```
/*========================================================================
======================


========================================================================
====================*/

// Michael A. Amann 91646
// Assignment Number 7, CSE 100           MWF @9:40
// December 1, 2001           Lab F       Friday
// Description: This program allows a user to create a database of
// students who are objects that have grades, names and gradepoint
averages.

// User input: The user will input a choice and then a student's name
and
// grades.

// User output: The user will the see the student's name, grades and
grade
// point average depending on the output requested.

// Assumptions and Restrictions. The user must input what is requested,
// the program is not written for error checking type of input, only
// range.

// Student.h: interface for the Student class.
//
//////////////////////////////////////////////////////////////////////

#if !defined(AFX_STUDENT_H__C7CCE480_E517_11D5_B741_40F09FC10000
__INCLUDED_)
#define AFX_STUDENT_H__C7CCE480_E517_11D5_B741_40F09FC10000__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
#include <iostream>
#include <iomanip>
#include <string>
using namespace std;

class Student
{
public:
      void printGrades();
      string calculateGrade();
      double averageGrades();
      void addGrade(double grade);
      string getName();
      int getID();
      void setID(int identificationNumber);
      void addGrades(int numberOfGrades);
      double studentGrades[12];
      void setName(string name);
      void displayGrades();
      Student();
      Student(string,int);
      virtual ~Student();
```

```
private:
	int studentID;
	string name;
};

#endif // !defined(AFX_STUDENT_H__C7CCE480_E517_11D5_B741_40F09FC10000
__INCLUDED_)


// Student.cpp: implementation of the Student class.
//
//////////////////////////////////////////////////////////////////////

/*#include "Student.h"
#include <iostream>
#include <iomanip>
#define GRADEWIDTH = 5;*/

//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

// This function is the class constructor and initializes the data
// members of each instance of a class.
// Postconditions: The data members of each instance of the class
Student
// are initialized at the time of instantiation.
Student::Student()
{
	// Initialize data members.

	// index for all arrays in this function
	int i = 0;
	// The name of each student object
	this->name = "";
	// The ID number for each student object,
	this->studentID = -1;

	// Initialize the student's array of grades.
	for(;i<12;i++)
	//*NOTE* Each element is initialized to -1 because a
	// student could recieve a grade of 0.0.
	this->studentGrades[i]= -1.0;

}

// This overloaded contructor allows the user to predefine
// a student's name and Id number.
// Postconditions: The datamembers of each student object
// are initialized to the values passed in by the caller.
Student::Student(string newName,int Id)
{
	// The student object's name.
	this->name = newName;

	// The student object's Id number.
	this->studentID = Id;
}

// The class destructor is not used in this program.
```

```
Student::~Student()
{

}

// This functions allows the caller to establish the name of a student.
// Preconditions: There must be an instantiated instance of class
student.
// Postconditions: The name of the student object is updated to that
which
// is passed in by the caller.
void Student::setName(string /*OUT*/studentName)
{
      // The student object's name.
      this->name = studentName;
}

// This function allows the caller to add a prespecified number of
// grades to a student's array of grades.
// Preconditions: There must be an instantiated object of type student
// in existence.
// Postconditions: Based on the number of grades requested, the user
// is allowed to place a new grade in each successive portion of the
// student's array of grade. Provided that element has been properly
// initialized to -1.
void Student::addGrades(int/*IN*/numberOfGrades)
{
      // Declare and initialize local variables
      // The index for all arrays in this function.
      int i = 0;

      // While there are still grades to add, check the next element of
      // the array, if its been initialized, allow the caller to add a
      // grade there.
      for(numberOfGrades; numberOfGrades > 0;numberOfGrades --)
      {
            // Prompt the user for a grade...
            cout <<"Please enter a grade in decimal form (0 - 100): ";

            // If the current element has been initialized allow the
caller
            // to add a grade.
            if(this->studentGrades[i]== -1.0)
            {
                  // Get the grade from the caller and increment the
                  // counter i;
                  cin >> this->studentGrades[i] ;

                  //*ERROR CHECK* If the grade entered by the caller is
out
                  // of range, print an error message and let the caller
re-
                  // enter the grade.
                  while(this->studentGrades[i] <0 ||
                        this->studentGrades[i] > 100)
                  {
                        cout << "**Invalid input! grade must be 0
through 100**";
                        cin >> this->studentGrades[i];
```

```
                }
                i++;
            }

      }
}

// This function allows the caller to set the student's ID number.
