



### The Big Idea:

In algebra we all learned to deal with numbers in the abstract by using letters to represent numbers. The equation for a straight line, for example, is  $y = mx + c$ . In this equation we know that  $y$  is a value that is found by evaluating the expression  $mx + c$ . But all the letters— $y$ ,  $m$ ,  $x$ , and  $c$ —are simply names for placeholders of actual values. The equation expresses how they relate to each other.

These names are called **variables**. Variables to store text (called **Strings**) were first introduced in Lesson 2. The C language also uses variables to store numbers. In this unit you will learn to work with variables and to be aware of some of the subtle differences between how they are used in C and in algebra.

### Background:

Lesson 3 introduced variables, in particular variables of the type **String**. **String** variables are useful when the programming task is collecting, manipulating, and displaying text. But Arduino™ sketches also need the ability work with numbers. A sketch used to control a quad copter, for example, must sense the angle of the craft relative to the earth and use that information to adjust the speed of the motors. This cannot be done without mathematics, and mathematics requires numbers.

In addition to the **String** type for text, the C language provides several variable types to use with numbers, as shown in Table 5-1.

*Table 5-1. Variables used in lessons in this book*

Type	Description	Examples
<code>String</code>	A collection of characters; a type of data	Hello, world!
<code>int</code>	signed integer with no decimal point. Range is from -32767 to 32767	5, 18, -2776, 0, 83, -21822
<code>double</code>	Precise number with a decimal, useful for most mathematic purposes	98.6, 0.7, -236.99, 0.0, -5280.1
<code>float</code>	Number with a decimal, not as precise as a double but with a large range—from -3.4028235E+38 to 3.4028235E+38	6.02E+23, 3.7E+8, 7.0, -89.2234E18

The Arduino™ website reference for float numbers cautions that the lack of precision of float numbers may make for some strange behaviors when compared with other numbers.



Important

This lesson works only with integers. For most applications of the Arduino,™ integer math is all that is required.

Mathematics in the C language for the Arduino™ is similar, but not identical, to the math commonly seen in algebra.

*Table 5-2. Differences between algebraic math and math in C*

	Algebra	C Programming language
Equals sign (=)	<p>Indicates equality. That is, the expression on the left of the equals sign evaluates to the same result as the expression on the right.</p> <p>Examples:</p> $5 + 2 = 7$ $4 + 8 = 7 + 5$ $AB + AC = A(B + C)$	<p>Does not indicate equality. The symbol is called the assignment operator. The left side must be the name of a variable. The right is an arithmetic expression. The results of the evaluation of that expression are "assigned to" the variable. That is, the result is stored to the variable.</p> <p>Example: The following declares a variable named <code>myInt</code> then assigns the result of an arithmetic expression to that variable.</p> <pre>int myInt; myInt = 7 + 12;</pre> <p>The variable, <code>myInt</code>, has been assigned the value of 19.</p> <p>The following programming statement modifies this value.</p> <pre>myInt = myInt + 3;</pre> <p>The expression on the right retrieves the value assigned to <code>myInt</code>, adds 3 to it, then stores the result to <code>myInt</code>. The value of <code>myInt</code> is now 22.</p>

	Algebra	C Programming language
Division ( / )	One integer divided by another may yield a quotient that contains a decimal. For example: $15 / 4 = 3.75$	No decimals are allowed. Instead, the integer portion of the quotient is the only result, and it is not rounded.  <pre>int myInt; myInt = 15 / 4;</pre> <p>The value assigned to myInt is 3.</p>
Modulus ( % )	Frequently called the modulo operator, the modulus has several uses in mathematics.	Determines the remainder of integer division.  <pre>int myInt; myInt = 13 % 4;</pre> <p>The value assigned to myInt is 1.</p>

