UNIVERSITY MANAGEMENT SYSTEM

Faculty of engineering

CSC301 LAB FINAL PROJECT

PRESENTED BY: DOMINIC EID – SECTION C (202201394)

INSTRUCTOR: DR. WAJDI ABBOUD

Table of contents:

PROJECT OBJECTIVES

- The main purpose of this management system is to register students and teachers in a university's engineering faculty in addition to managing their schedules and courses.
- This application allows the higher ups of the university to input and display the information of students and teachers while being able to add, remove and update said information. It also allows the user to sort and group the members of the university by age, semester started, majors and age and proceeds to save all the registered information into modifiable and protected binary files.
- This program also allows the user to set and manage the schedule of students and teachers by managing the courses taken/taught (course period, number of credits, date) and saving the information into a printable text file saved into the project directory. Note: Each student/teacher gets their own schedule text file.

DIFFICULTIES AND POSSIBLE IMPROVEMENTS

Difficulties:

- Working with binary files proved to be challenging
- Working with multiple header files
- Working with multiple classes that include inherited classes

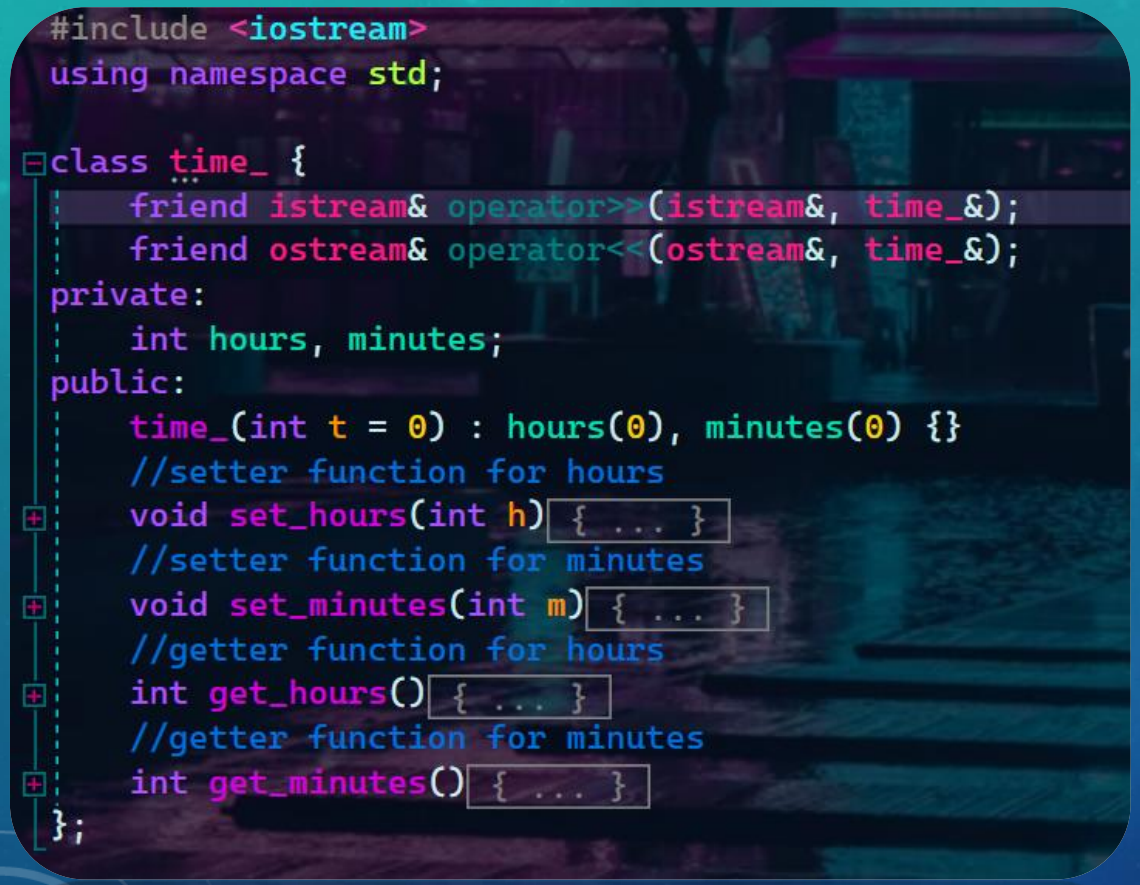
Possible improvements:

