,*** all things we will explore soon.

There are many more operators available and there is a very specific order of precedence that will be followed when they are used, just as in mathematics. Please take a moment to review the appendix section on operators and precedence, as several that do not have a common parallel in mathematics will use used in the future experiments.

Experiment: building blocks.

Components	Wiring Diagram
✓Computer ✓Arduino IDE	

Connection Instructions
No connections required, we will be using the onboard/internal LED for this example.

Sketch(es)	buildingBlocks.ino

Analysis Questions
Why do you think multiple blocks and nested blocks are required? Are there analogies in the real world? Do you think they will be confusing? If so, can you think of a way to make them less so?

Programming Tasks
Tasks are described in the comments of the program.