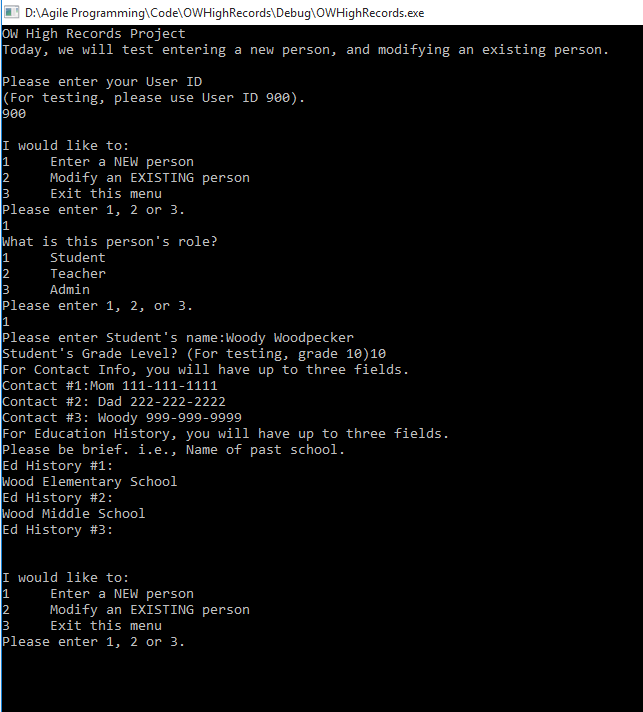
Sarah Wood

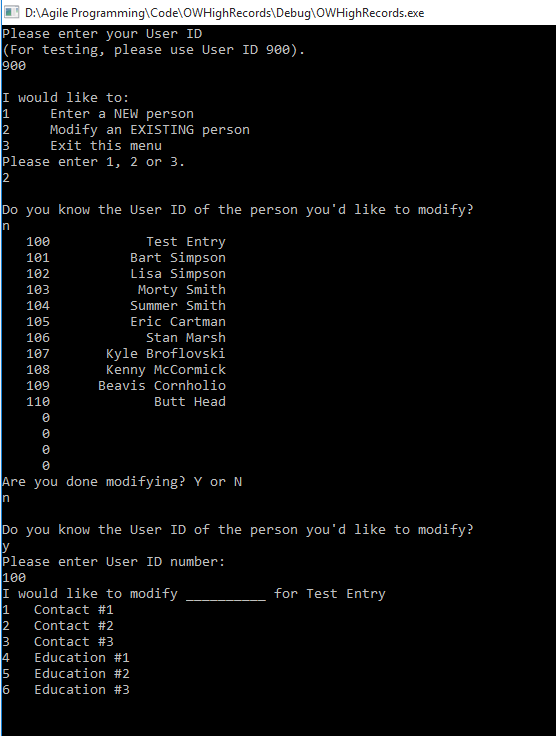
Agile Programming

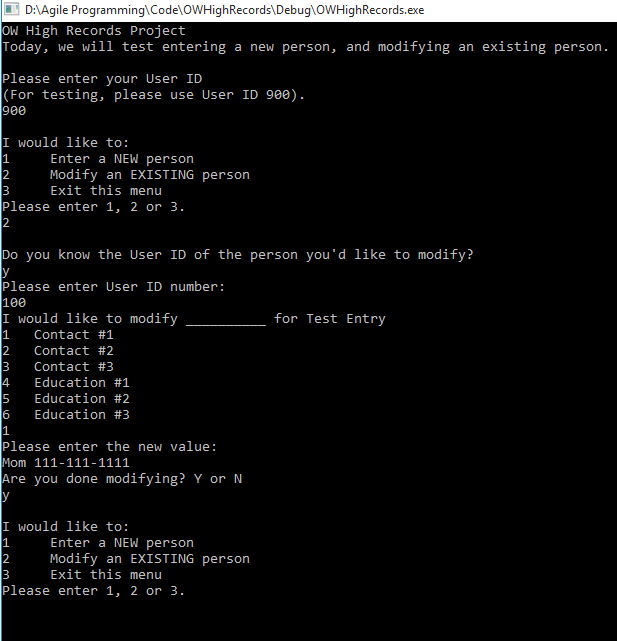
P2-Deliverable 2

Add Records



Modify Existing Record





Source Code (relevant files)

