Conditionals (What's up?)

Lab 5

SUBMITTED BY:

CODY BROOKS

SUBMISSION DATE:

2015-10-06

Problem

  The purpose of this lab is to create a program that will display the current orientation of the Esplora and will exit when the user presses the 'UP' button. The inputs to the program will be accelerometer data and time from `explore.exe` and the output will be the orientation of the device. The objective is to teach loops, conditional statements, functions and comparing floating point numbers.

Analysis

  The problem states that the program must display the orientation of the Esplora while it is not moving. The Esplora is still when the magnitude of its acceleration is equal to one. The magnitude of acceleration is calculated using the formula $\sqrt{x^2+y^2+z^2}$ .When the Esplora is still, it's orientation is dependent on which of the three accelerometer values (X, Y or Z) is equal to 1 or -1. If the accelerometer has an X value of 1, then the Esplora is facing left. If the accelerometer has an X value of -1 then the Esplora is facing right. If the accelerometer has a Y value of 1, then the Esplora is facing the front. If the accelerometer has a Y value of -1, then the Esplora is facing the back. If the accelerometer has a Z value of 1, then the Esplora is facing up. If the accelerometer has a Z value of -1, then the Esplora is facing down.

Design

  The problem requires that the the program runs until the user presses the 'UP' button on the Esplora. This means that the main `while` loop's condition will be based on the value of the 'UP' button that is collected using `scanf()`. To evaluate the orientation of the Esplora, the X, Y and Z accelerations are passed into the function `getDirection()`. To make the program easier to read, `getDirection()` returns an enum value describing the direction. The enum contains each direction (UP, DOWN, LEFT, RIGHT, FRONT and BACK) and is defined before the `main()` function.

  The `getDirection()` function compares each X, Y and Z input to 1 and -1 to determine its orientation as outlined in the above Analysis. Because X, Y and Z are floating point values, conditional statements cannot use '=='. To solve this, a `closeTo()` function was created. This function takes the parameters `double tolerance`, `double target` and `double val`. `closeTo()` checks to see if val is within the limits `(target - tolerance)` and `(target + tolerance)` and returns a boolean value (using the `stdbool.h` library). `closeTo()` is then used within `getDirection()` to compare each X, Y and Z value to 1 and -1.

Testing

  The main test is to see the output when the Esplora data is piped through `lab5.c`. By checking to see if the orientation displayed matches the actual orientation of the board, the main function of the program is checked. Another thing to check is to see if an orientation is displayed while the board is in motion. The same orientation should never be displayed twice because orientation should only be displayed if the orientation changes.

Place the board in the 'UP' position and start the program. Check to see if the orientation is displayed is 'UP'. Move the board and bring the board back to a still position in the same orientation as it started. Nothing should be displayed because the orientation did not change and nothing should be displayed while the board is in motion. Move the board to the next position and repeat the above steps.

