**Course Number: 420-P16-AS**  
**Course Title: Structured Programming**  
**Teacher: Quang Hoang Cao**  
**Session: Autumn 2015**  
**Group: 7150**

**Final Project**

**BY**

**ALIREZA GOODARZI**

**Introduction**

This program could temporarily keep information about a teacher and his/her students who could be part a group and choose several courses from the course list that's available in that group.

The program name is **STUDENTS GRADE DATA SYSTEM**

**Analysis and Design**

I analyzed and designed the program to be reusable and dynamic as far as possible.

Part of this aim could be seen in data structure design. The structure variables that keep the data mimicked the tables of a database and using common variables inside structures mimic primary and foreign keys as possible. A good result of this design is that we could easily add more teachers, groups, courses and students without any changes in the data structure.

At the same time it is almost ready to use a real database system, replacing data structures with tables and their field.

In order to simplify the passage of arguments and avoid using global variables and unsecure coding, and temporary data structure is implemented in program, named Buffer, which keeps some essential information like active user's user ID and type of user as teacher or student.

Handling several structures and passing their reference as an argument was a big challenge in this program.

In order to provide assistance and save time for program test, I already added 5 students as default and users can add 4 more students to check the functionality of the program.

To add more student user could increase the MAX variable in main () function of finalProjetcApp.cpp file.

For verification of default student list, please check initialValues () function in operationImp.cpp file.

To test a student, you can use student ID:1234567 pass:1234567

**Functions**

Here is a list of the functions I used in this program, they are in different files with related name and functionality.

|  |  |
| --- | --- |
| Function Name | Description |
| int checkDigits(); |  |
| char getMenuChoiceOK(int); |  |
| int intValue(int,string); |  |
| string strValue(string); |  |
| string firstLettersCapital(string); |  |
| string removeExcessSpace(string); |  |
| string shortenString(string,int); |  |
| char myYesNo(string); |  |
| char exitByCTRLX(); |  |
| int exitPrompt(string,int); |  |
| int strUser(); |  |
| float floatValue(int, string); |  |
| void welcomePage(); |  |
| void pageTitle(string); |  |
| void programTitle(); |  |
| void formTitle(string, int&, int); |  |
| void pausePrompt(string); |  |
| void myCout1(string, int, int ); |  |
| int login(); |  |
| int addUser(); // teacher& student |  |
| int checkDuplicateID(); |  |
| int addGroup(); |  |
| int addCourse(); |  |
| int footer(); |  |
| void middleText(string); |  |
| bool stdCrsDuplicate(StudentResult[],string, int, int&); |  |
| int finalResult(float, float , float ); |  |
| string getFullName(UserProfile[], int, int); |  |
| void reportTitle(); |  |
| void listDisplay(); |  |
| void initialValues( UserProfile[], Groups[], Course[], int, Buffer\*, int&, StudentResult[]); |  |
| string newNameFamily(string); |  |
| int chooseGroup(Groups[], int); |  |
| void chooseCourse(Course[], int, Buffer\*); |  |
| string leftMarginSpace(string[], int); |  |
| void listHeader(string[],int); |  |
| void teacherCoursList(Course[], string[], int[], Buffer \*); |  |
| void titleLength(int[], string[]); |  |
| void getCourseInfo(Course[], string , Buffer \*, int ); |  |
| void getTeacherInfo(UserProfile[], Buffer \*, int); |  |
| int listcourses(); // teacher & student |  |
| int mainMenu(); |  |
| int teacherMenu(); |  |
| int studentMenu(string); |  |
| int teacherGradeSearchMenu(); |  |
| int teacherOperations( int, int&, UserProfile[], Course[],Groups[],StudentResult[], Buffer \*); |  |
| int teacherCourses(Course[], Buffer \*); |  |
| int addStudentGrade( UserProfile[], Course[], Groups[], StudentResult[], Buffer \*, int, int&); |  |
| int newStdID(UserProfile[], int&,int); |  |
| int checkIdDuplication(UserProfile[], int&, int); |  |
| void displayNewStudent(Buffer\*); |  |
| void getStudentGrade(StudentResult[],Buffer\*, int &); |  |
| int searchStd(int); |  |
| int searchStd(string,char); |  |
| int sortStudent(); |  |
| int listCourseStudentsGrades(); |  |
| void saveStudentInfo(UserProfile[], StudentResult[], Buffer \*, int&); |  |
| void teacherSearchGradeOperation(UserProfile[], StudentResult[], Course[], Buffer \*, int &,int); |  |
| void gradeList(StudentResult [], Buffer \*, int &); |  |
| void courseGradeList(StudentResult[], Groups[], Course[], UserProfile[], Buffer \*, int , int); |  |
| void sortStudentList(UserProfile[],int, Buffer\*); |  |
| void courseStudentList(UserProfile[], int,int[],Buffer\*); |  |
| void getStudentRecord(UserProfile [], Buffer \*, int , int &); |  |
| //bool getUserProfile(UserProfile[], Buffer \*, int &, int); |  |
| //student |  |
| int studentOperation(StudentResult [], Groups[], Course[], UserProfile[], Buffer \*, int, int); |  |
| //int viewGradeComponent(); |  |
| int listCourseGades(); |  |
| void studentCourseList(UserProfile[], StudentResult[], Course[], int , Buffer \*, int); |  |
| void gradeComponent(Course[], StudentResult[], Buffer \*, int, int); |  |
| //returns User ID and sets tmpData or return -1 |  |
| int searchUser(UserProfile[],Buffer \*, int &); |  |
| int searchUser(UserProfile[],Buffer \*, int&, char); |  |
| int searchUser(UserProfile[],Buffer \*, int&, string); |  |
| bool userPass(UserProfile[], Buffer \*, int); |  |
| string hiddenPassPhrase(string, char); |  |
| bool checkUp(int, string, UserProfile [], Buffer \*, int); |  |