**PROJECT REPORT**

CLASS ORGANIZER PROGRAM IN C++

CSE 1206 || OBJECT ORIENTED PROGRAMMING LABORATORY

Name: Sheikh Md. Galib Mahim Roll : 2107020

Section : A

**INSTRUCTED BY**

SAFIN AHMMED

MD. REPON ISLAM LECTURER

LECTURER

DEPT. CSE KUET

Submission Date: 1/20/23

DEPT. CSE KUET

Table of Content

1. Introduction
2. Object Oriented Concepts
3. Design and Implementation
4. Features and functionality
5. Conclusion
6. References

**Introduction**

Class organizer Program is made to automate certain mundane tasks. It has a teacher section and a student section, each having their own

Functions. From Schedule set, CT fix, Sending message, Routine show, Attendance counter, and many more.

This project was made to understand key aspects of Object-oriented programming and create a working program covering all the concepts of object-oriented programming.

**Object-oriented Concepts**

C++ is a powerful language with all the functionality of a higher-level programming language also carrying aspects of object-oriented programming.

1. Objects
2. Classes
3. Encapsulation
4. Abstraction
5. Inheritance
6. Polymorphism

Also, there is function overloading and function overriding.

This project covers all of the key concepts of Object-oriented programming.

**Design and Implementation**

Designing this project was a heavy task, first of all, an overview was created of what to cover and what not, and how to implement all the core concepts of OOP.

The required classes were selected, and then according to them, parent and child classes was derived. All the functionalities were checked and distributed among the class.

Mainly, The program has a parent class **“Person”** under which **“Teacher”** and **“Student”** classes were declared. This is **Inheritance**

Property of OOP. Where “Person” is the base or parent class and “Teacher” and “student” are child classes inheriting property from parent class.

Under “person” there is a pure virtual function which then makes it an **Abstract class** meaning no object of this class can be declared, and All the classes inheriting from this class must have the definition of that pure virtual function.

Under the “Teacher” class, there are multiple functions

1. Default and parameterized constructor
2. Setter functions
3. Display function
4. giveMessage() function, which gives message to all students, this function is also overloaded function.
5. scheduleSet() function, sets the schedule and it has also an overloaded instance.
6. CT\_fix(), this shows all the empty slot in the routine and helps select a suitable time for CT.

Under “Student” Class, There are multiple fucntions,

1. Default and Parameterised constructor
2. Setter functions
3. Getter functions
4. Display function
5. Operator overloading for getting input
6. Friend function, attendenceChcker(), which calculates minimum required class to fulfill 60% attendance.

Here is a detailed explanation for each portion of the code.

**THE CODE || Features and functionalities**

**person class**

The person class in the provided code serves as an abstract base class for two other classes: Teacher and Student. This class introduces the concept of inheritance and abstraction.

The person class is declared as an abstract class using the

**“virtual void displayInfo() = 0;”** syntax. This means that it cannot be instantiated on its own but serves as a blueprint for other classes.

**Routine class**

The Routine class uses a 2D array (string arr[5][9]) to represent a weekly class schedule. Each cell in the array corresponds to a particular day and time slot, with the string value in each cell representing the class or activity scheduled for that period. Empty slots are represented as "Nil."

The Routine class initializes its schedule data by reading from a file named "routine.txt" in its constructor. This demonstrates the concept of constructors and file I/O operations. Loading data from a file allows for persistence and easy modification of the schedule data.