// Preconditions: There must be an instantiated object of type student
// in existence.
// Postconditions: The student object's ID number is updated to that
which
// is passed in by the caller.
void Student::setID(int/*OUT*/identificationNumber)
{
      // The student's ID number.
      this->studentID = identificationNumber;
}

// This function allows the caller to retrieve a student's ID number.
// Preconditions: There must be an instantiated object of type student
// in existence.
// Postconditions: The student object's ID number is returned to the
// caller.
int Student::getID()
{
      return this->studentID;
}

// This function allows the caller to retrieve the name of a student.
// Preconditions: There must be an instantiated object of type student
// in existence.
// Postconditions: The student object's name is returned to the caller.
string Student::getName()
{
      return this->name;
}

// This function allows the caller to add a single grade to the
// next available index of a student's array of grades.
// Preconditions: There must be an instantiated object of type student
// in existence.
// Postconditions: A new grade is added to the student's array of
// grades at the next available index.
void Student::addGrade(double grade)
{
      // Declare and initialize local variables.
      // The index for all arrays in this function.
      int i = 0;

      // Prompt the user for a grade...
      cout<<"Please enter a new grade in decimal form for "
            <<this->getName() <<"\n";

      // Check to see if the first location in the array has been
      // initialized. If so allow the user to add a grade there.
      if(this->studentGrades[i]== -1.0)
      {
            cin >> this->studentGrades[i];
```

```
            //*ERROR CHECK* If the grade entered by the user is out of
            // range, print and error message and let the user re-enter
            // the grade.
            while(this->studentGrades[i] <0 ||
                  this->studentGrades[i] > 100)
            {
                  cout << "**Invalid input! grade must be 0 through 100
**";
                  cin >> this->studentGrades[i];

            }

      }

      // If the first location in the array contains a grade, traverse
      // the array until you find the index of the next avaialable
      // location. Then allow the user to enter a grade there.
      else if(this->studentGrades[i] >= 1 && i+1 <12 )
      {
            for(i;i<12,this->studentGrades[i] >= 1;i++)
            {
            }
            cin >> this->studentGrades[i];

            //*ERROR CHECK* If the grade entered by the user is out of
            // range, print and error message and let the user re-enter
            // the grade.
            while(this->studentGrades[i] <0 ||
                  this->studentGrades[i] > 100)
            {
                  cout << "**Invalid input! grade must be 0 through 100
**";
                  cin >> this->studentGrades[i];

            }
      }

}

// This function allows a student object to display all grades
// in a pre-specified formatt.
// Preconditions: There must be an instantiated object of type student
// in existence.
// Postconditions: The student's array of grades is traversed and each
// grade is translated into a letter grade and printed to the screen.
void Student::displayGrades()
{
      // Declare and initialize local variables.
      // The index for all arrays in this function.
      int i = 0;

      // Display the name of the calling student.
      cout<<this->name <<"\n";

      // Traverse the student's array of grades, calculate a
      // letter grade and display it on the screen.
      for(;i<12,this->studentGrades[i]!= -1.0;i++)
      {
            if(this->studentGrades[i]<=100 && this->studentGrades[i]>89)
                  cout <<"A" <<"  ";
```

```
            else if(this->studentGrades[i]<=89 && this->studentGrades
[i]>79)
                  cout <<"B" <<"  ";
            else if(this->studentGrades[i]<=79)
                  cout <<"C" <<"  ";
      }

}

// This function allows a student object to calculate an average of
// all of the students grades.
