VIET NAM NATIONAL UNIVERSITY HO CHI MINH CITY
University of Science
Faculty of Information Technology

Programming Techniques

Data Storage Report

Lecturer:

Đinh Bá Tiến – Trương Phước Lộc – Đào Nguyên Kha

This report delves into the storage of data for the esteemed group endeavor, namely the ***Course Management System,*** which has been meticulously crafted as part of the ***Programming Techniques*** course.

22127488 Trương Thanh Toàn

22127391 Nguyễn Xuân Thành

22127254 Trương Nguyễn Hiền Lương

22127101 Lý Ngọc Hân

22126058 Nguyễn Minh Đạt

Group 1

May 4th, 2023

CONTENTS

I. ENUMERATIONS AND STRUCTURES

1. Enumerations

An enumeration is a data type that allows programmers to define a collection of named values.

`enum Program { APCS, CLC, VP };`	An enumeration used for representing the program that a student attending in.
`enum WeekDay {` `  MON, TUE, WED, THU, FRI, SAT, SUN` `};`	Represents weekday in the program, used for day performance of a course.
`enum Session { S1, S2, S3, S4 };`	Represents the shift that a course performs in the program.
`enum Type { Staff, Student };`	Represents type of the user.

2. Structures

a) Linked list

`template <typename data_type>` `  struct SLL {` `  // to declare a node of a singly linked list of integers: SLL<int> *head;` `  data_type data;` `  SLL<data_type> *next = nullptr;` `  };`	A node of a singly linked list that encapsulates three components: - `data`: the data type of `data` will be passed to the declaration of the instance as a template argument when this structure is called. - `next`: a pointer that points to the next node.
`template <typename data_type>` `  struct DLL {` `  // to declare a node of a doubly linked list of integers: DLL<int> *head;` `  data_type data;` `  SLL<data_type> *prev = nullptr;` `  SLL<data_type> *next = nullptr;` `  };`	A node of a doubly linked list that encapsulates three components: - `data`: the data type of `data` will be passed to the declaration of the instance as a template argument when this structure is called. - `prev`: a pointer that points to the previous node. - `next`: a pointer that points to the next node.
`template <template <typename> list_type, typename data_type>` `  struct LIST {`	A structure that keeps the first and the last node of a type linked list `list_type`. - `head`: the first node of the list

```
        // to declare a doubly linked list
`DLL` of integers: LIST<DLL, int> list;
        list_type<data_type> *head = nullptr;

     list_type <data_type> *tail = nullptr;
    };
```

- `tail`: the last node of the list

b) User

```
struct USER {
    std::string username = "";
    std::string password = "";
};
```

Represents a user account, including username and password.

```
struct STUDENT {
    unsigned int yearIn = 0;
    std::string firstname = "",
            lastname = "",
            studentID = "",
            socialID = "";
    const Type type = Student;
    USER user;
    DATE dateOfBirth;
    bool gender = 0; // 0: male && 1:
female
    CLASS* Class;
    LIST<DLL, COURSE*> courses; // list of
courses that the student enroll in in the
current semester
};
```

Represents a student in the school.
The struct has several member variables with different data types, including:

- An `unsigned integer` named `yearIn`: the enrollment year of this student.
- A `std::string` named `firstname / lastname / studentID / socialID`: firstname / lastname / student ID / social ID of this student.
- A constant variable of `Type` named `type`: to tell the system that this is a student or a staff. This component is unchangable.
- A `USER` variable named `user`: represents for the user account (including username and password only) associated with the student.
- A `DATE` variable named `dateOfBirth`: represents the student's date of birth.
- A `boolean` named `gender`: represents the gender of this student (1 for male, 0 for female).
- A `pointer` of `CLASS` named `Class`: contains the address of the data member inside the node of list the `L_Class` (*II – 3*) in

	which this student is participating.
`struct STAFF { std::string firstname = "", lastname = ""; const Type type = Staff; USER user; };`	Represents a staff in the school. Consists of 4 components: `first name`, `last name`, `type` of user (`constant`, set to `Staff`) and `user` account information.

