МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ



**Дніпровський національний університет  
залізничного транспорту імені академіка В. Лазаряна**

Кафедра «Комп’ютерні інформаційні технології»

**Лабораторна робота №9**

**з дисципліни «Основи програмування»**

**на тему: «Файлове введення-виведення. Бінарні файли. Пошук і сортування даних.»**

Виконав: студент гр. ПЗ1911

Сіньков Г.О.

Прийняла: ас. каф. КІТ

Нежуміра О. І.

Дніпро, 2020

**Лабораторна робота №9**

**Тема.** Файлове введення-виведення. Бінарні файли. Пошук і сортування даних.

**Мета:**

1. сформувати навики роботи з бінарними файлами;
2. засвоїти реалізацію файлового введення-виведення у стилі мови С++;
3. ознайомитися з алгоритмами пошуку і сортування даних.

**Постановка задачі**

1.Загальна постановка: Розробити багато-файлову програму для роботи з бінарними файлами для заданого індивідуального завдання і обраного рівня складності.

2.Індивідуальна постановка:

Варіант 9. Предметна область – квартира(2):

* додавання квартир у список
* виведення списку квартир(A-E)
* виведення списку квартир вказаної ціни(E)
* пошук квартири з мінімальною ціною(D)
* пошук вказаної квартири в списку(C)

3.Вимоги до програми:

- імена файлів ввести з клавіатури;

- в оперативної пам’яті дозволяється зберігати не більше двох блоків (структур) з вхідного файлу (файл в оперативну пам’ять цілком не зчитувати!);

- передбачити перевірку всіх операцій роботи з файлами (відкриття, читання, запису);

- управління виконанням програми здійснити на основі текстового меню користувача;

- передбачити послідовну обробку необмеженої кількості файлів.

**4.Вимоги до тексту програми:**

- специфікації програми і її функцій;

- коментарі щодо призначення блоків програми, дій окремих операторів для пояснення алгоритму;

- самодокументованість коду: всі ідентифікатори повинні мати назви, що відповідають суті змінних.

**Зовнішні специфікації**

1. Вхідні дані

Вхідні дані: menu choice, search\_price, one.name, one.built.street, one.built.building, one.built.number, one.price, name[20]

Формат вхідних даних

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Найменування даних | Умовне позначення | Вимоги до даних | Приклад |
| 1. | Номер меню | menu choice | Ціле число, n > 0 | 2 |
| 2. | ФІО и вулиця | one.name  one.built.street | Текст | Сідоров Заур Данилович  Карла Маркса |
| 3. | Номер будинку, квартири та ціни | one.built.building  one.built.number  one.price | Ціле число | 555  55  55000 |
| 4. | Пошук за ціною | search\_price | Ціле число | 12000 |
| 5. | Ім’я файлу | name[20] | Текст | text.bin |

2**.** Вихідні дані

Вихідні дані: tmp\_house.name, tmp\_house.built.street, tmp\_house.built.building, tmp\_house.built.number, tmp\_house.price

Формат вихідних даних

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Найменування даних | Умовне позначення | Вимоги до даних | Приклад |
| 1 | ФІО и вулиця | tmp\_house.name  tmp\_house.built.street | Текст | Сідоров Заур Данилович  Карла Маркса |
| 2 | Номер будинку, квартири та ціни | tmp\_house.built.building  tmp\_house.built.number  tmp\_house.price | Ціле число | 555  55  55000 |

**3. Функціональні вимоги до програми**

Програма повинна реалізувати такі дії:

