INTRODUCTION TO FUNCTIONS

LAB 3 SECTION 1

SUBMITTED BY:

ETHAN GRIESMAN

SUBMISSION DATE:

9/20/2022

Problem 1

The purpose of part one was to download the source code from the lab document and compile it. Then

Analysis

The problem states that we had to take the first row of four which was outputting milliseconds and convert the milliseconds to seconds. We had to output an 8 character area with 3 digits of decimal numbers. I had to use the knowledge that 1 second is equal to 1000 milliseconds to convert from milliseconds to seconds. It also wanted us to print the outputs of the other 3 rows with a 7-character with 4 digits of decimals

Design

The problem was to modify the code we downloaded from the lab document to output seconds instead of milliseconds since the DualShock controller. To accomplish this I did the following:

1) Get the output of milliseconds
2) Divide the time variable by 1000
3) Print the time in seconds instead of milliseconds

Using this basic outline I simply took the variable t, which I used to scan the time output abd divided the value being outputted by 1000. I also added %8.3lf to make sure that the seconds would be displayed with a 8-character area and 3 digits of decimal. As well as I added %7.4lf to the rest of the DualShock outputs to print a 7-character area with 4 digits of precision.

RESULT:

```
Echoing output:    8.266, 0.966795, 0.199414, 0.000061
Echoing output:    8.282, 0.968138, 0.196972, 0.000061
Echoing output:    8.297, 0.967405, 0.199048, 0.000061
Echoing output:    8.313, 0.969725, 0.196118, 0.000061
Echoing output:    8.328, 0.969969, 0.197217, 0.000061
Echoing output:    8.344, 0.966307, 0.194897, 0.000061
Echoing output:    8.359, 0.968626, 0.198804, 0.000061
Echoing output:    8.375, 0.966551, 0.196728, 0.000061
Echoing output:    8.390, 0.966917, 0.198315, 0.000061
Echoing output:    8.406, 0.967405, 0.198071, 0.000061
Echoing output:    8.421, 0.967161, 0.196362, 0.000061
Echoing output:    8.436, 0.965452, 0.195752, 0.000061
Echoing output:    8.452, 0.966795, 0.196606, 0.000061
Echoing output:    8.468, 0.971434, 0.198071, 0.000061
Echoing output:    8.483, 0.964231, 0.197461, 0.000061
Echoing output:    8.499, 0.969847, 0.198437, 0.000061
Echoing output:    8.515, 0.966429, 0.194897, 0.000061
Echoing output:    8.531, 0.967161, 0.197217, 0.000061
Echoing output:    8.546, 0.970945, 0.197583, 0.000061
Echoing output:    8.562, 0.965330, 0.196118, 0.000061
Echoing output:    8.578, 0.966917, 0.197827, 0.000061
Echoing output:    8.593, 0.967405, 0.195019, 0.000061
Echoing output:    8.608, 0.968870, 0.196484, 0.000061
Echoing output:    8.624, 0.964475, 0.196118, 0.000061
Echoing output:    8.640, 0.971068, 0.196362, 0.000061
Echoing output:    8.656, 0.963255, 0.194775, 0.000061
Echoing output:    8.671, 0.963743, 0.197461, 0.000061
Echoing output:    8.687, 0.968504, 0.198804, 0.000061
Echoing output:    8.702, 0.968626, 0.199292, 0.000061
Echoing output:    8.718, 0.966917, 0.197583, 0.000061
Echoing output:    8.733, 0.967649, 0.198193, 0.000061
Echoing output:    8.749, 0.969847, 0.198315, 0.000061
Echoing output:    8.764, 0.968748, 0.196850, 0.000061
Echoing output:    8.779, 0.971312, 0.195752, 0.000061
Echoing output:    8.795, 0.961668, 0.197339, 0.000061
Echoing output:    8.810, 0.963621, 0.195263, 0.000061
Echoing output:    8.826, 0.966795, 0.190380, 0.000061
