**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**

**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**

**імені ІГОРЯ СІКОРСЬКОГО»**

**Інститут прикладного системного аналізу**

**Кафедра системного проектування**

**Звіт**

**про виконання лабораторної роботи №14**

**з дисципліни «Алгоритмізація та програмування»**

Виконав:  
студент I курсу, групи ДА-02

Рудік Андрій Миколайович  
Прийняв:

к.т.н., доцент Безносик О. Ю.

Київ – 2021

Завдання

Створити масив структур. Кожна структура складається з таких елементів: інститут, факультет. Для інституту задати назву та план прийому на перший курс. Для факультету задати список спеціальностей та прохідний бал. Створений масив записати до бінарного файлу. Передбачити можливість доповнити файл, замінити дані в файлі, видалити дані з файлу. Реалізувати запити: 1) скласти список спеціальностей по факультетах із вказаним прохідним балом; 2) впорядкувати масив за прохідним балом і планом прийому; 3) визначити інститут із вказаним середнім прохідним балом.

Код програми

main.cpp

#include <iostream>

#include <cstdlib>

#include <ctime>

#include <clocale>

#include <fstream>

#include <windows.h>

#include "LinkedList.h"

#include "Faculty.h"

#include "Speciality.h"

#include "Institute.h"

#include "InstitutesList.h"

using namespace std;

int main()

{

srand(time(NULL));

setlocale(LC\_ALL, "rus");

InstitutesList institutesList("data.dat");

while (true) {

system("cls");

cin.clear();

fflush(stdin);

cout << "Главное меню:" << endl

<< "1) Вывести список университетов" << endl

<< "2) Отсортировать по проходному баллу" << endl

<< "3) Найти специальности по проходному баллу" << endl

<< "4) Найти институт по проходному баллу" << endl

<< "5) Добавить случайный институт" << endl

<< "6) Удалить интститут" << endl

<< "7) Выход" << endl << endl;

int key;

cin >> key;

system("cls");

cin.clear();

fflush(stdin);

switch (key) {

case 1: {

cout << institutesList << endl;

break;

}

case 2: {

institutesList.sortByPassingScore();

cout << "Готово" << endl;

break;

}

case 3: {

cout << "Введите промежуток на котором стоит искать специальности в формате \"от до\"" << endl;

float from, to;

cin >> from >> to;

cout << endl;

cout << institutesList.getSpecialitiesInRange(from, to);

break;

}

case 4: {

cout << "Введите средний проходной балл по институту чтобы найти подходящий" << endl;

float passingScore;

cin >> passingScore;

cout << endl;

try {

cout << institutesList.findByPassingScore(passingScore);

}

catch (const char\* e) {

cout << e;

}

break;

}

case 5: {

institutesList.append(Institute());

cout << "Готово" << endl;

break;

}

case 6: {

cout << "Введите номер института который желаете удалить" << endl;

int idx;

cin >> idx;

try {

institutesList.pop(idx - 1);

cout << "Готово" << endl;

}

catch (const char\* e) {

cout << e << endl;

}

break;

}

case 7: {

return 0;

break;

}

default: {

cout << endl << "Попробуйте еще раз" << endl;

break;

}

}

system("pause");

}

return 0;

}

stringToBinary.h

#pragma once

#include <cstring>

#include <fstream>

using namespace std;

void swrite(ofstream& fout, const string& str);

void sread(ifstream& fin, string& str);

stringToBinary.cpp

#include "stringToBinary.h"

void swrite(ofstream& fout, const string& str)

{

size\_t length = str.length() + 1;

fout.write((char\*)&length, sizeof(length));

fout.write((char\*)str.c\_str(), length);

}

void sread(ifstream& fin, string& str)

{

size\_t length;

fin.read((char\*)&length, sizeof(length));

char\* buffer = new char[length];

fin.read(buffer, length);

str = buffer;

delete[] buffer;

}

Speciality.h

#pragma once

#include <string>

#include "LinkedList.h"

struct Speciality

{

uint16\_t code;

std::string name;

Speciality();

Speciality(const uint16\_t&, const std::string&);

void write(std::ofstream& fout) const;

void read(std::ifstream& fin);

friend std::ostream& operator<< (std::ostream&, const Speciality&);

friend std::ostream& operator<< (std::ostream&, const LinkedList<std::pair<Speciality, float>>&);

};

Speciality.cpp

#include "Speciality.h"