- Adding a graphical user interface
- Using SQL to store all the data in a database
- Adding more course management options

METHODS USED IN THE PROJECT

- Iterative functions (input, output, sorting, swapping,…)
- Recursive functions (search functions)
- Pointers
- Pointers to pointers
- Structures (struct semester)
- Classes (students, teachers, date, time,…)
- Objects (schedule object)
- Array of objects (array of students, array of teachers)
- Friend functions
- Inheritance
- Operator overloading (ostream, istream and += operators)
- Binary files (students file, teachers file, major files)
- Text files (schedule files)

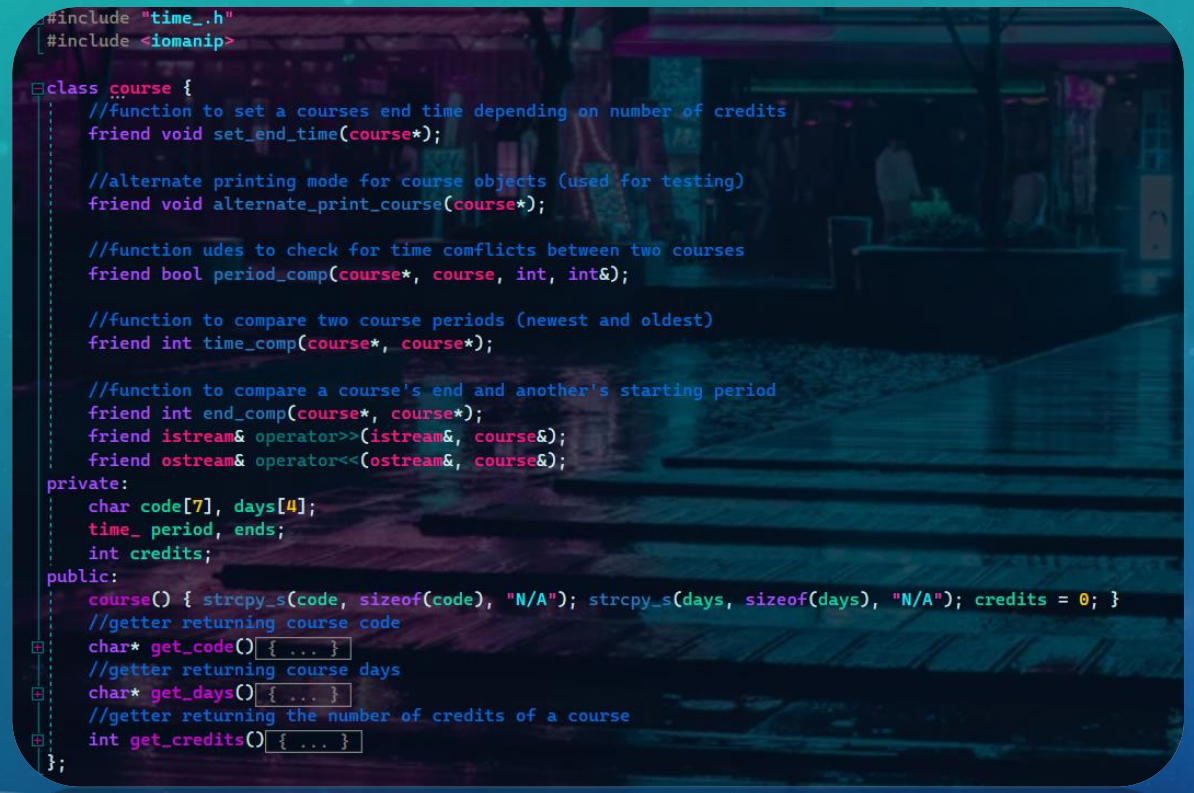
HEADER FILE – TIME_.H



```
#include <iostream>
using namespace std;

class time_ {
    friend istream& operator>>(istream&, time_&);
    friend ostream& operator<<(ostream&, time_&);
private:
    int hours, minutes;
public:
    time_(int t = 0) : hours(0), minutes(0) {}
    //setter function for hours
    void set_hours(int h) { ... }
    //setter function for minutes
    void set_minutes(int m) { ... }
    //getter function for hours
    int get_hours() { ... }
    //getter function for minutes
    int get_minutes() { ... }
};
```