-робота з бінарними файлами

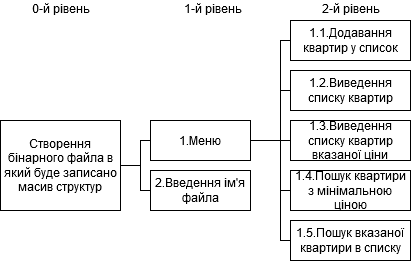
-введення даних у програму

-перевірку введених даних

-управління виконанням за допомогою меню

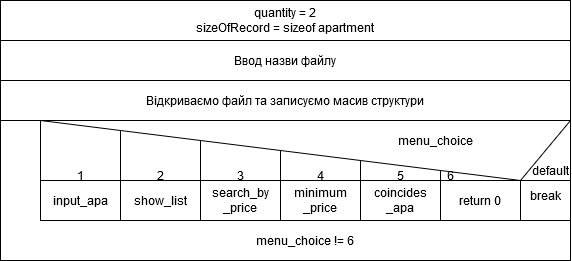
-вивід результату

**Модульна структура програми**

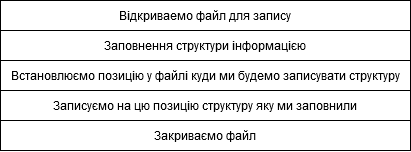


**Алгоритм розв’язання завдання**

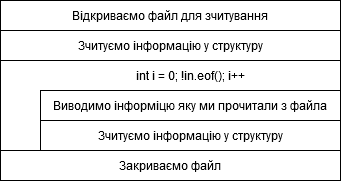
main



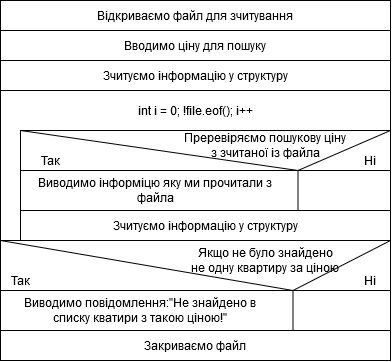
input\_apa



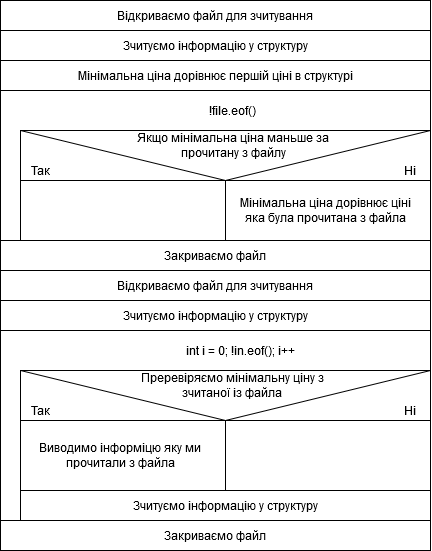
show\_list



search\_by\_price



minimum\_price



coincides\_apa



**Тести**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Назва | Вхідні данні | Результати |
| 1 | Додавання квартири у список | ФІО: Фомин Ждан Станиславович  Вулиця: Старокозацька  Номер будинка: 49  Номер квартири: 21  Ціна:12000 | Данні про квартиру будуть записані у файл |
| 2 | Виведення квартир у список | - | Буде виведений повний список квартир |
| 3 | Виведення списку квартир вказаної ціни | search\_price = 12000 | Буде виведена квартира з вказаної ціни |
| 4 | Виведення списку квартир вказаної ціни(немає) | search\_price = 49000 | Не знайдено в списку квартири з такою ціною |
| 5 | Пошук квартир з мінімальною ціною | - | Буде виведена квартира з мінімальною ціною |
| 6 | Пошук вказаної квартири в списку | ФІО: Сідоров Заур Данилович  Вулиця: Карла Маркса  Номер будинка: 555  Номер квартири: 55  Ціна:55000 | Буде виведене повідомлення: ”Квартира с такими реквізитами є у списку” та дані про цю квартиру |
| 7 | Пошук вказаної квартири в списку якої не має | Сідоров Заур Данилович  Вулиця: Карла Маркса  Номер будинка: 562  Номер квартири: 55  Ціна:45000 | Буде виведене повідомлення: ”Не має квартири в списку!” |

**Текст програми**

main.cpp

#include <iostream>

#include <fstream>

#include <Windows.h>

#include "Header.h"

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int quantity = 2;

int menu\_choice;

int sizeOfRecord = sizeof apartment;

char name[20];

apartment house[2] = { { "Сідоров Заур Данилович", "Карла Маркса", 555, 55, 55000 },

{ "Семёнов Родион Петрович", "Кіров", 485, 6, 45000 }

};

std::cout << "Введіть назву файла: ";

std::cin >> name;

std::ofstream file(name, std::ios::binary | std::ios::out | std::ios::trunc);

file.write(reinterpret\_cast<char\*>(house), sizeof house);

file.close();

do {

menu\_choice = input();

switch (menu\_choice) {

case 1:

std::cin.get();

input\_apa(quantity, house, sizeOfRecord);

break;

case 2:

show\_list(quantity, sizeOfRecord);

break;

case 3:

search\_by\_price(quantity, sizeOfRecord, name);

break;

case 4:

minimum\_price(quantity, sizeOfRecord);

break;

case 5:

std::cin.get();

coincides\_apa(sizeOfRecord);

break;

case 6:

return 0;

break;

default:

break;

