Міністерство освіти і науки України

ВСП - Технічний коледж Національного університету "Львівська політехніка"

Відділення

інформаційних технологій

та комп’ютерної техніки

**Звіт**

До лабораторної роботи № 1

з предмету «Навчальна практика з об’єктно-орієнтованого програмування»

тема: «Створення класу С++»

*Виконала:*

*Студентка групи 31 ПЗ*

*Павлисько Анастасія*

*Перевірив: Чистяк В.І.*

Львів – 2021

**Мета:** порівняння об’єктно-орієнтованого та функціонального підходів; початкове знайомство з класами, об’єктами та головними елементами об’єктного підходу

**Хід роботи**

Створити клас для обробки масиву даних. Розмістити інтерфейс класу у заголовочному файлі, а визначення функцій та головну функцію програми – у двох окремих файлах. Передбачити можливість роботи з довільним числом записів, а також реалізувати окремими функціями класу:

* конструктори без параметрів та з параметрами;
* додавання;
* знищення;
* виведення інформації на екран;
* пошук потрібної інформації за конкретною ознакою;
* редагування записів;
* сортування за різними полями.

**Варіант 5** «записна книжка»

Прізвище, ім’я, по батькові, домашня адреса, телефон, електронна пошта.

Написати програму, яка демонструє роботу з об’єктами цього класу. Програма повинна містити меню для перевірки усіх методів класу і операцій.

**Лістинг програми**

//Notebook.h

#pragma once

#include <string.h>

#include <iostream>

#include <vector>

class Notebook

{

std::string name;

std::string lastname;

std::string address;

std::string phone;

std::string mail;

public:

//конструктори без параметрів та з параметрами;

Notebook() {

name = "";

lastname = "";

address = "";

phone = "";

mail = "";

}

Notebook(const std::string& name, const std::string& lastname, const std::string& address, const std::string& phone, const std::string& mail);

Notebook(const Notebook& data);

bool operator==(const Notebook& note);

std::string& getName();// повертає ім'я з запису

std::string& getLastname();// повертає прізвище з запису

std::string& getAddress();// повертає адресу з запису

std::string& getPhone();// повертає телефон з запису

std::string& getMail();// повертає пошту з запису

void NoteSwap(Notebook& data);//міняє місцями дані об'єктів

//додавання;

void AddNote(const std::string& name, const std::string& lastname, const std::string& address, const std::string& phone, const std::string& mail);

void AddNote();

//знищення;

void ClearNote();

//виведення інформації на екран;

void WriteData()const;

void WriteShortData()const;//виводить лише ім'я і номер телефону

//редагування записів;

void ChangeNote(const std::string& changer, std::string& data);

};

int SearchNote(const Notebook data, std::vector<Notebook> records, int size);

//пошук потрібної інформації за ім'ям;

std::vector<int> SearchName(const std::string name, std::vector<Notebook> records, int size, int returnNumber);

//пошук потрібної інформації за прізвищем;

std::vector<int> SearchLastname(const std::string lastname, std::vector<Notebook> records, int size, int returnNumber);

//пошук потрібної інформації за адресою;

std::vector<int> SearchAddress(const std::string address, std::vector<Notebook> records, int size, int returnNumber);

//пошук потрібної інформації за телефоном;

int SearchPhone(const std::string phone, std::vector<Notebook> records, int size);

//сортування за різними полями.

void SortData(std::vector<Notebook>& records, int size, const std::string sortflag);

//Notebook.cpp

#include "Notebook.h"

#include <istream>

#include <string>

//міняє місцями два об'єкти

void Notebook::NoteSwap(Notebook& data)

{

Notebook temp(data);

data.name = name;

data.lastname = lastname;

data.address = address;

data.phone = phone;

data.mail = mail;

name = temp.name;

lastname = temp.lastname;

address = temp.address;

phone = temp.phone;

mail = temp.mail;

}

//конструктор з параметрами;

Notebook::Notebook(const std::string& name, const std::string& lastname, const std::string& address, const std::string& phone, const std::string& mail)

{

this->name = name;

this->lastname = lastname;

this->address = address;

this->phone = phone;

this->mail = mail;

