Transfer Application Program

Christian Conable

CSC 17C

November 16, 2014

48596

**Introduction**

This program demonstrates the use of classes and the STL library. The user enters information relating to their identity, address, and their current academic record. The student will be directed to reenter any input that does not meet the required specifications. The information is displayed to the screen, and the student is asked whether the information is correct. If so, the program ends, but if the information isn’t correct, the student will have to reenter his or her information from the beginning.

**Description**

Project Size: 753 lines

Number of variables: 54

Number of Methods: 23

Methods used: Iostream, for loop, while loop, do-while loop, classes, inheritance, if-else, arrays, pointers, dynamically allocated arrays, functions, Boolean statements, stacks, queues, maps, sets, algorithms, iterators, constructors, destructors, accessor and mutator functions, operator overloading,

Stack: Used to display the student’s first name, last name, state, city and address.

Queue: Used to display the SAT scores in the order they were entered. Defined in Student header file

Set: Used to display the SAT scores in ascending order form smallest to largest. Defined in Student header file

Vector: Used to enter and display GPA values for each quarter

Iterator: Used to iterate through the set variable to display the SAT scores.

Algorithm and Numeric: Used to accumulate GPA values for all quarters.

Map: Used to display the letter grade for each class from the previous quarter.

The first segment of the program is straightforward because it requests basic information from the student. The user enters their first and last names as string variables, which are stored in the mutator functions defined in the Student header file. Similarly, the student enters his or her age and date of birth as integer variables, and they are then stored in the Date header file as mutator and accessor functions. The student’s address, city, state, zip code, social security number, and high school graduation date are stored in the remaining Student and basicInfo header files.

In the second segment of the program, the student enters information regarding their academic history as well as their tentative plans for transfer. They are asked to enter the number of times they took the SAT’s, and the program then redirects to the Student implementation file to validate their tests scores. Set and queue variables are utilized to store the input, in which the queue operator directly displays the information. The set operator displays the same information in ascending order from the worst to the best test scores. The student’s average test score is then shown to the screen.

The next part of the programs asks the student the grade level they will be applying for. Their input will then determine the number of years they have attended their local community college, which will contribute to the number of quarters to enter their respective grade point averages. Next, the student must enter their GPA per quarter in order to determine a cumulative average. The final result shows a table with the GPA’s they entered for each quarter, and their total GPA is shown at the bottom of the table. Their entries must be in accordance within calculations standards of 1.0 to 4.0, and any attempts to enter values beyond this range will result in an error message. This part of the program utilizes the vector and algorithm libraries to determine the average GPA for the student. The student must then enter the number of classes they enrolled in for their final quarter before transfer, along with the grades they received in those classes. This part of the program utilizes the map library to display the letter grade associated with the integer entries. All the information is then displayed to the screen, and the student is given the option to repeat the program.

**Summary**

Although it was difficult to incorporate many useful functions from the STL library, this project was fairly straightforward. During the initial stages of writing the program, I struggled to properly format the variables of the map and set libraries. I would either misuse the variable types, such as switching an integer with a float, or I would declare fewer variables within the angled brackets for the map variable. Another issue I had regarded incorporating the arrays with the variables from the STL library. Similar to my first problem, I frequently mismatched variables within the arrays, which caused the program to either crash or display unanticipated results.