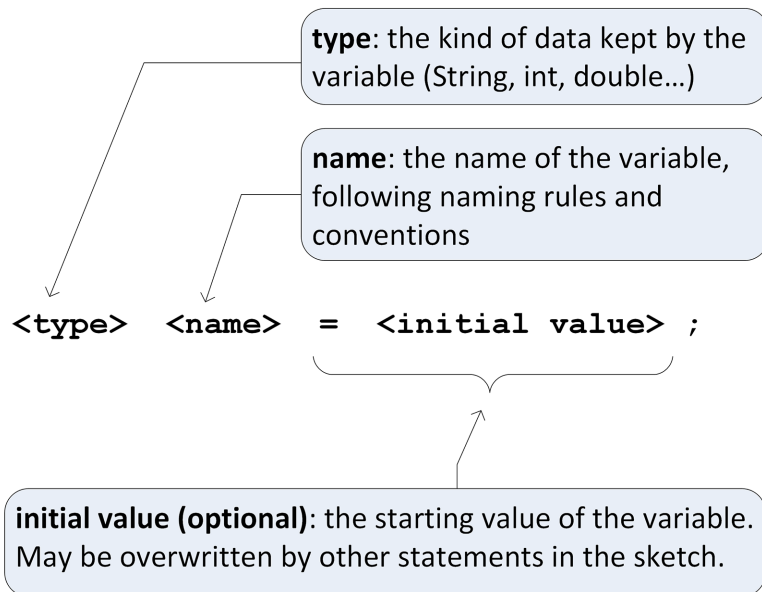
Table 5-3. Vocabulary

Term	Definition
<b>arithmetic operator</b>	A symbol representing a mathematical operation. C has six operators: + - * / % =
<b>assignment operator</b>	The arithmetic operator with the symbol =. The expression to the right of the operator is evaluated. The results are stored to the variable to the left. The act of storing a value to a variable is called assignment.
<code>double</code>	A precision number with a decimal.
<code>float</code>	A number with a decimal with a very large range but less precision than a <code>double</code> .
<code>int</code>	A numeric data type for integers. A variable of type <code>int</code> may have any value from -32767 through +32767.
<b>library</b>	A library is a set of prewritten Arduino™ sketches, each of which perform a particular service.
<b>Math library</b>	A set of mathematical operations prewritten for the C language.
<b>variable</b>	A name given to a location in memory where a value can be stored. A variable must be named according to certain rules, and a data type must be assigned to the variable.

## Description:

### Declaring number variables

#### Example 5-1. Form for declaring number variables



Just as with `String` variables discussed in Lesson 3, number variables must be declared; their names must follow the same naming rules and conventions; and a variable may be assigned an initial value.

#### Example 5-2. Number variables

```
int ageOfParticipant = 17; // initialized variable  
  
int highScore;
```

### Mathematical operators and equations

#### Arithmetic Operators:

Just as with algebra, mathematics in C involves addition, subtraction, multiplication, and division. In addition, C also has assignment and modulus. These are called arithmetic operators. Each has a symbol that is used to indicate the operation in a programming statement.

In Table 5-4, assume the letters A, B, and C are declared numeric variables of some type.

Table 5-4. Arithmetic operators used in C programming

Operator	Symbol	Description	Example
Addition	+	Adds two values.	A + B
Subtraction	-	Subtracts the second value from the first.	A - B
Multiplication	*	Multiplies two values.	A * B

Operator	Symbol	Description	Example
Division	/	Divides the first value by the second.	A / B
Assignment	=	Stores the value of the right side of the operator to the variable on the left.	C = A + B
Modulus (only works with integers and variables of type int)	%	Finds the remainder when the first integer is divided by the second integer.	A % B

### Equations:

Mathematical expressions in C are similar to those in algebra, except the equals sign serves a different purpose. Called the *assignment operator*, it causes the expression to the right of the operator to be resolved and the numeric results placed in the variable to the left. The code segment in Snippet 5-1 is an illustration. Additional examples appear in the practice sketch in Exercises 5-1 and 5-2.

The equation in Snippet 5-1 is:

```
areaOfRectangle = width * height;
```

Notice how a **String** variable is used to print the results in the Serial Monitor. Concatenation can be used to append integer values to a **String** variable.