}

//конструктор copy;

Notebook::Notebook(const Notebook& data)

{

this->name = data.name;

this->lastname = data.lastname;

this->address = data.address;

this->phone = data.phone;

this->mail = data.mail;

}

bool Notebook::operator==(const Notebook& note)

{

if (name == note.name && lastname == note.lastname && address == note.address && phone == note.phone && mail == note.mail)

return true;

else

return false;

}

//повертає значення полей

std::string& Notebook::getName()

{

return name;

}

std::string& Notebook::getLastname()

{

return lastname;

}

std::string& Notebook::getAddress()

{

return address;

}

std::string& Notebook::getPhone()

{

return phone;

}

std::string& Notebook::getMail()

{

return mail;

}

//додає вказані дані

void Notebook::AddNote(const std::string& name, const std::string& lastname, const std::string& address, const std::string& phone, const std::string& mail)

{

this->name = name;

this->lastname = lastname;

this->address = address;

this->phone = phone;

this->mail = mail;

}

//додає дані з клавіатури

void Notebook::AddNote() {

std::cout << "Enter data for note\n";

std::cout << "Name: "; std::getline(std::cin, name);

std::cout << "Lastame: "; std::getline(std::cin, lastname);

std::cout << "Address: "; std::getline(std::cin, address);

std::cout << "Phone: "; std::getline(std::cin, phone);

std::cout << "Mail: "; std::getline(std::cin, mail);

}

//знищення;

void Notebook::ClearNote()

{

name = "";

lastname = "";

address = "";

phone = "";

mail = "";

}

//виведення інформації на екран;

void Notebook::WriteData() const

{

std::cout << "\nName: " << name << " " << lastname;

std::cout << "\nAddress: " << address << "\nPhone: " << phone << "\nE-Mail: " << mail;

}

//виведення на екран скороченої інформації

void Notebook::WriteShortData() const

{

std::cout << "\nName: " << name << " " << lastname;

std::cout << "\nPhone: " << phone << '\n';

}

//редагування записів;

void Notebook::ChangeNote(const std::string& changer, std::string& data)

{

if (changer == "name") name = data;

else if (changer == "lastname") lastname = data;

else if (changer == "address") address = data;

else if (changer == "phone") phone = data;

else if (changer == "mail") mail = data;

else std::cout << "\nChange error\n";

}

//пошук потрібної інформації за співпадінням;

int SearchNote(const Notebook data, std::vector<Notebook> records, int size)

{

int index = -1;

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

if (records[i] == data) return i;

}

return index;

}

//пошук потрібної інформації за ім'ям;

std::vector<int> SearchName(const std::string name, std::vector<Notebook> records, int size, int returnNumber)

{

std::vector<int> index;

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

if (records[i].getName() == name && returnNumber > 0) {

index.push\_back(i);

returnNumber--;

}

if (returnNumber == 0) break;

}

return index;

}

//пошук потрібної інформації за прізвищем;

std::vector<int> SearchLastname(const std::string lastname, std::vector<Notebook> records, int size, int returnNumber)

{

std::vector<int> index;

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

if (records[i].getLastname() == lastname && returnNumber > 0) {

index.push\_back(i);

returnNumber--;

}

if (returnNumber == 0) break;

}

return index;

}

//пошук потрібної інформації за адресою;

std::vector<int> SearchAddress(const std::string address, std::vector<Notebook> records, int size, int returnNumber)

{

std::vector<int> index;

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

if (records[i].getAddress() == address && returnNumber > 0) {

index.push\_back(i);

returnNumber--;

}

if (returnNumber == 0) break;

}

return index;

}

//пошук потрібної інформації за телефоном;

int SearchPhone(const std::string phone, std::vector<Notebook> records, int size)

{

int index = -1;

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

if (records[i].getPhone() == phone) {

return i; //повертає індекс об'єкта де співпав номер телефону

}

}

return index; //повертає -1 якщо потрібного запису не знайдено

}

//сортування за різними полями.

void SortData(std::vector<Notebook>& records, int size, const std::string sortflag)