- Contains time_ class
- Class contains overloaded operators << and >>
- Class contains 2 integers: hours and minutes
- Used in course class to set a course's time period

HEADER FILE – COURSE.H



```
#include "time_.h"
#include <iomanip>

class course {
    //function to set a courses end time depending on number of credits
    friend void set_end_time(course*);

    //alternate printing mode for course objects (used for testing)
    friend void alternate_print_course(course*);

    //function udes to check for time comflicts between two courses
    friend bool period_comp(course*, course, int, int&);

    //function to compare two course periods (newest and oldest)
    friend int time_comp(course*, course*);

    //function to compare a course's end and another's starting period
    friend int end_comp(course*, course*);
    friend istream& operator>>(istream&, course&);
    friend ostream& operator<<(ostream&, course&);
private:
    char code[7], days[4];
    time_ period, ends;
    int credits;
public:
    course() { strcpy_s(code, sizeof(code), "N/A"); strcpy_s(days, sizeof(days), "N/A"); credits = 0; }
    //getter returning course code
    char* get_code() { ... }
    //getter returning course days
    char* get_days() { ... }
    //getter returning the number of credits of a course
    int get_credits() { ... }
};
```

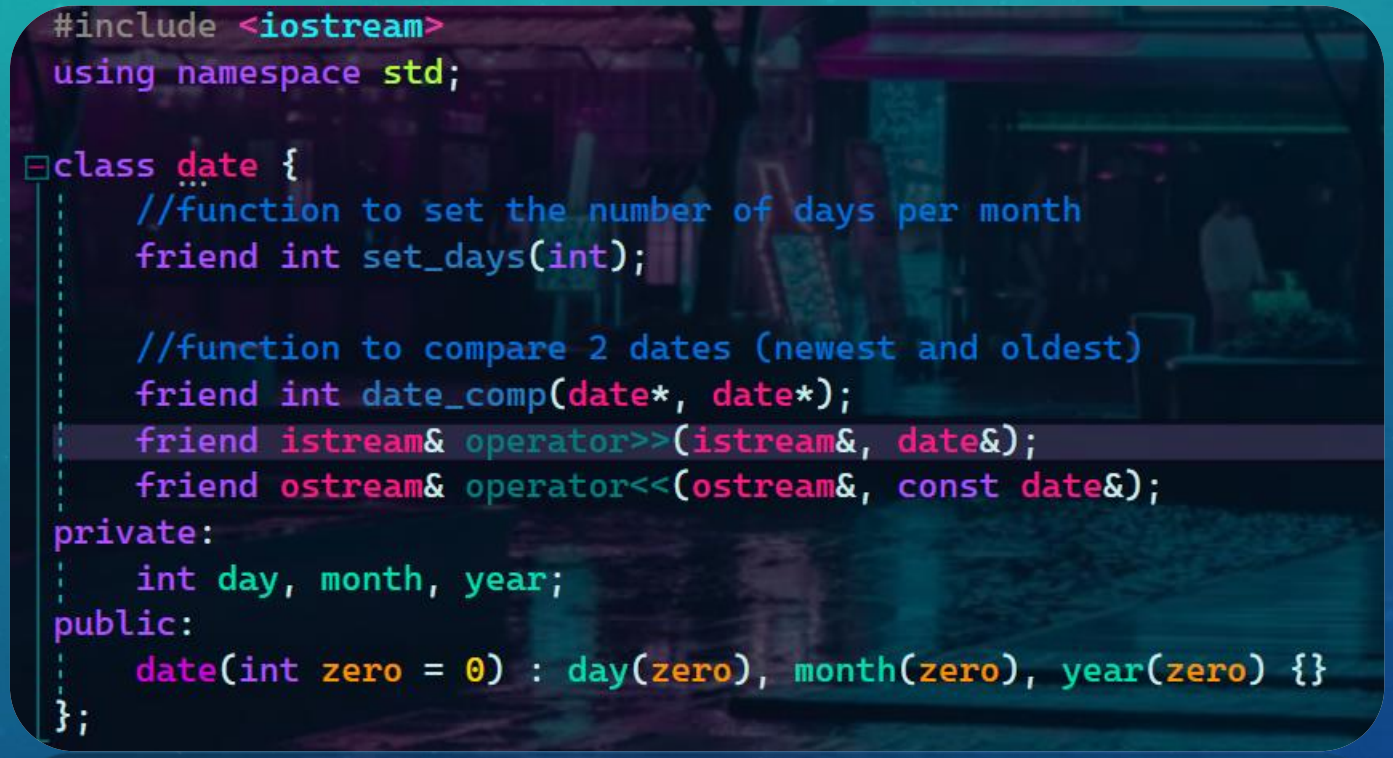
- Contains course class
- Includes time_.h header file
- Overloaded operator >> to set the course's code, the days the course takes place in, the beginning and end of a course (time_ objects) and the number of credits of a course
- Used in schedule class to set the courses taken/taught by students/teachers