**routineCheck():**

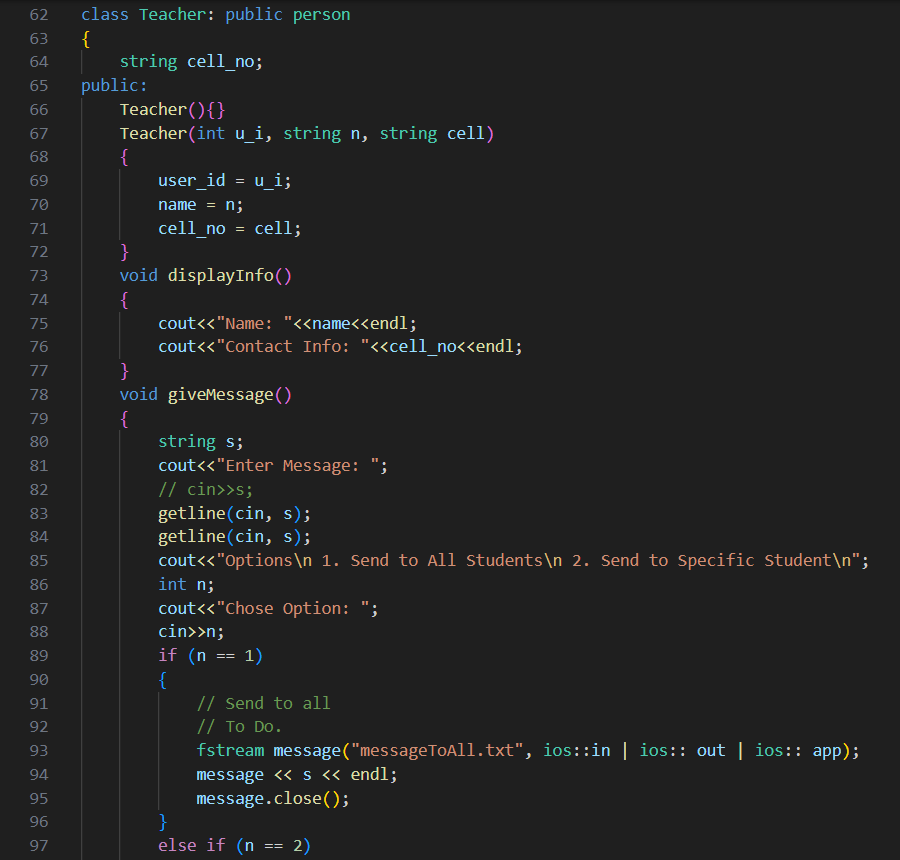
This member function displays the class schedule in a tabular format, showing days of the week and corresponding time slots. It provides a user-friendly view of the schedule, promoting encapsulation and abstraction by hiding the internal data representation.

**emptySlot():**

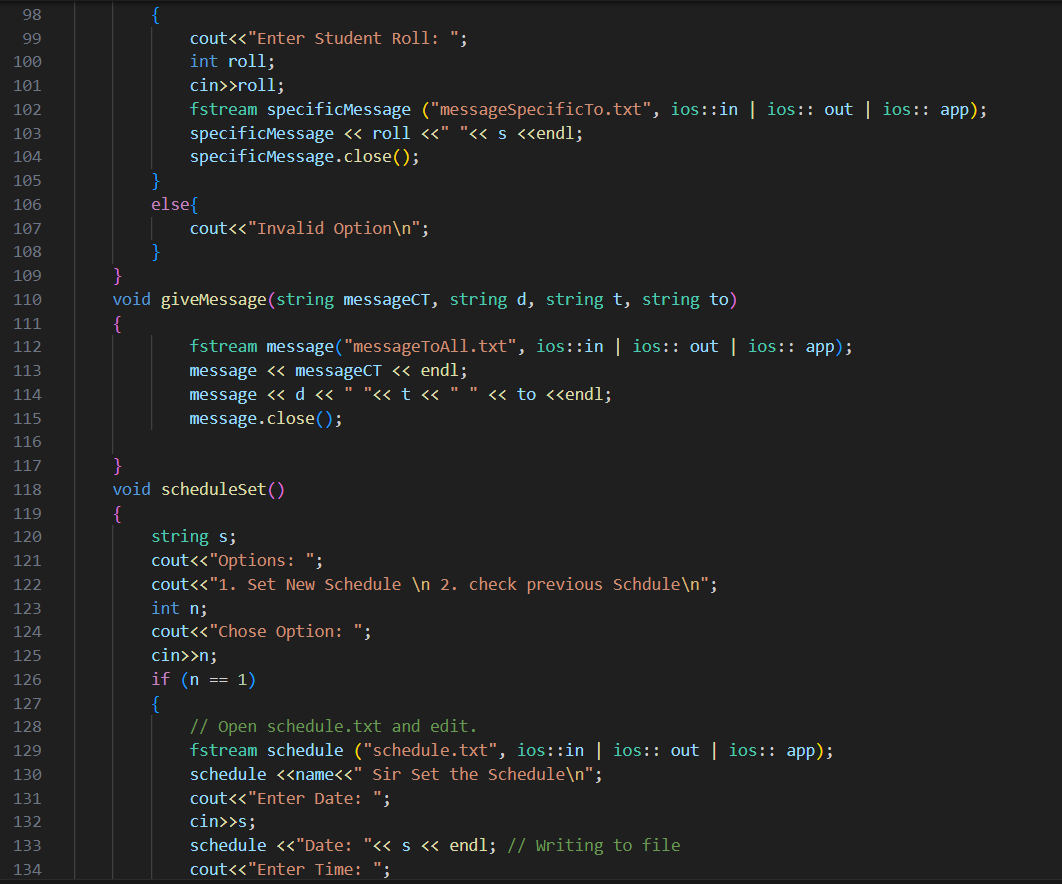
This function is declared as a friend function inside the Routine class. This allows the function to access and manipulate the private members of the Routine class. Friend functions can be useful for operations that need access to class internals but don't belong to the class itself.

