Creating Grading and Feedback System of a Console Based Time Scheduling of College Students System Utilizing Data Structures and Algorithm

**Subtitle focusing on your topic**

Randy, Sarte, Camposo[[1]](#footnote-1)\* (Student)

Bachelor of Science in Computer Science 2, Bohol Islands State University-Bilar Campus,randy.camposo@bisu.edu.ph

MAX ANGELO DAPITILLA PERIN (INSTRUCTOR)

Faculty of Department of Computer Science, College of Technology and Allied Sciences, Bohol Island State University-Bilar Campus, maxangelo.perin@bisu.edu.ph

Although there is no distinctive header, this is the abstract. This submission template allows authors to submit their papers for review to an ACM Conference or Journal without any output design specifications incorporated at this point in the process. The ACM manuscript template is a single column document that allows authors to type their content into the pre-existing set of paragraph formatting styles applied to the sample placeholder text here. Throughout the document you will find further instructions on how to format your text. If your conference’s review process will be double-blind: The submitted document should not include author information and should not include acknowledgments, citations or discussion of related work that would make the authorship apparent. Submissions containing author identifying information may be subject to rejection without review. Upon acceptance, the author and affiliation information must be added to your paper.

CCS CONCEPTS • Theory of computation • Design and analysis of algorithms• Data structures design and analysis

Additional Keywords and Phrases: Time Scheduling and Management, Data Structures, Algorithmic Optimization, Grading and Feedback.

ACM Reference Format:

**Randy Sarte Camposo**, MAX ANGELO DAPITILLA PERIN. 2023. **Creating Grading and feedback System of a Console Based Time Scheduling of College Students System Utilizing Data Structures and** **Algorithm**. In Research Project Presentation for Bachelor of Science in Computer Science 3 in CS 212 – Data Structures and Algorithm S.Y. 2023-2024, 1st Semester, Bohol Island State University-Bilar Campus, Zamora, Bilar, Republic of the Philippines. ACM, New York, NY, USA

1. INTRODUCTION

Creating a grading and feedback System for a console-based time scheduling system for college students is a vital and challenging task, and it plays a pivotal role in enhancing the overall educational experience. In today's fast-paced world, students need effective tools to manage their time efficiently and make the most of their college years. This project aims to address this need by utilizing data structures and algorithms to design a robust and usefriendly scheduling system, accompanied by a grading and feedback component.

