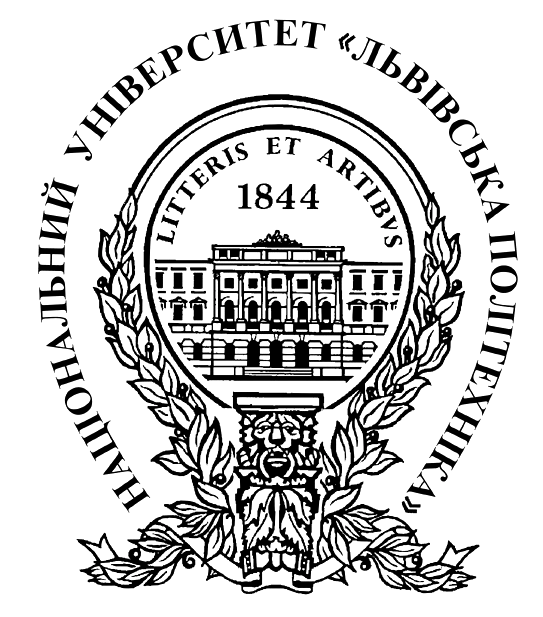
**Міністерство освіти і науки України**

**Національний університет «Львівська політехніка»**

Кафедра ЕОМ



**Курсова робота**

з предмету: «Програмування, частина 2 (Об’єктно-орієнтоване програмування)»

на тему:

**«Базові принципи об’єктно-орієнтованого програмування»**

Індивідуальне завдання: «Фармацевтична компанія»

**Виконав:**

ст. гр. КІ-15

Рикмас О. М.

**Прийняв:**

Козак Н.Б.

Львів-2020

**Анотація**

Даний курсовий проект являє собою створення програмного забезпечення, яке описує таку предметну галузь як кафе і написане на C++ - однією з найуживаніших мов програмування загального призначення. Тут реалізовані парадигми об’єктно-орієнтованого програмування – інкапсуляція, поліморфізм і наслідування, робота з об’єктами й класами, методами і даними, текстовими файлами. Класи дозволяють проводити конструювання із корисних компонент, що дає можливість абстрагуватись від деталей реалізації і створити безпечний об’єкт. Данні і методи разом утворюють певну сутність що призводить до локалізації коду і даних, покращує наочність і зручність розробки програми. Інкапсуляція захищає від несанкціонованого доступу найбільш критичну і важливу інформацію.

Зміст

[1. Завдання на курсову роботу 4](#_Toc41401726)

[2. Огляд та обґрунтовування вибору технологій об’єктно-орієнтованого програмування 5](#_Toc41401727)

[3. Аналіз та розробка алгоритму згідно індивідуального завдання. 8](#_Toc41401728)

[3.1. Опис алгоритму та оцінка його складності. 8](#_Toc41401729)

[3.2. Аналіз особливостей алгоритму. 8](#_Toc41401730)

[4. Програмна реалізація алгоритму згідно індивідуального завдання. 10](#_Toc41401731)

[4.1. Опис структур даних програми. 10](#_Toc41401732)

[4.2. Програма, записана псевдокодом: 10](#_Toc41401733)

[5. Опис реалізації основних функцій/методів. 11](#_Toc41401734)

[5.1. Програмна реалізація алгоритму згідно індивідуального завдання. 11](#_Toc41401735)

[6. Відлагодження та тестування реалізованої програми згідно індивідуального завдання. 14](#_Toc41401736)

[6.1. Відлагодження та тестування окремих частин програми. 14](#_Toc41401737)

[6.2. Відлагодження та тестування програми загалом. 14](#_Toc41401738)

[Висновок 20](#_Toc41401739)

[Список використаної літератури 20](#_Toc41401740)

[Додаток 1. Код програми 21](#_Toc41401741)

# 1. Завдання на курсову роботу

За допомогою об’єктно-орієнтованих засобів створити програмне забезпечення яке б описувало предметну галузь «Фармацевтична компанія».

**Проект має містити :**

* довідники;
* біжучий стан предметної галузі;
* файлові операції;
* алгоритм функціонування програми;
* текстовий інтерфейс вводу-виводу;
* оформлення.

**Програма має дозволяти:**

* перегляд інформації;
* запис інформації;
* редагування інформації;
* пошук інформації;
* видалення інформації.

# 2. Огляд та обґрунтовування вибору технологій об’єктно-орієнтованого програмування

**Абстракція** - в об'єктно-орієнтованому програмуванні це додання об'єкту характеристик, які відрізняють його від всіх інших об'єктів, чітко визначаючи його концептуальні межі. Основна ідея полягає в тому, щоб відокремити спосіб використання складових об'єктів даних від деталей їх реалізації у вигляді більш простих об'єктів, подібно до того, як функціональна абстракція розділяє спосіб використання функції і деталей її реалізації в термінах більш примітивних функцій, таким чином, дані обробляються функцією високого рівня за допомогою виклику функцій низького рівня. Такий підхід є основою об'єктно-орієнтованого програмування. Це дозволяє працювати з об'єктами, не вдаючись у особливості їх реалізації. У кожному конкретному випадку застосовується той чи інший підхід: інкапсуляція, поліморфізм або успадкування. Наприклад, при необхідності звернутися до прихованих даними об'єкта, слід скористатися інкапсуляцією, створивши, так звану, функцію доступу або властивість. Абстракція даних - популярна і в загальному невірно визначається техніка програмування. Фундаментальна ідея полягає в поділі несуттєвих деталей реалізації підпрограми і характеристик істотних для коректного її використання. Такий поділ може бути виражено через спеціальний «інтерфейс», зосереджують опис всіх можливих застосувань програми. З точки зору теорії множин, процес являє собою організацію для групи підмножин своєї множини.