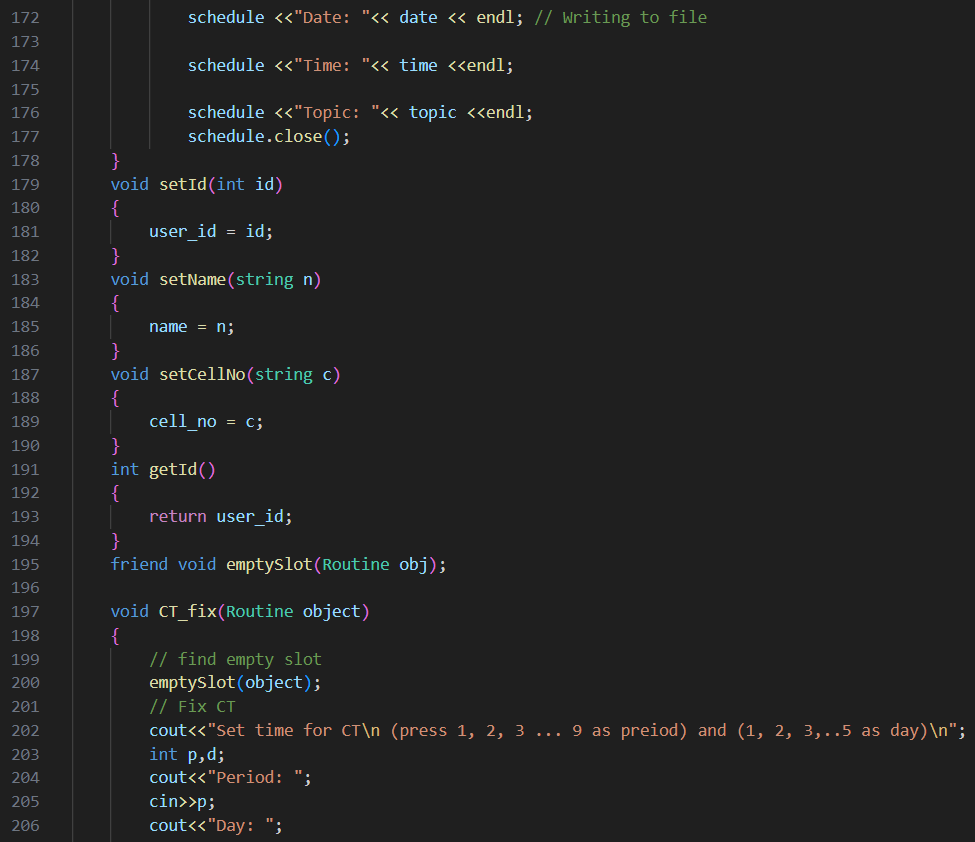
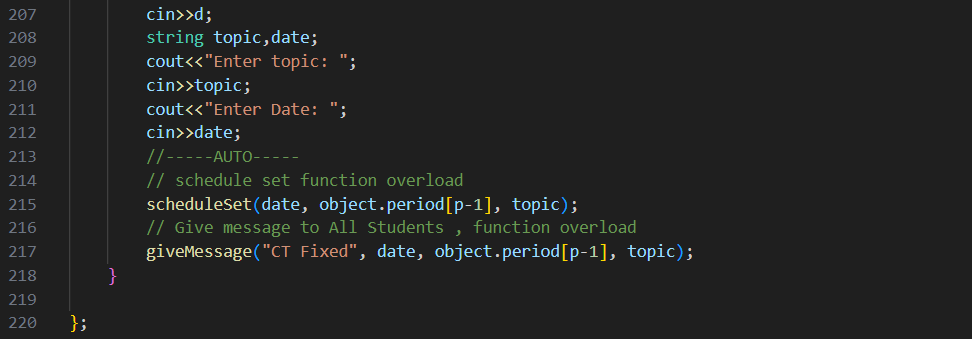
It provides a structured and organized way to manage and display class schedules, making it easier to work with schedule data in an educational context.