{

if (sortflag == "name") { //сортує за ім'ям

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

for (int j = 0; j < size - i; j++) {

if (records[j].getName() > records[j + 1].getName())

records[j].NoteSwap(records[j + 1]);

}

}

}

else if (sortflag == "lastname") { //сортує за прізвищем

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

for (int j = 0; j < size - i; j++) {

if (records[j].getLastname() > records[j + 1].getLastname())

records[j].NoteSwap(records[j + 1]);

}

}

}

else if (sortflag == "address") {//сортує за адресою

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

for (int j = 0; j < size - i; j++) {

if (records[j].getAddress() > records[j + 1].getAddress())

records[j].NoteSwap(records[j + 1]);

}

}

}

else if (sortflag == "phone") { //сортує за номером телефону

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

for (int j = 0; j < size - i; j++) {

if (records[j].getPhone() > records[j + 1].getPhone())

records[j].NoteSwap(records[j + 1]);

}

}

}

else if (sortflag == "mail") { //сортує за e-mail

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

for (int j = 0; j < size - i; j++) {

if (records[j].getMail() > records[j + 1].getMail())

records[j].NoteSwap(records[j + 1]);

}

}

}

else std::cout << "\nSorting error\n";//виводить повідомлення про помилку якщо параметр сортування не відповідний

}

//main()

#include "Notebook.h"

#include <iostream>

#include <string>

#include <vector>

void Change(std::vector<Notebook>& records){

std::string changer, changeTo;

std::cout << "What do you whant change? (name, lastname, address, phone, mail) ";

std::getline(std::cin, changer);

std::cout << "Change to ";

std::getline(std::cin, changeTo);

char param;

int index = -1; std::vector<int> indexes;

std::string str1, str2;

std::cout << "Enter index(i) or data(d) for changing ";

std::cin >> param; std::cin.ignore();

if (param == 'i') {

std::cout << "Enter index ";

std::cin >> index;

std::cin.ignore();

}

else if (param == 'd') {

std::cout << "Enter name: "; std::getline(std::cin, str1);

std::cout << "\nEnter lastname: "; std::getline(std::cin, str2);

int n = records.size();

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

if (records[i].getName() == str1 && records[i].getLastname() == str2)

if (index == -1) {

index = i;

indexes.push\_back(i);

}

else indexes.push\_back(i);

}

if (indexes.size() > 1) {

std::cout << "Enter address: "; std::getline(std::cin, str1);

n = indexes.size();

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

if (records[indexes[i]].getAddress() == str1)

index = i;

}

}

}

else std::cout << "\nInput incorect\n";

records[index].ChangeNote(changer, changeTo);

std::cout << "\n Note was changed: \n";

records[index].WriteData();

}

void Search(std::vector<Notebook>& records)

{

int parametr;

std::cout << "Choose parametr for search: 0 - note, 1 - name, 2 - lastname, 3 - address, 4 - phone\n";

std::cin >> parametr; std::cin.ignore();

std::string str\_param;

int count = 0;

std::vector<int> indexes;

switch (parametr) {

case 0: {

Notebook data;

data.AddNote();

std::cout << '\n';

int index = SearchNote(data, records, records.size());

if (index == -1)

std::cout << "\nNote is not found\n";

else {

std::cout << '\n';

records[index].WriteData();

std::cout << '\n';

}

}break;

case 1: {

std::cout << "Enter name for search: ";

std::getline(std::cin, str\_param);

std::cout << "Enter count of notes for search: ";

std::cin >> count; std::cin.ignore();

indexes = SearchName(str\_param, records, records.size(), count);

if (indexes.empty())

std::cout << "\nNote is not found\n";

else if (indexes.size() == 1) {

std::cout << '\n';

records[indexes[0]].WriteData();

}

else {

int n = indexes.size();

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

std::cout << '\n';

records[indexes[i]].WriteData();

std::cout << '\n';

}

}

}break;

case 2: {

std::cout << "Enter lastname for search: ";

std::getline(std::cin, str\_param);

std::cout << "Enter count of notes for search: ";

std::cin >> count; std::cin.ignore();

indexes = SearchLastname(str\_param, records, records.size(), count);

if (indexes.empty())

std::cout << "\nNote is not found\n";

else if (indexes.size() == 1) {

std::cout << '\n';

records[indexes[0]].WriteData();

std::cout << '\n';