**Інкапсуляція** - властивість мови програмування, що дозволяє користувачеві не замислюватися про складність реалізації використовуваного програмного компонента (що у нього всередині?), а взаємодіяти з ним за допомогою наданого інтерфейсу (публічних методів і членів), а також об'єднати і захистити життєво важливі для компонента дані. При цьому користувачеві надається тільки специфікація (інтерфейс) об'єкта. Користувач може взаємодіяти з об'єктом тільки через цей інтерфейс. Реалізується за допомогою ключового слова: public. Користувач не може використовувати закриті дані і методи. Реалізується за допомогою ключових слів: private, protected, internal. Інкапсуляція - один з чотирьох найважливіших механізмів об'єктно-орієнтованого програмування (поряд з абстракцією, поліморфізмом і спадкуванням). Приховування реалізації доцільно застосовувати в наступних випадках: гранична локалізація змін при необхідності таких змін, прогнозованість змін (які зміни в коді треба зробити для заданого зміни функціональності) і прогнозованість наслідків змін.

**Спадкування** - один з чотирьох найважливіших механізмів об'єктно-орієнтованого програмування (поряд з інкапсуляцією, поліморфізмом і абстракцією), що дозволяє описати новий клас на основі вже існуючого (батьківського), при цьому властивості і функціональність батьківського класу запозичуються новим класом. Іншими словами, клас-спадкоємець реалізує специфікацію вже існуючого класу (базовий клас). Це дозволяє користуватися об'єктами класу-спадкоємця так само, як з об'єктами базового класу. Просте наслідування: клас, від якого відбулося спадкування, називається базовим або батьківським (англ. Base class). Класи, які походять від базового, називаються нащадками, спадкоємцями або похідними класами (англ. Derived class). Деякі мови використовують абстрактні класи. Абстрактний клас - це клас, що містить хоча б один абстрактний метод, він описаний в програмі, має поля, методи і не може використовуватися для безпосереднього створення об'єкта. Тобто від абстрактного класу можна тільки наслідувати. Об'єкти створюються тільки на основі похідних класів, успадкованих від абстрактного. Наприклад, абстрактним класом може бути базовий клас «співробітник вузу», від якого успадковуються класи «аспірант», «професор» і т. Д. Так як похідні класи мають спільні поля і функції (наприклад, поле «рік народження»), то ці члени класу можуть бути описані в базовому класі. У програмі створюються об'єкти на основі класів «аспірант», «професор», але немає сенсу створювати об'єкт на основі класу «співробітник вузу».

**Множинне спадкування**

При множинному спадкуванні у класу може бути більш одного предка. В цьому випадку клас успадковує методи всіх предків. Переваги такого підходу в більшій гнучкості. Множинне спадкування реалізовано в C ++. Множинне спадкування - потенційне джерело помилок, які можуть виникнути через наявність однакових імен методів в предках. У мовах, які позиціонуються як спадкоємці C ++ (Java, C # і ін.), від множинного успадкування було вирішено відмовитися на користь інтерфейсів. Практично завжди можна обійтися без використання даного механізму. Однак, якщо така необхідність все-таки виникла, то, для вирішення конфліктів використання успадкованих методів з однаковими іменами, можливо, наприклад, застосувати операцію розширення видимості - «::» - для виклику конкретного методу конкретного батька. Спроба вирішення проблеми наявності однакових імен методів в предках була зроблена в мові Ейфель, в якому при описі нового класу необхідно явно вказувати імпортовані члени кожного з успадкованих класів і їх іменування в дочірньому класі. Більшість сучасних об'єктно-орієнтованих мов програмування (C #, Java, Delphi і ін.) Підтримують можливість одночасно успадковуватися від класу-предка і реалізувати методи декількох інтерфейсів одним і тим же класом. Цей механізм дозволяє багато в чому замінити множинне спадкування - методи інтерфейсів необхідно перевизначати явно, що виключає помилки при спадкуванні функціональності однакових методів різних класів-предків.

**Поліморфізм** - можливість об'єктів з однаковою специфікацією мати різну реалізацію. Мова програмування підтримує поліморфізм, якщо класи з однаковою специфікацією можуть мати різну реалізацію - наприклад, реалізація класу може бути змінена в процесі успадкування. Коротко зміст поліморфізму можна виразити фразою: «Один інтерфейс, безліч реалізацій». Поліморфізм - один з чотирьох найважливіших механізмів об'єктно-орієнтованого програмування (поряд з абстракцією, інкапсуляцією і спадкуванням). Поліморфізм дозволяє писати більш абстрактні програми і підвищити коефіцієнт повторного використання коду. Загальні властивості об'єктів об'єднуються в систему, яку можуть називати по-різному - інтерфейс, клас. Спільність має зовнішнє і внутрішнє вираження: зовнішня спільність проявляється як однаковий набір методів з однаковими іменами та сигнатурами (ім'ям методів і типами аргументів і їх кількістю); внутрішня спільність - однакова функціональність методів. Її можна описати інтуїтивно або висловити у вигляді строгих законів, правил, яким повинні підкорятися методи. Можливість приписувати різну функціональність одному методу (функції, операції) називається перевантаженням методу (перевантаженням функцій, перевантаженням операцій).

Оцінивши всі переваги та недоліки об’єктно-орієнтованого програмування, я зробив вибір на користь мови С++.

# 3. Аналіз та розробка алгоритму згідно індивідуального завдання.

## 

## 3.1. Опис алгоритму та оцінка його складності.

В алгоритмі наявні 3 конструкції switch, що утворюють меню консольної програми:

а) перейти до головного меню роботи з даними;

б) перейти до меню покупки;

в) завершити роботу.

