CS135 Fall 2024 - Assignment 6

Entering Grades in Canvas...

Benjamin Cisneros

1 Description

The goal of this assignment is to familiarize you with file I/O operations, loops, and selection statements in C++. This assignment simulates a basic grade management system for students like Canvas, allowing users to input, process, and retrieve student grades from a file. By extending the grade management system, you will also learn about sorting and searching algorithms, which are crucial for data organization and retrieval in programming. Finally, understanding these concepts is vital for real-world applications in software development, data handling, and systems design.

Skills Practiced:

- File I/O: Reading from and writing to files.
- Control Structures: Utilizing loops and selection statements effectively.
- Data Management: Storing, processing, and retrieving structured data.
- Problem-Solving: Developing algorithms to manage and manipulate data.
- Sorting Algorithms: Implementing sorting to organize student grades.
- Searching Algorithms: Implementing search functionality to find specific students' grades.

Knowledge Gained:

- Implement file operations to save and retrieve data.
- Use loops to iterate over data collections.
- Apply conditional statements to make decisions based on user input.
- Structure a C++ application with user interaction.

- Sort data using algorithms such as bubble sort.
- Implement search functions to find specific student records.
- Handle error conditions, such as searching for a student that do not exist.

Long-Term Relevance:

- The skills developed in this assignment are foundational for many programming tasks, including database management, user interface development, and system automation. Proficiency in file handling and control structures will benefit students in their future coursework and professional programming endeavors.

2 Collaboration

Students are encouraged to collaborate within the guidelines specified in the syllabus. However, it is important to emphasize that the primary objective of these assignments is to acquire the essential skills required for success in the field of computer science. Accordingly, students are expected to independently write their own code, ensuring that it is free from any form of plagiarism, whether from peers, past students, or current ones. **To maintain academic integrity, all assignments will undergo cross-referencing across all sections of CS135 using the Measure Of Software Similarity (Moss) tool, which automatically assesses program similarities. You may study together and discuss the assignments, but what you turn in must be your individual work. Assignment submissions will be checked for plagiarism using Moss. Copying another student's program or sharing your program is a violation of academic integrity. Moss is not fooled by renaming variables, reformatting source code, or re-ordering functions.**

3 Task

Write a C++ program that manages student grades. The program should:

1. Load data from a file at startup.
   - Open and read random names and grades from a file named `grades.txt`.
2. Allow the user to input student names and grades.
3. Provide options to:
   (a) Allow the user to add student's grades.
   (b) Display all students and grades.
   (c) Calcualte and display the average grade.
   (d) Find and display the highest and lowest grades.
   (e) Sort the students by name or grade.
   (f) Allow the user to update a student's grade.
4. Handle cases where the user searches for a non-existent student.
5. Test and Debug:
   - Test your program with different sets of input values to ensure it works correctly.

Actions to Take if Uncertain:

- Review the sorting algorithm and searching techniques covered in class.
- Consult the instructor or TA or guidance on specific implementations.

What to Avoid:

- Ignoring Edge Cases: Do not forget to handle user input errors (e.g., invalid names, grades, etc.).
- Hardcoding Values: Allow the user to input all necessary values rather than hardcoding them in the program. Ensure the program can handle any number of students.
- Neglecting User Experience: Provide clear instructions and format the output neatly for better readability.
- Assumptions: Avoid making assumptions about the input data format.
- Online Resources: Refrain from using online resources unless instructed otherwise by the instructor.

Criteria for Success

- Correct Operations: All specified features (sorting, searching, updating) must be implemented correctly.
- Code Organization: The code should be well-organized and commented to explain its functionality.
- User-Friendly Output: The results should be presented in a clear and formatted manner.
- Compilation: Compile the program using the `g++` compiler to ensure it runs without errors. Make sure that your output matches the expected resul when executed.
- Submission: Save the program as `main.cpp` and submit it to CodeGrade before the deadline. Use the feedback provided by CodeGrade to identify and correct any errors, then resubmit if necessary.

Checklist/Rubric

By following these guidelines and criteria, you will develop a solid understanding of basic programming concepts and their application in solving real-world problems.

- ☐ Input/Output Operations:
  - ☐ Program prompts user for all required inputs.
  - ☐ Program outputs the result correctly.
- ☐ Correct Implementation:
  - ☐ All specified features (sorting, searching, updating) are implemented correctly.
- ☐ Code Quality:
  - ☐ Code is well-structured and readable.
  - ☐ Comments are used to explain key sections of the code.
- ☐ Output Formatting:
  - ☐ Output is clear and easy to understand.
  - ☐ Results are formatted correctly (e.g., two decimal places for monetary values).

4 Contents of main

In the main function, students will first initialize two arrays: one for storing student names and another for their corresponding grades. They will also set up a variable to track the number of students currently loaded from the input file. The function begins by reading data from `grades.txt`, populating these arrays, and keeping count of the entries.

Once the data is loaded, students will implement a loop that displays a menu of options for the user to interact with the system. The menu will include choices for entering new student data, displaying existing grades, calculating statistics like average, highest, and lowest grades, sorting the grades, searching for a specific student, updating a student's grade, and exiting the program. Each menu option will lead to different blocks of code that handle the specified functionality.