Echoing output:    8.842, 0.966551, 0.196362, 0.000061
Echoing output:    8.858, 0.966184, 0.196484, 0.000061
Echoing output:    8.874, 0.970213, 0.198804, 0.000061
Echoing output:    8.890, 0.972044, 0.203809, 0.000061
Echoing output:    8.906, 1.025759, 0.208936, 0.000061
Echoing output:    8.922, 0.955930, 0.188793, 0.000061
Echoing output:    8.937, 0.984618, 0.196362, 0.000061
Echoing output:    8.953, 0.973387, 0.203565, 0.000061
```

```
Echoing output: 6609, 0.968626, 0.195019, 0.000061
Echoing output: 6625, 0.966184, 0.198193, 0.000061
Echoing output: 6640, 0.966062, 0.197217, 0.000061
Echoing output: 6655, 0.965574, 0.197461, 0.000061
Echoing output: 6671, 0.967405, 0.194653, 0.000061
Echoing output: 6686, 0.969114, 0.197949, 0.000061
Echoing output: 6702, 0.965452, 0.198193, 0.000061
Echoing output: 6718, 0.966551, 0.198437, 0.000061
Echoing output: 6733, 0.966429, 0.197583, 0.000061
Echoing output: 6748, 0.963865, 0.198193, 0.000061
Echoing output: 6763, 0.968260, 0.196240, 0.000061
Echoing output: 6779, 0.968504, 0.199048, 0.000061
Echoing output: 6794, 0.967161, 0.196728, 0.000061
Echoing output: 6809, 0.967771, 0.199414, 0.000061
Echoing output: 6825, 0.969847, 0.195630, 0.000061
Echoing output: 6841, 0.966795, 0.196362, 0.000061
Echoing output: 6855, 0.966551, 0.198804, 0.000061
Echoing output: 6871, 0.967283, 0.197339, 0.000061
Echoing output: 6887, 0.964720, 0.198560, 0.000061
Echoing output: 6903, 0.970579, 0.200513, 0.000061
Echoing output: 6917, 0.968260, 0.195874, 0.000061
Echoing output: 6933, 0.969236, 0.195508, 0.000061
Echoing output: 6948, 0.963499, 0.197827, 0.000061
Echoing output: 6964, 0.971800, 0.197095, 0.000061
Echoing output: 6979, 0.969481, 0.194653, 0.000061
Echoing output: 6994, 0.967283, 0.196362, 0.000061
Echoing output: 7009, 0.970457, 0.193676, 0.000061
Echoing output: 7026, 0.968504, 0.194287, 0.000061
Echoing output: 7042, 0.964964, 0.196362, 0.000061
Echoing output: 7058, 0.969481, 0.198926, 0.000061
Echoing output: 7073, 0.968138, 0.198071, 0.000061
Echoing output: 7089, 0.966184, 0.199170, 0.000061
Echoing output: 7109, 0.965452, 0.194897, 0.000061
Echoing output: 7125, 0.969725, 0.198682, 0.000061
Echoing output: 7139, 0.967161, 0.196728, 0.000061
Echoing output: 7155, 0.967527, 0.198315, 0.000061
Echoing output: 7170, 0.970945, 0.198315, 0.000061
Echoing output: 7186, 0.970823, 0.197339, 0.000061
Echoing output: 7202, 0.969358, 0.196362, 0.000061
Echoing output: 7217, 0.968992, 0.197705, 0.000061
Echoing output: 7232, 0.971556, 0.201978, 0.000061
Echoing output: 7247, 0.966551, 0.202832, 0.000061


egriesma@CO2048-06 /cygdrive/u/CPR_E_185/LAB3
$ gcc lab3.c -o lab3.exe
```

Testing

In order to verify that the code was modified correctly I had to save and compile it. After it was compiled I ran the program with the previously used explore.exe that we used in previous labs. As expected the program outputted the time in seconds with 8 characters and 3 digits of decimal, as well as the rest of the outputs printed with 7 characters and 4 digits of decimal.

Comments

In this problem I learned that it is very important to make sure that you do the correct conversion. When I first tried printing the values I accidentally multiplied the milliseconds by 1000 instead of dividing. This in turn created huge numbers instead of the smaller ones we were looking for. This was a very simple mistake.

