

CS 271 Lab 1 - C Basics

Goals:

- Learn to use Kate editor to create and debug C programs.
- Learn to use gcc compiler with math library and one source file.
- Learn Linux commands: `cd`, `pwd`, `cp`, `dir`, `mkdir`, `mv`, and `rm`
- Write a C program that uses assignment, input, output, selection, and repetition.

Note on grading:

Part of the points (12%) will be for demonstrating to a TA in the lab that you know how to use gcc and that your program1 is working (see step 7 in the instructions). You have to be attending the lab to do it.

Step by Step Lab Instructions

1. Log in. Open the Kate editor.

To make the Terminal window visible in Kate, do as follows.

Go to the Settings menu and choose

- ➔ Configure Kate
- ➔ Plugins
- ➔ check the box next to Terminal Tool View
- ➔ then click OK.

You may also set C-style syntax highlighting by going to the Tools menu and choosing

- ➔ Highlighting
- ➔ Sources
- ➔ C

And set C-style indentation by going to the Tools menu and choosing

- ➔ Indentation
- ➔ C Style.

(If you haven't used Linux before, refer to the "Essential Linux Commands" handout.)

2. **Move the cursor to the Terminal Window:** Click in the terminal window area at the bottom of Kate.

- Type `pwd` and press Enter. Note the directory information. Type `cd` and press Enter.
- Type `pwd` again and note the difference in the directory information (if any).
- Type `mkdir cs271` and press Enter.
- Change directory to `cs271` by typing `cd cs271`.
- Type `mkdir Lab1` and press Enter.
- Type `cd Lab1` to change directory to `Lab1`. This is where you will save your files.

3. **Editor:** To practice using Kate, type in the program shown below.

Reminder: In the terminal window, you must set your working directory to the folder where you saved your program. Each time you log in, you must set your working directory again.

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

// The purpose of this program is to calculate the length of hypotenuse
// of a right triangle.

int main (void) {

    float leg1, leg2, hypotenuse;

    // input the lengths of leg1 and leg2

    printf("Enter the length of side A\n");
    scanf("%f", &leg1);
    printf("\nEnter the length of side B: \n");
    scanf("%f", &leg2);

    // calculate the hypotenuse

    hypotenuse = sqrtf( leg1 * leg1 + leg2 * leg2 );

    // display the length of the hypotenuse with 2 decimal places

    printf("The hypotenuse is %.2f\n", hypotenuse);

} // end main function
```

Save the file as program1.c in the cs271/Lab1 directory.

4. Terminal: In the Terminal window, compile the program and produce an executable file named program1. Here's the command:

```
gcc program1.c -lm -o program1
```

"flags" or options for the compiler are preceded by a hyphen -.

-o must appear immediately before the name of the executable

-lm (an "el", not a one) means that the math library is needed

If you have syntax errors, go back to the editor window and fix them. Compile again and debug until the program compiles correctly.

5. Terminal: Run the program by typing a period, a slash, then the name of the executable:

```
./program1
```

Test the program by entering 3.0 and 4.0 for the legs. The output should be 5.00. If it's not, you have some debugging to do.

Run the program again. Test with values 5.0 and 8.0. (Determine what the expected output is **before** you run the program.)

6. Editor: Close program1.c.

7. (12% of the grade): Demonstrate to the TA or instructor that you know how to use gcc and that your program1 is working.

8. Editor: Write a C program (call it lab1.c) that will solve the following problem. Make sure that your program satisfies [Programming Documentation and Style Requirements](#).

An instructor wants to check the progress for each student in his class. The students have taken 4 exams and each exam score is an integer number between 0 and 100 (inclusive).

The instructor wants to see numeric average of the 4 scores and the letter grade that corresponds to the exam average. Here is the grading scale:

0 <= average < 60	F
60 <= average < 70	D
70 <= average < 80	C
80 <= average < 90	B
90 <= average < 100	A

For each student, your program should do the following:

- prompt the instructor to enter 4 exam scores,
- read and echo print the 4 exam scores,
- calculate and print the average with 2 decimal places,
- print the letter grade as a capital letter.

The program should repeat the process of reading 4 exam scores, calculating and printing the average and letter grade, until the instructor enters numbers -1 0 0 0 for the exam scores. (Note: the score sequence -1 0 0 0 is called the "sentinel".)

Do not display the sentinel in the output.

Sample run of the program may look like the following:

```
cs271/Lab1> ./lab1
Enter the four scores: 70 80 90 80
```

```
Exam scores:   70   80   90   80
Average:       80.00
Letter grade:  B
```

```
Enter the four scores: 65 71 73 68
```

```
Exam scores:   65   71   73   68
Average:       69.25
Letter grade:  D
```

```
Enter the four scores: -1 0 0 0
```

9. Terminal: Compile, debug, and compile again until the program compiles correctly.

```
gcc lab1.c -o lab1    to compile
./lab1                to run
```

Debug as needed until the program runs correctly with 2 students.

10. Canvas: Download the file "testdata.dat" from this assignment page in Canvas. Save the file as a plain text file in the same folder as your lab1.c program.

11. Terminal: Run the program again... this time using "input redirection".

This will take all input from the file you specify (instead of the keyboard).

```
./lab1 < testdata.dat
```

You can open the data file in your editor and determine the expected output of the program. Compare the expected output to the actual program output.

Close the program and data file once you have the correct output.

12. Submit lab1.c on Canvas

Log off before you leave the lab.