ITU Computer Engineering Department BLG 223E Data Structures, Fall 23/24 Homework 2

October 27, 2023

1 Introduction

In this assignment, you will read, write, delete and update a csv file in four different ways. First, you will store the data into an array of objects of Employee class and perform your operations on the array. Secondly, you will store the data into a linked list of objects of Employee class and perform your operations on the linked list. Then for the last two part you will incorporate STL functions and implement your solution with the vector and the list data structure.

Note that the first part of the homework as well as the dataset you will be working is the same as your previous homework. You are free to use your code that you have implemented for the first homework.

2 Dataset

You will work on two files. The first file contains the employee ID, salary and department numbers of a company's employees (You can work on all data in integer type). The data is in csv (Comma-separated values) format. A comma-separated values (CSV) file is a delimited text file that uses a comma to separate values. Each line of the file is a data record. Each record consists of three fields, separated by commas in this homework. The use of the comma as a field separator is the source of the name for this file format. A CSV file typically stores tabular data (numbers and text) in plain text, in which case each line will have the same number of fields. The CSV file format is not fully standardized. Separating fields with commas is the foundation, but commas in the data or embedded line breaks have to be handled specially. The semicolon(;) is used as a separator in the data given in the assignment. Your separator may be different depending on your operating system or the spreadsheet you are using. The second file, is a txt file, contains the operations that will be applied into array and file. There are three operations and their commands in the file is as follows:

- ADD; salary; department

- UPDATE; id; salary; department
- DELETE; id

3 Problem

You will perform the following operations on the file containing employee information:

- You will add a new employee
- You will update an employee's information
- You will delete an employee

You will do these operations in two different ways. First, you will perform the operations directly through the file. Secondly, you will save the data in the file to the array of objects of Employee class and work on it. You can examine the fstream library for file operations. How to add, update and delete operations for both solutions is explained in detail below. Your program must take csv and txt files as command line arguments. In Visual Studio Code you can add these files as arguments from the launch.json file. You have to change configuration of args value to "args": ["hw1.csv", "operations.txt"]. The command line arguments are passed to main function. And your main function should be look like int main(int argc, char** argv).

When you read line by line from the files, you will read a string. So you can use string and sstream libraries to convert integer type.

3.1 Array of objects of Employee class Solution

Read the data from the file and save it in the array of objects of Employee class data structure. Array size should be equal to the number of employees. The class can be defined as follows:

```
class Employee{
private:
    int id;
    int salary;
    int department;
public:
    Employee();
    ~Employee();
    void set_salary(int);
    void set_id();
    void set_department(int);
    int get_id();
};
```

Declare your private attributes and public methods. If you need you can add more functions.

After obtaining the array structure, close the employee information file. Do not read the file again. Your array structure should be look like the Figure 1.

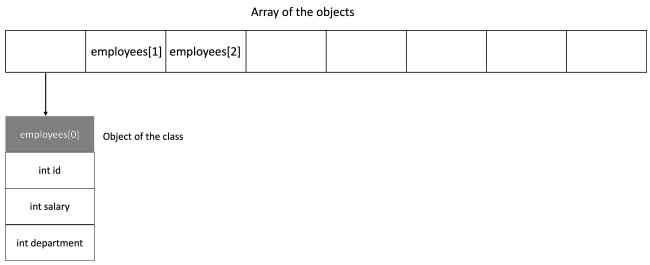

Figure 1: Array Structure

3.1.1 Add Employee Operation

For the "ADD" command in the operations text file, you will do the followings. And the command is look like "ADD;salary;department" template. Salary and department values must be integer. Increase the size of the array by one. The employee ID must be one more than the largest ID (ID numbers that will remain empty when the employee is deleted from the data set will not be used). For example, if the largest employee ID value is 100, the new employee ID should be 101. Salary and department information will also be the data you read from the operations file. Please note that the initial size given in the array data structure cannot be changed. So, you have to allocate a new array. To avoid memory leaks, deallocate the array structures that you have dynamically allocated from memory and that you will no longer use. Make sure that the added employee is added as the last element of the array.

3.1.2 Update Employee Operation

For the update operation the command in the operations text file is look like "UPDATE;id;salary;department" template. Id, salary and department values must be integer. Firstly, control the ID is valid or not. If the ID is valid, update the salary and department information. Since this is an update process, the size of the array will not be changed. If the ID is invalid , skip the operation and print out an error message like "ERROR: An invalid ID to update". Since the attributes are private, you have to use public set functions to update attributes.

