**PROGRAMMING TECHNIQUES**

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***

**Data Storage Report**

# Contents

[Contents 2](#_Toc133500125)

[I. ENUMERATIONS AND STRUCTURES 3](#_Toc133500126)

[1. Enumerations 3](#_Toc133500127)

[2. Structures 3](#_Toc133500128)

[a) SLL 3](#_Toc133500129)

[b) DLL 3](#_Toc133500130)

[c) LIST 4](#_Toc133500131)

[d) USER 4](#_Toc133500132)

[e) STUDENT 4](#_Toc133500133)

[f) SCHOOLYEAR 5](#_Toc133500134)

[g) DATE 5](#_Toc133500135)

[h) SEMESTER 5](#_Toc133500136)

[i) CLASS 6](#_Toc133500137)

[j) COURSE 7](#_Toc133500138)

[k) SCOREBOARD 7](#_Toc133500139)

[l) STAFF 8](#_Toc133500140)

[II. GLOBAL LISTS AND GLOBAL VARIABLES 8](#_Toc133500141)

[1. Why is it essential to set the scope of these lists and variables to global? 8](#_Toc133500142)

[2. Global variables used in the source code 8](#_Toc133500143)

[3. Global lists used in the source code 9](#_Toc133500144)

[III. DATA STORAGE IN DETAILS 10](#_Toc133500145)

[IV. STRUCTURE OF SOUCRE CODES 10](#_Toc133500146)

# ENUMERATIONS AND STRUCTURES

## 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. |

## Structures

### SLL

|  |  |
| --- | --- |
| **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. |

### DLL

|  |  |
| --- | --- |
| **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. |

### LIST

|  |  |
| --- | --- |
| **template <template <typename> list\_type, typename data\_type>**  **struct 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;**  **};** | A structure that keeps the first and the last node of the corresponding linked list.   * head: the first node of the list * tail: the last node of the list |

### USER

|  |  |
| --- | --- |
| **struct USER {  std::string username = “”;**  **std::string password = “”;**  **};** | Represents a user account, including username and password. |

### STUDENT

|  |  |
| --- | --- |
| **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*](#L_Class_definition)) in which this student is participating. |

### SCHOOLYEAR

|  |  |
| --- | --- |
| **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. |

### DATE

|  |  |
| --- | --- |
| **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. |

### SEMESTER

|  |  |
| --- | --- |
| **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. |

### CLASS

|  |  |
| --- | --- |
| **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. |

### COURSE

|  |  |
| --- | --- |
| **struct COURSE {**  **std::string ID = “”,**  **name = “”,**  **teacher = “”;**  **unsigned short credit = 0, maxStudents = 50;**  **WeekDay day = MON;**  **Session session = S1;**  **LIST<DLL, SCOREBOARD\*> students;**  **// member function**  **void add1Student(SCOREBOARD\* student);**  **};** | Represents a course a certain semester.  Encapsulates 8 components and 1 member function:   * 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. |

### SCOREBOARD

|  |  |
| --- | --- |
| **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. |

### STAFF

|  |  |
| --- | --- |
| **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. |

# GLOBAL LISTS AND GLOBAL VARIABLES

## 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.

## 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**. |

## 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. |

# DATA STORAGE IN DETAILS

fdfdvd

# STRUCTURE OF SOUCRE CODES

The root folders includes

Including a login form, a starting menu which contains all the buttons leading to the corresponding functional form and these functional forms

There are two sort of users: STUDENT and STAFF

[Đối với student thì code sẽ hoạt]