**Variables**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Type: | Name: | File location: | Line location: | Description: |
| String | First\_n | BasicInfo | 14 | Variable used to store student’s first name |
|  | Last\_n | BasicInfo | 15 | Variable used to store student’s last name |
|  | Month\_string | Date | 16 | Outputs the total graduation date |
|  | State | Student | 26 | State of residency |
|  | Social\_number | Student | 27 | For accessor and mutator functions ssn. |
|  | City | Student | 28 | City where student is from |
|  | First\_n | Student | 30 | Student’s first name for accessor and mutator functions. |
|  | Last\_n | Student | 31 | Student’s last name for accessor and mutator functions. |
|  | First | Main | 27 | Enter student’s first name |
|  | Last | Main | 27 | Enter student’s last name |
|  | State | Main | 27 | Enter state |
|  | Address | Main | 28 | Enter address |
|  | City | Main | 28 | Enter city |
|  | Social\_security\_number | Main | 29 | Enter ssn |
| Int | Age | BasicInfo | 13 | Accessor and mutator functions for student’s age |
|  | Month | Date | 17 | Accessor and mutator for grad month |
|  | Day | Date | 18 | Day of grad |
|  | Year | Date | 19 | Year of grad |
|  | Graduate | Date | 20 | Display all grad dates |
|  | Num | Student | 25 | Pointer array to hold scores |
|  | N | Student | 25 | Size of array that holds scores |
|  | Zip | Student | 33 | Student’s zip code |
|  | Classes | Main | 33 | Number of classes |
|  | Value | Main | 33 | Value for each class grade |
|  | Zip | Main | 34 | Enter zip code |
|  | Grad\_year | Main | 34 | Enter grad year |
|  | Age | Main | 35 | Enter age |
|  | Times | Main | 35 | Number of times taken sat’s |
|  | Hs\_month | Main | 36 | Month of graduation |
|  | Hs\_day | Main | 36 | day of graduation |
|  | Hs\_year | Main | 36 | Year of graduation |
|  | Last\_years | Main | 39 | Number of terms for gpa |
|  | This\_years | Main | 40 | Total gpa inputs |
|  | Col | Main | 41 | Used for double array |
|  | Array | Main | 49 | Dynamically allocated array |
|  | Sctimes | Main | 128 | Times sat was taken |
|  | Dgvalue | Main | 203 | Pointer array for letter grades |
| Char | Gender | BasicInfo | 11 | Accessor and mutator variable for student’s gender |
|  | Standing | BasicInfo | 16 | Accessor and mutator variable for student’s the grade level |
|  | Gender | Main | 37 | Enter m for male f for female |
|  | Is\_correct | Main | 37 | Determine whether the displayed info is correct |
|  | Stand | Main | 37 | The grade level you are applying for |
| Float | Average | Main | 144 | Compute average gpa for that year |
|  | Score | Main | 145 | Hold score for each quarter |
|  | St\_ave | Main | 146 | Computer quarter average |
|  | Total | Main | 147 | Total all the gpa’s |
|  | Sum | Main | 147 | Sum the gpa’s |
| Map<> | Grade | Main | 31 | Hold the letter grade |
| Vector<> | V | Main | 38 | Hold the gpa’s |
| Set<> | Major | Student | 21 | Use various functions for sat scores. |
| Queue<> | Major2 | Student | 22 | Use various functions to display sat scores. |
| Typedef | It | Student | 23 | Iterator used to pass through sat scores |

**Pseudocode**

*Display Introduction with basic instructions*

*Do*

*Enter student’s first name*

*While first name is not within set boundaries*

*Do*

*Enter student’s last name*

*While last name is not within set boundaries*

*Do*

*Enter student’s age*

*While age is not within limits of 16-100*

*Do*

*Enter student’s birthday*

*While entry is not within specified format*

*Do*

*Enter gender*

*While entry is not specified characters*

*Do*

*Enter street address*

*While address is not within boundaries*

*Do*

*Enter city*

*While characters do not meet specifications*

*Do*

*Enter state*

*While characters do not meet specifications*

*Do*

*Enter zip code*

*While entries are not numbers and are beyond specified boundaries*

*Do*

*Enter ssn*

*While entries are not within specified format*

*Do*

*Enter high school graduation date*

*While date is not within specified boundaries*

*Enter number of times taken sat’s*

*If entry is not within range*

*Reenter number of times*

*While entry is still not within range*

*Reenter number of times*

*While i is less than number of times*

*Enter the test scores for (i + 1) time*

*Display scores in regular order*

*Display scores in ascending order*

*Enter grade level user will be transferring as*

*If transferring as a freshman*

*Only four entries will be needed for gpa’s*

*Else*

*Eight entries will be needed for gpa’s*

*Enter gpa’s for each quarter*

*Display gpa’s for each quarter*

*Display cumulative gpa*

*Enter number of classes user took this past quarter*

*If classes are beyond specified range*

*Input number of classes*

*While classes are still beyond specified range*

*Input number of classes*

*Input score for each class*

*Display all information*

*Ask user if all information is correct*

*If correct*

*End program*

*Else*

*Begin program again*

***Code***

#include <cstdlib>

#include <iostream>

#include <iomanip>

#include <fstream>

#include <map>

#include <stack>

#include <vector>

#include <numeric>

#include <algorithm>

#include "Student.h"

#include "BasicInfo.h"

#include "Date.h"

using namespace std;

//Declare Functions

bool valid\_social(string social);

bool has\_symbols(string data);

bool data\_valid(int lower\_range, int upper\_range, string data\_name, int data);

void display\_social(string social\_security\_number);

void display\_social(string social\_security\_number, int dummy);

int main(int argc, char \*argv[]){

//string variables

string first, last, state;

string address, city;

string social\_security\_number;

//Integer, character and float variables

map<int, string> grade;

stack<string> stackinfo;

int classes, value;

int zip, grad\_year;

int age, times;

int hs\_month, hs\_day, hs\_year;

char gender, is\_correct, stand;

vector<float> v;

int last\_years;

const int this\_year = 4;

const int col = 4;