#include <iostream>

#include <iomanip>

#include <fstream>

#include "stringToBinary.h"

#define NUMBER\_OF\_SPECIALITIES 12

using namespace std;

const pair<uint16\_t, string> SPECIALITIES[NUMBER\_OF\_SPECIALITIES] = {

{11, "Науки про освiту"},

{12, "Дошкiльна освiта"},

{13, "Початкова освiта"},

{14, "Середня освiта"},

{15, "Професiйна освiта"},

{16, "Спецiальна освiта"},

{17, "Фiзична культура i спорт"},

{121, "Iнженерiя програмного забезпечення"},

{122, "Комп’ютернi науки та iнформацiйнi технологiї"},

{123, "Комп’ютерна iнженерiя"},

{124, "Системний аналiз"},

{125, "Кiбербезпека"},

};

Speciality::Speciality() {

const pair<uint16\_t, string> p = SPECIALITIES[rand() % NUMBER\_OF\_SPECIALITIES];

code = p.first;

name = p.second;

}

Speciality::Speciality(const uint16\_t& acode, const string& aname) {

code = acode;

name = aname;

}

void Speciality::write(ofstream& fout) const

{

fout.write((char\*)&code, sizeof(code));

swrite(fout, name);

}

void Speciality::read(std::ifstream& fin)

{

fin.read((char\*)&code, sizeof(code));

sread(fin, name);

}

ostream& operator<<(ostream &out, const Speciality &spec) {

out << setw(3) << setfill('0') << spec.code << " " << spec.name;

return out;

}

ostream& operator<<(ostream& out, const LinkedList<pair<Speciality, float>>& specs)

{

int maxNameWidth = 0;

for (const pair<Speciality, float>& spec : specs) {

if (spec.first.name.size() > maxNameWidth) maxNameWidth = spec.first.name.size();

}

for (const pair<Speciality, float>& spec : specs) {

out << " " << setw(3) << setfill('0') << spec.first.code << " ";

out << setw(maxNameWidth) << setfill(' ') << left << spec.first.name

<< " : " << spec.second << endl;

out << right;

}

return out;

}

Faculty.h

#pragma once

#include <string>

#include "LinkedList.h"

#include "Speciality.h"

struct Faculty

{

std::string name;

LinkedList<std::pair<Speciality, float>> specialities;

Faculty();

void write(std::ofstream& fout) const;

void read(std::ifstream& fin);

float getAveragePassingScore() const;

friend std::ostream& operator<< (std::ostream&, const Faculty&);

};

Faculty.cpp

#include "Faculty.h"

#include <iostream>

#include <iomanip>

#include "Speciality.h"

#include "stringToBinary.h"

#define NUMBER\_OF\_FACULTIES 10

using namespace std;

const string FACULTIES[NUMBER\_OF\_FACULTIES] = {

"ИПСА",

"ФИОТ",

"ФЭЛ",

"ТЭФ",

"ФСП",

"ХТФ",

"ФПМ",

"КВП",

"ИАТ",

"ФТИ",

};

Faculty::Faculty() {

name = FACULTIES[rand() % NUMBER\_OF\_FACULTIES];

short numberOfSpecialities = 1 + (rand() % 5);

for (short i = 0; i < numberOfSpecialities; i++) {

specialities.pushBack({ Speciality(), 120 + (rand() % 81) });

}

}

void Faculty::write(std::ofstream& fout) const

{

swrite(fout, name);

size\_t size = specialities.size();

fout.write((char\*)&size, sizeof(size));

for (const pair<Speciality, float>& p : specialities) {

p.first.write(fout);

fout.write((char\*)&p.second, sizeof(p.second));

}

}

void Faculty::read(std::ifstream& fin)

{

sread(fin, name);

size\_t size;

fin.read((char\*)&size, sizeof(size));

specialities.clear();

for (size\_t i = 0; i < size; i++) {

pair<Speciality, float> p;

p.first.read(fin);

fin.read((char\*)&p.second, sizeof(p.second));

specialities.pushBack(p);

}

}

float Faculty::getAveragePassingScore() const

{

if (!specialities.size()) return 0;

float accumulatedPassingScore = 0;

for (const pair<Speciality, float>& spec : specialities) {

accumulatedPassingScore += spec.second;

}

return accumulatedPassingScore / specialities.size();

}

ostream& operator<<(ostream& out, const Faculty& fac) {

out << "Факультет — " << fac.name << endl

<< "Список специальностей с их проходными баллами:" << endl;

out << fac.specialities;

return out;

