INTRO TO FUNCTIONS

LAB 4

SECTION A

SUBMITTED BY:

BRIAN REBER

SUBMISSION DATE:

10/2/2009

Lab Problem
The purpose of this lab is to introduce functions, and to practice the concepts of division and formatting output.

Analysis
In this lab, there are 5 different parts.

1. Edit section 0 to format the seconds and acceleration outputs.
2. Create the mag function that will compute the magnitude of the acceleration.
3. Make the time output in minutes, seconds, and milliseconds. To do this, create three functions to compute these values.
4. Use the output from your program to analyze the values. What does each mean?
5. Make a new file that will print out whether the Wiimote is on its face or tail. If it is neither, it won't print out anything.

Inputs:
There will be data coming in via the Wiimote and wiiwrap.

Outputs:

1. It will print out "Echoing output (time in ms), (acceleration x), (acc y), (acc z)" We need to format these values.
2. Same as 1. Also, it will print out "At (time in ms), the acceleration's magnitude was: (magnitude)"
3. Same as 2. Also, it will print out the time in minutes, seconds, milliseconds.
4. It will print out "The Wiimote is on its (TAIL or FACE) at time in minutes, seconds, and milliseconds."

Design
Algorithm:

1. For this part, we just need to print out a line containing the time in seconds, and the acceleration in the x, y and z directions. These numbers need to be formatted according to the instructions.
2. We need to create a function that takes three parameters, and then returns the square root of the sum of the squares of the three parameters. (sqrt(x*x+y*y+z*z))
3. We needed to create three methods that each takes one parameter (ms). They will return the number of minutes, seconds, and the remaining milliseconds.
4. For this part, there isn't much to set up. All we need to do is look at the output and analyze what each value means.
5. We can reuse a lot of the code from the first three parts. Using this, we can use nested if statements to test whether the values of the magnitude and acceleration represent the Wiimote being on its face or tail.

Testing

1. The main problem we had with this part was figuring out how to format the time correctly. We were a little confused because the instructions told us to format the time with three digits of precision, but the value the printf function was expecting was an int. We didn't know if we could format an int, so we changed it to expect a double, and formatted it with %8.3lf. Then in the variable part of the printf function, we changed t to t/1000.0, to get it into seconds, and make it a double. The accelerations were easy to format – we did %7.4lf to format those.
2. This was our first experience making our own function. Overall, it went well. We made the function that used the given formula, and returned that value. To test it, we let the Wiimote rest on the table, and the magnitude was 1, which is correct. Then we let it drop, and it was less than 1, and when it hit our hand it spiked above 1, which is correct.
3. For this section, we needed to create three new functions. Creating the functions

wasn't really difficult, but making them return the correct values was somewhat confusing. Finding minutes was easy. We divided the ms value by 60000. Then we ran into our first problem. We wanted to subtract the correct amount of ms from the value of t before it was passed into the seconds function, but we don't think we can do that. So in the seconds function, we took the value of ms, and subtracted the value of (minutes(ms)*60000). This way we took off the value of minutes. For milliseconds, we did the same thing, but subtracted minutes and seconds. To test this, we decided to take out the while loop, and just have the user input a number. This way we could isolate the piece we were working on. We used the given example, and it was correct.

4. For this part, we were able to notice that when the Wiimote was at rest (no matter which direction, either nose or tail) the value of the magnitude was very close to 1. When the Wiimote was moving, the magnitude told us the g-forces the Wiimote was experiencing. When it is in freefall, the value of the magnitude was less than 1. When it strikes something, the value of the magnitude spiked well above 1.
5. For this section, we were able to reuse the majority of the code we had already written. We needed to adjust the while loop to do what we needed, but the functions we needed were already written. To figure out whether the Wiimote was at rest, we used an if statement and checked that the magnitude was within the range of .9-1.1. The reason we needed to use a range was because the sensors are often slightly different, and we needed to compensate for this. To test whether the Wiimote was on its tail or face we used the acceleration in the y direction. If it was between -.9 and -1.1, the Wiimote was on its tail. If it was between .9 & 1.1, it was on its face.

Comments

This lab was quite interesting for many reasons. Firstly, being able to control whether the program is printing out lines on the screen by using a Wiimote was really cool. Also problem solving and not being walked through the steps was also enjoyable. I also liked doing more coding. I enjoy programming a lot, and being able to use extra peripherals is cool. Writing our own functions was also fun. I learned a lot through problem solving and figuring out what I needed to do to get the program to print out what I wanted in the correct format.

