Jonathan Hess

Section 3.2

## Week 7 Lab Report: Loops (II)

# Lab Report Rubric

|  |  |  |
| --- | --- | --- |
| **Category** | **Student Score** | **Grader Score** |
| **Organization** | | |
| **Appropriate sections** | **1/1** | **/1** |
| **Appearance and formatting** | **2/2** | **/2** |
| **Spelling, grammar, sentence structure** | **1/1** | **/1** |
| **Work** | | |
| **Experimental procedure** | **2/2** | **/2** |
| **Results (data, code, figure, graph, tables, etc.)** | **2/2** | **/2** |
| **Conclusion** | **2/2** | **/2** |
|  |  |  |
| **Total** | **10/10** | **/10** |

# Introduction

In this Lab we worked on Loops and more programming in both MatLab and Ardunio. Mostly this week was while loops but for loops were used again. I got to use bitwise operators for one of the extra challenges.

# Procedure

**While Loop in MatLab**

The while loop is the same as the one from C++/Java/Python. The while loop does a condition (such as x>0) while it is true. There is also a do while loop that goes through the code at least once, this type of loop is not present in Matlab.

C++ example

while(x > 0){

//loops

}

Matlab

while x> 0

%loops

end

1. What does the script do?

The script runs x from 10 to 1 printing the values as it goes.

2. Move the fprintf in line 3 one line down and run the script. What do you observe? Does the while loop react instantaneously to the variable change? Explain

If the x=x-1 is before the print then it will decrease before printing so it will display 9-0. It doesn’t change how the loop works because it still does the same number of loops

Challenge #1: Have the user input a number. The script will calculate and display the sum from 0 up to the input number. Use only the while loop to complete this challenge.

a = input("number to count up to: ");

count = 0;

sum = 0;

while(count <= a)

disp(sum);

sum = sum+ count;

count = count +1;

end

disp(sum);

In order to make this code you need to have a way of counting and since wants 0 included it must display first and then again outside the loop. Then you add a counter (count = count + 1) and make the condition that it goes up to whatever number the user inputed.

Challenge #2: Create a guess the number game. MATLAB generates a random integer in between 1 and 100. User will input a guess and MATLAB will inform them if the guess is too big, too small, or is correct.

randNum = randi(100);

disp("GUESS THE NUMBER! WIN PRIZES!");

guessNum = 0;

while randNum ~= guessNum

guessNum = input("number guess: ");

%if(isnumeric(guessNum))

if(guessNum > randNum)

disp("too high");

elseif (guessNum < randNum)

disp("ohh just a bit low");

else

disp("COOOOOORENT!");

end

%else

% disp("umm a number... please");

%end

end

disp("Sorry we ran out of prizes.");

I couldn’t get the isnumeric to work and I couldn’t tell why.

**Arduino Loops**

Challenge #3: Create an Arduino script so that the LEDs blink in odd-even intervals. LEDs 1 and 3 will blink first. void setup() {

// put your setup code here, to run once:

for(int i =1; i<5; i++){

pinMode(i, OUTPUT);

}

}

bool count = 1;

void loop() {

// put your main code here, to run repeatedly:

turnLedsOn(count);

delay(500);

turnLedsOff(count);

delay(500);

count = !count; //make 0 and 1

}

void turnLedsOn(bool c){

digitalWrite(c+1,HIGH); //osulate between 1 and 2

digitalWrite(c+3,HIGH); //osulate between 3 and 4

}

void turnLedsOff(bool c){

digitalWrite(c+1,LOW);

digitalWrite(c+3,LOW);

}

With this code I was originally going to use a int and reset it whenever it got too large but if there are only 2 sets than a bool is sufficient and easier to change because you can set it to not itself and oscillate between 0 and 1;

A problem occurred when I plugged it in because of the Yellow LED, I replaced it with a white one and everything worked. In the video I show how the yellow led makes the second light not work at all.

[Video of Even/Odd LEDS](https://www.youtube.com/watch?v=kHpyu1e9OVE)

EXTRA: What if we dial up the difficulty of this challenge? Are you familiar with binary counting? With four LEDs, we can count to 15 in binary by having 0’s as dimmed LED and 1’s as lighted LED. For example, 0 is 0000 which is four dimmed LEDs. Another example, 10 is 1010 which is 2 dimmed LEDs (1st& 3rd from right to left) and 2 lighted LEDs (2nd& 4th from right to left). Have the user input a value and MATLAB will display the binary equivalent on the LEDs. You can scale up the limit by increasing the number of LEDs.

