Due date: Jan 31st 11:59:59 PM

COSC2436 Hw1: Recursion and Arrays

V1.0

Created by Kunpeng Zhang (kzhang21@uh.edu)

1. Purpose of this homework

1) You will learn how to use ArgumentManager.h to parse command line.
2) You will learn how to test your homework.
3) You will practice recursion.

2. Introduction

You need to create a C++ program that can read data from input file, process it according to the requirement and write the result to a file. (You can learn how to read and write files from hw0 code.)

3. Input file, command file and Output file

1) The input file is a text file (txt file), where each line is terminated with an '\n' (except for the last line). One line may contain useful data or just empty. If one line contains useful data, you need to parse the useful data and store them into an array for further use.
2) The command file is a text file too. It contains the desired data.
3) You need to write your results to the output file.
4) All data should be processed sequentially from the beginning to the end.

4. Specification and examples

Your C++ source codes will be compiled to the executable to be tested by the TAs. The result should be written to another text file (output file) whose name is provided by the command line. Note that the input and the output files are specified in the console commands, not hardcoded in the C++ code. Also notice the quotes in the program call, to avoid Unix/Windows get confused. It is recommended to test your code on replit.com before submission.
The general call to the executable is as follows in Linux:

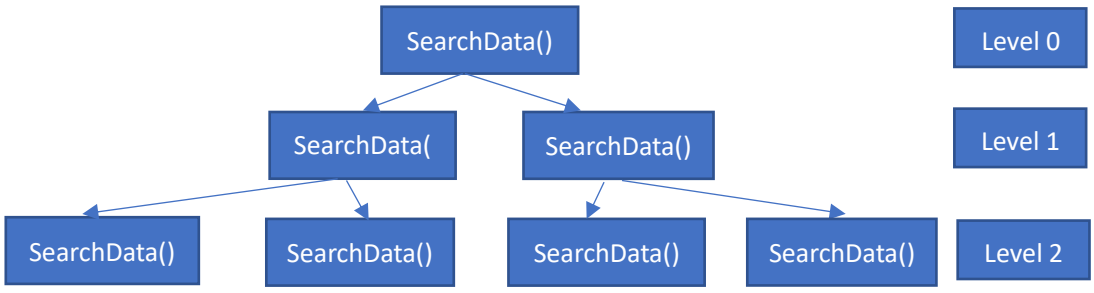
./arrayRec “input=input11.txt;command=command11.txt;output=output11.txt”
Call example with another console command type.
./arrayRec input=input11.txt command=command11.txt output=output11.txt
You can use the given ArgumentManager.h to parse the command or do your own argument parser with similar functionality.

Assumptions:

• The input file contains a small number of records (say < 10000 records); no need to handle binary files. All records stored in similar format (e.g. element order, delimiter character), refering examples for details. You can assume all records have correct format without missing sections. For each record, the useful info only included inside the paired curly braces. There will be no character outside the curly braces except for the comma which used for separating records.

• The records only contain numbers, alphabets, dots(.), colons(:), comma(,) and curly braces({}). The Input file may have spaces but should be ignored.

• The command file contains the target data. Every time when you call your function recursively, you should divide the record array into 2 parts. For example, if you have 4 input records, the first level (Level 0) when you call the function, you should spawn two functions and pass each of them 2 records. In the second level (Level 1) when you call the function, you should spawn two functions and pass each of them one record. The figure below shows the process. The function should compare the record with the target data only when one record was passed in, otherwise, the function should spawn 2 functions and pass each of them with half of the array. (If the number is odd, you should pass the first function with more records. The first half records should be passed into function first.)



• Output data has two parts including 1. the level number where you found the record and 2. The whole record which contains the target data. The level section in front. All output data should be written without leading spaces, each line contains only one record. When you cannot find the target data, you should create an empty file and name it with the output file name.

Example 1

Input file name: input11.txt
{name:Jone. Rock,age:25,dept:CS},{na me:Alice.Sun,age:46,dept:CS},
{name:Asen.Huan,age:30,dept:EE},{name:Bob.Carlos,ag
e:20,dept:EE}, {name:Am aya.Jeremy,age:24,dept:ISS
SO},{name:Kim.Jose,age:34,dept:Edu}

Command file name: command11.txt
name:JONE.ROCK

Linux Command:
./arrayRec "input=input11.txt;command=command11.txt;output=output11.txt"

Output file name: output11.txt
Level 3: {name:Jone.Rock,age:25,dept:CS}

Example 2

Input file name: input12.txt
{name:Jone. Rock,age:25,dept:CS},{na me:Alice.Sun,age:46,dept:CS},
{name:Asen.Huan,age:30,dept:EE},{name:Bob.Carlos,ag
e:20,dept:EE}, {name:Am aya.Jeremy,age:24,dept:ISS
SO},{name:Kim.Jose,age:34,dept:Edu}

Command file name: command12.txt
dept:CS

Linux Command:
./arrayRec input=input12.txt command=command12.txt output=output12.txt

Output file name: output12.txt
Level 3: {name:Jone.Rock,age:25,dept:CS}
Level 3: {name:Alice.Sun,age:46,dept:CS}

Example 3

Input file name:input13.txt
{name:Jone. Rock,age:25,dept:CS},{na me:Alice.Sun,age:46,dept:CS},
{name:Jone. Rock,age:28,dept:EE},{name:Bob.Carlos,ag
e:20,dept:EE}, {name:Am aya.Jeremy,age:24,dept:ISS
SO},{name:Kim.Jose,age:34,dept:Edu}

Command file name: command13.txt
name:Jone.Rock

Linux Command:
./arrayRec input=input13.txt command=command13.txt output=output13.txt

Output file name: output13.txt
Level 3: {name:Jone.Rock,age:25,dept:CS}
Level 2: {name:Jone.Rock,age:28,dept:EE}

5. Requirements

• Homework is individual. Your homework will be automatically screened for code plagiarism against code from the other students and code from external sources. Code that is copied from another student (for instance, renaming variables, changing for and while loops, changing indentation, etc, will be treated as copy) will be detected and result in ”0” in this homework. The limit is 50% similarity. Here are some previous homework which been found copy each other (the main function has been deleted).

• Timeout is set to 2s (If your code doesn’t finish within 2s for certain test case, the test case fails).

6. Turn in your homework

Homework 1 need to be turned in to our Linux server, follow the instruction here https://rizk.netlify.app/courses/cosc2430/4_homework_grading/ for more information.

1. Make sure to create a folder under your server’ root directory, name it hw1 (case sensitive)
2. Upload all your .cpp and .h to hw1 folder, don’t upload test cases. If you use ArgumentManager.h you need to turn it in too.

ps. This document may have typos, if you think something illogical, please email TAs for confirmation.