Programming Practicum Report: Meeting #6

R. Ethan Halim

October 14th, 2024

1 Averaging Student Scores

The entire source file is hosted on a GitHub repository **here**.

1.1 Explanation

The sole requirement of this assignment is the usage of a C++ struct in order to store each student's record. Below are the definition and the fields of the struct involved in the purpose thereof. Since the student ID might or might not contain non-numeral characters, I chose to use `std::string` as its data type to accommodate both.

```
struct Student {
    std::string id;
    double midterm;
    double final;
    double average;
};
```

Initially, the program requests the number of students to be inputted into the program. This will be used for the for-loop later on to receive the midterm and the final scores of the `n` students. Additionally, the program automatically exits and returns an error code if the number of students is zero.

```
int program(std::istream& cin, std::ostream& cout) {
    size_t n_students;
    cout << "Enter the amount of students: ";
    cin >> n_students;
```

```
    // Exits early if the inputted number of students is
    ↪  zero.
    if (n_students == 0) {
        cout << "Provide at least one input!\n";
        return 1;
    }

    ...
}
```

Using the received number of students, the for-loop below iterates for every student whose midterm and final score values are to be inputted into the vector of `Student` instances. On each iteration, the program provides the fields of each `Student` instance to be filled by the user: the student ID, the midterm exam score, and the final exam score.

```
int program(std::istream& cin, std::ostream& cout) {
    ...

    std::vector<Student> students;
    cout << '\n';
    for (size_t i = 1; i <= n_students; i++) {
        Student student;
        cout << "[Data for Student #" << i << "]\n";

        // Gets the student ID (as a string)
        cout << "- ID: ";
        while (student.id.empty()) {
            std::getline(cin, student.id);
        }

        // Gets the midterm exam score
        while (true) {
            cout << "- Midterm exam score: ";
            cin >> student.midterm;

            // Loops again if the inputted score is invalid
            if (student.midterm > 100) {
                cout << "The score must not exceed 100!\n";
            }
            else {
                break;
            }
        }
```

```
        // Gets the final exam score
        while (true) {
            cout << "- Final exam score: ";
            cin >> student.final;

            // Loops again if the inputted score is invalid
            if (student.final > 100) {
                cout << "The score must not exceed 100!\n";
            }
            else {
                break;
            }
        }

        ...
    }

    ...
}
```

Notice that there exist two of the template code below, whereby `[...]` refers to either the student's midterm score or final score. The code enclosed inside of the infinite while-loop ensures that the score inputted is within the range from 0 (by virtue of the underlying unsigned integer type used to store the score) to 100, as checked by the if-statement. If the user inputted a value outside of the said bounds, the program would ask them to reinput the value.

```
while (true) {
    cout << "- [...] score: ";
    cin >> student.[...];

    // Loops again if the inputted score is invalid
    if (student.[...] > 100) {
        cout << "The score must not exceed 100!\n";
    }
    else {
        break;
    }
}
```

Finally, upon the conclusion of each iteration, the average score is computed from the given midterm and final scores, and stored in the field available in the `Student` struct. The structure instance is pushed into the vector.

```
int program(std::istream& cin, std::ostream& cout) {
    ...

    for (size_t i = 1; i <= n_students; i++) {
        ...

        // Calculates the student average and pushes to the
        ↪ vector
        student.average = (student.midterm + student.final) /
        ↪ 2;
        students.push_back(student);
        cout << '\n';           }

    ...
}
```

Once the data and the average score of each student have been gathered and calculated, they are all displayed at the end of the program. It iterates over the vector of `Student` instances, in order to announce their average scores sequentially. Additionally, the program calculates the overall average as well.

```
int program(std::istream& cin, std::ostream& cout) {
    ...

    // Final output
    cout << "[Calculated Average(s) of All " << n_students << "
    ↪ Student(s)]\n";
    double sum;
    for (Student& student : students) {
        cout << "- Student ID #" << student.id << ": " <<
        ↪ student.average << '\n';
        sum += student.average; // Sums each of the students'
        ↪ average score
    }
    cout << "The overall average is " << (sum / n_students) <<
    ↪ ".\n";

    return 0;
}
```

