

# **Introduction to Functions**

## **Lab 4**

**SUBMITTED BY:**

**CODY BROOKS**

**SUBMISSION DATE:**

**2015-10-01**

## Problem

The purpose of this lab was to create a program which took data from the Esplora and made calculations and outputted it to the display. The program must use several functions and those functions must have prototypes before the **main()** method. The program must also use data that comes from the Esplora via the given **explore.exe** program. The objective of this lab is to create functions, learn how GCC compiles and executes these functions, practice mod and integer division expressions and to practice formatting various types of output.

## Analysis

The problem must be solved by creating prototype functions before the **main()** function and then giving that function functionality after the **main()** function. The program must scan data from the **explore.exe** program using the pipe command. This data may come in the form of 4 columns of floating point numbers representing accelerometer values, or five columns of integers representing the state of each of the Esplora's buttons.

The output of the program in the case of the accelerometer input is text followed by four columns of data. The first column is an integer representing the time in milliseconds formatted to an eight character area with three digits of precision. The second, third and fourth columns of data represented the x, y, and z accelerations (respectively) of the Esplora's accelerometer each formatted to a 7 character area with four digits of precision. These values are each specially formatted outputs of the same values as the inputs.

In the case of the button input, the output is an integer representing the number of buttons pressed simultaneously followed by some text. The outputted integer is the sum of all five columns of input.

## Design

In order to get the **lab4.c** program to read data from the Esplora, **explore.exe** must be piped through the **lab4.c** compiled program. When this happens, **lab4.c** will be able to use **scanf()** to assign data from the Esplora to variables. Once the data has been assigned to their respective variables, the values can be formatted and reprinted to the console in a way that is more readable as well as perform calculations to extend the meaning of the data.

One of these calculations is the magnitude of the acceleration of the Esplora. To do this, a function **mag** is written that takes three double inputs (one for each x, y and z acceleration) and returns a double representing the magnitude of acceleration. The function uses the formula  $\sqrt{x^2 + y^2 + z^2}$  to calculate the magnitude. The time in milliseconds and magnitude is printed to the console using **printf()**.

Another calculation made is converting time in milliseconds to time in minutes, seconds and milliseconds. Three functions are used to accomplish this conversion. One calculates the number of whole minutes elapsed, one calculates the number of whole minutes left and the last calculates the number of milliseconds left. Each function does not report time that has already been reported. For example, if 63230 milliseconds have elapsed, **minutes(63230)** will return 1 and the time remaining is 3230 milliseconds. **seconds(63230)** will return 3 seconds because only 3 whole seconds remained after the number of minutes reported was subtracted leaving 230 milliseconds remaining. **millis(63230)** will return 230 milliseconds because 230 milliseconds remain after 1 minute 3 seconds have been reported.

The last calculation made is the number of buttons simultaneously pressed on the Esplora. Using different flags while executing **explore.exe** will return five columns of

integers. Each column is a 1 or 0 representing whether or not the respective button is currently pressed. To return the total number of buttons simultaneously pressed, **explore.exe** is piped through lab4-4.c. The data is scanned from **explore.exe** via **scanf()** and each column's value is assigned to a variable. The program returns the sum of the values of these variables (an integer representing the total number of buttons pressed) followed by some text explaining what the number means. No functions are needed for this task.

## Testing

To test the **mag(x, y, z)** function, we plugged in the Esplora, found the board's COM port number, and piped **explore.exe** with the appropriate flags through the compiled lab4 program. To verify our results, we did several tests. The first was to leave the Esplora flat on the table to see if two of the three acceleration components were about 0 and the third about -9.8 (the acceleration due to gravity). We then picked up the board and shook it in a single direction. We saw that the harder we shook the board, the larger the respective component value and the larger the acceleration magnitude was. This confirmed that our **mag(x, y, z)** function worked correctly.

To test the **minute()**, **second()** and **millis()** functions, we piped **explore.exe** through our program while the Esplora was plugged in and watched each value as time elapsed. We compared the elapsed time calculated by our program to a stopwatch on our phones to confirm our results.

To test the **lab4-4.c**, we piped **explore.exe** through our program and pressed any combination of buttons. Since the numbers of buttons that we were pressing matched the integer printed to the console in every situation, we knew our program was correct.

## Comments

During this lab, I learned about how the GCC compiler looks at functions and how prototypes allow the developer more flexibility in the way the code is written and ordered. I also learned about how it is important to keep the functionality of all other functions in mind when developing a function. For example, when developing the **second()** and **millis()** functions, I had to remember what kind of result **minute()** would produce and how that output related to the problem I was working on.

```
/* Lab 4 Wrapper Program */

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

#define TRUE 1

/* Put your function prototypes here */

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

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

/* CODE SECTION 0 */
        printf("Echoing output: %5.3d, %3.4lf, %3.4lf, %3.4lf\n", (t / 1000), 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 */
```

[illegible]

```
/* Lab 4 Wrapper Program */

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

#define TRUE 1

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

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

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

/* CODE SECTION 0
        printf("Echoing output: %5.3d, %3.4lf, %3.4lf, %3.4lf\n", (t / 1000), 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 magnitude = sqrt(pow(x, 2) + pow(y, 2) + pow(z, 2));
    return magnitude;
}
```