In order to do this you can get the parts of the int via bitwise operators

(n >> k) & 1 (n is the integer and k is the bit wanted)

this is done because to find the k bit we shift the int over that many to have that bit in the 0-1 place of the int

Example: 10100>>2 = 00101

This then gets combined with the & (and) bitwise operator where only bits that are both true will be considered still true

Example: 10100 & 11011 = 10000 (only the 5 bit are both true)

If we use 1 (0001) then only the last bit would be true so any & operation would be testing the last bit (or if we shift it any individual bit)

int n = 5;

digitalWrite(1, (n >> 0) & 1); //bit 0

digitalWrite(2, (n >> 1) & 1); //bit 1

digitalWrite(3, (n >> 2) & 1); //bit 2

digitalWrite(4, (n >> 3) & 1); //bit 3

if you wanted to make it scalable you could use a for loop to go through all of the bits

I had trouble getting the matlab to send data to the arduino (I think because I didn’t have SimuLink installed)

**While loops in Arduino**

1. What does this script do?

It prints 0 through 10 into the serial monitor once

2. Usually, the content in the loop() loop will repeat after the Arduino is done running it. Is that the case for this script? Why?

Count will never be reset so even though it loops the while is untrue so it doesn’t run.

3. Does the value of variable ‘count’ still increase after 10?

No. it is in the while loop that doesn’t run after 10.

4. Can you think of other uses of the while loop in Arduino?

You can use them to keep a light on while a button is being pressed.

Challenge #4: Printing a pyramid. Using a combination of the loops you have learnt, create a script to print a pyramid of x height.

Problems had with design 1:

First I was going to design it where it used an array of chars (filled with spaces) and add 2 stars each time (using only one loop) at location (x+1+i) and x+1-i in the array.

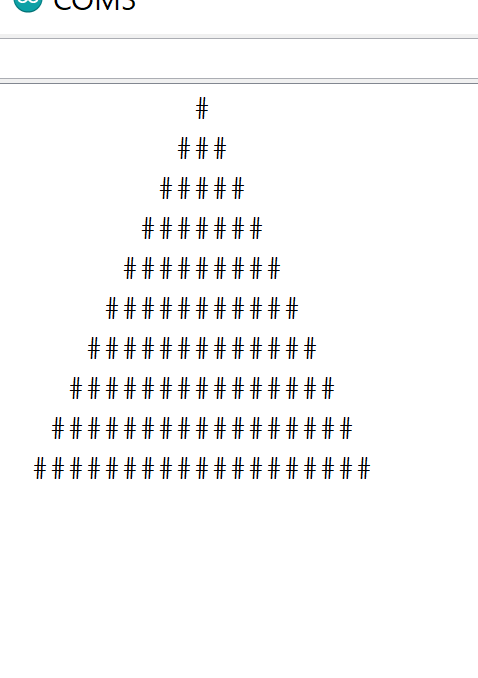
Array size has to be set at start and can’t be set by variable

int x =10;

int total = x\*2 + 1;

char str[total];

This throws an error because of this fact so I cant use an array like this or else it couldn’t scale if the user imputed something over the array length.

So instead of just adding 2 stars to an array with each loop I used the code bellow

int x =10;

void setup() {

Serial.begin(9600);

// put your setup code here, to run once:

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

for(int j=0; j<(x+1)-i;j++){

Serial.print(" ");

}

for(int j=0; j<(i\*2+1); j++){

Serial.print("#");

}

Serial.println("");

}

}

It works by calculating spaces which are (x+1) - i because (x+1) would be the center and with each iteration of the main loop the spaces go down by 1.

Next the stars are calculated by doing (i\*2)+1 because it starts with one star and then adds two every iteration of the main loop

I learned that the arduino code needs a loop or else there will be a compiler error. I learned this after deleting it because I only wanted it to run once and I had nothing in it so I deleted it.

# Results

## Results

I did all the challenges and they seemed interesting. I especially liked the last one with the stars. I learned a lot more about MatLab syntax such as how while loops work and more general knowledge of the IDE. I learned that Crt+C is how you break out of code which is very important to know when dealing with loops. I also learned how to create random ints. I also learned that the not symbol in MatLab is a bitwise not rather than the typical '!’ symbol. I also reaffirmed my understanding of bitwise operators and applied them in a useful way.

# Conclusions and Reflection

It was nice relearning loops because they are an integral part of programming. I find my favorite thing to do is programming problems because I get to apply all the tools I have learned towards a goal in a fairly straightforward manner. I really loved having an excuse to use bitwise operators because I don’t use them that often in programming and they are still confusing to use. (although they are very useful with arduino) The last time I used them was while trying to make a Gamecube to Wii classic controller converter where I had to check the individual bits. I was surprised that we didn’t use arrays because they work so well with loops, such as storing data. I still don’t know why my yellow LED would mess up the circuit because I couldn’t find any difference between yellow and any other color.