c) Date time – Scholastic year – Semester

`struct DATE { unsigned int year = 0; unsigned short day = 0, month = 0; };`	Represents a semester in the school year. Encapsulates 4 components: - A `unsigned integer` named `year` is used to store the year of the date - Two `unsigned short` integers named `day` and `month` are used to store the day and month of the date.
`struct SCHOOLYEAR { unsigned int begin = 0, end = 0; SEMESTER *sem1 = nullptr, *sem2 = nullptr, *sem3 = nullptr; };`	Represents a scholastic year in the school. Consists of 5 components: - Two `unsigned integers` named `begin` and `end` represent for starting year and ending year of this school year. - Three `pointer` of `SEMESTER` named `sem1`, `sem2` and `sem3` pointing to the corresponding semester of this school year.
`struct SEMESTER { int No = 0; DATE startdate, enddate; LIST<DLL, COURSE*> courses; };`	Represents a semester in the school year. Encapsulates 4 components: - `No`: ordinal number of this semester (1, 2 or 3 only). - Two variables of `DATE` named `startdate` and `enddate` represent the starting date and ending date of this semester

	- A `doubly linked list` of `pointers` of `COURSE` which contains the list of the courses performing in this semester.

d) Class – Course – Scoreboard

`` struct CLASS { unsigned int yearIn; unsigned short K = 0, No = 0; Program program; // for example: 22CLC2, K = 22, No = 2, Program = CLC (an enumeration) LIST<DLL, STUDENT*> students; // member function std::string convertToString(); }; ```	Represents for a class of students in the school. Consists of 5 components and 1 member function: - A `unsigned integer` named `yearIn` represents the enrollment year of the students in this class. - Two `unsigned short` integers named `K` and `No`: + `K = yearIn % ((yearIn / 1000) * 1000).` + `No` is the ordinal number of this class in the year it was created. - A `Program` variable named `program` represents the program that the students in this class study in (including `CLC`, `VP` and `APCS`). - A `doubly linked list` of `pointers` of `STUDENT` named `students` contains students that attend in this class. This list does not contain any new nodes but uses the same node with the `L_Student`. - Function `convertToString()` is a member function which converts a class to a `std::string` that represents the name of that class. It needs to be accessed using dot operator.
`` struct COURSE { std::string ID = “”, name = “”, teacher = “”; ```	Represents a course a certain semester. Encapsulates 8 components and 1 member function:

```
    unsigned short credit = 0, maxStudents =
50;
    WeekDay day = MON;
    Session session = S1;
    LIST<DLL, SCOREBOARD*> students;

    // member function
    void add1Student(SCOREBOARD* student);
};
```

- Three `std::string` named `ID`, `name` and `teacher` represent the course `ID` / course `name` the `teacher` who teaches this course.
- Two `unsigned short` integers named `credit` and `maxStudents` which represent for number of credits of this course and the maximum students (or the capacity) of the course.
- A `WeekDay` variable named `day` represents the day on which this course will perform.
- A `Session` variable named `session` represents the session on which this course will perform
- A `doubly linked list` of `pointers` of `SCOREBOARD` named `students` which contain all the nodes of student enrolling in this course and their scores.
- Function `add1Student(SCOREBOARD* student)` is a member function that add a new student with his/her score to the list `students`.

```
struct SCOREBOARD {
    double totalMark = 0, finalMark = 0,
midtermMark = 0, otherMark = 0;
    STUDENT* student = nullptr;
};
```

Represents a score board of a certain student in a certain course.
Encapsulates 5 members:

- Four `double` variables named `totalMark`, `finalMark`, `midtermMark` and `otherMark` for marks.
- A `pointer` of `STUDENT` named `student`, which is set to `nullptr` by default, points to the student that got these scores.

II. GLOBAL LISTS AND GLOBAL VARIABLES

1. Why is it essential to set the scope of these lists and variables to global?

- To ensure that these lists and variables can be accessed or modified without having to pass them as arguments to multiple functions.
- To facilitate communication and data sharing between different parts of the program.

2. Global variables used in the source code

`SCHOOLYEAR* g_currentSchoolYear = nullptr;`	Keeps the pointer that points to the current school year.
`SEMESTER* g_currentSemester = nullptr;`	Keeps the pointer that points to the current semester.
`STUDENT* g_currentStudent = nullptr;`	Keeps the pointer that points to the current student (the user that is logging in the system; otherwise, this pointer will point to **`nullptr`**) .
`STAFF* g_currentStaff = nullptr;`	Keeps the pointer that points to the current staff (the user that is logging in the system; otherwise, this pointer will point to **`nullptr`**) .
`std::string lastestUsername = "";`	If `*remember*` check box is selected, the system will assign the user name of the most recent user who logged in to it.
`std::string latestPassword = "";`	If `*remember*` check box is selected, the system will assign the password of the most recent user who logged in to it.
`bool latestCheckRememberLogin = false;`	When `*remember*` check box is checked, it itself will be set to **`true`**.

