ЗМІСТ

[ВСТУП 4](#_Toc390260761)

[1 АНАЛІЗ ТЕХНІЧНОГО ЗАВДАННЯ 6](#_Toc390260762)

[2 ОБҐРУНТУВАННЯ АЛГОРИТМУ І СТРУКТУРИ ПРОГРАМИ 7](#_Toc390260763)

[3 РОЗРОБКА ПРОГРАМИ 8](#_Toc390260764)

[4 ТЕСТУВАННЯ ПРОГРАМИ І РЕЗУЛЬТАТИ ЇЇ ВИКОНАННЯ 15](#_Toc390260765)

[ВИСНОВОК 20](#_Toc390260766)

[ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ 21](#_Toc390260767)

ДОДАТОК А………………………………………………………………..22

# ВСТУП

Курсовий проект є комплексним завданням по застосуванню та закріпленню вивченого матеріалу за курс з ведучої дисципліни (в даному випадку з об'єктно-орієнтованого програмування). Він включає в себе розробку повноцінної системи (згідно з варіантом) і дає можливість проявити креативність і свої здібності в програмуванні та на кодовій частині проекту.

Розробляючи проект дуже важливим аспектом є створення раціональної системи та її структури, які б підходили під усі вимоги, та найголовніше – виконувала свою задачу якомога найкраще. В даному курсовому проекті представлена програма обробки результатів сесії студентів ІІ-го курсу.

Програмування – це процес написання, тестування і підтримки комп'ютерних програм. З розвитком програмування виникла ідея поєднати в межах однієї сутності дані і код, що безпосередньо опрацьовує ці дані. Така сутність отримала назву об’єкт, а відповідний підхід до створення програм називають об’єктно-орієнтованим програмуванням.

Об’єктно-орієнтоване програмування (ООП) – це парадигма програмування, яка розглядає програму як сукупність гнучко пов’язаних між собою об’єктів.

Основні переваги концепції ООП:

* можливість створювати користувацькі типи даних (класи);
* приховування деталей реалізації (інкапсуляція);
* можливість повторного використання коду (наслідування);

Метою даної роботи є створення програми для обробки результатів сесії студентів ІІ-го курсу. Для цього використовувалось середовище розробки – Microsoft Visual Studio 2010.

Microsoft Visual Studio – серія продуктів фірми [Майкрософт](http://uk.wikipedia.org/wiki/%D0%9C%D0%B0%D0%B9%D0%BA%D1%80%D0%BE%D1%81%D0%BE%D1%84%D1%82), які включають [інтегроване середовище розробки](http://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B5_%D1%81%D0%B5%D1%80%D0%B5%D0%B4%D0%BE%D0%B2%D0%B8%D1%89%D0%B5_%D1%80%D0%BE%D0%B7%D1%80%D0%BE%D0%B1%D0%BA%D0%B8) програмного забезпечення та ряд інших інструментальних засобів. Ці продукти дозволяють розробляти як [консольні програми](http://uk.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2%D0%B8%D0%B9_%D1%96%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BA%D0%BE%D1%80%D0%B8%D1%81%D1%82%D1%83%D0%B2%D0%B0%D1%87%D0%B0), так і програми з [графічним інтерфейсом](http://uk.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84%D1%96%D1%87%D0%BD%D0%B8%D0%B9_%D1%96%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81), в тому числі з підтримкою технології  [Windows Forms](http://uk.wikipedia.org/w/index.php?title=Windows_Forms&action=edit&redlink=1), а також [веб-сайти](http://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B0%D0%B9%D1%82),  [веб-застосунки](http://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%B7%D0%B0%D1%81%D1%82%D0%BE%D1%81%D1%83%D0%BD%D0%BE%D0%BA), [веб-служби](http://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%BB%D1%83%D0%B6%D0%B1%D0%B0) як в [рідному](http://uk.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D0%B8%D0%B9_%D0%BA%D0%BE%D0%B4), так і в [керованому](http://uk.wikipedia.org/wiki/%D0%9A%D0%B5%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B9_%D0%BA%D0%BE%D0%B4) кодах для всіх платформ, що підтримуються [Microsoft Windows](http://uk.wikipedia.org/wiki/Microsoft_Windows), [Windows Mobile](http://uk.wikipedia.org/wiki/Windows_Mobile), [Windows CE](http://uk.wikipedia.org/wiki/Windows_CE), [.NET Framework](http://uk.wikipedia.org/wiki/.NET_Framework), [.NET Compact Framework](http://uk.wikipedia.org/wiki/.NET_Compact_Framework) та [Microsoft Silverlight](http://uk.wikipedia.org/wiki/Microsoft_Silverlight).

Інструменти Visual Studio 2010 допоможуть не тільки в створенні звичних програм для мобільних телефонів і персональних комп'ютерів, але в розробці хмарних застосунків. При цьому процес тестування, зневадження і розгортання програм в «хмарі» аналогічний створенню .NET-застосунків. Іншим важливим доповненням в Visual Studio 2010 є інструменти для багатонитевої розробки з використанням як некерованого коду, так і .NET Framework.

