Міністерство освіти і науки України

ВСП “Фаховий коледж Чернівецького національного університету імені Юрія Федьковича”

**Звіт**

З навчальної практики

Виконав:

студент 314 групи

Ляшенко А.Р.

Спеціальність: Комп’ютерні науки

Чернівці, 2021

ЗМІСТ

Лабораторна робота №1

Тема: Розробка програм з використанням найпростіших класів та об’єктів.

Мета: Набуття навичок в розробці найпростіших класів та роботі з об’єктами класів.

Завдання:

. Розробити структуру класу для роботи з динамічними масивами, яка повинна включати не менше 2-х даних-членів класу та 5-х методів класу, з яких обов'язково повинні бути метод-член класу для:

* створення масиву;
* заповнення масиву;
* виводу змісту масиву на екран;
* очищення пам'яті.

2. Розробити метод-член класу для виводу на екран суми першої і другої половини одновимірного динамічного масиву і кількості позитивних елементів у кожній

3. Розробити 2-3 тести для перевірки вірності роботи даної програми.

4. Оформити звіт до лабораторної роботи.

Код програми:

1. array.h

*#pragma* *once*

*#include* <iostream>

*#include* <vector>

class array {

int size{};

int \*arr{};

void init(int data);

void memory(int size);

struct result {

int sum1, sum2, pos1, pos2;

};

public:

array() = default;

array(int size);

array(int size, int data);

void fill\_array();

class Iterator {

int \*ptr;

public:

using difference\_type = std::ptrdiff\_t;

Iterator(int \*a);

Iterator &operator++();

Iterator operator++(int);

bool operator!=(const Iterator &other);

bool operator==(const Iterator &other);

int operator\*() const;

Iterator operator+(int a);

};

Iterator begin();

Iterator end();

int &operator[](int i) const;

~array();

friend std::ostream &operator<<(std::ostream &out, const array &a);

result func();

void copy\_from\_vec(std::vector<int> &v);

};

1. array.cpp

*#include* "array.h"

*#include* <algorithm>

*#include* <ctime>

*#include* <exception>

*#include* <numeric>

void array::init(int data) {

*for* (int i = 0; i < size; ++i)

arr[i] = data;

}

void array::memory(int size) { arr = new int[size]; }

array::array(int size) : size(size) {

memory(size);

init(0);

}

array::array(int size, int data) : size(size) {

memory(size);

init(data);

}

void array::fill\_array() {

std::srand(static\_cast<unsigned>(std::time(0)));

*for* (auto i = 0; i < size; i++) {

arr[i] = rand() % 20 - 10;

}

}

array::Iterator::Iterator(int \*a) : ptr(a) {}

array::Iterator &array::Iterator::operator++() {

ptr++;

*return* \**this*;

}

array::Iterator array::Iterator::operator++(int) {

auto tmp = \**this*;

ptr++;

*return* tmp;

}

bool array::Iterator::operator!=(const Iterator &other) {

*return* ptr != other.ptr;

}

bool array::Iterator::operator==(const Iterator &other) {

*return* ptr == other.ptr;

}

int array::Iterator::operator\*() const { *return* \*ptr; }

array::Iterator array::Iterator::operator+(int a) {

auto tmp = ptr;

*for* (int i = 0; i < a; i++) {

tmp++;

}

*return* Iterator{tmp};

}

array::Iterator array::begin() { *return* Iterator{arr}; }

array::Iterator array::end() { *return* Iterator{arr + size}; }

int &array::operator[](int i) const {

*if* (i < 0 || i > size)

*throw* std::out\_of\_range("Вихід за межі масиву");

*return* \*(arr + i);

}

array::~array() { delete[] arr; }

std::ostream &operator<<(std::ostream &out, const array &a) {

*for* (int i = 0; i < a.size; i++) {

out << a.arr[i] << " ";

}

out << "\n";

*return* out;

}

array::result array::func() {

auto mid = size / 2;

auto beginIt = *this*->begin();

auto endIt = *this*->end();

auto midIt = *this*->begin() + mid;

auto positive = [](const int &t) { *return* t > 0; };