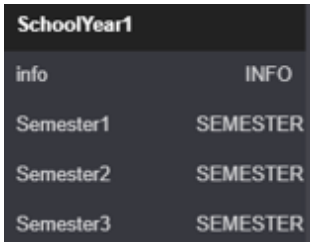
3. Global lists used in the source code

`LIST<DLL, SCHOOLYEAR*> L_SchoolYear;`	A doubly linked list of **`SCHOOLYEAR*`**. Encapsulates a comprehensive roster of all the scholastic year that the system has traversed. Each node of that list consists of: - `data` whose data type is **`SCHOOLYEAR*`**. - `next` which is a pointer of that node points to the next node.

	- prev which is a pointer of that node points to the previous node.
`LIST<DLL, STAFF*> L_Staff;`	A doubly linked list of **STAFF***. Encapsulates a comprehensive roster of all staff members. Each node of that list consists of: - data whose data type is **STAFF***. - next which is a pointer of that node points to the next node. - prev which is a pointer of that node points to the previous node.
`LIST<DLL, STUDENT*> L_Student;`	A doubly linked list of **STUDENT***. Encapsulates a comprehensive roster of all students in this school. Each node of that list consists of: - data whose data type is **STUDENT***. - next which is a pointer of that node points to the next node. - prev which is a pointer of that node points to the previous node.
`LIST<DLL, CLASS> L_Class;`	A doubly linked list of **CLASS**. Encapsulates a comprehensive roster of all classes in this school. Each node of that list consists of: - data whose data type is **CLASS**. - next which is a pointer of that node points to the next node. - prev which is a pointer of that node points to the previous node.

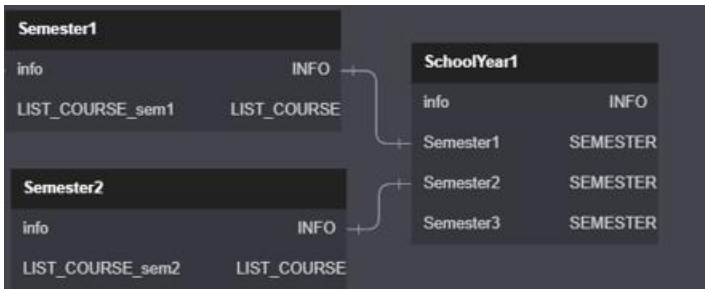
III. DATA STORAGE IN DETAILS

L_Schoolyear includes all school years; each node in the list consists of data stored by data type SCHOOLYEAR and 2 pointers: next and prev pointing to other school years – the year after and the previous year. Each SCHOOLYEAR data encapsulates a total of 3 semesters: sem1, sem2, sem3 which have the data type SEMESTER*. It



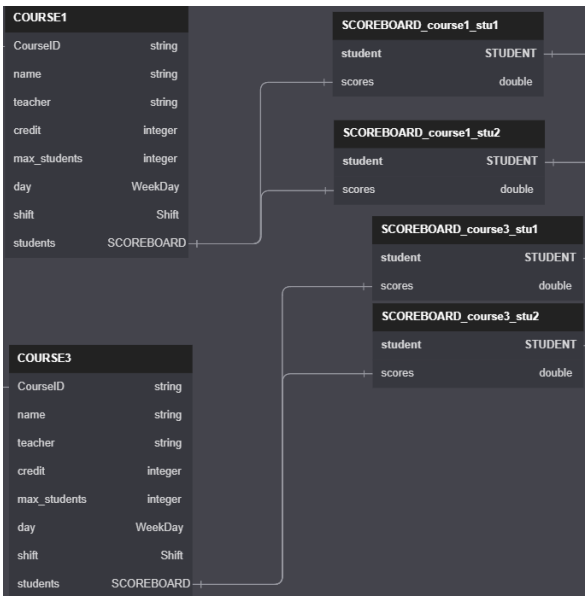
also contains the begin and end year of that school year stored by unsigned integer data type.

Each SEMESTER in a year will include the start and end date stored in a variable of DATE. It also stores the ordinal number of this semester in a school year. Especially, it includes a list of courses performing in this semester.