У Visual Studio 2010 повністю перероблений інтерфейс з використанням Windows Presentation Foundation (WPF), упроваджено наступне покоління інструментів ASP.NET, є підтримка динамічних розширень в мовах програмування C# і Visual Basic, використовуються нові шаблони проектів, інструментарій для документування тестових сценаріїв і велика кількість нових бібліотек, що підтримують Windows 7.

# АНАЛІЗ ТЕХНІЧНОГО ЗАВДАННЯ

Дана програма створена для обробки результатів сесії студентів ІІ-го курсу. Середовище розробки програми: Microsoft Visual Studio 2010. Програма написана на мові програмування С++, вона високого рівня з підтримкою декількох парадигм програмування: об’єктно-орієнованої, узагальненої та процедурної. С++ додає до С об'єктно-орієнтовані можливості. Вона вводить класи, які забезпечують три найважливіші властивості ООП: інкапсуляцію, успадкування і поліморфізм.

Інкапсулювання – це механізм в програмуванні, який пов’язує в одне ціле функції і дані, якими вони маніпулють, а також захищає їх від зовнішнього доступу і неправильного застосування.

Успадкування – це властивість, з допомогою якої один об’єкт може набувати властивостей іншого. При цьому підтримується концепція ієрархічної класифікації.

Поліморфізм дозволяє писати більш абстрактні програми і підвищити коефіцієнт повторного використання коду.

Програма виконує:

* Зчитування даних про студентів з файлу;
* Виведення середнього балу студента та його оцінок;
* Виведення кращого і гіршого студента;
* Сортування студентів по середньому балу від кращого до гіршого і від гіршого до кращого;
* Сортування студентів по балах з вказаного предмета від кращого до гіршого та від гіршого до кращого;
* Виведення студентів які здали сесію і котрі не здали сесії;
* Виведення студентів в яких бал вищий вказаного по вказаному предмету, і в яких нижчий.

# ОБґРУНТУВАННЯ АЛГОРИТМУ І СТРУКТУРИ ПРОГРАМИ

В програмі створено пару класів. Клас Parser, який виконує парсинг стрічки. Парсинг – це скрипт або програма, що займається, безпосередньо, збором, аналізом і перетворенням необхідної інформації. Клас Student який вертає ім’я студента, його рейтинг по предметах, середній бал, виводить значення на екран. Клас Loder, котрий має ім’я файлу Бази даних студентів і використовує клас Parser для розбору стрічок та клас Util ,котрий містить всі методи якими можна впливати на масив студентів. Виконує основне завдання програми, обробляє дані.

Класи та функції які знаходяться в програмі розбиті на модулі – кожен наслідуваний клас має відповідний набір методів, які виконують роль реалізації програми.

Програма реалізує вікно, в якому можна побачити результати сесії . В наслідуваних файлах заголовках включено основні файли заголовки зі стандартної бібліотеки С++.