//Class Declaration variables

Student student;

BasicInfo set;

BasicInfo size, employed;

Date hs\_graduation, date\_of\_birth, set2;

int \*array;

cout.setf(ios::fixed);

cout.setf(ios::showpoint);

cout.precision(3);

do{

cout << endl;

cout << setw(60) << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << setw(60) << "\*\*\*\*\*\*\*\* College Transfer Application \*\*\*\*\*\*\*\*\*" << endl;

cout << setw(60) << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\tIn this program, a transferring college student will enter"

<< "\ninformation for the application, and the program will then"

<< "\ndisplay the results.\n\n" << endl;

cout << setw(51) << "STUDENT PERSONAL INFORMATION: " << endl;

do{

//Enter Student's First name

cout << "Student First Name: ";

cin >> first;

}while(has\_symbols(first) == true);

student.setFirst(first);

do{

//Enter Student's last name

cout << "Student Last Name: ";

cin >> last;

}while(has\_symbols(first) == true);

student.setLast(last);

do{

//Enter Student's last name

cout << "Student's age: ";

cin >> age;

}while(data\_valid(16, 100, "age", age) == false);

set2.setAge(age);

do{

//Enter student's birthday

cout << "Student's birthday, ex 06 06 1985: ";

cin >> date\_of\_birth.month >> date\_of\_birth.day >> date\_of\_birth.year;

}while(Date::date\_valid(date\_of\_birth) == false);

do{

//Enter Student's gender

cout << "Student's gender, M / F: ";

cin >> gender;

}while(!(gender == 'm' || gender == 'f' || gender == 'M' || gender == 'F'));

do{

//Enter Student's Street Address

cout << "Street Address: ";

cin.ignore();

getline(cin, address);

}while(has\_symbols(address) == true);

do{

//Enter City

cout << "City: ";

getline(cin, city);

}while(has\_symbols(city) == true);

student.setCity(city);

do{

//Enter State

cout << "State: ";

cin >> state;

}while(has\_symbols(state) == true);

student.setState(state);

do{

//Enter Zip Code

cout << "Zip Code: ";

cin >> zip;

}while(data\_valid(00000, 99999, "zip code", zip) == false);

student.setZip(zip);

do{

//Enter Social Security Number

cout << "Your social security number, 123-45-6789: ";

cin >> social\_security\_number;

}while(valid\_social(social\_security\_number) == false);

do{

//Enter Date of graduation from high school

cout << "Date of high school graduation, ex 06 06 2000: ";

cin >> hs\_graduation.month >> hs\_graduation.day >> hs\_graduation.year;

}while(Date::date\_valid(hs\_graduation) == false);

cout << "How many times did you take the SAT? ";

cin >> times;

if(time < 0 || times > 5){

cout << "\nInvalid entries." << endl;

cout << "How many times did you take the SAT? ";

cin >> times;

while(time < 0 || times > 5){

cout << "How many times did you take the SAT? ";

cin >> times;

}

}

int \*sctimes = new int[times]; // <-- SET

student.Score(sctimes, times);

cout << endl;

//Grade Level Standing you choose to transfer as

cout << "Choose the grade level standing "

<< "you will be applying for: " << endl << endl;

cout << setw(25) << "a. Freshman " << endl;

cout << setw(26) << "b Sophomore " << endl;

cout << setw(23) << "c. Junior " << endl;

cout << setw(23) << "d. Senior " << endl;

cin >> stand;

if(stand == 'a' || stand == 'A')

last\_years = 1;

else if(stand == 'b' || stand == 'B' )

last\_years = 2;

else

last\_years = 2;

float average;

float score[last\_years][col];

float st\_ave[last\_years], total, sum = 0;

set.setStanding(stand);

cout << endl;

for (int i = 0; i <= last\_years - 1; i++){

total = 0;

for(int j = 0; j < col; j++){

do{

cout << endl << "Average GPA for " << (i + 1) << " year, "

<< j + 1 << " quarter: ";

cin >> score[i][j];

}while(score[i][j] < 0.0e0f || score[i][j] > 4.0e0f); //Score must be between 0 and 4.0

v.push\_back(score[i][j]);

total+= score[i][j];

}

st\_ave[i] = total/col;

cout << endl << endl << setw(45) << "Average GPA for " << i + 1 << " year: " << endl;

cout << setw(53) << st\_ave[i] << "." << endl;

}

//Calculate addition of all GPA's.

float total1 = 0.0e0f;

for (int i = 0; i < last\_years; i++){

for (int j = 0; j < col; j++)

total1 += score[i][j];}

//Compute and display the cumulative average

float average1;

average1 = total1/(last\_years\*col);