}

} while (menu\_choice != 6);

}

Header.h

#ifndef \_\_HEADER\_H\_\_

#define \_\_HEADER\_H\_\_

struct address {

char street[50];

int building;

int number;

};

struct apartment {

char name[50];

address built;

int price;

};

int input();

void input\_apa(int &quantity, apartment \*house, int sizeOfRecord, char \*name);

void show\_list(int& quantity, int sizeOfRecord, char \*name);

void search\_by\_price(int quantity, int sizeOfRecord, char \*name);

void minimum\_price(int quantity, int sizeOfRecord, char \*name);

void coincides\_apa(int sizeOfRecord, char \*name);

#endif

Header.cpp

#include <iostream>

#include <iomanip>

#include <fstream>

#include <string>

#include "Header.h"

int input() {

int n;

while (true) {

system("cls");

std::cout << "1. Додавання квартир у список" << std::endl;

std::cout << "2. Виведення квартир у список" << std::endl;

std::cout << "3. Виведення списку квартир вказаної ціни" << std::endl;

std::cout << "4. Пошук квартир з мінімальною ціною" << std::endl;

std::cout << "5. Пошук вказаної квартири в списку" << std::endl;

std::cout << "6. Выход" << std::endl;

std::cin >> n;

if ((std::cin.peek() != '\n') || !(n > 0)) {

std::cin.clear();

std::cin.ignore(32767, '\n');

}

else

return n;

}

}

void input\_apa(int &quantity, apartment \*house, int sizeOfRecord, char \*name) {// додавання квартир в список

std::system("cls");

long pos;

apartment one;

std::cout << "№" << quantity + 1 << std::endl;

std::fstream file(name, std::ios::binary | std::ios::in | std::ios::out);

std::cout << " Введіть ФІО: ";

std::cin.getline(one.name, 50);

std::cout << " Введіть адрес: ";

std::cin.getline(one.built.street, 50);

std::cout << " Введіть номер будинка: ";

std::cin >> one.built.building;

std::cout << " Введіть номер квартири: ";

std::cin >> one.built.number;

std::cout << " Введіть ціну квартири: ";

std::cin >> one.price;

file.seekg(quantity \* static\_cast<double> (sizeOfRecord), std::ios::beg); //set to fourth record

pos = file.tellg();

file.seekp(pos, std::ios::beg);

file.write(reinterpret\_cast<char\*>(&one), sizeOfRecord);

quantity++;

file.close();

system("pause");

}

void show\_list(int& quantity, int sizeOfRecord, char \*name) {// вивід списку всіх квартир

system("cls");

std::ifstream in(name, std::ios::binary | std::ios::in);

apartment tmp\_house;

in.read(reinterpret\_cast<char\*>(&tmp\_house), sizeOfRecord);

for (int i = 0; i < quantity || !in.eof(); i++) {

std::cout << "№" << i + 1 << std::endl;

std::cout << " ФІО: " << tmp\_house.name << std::endl;

std::cout << " Адрес: " << tmp\_house.built.street << ", " << tmp\_house.built.building << ", " << tmp\_house.built.number << std::endl;

std::cout << " Ціна: " << tmp\_house.price << std::endl << std::endl;

in.read(reinterpret\_cast<char\*>(&tmp\_house), sizeOfRecord);

}

system("pause");

in.close();

}

void search\_by\_price(int quantity, int sizeOfRecord, char \*name) {// Виведення списку квартир вказаної ціни

std::system("cls");

int search\_price;

bool find = false;

apartment tmp\_house;

std::ifstream file(name, std::ios::binary | std::ios::in);

std::cout << "Введіть ціну квартири: ";

std::cin >> search\_price;

std::cin.get(); // очистка буфера

file.read(reinterpret\_cast<char\*>(&tmp\_house), sizeOfRecord);

for (int i = 0; !file.eof(); i++) {

if (search\_price == tmp\_house.price) {

std::cout << "№" << i + 1 << std::endl;

std::cout << " ФІО: " << tmp\_house.name << std::endl;

std::cout << " Адрес: " << tmp\_house.built.street << ", " << tmp\_house.built.building << ", " << tmp\_house.built.number << std::endl;

std::cout << " Ціна: " << tmp\_house.price << std::endl << std::endl;

find = true;

}

file.read(reinterpret\_cast<char\*>(&tmp\_house), sizeOfRecord);

}

if (!find)

std::cout << "Не знайдено в списку кватири з такою ціною!" << std::endl << std::endl;

file.close();

system("pause");