Problem 2

The second problem had us further modify the code we downloaded from the lab document. We had to create a function, entitled "mag", that would calculate the magnitude of the acceleration of the DualShock. We then had to input three values into mag and print the output on the screen next to the time in seconds.

Analysis

The second problem states that we must create a function called mag and input three variables into it to output the magnitude of the accelerations. To do this we took three rows of data from the DualShock and assigned a variable to each, x, y, z. Then we returned the result of the mag function which is the square root x*x+y*y+z*z

Design

The problem was to create a function called mag to calculate the magnitude of the accelerations then to display it next to the seconds that have passed. To do this I created this step by step procedure:

1) Define the function at the top of the page
2) Use the mag formula to calculate the magnitude of the accelerations
3) Return the magnitude
4) Call the function and print the magnitude next to the seconds we calculated the last problem

Using this basic outline I designed the program with those steps. I first defined the function before writing it at the bottom of the code. Then I wrote the function that passed three variables and called in mag which returned a double. Then using the mag equations I returned the magnitude of the three inputted values. Then I called it in the print statement using x, y, and z values.

Testing

In order to make sure that I was getting the correct outputs I first made sure that the code would compile. Doing this verified to me the correctness of the code.

RESULT:

```
At 26 ms, the acceleration's magnitude was: 0.988685
At 41 ms, the acceleration's magnitude was: 0.993540
At 57 ms, the acceleration's magnitude was: 0.983713
At 73 ms, the acceleration's magnitude was: 0.980092
At 89 ms, the acceleration's magnitude was: 0.983951
At 104 ms, the acceleration's magnitude was: 0.993343
At 119 ms, the acceleration's magnitude was: 0.986745
At 135 ms, the acceleration's magnitude was: 0.984282
At 150 ms, the acceleration's magnitude was: 0.996368
At 166 ms, the acceleration's magnitude was: 0.990289
At 182 ms, the acceleration's magnitude was: 0.992205
At 198 ms, the acceleration's magnitude was: 0.985502
At 213 ms, the acceleration's magnitude was: 0.985623
At 229 ms, the acceleration's magnitude was: 0.985249
At 244 ms, the acceleration's magnitude was: 0.987340
At 260 ms, the acceleration's magnitude was: 0.985491
At 275 ms, the acceleration's magnitude was: 0.986536
At 291 ms, the acceleration's magnitude was: 0.988480
At 307 ms, the acceleration's magnitude was: 0.984496
At 323 ms, the acceleration's magnitude was: 0.990194
At 339 ms, the acceleration's magnitude was: 0.988757
At 355 ms, the acceleration's magnitude was: 0.987105
At 371 ms, the acceleration's magnitude was: 0.988258
At 386 ms, the acceleration's magnitude was: 0.989882
At 402 ms, the acceleration's magnitude was: 0.988661
At 417 ms, the acceleration's magnitude was: 0.982272
At 433 ms, the acceleration's magnitude was: 0.988223
At 448 ms, the acceleration's magnitude was: 0.989142
At 464 ms, the acceleration's magnitude was: 0.987435
At 480 ms, the acceleration's magnitude was: 0.990027
```

Comments

In this problem, I learned that one must be very careful while creating and initiating functions. It is very easy to forget that you need a squiggly brace, or to not call the function correctly. The most important thing is to make sure that your inputs match the number and type that you defined in the function before calling it.

Problem 3

In problem three we had to modify the code again to take the outputted seconds and convert them into minutes, seconds, and milliseconds. We had to write three separate functions to accomplish this, not just one.

Analysis

The problem was to take the millisecond output time and convert them to minutes, seconds, and milliseconds. To do this I had to figure out that there are 60,000 milliseconds in a minute, and from there I could use modulators to divide what I had left over after computing the minutes.

