

CONDITIONALS (WHAT'S UP?)

LAB 5

SECTION X

SUBMITTED BY:

NATHAN SHULL

SUBMISSION DATE:

OCTOBER 7, 2016

Problem

The purpose of this lab was to create a code that would determine the esplora's orientation, meaning was it facing up, down, left, right, front, or back. We also needed to interpret the acceleration data more carefully as well as use the magnitude to determine if the esplora was moving or at rest. The objective of this lab was to learn how to use conditionals, and also functions within them.

Analysis

The problem states that we must have three constraints on our code. The first described how we had to make the program end if the up button on the Esplora was pressed. The second, and for me, most challenging, constraint was to only print the output of the Esplora one time when the position was changed. This means that if the Esplora was sitting on a table with the joystick up, the program would only output "TOP" once, and not output the orientation again until it was different than "TOP." The third constraint had us incorporate a function, `close_to`, as well as a prototype to find what the orientation was, or what it was close to.

Design

For my code, I started by defining a value for true, and then writing the prototypes for my function `close_to`, and `close_toneg`. Once I defined the prototypes, I began writing in the main loop. I started with a while loop. For the loop to execute, the value had to be true, which would always be the case. Within the while loop, I incorporated a `scanf` statement and two `printf` statements to scan in the data output from the Esplora and print the data collected. Following the `scanf` and `printf` statements, I wrote out 7 if or else if statements.

The first 6 statements were responsible for determining what the orientation of the Esplora was. For TOP/BOTTOM, I used the acceleration in the z direction, and said what the orientation was based on if it was close to positive one or negative one, respectively. The same if else statements were then used for RIGHT/LEFT, compared with the acceleration in the x direction. Lastly, I compared the acceleration values in the y directions, and if it was within the tolerance of 0.05, the output was either FRONT or BACK, depending on if it was positive or negative. I used a tolerance of 0.05 because after experimenting with the output of the accelerometer values, I decided it was reasonable to hold the esplora in a direction within 0.05 of 1 or -1. It wasn't too precise to the point of being impossible, but it was accurate enough to ensure the correct direction. The last else if statement was rather simple, and was responsible for ending the program if the up button on the Esplora was pressed. I did this by saying if `b2` (the up arrow) was true (pressed), then the while loop would break/end.

To get the code to only print once per direction, I had to compare the if statement to something that would only be true once, so that it would only print once. I did this by creating a variable named "orientation" and initially defining it at 0. The first if statement compared it to not equaling 1, which at first is true. However, after running through the individual if statement, the variable "orientation" is redefined to 1, which would make the if statement not run the

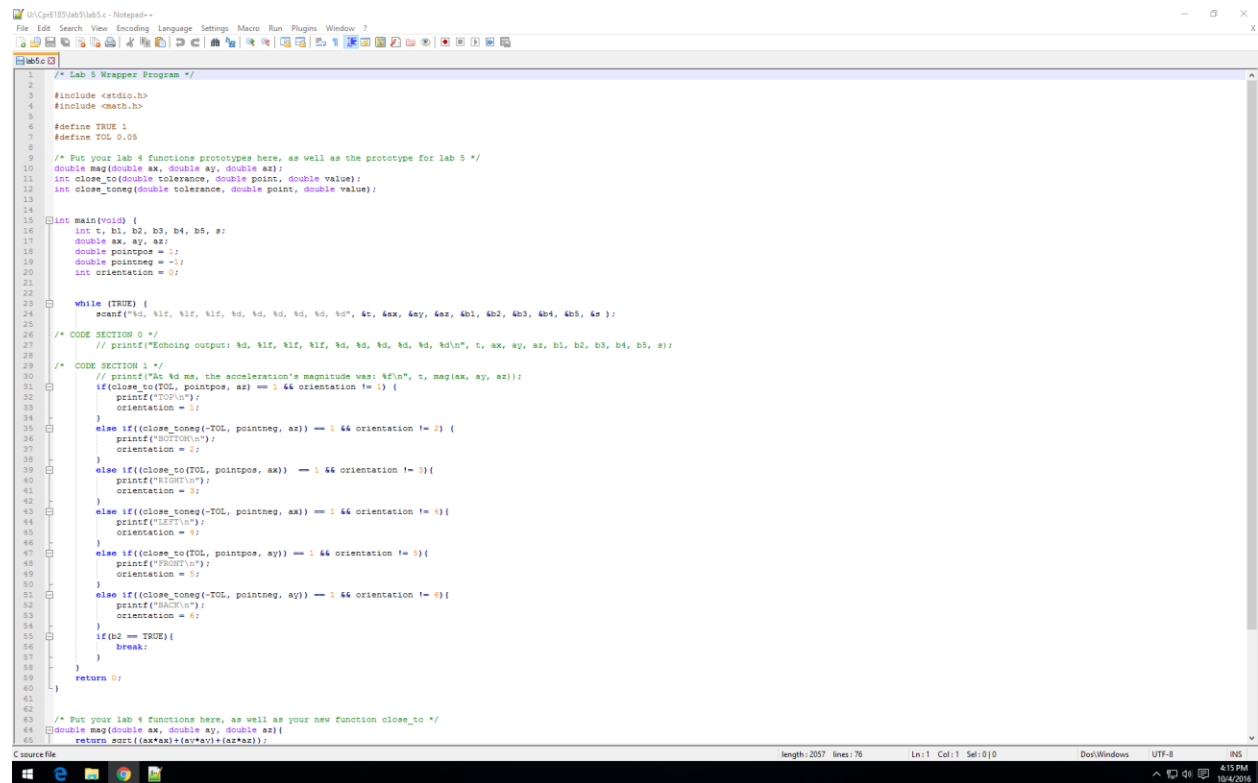
second time through. I then upped the number that orientation was compared to in each consecutive else if statement.