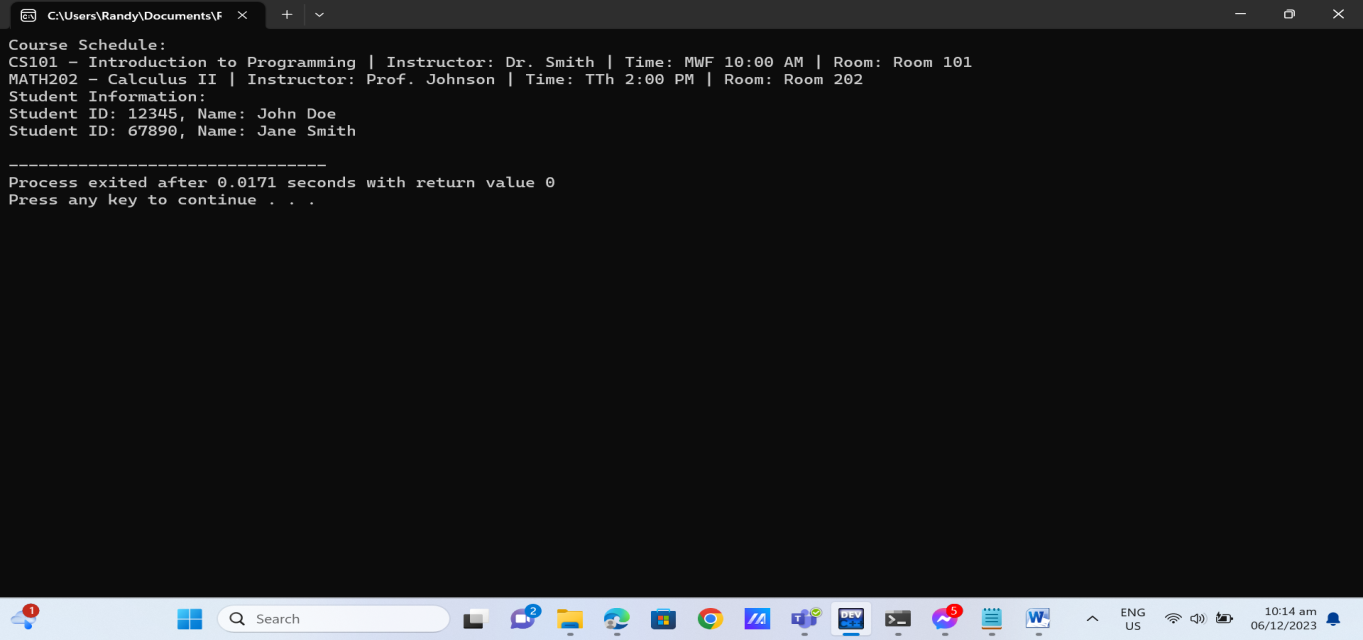


Figure1. Showing a different, choices of Grading Feedback System.

. A classification is proposed for academic timetabling problems, requirements for the timetables, mathematical models, solution methods, data representation, and interface design [[1](#bib1)]. The variety of data formats currently in use, and the diversity of existing timetabling problems, makes the comparison of research results and exchange of data concerning real problems extremely difficult [[2](#bib2)].

1. PROPOSED METHODOLOGY

Provide an overview of the importance of a grading and feedback system in educational settings. Develop a comprehensive methodology for implementing an effective grading and feedback system. And the first thing that I do is to Identify stakeholder needs and system requirements for grading and feedback processes and we can see the different choices, and each choices haves a capability to give a feedback of your choices. Hence it helps the students to View their Schedule and see if they are being enrolled of that subject.

* 1. Computer Code

Showing all of the different choices of the program

#include <iostream>

#include <fstream>

#include <vector>

#include <map>

class Course {

private:

std::string courseCode;

std::string courseName;

std::string instructor;

std::string timeSlot;

std::string room;

public:

Course(std::string code, std::string name, std::string instructor, std::string timeSlot, std::string room)

: courseCode(code), courseName(name), instructor(instructor), timeSlot(timeSlot), room(room) {}

friend std::ostream& operator<<(std::ostream& os, const Course& course) {

os << course.courseCode << " - " << course.courseName << " | Instructor: " << course.instructor

<< " | Time: " << course.timeSlot << " | Room: " << course.room;

return os;

}

};

class Student {

private:

std::string studentID;

std::string name;

std::map<std::string, int> grades;

public:

Student(std::string id, std::string name) : studentID(id), name(name) {}

friend std::ostream& operator<<(std::ostream& os, const Student& student) {

os << "Student ID: " << student.studentID << ", Name: " << student.name;

return os;

}

};

struct Schedule {

Course course;

Student student;

};

class SchedulingSystem {

private:

std::vector<Schedule> schedules;

public:

void addSchedule(const Schedule& schedule) {

schedules.push\_back(schedule);

}

void displaySchedules() const {

std::cout << "Schedules:" << std::endl;

for (const Schedule& schedule : schedules) {

std::cout << "Course: " << schedule.course << " | " << "Student: " << schedule.student << std::endl;

}

}

void saveToFile(const std::string& filename) const {

std::ofstream outFile(filename.c\_str());

std::streambuf\* coutBuffer = std::cout.rdbuf();

std::cout.rdbuf(outFile.rdbuf());

displaySchedules();

std::cout.rdbuf(coutBuffer);

}

};

int main() {

SchedulingSystem schedulingSystem;

// Add sample schedules

schedulingSystem.addSchedule({

{"CS101", "Introduction to Programming", "Dr. Smith", "MWF 10:00 AM", "Room 101"},

{"12345", "John Doe"}

});

schedulingSystem.addSchedule({

{"MATH202", "Calculus II", "Prof. Johnson", "TTh 2:00 PM", "Room 202"},

{"67890", "Jane Smith"}

});

// Display schedules

schedulingSystem.displaySchedules();

// Save the console output to a text file

schedulingSystem.saveToFile("output.txt");

return 0;