}

void minimum\_price(int quantity, int sizeOfRecord, char \*name){// Пошук квартир з мінімальною ціною

std::system("cls");

int min\_price;

apartment tmp\_house;

std::ifstream file(name, std::ios::binary | std::ios::in);

file.read(reinterpret\_cast<char\*>(&tmp\_house), sizeOfRecord);

min\_price = tmp\_house.price;

while (!file.eof()) {

if (min\_price < tmp\_house.price) {}

else

min\_price = tmp\_house.price;

file.read(reinterpret\_cast<char\*>(&tmp\_house), sizeOfRecord);

}

file.close();

std::ifstream in(name, std::ios::binary | std::ios::in);

in.read(reinterpret\_cast<char\*>(&tmp\_house), sizeOfRecord);

for (int i = 0; !in.eof(); i++) {

if (min\_price == tmp\_house.price) {

std::cout << "№" << i + 1 << std::endl;

std::cout << " ФІО: " << tmp\_house.name << std::endl;

std::cout << " Адрес: " << tmp\_house.built.street << ", " << tmp\_house.built.building << ", " << tmp\_house.built.number << std::endl;

std::cout << " Ціна: " << tmp\_house.price << std::endl << std::endl;

}

in.read(reinterpret\_cast<char\*>(&tmp\_house), sizeOfRecord);

}

in.close();

system("pause");

}

void coincides\_apa(int sizeOfRecord, char \*name){// Пошук вказаної квартири в списку

std::system("cls");

bool find = true;

apartment tmp\_house, comparison\_house;

std::ifstream file(name, std::ios::binary | std::ios::in);

std::cout << "Введіть ФІО: ";

std::cin.getline(comparison\_house.name, 50);

std::cout << "Введіть адрес: ";

std::cin.getline(comparison\_house.built.street, 50);

std::cout << "Введіть номер будинка: ";

std::cin >> comparison\_house.built.building;

std::cout << "Введіть номер квартири: ";

std::cin >> comparison\_house.built.number;

std::cout << "Введіть ціну квартири: ";

std::cin >> comparison\_house.price;

file.read(reinterpret\_cast<char\*>(&tmp\_house), sizeOfRecord);

for (int i = 0; !file.eof(); i++) {

if (strcmp(comparison\_house.name, tmp\_house.name) == 0) {

if (strcmp(comparison\_house.built.street, tmp\_house.built.street) == 0) {

if (comparison\_house.built.building == tmp\_house.built.building) {

if (comparison\_house.built.number == tmp\_house.built.number) {

if (comparison\_house.price == tmp\_house.price) {

std::cout << std::endl << "Квартира с такими реквізитами є у списку." << std::endl;

std::cout << "№" << i + 1 << std::endl;

std::cout << " ФІО: " << tmp\_house.name << std::endl;

std::cout << " Адрес: " << tmp\_house.built.street << ", " << tmp\_house.built.building << ", " << tmp\_house.built.number << std::endl;

std::cout << " Ціна: " << tmp\_house.price << std::endl << std::endl;

find = false;

}

}

}

}

}

file.read(reinterpret\_cast<char\*>(&tmp\_house), sizeOfRecord);

}

if (find)

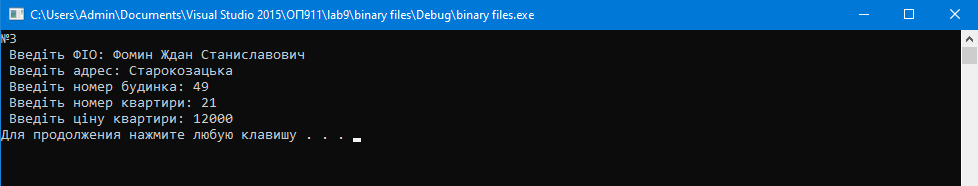
std::cout << std::endl << "Не має квартири в списку!" << std::endl;

system("pause");