HEADER FILE – SCHEDULE.H

```
//including course.h header file
#include "course.h"
#include <windows.h>
#include <iomanip>
#include <fstream>

class schedule {
    //function to read from schedule text file
    friend bool read_from_file(char[]);

    //function to input a students schdule and record it into a text file
    friend void text_schedule(schedule, char[]);

    //function to arrange courses by time taken
    friend void arrange_by_time(schedule*);
    friend istream& operator>>(istream&, schedule&);
    friend ostream& operator<<(ostream&, schedule&);

    //alernate print mode for schedule objects (used for testing)
    friend void alternate_print_schedule(schedule*);
private:
    int num_courses;
    course* courses;
public:
    //schedule class constructor
    schedule() { ... }
    //schedule class destructor
    ~schedule() { delete[]courses; }
};
```

- Contains schedule class
- Includes course.h header file
- Creating an object of type schedule allows the user to enter the number of courses taken/taught by a student/teacher, then creates an array of objects of type course allowing the user to register students and teachers into these courses

HEADER FILE – DATE.H



```
#include <iostream>
using namespace std;

class date {
    //function to set the number of days per month
    friend int set_days(int);

    //function to compare 2 dates (newest and oldest)
    friend int date_comp(date*, date*);
    friend istream& operator>>(istream&, date&);
    friend ostream& operator<<(ostream&, const date&);
private:
    int day, month, year;
public:
    date(int zero = 0) : day(zero), month(zero), year(zero) {}
};
```