При виконанні переходу до головного меню перед користувачем з’являється вікно вводу ім’я користувача, а згодом допоміжне меню вибору типу інформації (всього їх 5). Після чого користувач отримує інформацію про товар, і може подивитись інформацію, редагувати або добавляти і видаляти, якщо він адміністратор.

При виконанні переходу до меню покупки перед користувачем з’являється про доступний на продаж товар, і тоді він може додати до корзини його, після чого відкриється корзина, підрахується ціна купленого та скільки потрібно заплатити.

## 3.2. Аналіз особливостей алгоритму.

В кожній конструкції switch викликаються методи класу.

а) в першій конструкції switch при виборі завершення роботи викликаються функції очищення всіх допоміжних файлів, та їх закриття.

б) в другій конструкції switch при виборі певного товару викликається функція одного з шести дочірних класів і зчитується певний файл, після чого користувач одержує всю інформацію про певний тип товару.

в) в третій конструкції switch викликаються функції зчитування купленого товару, його сортування, підрахунок ціни купленого товару та зміни кількості товару (в залежності від того скільки товару куплено) в головних файлах, де міститься вся інформація про товари.

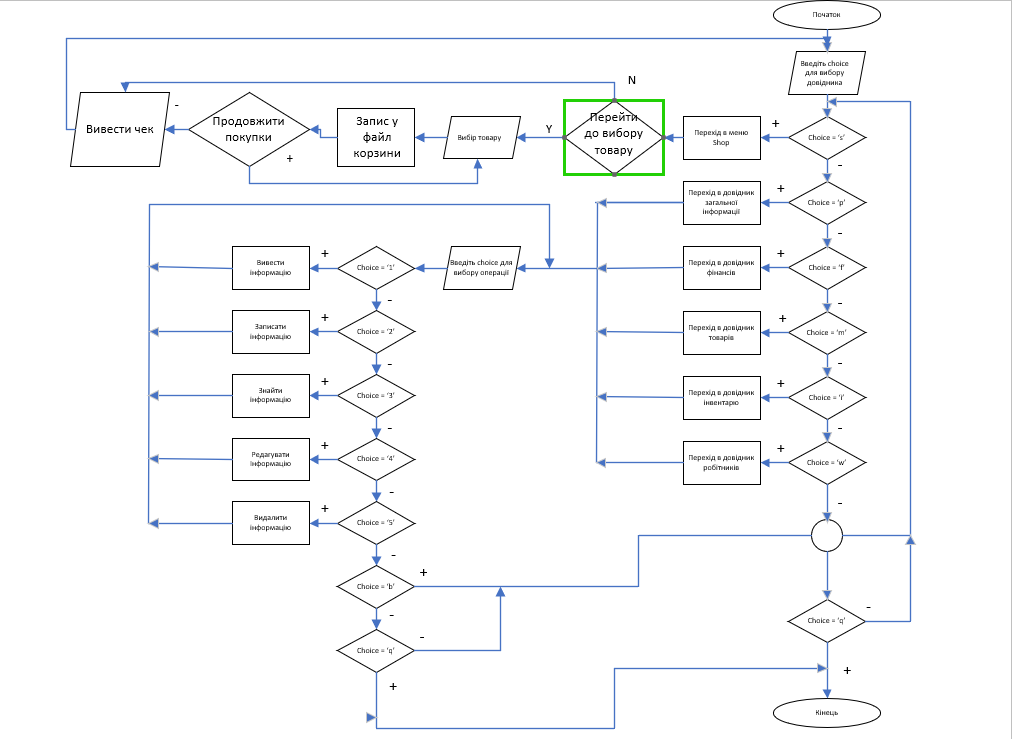


Рис. 3.1 Блок-схема алгоритму роботи програми

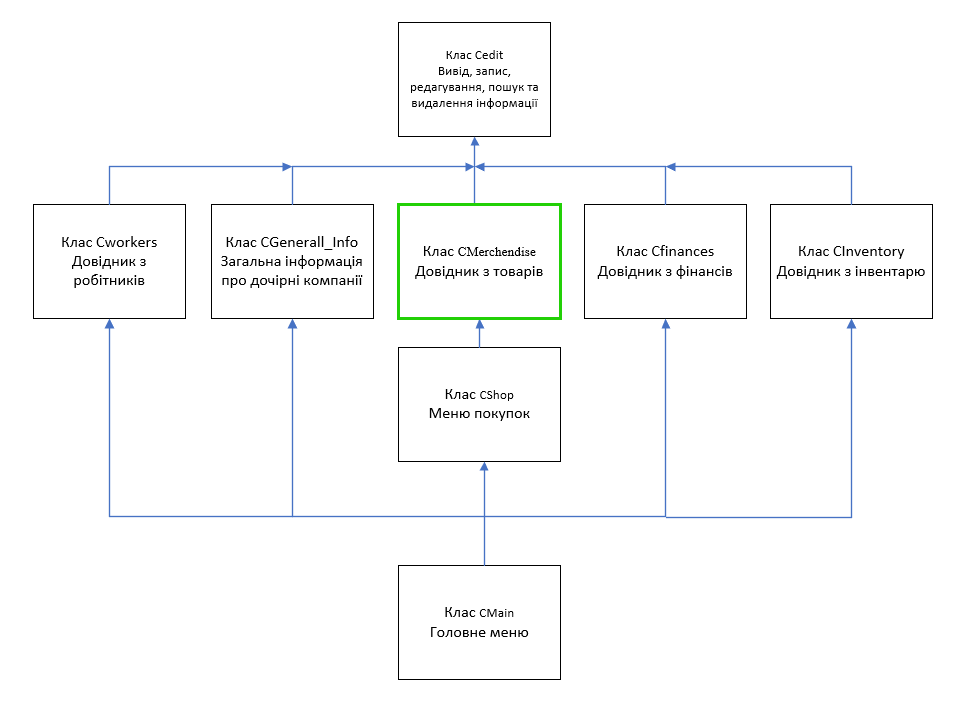


Рис. 3.2 Схема наслідування класів

# 4. Програмна реалізація алгоритму згідно індивідуального завдання.

## 4.1. Опис структур даних програми.

В програмі наявні змінні:

• цілого типу для виконання умови switch;

• цілого типу для зчитування з файлу, виконання обчислень та виведення в консоль/файл;

• масиви типу char для зчитування з файлу, виведення в консоль/файл;

• об’єкти класів для виконання методів.