&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

Source.cpp

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

/\*

Sarah Wood

Agile Programming

OW High Records Project

May 2016

Template for file output:

Student ID Date Time

3141592 1/2/1234 4.00

1234586 2/1/4321 2.50

Cpp files I will want to review:

ch16, a 15 & 16

ch15, a 3, 12, 13

Final Project

\*/

#include <iostream>

#include <iomanip>

#include <string>

#include <fstream>

#include <ctime>

#include "Person.h"

#include "StudentAttendance.h"

using namespace std; //no custom namespace is required

const int NUM\_OF\_DAYS = 367; //2016 is a leap year, 0 holds no data

const int NUM\_OF\_STUDENTS = 15;

const int NUM\_OF\_TEACHERS = 5;

const int NUM\_OF\_ADMINS = 3;

//make an array of generic students

//Student(int ID, string name, string role, int grade, string cont1, string cont2, string cont3, string ed1, string ed2, string ed3)

Student ArrayOfStudents[NUM\_OF\_STUDENTS] = {

Student(100, "Test Entry", "Student", 10, "111-111-1111", "", "", "Test Elementary","",""),

Student(101, "Bart Simpson", "Student", 10, "Mom Marge 222-222-2222", " ", " " , "Springfield Elementary", "Springfield Middle School", "Grade 4: Springfield School for the Gifted"),

Student(102, "Lisa Simpson", "Student", 10, "Mom Marge 222-222-2222", "", "", "Springfield Elementary", "Springfield Middle School", ""),

Student(103, "Morty Smith", "Student", 10, "Mom Beth 333-333-3333", "Dad Jerry 444-444-4444", "Grandpa Rick 555-555-5555", "", "", "" ),

Student(104, "Summer Smith", "Student", 10, "Mom Beth 333-333-3333", "Dad Jerry 444-444-4444", "Grandpa Rick 555-555-5555", "", "", ""),

Student(105, "Eric Cartman", "Student", 10, "Mom Liane 777-777-7777","","","South Park Elementary", "",""),

Student(106, "Stan Marsh", "Student", 10, "Dad Randy 888-888-8888", "Mom Sharon 999-999-9999", "", "South Park Elementary", "", ""),

Student(107, "Kyle Broflovski", "Student", 10, "Mom Sheila 123-456-7890", "Dad Gerald 234-567-8901", "", "South Park Elementary", "", ""),

Student(108, "Kenny McCormick", "Student", 10, "Mom Carol 345-678-9012", "Dad Stuart 345-678-9012", "", "South Park Elementary", "", ""),

Student(109, "Beavis Cornholio", "Student", 10, "Mom 456-789-0123", "", "", "","","" ),

Student(110, "Butt Head", "Student", 10, "Mom 567-890-1234","","","","",""),

};

//Make an array of teachers

//Teacher(ID, name, role, Room#, AttAccessYN)

Teacher ArrayOfTeachers[NUM\_OF\_TEACHERS] = {

Teacher(800, "Test Teacher", "Teacher", 999, true),

Teacher(801, "Edna Krabapple", "Teacher", 123, true),

Teacher(802, "Mr. Goldenfold", "Teacher", 125, true),

};

//Make an array of Admins

Admin ArrayOfAdmins[NUM\_OF\_ADMINS] = {

Admin(900, "Test Admin", "Admin", true),

Admin(901, "Seymour Skinner", "Admin", true),

};

//Build an array of attendance records for each student

StudentDailyAttendance AttendanceArray[9999] = {};

//function declarations

int fileIOforStudentAttendanceRecords(Student\*, StudentDailyAttendance\*);

void unitTestingFunction(Student\*, StudentDailyAttendance\*, Teacher\*, Admin\*);

void makeNewPerson(Student\*, StudentDailyAttendance\*, Teacher\*, Admin\*);

void modifyExistingPerson(Student\*, StudentDailyAttendance\*, Teacher\*, Admin\*);

void viewStudents(Student\*, StudentDailyAttendance\*);

void viewTeachers(Teacher\*);

void viewAdmins(Admin\*);

void main()