Testing

I encountered several errors when I first created and compiled my code. After my first test, my code was continuously printing my Esplora's orientation, rather than just printing it once. With some help from TA's and thinking about it, I made a variable called orientation. I compared this variable with the expression in the if statement, and made it so it was only true once. I also added this comparison of the orientation variable to all the other else if statements used for determining the direction.

Comments

I chose to use two functions. One called close_to, and another called close_to neg. I used the latter for seeing if values were within 0.05 (tolerance) of -1. The first function was for comparing values within 0.05 of positive 1.



```
1  /* Lab 5 Wrapper Program */
2
3  #include <stdio.h>
4  #include <math.h>
5
6  #define TRUE 1
7  #define TOL 0.05
8
9  /* Put your lab 4 functions prototypes here, as well as the prototype for lab 5 */
10 double mag(double ax, double ay, double az);
11 int close_to(double tolerance, double point, double value);
12 int close_to_neg(double tolerance, double point, double value);
13
14
15 int main(void) {
16     int t, b1, b2, b3, b4, b5, s;
17     double ax, ay, az;
18     double pointpos = 1;
19     double pointneg = -1;
20     int orientation = 0;
21
22
23     while (TRUE) {
24         scanf("%d, %f, %f, %f, %d, %d, %d, %d, %d, %d, %d, %d, %d, %d, %d, %d", &t, &ax, &ay, &az, &b1, &b2, &b3, &b4, &b5, &s);
25
26         /* CODE SECTION 0 */
27         // printf("Echoing output: %d, %f, %f, %f, %d, %d, %d, %d, %d, %d, %d, %d, %d, %d, %d, %d\n", t, ax, ay, az, b1, b2, b3, b4, b5, s);
28
29         /* CODE SECTION 1 */
30         // printf("Max %d ms, the acceleration's magnitude was: %f\n", t, mag(ax, ay, az));
31         if(close_to(TOL, pointpos, az) == 1 && orientation != 1) {
32             printf("TOP\n");
33             orientation = 1;
34         }
35         else if(close_to_neg(TOL, pointneg, az) == -1 && orientation != 2) {
36             printf("BOTTOM\n");
37             orientation = 2;
38         }
39         else if(close_to(TOL, pointpos, ax) == 1 && orientation != 3) {
40             printf("RIGHT\n");
41             orientation = 3;
42         }
43         else if(close_to_neg(TOL, pointneg, ax) == -1 && orientation != 4) {
44             printf("LEFT\n");
45             orientation = 4;
46         }
47         else if(close_to(TOL, pointpos, ay) == 1 && orientation != 5) {
48             printf("FRONT\n");
49             orientation = 5;
50         }
51         else if(close_to_neg(TOL, pointneg, ay) == -1 && orientation != 6) {
52             printf("BACK\n");
53             orientation = 6;
54         }
55         if(b2 == TRUE){
56             break;
57         }
58     }
59     return 0;
60 }
61
62
63 /* Put your lab 4 functions here, as well as your new function close_to */
64 double mag(double ax, double ay, double az){
65     return sqrt((ax*ax)+(ay*ay)+(az*az));
66 }
```

```
13
14
15 int main(void) {
16     int t, b1, b2, b3, b4, b5, s;
17     double ax, ay, az;
18     double pointpos = 1;
19     double pointneg = -1;
20     int orientation = 1;
21
22
23     while (TRUE) {
24         scanf("%d, %lf, %lf, %lf, %d, %d, %d, %d, %d, %d, %d, %d, %d, %d, %d, %d", &t, &ax, &ay, &az, &b1, &b2, &b3, &b4, &b5, &s);
25
26         /* CODE SECTION 0 */
27         // printf("Echoing output: %d, %lf, %lf, %lf, %d, %d, %d, %d, %d, %d, %d, %d, %d, %d, %d, %d\n", t, ax, ay, az, b1, b2, b3, b4, b5, s);
28
29         /* CODE SECTION 1 */
30         // printf("At %d ms, the acceleration's magnitude was: %f\n", t, mag(ax, ay, az));
31         if(close_to(TOL, pointpos, ax) == 1 && orientation != 1) {
32             printf("TOP\n");
33             orientation = 1;
34         }
35         else if(close_to(TOL, pointneg, ax) == -1 && orientation != 2) {
36             printf("BOTTOM\n");
37             orientation = 2;
38         }
39         else if(close_to(TOL, pointpos, ay) == 1 && orientation != 3) {
40             printf("FRONT\n");
41             orientation = 3;
42         }
43         else if(close_to(TOL, pointneg, ay) == -1 && orientation != 4) {
44             printf("LEFT\n");
45             orientation = 4;
46         }
47         else if(close_to(TOL, pointpos, az) == 1 && orientation != 5) {
48             printf("FRONT\n");
49             orientation = 5;
50         }
51         else if(close_to(TOL, pointneg, az) == -1 && orientation != 6) {
52             printf("BACK\n");
53             orientation = 6;
54         }
55         if(b2 == TRUE) {
56             break;
57         }
58     }
59     return 0;
60 }
61
62
63 /* Put your lab 4 functions here, as well as your new function close_to */
64 double mag(double ax, double ay, double az) {
65     return sqrt((ax*ax)+(ay*ay)+(az*az));
66 }
67 int close_to(double tolerance, double point, double value) {
68     if ((point-value) <= tolerance) {
69         return TRUE;
70     }
71 }
72 int close_toney(double tolerance, double point, double value) {
73     if ((point-value) >= tolerance) {
74         return TRUE;
75     }
76 }
```