*// size\_t pos = std::count\_if(beginIt, midIt, positive);*

result a(std::accumulate(beginIt, midIt, 0), std::accumulate(midIt, endIt, 0),

std::count\_if(beginIt, midIt, positive),

std::count\_if(midIt, endIt, positive));

std::cout << "Сума першої половини = " << a.sum1

<< " Кількість позитивних = " << a.pos1 << std::endl;

std::cout << "Сума другої половини = " << a.sum2

<< " Кількість позитивних = " << a.pos2 << std::endl;

*return* a;

}

void array::copy\_from\_vec(std::vector<int> &v) {

std::copy(v.begin(), v.end(), arr);

}

1. test.cpp

*#include* "array.h"

*#include* <gtest/gtest.h>

*#include* <vector>

class ArrayTest : public ::testing::Test {};

TEST\_F(ArrayTest, size\_not\_odd) {

array a(4);

std::vector<int> b = {-1, 5, 3, 3};

a.copy\_from\_vec(b);

auto [sum1, sum2, pos1, pos2] = a.func();

EXPECT\_EQ(sum1, 4);

EXPECT\_EQ(sum2, 6);

EXPECT\_EQ(pos1, 1);

EXPECT\_EQ(pos2, 2);

}

TEST\_F(ArrayTest, size\_odd) {

array a(5);

std::vector<int> b = {-1, 5, 3, 3, -5};

a.copy\_from\_vec(b);

auto [sum1, sum2, pos1, pos2] = a.func();

EXPECT\_EQ(sum1, 4);

EXPECT\_EQ(sum2, 1);

EXPECT\_EQ(pos1, 1);

EXPECT\_EQ(pos2, 2);

}

Виконання:



Лабораторна робота №2

Тема: Розробка програм з використанням класів з конструктором і деструктором.

Мета: Набуття навиків в проектуванні найпростіших класів, розробка найпростіших програм їх використання.

Завдання:

1. Ознайомитися з теоретичними основами розробки та правилами використання конструктора і деструктора.
2. Розробити клас для роботи з текстовими файлами з декількома конструкторами та деструктором, з функціями наповнення файлу, визначення його розміру, функцією виведення змісту файлу та функцією відповідно до завдання 1.
3. Розробити метод-член класу для порівняння двох текстових файлів та виведення номеру рядка та позиції символу, де вони відрізняються.
4. Розробити клас для роботи з бінарними файлами з декількома конструкторами, деструктором, функціями наповнення файлу, визначення його розміру, функцією виведення змісту файлу та функцією відповідно до завдання 2 Розробити метод-член класу для дозапису заданої кількості даних у кінець бінарного файлу (файл повинен складатися не менше ніж з п’яти компонентів).

Код програми:

1. file.h

*#pragma* *once*

*#include* <fstream>

*#include* <ios>

class MyFile : public std::fstream {

public:

struct Position {

size\_t line{1};

size\_t c{0};

};

static const Position npos;

MyFile() = default;

explicit MyFile(const std::string &s,

std::ios\_base::openmode mode = std::ios\_base::in |

std::ios\_base::out);

~MyFile() = default;

Position compare(MyFile &other);

int size\_B();

void fill\_f(const char\* str,size\_t bytes);

void fill\_f(std::string &str);

void read\_f(char \*out, int size);

void read\_f(std::string &out);

void end\_put(const char \*, int size);

};

bool operator==(const MyFile::Position &rhs, const MyFile::Position &lhs);

bool operator!=(const MyFile::Position &rhs, const MyFile::Position &lhs);

1. file.cpp

*#include* "file.h"

const MyFile::Position MyFile::npos = {SIZE\_MAX, SIZE\_MAX};

bool operator==(const MyFile::Position &rhs, const MyFile::Position &lhs) {

*return* (rhs.c == lhs.c) && (lhs.line == rhs.line);

}

bool operator!=(const MyFile::Position &rhs, const MyFile::Position &lhs) {

*return* !(rhs == lhs);

}

MyFile::Position MyFile::compare(MyFile &other) {

Position diff{};

char ch1{}, ch2{};

*while* (!*this*->eof()) {

*this*->get(ch1);

other.get(ch2);

diff.c += 1;

*if* (ch1 != ch2)

*return* diff;

*if* (ch1 == '\n') {

diff.line += 1;

diff.c = 0;

}

}

*return* npos;

}

MyFile::MyFile(const std::string &s, std::ios\_base::openmode mode)

: std::fstream(s, mode) {}

int MyFile::size\_B(){

*this*->seekg(0,std::ios::end);

auto size = *this*->tellg();