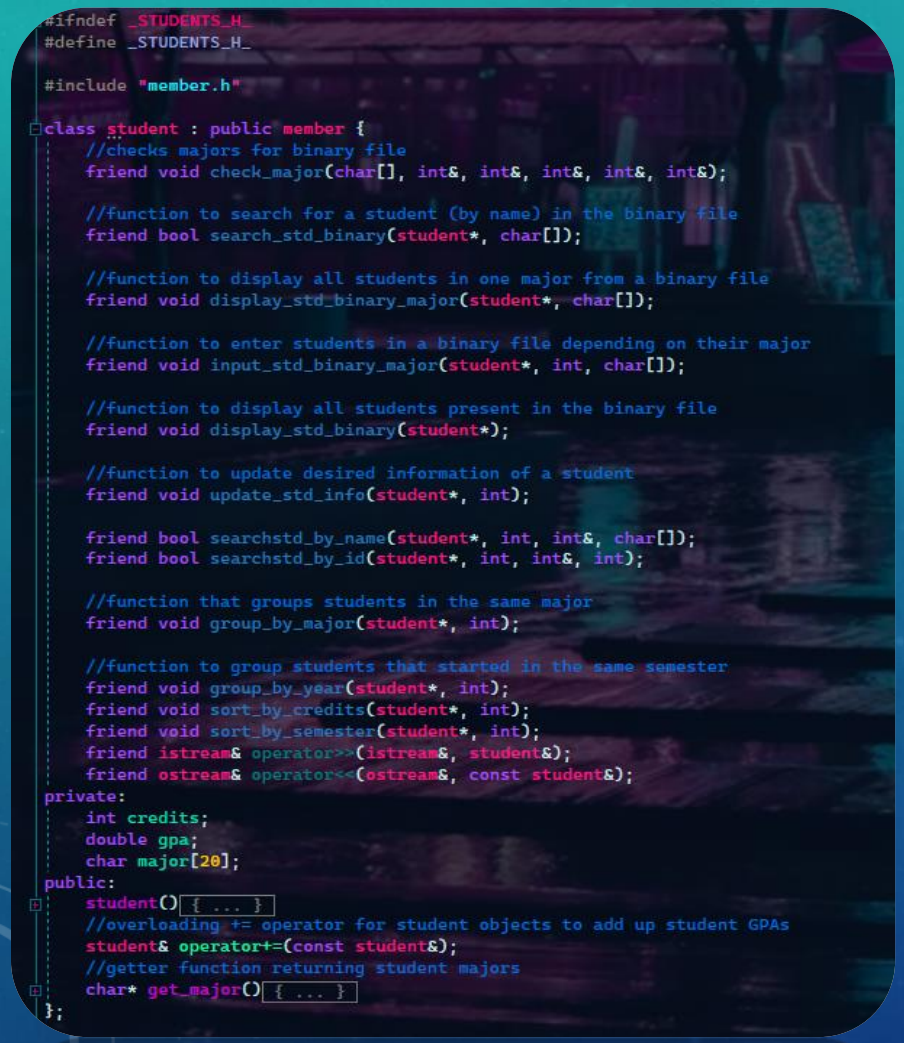
- Contains class date
- Class contains overloaded operators << and >>
- Class contains 3 integers: day, month, year
- Used in later classes to set a student/teacher's date of birth

HEADER FILE – MEMBER.H

```
#ifndef _MEMBER_H_
#define _MEMBER_H_

#include "date.h"
#include <cstring>
#include <windows.h>
#include <iomanip>
#include <fstream>

//semester structure (season-year)
struct semester { ... };

//function comparing 2 semesters (newest and oldest)
int semester_comp(semester*, semester*);

class member {
protected:
    int id;
    date birth;
    semester started;
    char gender, name[40];
public:
    member() { ... }
    //getter function to return member name
    char* get_name() { ... }
};
```

- Contains member class
- Contains semester structure
- Includes date.h header file
- Class contains common ground between teachers and students (id, date of birth, semester started, gender and name)
- Used later in student and teacher classes as an inherited class