3.1.3 Delete Employee Operation

For the delete operation the command in the operations text file is look like "DELETE;id" template. ID must be an integer. Please check two things while

doing deleting operation. First, the employee ID must be valid. If the ID is invalid , print out an error message like "ERROR: An invalid ID to delete". Secondly, the array size should be checked. If there is no employee hold in the array, an error message should be written and this operation should not be allowed: "ERROR: There is no Employee".

If you find that the ID is valid as a result of these checks, define a new array by decreasing the array size by one. Move the information of the employees in the old array to the new array, except for the deleted employee.

3.1.4 End of Operations File

When you reach the end of the operations file, you have to create a new csv file. And the last obtained array of objects of Employee class should be saved as "array_solution.csv" file. There should be three columns like the given dataset: employee id, salary, department. The number of rows must be the same as the number of rows of the last obtained array. Don't forget to deallocate all the memory you took dynamically.

3.2 Linked List of objects of Employee class Solution

For this solution implement a linked list data structure first. Then, read the data from the file and save it in the linked list. As an output of this part (and for the next parts of the homework) you will be creating an extra file "linked_list_solution.csv" just like the Array solution (array_solution.csv), Since you will make changes during the process of developing your code, I recommend that you work on a copy of the provided file for each part.

The class for nodes will be similar to array solution with a slight difference of next pointer of the same type.

Declare your private attributes and public methods. If you need you can add more functions. After obtaining the linked list structure, close the employee information file. Do not read the file again. Your linked list structure should be look like the Figure 2 .

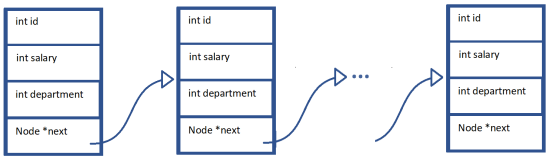

Figure 2: Linked List Structure

3.2.1 Add Employee Operation

For the "ADD" command in the operations text file, you will do the followings. And the command is look like "ADD;salary;department" template. Salary and department values must be integer. The employee ID must be one more than the largest ID (ID numbers that will remain empty when the employee is deleted from the data set will not be used). For example, if the largest employee ID value is 100, the new employee ID should be 101. Salary and department information will also be the data you read from the operations file. Make sure that the added employee is added as the last element of the linked list.

3.2.2 Update Employee Operation

For the update operation the command in the operations text file is look like "UPDATE;id;salary;department" template. Id, salary and department values must be integer. Firstly, control the ID is valid or not. If the ID is valid, update the salary and department information. Since this is an update process, the size of the linked list will not be changed. If the ID is invalid , skip the operation and print out an error message like "ERROR: An invalid ID to update". Since the attributes are private, you have to use public set functions to update attributes.

3.2.3 Delete Operation

For the delete operation the command in the operations text file is look like "DELETE;id" template. ID must be an integer. Please check two things while doing deleting operation. First, the employee ID must be valid. If the ID is invalid , print out an error message like "ERROR: An invalid ID to delete". Secondly, the linked list size should be checked. If there is no employee hold in the linked list, an error message should be written and this operation should not be allowed: "ERROR: There is no Employee". If you find that the ID is valid as a result of these checks

3.2.4 End of Operations

When you reach the end of the operations, you have to create a new csv file. And the last obtained linked list of objects of Employee class should be saved as "linked_list_solution.csv" file. There should be three columns like the given dataset: employee id, salary, department. The number of rows must be the same as the number of rows of the last obtained array. Don't forget to deallocate all the memory you took dynamically.

3.3 STL Vector of objects of Employee class Solution

Read the data from the file and save it in the vector of objects of Employee class data structure. The class object will be the same with the array solution. Declare your private attributes and public methods. If you need you can add more

functions. After obtaining the vector structure, close the employee information file. Do not read the file again.

3.3.1 Add Employee Operation

For the "ADD" command in the operations text file, you will do the followings. And the command is look like "ADD;salary;department" template. Salary and department values must be integer. The employee ID must be one more than the largest ID (ID numbers that will remain empty when the employee is deleted from the data set will not be used). For example, if the largest employee ID value is 100, the new employee ID should be 101. Salary and department information will also be the data you read from the operations file. Make sure that the added employee is added as the last element of the vector.

3.3.2 Update Employee Operation

For the update operation the command in the operations text file is look like "UPDATE;id;salary;department" template. Id, salary and department values must be integer. Firstly, control the ID is valid or not. If the ID is valid, update the salary and department information. Since this is an update process, the size of the vector will not be changed. If the ID is invalid , skip the operation and print out an error message like "ERROR: An invalid ID to update". Since the attributes are private, you have to use public set functions to update attributes.