*this*->seekg(0,std::ios::beg);

*return* size;

}

void MyFile::fill\_f(const char\* str, size\_t bytes)

{

\**this* << std::endl;

*this*->seekp(0, std::ios::end);

*this*->write(str, bytes);

}

void MyFile::fill\_f(std::string &str)

{

\**this* << std::endl;

*this*->seekp(0, std::ios::end);

*this*->write(str.c\_str(), str.size());

}

void MyFile::read\_f(char \*out, int size)

{

*this*->read(out, size);

}

void MyFile::read\_f(std::string &out)

{

char \*tmp = new char[size\_B()];

out.resize(size\_B());

*this*->read(tmp, size\_B());

out.assign(tmp);

}

void MyFile::end\_put(const char\* c, int size)

{

seekp(0, std::ios::end);

write(c, size);

}

1. files.cpp

*#include* "file.h"

*#include* <iostream>

int main() {

MyFile file1{"1.txt"};

MyFile file2{"2.txt"};

*if* ((!file1) || !file2) {

std::cerr << "Не можливо відкрити файл" << std::endl;

*return* EXIT\_FAILURE;

}

auto pos = file1.compare(file2);

std::string str;

file1.read\_f(str);

std::cout << str << std::endl;

file1.close();

file2.close();

*if* (pos == MyFile::npos) {

std::cout << "Файли однакові" << std::endl;

} *else* {

std::cout << "Відмінність у рядку - " << pos.line << ", позиції - " << pos.c

<< std::endl;

}

std::cout << "\nБінарні файли\n";

MyFile file3("3.dat");

char c[] = "abcdefg";

char c1[] = "hijklmn";

*// file3.fill\_f(c, 8);*

*// std::string tmp;*

*// file3.read\_f(tmp);*

*// std::cout << tmp << std::endl;*

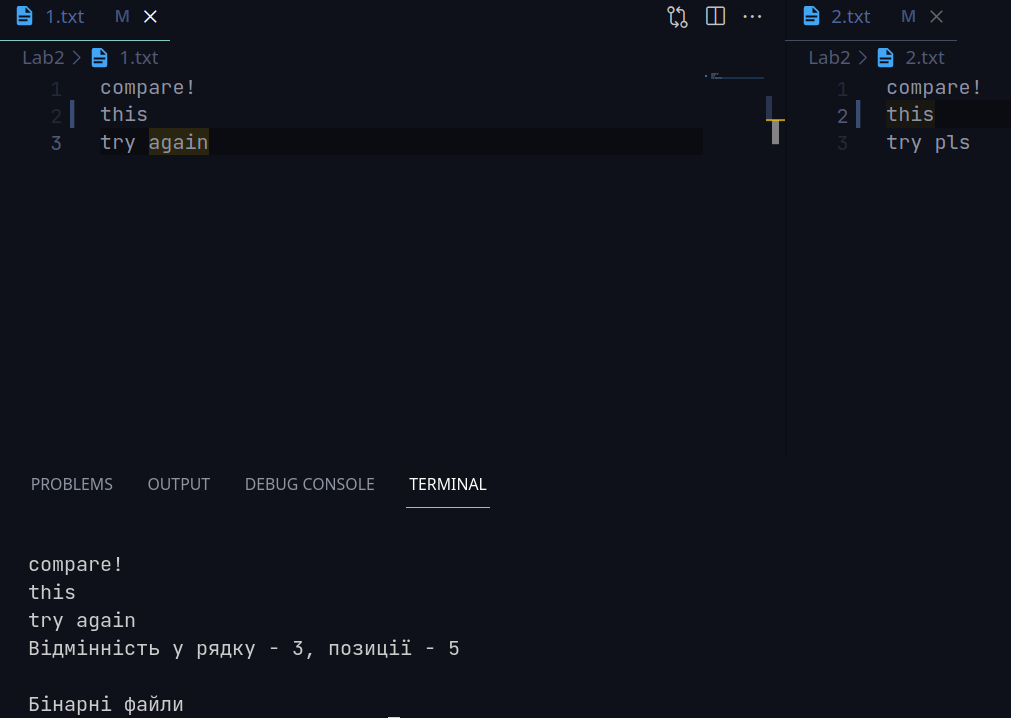
file3.end\_put(c1, 8);

file3.close();

*return* 0;

}

Виконання:



Лабораторна робота №3

Тема: Розробка ієрархії класів з використанням абстрактних класів і чистих віртуальних функцій. ( Лабораторну роботу, виконують студенти, які претендують на оцінку в діапозоні B-A)

Мета: Набуття навичок в розробці класів з використанням властивостей поліморфізму.