//Compute average GPA.

float product;

product = average1\*average;

//Table showing grade point averages for each quarter

cout << "Year" << setw(23) << "GPA" << endl;

for (int i = 0; i < last\_years; i++){

cout << i + 1 << " ";

for (int j = 0; j < col; j++)

cout << setw(9) << score[i][j];

cout << endl << endl;

}

sum = accumulate(v.begin(), v.end(), 0.0);

float final;

final = sum/(last\_years\*4);

cout << "Enter the number of classes you took this\n"

<< "past quarter: "; // <-- MAP

cin >> classes;

if(classes <= 0 || classes > 6){

cout << "Invalid entry." << endl;

cout << "Enter the number of classes you took this\n"

<< "past quarter: ";

cin >> classes;

while (classes <= 0 || classes > 6){

cout << "Invalid entry." << endl;

cout << "Enter the number of classes you took this\n"

<< "past quarter: ";

cin >> classes;

}

}

int \*gdvalue = new int[classes];

for (int i = 0; i < classes; i++){

cout << "Enter " << i + 1 << " grade: ";

cin >> gdvalue[i];

if(gdvalue[i] >= 90)

grade[gdvalue[i]] = "A";

else if(gdvalue[i] < 90 && gdvalue[i] >= 80)

grade[gdvalue[i]] = "B";

else if(gdvalue[i] < 80 && gdvalue[i] >= 70)

grade[gdvalue[i]] = "C";

else if(gdvalue[i] < 70 && gdvalue[i] >= 60)

grade[gdvalue[i]] = "D";

else

grade[gdvalue[i]] = "F";

}

cout << endl;

//Display all the information

stackinfo.push(state);

stackinfo.push(city);

stackinfo.push(address);

stackinfo.push(last);

stackinfo.push(first);

cout << endl;

cout << "\n\n\t\*\*\*\*\*\*Summary Sheet\*\*\*\*\*\n";

while(!stackinfo.empty()){

cout << "\t" << stackinfo.top() << endl;

stackinfo.pop();}

cout << "\t" << student.getZip()<< endl;

cout << "\tAge: " << set2.getAge() << endl;

cout << "\tDate of Birth: ";

Date::display\_date(date\_of\_birth);

cout << "\n\tSex: ";

if(gender == 'M' || gender == 'm')

cout << "\tMale";

else

cout << "\tFemale";

cout << "\n\tSSN: ";

display\_social(social\_security\_number, 0);

cout << endl;

cout << "\tHigh School Graduation date: ";

Date::display\_date(hs\_graduation);

cout << endl;

if (stand == 'a' || stand == 'A')

cout << "\tFreshman Transfer" << endl;

else if (stand == 'b' || stand == 'B')

cout << "\tSophomore Transfer." << endl;

else if (stand == 'c' || stand == 'C')

cout << "\tJunior Transfer" << endl;

else

cout << "\tSenior Transfer " << endl;

cout << "\tGrades you received for last quarter: " << endl;

for(int i = 0; i < classes; i++){

cout << "\tGrade " << i + 1 << ": "

<< grade[gdvalue[i]] << endl;

}

cout << "\tCumulative GPA: " << final << endl;

cout << "\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*";

cout << "\nIs the above information correct, y/n: ";

cin >> is\_correct;

if(is\_correct == 'Y' || is\_correct == 'y')

return 0;

}while(is\_correct == 'N' || is\_correct == 'n');

Student new\_student = Student(first, last, gender, age, date\_of\_birth, state, city, zip);

cin.ignore();

cin.get();

system("PAUSE");

return EXIT\_SUCCESS;}

//Validates that the zip code will be five digits long

bool data\_valid(int lower\_range, int upper\_range, string data\_name, int data){

if((data >= lower\_range) && (data <= upper\_range))

return true;

else{

cout << "The " << data\_name << " is not valid";

cout << endl;

return false;}}

//String variables must follow specified format in order to run

bool has\_symbols(string data){

for(int i = 0; i < data.length(); i++){

if(isalnum(data[i]) == false && isspace(data[i]) == false){

cout << "The input can only have letters";

cout << "The illegal character: [" << data[i] << "]";

return true;}

}

return false;}

//Validate the format for the social security number

bool valid\_social(string social){

if(social.length() != 11){

cout << "Social not in right format";

return false;}

return true;}

//Display the Social Security Number

void display\_social(string social\_security\_number){

cout << "\*\*\*-\*\*-";

for(int i = 7; i < 11; i++)

cout << social\_security\_number[i];}

void display\_social(string social\_security\_number, int dummy){

for(int i =0; i < 11; i++)

cout << social\_security\_number[i];}