# РОЗРОБКА ПРОГРАМИ

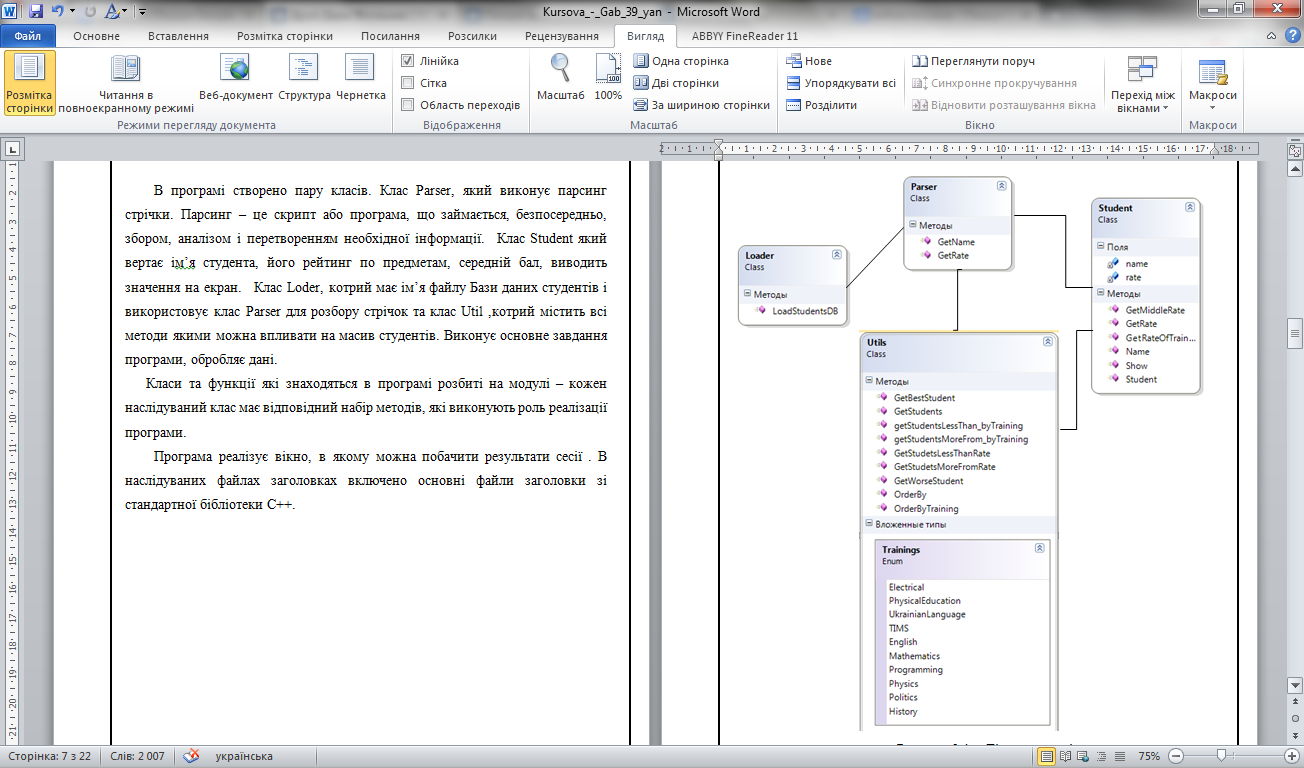


Рисунок 3.1 – Діаграма класів

Клас Parser (Рис. 3.2) виконує парсинг стрічки. У цьому класі представлені два методи. Метод GetName – парсит( дістає) з стрічки ім’я студента та метод GetRate, який витягає вектор оцінок

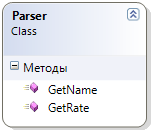


Рис. 3.2 Клас Parser

Клас Loder (Рис 3.3) використовує клас Parser для розбору стрічок. У цьому класі представлений метод LoadStudentsDB , який загружає дані у масив студента.

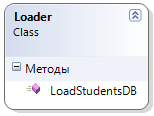
vector<Student> studentDB = vector<Student>();

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

studentDB.push\_back(Student(Parser::GetName(lines[i]), Parser::GetRate(lines[i])));

return studentDB;

}

Рис. 3.3 Клас Loder

Клас Student (Рис. 3.4.) вертає ім’я студента і масив оцінок по предметам, переліченим в enum Trainings. Він має такі поля як name (ім’я) та rate (рейтинг), тобто оцінки по предмету доступні по індексу предмета.

Клас Student представлений такими методати , як:

GetRateofTraining – повертає рейтинг по вказаному предмету.

int Student::GetRateOfTraining(int trainingIndex)

{

return (trainingIndex < rate.size()) ? this->rate[trainingIndex] : -1;

}

GetMiddleRate – повертає середній бал по всім предметам

double Student::GetMiddleRate()

{

double tmpMiddleRate = 0;

for (int i = 0; i < this->rate.size(); i++)

tmpMiddleRate += ((double)this->rate[i]);

return tmpMiddleRate /= ((double)(this->rate.size()));

}

Show – виводить на екран інформацію про студента

void Student::Show()

{

cout << this->name << endl;

for (int i = 0; i < this->rate.size(); i++)

cout << rate[i] << ' ';

cout << endl;

}

Name – вивиодить ім’я студента

string Student::Name()

{

return this->name;

}

GetRate – виводить список оцінок студента

vector<int> Student::GetRate()

{

return this->rate;

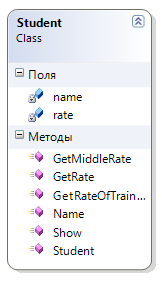


Рис. 3.4 Клас Student

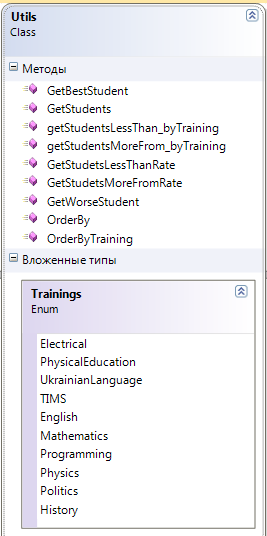
Клас Utils (Рис. 3.5) містить всі методи якими можна впливати на масив студентів. А також виконує основне завдання програми, обробляє дані.

Рис. 3.5 клас Utils

Клас Utils представлений такими методати як: GetBestStudent – виводить кращого студента (по середньому балу)

Student Utils::GetBestStudent(vector<Student> \_dbStudent)

{

double minMiddleRate = 0;

int index = -1;

for (int i = 0; i < \_dbStudent.size(); i++)

if (\_dbStudent[i].GetMiddleRate() > minMiddleRate)

{

minMiddleRate = \_dbStudent[i].GetMiddleRate();

index = i;

}

return \_dbStudent[index];

}

GetWorsesStudent - виводить гіршого студента (по середньому балу)

Student Utils::GetWorseStudent(vector<Student> \_dbStudent)

{

double maxMiddleRate = 100;

int index = -1;

for (int i = 0; i < \_dbStudent.size(); i++)

if (\_dbStudent[i].GetMiddleRate() < maxMiddleRate)

{

maxMiddleRate = \_dbStudent[i].GetMiddleRate();

index = i;

}

return \_dbStudent[index];

}

OrderBy – сортує студентів від меншого до гіршого і навпаки

vector<Student> Utils::OrderBy(vector<Student> \_dbStudent, bool reverse)

{

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

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

if (reverse)

{

OrderByTraining – сортує студентів від гіршого до кращого по вказаному предмету

vector<Student> Utils::OrderByTraining(vector<Student> \_dbStudent, Trainings trainingIndex, bool reverse)

{

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

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

if (reverse)

{

GetStudents – повертає студентів які здали або нездали сесію

vector<Student> Utils::GetStudents(vector<Student> \_dbStudent, double minMiddleRate, bool sessionComplete)

{

return (sessionComplete) ?