Implementation

```c
/* Lab 4 Wrapper Program
 Brian Reber & Nathan Brinkman
 */

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

#define TRUE 1

/* Put your function prototypes here */
double mag(float x, float y, float z);
int minutes(float ms);
int seconds(float ms);
int leftover(float ms);


int main(void) {
     int t;
     float  ax, ay, az;


     while (TRUE) {
      scanf("%d,%f,%f,%f", &t, &ax, &ay, &az);

      /* CODE SECTION 0 */
      printf("Echoing output: %8.3lf, %7.4lf, %7.4lf, %7.4lf\n", t/1000.0, ax,
ay, az);

      /*     CODE SECTION 1 */
      printf("At %d ms, the acceleration's magnitude was: %f\n", t, mag(ax, ay,
az));

      /*     CODE SECTION 2 */
      printf("At %d minutes, %d seconds, and %d ms it was: %f\n", minutes(t),
seconds(t), leftover(t), mag(ax,ay,az));

      }

     return 0;

}

double mag(float x, float y, float z)
{
     return sqrt(x*x + y*y + z*z);
}

int minutes(float ms)
{
     return ms/60000;
}

int seconds(float ms)
{
     ms = ms - (minutes(ms)*60000);

     return ms/1000;
}

int leftover(float ms)
{
     return ms - (minutes(ms)*60000) - (seconds(ms)*1000);
}
```

```
/* Lab 4 Wrapper Program Final
   Brian Reber & Nathan Brinkman
 */

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

#define TRUE 1

/* Put your function prototypes here */
double mag(float x, float y, float z);
int minutes(float ms);
int seconds(float ms);
int leftover(float ms);


int main(void) {
     int t;
     float  ax, ay, az;


     while (TRUE) {
      scanf("%d,%f,%f,%f", &t, &ax, &ay, &az);

      if (mag(ax,ay,az) > .9 && mag(ax,ay,az) < 1.10)
      {
             if (ay < -.9 && ay > -1.1)
                    printf("The Wiimote is on its TAIL at %d minutes, %d
seconds and %d miliseconds.\n", minutes(t), seconds(t), leftover(t));
             if (ay > .9 && ay < 1.1)
                    printf("The Wiimote is on its FACE at %d minutes, %d
seconds and %d miliseconds.\n", minutes(t), seconds(t), leftover(t));

      }

     }

     return 0;

}

double mag(float x, float y, float z)
{
     return sqrt(x*x + y*y + z*z);
}

int minutes(float ms)
{
     return ms/60000;
}

int seconds(float ms)
{
     ms = ms - (minutes(ms)*60000);

     return ms/1000;
}

int leftover(float ms)
{
     return ms - (minutes(ms)*60000) - (seconds(ms)*1000);
}
```

Part 4 Output

Face – At Rest

```
Echoing output:    0.074,  0.0000,  1.0000,  0.0741
At 74 ms, the acceleration's magnitude was: 1.002740
At 0 minutes, 0 seconds, and 74 ms it was: 1.002740
Echoing output:    0.084,  0.0000,  1.0000,  0.0741
At 84 ms, the acceleration's magnitude was: 1.002740
At 0 minutes, 0 seconds, and 84 ms it was: 1.002740
Echoing output:    0.092,  0.0000,  1.0000,  0.0741
At 92 ms, the acceleration's magnitude was: 1.002740
At 0 minutes, 0 seconds, and 92 ms it was: 1.002740
```

Tail – At Rest

```
Echoing output:    0.068,  0.0000, -0.9259,  0.0741
At 68 ms, the acceleration's magnitude was: 0.928884
At 0 minutes, 0 seconds, and 68 ms it was: 0.928884
Echoing output:    0.084,  0.0000, -0.9259,  0.0741
At 84 ms, the acceleration's magnitude was: 0.928884
At 0 minutes, 0 seconds, and 84 ms it was: 0.928884
Echoing output:    0.100, -0.0400, -0.9259,  0.0741
At 100 ms, the acceleration's magnitude was: 0.929745
At 0 minutes, 0 seconds, and 100 ms it was: 0.929745
```

Back – At Rest

```
Echoing output:    0.078,  0.0000,  0.0370,  1.0370
At 78 ms, the acceleration's magnitude was: 1.037698
At 0 minutes, 0 seconds, and 78 ms it was: 1.037698
Echoing output:    0.094,  0.0000,  0.0370,  1.0741
At 94 ms, the acceleration's magnitude was: 1.074712
At 0 minutes, 0 seconds, and 94 ms it was: 1.074712
Echoing output:    0.101,  0.0000,  0.0370,  1.0370
At 101 ms, the acceleration's magnitude was: 1.037698
At 0 minutes, 0 seconds, and 101 ms it was: 1.037698
```

Moving

```
Echoing output:    0.070,  0.8400,  0.5556,  1.0741
At 70 ms, the acceleration's magnitude was: 1.472371
At 0 minutes, 0 seconds, and 70 ms it was: 1.472371
Echoing output:    0.084,  1.0800,  0.7407,  1.2963
At 84 ms, the acceleration's magnitude was: 1.842683
At 0 minutes, 0 seconds, and 84 ms it was: 1.842683
Echoing output:    0.094,  1.4000,  0.8519,  1.5556
At 94 ms, the acceleration's magnitude was: 2.259515
At 0 minutes, 0 seconds, and 94 ms it was: 2.259515
Echoing output:    0.102,  1.6400,  0.8519,  1.7407
At 102 ms, the acceleration's magnitude was: 2.538785
At 0 minutes, 0 seconds, and 102 ms it was: 2.538785
Echoing output:    0.111,  1.8800,  0.8148,  1.9259
At 111 ms, the acceleration's magnitude was: 2.812030
At 0 minutes, 0 seconds, and 111 ms it was: 2.812030
Echoing output:    0.123,  2.2400,  0.9259,  2.1111
At 123 ms, the acceleration's magnitude was: 3.214301
At 0 minutes, 0 seconds, and 123 ms it was: 3.214301
Echoing output:    0.133,  2.4400,  0.8889,  2.1481
At 133 ms, the acceleration's magnitude was: 3.370202
At 0 minutes, 0 seconds, and 133 ms it was: 3.370202
```