W2020. LAB 1 (Jan 10-19)

Introduction to C, Function Declaration vs. Definition, Basic I/O (scanf/printf, getchar/putchar, input/output redirection)

Due: Jan 19 (Sun), 11:59 pm.

---

problem 0 printf() in Java

Specification

While this is a course on C programming, let's start our very first lab with a JAVA question. Download the partially implemented Java program `Lab1.java` . This program reads a name and an integer from the user, then outputs the double and triple values of the entered integer.

Implementation

- First, complete the `println()` statement, so that it produces the output as shown below. Don't add new variables. Do all the formatting in `println()`. Probably you need to use string concatenations + multiple times in `println()`.
- Next, complete the `printf()` statement so that it produces the same output, as shown below. Try to do it without using string concatenations, i.e., don't use +. Again, don't introduce new variables. Do all the formatting in `printf()`.

Objectives

The purpose of this exercise is for you to:

- be aware that `printf` also exists in JAVA (actually JAVA borrows the idea from C)
- learn the syntax of `printf` in JAVA and C
- observe that sometimes using `printf` can be easier than `println`, and the code is more readable (IMHO)
- be aware that Java programs can be compiled and run in command-line.

Sample Inputs/Outputs: (In a terminal, navigate to the directory where the Java file is located, e.g., if your file is downloaded into folder **`lab1`** of Desktop, navigate to the directory by issuing **`cd Desktop/lab1`)**

red 300 % is the terminal prompt if you login to the red server.
If you are in Prism lab, your prompt will be **ea xx %** where xx is the terminal id.

compile Java program in command_line

run Java program in command_line

```
red 300 % javac Lab1.java
red 301 % java Lab1
Please enter the name: Judy
Please enter the number: 22
Hi Judy, double and triple of input 22 is 44 and 66, respectively.
Hi Judy, double and triple of input 22 is 44 and 66, respectively.
red 302 % java Lab1
Please enter the name: Joe
Please enter the number: 100
Hi Joe, double and triple of input 100 is 200 and 300, respectively.
Hi Joe, double and triple of input 100 is 200 and 300, respectively.
```

Submission: In a terminal, navigate to the directory where your file is located, and then issue the command **`submit 2031M lab1 Lab1.java`**

1 problem A scanf(), printf() in C

Specification

Download program `scanf2.c`, which takes two inputs from *standard input* (default: keyboard), and output the sum of the two numbers to *standard output* (default: screen). (This is the program shown in class and slides). Read the code and observe that,

- you specify what the input should "look like" in the formatting string of `scanf`. Here the formatting string "`%d %d`" indicates that `scanf` expects two integers separated by a blank.
- to store the input values into variable `a` and `b`, you need to use `&a` and `&b` as additional arguments of `scanf`. We will explain later why `&` is needed.
- `printf` also uses `%d` for integer conversion. A conversion specification `%d` in the formatting string will be replaced/filled by a value of type `int`, which can be either an integer constant such as 32, or an `int` variable, or a function call which returns an `int`.
- there is a `\n` at the end of the formatting string of the last printf statement.

Compile and run the program with different inputs. Also try to delete the `\n` in the formatting string of the last `printf` statement, see what happens.

Next, modify the program so that it takes as its input two integers separated by <><><>.

Sample input and outputs

```
red 306 % gcc scanf2.c
red 307 % a.out
Enter two integers separated by <><><>: 4<><><>32
Entered 4 and 32, Sum is 36.
red 308 % a.out
Enter two integers separated by <><><>: 40<><><>302
Entered 40 and 302, Sum is 342.
```

Try to enter **`4<><>32`** or **`4  32`**, and observe that `b` does not get the value 32.
Finally, remove the `&` before `a` or `b`, then compile and run the program again. The program compiles but crashes, generating error message `Segmentation fault (core dumped)`. We will explain this later.
You don't need to submit anything for this question but doing this gets you better prepared for the following questions.

2 Problem B. scanf(), printf() in C

Specification

Write an ANSI-C program that reads input from the *standard input* (keyboard), and then outputs the reformatted versions of the input to *standard output* (screen).

Implementation

- name your program `lab1B.c`
- use `scanf()` to read input (from standard input), which are in the form of `Month Day Year` (i.e., three integers separated by white spaces).
- use `printf()` to generate output in the format of `Year/month/day` and `Year-month-day`
- display the following prompt (leave a white space after the colon)
  `Enter month, day and year separated by spaces:`

display output as follows
`The input 'm d y' is reformatted as y/m/d and y-m-d`

- Note: you should do the reformatting only within `printf()`. In particular, you should use at most three variables in your program, and feed them into `printf()` judiciously.