3.3.3 Delete Employee Operation

For the delete operation the command in the operations text file is look like "DELETE;id" template. ID must be an integer. Please check two things while doing deleting operation. First, the employee ID must be valid. If the ID is invalid , print out an error message like "ERROR: An invalid ID to delete". Secondly, the linked list size should be checked. If there is no employee hold in the vector, an error message should be written and this operation should not be allowed: "ERROR: There is no Employee". If you find that the ID is valid as a result of these checks, then delete that element from the vector.

3.3.4 End of Operations File

When you reach the end of the operations file, you have to create a new csv file. And the last obtained vector of objects of Employee class should be saved as "vector_solution.csv" file. There should be three columns like the given dataset: employee id, salary, department. The number of rows must be the same as the number of rows of the last obtained array. Don't forget to deallocate all the memory you took dynamically.

3.4 STL List of objects of Employee class Solution

Read the data from the file and save it in the list of objects of Employee class data structure. The class object will be the same with the array solution. Declare your private attributes and public methods. If you need you can add more functions. After obtaining the vector structure, close the employee information file. Do not read the file again.

3.4.1 Add Employee Operation

For the "ADD" command in the operations text file, you will do the followings. And the command is look like "ADD;salary;department" template. Salary and department values must be integer. The employee ID must be one more than the largest ID (ID numbers that will remain empty when the employee is deleted from the data set will not be used). For example, if the largest employee ID value is 100, the new employee ID should be 101. Salary and department information will also be the data you read from the operations file. Make sure that the added employee is added as the last element of the list.

3.4.2 Update Employee Operation

For the update operation the command in the operations text file is look like "UPDATE;id;salary;department" template. Id, salary and department values must be integer. Firstly, control the ID is valid or not. If the ID is valid, update the salary and department information. Since this is an update process, the size of the list will not be changed. If the ID is invalid , skip the operation and print out an error message like "ERROR: An invalid ID to update". Since the attributes are private, you have to use public set functions to update attributes.

3.4.3 Delete Employee Operation

For the delete operation the command in the operations text file is look like "DELETE;id" template. ID must be an integer. Please check two things while doing deleting operation. First, the employee ID must be valid. If the ID is invalid , print out an error message like "ERROR: An invalid ID to delete". Secondly, the linked list size should be checked. If there is no employee hold in the vector, an error message should be written and this operation should not be allowed: "ERROR: There is no Employee". If you find that the ID is valid as a result of these checks, then delete that element from the list.

3.4.4 End of Operations File

When you reach the end of the operations file, you have to create a new csv file. And the last obtained list of objects of Employee class should be saved as "list_solution.csv" file. There should be three columns like the given dataset: employee id, salary, department. The number of rows must be the same as the

number of rows of the last obtained array. Don't forget to deallocate all the memory you took dynamically.

4 Execution Time Measuring

Print and compare the execution times to the terminal to compare these two solutions. You can use the following code and time.h library:

```
#include <time.h>   // library
clock_t start = clock();   // start to measure
//.... code for measuring
clock_t end = clock();   // finish measure

//measurement time in milliseconds
(double)(end - start) * 1000 / CLOCKS_PER_SEC ;
```

You can examine execution times for addition, deletion and update operations with printing to terminal. For example "ADD: array solution 100 milliseconds". After performing these operations on both solutions, find out which solution is faster for each add, update and delete operations. This will be done just to see comparison of the time it takes to perform different operations on different solutions. Therefore, comment (do not delete) these parts later.

5 Submission Rules

- You cannot use Standard Template Library (STL) other than the parts it specificaly stated so.
- Make sure you write your name and number in all of the files of your project, in the following format:
  $/*$ @*Author*
  $StudentName :< studentname >$
  $StudentID :< studentid >$
  $Date :< date > */$
- You will submit 4 cpp files, one for the array solution, one for the linked list solution, one for the vector solution and one for the list solution. (Note that the reason we ask for a single cpp file rather than what we have discussed in the recitation lecture is for the sake of simplicity during evaluation.)
- Use comments wherever necessary in your code to explain what you did.
- Your program will be checked by using Calico (https://github.com/uyar/calico) automatic checker.

- DO NOT SHARE ANY CODE or text that can be submitted as a part of an assignment.
- Only electronic submissions through Ninova will be accepted no later than deadline.
- You may discuss the problems at an abstract level with your classmates, but you should not share or copy code from your classmates or from the Internet. You should submit your **own, individual homework.**
- Academic dishonesty, including cheating, plagiarism, and direct copying, is unacceptable.
- If you have any question about the homework, you can send e-mail to Serra Uysal (uysals17@itu.edu.tr).
- Note that **YOUR CODES WILL BE CHECKED WITH THE PLAGIARISM TOOLS!**