GetStudetsMoreFromRate(\_dbStudent, minMiddleRate) :

GetStudetsLessThanRate(\_dbStudent, minMiddleRate);

}

GetStudentsMoreFromRate – вертає студентів, чий середній рейтинг більший-рівний вказаному

vector<Student> Utils::GetStudetsMoreFromRate(vector<Student> \_dbStudent, double middleRate)

{

vector<Student>resDB\_students = vector<Student>();

for (int i = 0; i < \_dbStudent.size(); i++)

if (\_dbStudent[i].GetMiddleRate() >= middleRate)

resDB\_students.push\_back(\_dbStudent[i]);

return resDB\_students;

}

GetStudentsLessThan – вертає студентів, чий середній рейтинг менший ніж вказаний

vector<Student> Utils::GetStudetsLessThanRate(vector<Student> \_dbStudents, double middleRate)

{

vector<Student>resDB\_students = vector<Student>();

for (int i = 0; i < \_dbStudents.size(); i++)

if (\_dbStudents[i].GetMiddleRate() < middleRate)

resDB\_students.push\_back(\_dbStudents[i]);

return resDB\_students;

}

GetStudentsMoreFromRatebyTraining – повертає студентів, чий середній рейтинг більший-рівний вказаному рейтингу по заданому предмету

vector<Student> Utils::getStudentsMoreFrom\_byTraining(vector<Student> \_dbStudent, int rate, Trainings indexOfTraining)

{

vector<Student>resDB\_students = vector<Student>();

for (int i = 0; i < \_dbStudent.size(); i++)

if (\_dbStudent[i].GetRate()[(int)indexOfTraining] >= rate)

resDB\_students.push\_back(\_dbStudent[i]);

return resDB\_students;

GetStudentsMoreLessThanbyTraining – повертає студентів, чий середній рейтинг менший за вказаний рейтинг по заданому предету

vector<Student> Utils::getStudentsLessThan\_byTraining(vector<Student> \_dbStudents, int rate, Trainings indexOfTraining)

{

vector<Student>resDB\_students = vector<Student>();

for (int i = 0; i < \_dbStudents.size(); i++)

if (\_dbStudents[i].GetRate()[(int)indexOfTraining] < rate)

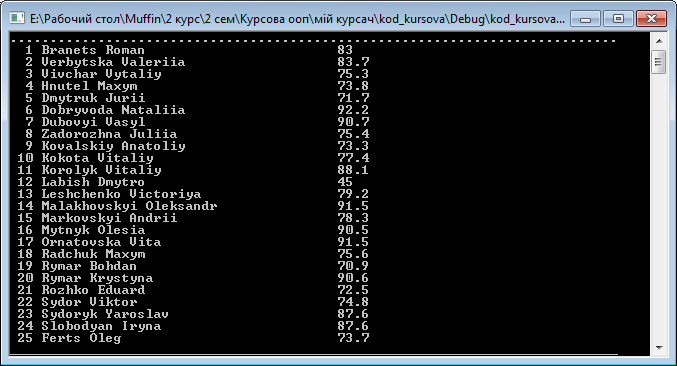
resDB\_students.push\_back(\_dbStudents[i]);

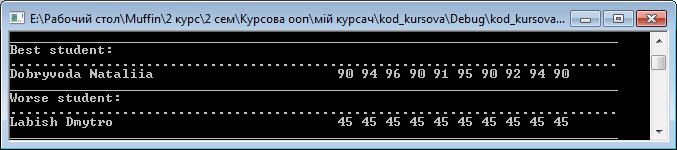
return resDB\_students;

}

# ТЕСТУВАННЯ ПРОГРАМИ І РЕЗУЛЬТАТИ ЇЇ ВИКОНАННЯ

При запуску програми відображається вікно на Рис. 4.1.