// Preconditions: There must be an instantiated object of type student
// in existence.
// Postconditions: The students' grades are added and the result is
// divided by the number of grades to obtain an average. That average is
// returned to the caller.
double Student::averageGrades()
{
      // Declare and initialize local variables.

      // The index for all arrays in this function.
      int i = 0;
      // The grade point average for a student.
      double gpa = 0.0;
      // The number of grades in a student's array of grades.
      double numberOfGrades = 0.0;

      // Traverse the student's array of grades. If the location
      // is a valid grade value, add it to gpa and increment the
      // number of grades counter.
      for(;i<12,this->studentGrades[i]!= -1.0;i++)
      {
            gpa += this->studentGrades[i];
                  numberOfGrades++;
      }

      // Determine the grade point average by dividing
      // the sum of all the grades by the number of grades.
      // Return the average to the caller.
      gpa = gpa/numberOfGrades;

      return gpa;
}

// This function takes the same procedure from averageGrades but uses
// the result to calculate a letter grade based upon the average of the
// student's grades.
// Preconditions: There must be an instantiated object of type student
// in existence.
// Postconditions: After calculating the grade, a letter grade is
assigned
// to the average and returned to the caller.
string Student::calculateGrade()
{
      // Declare and initialize local variables.

      // The index of all arrays in this function.
      int i = 0;
      // The number of grades a student has in their array of grades.
      double  numberOfGrades = 0.0;
```

```
	// The grade point average of a student.
	double gpa = 0.0;
	// The grade calculated from the average grade.
	string grade = "";

	// Traverse the student's array of grades and sum them up.
	// Then divide the result by the number of grades to find an
	// average.
	for(;i<12,this->studentGrades[i]!= -1.0;i++)
	{
		gpa += this->studentGrades[i];
			numberOfGrades++;
	}

	gpa = gpa/numberOfGrades;

	// Calculate a letter grade based upon the average
	// and return it to the caller.
	if(gpa<=100 && gpa>89)
		grade = "A";
	else if(gpa<=89 && gpa>70)
		grade = "B";
	else if(gpa<=79)
		grade = "C";

	return grade;
}

//*This method was used for debugging purposed only*
void Student::printGrades()
{
	// The index of every array in this function.
	int i = 0;

	for(;i<12;i++)
		cout << this->studentGrades[i];
}


/*#include <iostream>
#include <iomanip>
#include <string>
#include "Student.h"

using namespace std;*/

// Function prototypes...
void doSelection(char,Student[40],int&);
void getEntry(char &);
void displayMenu();
int getNumberOfGrades();
void addGrade(Student [40]);
void studentInfo(int &,Student[40]);
void displayStudent(Student [40]);
void displayAll(Student[40],int &);

// Begin program here...
void main (void)
{
//    _____________________________
```

```
      // Delcare and intialize local variables.

    char choice = '\0';
    int index = 0;

      // Declare an array of type Student to store
      // student objects.
      Student myclass[40];

    do
    {
       system("CLS");                          // Clear the screen
       displayMenu();
       getEntry(choice);
      if(choice != 'Q'){                       // If user does not want
to quit
       doSelection(choice,myclass,index);
       system("PAUSE");                        // Pause system for
report displays
                                                          // "Press
any key to continue ..."
      }

    }while(choice != 'Q');

    cout << endl << endl << endl;  // For formatting
}

void doSelection(char /*IN*/action, Student /*INOUT*/myclass[40],
                         int /*INOUT*/&index)
{

      // Call a function that corresponds with the user's choice...
      if(action == 'A')

            // Add a student and his/her grades to the organizer
            studentInfo(index,myclass);

      else if(action == 'B')

            // Add a grade to an exisiting student.
             addGrade(myclass);


      else if (action == 'C')

            // Display a particular student and all their information
            // based on a predefined format.
            displayStudent(myclass);



      else if (action == 'D')

            // Display all the students and all their information
            // based on a predefined format.
            displayAll(myclass,index);
```

```
      else if (action == 'Q')
      {
            // Quit the program. This functionality is handled
            // in main.