Design

The third problem assigned was to take the milliseconds being outputted by the DualShock and convert them into minutes, seconds, and milliseconds. To do this I broke it down into several steps:

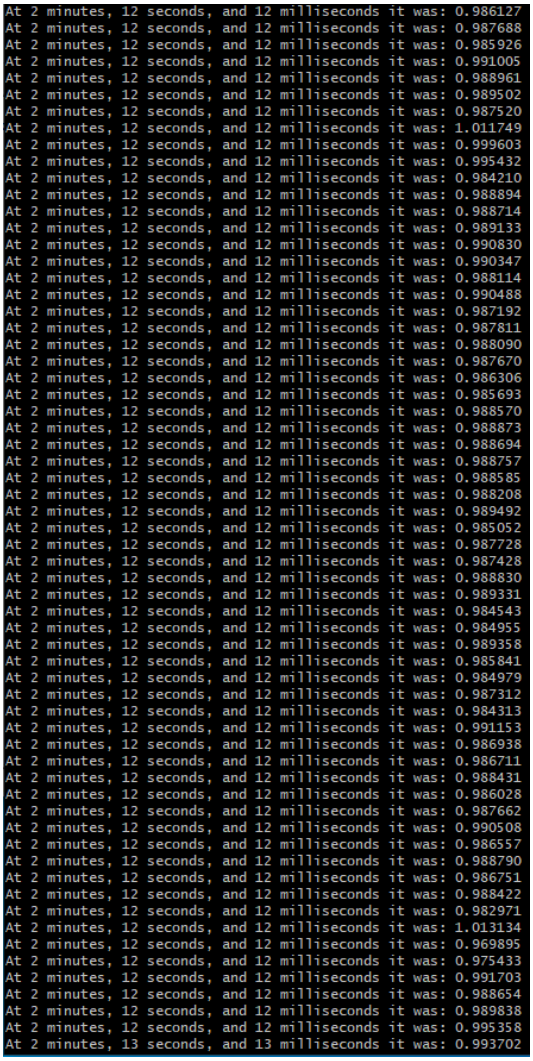
1) Create a function to convert the milliseconds to minutes by dividing the milliseconds by 60,000
2) Return the minutes
3) Create a function to convert the remaining milliseconds to seconds by taking the milliseconds and dividing that by 1000
4) Return the seconds
5) Create a function to take the remaining milliseconds by milliseconds%1000
6) Return the milliseconds

Using this outline, I designed the functions in that order. I created three functions to calculate each, then called each of them above in the print function next to mag function.

Testing

In order to test if the code I wrote worked I compiled it. Since I was getting a reasonably small amount of minutes, and the milliseconds counted all the way up to 1000 before switching to a second I knew that I had done the code correctly.

RESULT:



Problem 4

The purpose of this part was to write our own code to output the number of buttons pressed each time the DualShock Controller produces a line.

Analysis

As stated previously, the objective is to find a function that will output the number of DualShock buttons being pressed at a given time. The function that best suits this purpose is the fflush(stdout) function, which makes it easier to narrow down just the input and output data from the number of buttons being pressed.

Design

The way I approached this was through first writing a new function titled numButtons, containing all ints (since we can't have a double or float value for the number of buttons being pressed of course) titled b1, b2, b3, and b4. Then I repeated the function declaration further down in the source code, and above it I wrote a printf statement that, essentially, translates b1, b2, b3, and b4 to a, b, c, and d respectively for the sake of presenting the output in a more concise way. This is done outside of the function however. Below my function declaration, I also defined numButt as the sum of the buttons, since we are after all looking for the total number of buttons being pressed.

Testing

Testing for this part was fairly easy as it just involved pressing the 4 buttons of the Dualshock controller in any given combination and observing the resulting output through the terminal.