Завдання:

1. Ознайомитись з теоретичними основами розробки абстрактних класів.

Розробити ієрархію не менше, ніж 2 класів, та програму відповідно варіанту **(номер варіанту відповідає № студента в стиску підгрупи).**Написати клас для ефективної роботи з рядками, що дозволяє форматувати і порівнювати рядки, зберігати в рядках числові значення і “витягати ” їх. Для цього необхідно реалізувати:

- перевантажені оператори присвоювання і конкатенації;

- операції порівняння і приведення типів;

- перетворення в число будь-якого типу;

- форматний вивід рядка.

Написати програму, що демонструє роботу з цим класом. Програма повинна містити меню, що дозволяє здійснити перевірку всіх методів класу.

1. Розробити 2-3 тести для перевірки правильності роботи розробленої програми.
2. Оформити звіт до лабораторної роботи.

Код програми:

1. simple\_s.h

*#pragma* *once*

*#include* <iostream>

class String{

std::string str;

public:

String() = default;

String(std::string str);

String(int);

String(const String &a);

String(double);

String &operator=(const String &other);

String &operator+=(const String &other);

bool operator==(const String &other);

bool operator>(const String &other);

bool operator<(const String &other);

bool operator!=(const String &other);

friend std::istream &operator>>(std::istream &in, String &obj);

explicit operator int()

{

std::string tmp(str);

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

{

*try*{

auto a = std::stoi(tmp);

*return* a;

}*catch*(...)

{

auto len = tmp.size() - 1;

*for* (int i = 0; i < len; i++)

{

char c = tmp[i];

tmp[i] = tmp[i + 1];

tmp[i + 1] = c;

}

tmp.resize(len);

}

}

*return* EXIT\_FAILURE;

}

explicit operator double()

{

*return* atof(str.c\_str());

}

void format(int setw, char c );

void format\_print(int setw, char c );

};

1. simple\_h.cpp

*#include* "simple\_s.h"

*#include* <iomanip>

*#include* <string>

String& String::operator=(const String & other)

{

str = other.str;

*return* \**this*;

}

String& String::operator+=(const String & other)

{

str += other.str;

*return* \**this*;

}

String::String(const String &a)

{

str = a.str;

}

bool String::operator==(const String& other)

{

*return* other.str == str;

}

bool String::operator>(const String& other)

{

*return* str > other.str;

}

bool String::operator<(const String& other)

{

*return* str < other.str;

}

bool String::operator!=(const String &other)

{

*return* str != other.str;

}

String::String(int a)

{

str = a;

}

String::String(double a)

{

str = a;

}

void String::format(int setw, char c ){

std::string tmp(setw-str.length(), c);

str = tmp + str;

}

void String::format\_print(int setw, char c){

std::cout << std::setw(setw) << std::setfill(c) << str << std::endl;

}

std::istream& operator>>(std::istream &in, String &obj)

{

in >> obj.str;

*return* in;

}

1. main.cpp

*#include* <simple\_s.h>

*#include* <unistd.h>

void Menu();

int main(){

*for* (;;)

{

sleep(5);

system("clear");

std::cout << "\nВведіть текст: ";

String str;

std::cin >> str;

Menu();

int choice;

std::cin >> choice;

*switch* (choice)

{

*case* 1:

{

std::cout << "Інший текст: ";

String str2;

std::cin >> str2;

std::cout << "Текст 1 == інший текст " << std::boolalpha << (str==str2) << std::endl;

*break*;

}

*case* 2:

{

std::cout << static\_cast<int>(str);

*break*;

}

*case* 3:

{

std::cout << static\_cast<double>(str);

*break*;

}

*case* 4:

{

std::cout << " (int)<setw> (char)<setfill> \n";

int w;

char fill;

std::cin >> w >> fill;

str.format\_print(w,fill);

*break*;

}

*case* 5:

{

std::cout << "Інший текст: ";

String str2;

std::cin >> str2;

str += str2;

str.format\_print(10,' ');

*break*;

}

*case* 0:

*return* EXIT\_SUCCESS;

*break*;

*default*:

*break*;

}

}

*return* 0;

}

void Menu(){

std::cout << "1| Порівняти " << std::endl;

std::cout << "2| Витягнути число " << std::endl;

std::cout << "3| Привести до double " << std::endl;

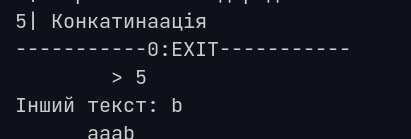
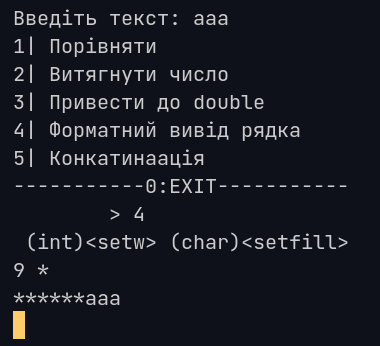
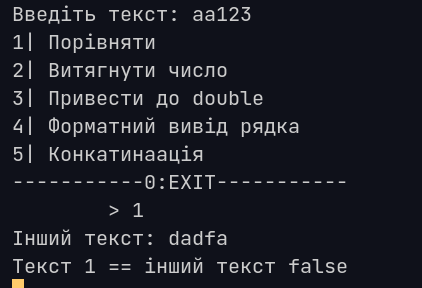
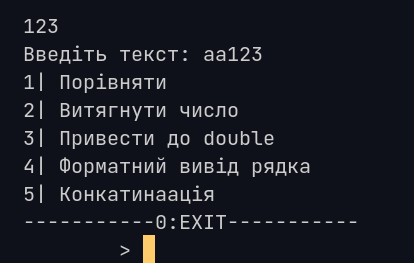
std::cout << "4| Форматний вивід рядка" << std::endl;

std::cout << "5| Конкатинаація " << std::endl;

std::cout << "-----------0:EXIT-----------\n\t> ";

}

Виконання:



Лабораторна робота №3

Тема:

Мета:

Завдання, код програми, скрін з тестами про успішне виконання і так по всім лабораторним роботам

Лабораторна робота №4

Тема: Особливості обробки виняткових ситуацій в мові С++.

Мета: Набуття навичок обробки виняткових ситуацій в професійних програмах.

Завдання:

1. Ознайомитись з теоретичними відомостями.

2. Розробити ієрархію класів для ефективної роботи з об’єктом, що заданий в завданні 1. Розробити не менш ніж дві функції для аналізу виняткових ситуацій при роботі з цим об’єктом.

Розробити ієрархію класів для реалізації «записної книжки». Передбачити можливість роботи з довільним числом записів, пошуку запису по якій-небудь ознаці (наприклад, по прізвищу чи номеру телефону), додавання і видалення записів, сортування по різних полях. Довжину рядків задавати динамічно. Розробити не менш ніж дві функції для аналізу виняткових ситуацій при роботі з файлами.

3. Написати програму, що демонструє роботу з побудованою ієрархією класів. Програма повинна містити меню, що дозволяє здійснити перевірку всіх методів класу. **(номер варіанту відповідає № студента в стиску підгрупи).**

4. Розробити не менше трьох ситуацій для тестування варіанту завдання 1. Провести тестування розробленого програмного забезпечення.

5. Скласти звіт за результатами лабораторної роботи.

6. Зробити висновки.

Код програми:

1. book.h

*#pragma* *once*

*#include* "person.h"

class book\_p : public person

{

std::string tel{};

public:

book\_p();

book\_p(std::string name, std::string surname, std::string tel);

void print() final;

std::string get\_tel() const;

void set\_tel(std::string t);

friend std::istream &operator>>(std::istream &in, book\_p &obj);

friend std::ostream &operator<<(std::ostream &out, const book\_p &obj);

bool operator==(const book\_p &other);

bool operator!=(const book\_p &other);

};

1. book.cpp

*#include* "book.h"

*#include* <iostream>

book\_p::book\_p() : person(){}

book\_p::book\_p(std::string name, std::string surname, std::string tel) : person(name,surname){

*this*->tel = tel;

}

void book\_p::print(){

person::print();

std::cout << tel << std::endl;

}

std::string book\_p::get\_tel() const { *return* tel; }

std::ostream &operator<<(std::ostream &out, const book\_p &obj)

{

out << obj.get\_name() << " " << obj.get\_surname() << " " << obj.get\_tel();