Comments

The design choice of using enums instead of arbitrary integers had benefits and drawbacks. It made the program much easier to read and debug than if those enum constants were arbitrary integers. Because of the nature of how GCC compiles the source code the declaration of the enum `Direction` had to come outside of the `main()` function and before the prototype of the `getDirection()` function. Even though global variables are generally considered bad practice, it was worth having because it was never changed within the function.

```c
/* Lab 5 Wrapper Program */

#include <stdio.h>
#include <math.h>
#include <stdbool.h>

/* Defines the possible orientations of the Esplora */
enum Direction {UP, DOWN, LEFT, RIGHT, FRONT, BACK};

/* Put your lab 4 functions prototypes here, as well as the prototype for lab 5 */
double mag (double x, double y, double z);
bool closeTo(double tolerance, double target, double val);
enum Direction getDirection(double x, double y, double z);


int main(void) {
    int t, b1, b2, b3, b4, b5, s;
    double ax, ay, az;
    enum Direction orientation;
    enum Direction lastOrientation;

    while (true) {
        scanf("%d, %lf, %lf, %lf, %d, %d, %d, %d, %d, %d", &t, &ax, &ay, &az, &b1, &b2, &b3,
        &b4, &b5, &s );
        orientation = getDirection(ax, ay, az);

/* CODE SECTION 0
        printf("Echoing output: %d, %lf, %lf, %lf, %d, %d, %d, %d, %d, %d\n", t, ax, ay, az,
        b1, b2, b3, b4, b5, s);
*/

/*  CODE SECTION 1 */

        if (orientation != -1) {
            if (orientation != lastOrientation){

                if (b2 == 1){
                    return 0;  // Exits the program when the user presses the UP ARROW button
                    on the Esplora
                }

                if (orientation == UP) {
                    printf("UP\n");
                } else if (orientation == DOWN) {
                    printf("DOWN\n");
                } else if (orientation == LEFT) {
                    printf("LEFT\n");
                } else if (orientation == RIGHT){
                    printf("RIGHT\n");
                } else if (orientation == FRONT) {
                    printf("FRONT\n");
                } else if (orientation == BACK) {
                    printf("BACK\n");
```

```c
        } else {
            printf("MOVING\n");
        }

    }

    lastOrientation = orientation;

    }

    }

    return 0;

}

/* Put your lab 4 functions here, as well as your new function close_to */
double mag (double x, double y, double z) {
    double magnitude = sqrt(pow(x, 2) + pow(y, 2) + pow(z, 2));
    return magnitude;
}

bool closeTo(double tolerance, double target, double val) {
    double lower, higher;
    lower = (target - tolerance);
    higher = (target + tolerance);
    if (val > lower && val < higher) {
        return true;
    } else {
        return false;
    }
}

enum Direction getDirection (double x, double y, double z) {
    if (closeTo(0.25, 1, mag(x, y,z))) { // When the Esplora is not moving
        if (closeTo(0.25, 1, z)) {
            return UP;
        } else if (closeTo(0.25, -1, z)) {
            return DOWN;
        } else if (closeTo(0.25, 1, y)) {
            return FRONT;
        } else if (closeTo(0.25, -1, y)) {
            return BACK;
        } else if (closeTo(0.25, 1, x)) {
            return LEFT;
        } else if (closeTo(0.25, -1, x)) {
            return RIGHT;
        } else {
            return -1;
        }

    } else {
        return -1;
```
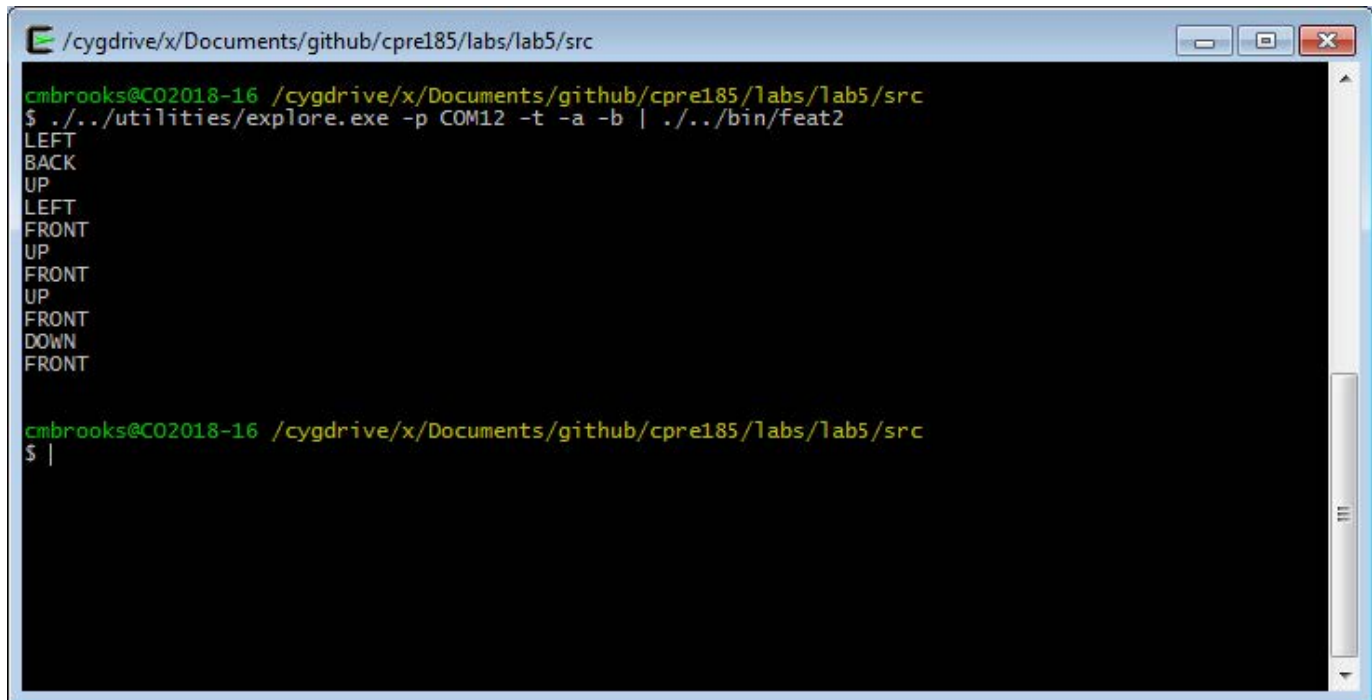
```
        }
}
```

-3-

```
/cygdrive/x/Documents/github/cpre185/labs/lab5/src

cmbrooks@CO2018-16 /cygdrive/x/Documents/github/cpre185/labs/lab5/src
$ ./../utilities/explore.exe -p COM12 -t -a -b | ./../bin/feat2
LEFT
BACK
UP
LEFT
FRONT
UP
FRONT
UP
FRONT
DOWN
FRONT


cmbrooks@CO2018-16 /cygdrive/x/Documents/github/cpre185/labs/lab5/src
$ |
```