Design Algorithm for Homework 2

The following table is a general description of the classes to be used in Homework 2:

|  |  |
| --- | --- |
| Class | Description |
| Student.cpp | This class represents the basic entities of students, containing accessor and mutator methods to obtain student data. It has a corresponding header file (Student.h). |
| StudentArray.cpp | This class represents the collection of dynamically allocated student pointers. Functions such as adding students, finding students, printing a list of all students, and printing student details given a list of ID’s are contained here. It has a corresponding header file (StudentArray.h). |
| Course.cpp | This class represents the basic entities of courses, containing accessor and mutator methods to obtain course data. It has a corresponding header file (Course.h). |
| CourseArray.cpp | This class represents the collection of dynamically allocated course pointers. Functions such as adding courses, finding courses, and printing a list of all courses are contained here. It has a corresponding header file (CourseArray.h). |
| Enrollment.cpp | This class represents the basic entities of enrollments, containing accessor and mutator methods to obtain enrollment data. It had a corresponding header file (Enrollment.h). |
| EnrollmentArray.cpp | This class represents the collection of dynamically allocated enrollment pointers. Functions such as adding enrollments, finding enrollments, printing a list of all students enrolled in a course, and calculating average course grade are contained here. It has a corresponding header file (EnrollmentArray.h). |

Course accessors and mutators:

string getLocation();

string getMeetingTime();

string getCourseID();

string getCourseName();

int getEnrolledStudentCount();

void setEnrolledStudentCount(int count);

void setCourseID(string id);

void setCourseName(string nameOfCourse);

void setLocation(string loc);

void setMeetingTime(string time);

Course Array Functions:

void saveToFile(ofstream& gradebookStream);

void loadFromFile(ifstream& gradebookStream);

void addACourse(Course\* course);

void printAllCourses();

Course\* findCourse(string courseID);

Enrollment accessors and mutators:

int setGrade(int grade);

int getEnrollmentID();

void setEnrollmentID(int enrollID);

string getStudentID();

void setStudentID();

string getCourseID();

void setCourseID();

Enrollment functions dealing with grades:

void saveAllGrades(ofstream& gradebookStream);

float averageGradeForAStudent();

char letterGradeForAStudent();

void printAllGradesForAStudentInACourse();

Enrollment Array functions:

bool enrollAStudent(string studentID, string courseID);

void addAnEnrollment(Enrollment \*enrollment);

string\*\* getAllStudentsInACourse(string courseID, int& studentCount);

float averageCourseGrade(string courseID);

Enrollment\* findEnrollment(string studentID, string courseID);

void saveToFile(ofstream& gradebookStream);

void loadFromFile(ifstream& gradebookStream);

Student accessors and mutators:

string getFirstName();

string getLastName();

int getEnrolledCourseCount();

void setEnrolledCourseCount(int count);

void setStudentName(string first, string last);

string getStudentID();

void setStudentID(string id);

Classification getStudentClassification();

string getStudentClassificationStr();

void setStudentClassification(Classification c);

Student Array functions:

void addAStudent(Student\* student);

string classificationToString(Classification c);

Classification intToClassification(int c);

void printAllStudents();

Student\* findStudent(string studentID);

void printStudentsInCourse(string\*\* studentIDs, int studentCount);

void saveToFile(ofstream& gradebookStream);

void loadFromFile(ifstream& gradebookStream);

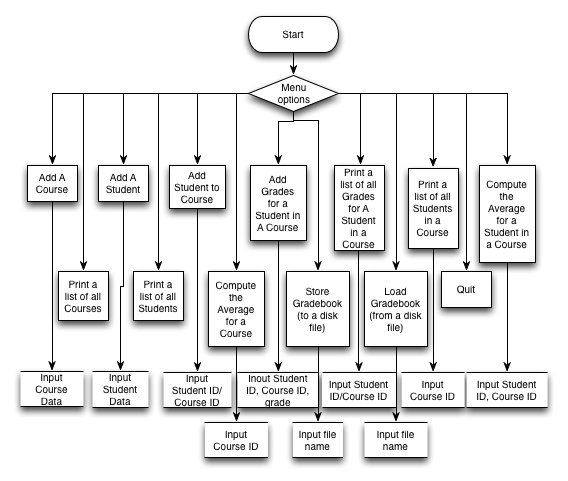
In general, it is clear that the array classes will be managing most of the actual data, while the classes themselves for students, courses, and enrollments contains more accessors and mutators.

The following are the test scenarios for Homework 2:

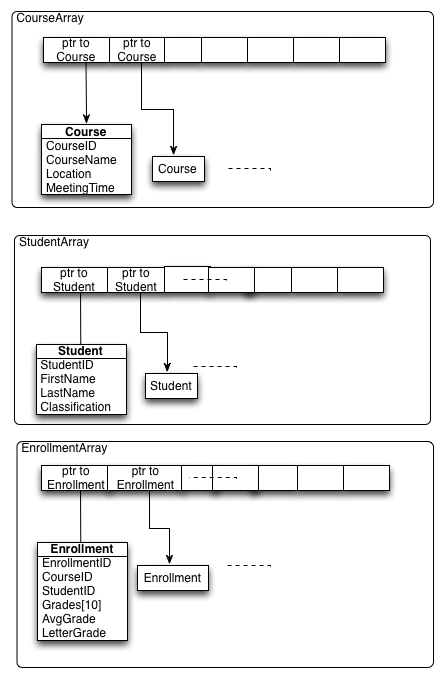
1. Repeated printing of enrolled students   
2. Load file, save it   
3. Print students, courses, enrollment   
4. Max 48 students per course   
5. Max 5 courses per student   
6. Max 10 grades per student in a course   
7. Load file, add students and courses, enroll student, save file, load again   
8. Average grade computation (options 7 and 10)   
9. Letter grade computation

10. Grow array scenarios - > 100 students, courses, enrollments

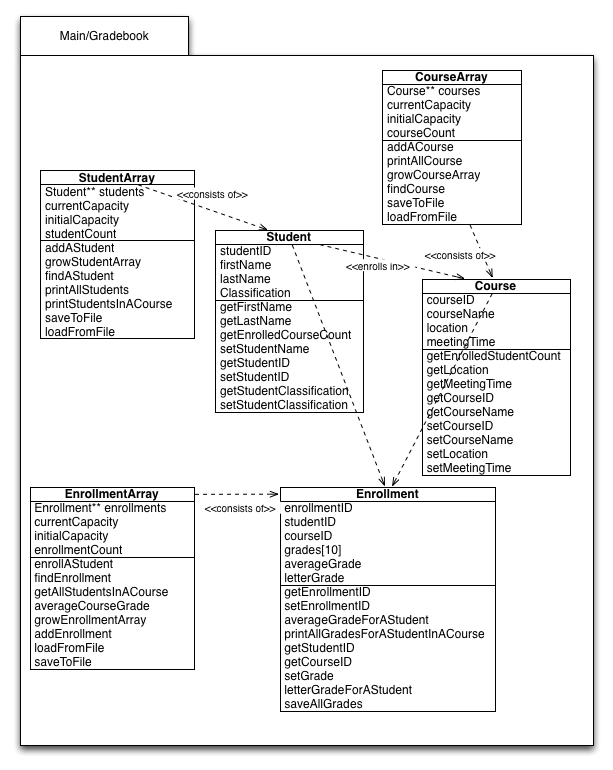
Each of the test scenarios were tested and resulted in a success.



The above flow chart describes the general flow process of the gradebook program.



This diagram represents the dynamic arrays that have been used for storing Student, Course, and Enrollment data.



The above diagram represents how the classes are dependent upon each other in the main program flow.