Рис. 4.1 Виведення середнього балу студента

Рис. 4.2 Виведення кращого та гіршого студента по балах

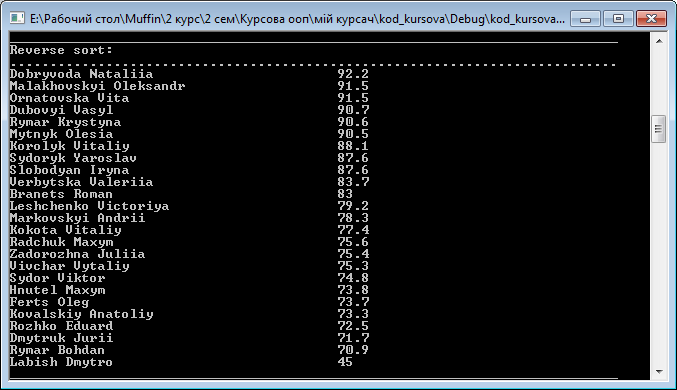


Рис. 4.3 Сортування студентів по середньому

балу від кращого до гіршого

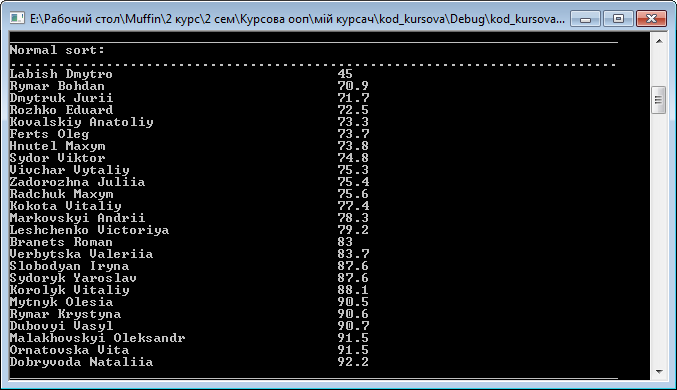


Рис. 4.4 Сортування студентів по середньому

балу від гіршого до кращого

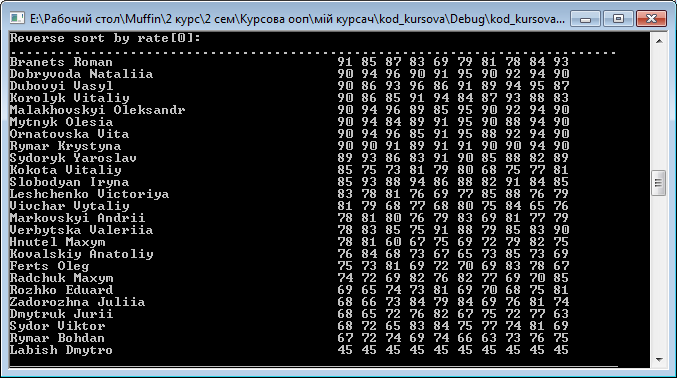


Рис. 4.5 Сортування студентів по балах з вказаного предмета

від кращого до гіршого

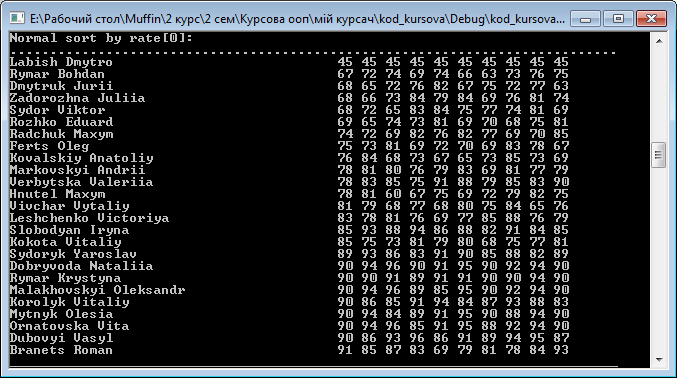


Рис. 4.6 Сортування студентів по балах з вказаного предмета

від гіршого до кращого

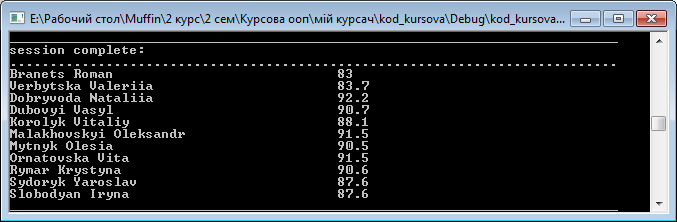


Рис. 4.7 Студенти які здали сесію

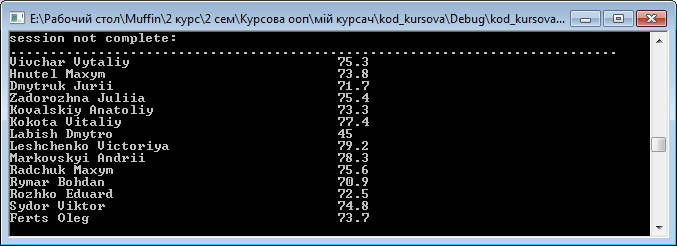


Рис. 4.8 Студенти котрі не здали сесії

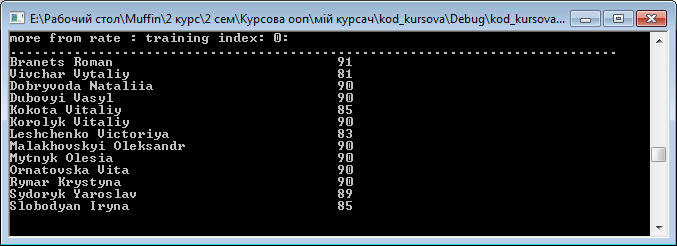
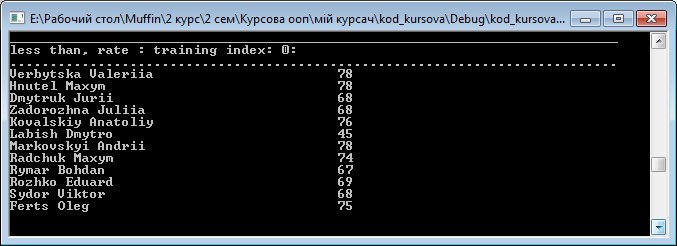


Рис. 4.9 Студенти в яких бал вищий вказаного по вказаному предмету



*Рис. 4.10 Студенти в яких бал нижчий вказаного по вказаному предмету*

# ВИСНОВОК

В результаті виконання роботи було розроблено програму для обробки результату сесії студентів ІІ-го курсу, вона створена з використанням компілятора Microsoft Visual Studio 2010, на мові програмування С++.

