|  |
| --- |
| ~ |
| Program 9 – Student Class 2 |
| A classic comedy romp for the whole family! |
|  |
| **Brett Kriz** |
| **4/21/2014** |

|  |
| --- |
| ` |

Prog9.cpp:

/\*------------------------------------------

Brett Kriz

Program 8.2 AKA 9

-------------------------------------------\*/

#include <algorithm> // Algorithms >.>

#include <cmath> // Basic Math Operations

#include <cstdlib> // control of stdlib

#include <cstring> // Cstrings

#include <ctime> // Time stuff

#include <ctype.h> // typeing

#include <fstream> // File stream (I/O)

#include <iomanip> // Manipulation of I/O

#include <iterator>

#include <iostream> // Input\Output

#include <sstream> // String stream

#include <stdlib.h> // Standard Library

#include <string> // String Stuff

#include <string.h> // String Stuff

#include <time.h> // More time stuff

#include "Student.h"

// SEE HEADER

using namespace std;

// Variables

const int ARRAY\_SIZE = 250;

struct List{ // Nice struct for lists

int size;

Student students[ARRAY\_SIZE];

};

void body(); //Module of program, for repetition if needed

void sleep(unsigned int mseconds = 250);// Sleep for slower outputs mostly

void printall(List s); // Prints all the students in s[]

int upd(int& n); // unbugable ++iterator, ++x is prone to certian code step problems :/

void resetpanel(bool cls = true); // default to true/use clear screen

string trim(const string &t); // trims the string of leading and ending space

string token\_burn(string& line, string den, int lookat = 0); // Shortens the line by finding the next den, returns a token string

bool Read\_IN(List& data, string des = "studentdata.txt");

int main()

{

resetpanel();

body();

cout << "\n";

system("pause");

return 0;

}

void resetpanel(bool cls){

system("color F0");

system("title PROGRAM 9: Student Class v2");

if (cls) {

system("cls");

cin.clear();

}else{

cout << "\n";

cin.clear();

}

}

// Code Body

void body(){

// VARIABLES

int x = 0, size = 0;

bool go = true;

List data;

data.size = 0;

string arg = "";

// WORK

Read\_IN( data ); // Read in data

resetpanel(); // Just fo decoration

do{ // LOOP for a menu

cout << "\nWould you like to (add,view,exit)?: ";

cin >> arg;

cin.clear();

if (arg.find("add") < arg.npos){ // npos for not found

Student cur(true); // tag it for clients to fill

data.students[ upd(data.size) ] = cur;

sleep();

}else if(arg.find("view")<arg.npos){ // DISPLAY

system("cls");

printall(data); // PRINT ALL RECORDS

sleep(1000);

}else if(arg.find("exit")<arg.npos){ // EXIT

go = false; // Itll just stop this loop

}else{

cout << "\n\n\t What? I dont know what that means!\n"; // bad in

sleep();

}

}while(go); // end menu

}

void printall(List s){

int x = 0;

while(x < s.size){

s.students[x].Print(); // Print record

x++;

sleep(); // Fo decoration

}

}

void sleep(unsigned int mseconds){

// Sleep...

clock\_t goal = mseconds + clock();

while (goal > clock());

}

int upd(int& n){

int nn = n;

n++; // I know, I know "Just use ++n!", but that was messin up!!

return nn;

}

string trim(const string &t){

string ans = t;

size\_t found;

found = ans.find\_last\_not\_of( " \n\r\t" ); // find Whitespace

if (found != t.npos){

ans.erase( found + 1 );

}else{

ans.clear(); // str is whitespace

}

return ans;

}

string token\_burn(string& line, string den, int lookat){

//Trim

line = trim(line);

int pos = line.find( den, lookat ), x = 0;

string z = "";

// Show where the break will occure

// Under the text! ^

while(x < pos){z += " "; x++;}

z += "^";// HERE

//cout << line << "\n" << z << "\n";

string token = line.substr( 0 , pos );

line = line.substr( pos + 1 ); // till end

//cout << line << "\n\"" << token << "\""

//<< endl;

sleep(20);

return token;

}

bool Read\_IN(List& data, string des){

system("title Program 9: Read in Student List");

ifstream IN;

int x = 0;

bool good = true;

IN.open(des); // Modularity >:3

//cout << endl;

if (IN.is\_open()){

system("color F8");

while(!IN.eof()){

// Variables

Student cur = Student(); // Student Variable

char rawline[ 256 ];

string s = ",", line = "", den = ",";

// Work

IN.getline( rawline, 256 ); // snag all vic C string

line = string(rawline); // create string

if (line.length() < 3 ){

// ABORT Read, bad line

break;