*return* out;

}

void book\_p::set\_tel(std::string t)

{

tel = t;

}

std::istream& operator>>(std::istream &in, book\_p &obj)

{

std::string name,surname,tel;

in >> name >> surname >> tel;

obj.set\_name\_surname(name, surname);

obj.set\_tel(tel);

*return* in;

}

bool book\_p::operator==(const book\_p &other)

{

*return* (get\_name() == other.get\_name() && get\_surname() == other.get\_surname() && tel == other.tel);

}

bool book\_p::operator!=(const book\_p&other)

{

*return* (get\_name() != other.get\_name() && get\_surname() != other.get\_surname() && tel != other.tel);

}

1. person.h

*#pragma* *once*

*#include* <string>

class person

{

std::string name, surname;

public:

person() = default;

person(std::string name, std::string surname);

void set\_name\_surname(std::string n, std::string s);

virtual void print();

std::string get\_name() const;

std::string get\_surname() const;

};

1. person.cpp

*#include* "person.h"

*#include* <iostream>

person::person(std::string name, std::string surname): name(name), surname(surname){}

void person::print(){

std::cout << name << " " << surname << " ";

}

std::string person::get\_name() const { *return* name; }

std::string person::get\_surname() const { *return* surname; }

void person::set\_name\_surname(std::string n,std::string s)

{

name = n;

surname = s;

}

1. register.h

*#pragma* *once*

*#include* "book.h"

*#include* <vector>

*#include* <iostream>

*#include* <fstream>

*#include* <exception>

*#include* <ranges>

class Register{

*// struct compare*

*// {*

*// bool operator()(book\_p &pers1,book\_p &pers2)*

*// {*

*// return pers1.get\_surname() < pers2.get\_surname() ||*

*// (pers1.get\_surname() == pers2.get\_surname() && pers1.get\_name() < pers2.get\_name());*

*// }*

*// };*

public:

std::vector<book\_p> r\_book;

void add\_person(const book\_p &a);

friend std::istream &operator>>(std::istream &in, Register &obj);

friend std::ostream &operator<<(std::ostream &out, Register &obj);

void push\_to\_file();

void load\_from\_file();

void search\_(std::string&);

void remove\_person(const book\_p &person);

void sort\_(const std::string &param);

};

1. register.cpp

*#include* "register.h"

void Register::add\_person(const book\_p &a)

{

*if*(r\_book.empty())

r\_book.emplace\_back(a);

*else* *if* (auto exists = std::find(r\_book.begin(), r\_book.end(), a); exists == std::end(r\_book))

r\_book.emplace\_back(a);

*if*(a.get\_name() == "" || a.get\_surname() == "" || a.get\_tel() =="")

*throw* std::logic\_error("пусте поле");

}

std::ostream& operator<<(std::ostream &out, Register &obj)

{

*for* (size\_t i = 0; i < obj.r\_book.size(); i++)

{

out << obj.r\_book[i];

*if*(i != obj.r\_book.size()-1) out << std::endl;

}

*return* out;

}

void Register::push\_to\_file()

{

std::fstream file("record\_book.txt");

file.clear();

file << \**this*;

file.close();

}

void Register::load\_from\_file()

{

std::fstream file("record\_book.txt");

*if*(!file.is\_open())

*throw* std::underflow\_error("Немає файлу");

*while* (!file.eof())

{

book\_p tmp;

file >> tmp;

*try*{

add\_person(tmp);

}*catch*(std::logic\_error &err)

{

std::cout << "Проблема файлу: пустий рядок " << err.what();

}

}

file.close();

}

void Register::search\_(std::string &search)

{

auto lambda = [&search](const book\_p &a)

{ *return* (a.get\_name() == search || a.get\_surname() == search || a.get\_tel() == search); };

*for* (const auto &el : r\_book | std::ranges::views::filter(lambda))

std::cout << el << std::endl;

}

void Register::remove\_person(const book\_p &person)

{

auto del = std::find(r\_book.begin(), r\_book.end(), person);

*if*(del != r\_book.end())

r\_book.erase(del);

*else*

*throw* std::logic\_error("Немає такого запису");

}

void Register::sort\_(const std::string &param){

*if*(param == "name"){

auto comp = [](const book\_p &person1, const book\_p &person2)