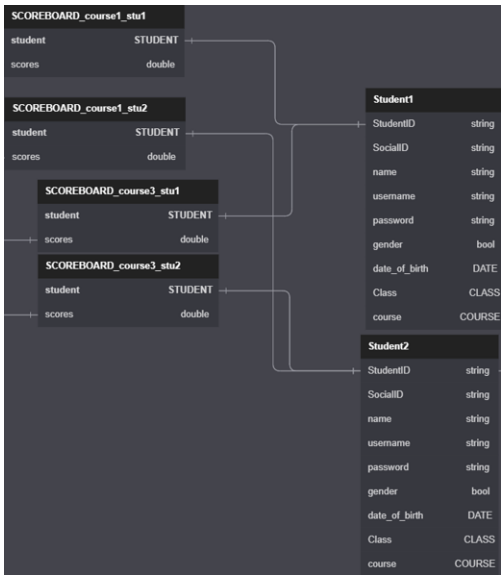


This list of courses includes nodes that each node contains data stored in the COURSE* data type. It points to courses which are performed in that semester. Each node of which is a variable of COURSE* data type has 2 pointers: next and prev, pointing to the previous course and the next course. Each COURSE includes a course's information, which are consists of course ID, course name, teacher name stored as string data type; number of credits, maximum number of students stored as integer data type, performing date stored as data type WeekDay, shift on which the course will be performed stored as SESSION data type and finally a list of students who enroll in that course.

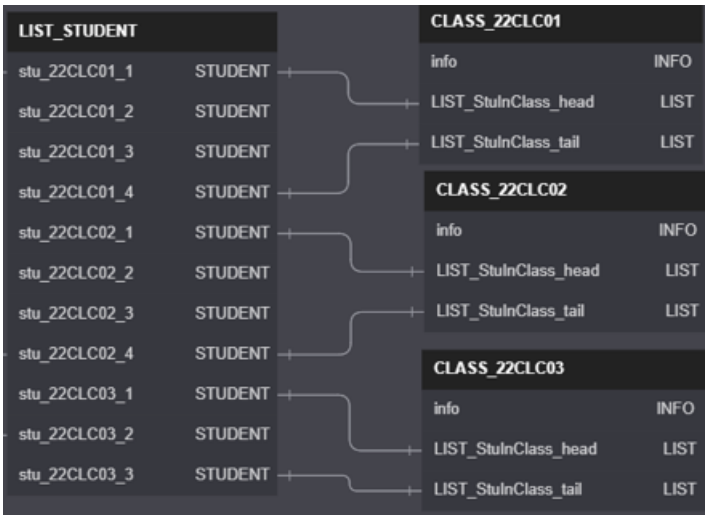
The list of students includes nodes that each node contains data stored by SCOREBOARD*, pointing to related information of each student. In detail, each SCOREBOARD includes the data of different types of score in that course stored as double data type and student's personal information stored as STUDENT data type.



Each STUDENT includes student ID, social ID, name (firstname and lastname), username, password; all stored as string data type, gender stored as bool data type, date of birth stored as DATE data type, class stored as CLASS* point to class student takes part in, courses student participate in stored as a list of COURSE* point to courses student joins.



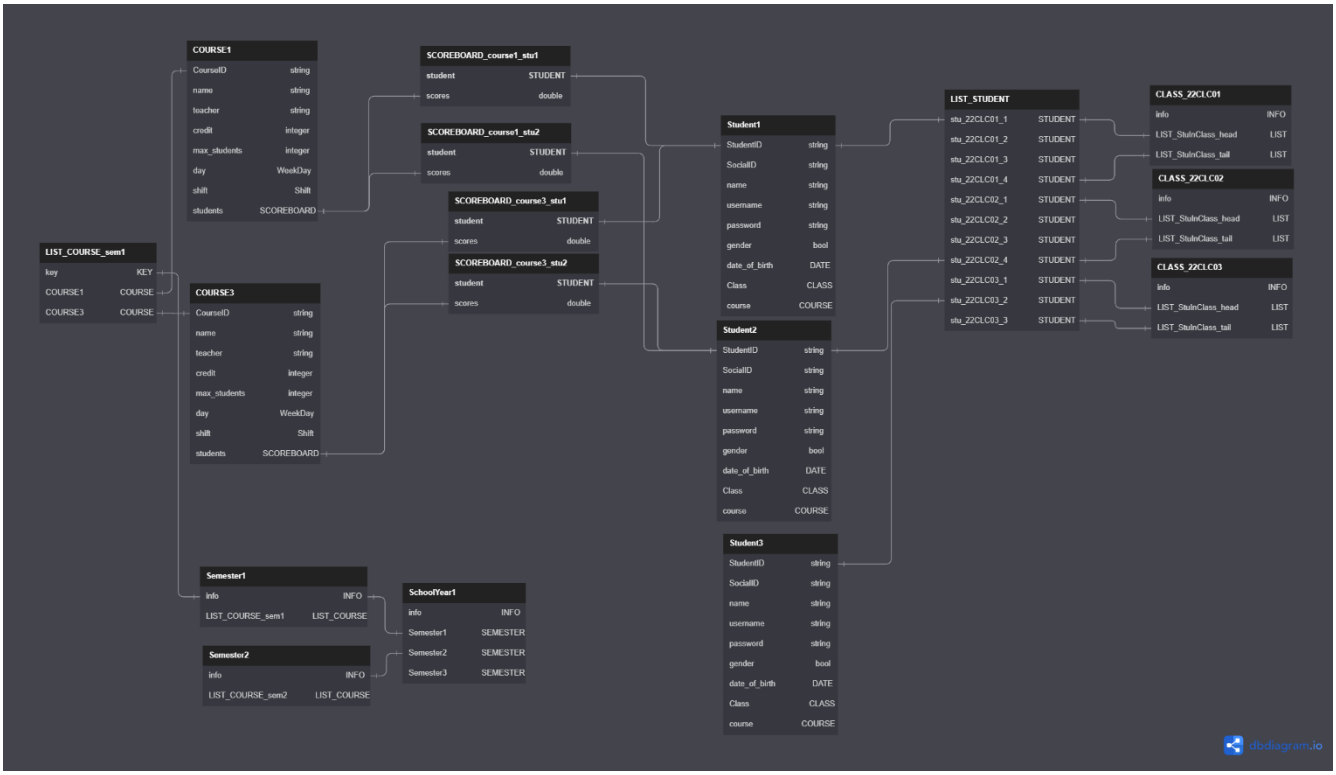
L_Student includes students in the current school year, each node includes data with data type STUDENT and 2 pointers next prev pointing to other students. Each STUDENT data includes student ID, social ID of that student, name, username password stored as string data type, gender stored as bool data type, date of birth stored as DATE data type, courses student participate in stored as a list of COURSE* point to courses student joins, class stored as CLASS* point to class student takes part in in **L_Class**, student of each class are pointed to students in the same class until all students of that class are pointed to then point to students of another class and repeat until the end.