{

//init console

cout << "OW High Records Project" << endl;

//fileIOforStudentAttendanceRecords(ArrayOfStudents, AttendanceArray);

//unitTestingFunction(ArrayOfStudents, AttendanceArray, ArrayOfTeachers, ArrayOfAdmins);

cout << "Today, we will test entering a new person, and modifying an existing person." << endl;

makeNewPerson(ArrayOfStudents, AttendanceArray, ArrayOfTeachers, ArrayOfAdmins);

//keep window open until we want to close it

cout << '\n' << "Press ENTER to exit program.";

cin.ignore(256, '\n');//ignore up to 256 characters, stop ignoring at any instance of '\n'

cin.get();

}

int fileIOforStudentAttendanceRecords(Student \*stArr ,StudentDailyAttendance \*att)

{

fstream attRecordOUT, attRecordIN;

string tempDate;

int tempID;

double tempHrs;

cout << "Processing record file output." << endl;

//open file, write

attRecordOUT.open("testrecords2.txt");

if (attRecordOUT)

{

attRecordOUT << "ID Number" << '\t' << "Date" << '\t' << "Hours" << '\n';

for (int i = 1; i < 20; i++)

{

attRecordOUT << att[i].IDNumber << '\t' << att[i].Date << '\t' << att[i].TimeInClass << '\n';

}

}

else

{

cout << "Record File Output Error." << endl;

return 0;

}

//open file, read

attRecordIN.open("testrecord.txt");

if (attRecordIN)

{

cout << "Reading from file." << '\n';

cout << "ID Number" << '\t' << "Date" << '\t' << "Hours" << '\n';

attRecordIN.ignore(256, '\n'); //ignore the first line: it's just headers

for (int i=1; !attRecordIN.eof(); i++) // "i" will correspond to the unique attendance entry ID.

{

attRecordIN >> tempID >> tempDate >> tempHrs;

att[i].IDNumber = tempID;

att[i].Date = tempDate;

att[i].TimeInClass = tempHrs;

cout << setprecision(2);

cout <<tempID << '\t' << '\t' << tempDate << '\t' << tempHrs << '\n';

i++;

}

}

else

{

cout << "Record File Input Error." << endl;

return 0;

}

return 1;

}

void unitTestingFunction(Student \*stArr, StudentDailyAttendance \*att, Teacher \*tArr, Admin \*aArr)

{

//quick visual check of objects, structures, arrays, storage, and populations

/\*

cout << "Our people are:" << endl;

cout << endl << "Admins" << endl;

for (int i = 0; i < NUM\_OF\_ADMINS ; i++)

{

cout <<setw(6)<< aArr[i].getPersonIDNumber() << '\t' <<setw(20)<< aArr[i].getPersonName() << '\t' <<setw(10)<< aArr[i].getPersonRole() << '\n';

}

cout << endl << "Faculty" << endl;

for (int i = 0; i < NUM\_OF\_TEACHERS; i++)

{

cout << setw(6) << tArr[i].getPersonIDNumber() << '\t' << setw(20) << tArr[i].getPersonName() << '\t' << setw(10) << tArr[i].getPersonRole()

<< '\t' << setw(6) << tArr[i].getClassRoomNumber() << '\n';

}

cout << endl << "Student Body" << endl;

for (int i = 0; i < NUM\_OF\_STUDENTS; i++)

{

cout << setw(6) << stArr[i].getPersonIDNumber() << '\t' << setw(20) << stArr[i].getPersonName() << '\t' << setw(10) << stArr[i].getPersonRole() <<'\t'<< stArr[i].getCurrentGradeLevel() << '\n';

cout << '\t' << setw(20) << stArr[i].getStudentContactInfo1() << '\t' << stArr[i].getStudentContactInfo2() << '\t' << stArr[i].getStudentContactInfo3() << '\n';

cout << '\t' << stArr[i].getStudentEducationHistory1() << '\n';

cout << '\t' << stArr[i].getStudentEducationHistory2() << '\n';

cout << '\t' << stArr[i].getStudentEducationHistory3() << '\n';