}

* 1. Data Structures Visualization

The critical component is the Scheduling System class, where schedules are managed as instances of the Schedule struct, combining course and student instances. The system allows dynamic scheduling, addition of courses and students, and effective display of schedules. Furthermore, the implementation facilitates data persistence by redirecting console output to a text file using the save To File method. This visualization offers a clear understanding of the underlying data structures and their interactions, demonstrating a practical approach to handling college scheduling and grading processes in a console-based environment, Shown in [Figure 2.](#fig2)

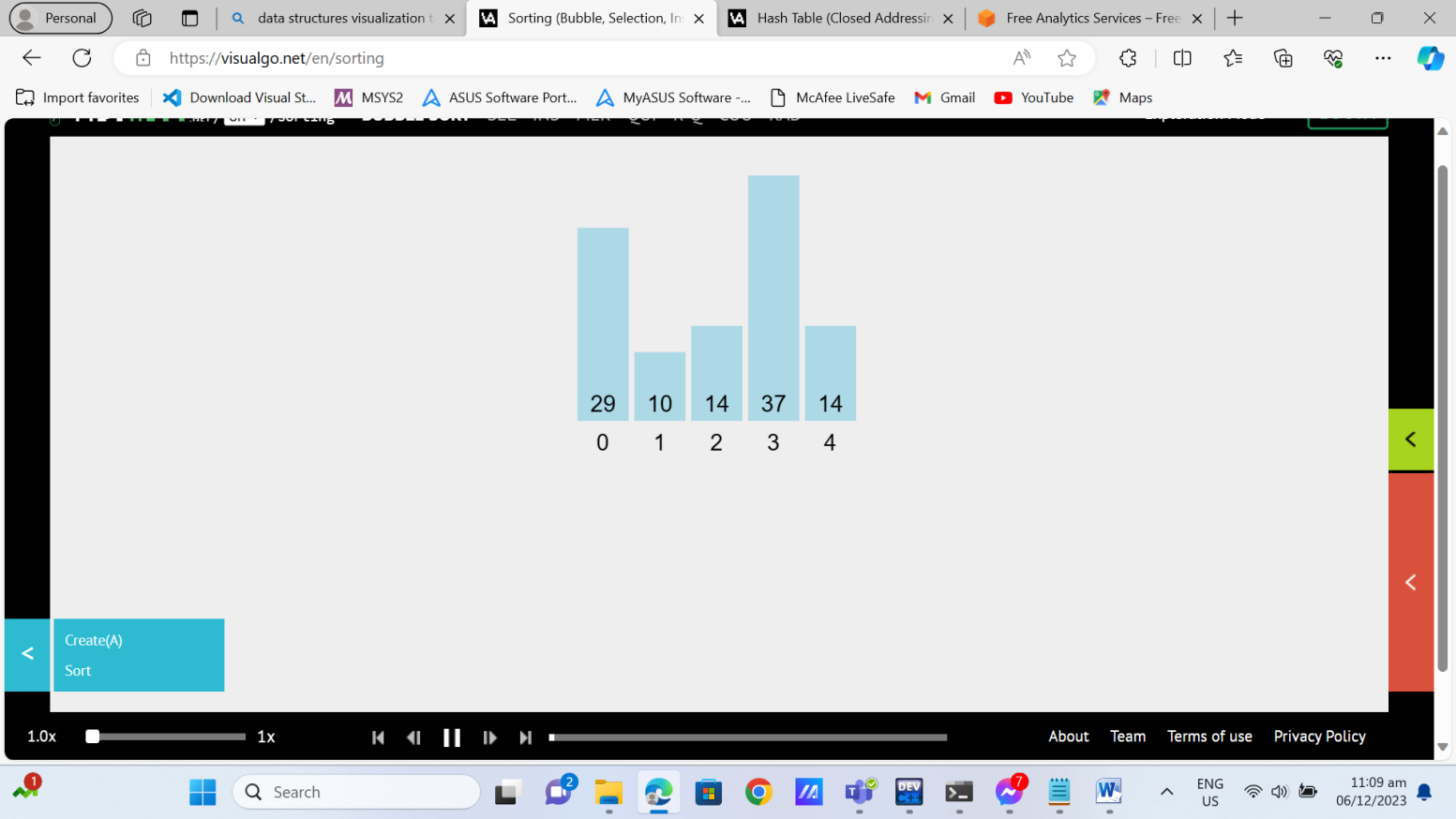


Figure 2. Data Structures Visualization.

