

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**   
**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**   
**імені ІГОРЯ СІКОРСЬКОГО»**

**Інститут прикладного системного аналізу**   
**Кафедра системного проектування**  
   
   
 

**Звіт**

**про виконання лабораторної роботи №14**   
**«****Структури та бінарні файли»**

Виконав:   
студент I курсу, групи ДА-12   
Ніколаєв Роман Юрійович

Прийняв:

к.т.н., доцент Безносик О. Ю.

**Мета роботи**

Вивчити особливості використання структур, об’єднань та файлів, навчитися застосовувати структури, об’єднання, файли в програмуванні.

**Завдання**

Cтворити 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.

**Аналіз умови задачі.**

Відповідно до умови задачі, я створю вектор, який зберігає об’єкти структур, запишу його до бінарного файлу. Також створю методи для роботи з бінарним файлом та функції для виконання пунктів завдання.

**Код програми**

#include <iostream>

#include <vector>

#include <bits/stdc++.h>

typedef struct university{

char name[32];

int details;

} University;

typedef struct faculty{

char list[32];

int score;

} Faculty;

typedef struct strct{

University university;

Faculty faculty;

} Strct;

void log\_out(const std::string &str){

std::cout << "\033[33m" << "[LOG]" << str << "\033[00m" << std::endl;

}

void struct\_in(Strct &str){

std::cout << "(without spaces)" << std::endl;

std::cout << "- University name: ";

std::cin >> str.university.name;

std::cout << "- Admission details (int): ";

std::cin >> str.university.details;

std::cout << "- Specialisation list: ";

std::cin >> str.faculty.list;

std::cout << "- Faculty minimal score: ";

std::cin >> str.faculty.score;

}

void struct\_out(Strct &str){

std::cout << "- University name: " << str.university.name << std::endl;

std::cout << "- Admission details: " << str.university.details << std::endl;

std::cout << "- Specialisation list: " << str.faculty.list << std::endl;

std::cout << "- Faculty minimal score: " << str.faculty.score << std::endl;

}

void menu(){

log\_out("Menu:");

std::cout << "0: Exit." << std::endl;

std::cout << "1: Create an array and a new file using array data." << std::endl;

std::cout << "2: Append existing file." << std::endl;

std::cout << "3: Replace object in file." << std::endl;

std::cout << "4: Remove object in file." << std::endl;

std::cout << "5: Read data." << std::endl;

std::cout << "6: Call 1." << std::endl;

std::cout << "7: Call 2." << std::endl;

std::cout << "8: Call 3." << std::endl;

std::cout << "- Choose an option: ";

}

void create\_array(std::vector<Strct> &array, int &counter){

array.clear();

Strct str;

char answer;

while(answer != 'n'){

struct\_in(str);

array.push\_back(str);

counter++;

std::cout << "\033[33m" << "[LOG]" << "Continue?[y/n]: " << "\033[00m";

std::cin >> answer;

}

}

void new\_file(std::fstream &fl, const std::string &path, int &counter, std::vector<Strct> &array){

std::remove("file.b");

counter = array.size();

fl.open(path,std::ios\_base::out | std::ios::binary);

for(int i = 0; i < counter; i++){

fl.write((char\*)&array[i], sizeof(Strct));

}

fl.close();

log\_out("New file was created");

}

void append\_file(std::fstream &fl, const std::string &path, int &counter){

Strct str;

fl.open(path,std::ios\_base::app);

char answer;

log\_out("Append file with new objects: ");

while(answer != 'n'){

struct\_in(str);

fl.write((char\*)&str, sizeof(Strct));

counter++;

std::cout << "\033[33m" << "[LOG]" << "Continue?[y/n]: " << "\033[00m";

std::cin >> answer;

}

fl.close();

}

void replace\_object(std::fstream &fl, const std::string &path, const int &counter){

log\_out("Replacing object.");