}

\*/

for (int i = 0; i < NUM\_OF\_STUDENTS; i++)

{

cout << setw(6) << stArr[i].getPersonIDNumber() << '\t' << setw(20) << stArr[i].getPersonName() << '\t' << setw(10) << stArr[i].getPersonRole() << '\t' << stArr[i].getCurrentGradeLevel() << '\n';

//check attendance per student

}

}

void makeNewPerson(Student \*stArr, StudentDailyAttendance \*att, Teacher \*tArr, Admin \*aArr)

{

//local variables

int tempID, userChoice, userChoice2;

int tempGrade;

string tempRole, tempName;

string tempCont1, tempCont2, tempCont3, tempEd1, tempEd2, tempEd3;

int tempRoom;

bool tempAttendanceAccess = false;

bool tempAllAccess = false;

bool valid = false;

bool IamDoneEnteringNewPeople = false;

//init interface

cout << endl << "Please enter your User ID" << endl;

cout << "(For testing, please use User ID 900)." << endl;

cin >> tempID;

while (tempID < 900 || tempID >905)

{

cout << endl << "You do not have permission to do this task. Try again."<<endl;

cout << "Please enter your User ID." << endl; //eventually, I would like to implement actual security here. Motivated students can per crafty and persistent.

cin >> tempID;

}

while (IamDoneEnteringNewPeople != true)

{

cout << '\n' << "I would like to:" << endl;

cout << "1 Enter a NEW person" << endl;

cout << "2 Modify an EXISTING person" << endl;

cout << "3 Exit this menu" << endl;

cout << "Please enter 1, 2 or 3." << endl;

cin >> userChoice;

while (!cin)

{

cout << "Invalid Input. Try again. 1, 2 or 3." << '\t';

cin.clear();

cin.ignore();

cin >> userChoice;

}

while (userChoice < 1 || userChoice>3)

{

valid = !true; //declare this attampt !valid

cin.clear(); //clear the input stream

cin.ignore(); //ignore invalid stuff.

cout << "Invalid Input. Try again. 1, 2 or 3. " << '\t'; //user prompt

cin >> userChoice; //try again...

}

if (userChoice == 1)

{

cout << "What is this person's role? " << endl;

cout << "1 Student" << endl;

cout << "2 Teacher" << endl;

cout << "3 Admin" << endl;

cout << "Please enter 1, 2, or 3." << endl;

cin >> userChoice2;

//start with entry validation (data type)

while (!cin)

{

cout << "Invalid Input. Try again. 1, 2 or 3: " << '\t';

cin.clear();

cin.ignore();

cin >> userChoice2;

}

while (userChoice2 < 1 || userChoice>3)

{

valid = !true; //declare this attampt !valid

cin.clear(); //clear the input stream

cin.ignore(); //ignore invalid stuff.

cout << "Invalid Input. Try again. 1, 2 or 3: " << '\t'; //user prompt

cin >> userChoice2; //try again...

}

if (userChoice2 == 1)

{

tempRole = "Student";

//ID# will be generated in the range of 100-799

int highestID = 0;

for (int i = 1; i < NUM\_OF\_STUDENTS;i++)

{

if (stArr[i].getPersonIDNumber() > highestID)

highestID = stArr[i].getPersonIDNumber();

}

//assign a new ID number that is one more than the highest one we currently have

tempID = highestID + 1;

//count current students to determine array placement of new student

int count = 1;

for (int i = 1; i < NUM\_OF\_STUDENTS;i++)

{

if (stArr[i].getPersonIDNumber() != 0)

{

count++;

}

}

//get info to make a Student

cout << "Please enter Student's name:";

cin >> tempName;

cin.ignore(256, '\n');

cout << "Student's Grade Level? (For testing, grade 10)";

cin >> tempGrade;

cin.ignore(256, '\n');

cout << "For Contact Info, you will have up to three fields." << endl;;

cout << "Contact #1:";

getline(cin, tempCont1);

cout << "Contact #2: ";