{ *return* (person1.get\_name()) <= person2.get\_name(); };

std::sort(r\_book.begin(), r\_book.end(), comp);

}

*else* *if*(param == "surname")

{

auto comp = [](const book\_p &person1, const book\_p &person2)

{ *return* (person1.get\_surname()) <= person2.get\_surname(); };

std::sort(r\_book.begin(), r\_book.end(), comp);

}

*else* *if*(param == "tel")

{

auto comp = [](const book\_p &person1, const book\_p &person2)

{ *return* (person1.get\_tel()) <= person2.get\_tel(); };

std::sort(r\_book.begin(), r\_book.end(), comp);

}

*else*

*throw* std::logic\_error("Параметр сортування хибний");

*for* (int i = 0; i < r\_book.size(); i++)

{

std::cout << r\_book[i] << std::endl;

}

}

1. main.cpp

*#include* "register.h"

*#include* "unistd.h"

void Menu();

void add\_new\_record();

void main\_search();

void delete\_record();

void main\_sort();

void quit();

Register rec;

int main()

{

rec.load\_from\_file();

*for* (;;)

{

sleep(5);

*// pause();*

system("clear");

Menu();

int choice;

std::cin >> choice;

*switch* (choice)

{

*case* 1:

add\_new\_record();

*break*;

*case* 2:

delete\_record();

*break*;

*case* 3:

main\_search();

*break*;

*case* 4:

main\_sort();

*break*;

*case* 0:

quit();

*break*;

*default*:

*break*;

}

}

*return* 0;

}

void add\_new\_record(){

rec.load\_from\_file();

std::string name, surname, tel;

std::cout << "Введіть ім'я прізвище та телефон: ";

std::cin >> name >> surname >> tel;

*try*{

rec.add\_person(book\_p{name, surname, tel});

}*catch*(std::logic\_error &err){

std::cout << "\nПоле пусте\n";

}

}

void main\_search(){

std::string key\_word;

std::cout << "Ключве слово: ";

std::cin >> key\_word;

rec.search\_(key\_word);

}

void delete\_record(){

std::string name, surname, tel;

rec.sort\_("name");

std::cout << "Введіть ім'я прізвище та телефон: ";

std::cin >> name >> surname >> tel;

*try*{

rec.remove\_person(book\_p{name,surname,tel});

}*catch*(std::logic\_error &err)

{

std::cout << err.what();

}

}

void main\_sort(){

std::string attribute;

std::cout << "введіть атрибут (name,surname,tel): ";

std::cin >> attribute;

*try*{

rec.sort\_(attribute);

}*catch*(std::logic\_error &err)

{

std::cout << err.what();

}

}

void quit(){

rec.push\_to\_file();

exit(EXIT\_SUCCESS);

}

void Menu(){

std::cout << "1| Додати новий запис " << std::endl;

std::cout << "2| Видалити запис " << std::endl;

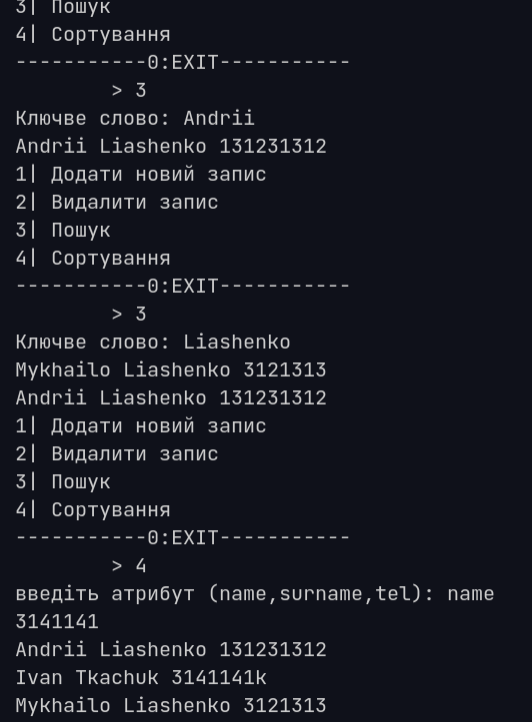
std::cout << "3| Пошук " << std::endl;

std::cout << "4| Сортування" << std::endl;

std::cout << "-----------0:EXIT-----------\n\t> ";

}

Виконання:



**ВИСНОВКИ**

**(копіюємо з щоденника робочі записи)**