      }

}

// This function gets the user's choice for program execution
// and converts it to uppercase.
void getEntry(char /*INOUT*/&selection)
{
      //char selection;
    cin >> selection;
    selection = toupper(selection);

    //Error check menu selection
    while(selection != 'A' && selection != 'B' && selection != 'C' &&
        selection != 'D' && selection != 'Q')
    {
       cout << "\nError Selection must be A, B, C, or D. "
            << "Please enter a selection: ";
       cin >> selection;
         selection = toupper(selection);

    }


}

// This function formats and puts the user menu onto the screen
// to guide the user when executing the program.
void displayMenu()
{

    cout << "\n\n\n\n\n\n\n" << setw(44) << "The Student Organizer\n"
            << setw(48) << "------------------------------\n"
            << setw(49) << "A)  Enter Student Information\n\n"
            << setw(58) << "B)  Add a Grade to an Existing Student\n\n"
            << setw(58) << "C)  Display a Student's Grades and Gpa\n\n"
            << setw(63) << "D)  Sort and Display all Student's by Grade
\n\n"
            << setw(36) << "Q)  Quit Program\n\n"
            << setw(49) << "------------------------------\n\n";

    cout << setw(50) << "Please enter your selection ---> ";
}

// This function prompts the user to enter a number of grades for
// a particular student.
// Precondtions: There must be a student object in existence.
// Postconditions: The function takes the user's request and checks
// to see if it is within a permissible range (1-12) then either
// prompts the user for a new grade number if its out of range, or
// accepts the number.
int getNumberOfGrades()
{
      // Declare and initialzie local variables.

      // The number of grades a user wishes to enter.
```

```
	int gradeNumber = 0;

	// Prompt the user for a number of grades to enter.
	cout <<"How many grades do you want to enter? (1 - 12)\n";
	cin >> gradeNumber;

	//*ERROR CHECK* Make sure the requted number of grades is within
	// the acceptable parameters. Keep loopin until a usable number
	// is entered.
	while(gradeNumber > 12 || gradeNumber < 1)
	{
		cout <<"**Invalid input! Must be 1 through 12 grades**\n";
		cout <<"How many grades do you want to enter? (1 - 12)";
		cin >> gradeNumber;
	}

	return gradeNumber;
}


// This function allows a user to add one grade to an existing student's
// array of grades.
// Preconditions: There must be a student object in existence whose name
// matches that requested by the user.
// Postconditions:
// (1) The array of students is searched for the name requested
// by the user.
// (2) If there is a match the user is allowed to enter a grade
// for that student. If not, and error message is printed to the screen
// and the function ends.
void addGrade(Student /*IN*/myclass[40])
{
	// Declare and initialize local variables.

	// The index of every array in this function.
	int i = 0;

	// The new grade added by the user.
	double newGrade = 0.0;

	// The name entered by the user.
	string newName = "";

	// Prompt the user to enter a student's name.
	cout <<"Enter the student's name you wish to find. ";
	cin.ignore();
	getline(cin,newName);

	// Traverse the array of students until a match is found.
	for(;i<40 && myclass[i].getName()!=newName;i++)
	{
	}
	if(i ==40)
		--i;
	// If a match is found allow the user to enter a grade.
	if(myclass[i].getName()==newName)
		myclass[i].addGrade(newGrade);

	// If not match is found, print an erro message and end the
	// function.
```

```
	else
		cout <<"*Entered name cannot be found*\n";
}

// This function allows the caller to create a new student object
// and enter grades for that student.
// Postconditions:
// (1) A new student is created.
// (2) The user is allowed to enter grades for that student.