}

Institute.h

#pragma once

#include <string>

#include "Faculty.h"

#include "LinkedList.h"

struct Institute

{

std::string name;

std::string admissionPlan;

LinkedList<Faculty> faculties;

void write(std::ofstream& fout) const;

void read(std::ifstream& fin);

float getAveragePassingScore() const;

Institute();

friend std::ostream& operator<< (std::ostream&, const Institute&);

};

Institute.cpp

#include "Institute.h"

#include "Faculty.h"

#include <cstring>

#include <iostream>

#include "stringToBinary.h"

#define NUMBER\_OF\_INSTITUTE\_NAMES 8

#define NUMBER\_OF\_INSTITUTE\_PLANS 3

using namespace std;

const string INSTITUTE\_NAMES[NUMBER\_OF\_INSTITUTE\_NAMES] = {

"KПИ",

"УКУ",

"КНЕУ",

"ХНУРЕ",

"ХНУ",

"КНУТШ",

"НАУ",

"Могилянка",

};

const string INSTITUTE\_PLANS[NUMBER\_OF\_INSTITUTE\_PLANS] = {

"Делаем вид что нам не все равно.",

"Скидываем все на абит отдел там разберутся.",

"Кому вообще нужны перваши?",

};

void Institute::write(std::ofstream& fout) const

{

swrite(fout, name);

swrite(fout, admissionPlan);

size\_t size = faculties.size();

fout.write((char\*)&size, sizeof(size));

for (const Faculty& fac : faculties) {

fac.write(fout);

}

}

void Institute::read(std::ifstream& fin)

{

sread(fin, name);

sread(fin, admissionPlan);

size\_t size;

fin.read((char\*)&size, sizeof(size));

faculties.clear();

for (size\_t i = 0; i < size; i++) {

Faculty tmp;

tmp.read(fin);

faculties.pushBack(tmp);

}

}

float Institute::getAveragePassingScore() const

{

if (!faculties.size()) return 0;

float accumulatedPassingScore = 0;

for (const Faculty& fac : faculties) {

accumulatedPassingScore += fac.getAveragePassingScore();

}

return accumulatedPassingScore / faculties.size();

}

Institute::Institute()

{

name = INSTITUTE\_NAMES[rand() % NUMBER\_OF\_INSTITUTE\_NAMES];

admissionPlan = INSTITUTE\_PLANS[rand() % NUMBER\_OF\_INSTITUTE\_PLANS];

for (int i = 1 + (rand() % 4); i >= 0; i--) faculties.pushBack(Faculty());

}

std::ostream& operator<<(std::ostream& out, const Institute& institute)

{

out << "Институт — " << institute.name << endl;

out << "Список факультетов:" << endl << endl;

for (const Faculty& fac : institute.faculties) {

out << fac << endl;

}

return out;

}

InstitutesList.h

#pragma once

#include "Institute.h"

#include "Speciality.h"

#include "LinkedList.h"

class InstitutesList

{

public:

InstitutesList(std::string fname);

~InstitutesList();

void sortByPassingScore();

Institute findByPassingScore(float pScore) const;

LinkedList<std::pair<Speciality, float>> getSpecialitiesInRange(float from, float to);

Institute pop(size\_t idx);

void append(const Institute& institute);

friend std::ostream& operator<< (std::ostream& out, const InstitutesList& institutesList);

private:

void write(std::ofstream& fout);

void read(std::ifstream& fin);

LinkedList<Institute> m\_institutes;

std::string m\_fname;

};

InstitutesList.cpp

#include "InstitutesList.h"

#include "Institute.h"

#include <fstream>

using namespace std;

InstitutesList::InstitutesList(std::string fname)

{

m\_fname = fname;

ifstream fin(m\_fname);

if (fin.is\_open() && !fin.eof()) read(fin);

fin.close();

}

InstitutesList::~InstitutesList()

{

ofstream fout(m\_fname);

write(fout);

fout.close();

}

void InstitutesList::sortByPassingScore()

{

m\_institutes.sort(

[](const Institute& left, const Institute& right) {

return left.getAveragePassingScore() < right.getAveragePassingScore();

}

);

}

Institute InstitutesList::findByPassingScore(float pScore) const

{

for (const Institute& institute : m\_institutes) {

if (fabs(pScore - institute.getAveragePassingScore()) < 1e-5) {

return institute;

}

}

throw "There're no institute with the passing score";

