Mechanics of Programming CSCI.243

Homework 2: Reading CSV Files

Due Date: 6/27/2017 at 11:59pm

---

1. Overview

Two of the most fundamental programming tasks are to read and write formatted data. The degree of complexity of the data formats can vary wildly. The command 'cp' simply moves data byte-by-byte with no real formatting. The command 'grep' on the other hand operates line-by-line. The command 'cut' operates field-by-field. You will write a program to process a particularly common type of structured data: comma separated values (CSV).

---

2. Revision History

- 1.0 Initial version
- 1.1 Feb 3 @ 8:23am -- lowered maximum lines from 4096 to 1024.
- 1.2 Feb 3 @ 9:48am -- emphasized required precision for mean.

3. Reading Textual Data

It is recommended to process the input using `fgets` or `getchar`. If you are already familliar with the C function `scanf` you may use it to complete this assignment. If you would like to try to use it, please be cautious. To truly understand `scanf` you must also understand pointers, which will not be covered until later. You will need the function `strtod` to convert the population from a string to a double-precision number. You may read the entire file into a large 2D array, but that is not necessary. It is possible to do this in a single while loop with `fgets`. It is possible to solve this in a nested while loop with `getchar`. You will need to understand character arrays and array indexing.

4. Parsing The Data

You should consider how to break up the data by searching across character arrays for the ',' character. You should focus on the low-level aspects of your program, such as how you transform a string into a number, and how to avoid going out of bounds. *You are encouraged to hard-code the CSV parsing component. You are not required to implement a general CSV parser -- that would be too much to ask for the second homework assignment!*

5. The Problem

We are interested in a form of summary statistics for the dataset. Your objective is to write a program which accepts data in the form supplied above from standard in, and produces a report containing the number of zip codes, the total population, and the mean population by zipcode. Note that the mean population should be printed with exactly one digit past the decimal. The output for the sample csv file should be:

```
ZipCodes=50
Population=1750367
MeanPopByZip=35007.3
```

Please take a look at the following four pairs of sample input and output. You may be prompted for a program to read the files -- we recommend viewing them as plain text files using a text editor such as `vi`, `emacs`, `gedit`, or `notepad`. If you notice any discrepancies, please report them to your instructor ASAP.

INPUT	*OUTPUT*
sample0.csv	output0.txt
sample1.csv	output1.txt
sample2.csv	output2.txt
sample3.csv	output3.txt

You may also copy the above files to your local directory via get:

get csci243 hw2

You may make the following assumptions:

- No line will be longer than 256 characters.
- There will be at most 1024 lines of input.
- The input is correctly formatted (e.g., number and type of columns).
- The input begins with a header line, which you may wish to ignore.

6. Acknowledgements

The census dataset was obtained from data.gov.

7. Program Structure

You should break your program up into two files:

- readcsv.h (function declarations (if any), #define'd limits)
- readcsv.c (function definitions go here, main goes here)

8. Compilation

After the first homework, and our discussion of compiling and linking, you are probably ready to compile programs without help. Just in case you would like a refresher, one way to compile your program is to run the command:

```
gcc -Wall -Wextra -std=c99 -pedantic readcsv.c -o readcsv
```

WARNING: If you mistype that line you can overwrite and permanently delete your source file! Make backups or use version control.

9. Execution

Once you have compiled your program into an executable file named `readcsv`, you may run it using either of the following methods. Both rely on reading a file by redirection through standard input (`stdin`)

- ./readcsv < sample.csv
- cat sample.csv | ./readcsv

10. Grading

This assignment is worth a total of 100 points, distributed as follows:

- 20 You submitted a non-trivial program that compiles without error.
- 20 Your header file has your real name, user name / email address, and a high-level description of your program. In addition, you must document each of the functions you have written. This is intended to help your grade. Please note that if you do not do this, it will incur a painful 20 point penalty.
- 10 Your program's output is correctly formatted. (I.e., exactly three lines, no spaces.)
- 50 Your program's numerical output is correct.

11. Program Submission

Submit your program via try:

```
try grd-243 hw2-1 readcsv.h readcsv.c
```