**Teacher Class**









**Teacher() (Default Constructor):** Initializes a teacher object with default values.

**Teacher(int u\_i, string n, string cell):** Parameterized constructor that initializes a teacher object with provided values for user ID, name, and cell number.

**void displayInfo():** This function displays the name and contact information (cell number) of the teacher.

**void giveMessage():** Allows the teacher to send a message. It takes user input for the message and provides options to send it to all students or a specific student.

**void giveMessage(string messageCT, string d, string t, string to):**

Overloaded function that allows the teacher to send a message with specific details (message content, date, time, and recipient).

**void scheduleSet():**

Allows the teacher to set a new schedule. It takes user input for the date, time, and topic of the schedule and writes this information to a file.

**void scheduleSet(string date, string time, string topic):** Overloaded function that allows the teacher to set a schedule with specific details without user input.

**void setId(int id):** Sets the user ID of the teacher.

**void setName(string n):** Sets the name of the teacher.

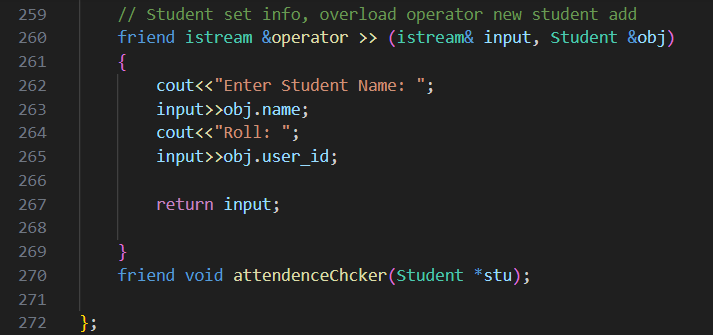
**void setCellNo(string c):** Sets the cell number of the teacher.

**void CT\_fix(Routine object):**

This function interacts with a Routine object to find an empty slot and fix a Class Test (CT) schedule. It prompts the teacher to input the period, day, topic, and date for the CT, and then it updates the schedule and sends a message to all students.

**Student class**





**Student() (Default Constructor):** Initializes a student object with default values.

**Student(int roll, string n):** Parameterized constructor that initializes a student object with provided values for roll (user ID) and name.

**void displayInfo():** This function displays the name and roll number (user ID) of the student.

**void setRoll(int roll):** Sets the roll number (user ID) of the student.

**void setName(string n):** Sets the name of the student.

**void setAttendence(int a):** Sets the attendance percentage of the student.

**int getRoll():** Returns the roll number (user ID) of the student.

**string getName():** Returns the name of the student.

**friend istream &operator >> (istream& input, Student &obj):** Overloaded input operator (>>) that allows user input to set the name and roll number (user ID) of a student object.