Strct str, str\_to\_replace, str\_new;

int index = 0;

std::cout << "\033[33m" << "[LOG]Type needed index: \033[00m";

std::cin >> index;

if(index-1 >= counter || index-1 < 0){

log\_out("Out of range.");

fl.close();

return;

}

fl.open(path, std::ios\_base::in | std::ios::binary);

for(int i = 0; i < index; i++) fl.read((char\*)&str\_to\_replace, sizeof(Strct));

fl.close();

struct\_out(str\_to\_replace);

log\_out("Set new object: ");

struct\_in(str\_new);

std::fstream fl\_new;

fl.open(path, std::ios\_base::in | std::ios::binary);

fl\_new.open("new\_file.b", std::ios\_base::out | std::ios::binary);

for(int i = 0; i < counter; i++){

fl.read((char\*)&str, sizeof(Strct));

if(i == index-1){

fl\_new.write((char\*)&str\_new, sizeof(Strct));

} else fl\_new.write((char\*)&str, sizeof(Strct));

}

fl.close();

fl\_new.close();

std::remove("file.b");

std::rename("new\_file.b", "file.b");

}

void remove\_object(std::fstream &fl, const std::string &path, int &counter){

log\_out("Removing object.");

Strct str, str\_to\_remove;

int index = 0;

std::cout << "\033[33m" << "[LOG]Type needed index: \033[00m";

std::cin >> index;

if(index-1 >= counter || index-1 < 0){

log\_out("Out of range.");

fl.close();

return;

}

fl.open(path, std::ios\_base::in | std::ios::binary);

for(int i = 0; i < index; i++) fl.read((char\*)&str\_to\_remove, sizeof(Strct));

fl.close();

struct\_out(str\_to\_remove);

std::fstream fl\_new;

fl.open(path, std::ios\_base::in | std::ios::binary);

fl\_new.open("new\_file.b", std::ios\_base::out | std::ios::binary);

for(int i = 0; i < counter; i++){

fl.read((char\*)&str, sizeof(Strct));

if(i != index-1){

fl\_new.write((char\*)&str, sizeof(Strct));

}

}

counter--;

fl.close();

fl\_new.close();

std::remove("file.b");

std::rename("new\_file.b", "file.b");

}

void read\_file(std::fstream &fl, const std::string &path, int &counter){

Strct str;

fl.open(path, std::ios\_base::in | std::ios::binary);

log\_out("File data:");

for(int i = 0; i < counter; i++){

fl.read((char\*)&str, sizeof(Strct));

std::cout << "- Index: " << i+1 << std::endl;

struct\_out(str);

std::cout << "---------------------------------" << std::endl;

}

fl.close();

}

std::string call\_1(std::fstream &fl, const std::string &path, int &counter){

int score;

std::string result;

std::cout << "Enter minimal score: ";

std::cin >> score;

Strct str;

fl.open(path, std::ios\_base::in | std::ios::binary);

log\_out("Call results:");

for(int i = 0; i < counter; i++){

fl.read((char\*)&str, sizeof(Strct));

if(str.faculty.score >= score){

result.append(str.university.name);

result.append(": ");

result.append(str.faculty.list);

result.append("\n");

}

}

fl.close();

return result;

}

bool operator<(const Strct &a, const Strct &b){

if(a.faculty.score > b.faculty.score){

return true;

} else if(a.faculty.score < b.faculty.score){

return false;

} else if(a.university.details > b.university.details){

return true;

} else if(a.university.details < b.university.details){

return false;

} else return false;

}

void call\_2(std::fstream &fl, const std::string &path, int &counter){

std::vector<Strct> array;

Strct str;

fl.open(path, std::ios\_base::in | std::ios::binary);

for(int i = 0; i < counter; i++){

fl.read((char\*)&str,sizeof(Strct));

array.push\_back(str);

}

std::sort(array.begin(), array.end());

fl.close();

new\_file(fl, path, counter, array);

read\_file(fl, path, counter);