}

else {

int n = indexes.size();

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

std::cout << '\n';

records[indexes[i]].WriteData();

std::cout << '\n';

}

}

}break;

case 3: {

std::cout << "Enter address for search: ";

std::getline(std::cin, str\_param);

std::cout << "Enter count of notes for search: ";

std::cin >> count; std::cin.ignore();

indexes = SearchAddress(str\_param, records, records.size(), count);

if (indexes.empty())

std::cout << "\nNote is not found\n";

else if (indexes.size() == 1) {

std::cout << '\n';

records[indexes[0]].WriteData();

std::cout << '\n';

}

else {

int n = indexes.size();

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

std::cout << '\n';

records[indexes[i]].WriteData();

std::cout << '\n';

}

}

}break;

case 4: {

std::cout << "Enter phone for search: ";

std::getline(std::cin, str\_param);

std::cout << "Enter count of notes for search: ";

std::cin >> count; std::cin.ignore();

int index = SearchPhone(str\_param, records, records.size());

if (index == -1)

std::cout << "\nNote is not found\n";

else {

std::cout << '\n';

records[index].WriteData();

std::cout << '\n';

}

}break;

default: {

std::cout << "Parametr incorect!\n";

break;

}break;

}

}

void Clear(std::vector<Notebook>& records) {

char param;

int index = -1; std::vector<int> indexes;

std::string str1, str2;

std::cout << "Enter index(i) or data(d) to clear ";

std::cin >> param; std::cin.ignore();

if (param == 'i') {

std::cout << "Enter index "; std::cin >> index;

}

else if (param == 'd') {

std::cout << "Enter name: "; std::getline(std::cin, str1);

std::cout << "\nEnter lastname: "; std::getline(std::cin, str2);

int n = records.size();

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

if (records[i].getName() == str1 && records[i].getLastname() == str2)

if (index == -1) {

index = i;

indexes.push\_back(i);

}

else indexes.push\_back(i);

}

if (indexes.size() > 1) {

std::cout << "Enter address: "; std::getline(std::cin, str1);

n = indexes.size();

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

if (records[indexes[i]].getAddress() == str1)

index = i;

}

}

}

else std::cout << "\nInput incorect\n";

records[index].ClearNote();

std::cout << "This note is clear: \n";

int n = records.size();

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

records[i].WriteShortData();

std::cout << '\n';

}

int flag;

std::cout << "Do you wont delete this note?\n 1 - yes 2 - no: ";

std::cin >> flag; std::cin.ignore();

if (flag == 1) {

records.erase(records.begin()+index);

}

}

void Delete(std::vector<Notebook> records) {

char param;

int index = -1; std::vector<int> indexes;

std::string str1, str2;

std::cout << "Enter index(i) or data(d) to clear ";

std::cin >> param; std::cin.ignore();

if (param == 'i') {

std::cout << "Enter index "; std::cin >> index;

}

else if (param == 'd') {

std::cout << "Enter name: "; std::getline(std::cin, str1);

std::cout << "\nEnter lastname: "; std::getline(std::cin, str2);

int n = records.size();

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

if (records[i].getName() == str1 && records[i].getLastname() == str2)

if (index == -1) {

index = i;

indexes.push\_back(i);

}

else indexes.push\_back(i);

}

if (indexes.size() > 1) {

std::cout << "Enter address: "; std::getline(std::cin, str1);

n = indexes.size();

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

if (records[indexes[i]].getAddress() == str1)

index = i;

}

}

}

records.erase(records.begin() + index);

std::cout << "\nNote was deleted\n ";

}

int main(){

int key;

bool toExit = false;

std::vector<Notebook> notes;

while(!toExit){

std::cout << "What do you want to do?\n";

std::cout << " 1 - add note\n 2 - change note\n 3 - search note\n 4 - sort and show notes\n 5 - show note\n 6 - clear note\n 7 - delete note\n 8 - exit\n";

std::cin >> key; std::cin.ignore();

switch (key) {

case 1: {

Notebook data; data.AddNote();

notes.push\_back(data);

std::cout << '\n';