• Порядок викликів функцій або діаграма класів.

## 4.2. Програма, записана псевдокодом:

* Підключення необхідних класів і заготовочних файлів.
* Створення об’єкту класу CMain.
* Режим покупки, читання інформації.
* Режим адміністратора, користувача.
* Перехід в меню вибору довідника.
* Вибір необхідного довідника для роботи.
* Створення об’єкту необхідного довідника.
* Метод виводу інформації.
* Метод запису інформації.
* Метод пошуку інформації.
* Метод редагування інформації.
* Метод видалення інформації.

# 5. Опис реалізації основних функцій/методів.

## 5.1. Програмна реалізація алгоритму згідно індивідуального завдання.

Клас **CEdit** містить стандартизовані методи для редагування інформації в довідниках : вивід, запис, пошук, редагування, видалення.

Метод **void Show\_Info(Список параметрів)**

Даний метод отримує в якості аргументів назву файлу з якого треба зчитати дані і шапку для виводу таблиці. Метод відкриває файл, зчитує з нього дані, виводить їх на екран, закриває файл і завершує свою роботу. Метод **void Write\_Info(Список параметрів)**

Даний метод отримує в якості аргументів назву файлу в який треба записати дані і фрази для запиту вводу користувача. Метод відкриває файл, запрошує у користувача ввід потрібних даних, записує ці дані в кінець файлу в рядок, закриває файл і завершує свою роботу. Метод **void Find\_Info(Список параметрів)**

Даний метод отримує в якості аргументу назву файлу в якому треба знайти дані. Метод відкриває файл, запрошує у користувача ввід потрібних даних для пошуку, виконує пошук у файлі, у випадку знаходження або незнаходження даних видає відповідне повідомлення, закриває файл і завершує свою роботу.

Метод **void Edit\_Info(Список параметрів)**

Даний метод отримує в якості аргументів назву файлу в якому треба знайти дані для заміни і назву файлу, в який запишуться відредаговані дані. Метод відкриває файл для заміни і створює файл для перезапису, запрошує у користувача ввід потрібних даних, які треба замінити і якими замінити, виконує пошук у файлі, у випадку знаходження даних які треба замінити, замінює їх, перезаписує змінений вміст файл в раніше створений файл, закриває файли, старий файл знищує, а новому присвоює назву старого файлу і завершує свою роботу. Метод **void Delete\_Info(Список параметрів)**

Даний метод отримує в якості аргументу назву файлу в якому треба видалити дані. Метод відкриває файл у режимі ios::out, що призводить до видалення змісту файлу, закриває файл і завершує свою роботу.

Класи **CGenerall\_Info,** **CMerchendise, FinancesCFinanses, CInventory, CWorkers** наслідують клас **CEdit** і описують відповідні їм довідники у вигляді методів. Ці методи приймають в якості аргументів вибір користувачем файлової операції і ім’я користувача(тільки користувач з ім’ям admin має доступ до всіх довідників і операцій). В залежності від цих аргументів метод викликає відповідні методи класу **CEdit**.

Клас **CShop** наслідую клас **CMerchendise,** та неявно **CEdit.** І описує роботу меню покупки та корзини.

Метод **void ShopMenu()**

Даний метод описує меню з якого можна перейти до меню покупок або до корзини.

Метод **void Buy()**

Даний метод описує меню з якого можна додати до файла корзини відповідний товар за його ідентифікатором.

Метод **void Basket()**

Даний метод описує роботу корзини, яка читає файл корзини та підраховує остаточну вартість покупки.

Клас **CMain** множинно наслідує класи **CGenerall\_Info, CFinanses, CInventory, CWorkers, CShop** і неявно **CEdit та** **CMerchendise**.

Метод **void StartMenu()**

Даний метод описує перше текстове меню, з якого можна перейти до меню авторизації або до меню купівлі або вийти з програми.

Метод **void AuthorizationMenu()**

Даний метод описує меню авторизації, з якого можна перейти до головного меню.

Метод **void MainMenu()**

Даний метод описує головне меню, з якого можна перейти до довідників, до попереднього меню або вийти з програми.

Метод **void info\_Menu()**

Даний метод описує меню довідників, з якого можна керувати файловими операціями, перейти до попереднього меню або вийти з програми.

Метод **int Main()**

Даний метод запрошує і приймає ввід керуючих символів користувача і відповідно до них керує методами **void StartMenu(), void AuthorizationMenu(), void MainMenu(), void info\_Menu().**

# 6. Відлагодження та тестування реалізованої програми згідно індивідуального завдання.

## 6.1. Відлагодження та тестування окремих частин програми.

Під час тестування окремих частин програм з’являлись певні недоліки та помилки. Щоб все правильно працювало потрібно було не дати можливості користувачу змушувати програму некоректно працювати. Тож всі можливі неполадки були ізольовані і опрацьовані так, щоб з’являлось певне повідомлення з певною інформацією, звідки користувач дізнавався як слід користуватися програмою, щоб уникати таких невизначеностей.