//getline(cin, tempCont2);

getline(cin, tempCont2);

cout << "Contact #3: ";

//getline(cin, tempCont3);

getline(cin, tempCont3);

cout << "For Education History, you will have up to three fields." << endl;

cout << "Please be brief. i.e., Name of past school." << endl;

cout << "Ed History #1:" << endl;

getline(cin, tempEd1);

cout << "Ed History #2: " << endl;

getline(cin, tempEd2);

cout << "Ed History #3: " << endl;

getline(cin, tempEd3);

//make a Student

//Student(int ID, string name, string role, int grade, string cont1, string cont2, string cont3, string ed1, string ed2, string ed3)

stArr[count].setPersonIDNumber(tempID);

stArr[count].setPersonName(tempName);

stArr[count].setPersonRole(tempRole);

stArr[count].setCurrentGradeLevel(tempGrade);

stArr[count].setStudentContactInfo1(tempCont1);

stArr[count].setStudentContactInfo2(tempCont2);

stArr[count].setStudentContactInfo3(tempCont3);

stArr[count].setStudentEducationHistory1(tempEd1);

stArr[count].setStudentEducationHistory2(tempEd2);

stArr[count].setStudentEducationHistory3(tempEd3);

}

else if (userChoice2 == 2)

{

tempRole = "Teacher";

//ID# will be generated in the range 800-899

int highestID = 0;

for (int i = 1; i < NUM\_OF\_TEACHERS;i++)

{

if (stArr[i].getPersonIDNumber() > highestID)

highestID = stArr[i].getPersonIDNumber();

}

//assign a new ID number that is one more than the highest one we currently have

tempID = highestID + 1;

//count current students to determine array placement of new student

int count = 1;

for (int i = 1; i < NUM\_OF\_TEACHERS;i++)

{

if (stArr[i].getPersonIDNumber() != 0)

{

count++;

}

}

//get Info to make a Teacher

tempAttendanceAccess = true;

cout << "Please enter Teacher's name: " << endl;

cin >> tempName;

cout << "Please enter Teacher's Room Number: " << endl;

cin >> tempRoom;

//make a Teacher

//Teacher(int id, string name, string role, int room, bool AttAcc)

tArr[count].setPersonIDNumber(tempID);

tArr[count].setPersonRole(tempRole);

tArr[count].setPersonName(tempName);

tArr[count].setClassRoomNumber(tempRoom);

tArr[count].setAttendanceRecordsAccess(tempAttendanceAccess);

}

else if (userChoice2 == 3)

{

tempRole = "Admin";

//ID# will be generated in the range 900-999

int highestID = 0;

for (int i = 1; i < NUM\_OF\_ADMINS;i++)

{

if (stArr[i].getPersonIDNumber() > highestID)

highestID = stArr[i].getPersonIDNumber();

}

//assign a new ID number that is one more than the highest one we currently have

tempID = highestID + 1;

//count current students to determine array placement of new student

int count = 1;

for (int i = 1; i < NUM\_OF\_ADMINS;i++)

{

if (stArr[i].getPersonIDNumber() != 0)

{

count++;

}

}

//get info to make a new admin

cout << "Please enter new Admin's name: " << endl;

cin >> tempName;

tempAllAccess = true;

//make an Admin

//Admin(int id, string name, string role, bool AllAcc)

aArr[count].setPersonIDNumber(tempID);

aArr[count].setPersonName(tempName);

aArr[count].setPersonRole(tempRole);

aArr[count].setAllRecordsAccess(tempAllAccess);

}

else

{

cout << "Unknown error occurred." << endl;

}

}//close "Enter new person"

else if (userChoice == 2)

{

modifyExistingPerson(ArrayOfStudents, AttendanceArray, ArrayOfTeachers, ArrayOfAdmins);

}

else if (userChoice == 3)

{

IamDoneEnteringNewPeople = true;

}

}//IamDoneEnteringNewPeople

}

void modifyExistingPerson(Student \*stArr, StudentDailyAttendance \*att, Teacher \*tArr, Admin \*aArr)