}break;

case 2: {

if (notes.empty()) std::cout << "\nYou have not notes to change!\n";

else {

Change(notes);

std::cout << '\n';

}

}break;

case 3: {

if (notes.empty()) std::cout << "\nYou have not notes to search!\n";

else if (notes.size() == 1) {

std::cout << "\nYou have only one note: \n";

notes[0].WriteData();

std::cout << '\n';

}

else {

Search(notes);

std::cout << '\n';

}

}break;

case 4: {

if (notes.empty() || notes.size() == 1) std::cout << "\nYou have not notes to sort yet!\n";

else {

std::string sortto;

std::cout << "Sortig order to (name, lastname, address, phone, mail) ";

std::getline(std::cin, sortto);

SortData(notes, notes.size()-1, sortto);

std::cout << '\n';

}

}

case 5: {

if (notes.empty()) std::cout << "\nYou have not notes yet!\n";

else {

char form;

std::cout << "\nChoose output form: short(s) or full(f) ";

std::cin >> form; std::cin.ignore();

int n = notes.size();

if (form == 's') {

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

notes[i].WriteShortData();

std::cout << '\n';

}

}

else {

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

notes[i].WriteData();

std::cout << '\n';

}

}

std::cout << '\n';

}

}break;

case 6: {

if (notes.empty()) std::cout << "\nYou have not notes to clear!\n";

else {

Clear(notes);

std::cout << '\n';

}

}break;

case 7: {

if (notes.empty()) std::cout << "\nYou have not notes to delete!\n";

else{

Delete(notes);

std::cout << '\n';

}

}break;

case 8: {

toExit = true;

}break;

default: std::cout << "\nIncorect input - try again\n";