RESULT (output saved as .txt file as instructed):

```
.The total number of buttons being pressed is 0
.The total number of buttons being pressed is 1
.The total number of buttons being pressed is 1
.The total number of buttons being pressed is 1
.The total number of buttons being pressed is 1
.The total number of buttons being pressed is 1
.The total number of buttons being pressed is 1
.The total number of buttons being pressed is 1
.The total number of buttons being pressed is 1
.The total number of buttons being pressed is 1
.The total number of buttons being pressed is 1
.The total number of buttons being pressed is 1
.The total number of buttons being pressed is 1
.The total number of buttons being pressed is 2
.The total number of buttons being pressed is 2
.The total number of buttons being pressed is 2
.The total number of buttons being pressed is 2
.The total number of buttons being pressed is 2
.The total number of buttons being pressed is 2
.The total number of buttons being pressed is 2
.The total number of buttons being pressed is 1
.The total number of buttons being pressed is 1
.The total number of buttons being pressed is 1
```

Comments:

My greatest takeaway from this lab is an increased awareness of the different types of data that can be collected from the DualShock controller. I had no idea you could measure the acceleration of the controller as well as the number of buttons being pressed. This definitely makes me curious as to what other kinds of data could be collected and utilized in unique kinds of ways in future labs (i.e toggling the back buttons, pressing down the joysticks, connections to different types of devices, etc)

```
/* 185 Lab 3 Template */

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

/* Put your function prototypes here */


int main(void) {
    /* DO NOT MODIFY THESE VARIABLE DECLARATIONS */
    int t;
    double  ax, ay, az;


    /* This while loop makes your code repeat. Don't get rid of it.
*/
    while (1) {
        scanf("%d,%lf,%lf,%lf", &t, &ax, &ay, &az);

/* CODE SECTION 0 */
double t2 = t/1000.0;
        printf("Echoing output: %8.3lf, %lf, %lf, %lf\n", t2, ax,
ay, az);

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

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

    }

return 0;
}

/* Put your functions here */
```

```
/* 185 Lab 3 Template */

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

/* Put your function prototypes here */
double mag(double x, double y, double z);

int main(void) {
    /* DO NOT MODIFY THESE VARIABLE DECLARATIONS */
    int t;
    double  ax, ay, az;


    /* This while loop makes your code repeat. Don't get rid of it.
*/
    while (1) {
        scanf("%d,%lf,%lf,%lf", &t, &ax, &ay, &az);

/* CODE SECTION 0
double t2 = t/1000.0;
        printf("Echoing output: %8.3lf, %lf, %lf, %lf\n", t2, ax,
ay, az);

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

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

    }

return 0;
}

/* Put your functions here */
double mag(double x, double y, double z){
double magg = sqrt((x*x)+(y*y)+(z*z));
return magg;
}
```

```
/* 185 Lab 3 Template */

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

/* Put your function prototypes here */
double mag(double x, double y, double z);
int minutes(int time);
int seconds(int time);
int millis(int time);


int main(void) {
    /* DO NOT MODIFY THESE VARIABLE DECLARATIONS */
    int t;
    double  ax, ay, az;


    /* This while loop makes your code repeat. Don't get rid of it.
*/
    while (1) {
        scanf("%d,%lf,%lf,%lf", &t, &ax, &ay, &az);

/* CODE SECTION 0
double t2 = t/1000.0;
        printf("Echoing output: %8.3lf, %lf, %lf, %lf\n", t2, ax,
ay, az);

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

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

    }

return 0;
}

/* Put your functions here */
double mag(double x, double y, double z){
double magg = sqrt((x*x)+(y*y)+(z*z));
return magg;
}
int minutes(int time){
int mins = time / 6000.0;
```

```
return mins;
}

int seconds(int time){
int secs = (time % 60000)/1000;
return secs;
}

int millis(int time){
int mils = (time % 60000)/1000;
return mils;

}
```

```c
/* 185 Lab 3 Template */

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

/* Put your function prototypes here */
int numButtons(int b1, int b2, int b3, int b4);


int main(void) {
    /* DO NOT MODIFY THESE VARIABLE DECLARATIONS */
    int t;
    double  ax, ay, az;
int a, b, c, d;

    /* This while loop makes your code repeat. Don't get rid of it. */
    while (1) {
        scanf("%d, %d, %d, %d", &a, &b, &c, &d);
printf("The total number of buttons being pressed is %d\n.", numButtons(a, b, c, d));
fflush(stdout);
   }

return 0;
}

/* Put your functions here */
int numButtons(int b1, int b2, int b3, int b4){
    int numButt = b1+b2+b3+b4;
    return numButt;

}
```