## 6.2. Відлагодження та тестування програми загалом.

Подібні кроки були виконані і з перевіркою всієї програми. Нижче наведені скріншоти, які показують коректність роботи програми:



Рис. 1 Перше текстове меню



Рис. 2 Меню авторизації

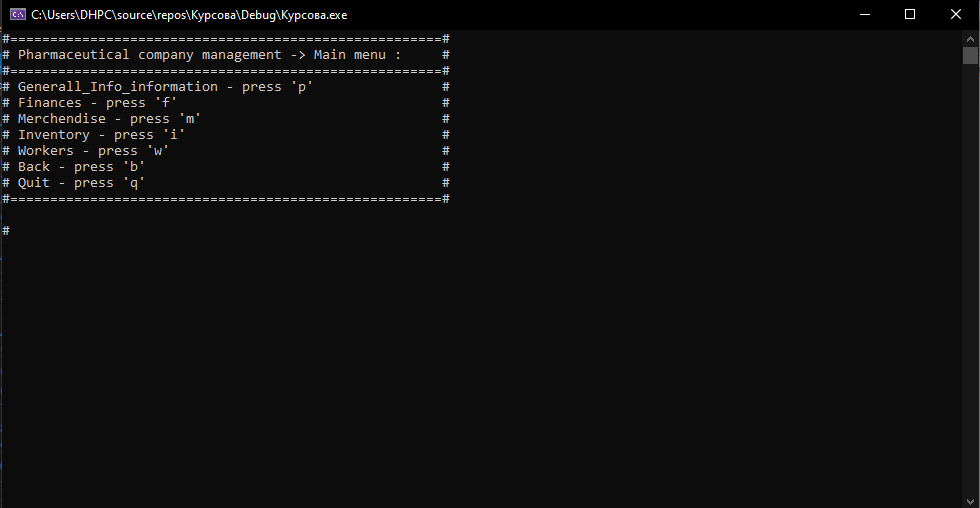


Рис. 3 Головне текстове меню

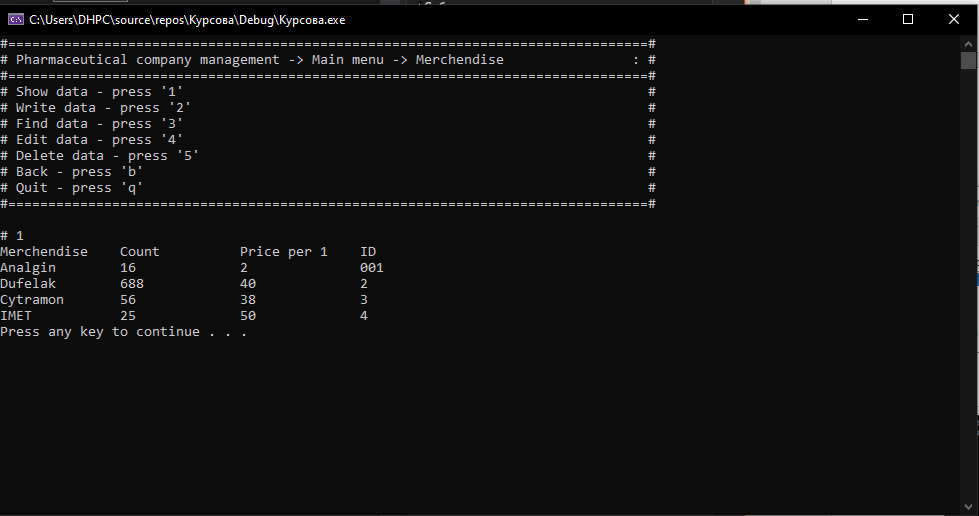


Рис. 4 Вивід інформації

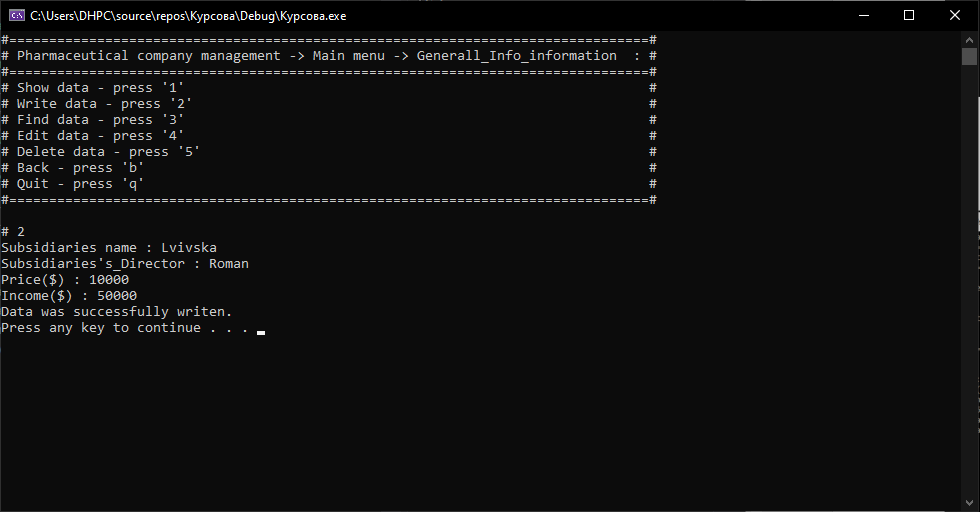


Рис. 5 Запис інформації

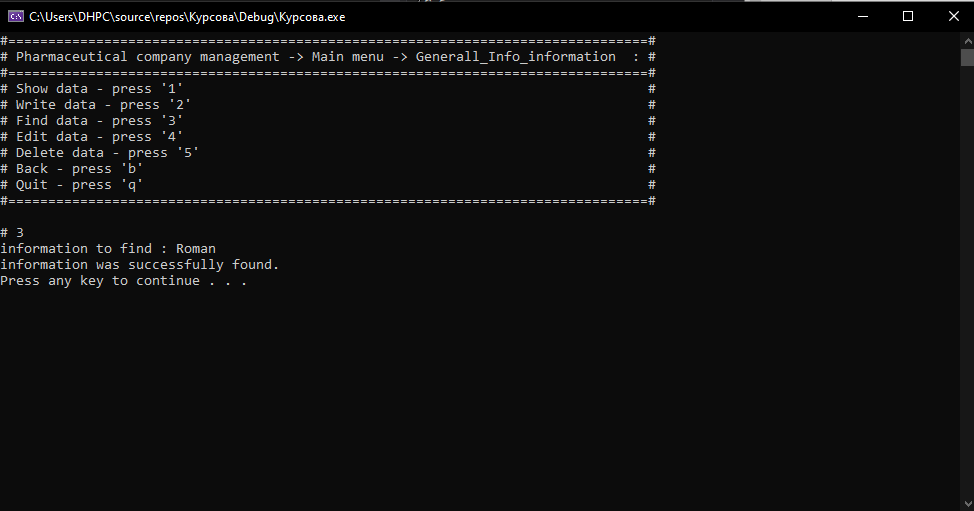


Рис. 6 Пошук інформації

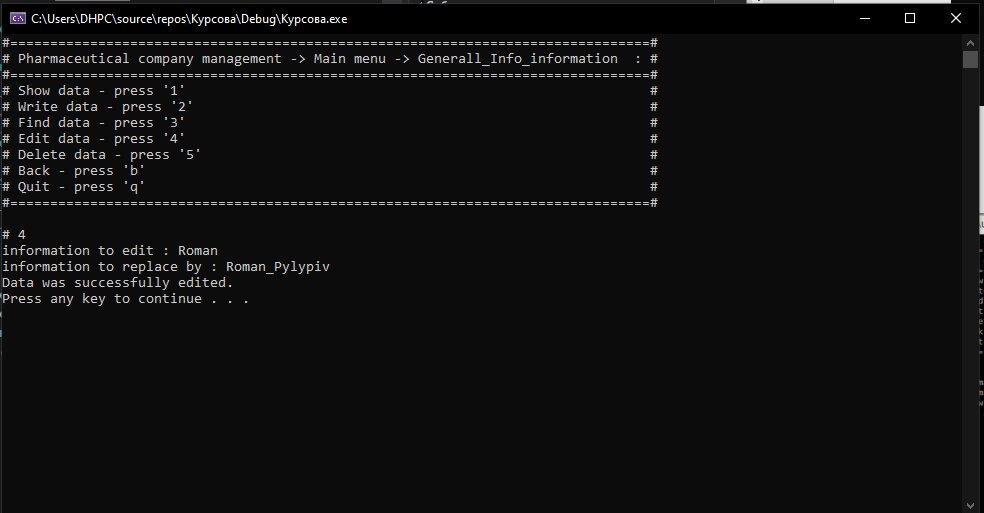


Рис. 7 Редагування інформації

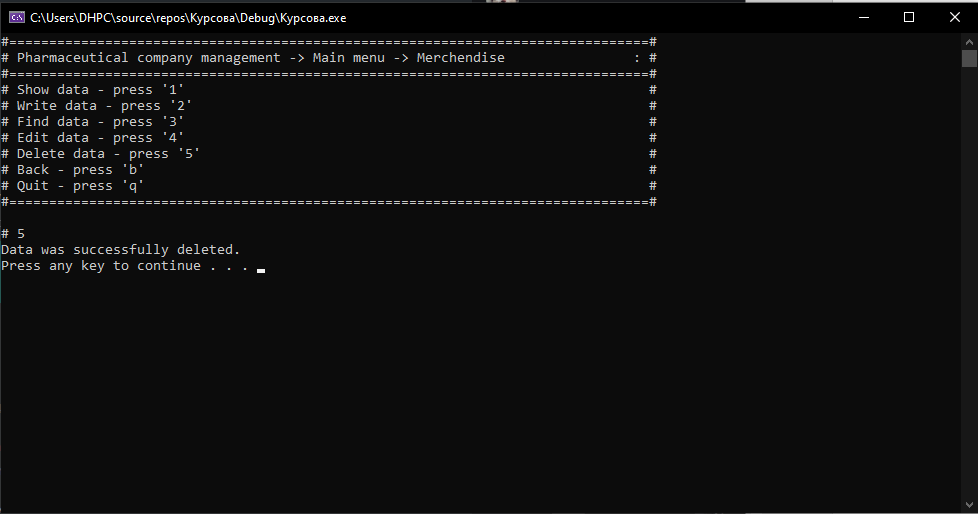


Рис. 8 Видалення інформації