}

LinkedList<pair<Speciality, float>> InstitutesList::getSpecialitiesInRange(float from, float to)

{

LinkedList<pair<Speciality, float>> specs;

for (const Institute& institute : m\_institutes) {

for (const Faculty& fac : institute.faculties) {

for (const pair<Speciality, float>& spec : fac.specialities) {

if (from <= spec.second && spec.second <= to)

specs.pushBack(spec);

}

}

}

return specs;

}

Institute InstitutesList::pop(size\_t idx)

{

return m\_institutes.pop(idx);

}

void InstitutesList::append(const Institute& institute)

{

m\_institutes.pushBack(institute);

}

void InstitutesList::write(ofstream& fout)

{

size\_t size = m\_institutes.size();

fout.write((char\*)&size, sizeof(size));

for (const Institute& institute : m\_institutes) {

institute.write(fout);

}

}

void InstitutesList::read(ifstream& fin)

{

size\_t size;

fin.read((char\*)&size, sizeof(size));

m\_institutes.clear();

for (size\_t i = 0; i < size; i++) {

Institute institute;

institute.read(fin);

m\_institutes.pushBack(institute);

}

}

ostream& operator<<(ostream& out, const InstitutesList& institutesList)

{

out << "Список институтов:" << endl << endl << "———————" << endl << endl;

for (const Institute& institute : institutesList.m\_institutes) {

out << institute << "———————" << endl << endl;

}

return out;

}

Результат роботи програми наведено на рисунках нижче

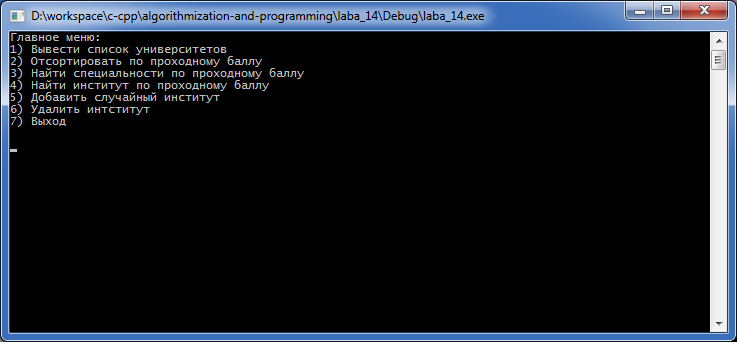


Рисунок 1 – Головне меню

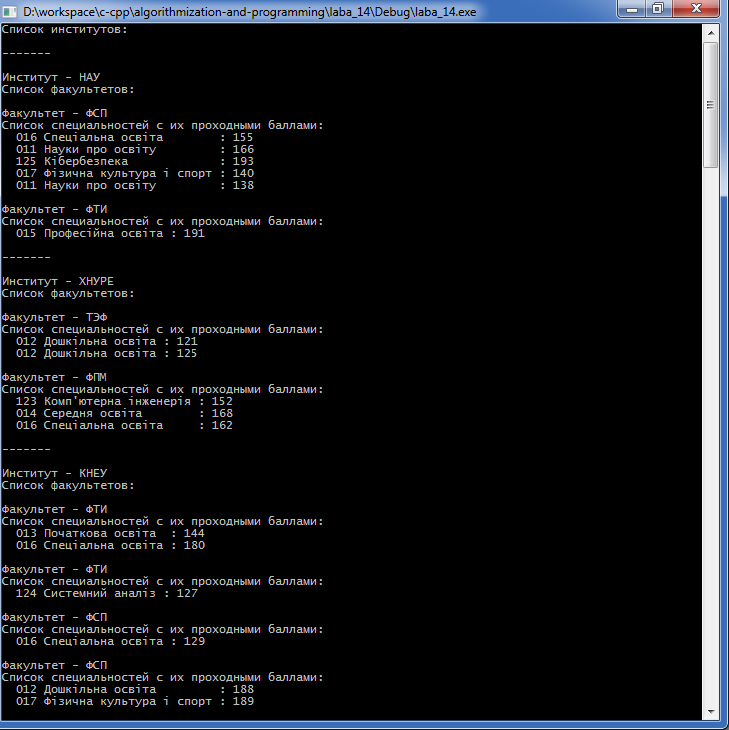


Рисунок 2 – Список інститутів

Висновки

В ході виконання лабораторної роботи №14 я вивчив особливості двійкових файлів користуючись мовою програмування С++.