**iCREAT I: Module 3 - Programming Cheat Sheet**

**HELP ME I NEED HELP WITH CODING!!!!!!!!!!!**



By [Jeff Jackowski](https://www.flickr.com/photos/jjackowski/15659707052)

horizontal line

**Introduction** In this class you will be programming with a flavor of C++ for controlling the Arduino microcontroller. If you have programmed before, this may be a good quick refresher. If you haven't, then you will want to read this thoroughly and ask questions along the way.

**Variables** A storage location for information – an address for where information is stored. In C++ variables also have a type, which means if they're a whole number they're an "int" if they're a decimal number they're **normally** a "double" and if they are a word, or mixture of letters, or a number for a code or whatever else, they most likely are a "String." These are the main three types of variables you will be using in this class, there are other ones and if you're expected to use them they will be talked about.

Examples: you declare the variable type (to be defined later), then the name, and then what it stores

* int x = 5; now whenever you reference x the program is going to look it up and find the value 5. what the int before the x means, is that x is going to be an integer, or a whole number. This is important because otherwise the computer doesn't know if x is holding the number 5, or the character '5'. Another type is "long" which is basically an integer, but for when you have really big numbers, the general usage for "long" in this class is going to be for time in nanoseconds or microseconds.
* String alphabet = "abcdefghijklmnopqrstuvwxyz"; here you just have a big string full of letters, in this class you'll mostly be using strings to print messages for troubleshooting your Arduino code
* double multiplier = 1.25; a double is a number like an int, but instead of having to be a whole number can be any value with decimal point.

Now that there are some examples set up you could do x+number and get a returned value of 6.25

**But why are variables important and good?** Variables at first may seem like more work than they're worth, but quickly you'll find that if you're ever referencing the same number more than once then the variable makes it easy to change the value. Variables also provide a way to keep track of what a number is in reference to, since you can name the variable.

**Conditionals** A block of code that will be executed **IF** the statement is true. You can include an ELSE statement after to execute a different block of code assuming your condition is not true

How it looks:

if(condition){

execute this code

}

else{

execute this code

}

You probably noticed the curly braces {} **ALWAYS REMEMBER TO MATCH YOUR CURLY BRACES – IF YOU DON'T YOUR CODE WON'T WORK!!!!!!!!** You also probably noticed the parentheses () remember to match these too 😊

Example: if int a is greater than int b then the code will add a and b together. Otherwise it will subtract them

if( a < b){

a+b;

}else{

a-b;

}

NOTE: You can also add an "else if" statement which will check multiple conditions until it finds one that's true. Here's how it looks:

if(a < b){

a+b;

}

else if(a < c){

a+c;

}

else{

a+b+c;

}

Every line in a program (or sketch as the Arduino refers to it) **MUST end in a semicolon (;**) one of these missing will also wreak havoc on your code!!!

**Operators:**

|  |  |  |
| --- | --- | --- |
| Symbol | What it means | Example |
| < | Less than | 1<2 |
| <= | Less than OR equal to | 1<=2,2<=2 |
| > | Greater than | 2>1 |
| >= | Greater than OR equal to | 2>=1,1>=1 |
| == | equals | 1==1 |
| != | NOT equal | 1!=2 |
| && | AND | (x==1)&&(x!=0) |
| || | OR | (x==1)||(x==0) |
| + | add | X+1 |
| - | subtract | X-1 |
| ++ | Increase by 1 | 1++=2 also means 1+1=2 |
| \* | multiply | 2\*2=4 |
| / | divide | 2/2=1 |
| % | 'Mod' or remainder | 1%1=0 because the remainder is 0, helpful when trying to find even or odd values |

**Loops:**

A loop is a piece of code that will repeat multiple times and allows you to perform more operations with less code. In this class the Arduino will run one loop forever, and that is where all of your code needs to run from.

A **while loop** is a loop that runs infinitely until a condition is met.

Example:

while(sensorReading() > 10){

moveForward();

}

In this Example the loop is going to repeat until a specific sensor reads a value less than 10, let's say it's an ultrasonic sensor reading the distance. As long as nothing is close to it, it will keep moving forward.

Another loop is the **for loop**, a for loop will go for a specific number of iterations and is useful for referencing which iteration it is currently on.

Example:

for(int i = 0; i < 10; i++){

Serial.print("iteration: " + i")

}

Here, the program will start with variable "i" equal to 0, check if I is less than 10, if it is then it will run whatever is inside the loop, in this case print which iteration the loop is on. Then after the loop is executed it will increment i by 1 and repeat the whole process again. The loop will run until i = 10

**Functions:**

A function is basically a set of directions that has a name and you can run easily. If you have the same set of code you're going to want to use in multiple places but not at the same time (what a loop would do) then you're going to want to use a function.

The format for a function is return type, function name(input type

Example:

int add(int a, int b){

return a+b;  
}

The return type in this example is int, right at the start of the function, a return type is saying what type any value the function returns will be. Here when you say add(5,6) it will return 11, how to implement that would be

Serial.print(add(5,6))

That would print the added values as 11.

void setSpeed(int value){

speed = value;

}

For this example, we're using void as the return type, void means it returns nothing. This is common for setting a value or running a little set of code. If you still want output from a void function, you can still use serial print.

**Vocabulary:**

Some terminology used in the examples may be confusing, so here is a list that may help

|  |  |  |  |
| --- | --- | --- | --- |
| Word | Meaning | Word | Meaning |
| Call | run/do | Parameter | An input or requirement for a function. |
| int | Whole number | String | Data in word form |
| double | Decimal number | Upload | Send the code from computer to Arduino |
| print | Makes text in the console box | Sketch | A program in Arduino is called a sketch |
| return | Ends the function and gives back data to whatever called the function | Library | Collection of already made functions that can be brought "imported" into your program |

**Arduino Programming:**   
Now that we have discussed the basics that apply to most programming languages we'll get into the specifics of the Arduino. As stated on the first page, the Arduino has a C++ flavor – the Arduino libraries are written in C and C++. Their sketch structure and functions are built specifically for the Arduino. The initialization happens in the setup() function, like this:

void setup(){

Serial.begin(9600);

pinMode(pin,OUTPUT);

}

You'll notice the call to the 'pinMode' function- this is one of the Arduino-specific additions that enable a smooth run. The function takes an int and another Arduino-specific constant –INPUT or OUTPUT- as parameters. These enable whatever component that is plugged into the specified pin to be categorized as input – giving information to the Arduino – or as output – receiving information from the Arduino. All of your LEDs and buzzers will be set as output. Things like a potentiometer or sensor will be set as INPUT. To make all of these initialized components do something we'll move to the loop() function.

void loop(){

digitalWrite(13,HIGH);

delay(1000);//delay 1000 microseconds

digitalWrite(13,LOW);

delay(1000);//delay 1000 microseconds

}

This block of code probably looks familiar to you – it is the code from lab 2. The digitalWrite() function is another Arduino-specific one. You use it to turn your component on and off with the constants HIGH and LOW coinciding as such. The delay() function does just what it sounds like it does – delay something. In this case the delay made the LED blink on and off. You also probably noticed the // , this is what is known as commenting. The // lets the compiler know that this is not code for the computer or Arduino to read, but a comment for the programmer.

For more information and a more detailed list of Arduino functions you'll be using throughout this class check out their reference website: <https://www.arduino.cc/reference/en/>

Happy coding 😀