}

// Token\_Burn was such a good idea

// 5 , then newline = 6

string t1, t2, t3, t4, t5, t6;

// TOKENIZE ME CAPTIN!

t1 = token\_burn( line, den, 3 ); // Last

line = line.substr(1); // Space...

t2 = token\_burn( line, den, 2 ); // First & Middle

t3 = token\_burn( line, den, 0 ); // Score 1

t4 = token\_burn( line, den, 0 ); // Score 2

t5 = token\_burn( line, den, 0 ); // Score 3

t6 = line; // Absences (The remaindedr)

cout << "\n RAW #" << x

<< " ~~~~~~~~~~~~~~~~~~~~"

<< "\n LName: >" << t1 << "<"

<< "\n First & Middle: >" << t2 << "<"

<< "\n Score1: >" << t3 << "<"

<< "\n Score2: >" << t4 << "<"

<< "\n Score3: >" << t5 << "<"

<< "\n Absences: >" << t6 << "<"

<< endl;

// Work with the data

char m = t2.substr( t2.size() - 2).c\_str()[0];

string first = t2.substr( 0, t2.find\_first\_of(' '));

// Set

cur.setLName( t1 );

cur.setMInit( m );

cur.setFName( first );

cur.setScore1( atoi(t3.c\_str()) );

cur.setScore2( atoi(t4.c\_str()) );

cur.setScore3( atoi(t5.c\_str()) );

cur.setAbsences( atoi(t6.c\_str()) );

// atoi(line.c\_str());

// Save it!

data.students[x] = cur;

x++; // yup increment

sleep( 50 );

} // end loop

cout << "\n Student read in complete!\n\n";

data.size = x; // Store size

//system("pause");

sleep(950);

}else{ // not Open?

cout << "\n\nInput File is empty!\n\t Check path!\a\n";

good = false;

}// end if

IN.close();// Better close that...

return good;

}// end func

Student.h:

/\*------------------------------------------

Brett Kriz

Student Class

-------------------------------------------\*/

// Include

#include <cmath>

#include <math.h>

#include <iostream>

#include <iomanip>

#include <string>

#include <string.h>

#include <sstream>

#include <fstream> // File stream (I/O)

using namespace std;

//Proto

//bool doover(bool b); // 1 proto

class Student{

private:

// VARIABLES

string fname, lname;

char minit;

int score1, score2, score3, absences;

// FUNCTIONS

bool doover(bool b);// For Loops! Checks if false, prompts user to re-enter; Returns the input bool

// @PUBLIC

public:

// @FUNCTIONS

// Contstructors

Student();

Student(string fn, char mi, string ln, int s1, int s2, int s3, int a);

Student(char to\_ask\_user); // Just to change how its stacked

// Gets

char getMInit();

string getFName();

string getLName();

int getScore1();

int getScore2();

int getScore3();

int getAbsences();

// Sets

void setMInit( char i = ' ' );

void setFName( string s = "" );

void setLName( string s = "" );

void setScore1( int n = -1 );

void setScore2( int n = -1 );

void setScore3( int n = -1 );

void setAbsences( int n = -1 );

// Other

double getFinal();

double getAverage();

string getLetterGrade(double tot = -1);

string gageSuccess(double f);

void Print();

};

// @Constructors

Student::Student(char to\_ask\_user){ // Menu creation style

// char doesnt matter

setFName();

setMInit();

setLName();

setScore1();

setScore2();

setScore3();

setAbsences();

}

Student::Student(){

fname = "";

char minit = ' ';

lname = "";

score1 = 0;

score2 = 0;

score3 = 0;

absences = 0;

}

Student::Student(string fn, char mi, string ln, int s1, int s2, int s3, int a){

fname = fn;

minit = mi; // Edit this area

lname = ln;

score1 = s1;

score2 = s2;

score3 = s3;

absences = a;

}

// @PRIVATE

bool Student::doover(bool b){

if (b == true){ // For while loops

//system("cls");

cout << "\nBad Input! Please re-enter!\n";

cin.clear();

}

return b;

}

// @GETS

void Student::setMInit(char i){

cin.clear();

string n = ""; // for answer processing

if (isalpha(i)){

minit = i;

return;

}

do{

cout << "\nPlease enter Initial: ";

cin >> n;

i = n.c\_str()[0]; // Remove char

cout << endl;

}while( doover( !isalpha(i) || i == ' ' ) );

minit = i;

cout << endl;

}

void Student::setFName(string s){ // overload for file in

cin.clear();

if (s.size() > 3) {

fname = s;

return;