{

char YN, YN2;

bool IamDoneModifyingThisPerson = false;

while (IamDoneModifyingThisPerson != true)

{

//modify existing person

cout << endl << "Do you know the User ID of the person you'd like to modify?" << endl;

cin >> YN;

while (YN != 'Y' && YN != 'y'&& YN != 'n' && YN != 'N')

{

cout << "Invalid input. Try again. Y or N." << '\t';

cin.clear();

cin.ignore();

cin >> YN;

}

if (YN == 'y' || YN == 'Y')

{

cout << "Please enter User ID number: " << endl;

int value;

cin >> value;

//search stArr for id#

int i = 0;

int position = -1;

bool found = false;

while (i < NUM\_OF\_STUDENTS && !found)

{

if (stArr[i].getPersonIDNumber() == value)

{

found = true;

position = i;

}

i++; //else add one and test again

}//while !found

if (!found)

{

cout << "User not found." << endl;

}

else if (found)

{

int userChoice3;

//UI "I would like to modify X value for this student"

cout << "I would like to modify \_\_\_\_\_\_\_\_\_\_ for " << stArr[position].getPersonName() << endl;

cout << "1 Contact #1" << endl;

cout << "2 Contact #2" << endl;

cout << "3 Contact #3" << endl;

cout << "4 Education #1" << endl;

cout << "5 Education #2" << endl;

cout << "6 Education #3" << endl;

cin >> userChoice3;

while (userChoice3 < 1 || userChoice3>6)

{

cin.clear(); //clear the input stream

cin.ignore(); //ignore invalid stuff.

cout << "Invalid Input. Try again. 1, 2 , 3, 4, 5, or 6." << '\t'; //user prompt

cin >> userChoice3;

}

string userEntry;

if (userChoice3 == 1)

{

cout << "Please enter the new value:" << endl;

getline(cin, userEntry);

cin.ignore(256, '\n');

stArr[position].setStudentContactInfo1(userEntry);

}

else if (userChoice3 == 2)

{

cout << "Please enter the new value:" << endl;

getline(cin, userEntry);

cin.ignore(256, '\n');

stArr[position].setStudentContactInfo2(userEntry);

}

else if (userChoice3 == 3)

{

cout << "Please enter the new value:" << endl;

getline(cin, userEntry);

cin.ignore(256, '\n');

stArr[position].setStudentContactInfo3(userEntry);

}

else if (userChoice3 == 4)

{

cout << "Please enter the new value:" << endl;

getline(cin, userEntry);

cin.ignore(256, '\n');

stArr[position].setStudentEducationHistory1(userEntry);

}

else if (userChoice3 == 5)

{

cout << "Please enter the new value:" << endl;

getline(cin, userEntry);

cin.ignore(256, '\n');

stArr[position].setStudentEducationHistory2(userEntry);

}

else if (userChoice3 == 6)

{

cout << "Please enter the new value:" << endl;

getline(cin, userEntry);

cin.ignore(256, '\n');

stArr[position].setStudentEducationHistory3(userEntry);

}

}

}//if Yes

else if (YN == 'n' || YN == 'N')

{

//display list of students

for (int i = 0; i < NUM\_OF\_STUDENTS; i++)

{

cout << setw(6) << stArr[i].getPersonIDNumber() << '\t' << setw(20) << stArr[i].getPersonName() << endl;

}

}//if NO

cout << "Are you done modifying? Y or N" << endl;

cin >> YN2;

while (YN2 != 'Y' && YN2 != 'y'&& YN2 != 'n' && YN2 != 'N')

{

cout << "Invalid input. Try again. Y or N." << '\t';

cin.clear();

cin.ignore();

cin >> YN2;

}

if (YN == 'y' || YN == 'Y')

{

IamDoneModifyingThisPerson = true;

}

else if (YN2 == 'n' || YN2 == 'N')

{

IamDoneModifyingThisPerson = false;

}

}//IamDoneModifyingThisPerson

}

void viewStudents(Student \*stArr, StudentDailyAttendance \*att)

{

}

void viewTeachers(Teacher \*tArr)

{

}

void viewAdmins(Admin \*aArr)

{

}