* 1. Pseudocode

Step 1: Import necessary libraries for input/output, file operations, vectors, and maps.

Step 2: Declare the Course class to represent information about a course.

Step 3: Declare the Student class to represent information about a student.

Step 4: Create a struct named Schedule that includes instances of the Course and Student classes.

Step 5: Declare the SchedulingSystem class that contains a vector of Schedule instances.

Step 6: Implement a method in SchedulingSystem to add schedules to the vector.

Step 7: Implement a method to display the schedules stored in the system.

Step 8: Implement a method to redirect the console output to a file, allowing saving the displayed Step Step 9: schedules to a text file.

Step 10: Start the main function.

Step 11: Instantiate an object of the SchedulingSystem class.

Step 12: Add two sample schedules to the system.

Step 13: Call the displaySchedules method to show the added schedules on the console.

Step 14: Call the saveToFile method to save the displayed schedules to a text file named "output.txt".

Step 15: End the main function with a return statement.

Step 16: Initialize the information for the first schedule, including course details and student details.

Step 17: Add the first schedule to the SchedulingSystem instance.

Step 18:Initialize the information for the second schedule, including course details and student details.

Step 19: Add the second schedule to the SchedulingSystem instance.

Step 20: Call the displaySchedules method to print the schedules to the console.

Step 21:Call the saveToFile method to redirect the console output to "output.txt" and save the schedules.

3 Result and Dicussion

The provided C++ code defines a simple scheduling system for college courses and students. It uses three classes: Course, Student, and SchedulingSystem. The Schedule struct contains instances of the Course and Student classes, representing the pairing of a course with a student. In the SchedulingSystem class, schedules are stored in a vector.

In the main function, sample schedules are added to the SchedulingSystem. The displaySchedules method prints the schedules to the console, and the saveToFile method redirects the console output to a file named "output.txt", effectively saving the displayed schedules to that file.This code provides a foundation for a console-based scheduling system, allowing the addition, display, and storage of course-student pairs. It can be extended for more complex functionalities, such as user input for dynamic scheduling or additional data fields for courses and students. The simplicity of the code makes it a good starting point for further development based on specific requirements.

1. CITING RELATED WORK

[[1](#bib1)]Bardadym, V. A. (1995, August). Computer-aided school and university timetabling: The new wave. In international conference on the practice and theory of automated timetabling (pp. 22-45). Berlin, Heidelberg: Springer Berlin Heidelberg. [[2](#bib2)]

Reis, L. P., & Oliveira, E. (2001). A language for specifying complete timetabling problems. In Practice and Theory of Automated Timetabling III: Third International Conference, PATAT 2000 Konstanz, Germany, August 16–18, 2000 Selected Papers 3 (pp. 322-341). Springer Berlin Heidelberg.

REFERENCES

<bib id="bib1"><number>[1]</number>Anisimov B.V. and Vlasov V.P. (1971) “Analysis of Decision Rules Applied to Lessons Timetabling”, *Mathematical and Information Problems of Science Prognostication and Management*, Kiev, V. Glushkov Inst. for Cybernetics, 113–122 (Russian).*</bib>*

<bib id="bib2"><number>[2]</number>Y.H. Cho et al.A personalized recommender system based on web usage mining and decision tree inductionExpert Systems with Applications(2002)</bib>