}

std::string call\_3(std::fstream &fl, const std::string &path, int &counter){

std::string result = "University name(s): ";

int score;

Strct str;

std::cout << "Enter the score: ";

std::cin >> score;

fl.open(path, std::ios\_base::in | std::ios::binary);

for(int i = 0; i < counter; i++){

fl.read((char\*)&str,sizeof(Strct));

if(str.faculty.score >= score){

result.append(str.university.name);

result.append(", ");

}

}

fl.close();

return result;

}

int main(){

std::remove("file.b");

int counter = 0;

const std::string path = "file.b";

std::fstream fl;

std::vector<Strct> array;

int choice = 1;

while(choice != 0){

menu();

std::cin >> choice;

switch (choice) {

case 1: {

create\_array(array, counter);

new\_file(fl, path, counter, array);

break;

}

case 2: {

append\_file(fl, path, counter);

break;

}

case 3: {

replace\_object(fl, path, counter);

break;

}

case 4: {

remove\_object(fl, path, counter);

break;

}

case 5: {

read\_file(fl, path, counter);

break;

}

case 6: {

std::cout << call\_1(fl, path, counter) << std::endl;

break;

}

case 7: {

call\_2(fl, path, counter);

break;

}

case 8: {

std::cout << call\_3(fl, path, counter) << std::endl;

break;

}

default: {

log\_out("Invalid option");

break;

}

}

}

return 0;

}

Результат виконання програми зображено на рисунках 1,2,3,4,5.

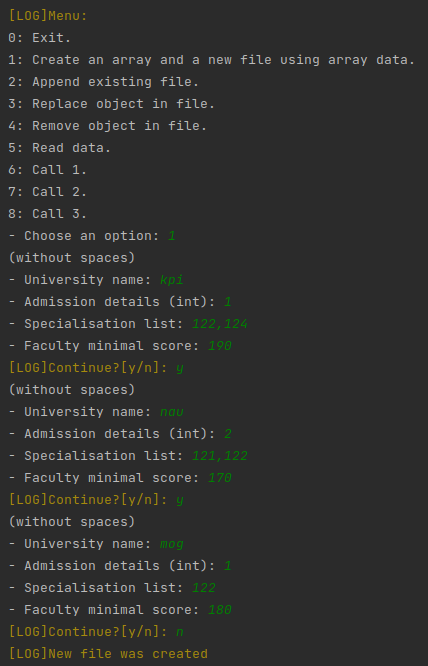


рисунок 1

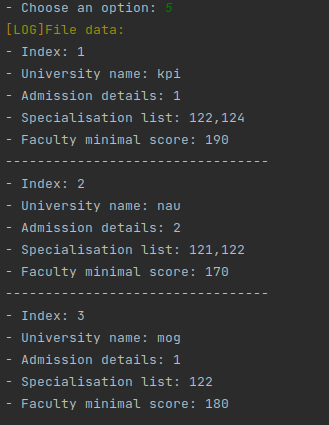


рисунок 2

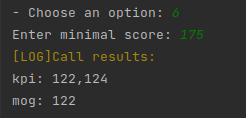


рисунок 3

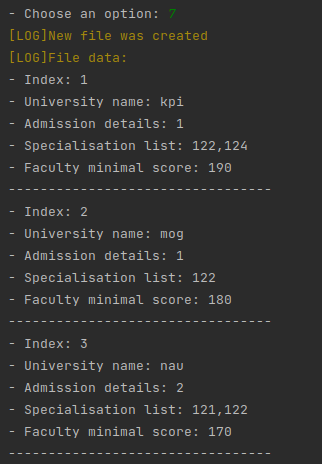


рисунок 4

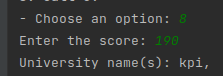


рисунок 5

**Виновки**

У ході виконання лабораторної роботи №14 я розробив інтерфейс для роботи з бінарними файлам. Інтерфейс засновано на типі данних std::fstream та методах open(), read(), write().