As users navigate through the menu, students will need to implement the corresponding logic for each choice. For example, when entering new data, they will prompt the user for a student's name and grade, update the arrays, and save the changes back to the file. When displaying grades, they will loop through the arrays to print each student's name alongside their grade. For calculating statistics, students will compute the average and determine the highest and lowest grades from the arrays. Sorting and searching functionalities will also require them to manipulate the arrays appropriately, ensuring that user inputs are handled correctly and that meaningful feedback is provided.

Pseudocode:

1. **Data loading:** The program begins by loading students names and grades from `grades.txt` into arrays.
2. **Menu Loop:** A loop displays a meny for user interaction.
3. **Choices:** Based on user input, various oeprations are performed:
   - **Entering Data:** Users can add new students and grades.
   - **Displaying Grades:** The program shows all students and their grades.
   - **Calculating Statistics:** It computes and displays average, highest, and lowest grades.
   - **Sorting:** The program sorts grades in ascending order.
   - **Searching:** It allows searching for a student by first or last name.
   - **Updating Grades:** Users can update a student's grade.
4. **File Operations:** After adding or updating data, the program saves the changes back to `grades2.txt`
5. **Exit:** The program exits upong user request.

5 Sample Output

- Invalid choice:

```
$> g++ as06.cpp
$> ./a.out
1. Enter student data
2. Display all grades
3. Calculate average, highest, lowest
4. Sort grades
5. Search for a student
6. Update a student's grade
7. Exit
Invalid choice. Please try again.
1. Enter student data
2. Display all grades
3. Calculate average, highest, lowest
4. Sort grades
5. Search for a student
6. Update a student's grade
7. Exit
```

```
Enter your choice: 7
Exiting...
```

- Displaying all grades:

```
$> ./a.out
1. Enter student data
2. Display all grades
3. Calculate average, highest, lowest
4. Sort grades
5. Search for a student
6. Update a student's grade
7. Exit
Enter your choice: 2
Student Grades:
-----------------------------------
Name: Bella Lee, Grade: 3
Name: Oliver Lewis, Grade: 14
Name: Ethan Johnson, Grade: 50
Name: Victor Edwards, Grade: 68
Name: Xavier Mitchell, Grade: 90
Name: Rachel Scott, Grade: 28
Name: Olivia Hall, Grade: 37
Name: Aaron Jones, Grade: 21
Name: David Anderson, Grade: 80
Name: Wendy Collins, Grade: 94
Name: Kevin Cooper, Grade: 92
Name: Victor Stewart, Grade: 37
Name: Mia Green, Grade: 9
Name: Bella Robinson, Grade: 91
Name: Victor Smith, Grade: 78
Name: Xavier Lewis, Grade: 1
Name: Ethan Rogers, Grade: 72
Name: Paul Adams, Grade: 97
Name: Wendy Evans, Grade: 74
Name: Olivia Parker, Grade: 34
...
Name: Isabella Lee, Grade: 78
Name: Vince Hall, Grade: 20
Name: Mia Scott, Grade: 52
Name: Lily Phillips, Grade: 34
Name: Zachary Reed, Grade: 47
Name: Michael Perez, Grade: 73
Name: Rose Hernandez, Grade: 83
Name: Isabella Carter, Grade: 21
-----------------------------------
1. Enter student data
2. Display all grades
3. Calculate average, highest, lowest
4. Sort grades
5. Search for a student
6. Update a student's grade
7. Exit
Enter your choice: 7
Exiting...
```

- Calculating statistics:

```
$> ./a.out
1. Enter student data
2. Display all grades
3. Calculate average, highest, lowest
4. Sort grades
5. Search for a student
6. Update a student's grade
7. Exit
Enter your choice: 3
Average Grade: 49.80
Highest Grade: 100 (Nora Robinson)
Lowest Grade: 0 (Michael Anderson)
1. Enter student data
2. Display all grades
3. Calculate average, highest, lowest
4. Sort grades
5. Search for a student
6. Update a student's grade
7. Exit
Enter your choice: 7
Exiting...
```

- Search a student by name:

```
$> ./a.out
1. Enter student data
2. Display all grades
3. Calculate average, highest, lowest
4. Sort grades
5. Search for a student
6. Update a student's grade
7. Exit
Enter your choice: 5
Enter the name of the student to search: Samuel
Found: Samuel Perez, Grade: 32
1. Enter student data
2. Display all grades
3. Calculate average, highest, lowest
4. Sort grades
5. Search for a student
6. Update a student's grade
7. Exit
Enter your choice: 7
Exiting...
```

- Search a student by lastname:

```
$> ./a.out
1. Enter student data
2. Display all grades
3. Calculate average, highest, lowest
4. Sort grades
5. Search for a student
```

```
6. Update a student's grade
7. Exit
Enter your choice: 5
Enter the name of the student to search: Perez
Found: Samuel Perez, Grade: 32
1. Enter student data
2. Display all grades
3. Calculate average, highest, lowest
4. Sort grades
5. Search for a student
6. Update a student's grade
7. Exit
Enter your choice: 7
Exiting...
```