### Snippet 5-1. Illustration of use of assignment operator

```
...
int areaOfRectangle;
int width;
int height;

// assign width and height values
width = 3;
height = 7;

...
// calculate area and store value to areaOfRectangle
areaOfRectangle = width * height;

// use String, Serial, and concatenation
// to show results in the Serial Monitor
String results = "Area of the rectangle is: ";
results += areaOfRectangle;
Serial.println(results);
...
```

The Serial Monitor will display:

Area of the rectangle is: 21

### The Math library

The Arduino™ IDE includes many prewritten services that can be accessed via libraries. A *library* is a set of pre-written Arduino™ sketches, each of which perform a particular service. Later lessons will make use of many libraries. For now you only need to know the following:

1. The *Math library* offers many services such as square root, raising one number to the power of another, determining absolute value, and trigonometric functions. The last service is very useful for writing sketches that perform navigation, such as is used with quad copters.
2. The services of the Math library can be accessed by name simply by placing the statement `#include <math.h>` at the top of an Arduino™ sketch, just under header comments.



Note

The statement in Snippet 5-2 is not followed by a semicolon.

3. Snippet 5-2 illustrates the use of the Math library to calculate the square root of an integer. A description of the complete services of the Arduino™ Math library can be found at <http://www.arduino.cc/en/Math/H>.

### Snippet 5-2. Use of the Math library

```
...
#include <math.h>

int myNumber;

...
myNumber = 67;

int squareRootOfMyNumber;
squareRootOfMyNumber = sqrt(myNumber);

String results = "The square root of ";
results += myNumber;
results += " is: ";
results += squareRootOfMyNumber;
Serial.println(results);
...
```

The Serial Monitor will display something like:

The square root of 67 is: 8



Notice that the square root is an integer. When working with integers, C returns only integers. Further, as with integer division, integer square roots are not rounded. Thus, the square root of 9 and the square root of 15 are both 3.

**Goals:**

By the end of this lesson readers will:

1. Know how to declare and use number variables of type `int`.
2. Know how to use all six arithmetic operators used in C.
3. Understand that the structure of an equation in C is an expression, the results of which are assigned to a variable.
4. Know that integer division in C yields only the integer quotient and does not round.
5. Know how to use the modulus operator to retrieve the remainder of integer division.
6. Know how to use the Arduino™ Math library.
7. Be able to apply `String` concatenation to prepare `String` values that contain integers.
8. Be able to write Arduino™ sketches that perform mathematical operations.

**Materials:**

Quantity	Part	Image	Notes	Catalog Number
1	Arduino™ Uno		Single-board computer. This board is delicate and should be handled with care. When you are not using it, keep it in a box or plastic bag.	3102
1	USB Cable		This is the standard USB adapter cable with the flat connector on one end and the square connector on the other.	2301
1	Computer with at least one USB port and access to the Arduino™ website, <a href="http://www.arduino.cc">http://www.arduino.cc</a> .	---	The operating system of this computer must be Windows, Macintosh OS/X, or Linux.	---

## Procedure:

1. Begin by connecting your Arduino™ to the computer. Then start the Arduino™ Integrated Development Environment (IDE).
2. In the Arduino™ IDE, enter the multi-line header comments as shown in Snippet 5-3.

### Snippet 5-3.

```
/* Lesson5IntegersAndMath  
   by <your name here>  
   <date goes here>  
*/
```

Substitute the programmer's name and the actual date where indicated by the angle brackets.

3. Underneath the heading comments, import the Math library as shown in Snippet 5-4.

### Snippet 5-4.

```
...  
#include <math.h>
```

4. Declare three variables of type `int` as shown in Snippet 5-5. Notice how multiple variables can be created in a single programming statement if the names are separated by commas.

### Snippet 5-5.

```
...  
int integerA, integerB, integerC;
```

5. Add the `setup()` method as shown in Snippet 5-6. This sketch allows time for the serial port to initialize. Next, a ready message is sent to the Serial Monitor.

### Snippet 5-6.

```
...  
void setup(){  
    Serial.begin(9600);  
    Serial.println("Ready for numbers.");  
}
```

6. Add the `loop()` method as shown in Snippet 5-7. Do not put any programming statements into it yet.



*Snippet 5-7.*

```
...  
void loop(){  
  
}
```

- 7. Save the sketch as `Lesson5IntegersAndMath`.
- 8. Upload the sketch to the Arduino™ and open the Serial Monitor. If successful the Serial Monitor should display the following:

Ready for numbers.