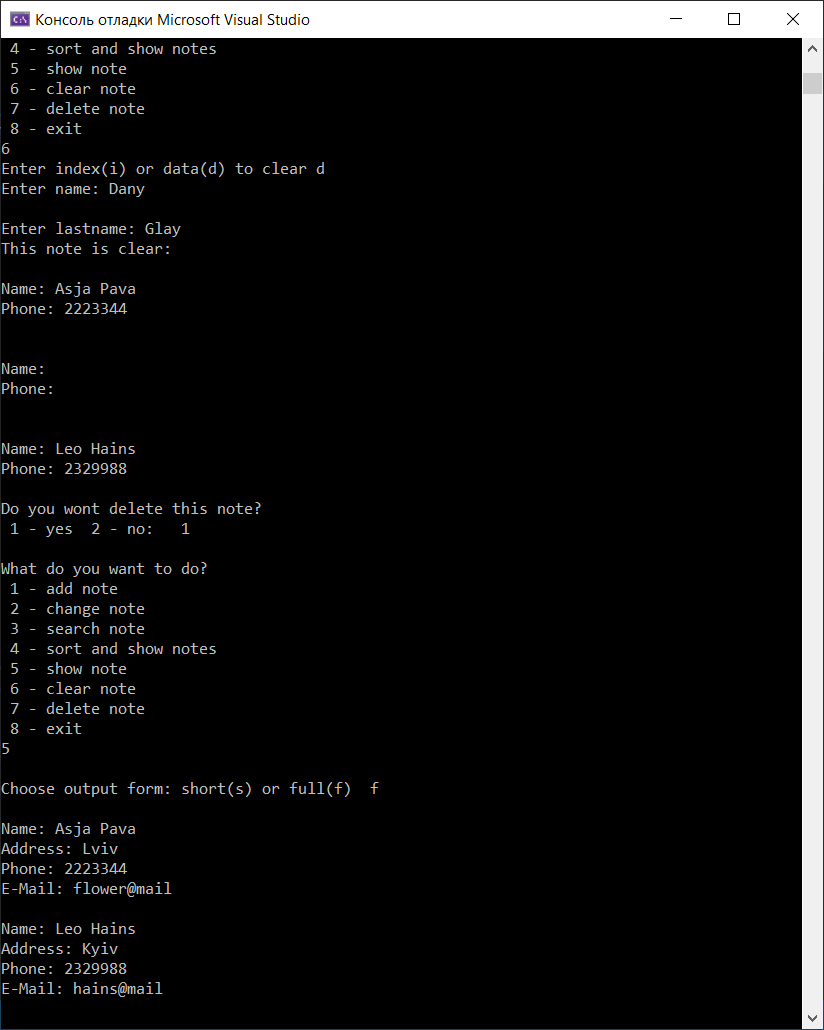
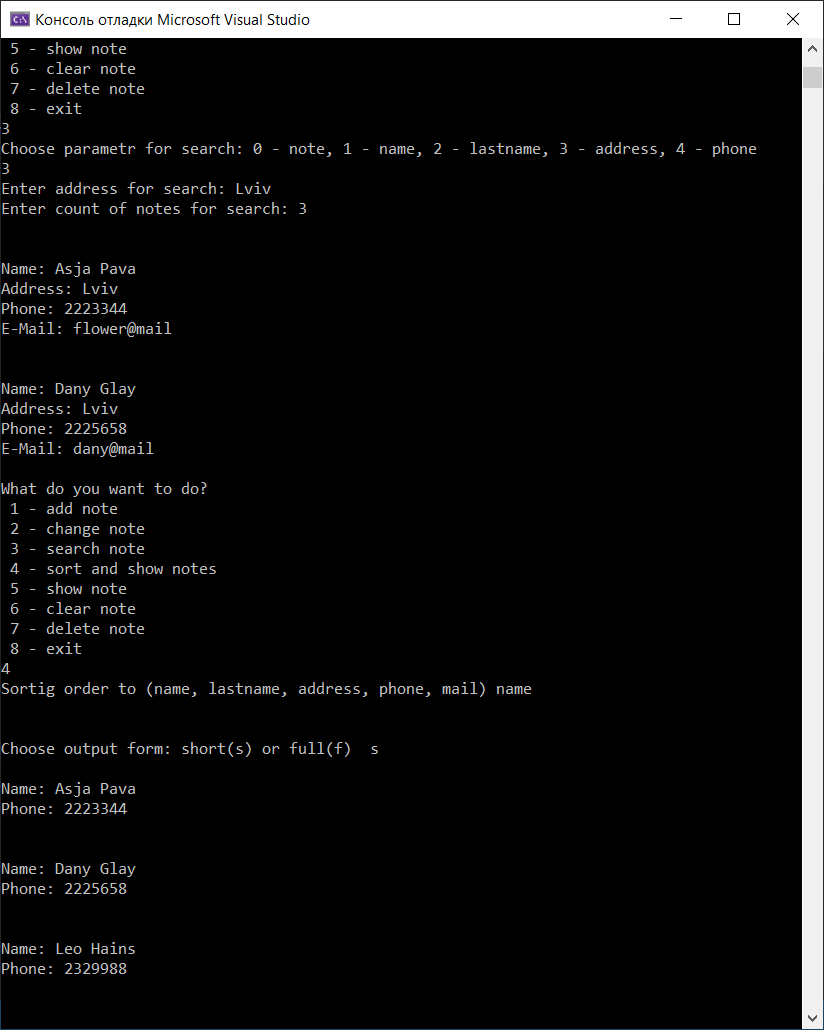