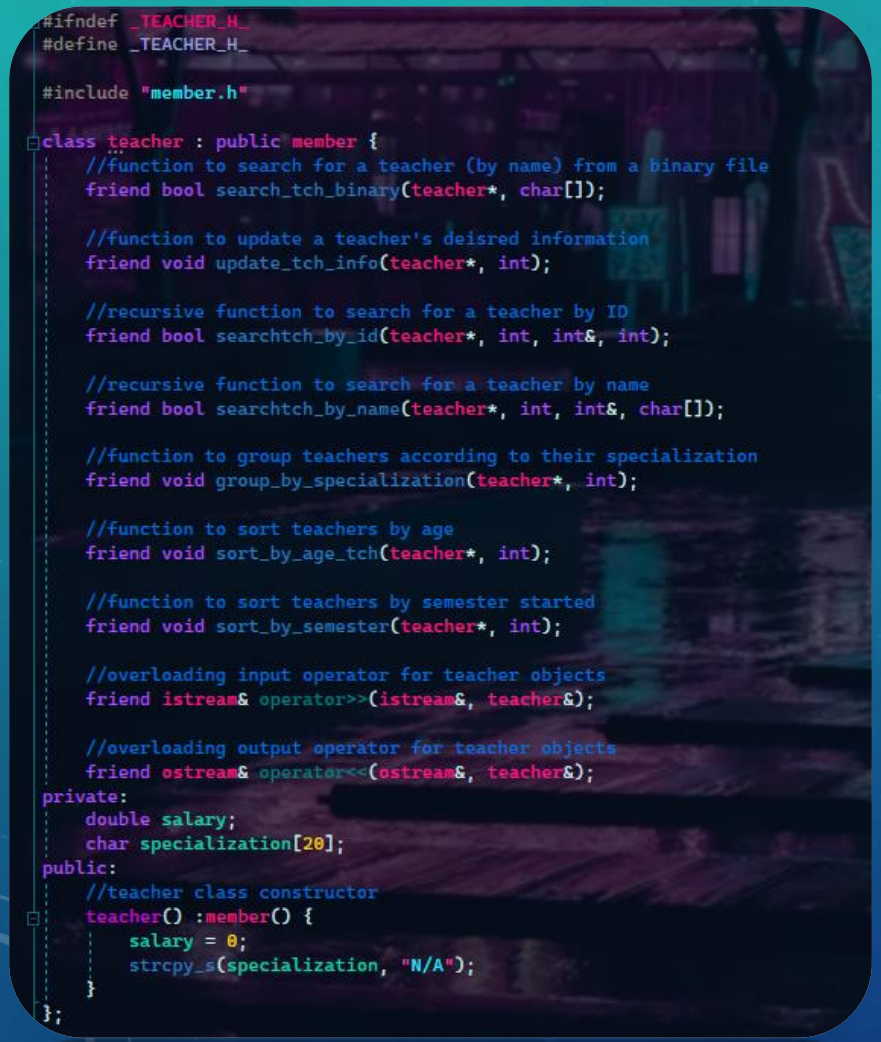
HEADER FILE – STUDENTS.H



```
#ifndef _STUDENTS_H_
#define _STUDENTS_H_

#include "member.h"

class student : public member {
    //checks majors for binary file
    friend void check_major(char[], int&, int&, int&, int&, int&);

    //function to search for a student (by name) in the binary file
    friend bool search_std_binary(student*, char[]);

    //function to display all students in one major from a binary file
    friend void display_std_binary_major(student*, char[]);

    //function to enter students in a binary file depending on their major
    friend void input_std_binary_major(student*, int, char[]);

    //function to display all students present in the binary file
    friend void display_std_binary(student*);

    //function to update desired information of a student
    friend void update_std_info(student*, int);

    friend bool searchstd_by_name(student*, int, int&, char[]);
    friend bool searchstd_by_id(student*, int, int&, int);

    //function that groups students in the same major
    friend void group_by_major(student*, int);

    //function to group students that started in the same semester
    friend void group_by_year(student*, int);
    friend void sort_by_credits(student*, int);
    friend void sort_by_semester(student*, int);
    friend istream& operator>>(istream&, student&);
    friend ostream& operator<<(ostream&, const student&);
private:
    int credits;
    double gpa;
    char major[20];
public:
    student() { ... }
    //overloading += operator for student objects to add up student GPAs
    student& operator+=(const student&);
    //getter function returning student majors
    char* get_major() { ... }
};
```

