//Kevin Zhu

//CIS 3100 Spring 2023

#include <iostream>

#include <fstream>

#include <string>

#include <iomanip>

using namespace std;

enum Grade {F, D, CMINUS, C, CPLUS, BMINUS, B, BPLUS, AMINUS, A};

struct StudentData

{

string lastName;

string firstName;

double score;

double percentage;

Grade pointGrade;

Grade curveGrade;

Grade finalGrade;

};

bool insert(StudentData\*\* v, int& b, int& s, string rawstring);

void outputGradeBook(string ofileName, StudentData s[], int size);

bool getwords(string\* array, string rawstring);

Grade scoretoPointGrade(double score);

string letterGrade(Grade g);

const int CLASSSIZE = 35; //the maximum number of students in a single section class

const int BAR = 60; //student needs to pass the minimum bar to enjoy the curve grade

int main()

{

StudentData\* student = new StudentData[CLASSSIZE];

int capacity = CLASSSIZE;

int b = 0;

string rawstring;

ifstream scores("./spring2023.txt");

string grades = "Assign5\_Zhu\_Kevin\_output.txt";

cout << "Initial Class size = " << CLASSSIZE << " students." << endl;

if (scores.is\_open())

{

while (getline(scores, rawstring))

{

insert(&student, b, capacity, rawstring);

}

scores.close();

}

outputGradeBook(grades, student, b);

delete[] student;

return 0;

}

bool insert(StudentData\*\* v, int& b, int& s, string rawstring)

{

if (b == s)

{

StudentData\* original = \*v; //original is a pointer to a Struct array

StudentData\* student = new StudentData[s \* 2]; //initialize new array 2 times the size of before

for (int i = 0; i < s; i++) //copy over the data

{

student[i].lastName = original[i].lastName;

student[i].firstName = original[i].firstName;

student[i].score = original[i].score;

student[i].pointGrade = original[i].pointGrade;

}

delete[] original;

\*v = student;

s \*= 2;

cout << "The Class size has been increased to " << s << " students." << endl;

}

string\* data = new string[3];

getwords(data, rawstring);

(\*v)[b].lastName = data[0];

(\*v)[b].firstName = data[1];

(\*v)[b].score = stod(data[2]);

(\*v)[b].pointGrade = scoretoPointGrade((\*v)[b].score);

delete[] data;

b++;

return true;

}

void outputGradeBook(string ofileName, StudentData s[], int size)

{

//first, get percentages for each student

//sort s[] by scores descending

StudentData temp;

int highest\_scorer\_index;

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

{

highest\_scorer\_index = i;

for (int j = i + 1; j < size; j++)

{

if (s[j].score > s[highest\_scorer\_index].score)

{

highest\_scorer\_index = j;

}

}

temp = s[i];

s[i] = s[highest\_scorer\_index];

s[highest\_scorer\_index] = temp;

}

//give students percentages, then later on check for equal scores

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

{

//round to 2 decimal places

s[i].percentage = 100.0 - ((static\_cast<double>(i) / static\_cast<double>(size)) \* 100.0);

}

//check for equal scores

for (int i = 0; i < size - 1; i++)

{

if (s[i].score == s[i + 1].score)

{

s[i + 1].percentage = s[i].percentage;

}

}

//second, get curve grade based on percentages

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

{

if (s[i].percentage > 90)

s[i].curveGrade = A;

else if (s[i].percentage > 75)

s[i].curveGrade = AMINUS;

else if (s[i].percentage > 60)

s[i].curveGrade = BPLUS;

else if (s[i].percentage > 45)

s[i].curveGrade = B;

else if (s[i].percentage > 35)

s[i].curveGrade = BMINUS;

else if (s[i].percentage > 25)

s[i].curveGrade = CPLUS;

else if (s[i].percentage > 15)

s[i].curveGrade = C;

else if (s[i].percentage > 10)

s[i].curveGrade = CMINUS;

else if (s[i].percentage > 5)

s[i].curveGrade = D;

else

s[i].curveGrade = F;

}

//third, find final grade

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

{

if (s[i].score <= 60)

{

s[i].finalGrade = F;

}

else if (s[i].curveGrade > s[i].pointGrade)

{

s[i].finalGrade = s[i].curveGrade;

}

else

{

s[i].finalGrade = s[i].pointGrade;

}

}

//sort s[] by last name using selection sort

int min\_LN\_index;

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

{

min\_LN\_index = i;

for (int j = i + 1; j < size; j++)

{

if (s[min\_LN\_index].lastName.compare(s[j].lastName) > 0)

{

min\_LN\_index = j;

}

}

temp = s[i];

s[i] = s[min\_LN\_index];

s[min\_LN\_index] = temp;

}

//output into ofstream

ofstream grades(ofileName);

cout << "The total enrollment = " << size << " students.\n";

grades << "123456789123456789123456789123456789132456789123456789123456789123456789123456789123456789123456789123456789123456789\n";

grades << setw(20) << left << "Last Name"

<< setw(20) << left << "First Name"

<< setw(10) << right << "Points"

<< " "

<< setw(10) << left << "Grade"

<< setw(10) << right << "Percent"

<< " "

<< setw(10) << left << "Grade"

<< setw(10) << left << "Final"

<< endl;

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

{

grades << fixed << setprecision(2);

grades << setw(20) << left << s[i].lastName

<< setw(20) << left << s[i].firstName

<< setw(10) << right << s[i].score

<< " "

<< setw(10) << left << letterGrade(s[i].pointGrade)

<< setw(10) << right << s[i].percentage

<< " "

<< setw(10) << left << letterGrade(s[i].curveGrade)

<< setw(10) << left << letterGrade(s[i].finalGrade)

<< endl;

}

grades.close();

}

bool getwords(string\* array, string rawstring)

{

//split data into an array with the delimiiter as ' ' or '\t'

int index = 0;

for (int i = 0; i < rawstring.length() - 1; i++)

{

if (rawstring[i] != ' ' && rawstring[i] != '\t')

{

array[index] += rawstring[i];

}

//assuming it can have multiple ' ' or '\t'

//if there is a whitespace after a nonwhite space character, it is the end of one word.

if (rawstring[i] != ' ' && (rawstring[i + 1] == ' ' || rawstring[i + 1] == '\t'))

{

index++;

}

}

//loop only went up to length-1 so append last char

array[index] += rawstring[rawstring.length()];

return true;

}

Grade scoretoPointGrade(double score)

{

if (score > 93)

return A;

if (score > 90)

return AMINUS;

if (score > 87)

return BPLUS;

if (score > 83)

return B;

if (score > 80)

return BMINUS;

if (score > 77)

return CPLUS;

if (score > 73)

return C;

if (score > 70)

return CMINUS;

if (score > BAR)

return D;

return F;

}

string letterGrade(Grade g)

{

string s;

switch (g)

{

case F: s = "F";

break;

case D: s = "D";

break;

case CMINUS: s = "C-";

break;

case C: s = "C";

break;

case CPLUS: s = "C+";

break;

case BMINUS: s = "B-";

break;

case B: s = "B";

break;

case BPLUS: s = "B+";

break;

case AMINUS: s = "A-";

break;

case A: s = "A";

break;

default: s = "Error";

}

return s;

}

Text

Description automatically generated