- 9. Add programming statements to the `loop()` method of the sketch, as shown in Snippet 5-8. These statements initialize the first two integers and show them to the user. Also add the `while(true)` statement at the very end of the `loop()` method. This prevents the `loop()` method from running more than once. This statement should remain the last one in the `loop()` method for this sketch.

*Snippet 5-8.*

```
...  
void loop(){  
  
    // exploring integers  
    integerA = 37;  
    integerB = 12;  
  
    String message;  
    message = "Exploring integers. For the following:";  
    message += "\n integerA = ";  
    message += integerA;  
    message += "\n integerB = ";  
    message += integerB;  
    Serial.println(message); // show values of integers  
  
    while(true); // keep this the last line of the loop  
    // method  
}
```

Upload the sketch and open the Serial Monitor. The following should appear:

Ready for numbers.

Exploring integers. For the following:  
integerA = 37  
integerB = 12

10. Above the `while(true)` programming statement, add the programming statements shown in Snippet 5-9 to demonstrate the addition of integers:

#### Snippet 5-9.

```
...  
    // perform integer calculations  
    message = "integerA plus integerB is: ";  
    integerC = integerA + integerB;  
    message += integerC;  
    Serial.println(message);  
  
    while(true);  
}
```

The Serial Monitor should now display:

```
Ready for numbers.  
Exploring integers. For the following:  
integerA = 37  
integerB = 12  
integerA plus integerB is: 49
```

11. Adding statements similar to those in Snippet 5-9 that illustrate subtraction and multiplication are left as an exercise (Exercise 5-1). Next, as shown in Snippet 5-10, add statements that demonstrate integer division.

#### Snippet 5-10.

```
...  
    // subtraction exercise goes here  
    // multiplication exercise goes here  
    // division  
    message = "integerA divided by integerB is: ";  
    integerC = integerA / integerB;  
    message += integerC;  
    Serial.println(message);  
  
    while(true);  
}
```

The Serial Monitor should now display:

```
Ready for numbers.  
Exploring integers. For the following:  
integerA = 37  
integerB = 12  
integerA plus integerB is: 49  
integerA divided by integerB is: 3
```

12. The division operation is not complete without determining and displaying the remainder. This is the function of the modulus operator. Add the programming statements shown in Snippet 5-11 to the `loop()` method just before the `while(true)` statement.

#### Snippet 5-11.

```
...  
// remainder  
message = "the remainder of integerA / integerB is: ";  
integerC = integerA % integerB;  
message += integerC;  
Serial.println(message);  
  
while(true);  
}
```

The Serial Monitor will display:

```
Ready for numbers.  
Exploring integers. For the following:  
integerA = 37  
integerB = 12  
integerA plus integerB is: 49  
integerA divided by integerB is: 3  
the remainder of integerA divided by integerB is: 1
```

13. Finally, use the Math library function to find the square root of an integer by adding the programming statements shown in Snippet 5-12 to the `loop()` method just above the `while(true)` statement:

### Snippet 5-12.

```
...  
  // library function, square root  
  integerA = 143;  
  integerC = sqrt(integerA);  
  message = "the square root of ";  
  message += integerA;  
  message += " is ";  
  message += integerC;  
  Serial.println(message);  
  
  while(true);  
}
```

The Serial Monitor should display:

```
Ready for numbers.  
Exploring integers. For the following:  
  integerA = 37  
  integerB = 12  
integerA plus integerB is: 49  
integerA divided by integerB is: 3  
the remainder of integerA divided by integerB is: 1  
the square root of 143 is 11
```

14. Save the sketch to preserve your work.

### Exercises:

#### *Exercise 5-1. Create subtraction and multiplication examples*

Add the subtraction and multiplication examples to the `Lesson5IntegersAndMath` sketch.

#### *Exercise 5-2. Calculate area and perimeter*

Write a new sketch called `AreaAndPerimeter` that calculates and displays the area and perimeter of a rectangle. Set the width to 27 and the height to 40.