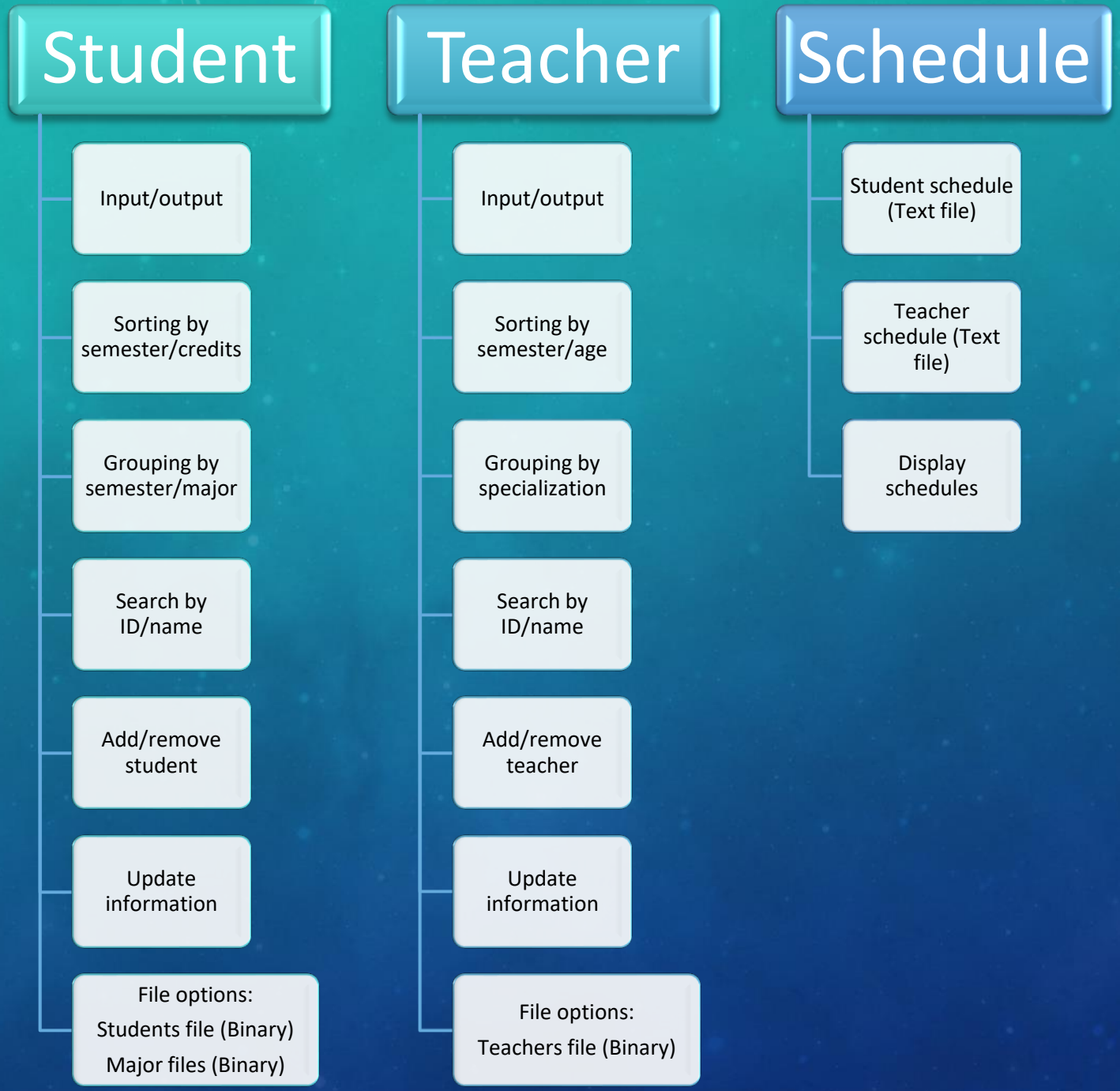
- Contains student class derived from member class
- Includes member.h header file
- Overloaded operator >> to input student information such as id, name, gender, date of birth, semester started as well as number of credits completed, GPA and major
- Overloaded operator += to sum up student GPAs and calculate the average in the binary file operation
- Contains functions to sort students by semester started and number of credits completed
- Contains functions to group students by major and semester started
- Contains a recursive function to search for students
- Contains functions that allow the user to add, remove and modify students
- Contains functions to input the students in a binary file and in another according to their major and display or remove the saved info

HEADER FILE – TEACHERS.H



```
#ifndef _TEACHER_H_
#define _TEACHER_H_

#include "member.h"

class teacher : public member {
    //function to search for a teacher (by name) from a binary file
    friend bool search_tch_binary(teacher*, char[]);

    //function to update a teacher's deisred information
    friend void update_tch_info(teacher*, int);

    //recursive function to search for a teacher by ID
    friend bool searchtch_by_id(teacher*, int, int&, int);

    //recursive function to search for a teacher by name
    friend bool searchtch_by_name(teacher*, int, int&, char[]);

    //function to group teachers according to their specialization
    friend void group_by_specialization(teacher*, int);

    //function to sort teachers by age
    friend void sort_by_age_tch(teacher*, int);

    //function to sort teachers by semester started
    friend void sort_by_semester(teacher*, int);

    //overloading input operator for teacher objects
    friend istream& operator>>(istream&, teacher&);

    //overloading output operator for teacher objects
    friend ostream& operator<<(ostream&, teacher&);
private:
    double salary;
    char specialization[20];
public:
    //teacher class constructor
    teacher() :member() {
        salary = 0;
        strcpy_s(specialization, "N/A");
    }
};
```