Sample Inputs/Outputs:

```
red 306 % gcc lab1B.c
red 307 % a.out
Enter month, day and year separated by spaces: 3 20 2019
The input '3 20 2019' is reformatted as 2019/3/20 and 2019-3-20
red 308 % a.out
Enter month, day and year separated by spaces: 1 16 2020
The input '1 16 2020' is reformatted as 2020/1/16 and 2020-1-16
red 309 %
```

Submit your program by issuing **`submit 2031M lab1 lab1B.c`**

3 Problem C. Functions in C

Download the program `lab1C.c`, compile it using **`gcc lab1C.c`**
Observe that the compilation process fails (why?), and `a.out` is not generated.

Modify the program to make it compile. Note that you should not modify or move the existing code. That is, **do not modify the code of main() and sum(), and also do not move the functions.** Instead, add something to make the program compile.
Then, run the program to see the output.

Submit your program by issuing **`submit 2031M lab1 lab1C.c`**

4 Problem D. Functions, scanf(), printf(), floats

Specification

Improve program `lab1C`, so that it can read two float numbers from the standard input, separated by two pound (#) signs, and then output the sum of the two float numbers to standard output.

Implementation

- name your program `lab1D.c`
- use `scanf()` to read inputs (from standard input), which are in the form of `float1##float2` (i.e., two float numbers separated by two pound signs).
- use `printf()` to generate output to the standard output

Sample Inputs/Outputs: (note that by default `printf()` displays six digits after decimal points of a floating point number.)

```
red 338 % gcc lab1D.c  -o lab1D
red 339 % lab1D
Enter two float numbers separated by ##: 2.35##5.64
2.350000 + 5.640000 = 7.990000
red 340 %
```

Submit your program by issuing **`submit 2031M lab1 lab1D.c`**

5 Problem E. Simple loops

Specification

Extend program `lab1D.c` above, in such a way that it first prompts the user to enter an integer number, which indicates how many times the user wants to interact with the program. Then the program interacts with the user accordingly.

Implementation

- name your program `lab1E.c`
- use a loop (`for` or `while`) to interact (i.e., read input and generate output) `n` times, where `n` is a positive number entered by the user.

Sample Inputs/Outputs: (ONE blank line between each interaction/iteration):

```
red 338 % gcc lab1E.c  -o lab1E
red 339 % lab1E
Enter the number of interactions: 3

Enter two float numbers separated by ##: 2.35##5.64
2.350000 + 5.640000 = 7.990000

Enter two float numbers separated by ##: 1.1##2.2
1.100000 + 2.200000 = 3.300000

Enter two float numbers separated by ##: 2.5##6
2.500000 + 6.000000 = 8.500000
red 340 %
```

Submit your program by issuing **`submit 2031M lab1 lab1E.c`**

6. Problem F0. getchar, putchar, input/output redirection

6.1 Specification

Download the provided program `countChar.c`, which uses function `getchar()` to read user input from *standard input* (default: keyboard) and count the total number of characters in the input, and output to *standard output* (default: screen). (This program is also in the lecture slides.) Play with the program and make sure you understand the program. In particular, observe a few things about `getchar()`:

- `getchar()` reads characters from standard input (stdin), which by default is the keyboard. But the input can be redirected (substituted) from an input file using `< filename`. In the latter case `getchar()` will read from the input file instead.
- `getchar()` returns **EOF** (which is a special negative integer number defined in C) when the "*end of file*" is reached.
  - If the program reads from a text file (redirected using `<`), then the end of the text file is "*end of file*";
  - If the program reads from default standard in (i.e., keyboard) , then in Unix, `ctrl D` indicates "*end of file*" (in Windows, it is `ctrl Z`)
- Instead of a `char`, function `getchar` returns an `int`. This will be explained later in class.

6.2 Sample Inputs/Outputs (from standard input - keyboard):

```
red 308 % gcc countChar.c
red 309 % a.out
```

```
hello
how are you
I am good
^D (press Ctrl and D)
# of chars: 28
red 310 %
red 311 % a.out
hello
how are you?
I am good and thanks!
^D (press Ctrl and D)
# of chars: 41
red 312 %
```

6.3 Sample Inputs/Outputs (use redirected input/output files):

All the program run so far take inputs from *standard in (stdin)* which by default is the keyboard, and write output to *standard out* (stdout) which by default is the screen.