Attendance Checker Function:

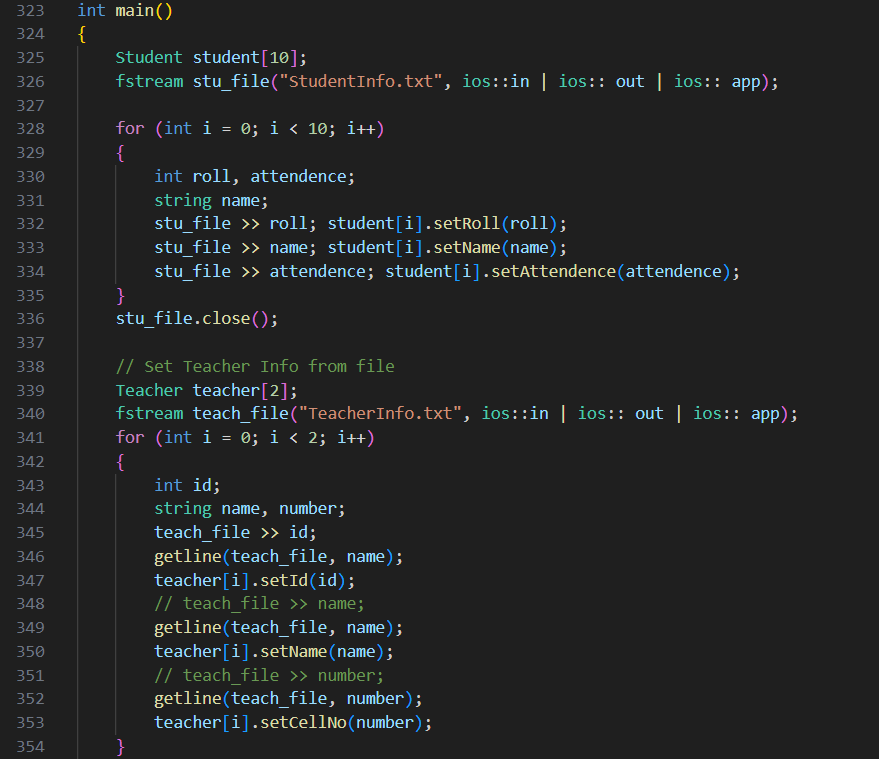
**void attendenceChcker(Student \*stu):**

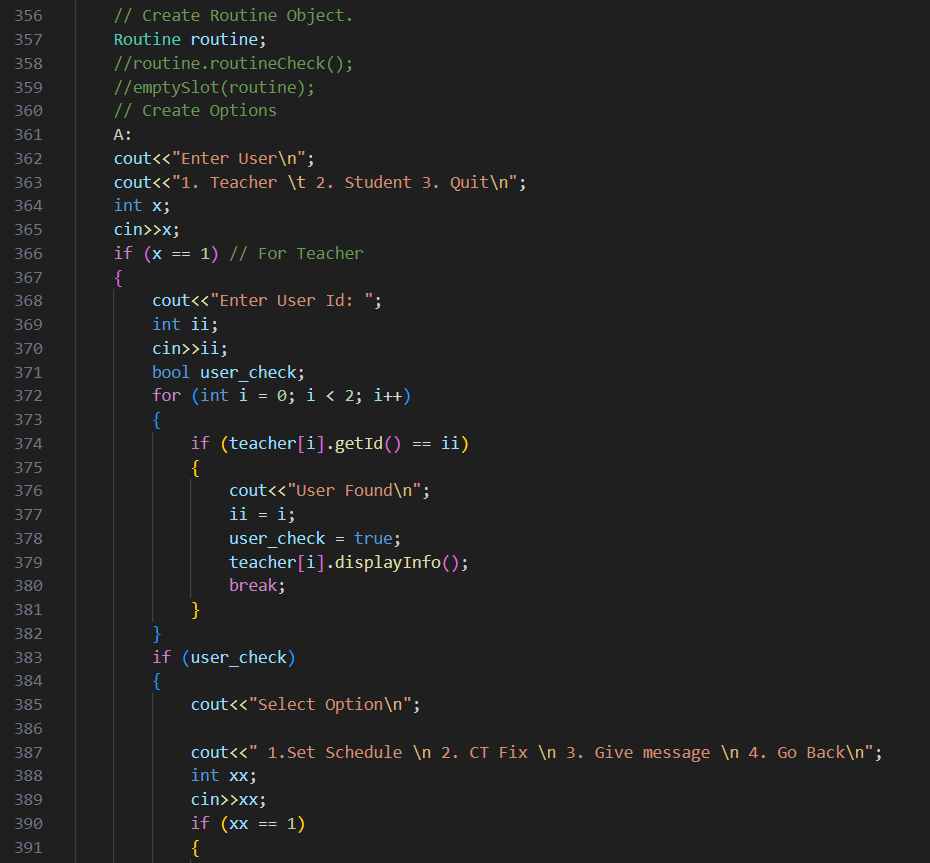
This function calculates and displays the attendance percentage for a student. It takes user input for the remaining weeks, per week class count, and the number of classes already completed. Based on this input, it calculates the attendance percentage and provides suggestions if the attendance is below 60%.