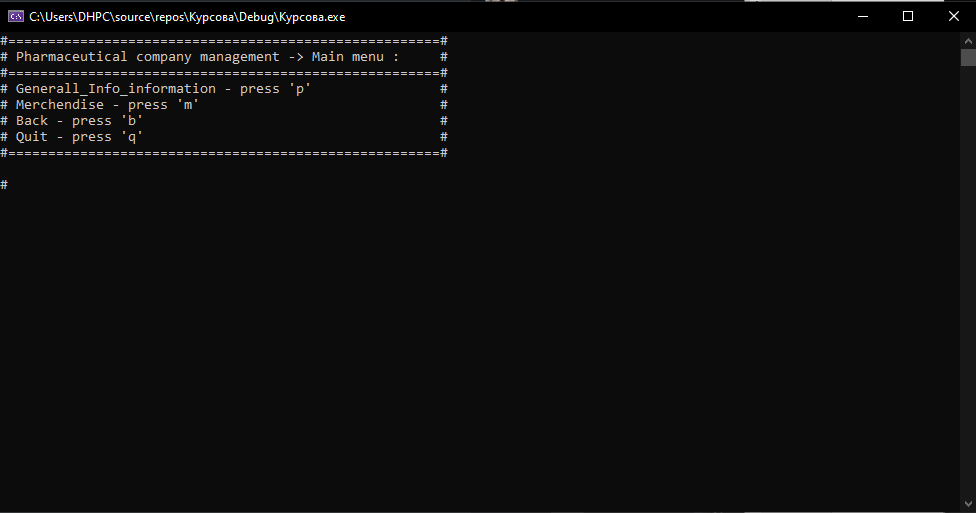


Рис. 9 Доступні довідники для користувача

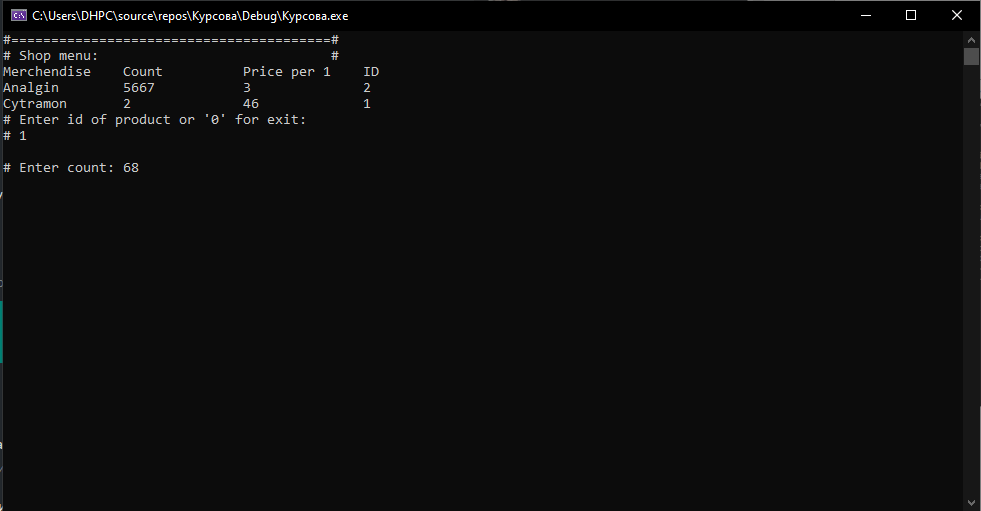


Рис. 10 Меню покупки

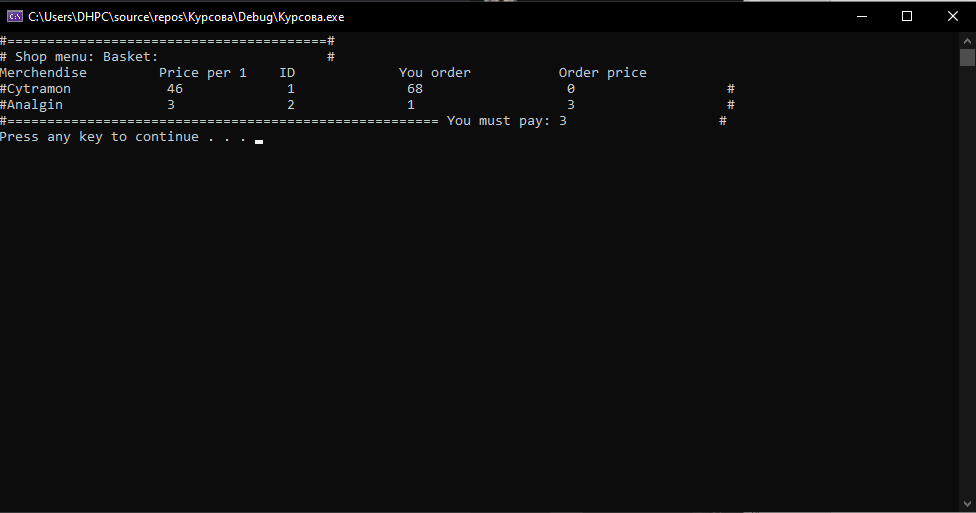


Рис. 11 Меню корзини

# Висновок

В ході виконання курсової роботи я написав програмне забезпечення для адміністрування і керування роботою кафе, дослідив загальноприйняті парадигми побудови робочого процессу відповідних структур закладів громадського харчування, засвоїв головні доктрини і особливості об’єктно-орієнтованого програмування : інкапсуляцію, наслідування, поліформізм. Вдосконалив свої навички написання та відлагодження складних програм, отримав досвід в ході роботі над крупним проектом.

# Список використаної літератури

1. Об'єктно-орієнтоване програмування [Інтернет ресурс]: [https://uk.wikipedia.org/wiki/Об%єктно-орієнтоване\_програмування](https://uk.wikipedia.org/wiki/Об%25єктно-орієнтоване_програмування)**.**

2. Основные принципы ООП: инкапсуляция, наследование, полиморфизм [Інтернет ресурс]: http://gos-it.wikia.com/wiki/ Основные\_принципы\_ООП:\_инкапсуляция,\_наследование,\_полиморфизм

3. Морозов Ю. В., Олексів М.В., Мороз І. В. Методичні вказівки до виконання циклу лабораторних робіт з дисципліни “Програмування, частина 2 (Об’єктно-орієнтоване програмування)” для студентів базового напряму 6.050102 “Комп’ютерна інженерія”, ст. 95-109.