}

cout << "\nPlease enter First Name: ";

cin >> s;

fname = s;

cout << endl;

}

void Student::setLName(string s){

cin.clear();

if (s.size() > 3) {

lname = s;

return;

}

cout << "\nPlease enter Last Name: ";

cin >> s;

lname = s;

cout << endl;

}

void Student::setScore1(int n){

// int n = -1; // Trickledown With Error Precention

cin.clear();

if (n >= 0 && n <= 100){

score1 = n;

return;

}

do{

cout << "\nPlease enter Score1: ";

cin >> n;

cout << endl;

}while( doover(n<0 || n>100) );

score1 = n;

cout << endl;

}

void Student::setScore2(int n){

// int n = -1; // Trickledown With Error Precention

cin.clear();

if (n >= 0 && n <= 100){

score2 = n;

return;

}

do{ // If 'n' wasnt actually given , ask for it!

cout << "\nPlease enter Score2: ";

cin >> n;

}while( doover(n<0 || n>100) );

score2 = n;

cout << endl;

}

void Student::setScore3(int n){

// int n = -1; // Trickledown With Error Precention

cin.clear();

if (n >= 0 && n <= 100){

score3 = n;

return;

}

do{ // If 'n' wasnt actually given , ask for it!

cout << "\nPlease enter Score3: ";

cin >> n;

}while( doover(n<0 || n>100) );

score3 = n;

cout << endl;

}

void Student::setAbsences(int n){

// int n = -1; // Trickledown With Error Precention

cin.clear();

if (n >= 0 && n <= 365\*4){

absences = n;

return;

}

do{ // If 'n' wasnt actually given , ask for it!

cout << "\nPlease enter Absences: ";

cin >> n;

}while( doover(n < 0 || n > 365\*4) );

absences = n;

cout << endl;

}

// @GETS

char Student::getMInit(){

return minit;

}

string Student::getFName(){

return fname;

}

string Student::getLName(){

return lname;

}

int Student::getScore1(){

return score1;

}

int Student::getScore2(){

return score2;

}

int Student::getScore3(){

return score3;

}

int Student::getAbsences(){

return absences;

}

// @OTHER FUNCTIONS

double Student::getAverage(){ // The literal average

return ( score1 + score2 + score3 )/3.0 ;

}

double Student::getFinal(){ // The number for percent display/total score

int xtra = 0;

if (absences == 0){

xtra = 2;

}

return (( score1 + score2 + score3 )/300.0)\*100 + xtra;

}

string Student::gageSuccess(double f){

if (f >= 73){ // Greater than or equal to a 'C'

return "Successful!";

}else{

return "Unsuccessful..";

}

}

void Student::Print(){

cout << "\nName: " << getFName() << " " << getMInit() << ". " << getLName()

<< "\nTest Average: " << getAverage()

//<< "\nScore1: " << getScore1()

//<< "\nScore2: " << getScore2()

//<< "\nScore3: " << getScore3()

<< "\nAbsences: " << getAbsences()

//<< "\nFinal Grade: " << fixed << setprecision(2) << getFinal() << "%"

<< "\nLetter Grade: " << getLetterGrade()

<< "\n\t" << gageSuccess(getFinal())

<< endl;

} // End Print

string Student::getLetterGrade(double arg){

string ans = "F"; // Default it

if (arg == -1) { // Handle unsent info :D

arg = getFinal();

}

if (arg > 100){

ans = "A+";

//A

//93.0% - 100%

}else if( arg >= 93.0 && arg <= 100 ){

ans = "A";

//A-

//90.0% - 92.99%

}else if( arg >= 90 && arg < 93 ){

ans = "A-";

//B+

//87.0% - 89.99%

}else if( arg >= 87 && arg < 90 ){

ans = "B+";

//B

//83.0% - 86.99%

}else if( arg >= 83 && arg < 87 ){

ans = "B";

//B-

//80.0% - 82.99%

}else if( arg >= 80 && arg < 83 ){

ans = "B-";

//C+

//77.0% - 79.99%

}else if( arg >= 77 && arg < 80 ){

ans = "C+";

//C

//73.0% - 76.99%

}else if( arg >= 73 && arg < 78 ){

ans = "C";

//C-

//70.0% - 72.99%

}else if( arg >= 70 && arg < 73 ){

ans = "C-";

//D+

//67.0% - 69.99%

}else if( arg >= 67 && arg < 70 ){

ans = "D+";

//D

//60.0% - 66.99%

}else if( arg >= 60 && arg < 67 ){

ans = "D";

} // I guess no D- then...

//F

//Below 60.0%

return ans;

}

File: studentdata.txt

Output: Same menu, new output