You can always redirect the standard in (keyboard) from an input file using <
You can always redirect the standard out (screen) to an output file using >

Download (**don't copy and paste**) file `greetings.txt`, whose content is

```
hello
how are you
I am good
```

In prism lab, right-click the filename and select "Save link as"

```
red 313 % a.out < greetings.txt
# of chars: 28
```

This time the program does not ask you to enter anything, because standard input is redirected/substituted from a text file, so program reads inputs from file `greetings.txt`.

```
red 314 % a.out > output.txt
hello
how are you
I am good
^D (press Ctrl and D)
red 315
```

This time the program reads inputs from user (keyboard), but nothing was generated on the standard out (screen), because all outputs are redirected to a text file using `>` .
Now a new file `output.txt` should be generated (in the current directory). Use command `ls` or `ls -l` to confirm this. Then use command `cat` or `more` to view the content of `output.txt` (If you don't know what is happening here, please review the CSE1020 Guided Tour or the Unix tutorial posted on the course website.)

```
red 316 % ls -l
red 317 % cat output.txt
# of chars: 28
```

Finally try

```
red 318 % a.out < greetings.txt > output2.txt
```

```
red 319 %
```
This time both the standard input and standard out are redirected. Thus the program reads inputs from `greetings.txt`, and write outputs to `output2.txt`. Check the content of file `output2.txt`.
Also in the terminal, issue `cal` or `date` to view the output, and then issue `cal > temp.txt` or `date > temp.txt` to see how the output is redirected to a file .
You don't need to submit anything for this question but doing this gets you prepared for the next questions.

7. Problem F2 getchar(), character comparison

Specification

The provided program uses `getchar()` to read input character by character, counting the number of characters from the standard input (keyboard or redirected from an input file). Modify the program so that it also counts the number of character '`a`' in the input.

Implementation

- Name your program **`countChar2.c`**
- Hint: you may want to compare every character `getchar` reads in against the character '`a`'. In Java or C, how to compare two characters?

Sample Inputs/Outputs

```
red 307 % gcc countChar2.c -o cc2
red 308 % cc2
hello
how are you?
I am good and thanks!
^D (press Ctrl and D)
# of chars:  41
# of char 'a': 4
red 309 % cc2
hello
how are you
I am good
^D (press Ctrl and D)
# of chars:  28
# of char 'a': 2
red 310 % cc2 < greetings.txt
# of chars:  28
# of char 'a': 2
red 311 %
```

Submit your program by issuing **`submit 2031M lab1 countChar2.c`**

7. Problem F3 getchar(), character comparison, special chars

Specification

Each input line to the above programs, either from keyboard or from the text file, ends with an invisible new line character. The new line characters are counted in the above programs.

Modify the program `countChar` so that it counts only visible characters, i.e., new line characters are not counted.

Implementation

- Name your program **`countChar3.c`**
- Hint: you may want to compare every character `getchar` reads in against the new line character. In Java or C, how is the new line character represented?

Sample Inputs/Outputs

```
red 318 % gcc countChar3.c -o countChar3
red 319 % countChar3
hello
how are you?
I am good and thanks!
^D (press Ctrl and D)
# of chars:  38
red 320 % countChar3
hello
how are you
I am good
^D (press Ctrl and D)
# of chars:  25
red 321 % countChar3 < greetings.txt
# of chars:  25
red 322 %
```

Submit your program by issuing **`submit 2031M lab1 countChar3.c`**

8. Problem F4 getchar(), character comparison, special chars

Extend the program **`countChar3.c`** so that it also counts the number of lines in the input. Name your program **`countChar4.c`**

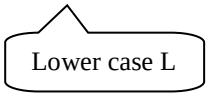
```
red 308 % gcc countChar4.c
red 309 % a.out
hello
how are you
I am good
^D (press Ctrl and D)
# of chars:  25
# of lines:  3
red 310 % a.out < greetings.txt
# of chars:  25
# of lines:  3
red 311 % a.out < greetings.txt > output4.txt
red 312 % cat output4.txt
# of chars:  25
# of lines:  3
```

Submit your program by issuing **`submit 2031M lab1 countChar4.c`**

In summary, for this lab you should submit the following files:

`Lab1.java lab1B.c lab1C.c lab1D.c lab1E.c countChar2.c countChar3.c countChar4.c`

At any time and from any directory, you can issue **`submit -l 2031M lab1`** to view a list of files that you have submitted for lab1.



Also note that you can submit the same file multiple times. Then the latest file will overwrite the old one.

Common Notes

All submitted files should contain the following header:

```
/**************************************
* EECS2031M – Lab1 *
* Author: Last name, first name *
* Email: Your email address *
* eecs_username: Your eecs login user name *
* York Student #: Your student number
**************************************/
```