At 218909 ms, the acceleration's magnitude was: 0.973009  
At 218911 ms, the acceleration's magnitude was: 0.987251  
At 218913 ms, the acceleration's magnitude was: 0.960301  
At 218916 ms, the acceleration's magnitude was: 0.991983  
At 218918 ms, the acceleration's magnitude was: 0.999988  
At 218920 ms, the acceleration's magnitude was: 1.087952  
At 218924 ms, the acceleration's magnitude was: 1.100086  
At 218926 ms, the acceleration's magnitude was: 1.110186  
At 218928 ms, the acceleration's magnitude was: 1.155636  
At 218931 ms, the acceleration's magnitude was: 1.124562  
At 218933 ms, the acceleration's magnitude was: 1.100616  
At 218935 ms, the acceleration's magnitude was: 1.100545  
At 218937 ms, the acceleration's magnitude was: 1.137576  
At 218940 ms, the acceleration's magnitude was: 1.154926  
At 218942 ms, the acceleration's magnitude was: 1.156570  
At 218944 ms, the acceleration's magnitude was: 1.178695  
At 218947 ms, the acceleration's magnitude was: 1.200629  
At 218949 ms, the acceleration's magnitude was: 1.166326  
At 218951 ms, the acceleration's magnitude was: 1.189078  
At 218954 ms, the acceleration's magnitude was: 1.232474  
At 218956 ms, the acceleration's magnitude was: 1.252648  
At 218958 ms, the acceleration's magnitude was: 1.252380  
At 218960 ms, the acceleration's magnitude was: 1.258532  
At 218963 ms, the acceleration's magnitude was: 1.264386  
At 218966 ms, the acceleration's magnitude was: 1.300710  
At 218968 ms, the acceleration's magnitude was: 1.282040  
At 218971 ms, the acceleration's magnitude was: 1.324639  
At 218973 ms, the acceleration's magnitude was: 1.290505  
At 218975 ms, the acceleration's magnitude was: 1.284073  
At 218977 ms, the acceleration's magnitude was: 1.276134  
At 218980 ms, the acceleration's magnitude was: 1.224283  
At 218982 ms, the acceleration's magnitude was: 1.215235  
At 218984 ms, the acceleration's magnitude was: 1.170462  
At 218987 ms, the acceleration's magnitude was: 1.134349  
At 218989 ms, the acceleration's magnitude was: 1.107572  
At 218991 ms, the acceleration's magnitude was: 1.071345  
At 218993 ms, the acceleration's magnitude was: 1.012673  
At 218996 ms, the acceleration's magnitude was: 0.986218  
At 218998 ms, the acceleration's magnitude was: 0.930937  
At 219000 ms, the acceleration's magnitude was: 0.898207  
At 219003 ms, the acceleration's magnitude was: 0.890244  
At 219005 ms, the acceleration's magnitude was: 0.867655  
At 219008 ms, the acceleration's magnitude was: 0.867655  
At 219011 ms, the acceleration's magnitude was: 0.875184  
At 219013 ms, the acceleration's magnitude was: 0.944117  
At 219015 ms, the acceleration's magnitude was: 0.982212  
At 219017 ms, the acceleration's magnitude was: 1.165264  
At 219020 ms, the acceleration's magnitude was: 1.104431  
At 219022 ms, the acceleration's magnitude was: 1.048972  
At 219024 ms, the acceleration's magnitude was: 1.115735  
At 219027 ms, the acceleration's magnitude was: 0.981664  
At 219029 ms, the acceleration's magnitude was: 1.118547  
At 219031 ms, the acceleration's magnitude was: 1.143507  
At 219033 ms, the acceleration's magnitude was: 1.078236  
At 219036 ms, the acceleration's magnitude was: 1.069490  
At 219038 ms, the acceleration's magnitude was: 1.038928  
At 219040 ms, the acceleration's magnitude was: 0.900870  
At 219043 ms, the acceleration's magnitude was: 0.862764  
At 219045 ms, the acceleration's magnitude was: 0.867357  
At 219047 ms, the acceleration's magnitude was: 0.765146  
At 219051 ms, the acceleration's magnitude was: 0.827998

```

/* Lab 4 Wrapper Program */

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

#define TRUE 1

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

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

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

/* CODE SECTION 0
        printf("Echoing output: %5.3d, %3.4lf, %3.4lf, %3.4lf\n", t, 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));

        fflush(stdout);
    }

return 0;
}

/* Put your functions here */
double mag (double x, double y, double z) {
    double magnitude = sqrt(pow(x, 2) + pow(y, 2) + pow(z, 2));
    return magnitude;
}

int minutes (int time) {
    int minsLeft = time / 60 / 1000; // Converts ms to minutes
    return minsLeft;
}

int seconds (int time) {
    int secsLeft = time / 1000;
    int minsUsed = minutes(time) * 60; // Converts minutes already used to seconds
    secsLeft -= minsUsed; // Discounts time already reported
    return secsLeft;
}

```



---

}

```
int millis (int time) {  
    int msLeft = time;  
    int minsUsed = minutes(msLeft) * 60 * 1000; // Converts minutes already reported to  
    milliseconds  
    msLeft -= minsUsed; // Discounts time already reported  
    int secsUsed = seconds(msLeft) * 1000; // Converts seconds already reported to milliseconds  
    msLeft -= secsUsed; // Discounts time already reported  
    return msLeft;  
}
```

[illegible]

```
/* Lab 4 Wrapper Program */

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

#define TRUE 1

/* Put your function prototypes here */

int main(void) {
    int a, b, x, y, joystick, buttonsPressed;

    while (TRUE) {
        scanf("%d, %d, %d, %d, %d,", &a, &y, &x, &b, &joystick);
        buttonsPressed = (a + b + x + y + joystick);
        printf("%d buttons are being pressed\n", buttonsPressed);
        fflush(stdout);
    }

    return 0;
}

/* Put your functions here */
```

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



0 buttons are being pressed  
0 buttons are being pressed  
0 buttons are being pressed  
0 buttons are being pressed  
0 buttons are being pressed  
0 buttons are being pressed