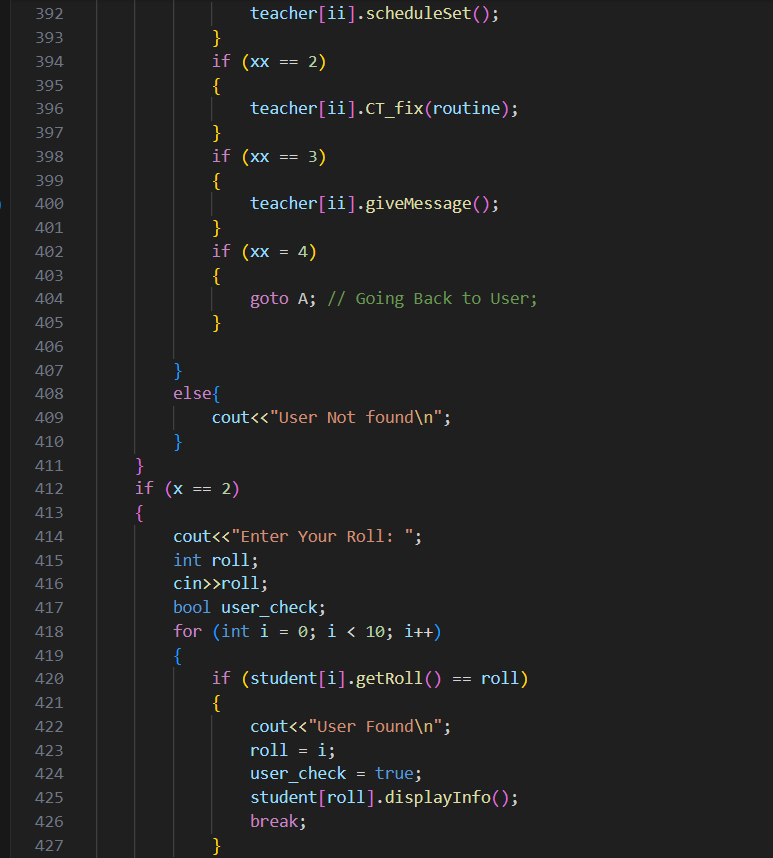
This is a **friend function** of **student** class.