APPENDICES

Dev C++

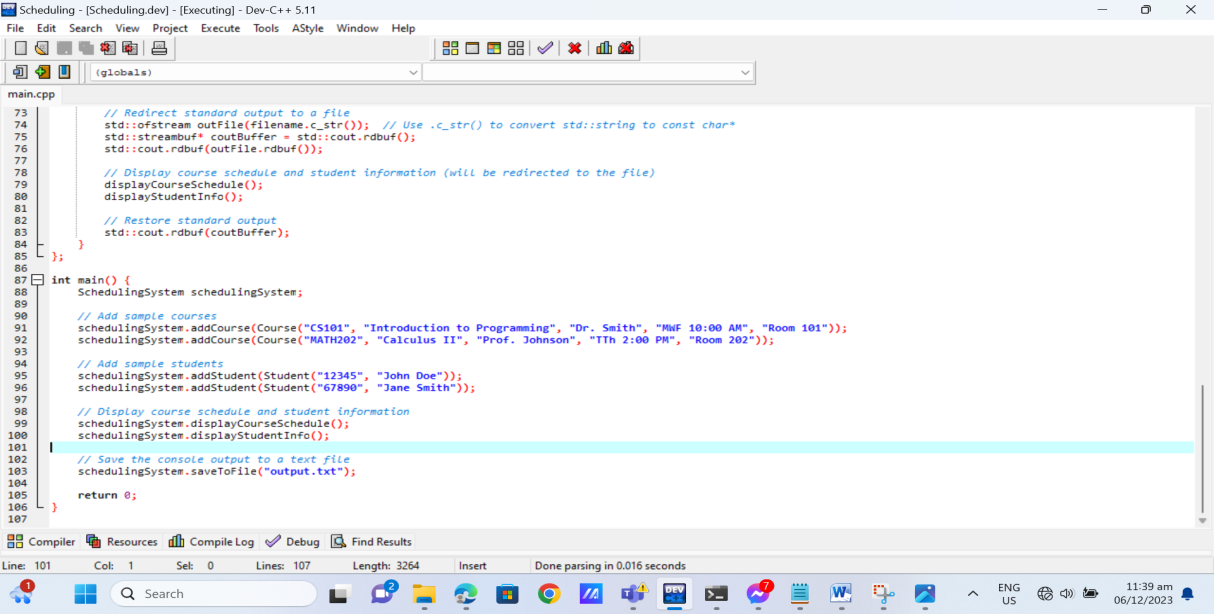


Figure 3. Appendices

Github Contributions

[Commits · MAKASA-LABORATORY/223-DOCBTSOCSUDSAA-AOTSOCSSMTDS (github.com)](https://github.com/MAKASA-LABORATORY/223-DOCBTSOCSUDSAA-AOTSOCSSMTDS/commits/main)

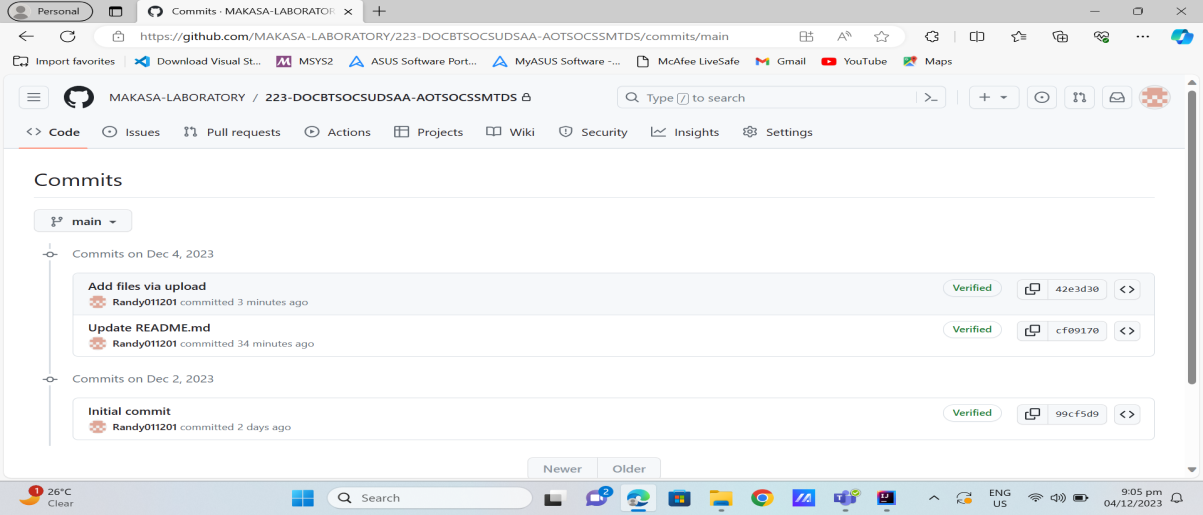


Figure 4. Github Contributions.

1. [↑](#footnote-ref-1)