L_Class includes nodes, each node consists of data with CLASS data type stored a class in this school year and 2 pointers next prev pointing to other classes: the next class and the previous class in that school year. Each CLASS data includes year which students in that class entered school stored as unsigned integer, program of that class stored as Program, class number stored as unsigned short and list students in that class which is a sublist of **L_Student**, this sublist has head point to first student of class and tail point to last student of class.

L_Staff includes nodes, each node contains difference staff information stored as STAFF data type and 2 pointers next prev pointing to the next staff and the staff before. Each STAFF data includes firstname lastname stored as string and username password stored as string, too.

The full diagram of data storage:



IV. STRUCTURE OF SOUCRE CODES

Root directory: **CMS** (stands for Course Management System)

In **CMS** we have:

- A **main.cpp** file where everything starts.
- A **Structs.h** file contains all the structures used in the program.
- A header file **.h** and a implement file **.cpp** of **exporting** data.
- A header file **.h** and a implement file **.cpp** of **importing** data.
- A header file **.h** named **GlobalVariables** contains all global variables and global lists.
- A header file **.h** and an implement file **.cpp** named **helperFunctions** contains all helper functions used in the whole program.
- A **header.h** file which was included all the header files that could be used in each form (in the other words, it is more easy to include multiple header files at once).
- A **main** form used for greeting.
- 24 forms which represents for 24 functions (or 24 tasks).
- A **Logout.cpp** contains all the implementations of **logout_buttonClick** of all the forms in the source code (these **logout_buttonClick** functions were brought out of the its form to avoid *circular dependencies*).
- A **login** form used for logging in.
- An **about us** form used for group credits.
- A **profile** form used for displaying user's information.
- A **CSV** directory for data base, including:
  - `Class.csv` (all classes in the school).
  - `SchoolYear.csv` (all school years in the system). Each line contains 1 school year. Each school year includes its information, the validation of semesters (1 is available, 0 is not) and the relative path to file that contains information of all semester of this school year.
  - `Staff.csv` (all staffs in the school).
  - `Student.csv` (all students in the school).
  - `DATA.csv` (the last user loging in the system, for remember me function).
  - A `SemInSchoolYear` directory including:
    - All the files each of which represents a school year in the system. One file contains the information of this school year and a relative path to the file containing all the course that will be performed in this semester.
    - A `CourseInSem` directory including:
      - All the files each of which represents a semester of a school year. One file contains information of this semester, all course will be performed and a relative path to the file containing all the student enrolling in each course and their scores.
      - A `StudentsInCourse` directory including:

- All the files each of which represents a course in a certain semester of a school year. One file contains all the students enrolling in this course including: student ID and their scores.

Note: Each form includes 1 header file `.h` and 1 `.resx` file for resources (.cpp file had been removed so that the source codes could be more clean).