Дана програма надає користувачеві можливість легко і зручно обробляти і зберігати дані про успішність студентів. Завдяки інтуїтивно зрозумілому інтерфейсу будь-який користувач не зіткнеться з жодними проблемами при її використанні. у якій користувач створити власний розклад або відкрити вже створений раніше. Було використано мову програмування С++.

Організовано роботу з файлами: створення файлу, зчитування файлу, зміни файлу та вивід даних на екран.

# ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Орлов С. А. Технологии разработки программного обеспечения: Учебник. - СПб.: Питер, 2002. – 464 с. ISBN: 5-94723-145-X(рус.).
2. Прата С. Язык программирования С++. Лекции и упражнения,  
   6-е изд. / Стивен Прата : Пер. с англ. – М.: ООО “И.Д. Вильямс”, 2012. – 1248 с. ISBN 978-5-8459-1778-2 (рус.).
3. Страуструп Б. Язык программирования С++: Специальное издание. / Бьерн Страуструп. Пер. с англ. – М.: Издательство Бином, 2011. – 1136 с. ISBN 978-5-7989-0425-9 (рус.).
4. Дейтел Х.М. Как программировать на С++: 5-е издание. /  
    Х.М. Дейтел, П. Дж. Дейтел : Пер. с англ. – М.: ООО «Бином-Пресс», 2008. – 1456 с. ISBN 978-5-9518-0224-8 (рус.)

# ДОДАТОК А

Лістинг Source

#include<fstream>

#include"utils.h"

#include"Student.h"

#include"parser.h"

#include"global.h"

#include"loader.h"

using namespace std;

int main()

{

vector<Student> tmpArray = Loader::LoadStudentsDB("d:\\students.txt"); //диск D,

string tmpLine = "=================================================\n";

// вивід всіх студентів і їхнього середнього рейтингу.

cout<< tmpLine;

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

{

cout<<i+1<<": "<<" middleRate = "<<tmpArray[i].GetMiddleRate()<<'\n';

tmpArray[i].Show();

}

//кращий студент

cout<< tmpLine;

cout<<"Best student: \n";

cout<< tmpLine;

Utils::GetBestStudent(tmpArray).Show();

//гірший студент

cout<< tmpLine;

cout<<"Worse student: \n";

cout<< tmpLine;

Utils::GetWorseStudent(tmpArray).Show();

//відсортовані від гіршого до кращого

cout<< tmpLine;

cout<<"Normal sort: \n"<<endl;

cout<< tmpLine;

vector<Student> sortedArray1 = Utils::OrderBy(tmpArray);

for(int i=0;i<sortedArray1.size();i++)

cout<<sortedArray1[i].Name()<<" : "<<sortedArray1[i].GetMiddleRate()<<'\n';

//відсортовані від кращого до гіршого

cout<< tmpLine;

cout<<"Reverse sort: \n";

cout<< tmpLine;

vector<Student> sortedArray2 = Utils::OrderBy(tmpArray, true);

for(int i=0;i<sortedArray2.size();i++)

cout<<sortedArray2[i].Name()<<" : "<<sortedArray2[i].GetMiddleRate()<<'\n';

//Сортування від гіршого до кращого по балу з Electrical.

cout<< tmpLine;

cout<<"Normal sort by rate[0]: "<<endl;

cout<< tmpLine;

vector<Student> sortedArray3 = Utils::OrderByTraining(tmpArray,Utils::Electrical);

for(int i=0;i<sortedArray3.size();i++)

sortedArray3[i].Show();

//Сортування від кращого до гіршого по балу з Electrical.

cout<< tmpLine;

cout<<"Reverse sort by rate[0]: "<<endl;

cout<< tmpLine;

vector<Student> sortedArray4 = Utils::OrderByTraining(tmpArray,Utils::Electrical,true);

for(int i=0;i<sortedArray4.size();i++)

sortedArray4[i].Show();

//отримання всіх студентів, які здали сесію.

vector<Student> sesCompl = Utils::GetStudents(tmpArray, 80, true);

cout<< tmpLine;

cout<<"session complete: \n"<<endl;

cout<< tmpLine;

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

cout<<sesCompl[i].Name()<<" "<<sesCompl[i].GetMiddleRate()<<endl;

//отримання студентів, які не здали сесію.

vector<Student> sesNonCompl = Utils::GetStudents(tmpArray, 80, false);

cout<< tmpLine;

cout<<"session not complete: \n"<<endl;

cout<< tmpLine;

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

cout<<sesNonCompl[i].Name()<<" "<<sesNonCompl[i].GetMiddleRate()<<endl;

cout<<"\n\n";

//отримання списку студентів, чий бал по предмету Utils::Electrical більший від вказаного 80

vector<Student> stdMFRBT = Utils::getStudentsMoreFrom\_byTraining(tmpArray, 80,Utils::Electrical);

cout<< tmpLine;

cout<<"more from rate : training index: "<< (int)Utils::Electrical <<": \n"<<endl;

cout<< tmpLine;