- Contains teacher class derived from member class
- Includes member.h header file
- Overloaded operator >> to input teacher information such as id, name, gender, date of birth, semester started as well as the teacher's salary and specialization
- Contains functions to sort teacher by semester started and age
- Contains functions to group teachers by specialization
- Contains a recursive function to search for teachers
- Contains functions that allow the user to add, remove and modify teachers
- Contains functions to input the teachers in a binary file as well as display and remove saved info


Student
Input/output
Sorting by semester/credits
Grouping by semester/major
Search by ID/name
Add/remove student
Update information
File options:
Students file (Binary)
Major files (Binary)
Teacher
Input/output
Sorting by semester/age
Grouping by specialization
Search by ID/name
Add/remove teacher
Update information
File options:
Teachers file (Binary)
Schedule
Student schedule (Text file)
Teacher schedule (Text file)
Display schedules

EXPLANATION OF SOME DIFFICULT FUNCTIONS

“remove_std_binary” and “remove_tch_binary” functions:

These two functions open the original binary file and a new temporary binary file, then goes through the original in read mode and searches for the position of the desired student/teacher to remove.
The function then stores the position in an integer, then starts to write the info into the temporary file while using the seekg() function to skip over the position of the desired student/teacher.
The function then closes the files and removes the original files using the remove() function, and then proceeds to rename the new files as the older files using the rename() function.

“update_std_info” and “update_tch_info” functions:

These two function allow the user to search for the student/teacher by name or ID using the searchstd_by_id and searchstd_by_name or the searchtch_by_id and searchtch_by_name function by way of a switch operation.
The function then gives the user the options for updating also by way of a switch case.
The changes are applied to the array according to the position returned by the search functions

Overloaded operator+= function:

This function overloads the += operator for student objects.
It’s a member function of class student and it consists of adding up the student’s GPAs

“period_comp” function:

This function compares two courses by checking if they take place on the same days, then proceeds to check if the two courses start at the same time and checks if there’s a time conflict between the end time of the first course and the starting time of the second course.
The function returns true if there’s a conflict and returns false otherwise.

“semester_comp” function:

This function compares two semesters by first checking the year, then compares the seasons (fall, spring, summer) using strcmp().
The function returns 1 if the first semester is more recent, -1 if the second semester is more recent and 0 if both are the same semester.

“text_schedule” function:

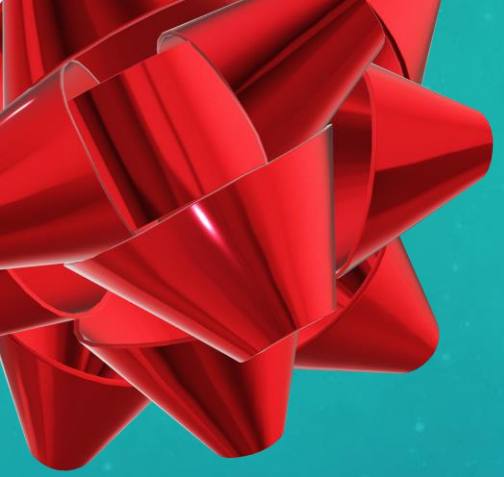
This function creates a text file with the student/teacher’s name and write all the courses (time, days, code) into a calendar format

“read_from_file” function:

This function takes the name of the student/teacher and opens their specific schedule file.
The function proceeds to read line by line the text file containing the schedule and outputs it directly on the terminal.

“set_end_time” function:

This function takes the number of credits of a course and according to that number, sets the end time of each course: 3 hours for 1 credit courses and 1.5 hours for 2 and 3 credit courses.



That is all for this project

Thank you for your time and patience

Merry Christmas and happy new year! 🎄