1.2 Manual Testing

Below is the compilation and the testing of the source code.

```
avaxar@AvaxarTUF:~/Repos/uni-practica-1/week_6/01_student_scores$ make
g++ -Wall student_scores.cpp -o student_scores
./student_scores
Enter the amount of students: 3

[Data for Student #1]
- ID: 150
- Midterm exam score: 75
- Final exam score: 83

[Data for Student #2]
- ID: 155
- Midterm exam score: 92
- Final exam score: 86

[Data for Student #3]
- ID: 160
- Midterm exam score: 56
- Final exam score: 38

[Calculated Average(s) of All 3 Student(s)]
- Student ID #150: 79
- Student ID #155: 89
- Student ID #160: 47
The overall average is 71.6667.
```

1.3 Test Cases

1.3.1 Tests

Below is copied directly from the `tests.txt` file.

```
%INPUT
1
42420
69
100
%OUTPUT
Enter the amount of students:
[Data for Student #1]
- ID: - Midterm exam score: - Final exam score:
[Calculated Average(s) of All 1 Student(s)]
- Student ID #42420: 84.5
The overall average is 84.5.
%END

%INPUT
2
10001
50
75
10002
30
90
%OUTPUT
Enter the amount of students:
[Data for Student #1]
- ID: - Midterm exam score: - Final exam score:
[Data for Student #2]
- ID: - Midterm exam score: - Final exam score:
[Calculated Average(s) of All 2 Student(s)]
- Student ID #10001: 62.5
- Student ID #10002: 60
The overall average is 61.25.
%END

%INPUT
3
29999
50
75
```

```
29998
30
90
29997
0
100
%OUTPUT
Enter the amount of students:
[Data for Student #1]
- ID: - Midterm exam score: - Final exam score:
[Data for Student #2]
- ID: - Midterm exam score: - Final exam score:
[Data for Student #3]
- ID: - Midterm exam score: - Final exam score:
[Calculated Average(s) of All 3 Student(s)]
- Student ID #29999: 62.5
- Student ID #29998: 60
- Student ID #29997: 50
The overall average is 57.5.
%END

%INPUT
4
999999
50
75
999998
30
90
999997
0
100
999996
100
100
%OUTPUT
Enter the amount of students:
[Data for Student #1]
- ID: - Midterm exam score: - Final exam score:
[Data for Student #2]
- ID: - Midterm exam score: - Final exam score:
[Data for Student #3]
- ID: - Midterm exam score: - Final exam score:
[Data for Student #4]
```

```
- ID: - Midterm exam score: - Final exam score:
[Calculated Average(s) of All 4 Student(s)]
- Student ID #999999: 62.5
- Student ID #999998: 60
- Student ID #999997: 50
- Student ID #999996: 100
The overall average is 68.125.
%END

%INPUT
5
5999999
50
75
5999998
30
90
5999997
0
100
5999996
100
100
5999995
0
0
%OUTPUT
Enter the amount of students:
[Data for Student #1]
- ID: - Midterm exam score: - Final exam score:
[Data for Student #2]
- ID: - Midterm exam score: - Final exam score:
[Data for Student #3]
- ID: - Midterm exam score: - Final exam score:
[Data for Student #4]
- ID: - Midterm exam score: - Final exam score:
[Data for Student #5]
- ID: - Midterm exam score: - Final exam score:
[Calculated Average(s) of All 5 Student(s)]
- Student ID #5999999: 62.5
- Student ID #5999998: 60
- Student ID #5999997: 50
- Student ID #5999996: 100
- Student ID #5999995: 0
```

```
The overall average is 54.5.
%END
```

1.3.2 Execution

Below are the results of the test cases. No test cases failed.