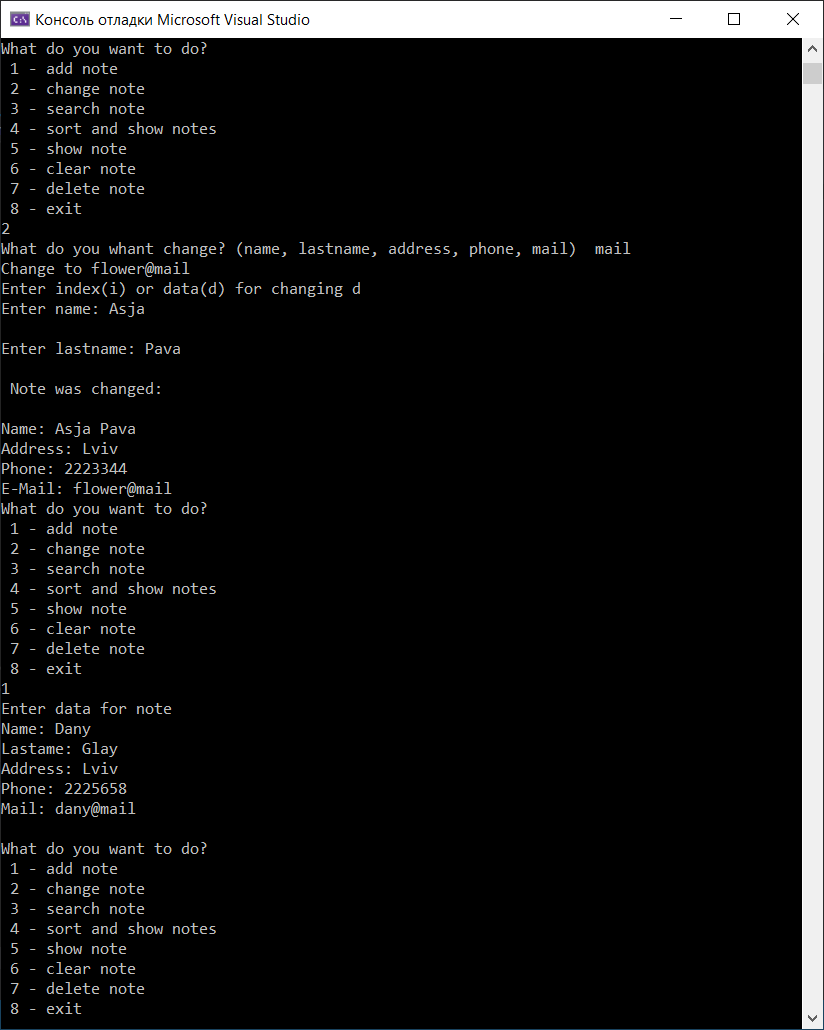
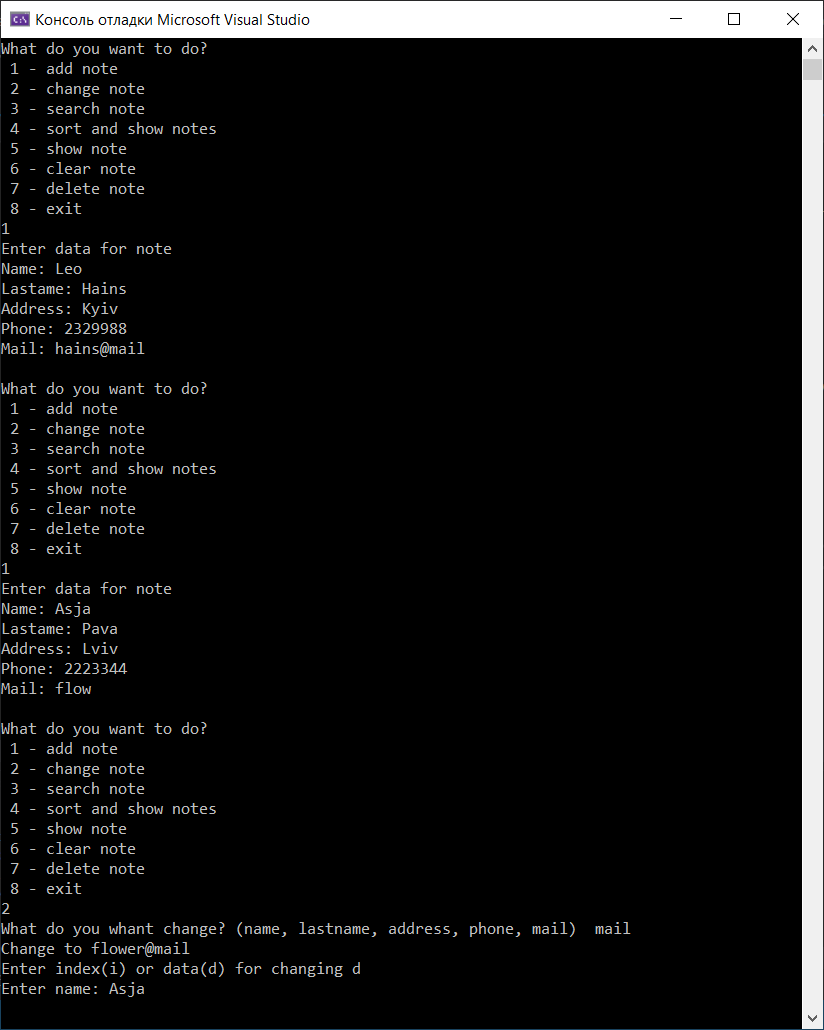
}