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

cout<<stdMFRBT[i].Name()<<" "<<stdMFRBT[i].GetRateOfTraining((int)Utils::Electrical)<<endl;

cout<<"\n\n";

//отримання списку студентів, чий бал по предмету Utils::Electrical менший від вказаного 80

vector<Student> stdLSBT = Utils::getStudentsLessThan\_byTraining(tmpArray, 80,Utils::Electrical);

cout<< tmpLine;

cout<<"less than, rate : training index: "<< Utils::Electrical <<": \n"<<endl;

cout<< tmpLine;

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

cout<<stdLSBT[i].Name()<<" "<<stdLSBT[i].GetRateOfTraining((int)Utils::Electrical)<<endl;

cout<<"\n\n";

return 0;

}

Лістинг class Student

#include"Student.h"

#include<string>

using namespace std;

//створення нового студента (конструктор)

Student::Student(string \_name, vector<int> \_rate)

{

this->name = \_name;

this->rate = \_rate;

}

//вивід рейтингу по вказаному предмету

int Student::GetRateOfTraining(int trainingIndex)

{

return (trainingIndex < rate.size()) ? this->rate[trainingIndex] : -1;

}

//вивід середнього балу по всім предметам

double Student::GetMiddleRate()

{

double tmpMiddleRate = 0;

for (int i = 0; i < this->rate.size(); i++)

tmpMiddleRate += ((double)this->rate[i]);

return tmpMiddleRate /= ((double)(this->rate.size()));

}

//вивід на екран інформації про студента

void Student::Show()

{

cout << this->name << endl;

for (int i = 0; i < this->rate.size(); i++)

cout << rate[i] << ' ';

cout << endl;

}

//вивід імені студента

string Student::Name()

{

return this->name;

}

//вивід списку оцінок студента

vector<int> Student::GetRate()

{

return this->rate;

}

Лістинг class Utils

#include"utils.h"

//вивід кращого студента (по середньому балу)

Student Utils::GetBestStudent(vector<Student> \_dbStudent)

{

double minMiddleRate = 0;

int index = -1;

for (int i = 0; i < \_dbStudent.size(); i++)

if (\_dbStudent[i].GetMiddleRate() > minMiddleRate)

{//якщо в студента середній бал більший ніж максимальний середній бал - запам'ятовуємо його індекс

//і змінюємо максимальний бал на бал цього студента.

minMiddleRate = \_dbStudent[i].GetMiddleRate();

index = i;

}

return \_dbStudent[index]; //повертаємо студента, який має найбільший бал.

}

//вивід гіршого студента (по середньому балу)

Student Utils::GetWorseStudent(vector<Student> \_dbStudent)

{

double maxMiddleRate = 100;

int index = -1;

for (int i = 0; i < \_dbStudent.size(); i++)

if (\_dbStudent[i].GetMiddleRate() < maxMiddleRate)

{

//якщо в студента середній бал менший ніж найменший середній бал - запам'ятовуємо його індекс

//і змінюємо мінімальний бал на бал цього студента

maxMiddleRate = \_dbStudent[i].GetMiddleRate();

index = i;

}

return \_dbStudent[index];

}

vector<Student> Utils::OrderBy(vector<Student> \_dbStudent, bool reverse)

{

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

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

if (reverse)

{

if (\_dbStudent[j].GetMiddleRate() > \_dbStudent[i].GetMiddleRate())

{

Student tmp = \_dbStudent[i];

\_dbStudent[i] = \_dbStudent[j];

\_dbStudent[j] = tmp;

}

}

else

{

if (\_dbStudent[j].GetMiddleRate() < \_dbStudent[i].GetMiddleRate())

{

Student tmp = \_dbStudent[i];

\_dbStudent[i] = \_dbStudent[j];

\_dbStudent[j] = tmp;

}

}

return \_dbStudent;

}

//сортування студентів від гіршого до кращого по вказаному предмету

vector<Student> Utils::OrderByTraining(vector<Student> \_dbStudent, Trainings trainingIndex, bool reverse)

{

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

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

if (reverse)

{

//порівнює бали студентів по вказаному trainingIndex

if (\_dbStudent[j].GetRateOfTraining(trainingIndex) > \_dbStudent[i].GetRateOfTraining(trainingIndex))

{

Student tmp = \_dbStudent[i];

\_dbStudent[i] = \_dbStudent[j];

\_dbStudent[j] = tmp;

}

}

else

{

if (\_dbStudent[j].GetRateOfTraining(trainingIndex) < \_dbStudent[i].GetRateOfTraining(trainingIndex))

{

Student tmp = \_dbStudent[i];

\_dbStudent[i] = \_dbStudent[j];

\_dbStudent[j] = tmp;

}

}

return \_dbStudent;

}

//сортування студентів від кращого до гіршого по вказаному предмету

vector<Student> Utils::GetStudents(vector<Student> \_dbStudent, double minMiddleRate, bool sessionComplete)

{

return (sessionComplete) ?