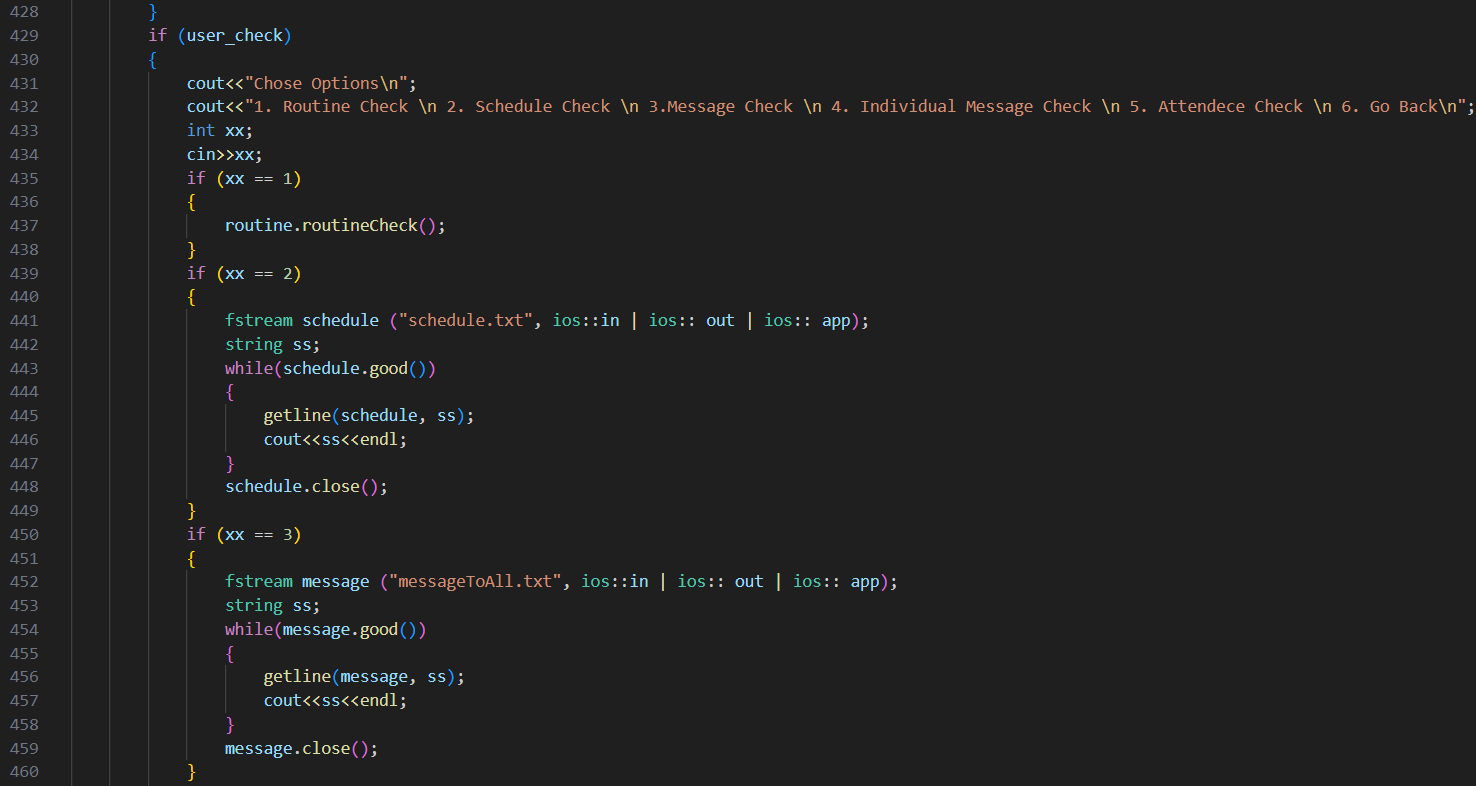
The primary purpose of the emptySlot function is to display the class schedule with an emphasis on empty slots or periods where no classes or activities are scheduled. These empty slots are indicated in the output for better visualization.

