Parsing and File I/O (A5)

2/10/2023

82/100 Points

Attempt 1

Review Feedback
2/10/2023

Attempt 1 Score: **82/100**

View Feedback

Anonymous Grading: **no**

Unlimited Attempts Allowed

Details

The purpose of this lab is to familiarize you with the C++ process of parsing and file input/output through the basics of the **fstream** library. The **getline** function will be used to manipulate large pieces of data into progressively smaller chunks. Together with the **stringstream** function, you can swap back and forth to 'tokenize' a large line of input and control where each piece goes.

Your job is to write a program that takes an expected input file named "input.txt" of this format:

(int),(int),(int)
(string)
(int),(int),(int)
(string)
(int),(int),(int)
(string)
(int),(int),(int)
(string)
(int),(int),(int)
(string)

- There will be a line with exactly three integers, separated by commas with no spaces
- There will be no trailing comma on the integer lines
- The next line will be a string. It might be a single word, or it may be an entire phrase or sentence
- There might be any number of lines, but they will always come in that pattern
- There will not be any missing lines or values
- You can count on the integers all being legitimate integer values

Your program should then sum the ints of each line and output the immediately following line's string, that many times, seperated by commas such that:

1,2,3
ham

becomes:

ham,ham,ham,ham,ham,ham

NOTE: you may add a trailing comma to the end of the line if that is easier for you.

You might want to use a stringstream for data-handling. Recall that the getline function allows you to set a delimiter, but it only returns string values.
You may also find the **atoi()** and **.c_str()** functions useful.

Your final output format should not be the console. Instead, your program should open a second file and write your output to a file named "output.txt" which should be five lines of comma-seperated words.

Your assignment must include a makefile that follows the assignment requirements from OOP Horses. It must include a make run command.

Do not use namespace std. It will lose you points

Algorithm

As always, begin with an algorithm in markdown. This document should begin by describing the main purpose of the program using the Goals - Input - Output - Steps technique we've been using. For EACH function/method you have in the program or classes, you must go through the same process (Defining goals, input, output, steps). Remember that your algorithm should be complete enough that by the time you are ready to start writing code, you pretty much know what you are going to do. You should get mostly through the algorithm step together in the recitation center, but definitely try to write this on your own as well.

Your algorithm file must be in a .md file named: algorithm.md

Points will be taken off if it is not named this exactly, or if it is in another file type (including but not limited to; .rtf, .docx, .doc, .pdf, ... etc.)

Turning in the project

- This assignment will be turned in through iu github. Please name your repo CSCI24000_spring23_A5
- If your repo has another name, it will not be graded.
- After you've submitted your assignment in Github, come back to this assignment page and submit the full Github URL (**https://github.iu.edu/username/reponame (https://github.iu.edu/username/reponame)**) for your repo here.
- Your code must be placed in a folder base for your base assignment. And if you have a blackbelt it must be placed in a folder called blackbelt.
- Your repo must be private. We will not grade projects in public repos, and we will require you to take down any homework assignments in a public repo.
- Please ensure to add all the following collaborators:
- **Git Collaborators (https://iu.instructure.com/courses/2131288/pages/assignments-submission-and-grading)**

This is how we will be finding your code, so if you do not add the collaborators, your code will not be graded.

Note that your code will be tested with an online plagiarism tool. Please DO NOT turn in work that is not yours. We will know, and we will be displeased.

Black Belt Extension

In a problem like this, you don't always control the input file. To improve your program think about 'edge cases'

- What if the input file has a line of three zeros?
- What about negative numbers? floating point values?
- What if there are too few or too many numbers?
- What if the text line is blank?

Of course you can go crazy and improve the program in other ways if you wish.

---

https://github.iu.edu/parmsing/CSCI24000_spring23_A5

(https://iu.instructure.com/courses/2131288/modules/items/28790599)

(https://iu.instructure.com/courses/2131288/modules

You are unable to submit to this assignment as your enrollment in this course has been concluded.