GetStudetsMoreFromRate(\_dbStudent, minMiddleRate) :

GetStudetsLessThanRate(\_dbStudent, minMiddleRate);

}

//всі студенти, чий середній рейтинг більший-рівний вказаному

vector<Student> Utils::GetStudetsMoreFromRate(vector<Student> \_dbStudent, double middleRate)

{

vector<Student>resDB\_students = vector<Student>();

for (int i = 0; i < \_dbStudent.size(); i++)

if (\_dbStudent[i].GetMiddleRate() >= middleRate) // студент в якого середні бал більше рівне вказаного додається в базу

resDB\_students.push\_back(\_dbStudent[i]);

return resDB\_students;

}

//всі студенти, чий середній рейтинг меньший ніж вказаний

vector<Student> Utils::GetStudetsLessThanRate(vector<Student> \_dbStudents, double middleRate)

{

vector<Student>resDB\_students = vector<Student>();

for (int i = 0; i < \_dbStudents.size(); i++)

if (\_dbStudents[i].GetMiddleRate() < middleRate)

resDB\_students.push\_back(\_dbStudents[i]);

return resDB\_students;

}

//всі студенти, чий середній рейтинг більший-рівний вказаному рейтингу по заданому предету.

vector<Student> Utils::getStudentsMoreFrom\_byTraining(vector<Student> \_dbStudent, int rate, Trainings indexOfTraining)

{

vector<Student>resDB\_students = vector<Student>();

for (int i = 0; i < \_dbStudent.size(); i++)

if (\_dbStudent[i].GetRate()[(int)indexOfTraining] >= rate)

resDB\_students.push\_back(\_dbStudent[i]);

return resDB\_students;

}

//всі студенти, чий середній рейтинг меньший за вказаний рейтинг по заданому предету.

vector<Student> Utils::getStudentsLessThan\_byTraining(vector<Student> \_dbStudents, int rate, Trainings indexOfTraining)

{

vector<Student>resDB\_students = vector<Student>();

for (int i = 0; i < \_dbStudents.size(); i++)

if (\_dbStudents[i].GetRate()[(int)indexOfTraining] < rate)

resDB\_students.push\_back(\_dbStudents[i]);

return resDB\_students;

}

Лістинг class Parser

#include"parser.h"

#include<iostream>

#include<string>

#include<sstream>

using namespace std;

string Parser::GetName(string sourceLine)

{

//istringstream(string("11")) » k;

char\* tmpName = new char[sourceLine.length()];

int nameLength = 0;

//аналізуємо символи, і якщо це букви або пробіли - значить це може бути ім'ям

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

if (

(sourceLine[i] >= 'A' && sourceLine[i] <= 'Z') ||

(sourceLine[i] >= 'a' && sourceLine[i] <= 'z') || sourceLine[i] == ' ')

{

tmpName[nameLength++] = sourceLine[i];

}

else

break;

tmpName[nameLength] = '\0';

for (int i = nameLength - 1; i >= 0; i--)

if (tmpName[i] == ' ')

tmpName[i] = '\0';

else

break;

return string(tmpName);

}

vector<int> Parser::GetRate(string sourceLine)

{

sourceLine += ".";

//масив оцінок у вигляді стрічок, для подальшого парсингу в інти

vector<string> strRate = vector<string>();

//це потік, який знає як перетворити стрічку в число.

stringstream ss;

//тимчасова стрічка з оцінкою

string tmpStrRate = "";

//змінна, яка вказує чи наразі відбувається "збірка" числа.

bool isNumb = false;

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

{

if (sourceLine[i] >= '0' && sourceLine[i] <= '9')

{

isNumb = true;

tmpStrRate += sourceLine[i];

}

else

{

if (isNumb)

{

//запам'ятовуємо стрічку, яка містить число.

strRate.push\_back(tmpStrRate);

//готуємось для зчитування нової стрічки-числа.

tmpStrRate = "";

isNumb = false;

}

}

}

//перетворюємо стрічку в число (string -> int)

vector<int> retRate = vector<int>();

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

{

int tmpNumb = 0;

ss<<strRate[i];

ss>>tmpNumb;

ss.clear();

//запам'ятовуємо число.

retRate.push\_back(tmpNumb);

}

return retRate;

}

Лістинг class Loader

#include"loader.h"

#include"parser.h"

vector<Student> Loader::LoadStudentsDB(string \_pathOfBD)

{

//масив стічок, в яких міститься ім'я і рейтинг.

vector<string> lines = vector<string>();

char\* tmpLine = new char[200];

ifstream\* DBfile = new ifstream(\_pathOfBD);

//зачитуємо стрічки з файлу імпровізованої БД.

while(!DBfile->eof()){

if(DBfile->getline(tmpLine,200))

lines.push\_back(string(tmpLine));

}

//закриваємо і занулюємо стрім

DBfile->close();

delete DBfile;

DBfile = NULL;

//масив студентів.

vector<Student> studentDB = vector<Student>();

//одночасно парсимо кожну стрічку, витягуючи ім'я і

//рейтинг студента і зразу створюємо студента + заносимо його в масив.

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

studentDB.push\_back(Student(Parser::GetName(lines[i]), Parser::GetRate(lines[i])));

//видаляємо масив стрічок

return studentDB;

}