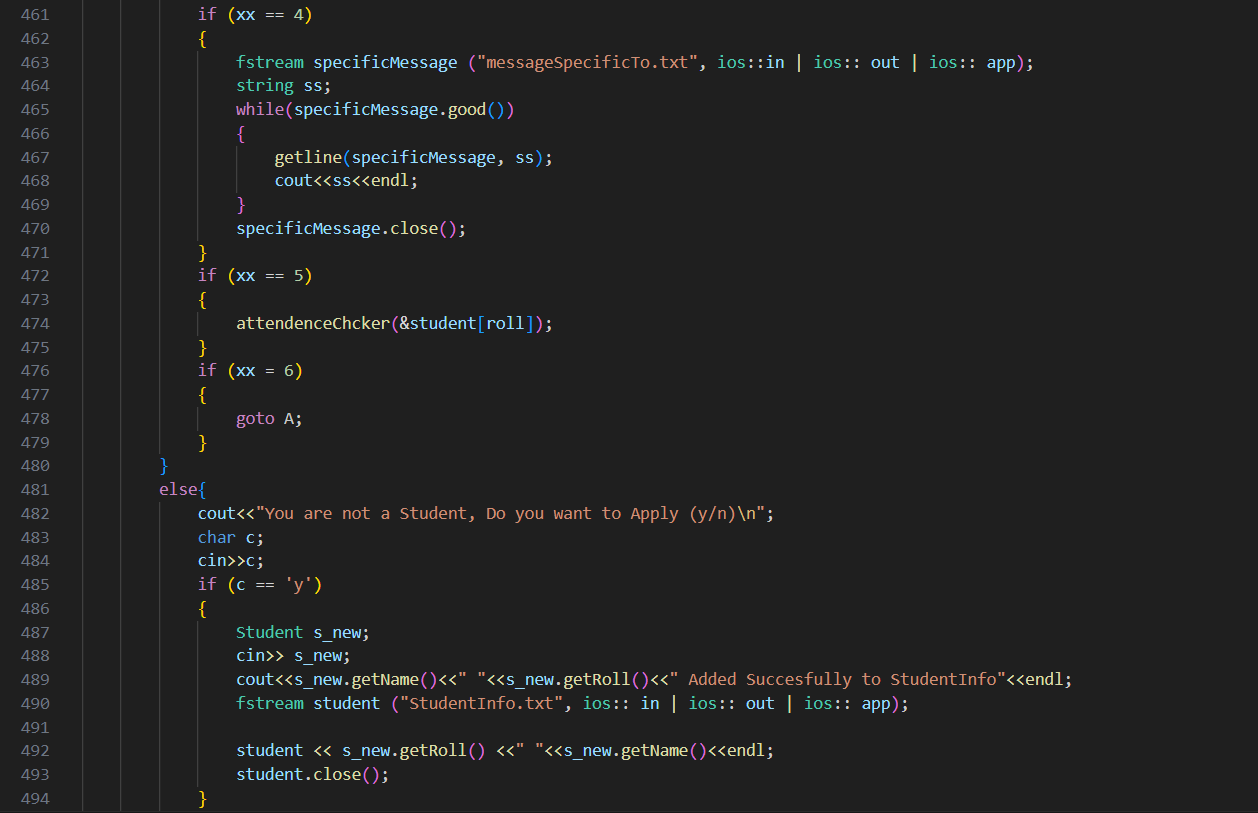
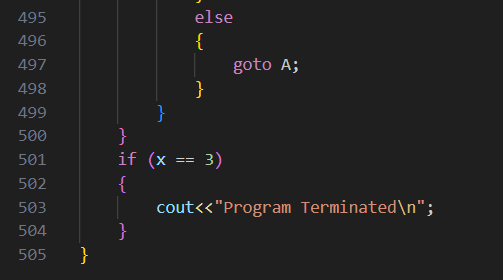
The use of emptySlot as a friend function inside the Routine class showcases the concept of friend functions in OOP. Friend functions can access private members of a class without being members of that class. This is a mechanism to allow specific external functions to interact with a class while maintaining data encapsulation.











main() function is the control center of the program, responsible for managing user interactions, reading data from files, populating arrays of Student and Teacher objects, and providing options for users to perform various actions based on their roles. It demonstrates the use of arrays, file input/output, and conditional branching to create an interactive system for managing student and teacher information.

**Conclusion**

This project was made to learn about Object-oriented programming concepts and how to use them in solving real life problems.

This was successfully done. The final project worked well but it needs some update. All the data was kept in files, and whenever needed, data was read from the file and outputted in the file, thus making it dynamic.

The main objective of making the project was fulfilled.

**References**

* [**https://www.geeksforgeeks.org/object-oriented-programming-in-cpp/**](https://www.geeksforgeeks.org/object-oriented-programming-in-cpp/)
* [**https://www.geeksforgeeks.org/file-handling-c-classes/**](https://www.geeksforgeeks.org/file-handling-c-classes/)
* **Book : Object oriented programming with C++**

**Author: Balagurussamy**