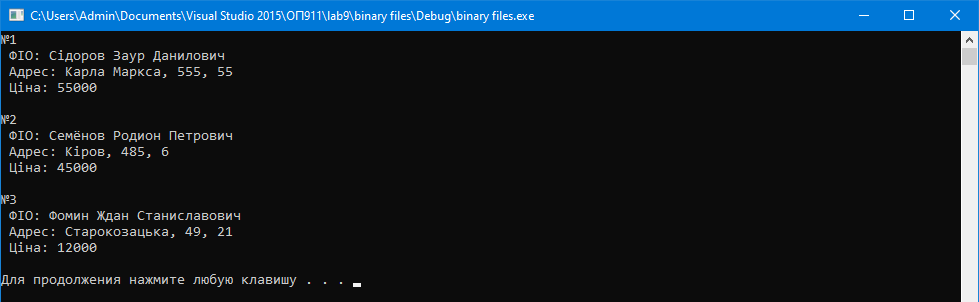
}

**Результати тестування програми**

1.Додавання квартир у список

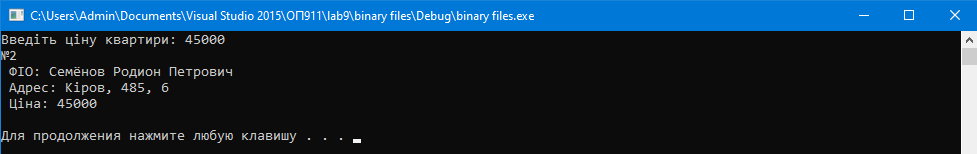


2.Виведення списку квартир

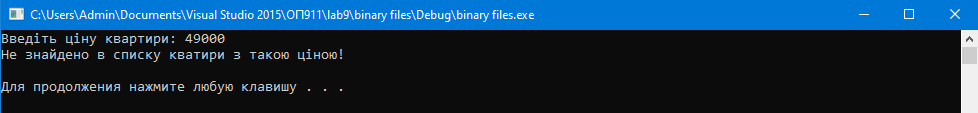


3.Виведення списку квартир вказаної ціни

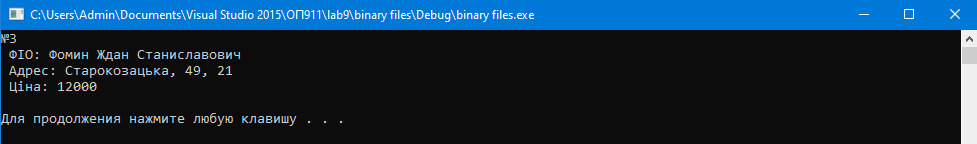
а) Є квартири за такою ціною



б) Немає квартири за такою ціною

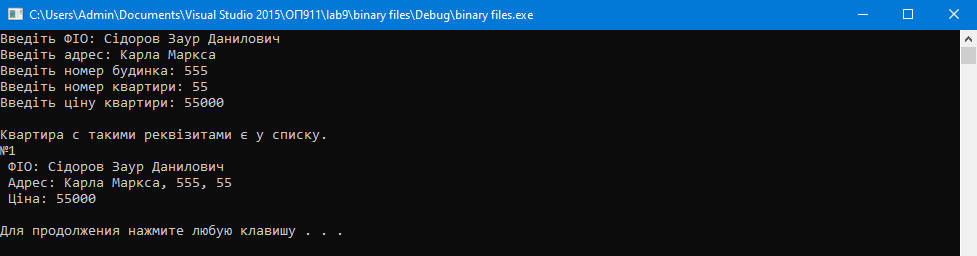


4.Пошук квартир з мінімальною ціною

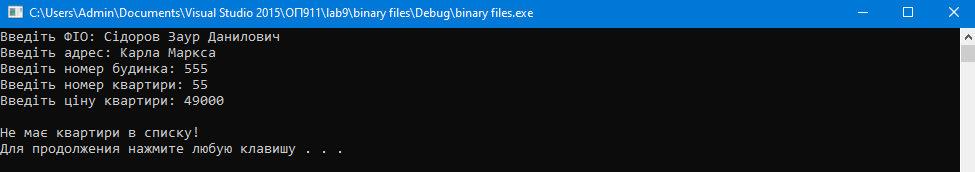


5.Пошук вказаної квартири у списку

а) Є вказана квартира у списку



б) Немає вказаної квартири у списку



Висновок: залежно від завдання буває зручно вибрати один з цих варіантів. Наприклад, для налаштувань якої-небудь гри або програми може бути зручно зберігати файли в текстовому вигляді, щоб можна було взяти відкрити будь-яким текстовим редактором і поміняти якусь настройку швидко і просто. А ось збереження ігор зручніше зберігати в бінарних файлах. Як мінімум це легка захист від застосування читов. Адже що за гра, якщо тобі система дала 100 монет, а ти відкриваєш файл, пишеш 1000.000 монет, купуєш найкрутіше і такий прям класний-першокласний гравець. Все це до того, що вибір довічного або текстового файлу може залежати від ситуації.