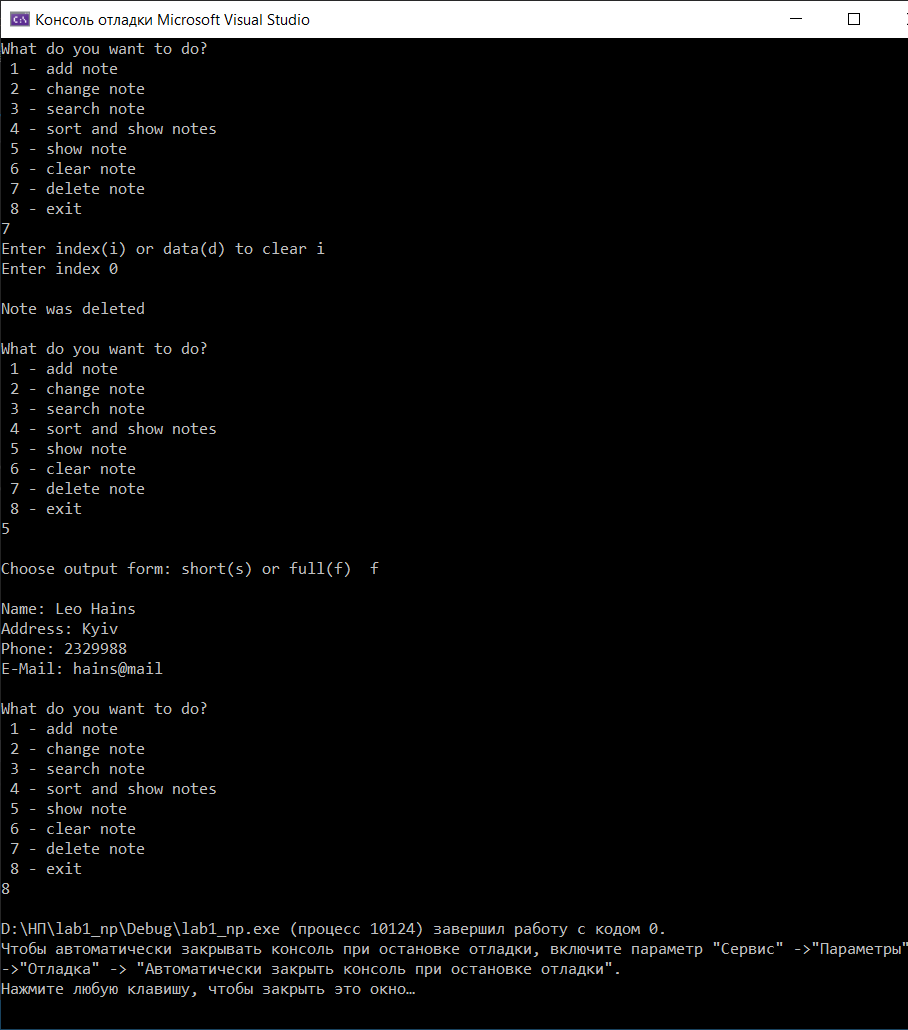
}

return 0;

}

**Результати роботи програми**





**Висновок:**

Виконуючи цю лабораторну роботу, я порівняла об’єктно-орієнтований та функціональний підходи; ознайомилася з класами, об’єктами та головними елементами об’єктного підходу програмування