// (3) The student is added to the Student Organizer.
void studentInfo(int /*IN*/&index, Student /*IN*/myclass[40])
{
	// Declare and intialize local varibles.

	// The new studen object to be addes to the organizer.
	Student newStudent;

	// The student's name.
	string newName = "";

	// The number of grades the user wishes to enter for the student.
	int numberOfGrades = 0;

	// Prompt the user for a name for this student.
	cout << "Please enter the student's name: ";
	cin.ignore();
	getline(cin,newName);

	// Initialize the student with the new name.
	newStudent.setName(newName);

	// Give the student and ID number.
	newStudent.setID(index);
	index++;

	// Prompt the user for the number of grades to be entered.
	numberOfGrades = getNumberOfGrades();

	// Allow the user to enter that number of grades.
	newStudent.addGrades(numberOfGrades);

	// Add the new student to the organizer.
	myclass[index] = newStudent;

}

// This function displays the name, grades and grade point average of
// the student requested by the user.
// Preconditions: There must be a student in the organizer whose name
// matches that requested by the user.
// Postconditions:
// (1) The Student Organizer is searched for the student requested.
// (2) If a match is found that student's information is displayed.
// (3) If not match is found the function ends and returns to the
// main screen.
void displayStudent(Student /*IN*/myclass[40])
{
	// Declare and initialize local varibles.

	// The index of every array in this function.
```

```
	int i = 0;

	// The name of the student the user is looking for.
	string studentName = "";

	// Prompt the user to identify which studen they want to display
	// information for.
	cout<<"Enter the student's name whose grades you want to display:
";
	cin.ignore();
	getline(cin,studentName);

	// Search the organizer for the student. If a match is found,
	// display the information for that student. If no match is
	// found, print and error message and end the function.
	for(;i<40 && myclass[i].getName()!=studentName;i++)
	{
	}
	if(i ==40)
		--i;
	if(myclass[i].getName()==studentName)
	{
		cout << fixed;
		myclass[i].displayGrades();
		cout <<" " <<"Ending Grade: " <<myclass[i].calculateGrade();
		cout <<"\t" << "GPA: ";
		cout.precision(2);
		cout <<myclass[i].averageGrades() <<"\n\n";
	}
	else
		cout <<"*Entered name cannot be found*\n";
}

// This function displays the names, ending grades and gradepoint
// averages for all the students in the organizer.
// Precondtions: There must be at least one student in the organizer.
// Postconditions: The organizer is searched and all students found
// within are displayed on the screen ordered by grade.
void displayAll(Student /*IN*/myclass[40], int /*IN*/&index)
{
	for(int i = 0;i<40,i<=index;i++)
	{
		if (myclass[i].getID()!=-1 && myclass[i].calculateGrade() ==
"A")
		{
		cout << fixed;
		cout << myclass[i].getName() <<"\n";
		cout <<"Ending Grade: " <<myclass[i].calculateGrade();
		cout <<"\t    " << "GPA: ";
		cout.precision(2);
		cout <<myclass[i].averageGrades() <<"\n\n";
		}
	}

	for(i = 0;i<40,i<=index;i++)
	{
		if (myclass[i].getID()!=-1 && myclass[i].calculateGrade() ==
"B")
		{
		cout << fixed;
```

```
            cout << myclass[i].getName() <<"\n";
            cout <<"Ending Grade: " <<myclass[i].calculateGrade();
            cout <<"\t    " << "GPA: ";
            cout.precision(2);
            cout <<myclass[i].averageGrades() <<"\n\n";
            }
      }

      for(i = 0;i<40,i<=index;i++)
      {
            if (myclass[i].getID()!=-1 && myclass[i].calculateGrade() ==
"C")
            {
            cout << fixed;
            cout << myclass[i].getName() <<"\n";
            cout <<"Ending Grade: " <<myclass[i].calculateGrade();
            cout <<"\t    " << "GPA: ";
            cout.precision(2);
            cout <<myclass[i].averageGrades() <<"\n\n";
            }
      }
}
```