4.Плюсы и минусы объектно-ориентированного программирования [Інтернет ресурс]: http://www.uni-vologda.ac.ru/oberon/infoart/plus&min.htm

# Додаток 1. Код програми

//CEdit.h

#pragma once

#include <string>

#include <fstream>

#include <iostream>

#include <iomanip>

using namespace std;

extern bool shop\_process;

extern int buy\_count\_max;

class CEdit

{

protected:

string info\_1, info\_2, info\_3, info\_4, findinfo\_, info\_, repinfo\_, edinfo\_;

bool isFound = false;

public:

void Show\_Info(const char fileName[], string header1, string header2, string header3, string header4);

void Write\_Info(const char fileName[], string header1, string header2, string header3, string header4);

void Find\_Info(const char fileName[]);

void Edit\_Info(const char fileName[], const char fileName1[]);

void Delete\_Info(const char fileName[]);

};

//CEdit.cpp

#include"CEdit.h"

using namespace std;

void CEdit::Show\_Info(const char fileName[], string header1, string header2, string header3, string header4) //Показати інформацію

{

fstream fin(fileName); //Відкрити файл для зчитування

while (!fin)

{

cout << "Cannot open file!\n";

exit(0);

}

cout << setw(15) << left << header1 << setw(15) << header2 << setw(15) << header3 << setw(15) << header4 << '\n'; //Вивід шапки таблиці

while (!fin.eof())

{

fin >> info\_1 >> info\_2 >> info\_3 >> info\_4; //Зчитування даних з файлу для виводу

if (!fin.eof())

cout << setw(15) << left << info\_1 << setw(15) << info\_2 << setw(15) << info\_3 << setw(15) << info\_4 << '\n'; //Вивід даних

}

if (!::shop\_process) system("pause");

fin.close(); //Закрити файл для зчитування

};

void CEdit::Write\_Info(const char fileName[], string header1, string header2, string header3, string header4) //Записати інформацію

{

cout << header1;

cin >> info\_1;

cout << header2;

cin >> info\_2;

cout << header3;

cin >> info\_3;

cout << header4;

cin >> info\_4;

fstream fout(fileName, ios::app); //Відкрити файл для запису

if (!fout)

{

cout << "Cannot open file!\n";

exit(0);

}

fout << setw(9) << left << info\_1 << ' ' << setw(9) << info\_2 << ' ' << setw(9) << info\_3 << ' ' << setw(9) << info\_4 << '\n'; //Запис

cout << "Data was successfully writen.\n";

system("pause");

fout.close();

};

void CEdit::Find\_Info(const char fileName[]) //Знайти інформацію

{

cout << "information to find : "; //Ввести інформацію для пошуку

cin >> findinfo\_;

fstream fin(fileName); //Відкрити файл для зчитування

while (!fin)

{

cout << "Cannot open file!\n";

exit(0);

}

while (!fin.eof())

{

fin >> info\_;

if (findinfo\_ == info\_)

{

cout << "information was successfully found.\n";

isFound = true;

}

}

if (!isFound)

cout << "Not found!\n";

system("pause");

fin.close(); //Закрити файл для зчитування

};

void CEdit::Edit\_Info(const char fileName[], const char fileName1[]) //Редагувати інформацію

{

cout << "information to edit : "; //Ввести слово, яке треба замінити

cin >> edinfo\_;

cout << "information to replace by : "; //Ввести слово, яким замінити

cin >> repinfo\_;

fstream fin(fileName); //Відкрити файл для зчитування

while (!fin)

{

cout << "Cannot open file!\n";

exit(0);

}

fstream fout(fileName1, ios::out); //Відкрити файл перезапису

if (!fout)

{

cout << "Cannot open file!\n";

exit(0);

}

while (!fin.eof())

{

fin >> info\_1 >> info\_2 >> info\_3 >> info\_4; //Зчитування даних з файлу для виводу

if (edinfo\_ == info\_1)

{

info\_1 = repinfo\_;

isFound = true;

}

if (edinfo\_ == info\_2)

{

info\_2 = repinfo\_;

isFound = true;

}

if (edinfo\_ == info\_3)

{

info\_3 = repinfo\_;

isFound = true;

}

if (edinfo\_ == info\_4)

{

info\_4 = repinfo\_;

isFound = true;

}

if (!fin.eof())

fout << setw(9) << left << info\_1 << ' ' << setw(9) << info\_2 << ' ' << setw(9) << info\_3 << ' ' << setw(9) << info\_4 << '\n'; //Перезапис

}

if (!isFound)

cout << "Not found!\n";

else

cout << "Data was successfully edited.\n";

system("pause");

fin.close(); //Закрити файл для рзчитування

fout.close(); //Закрити файл для перезапису

remove(fileName);

rename(fileName1, fileName);

};

void CEdit::Delete\_Info(const char fileName[]) //Видалити інформацію

{

fstream fout(fileName, ios::out); //Відкрити файл для видалення

cout << "Data was successfully deleted.\n";

system("pause");

fout.close();

};

//CGenerall\_Info.h

#pragma once

#include <iostream>

#include <iomanip>

#include <string>

#include "CEdit.h"

class CGenerall\_Info : public CEdit

{

public:

char Generall\_Info(char UserMenuChoice, string name);

};

//CGenerall\_Info.cpp

#include "CGenerall\_Info.h"

using namespace std;

char CGenerall\_Info::Generall\_Info(char UserMenuChoice, string name)

{

if (UserMenuChoice == '1') //Прочитати інформацію з файлу

{

Show\_Info("Generall\_Info\_information.txt", "Subsidiaries name", "Subsidiaries's\_Director", "Price($)", "Income($)");

return UserMenuChoice = 'p';

}

if ((UserMenuChoice == '2') && (name == "admin")) //Записати інформацію, доступно тільки адміністратору

{

Write\_Info("Generall\_Info\_information.txt", "Subsidiaries name : ", "Subsidiaries's\_Director : ", "Price($) : ", "Income($) : ");

return UserMenuChoice = 'p';

}

if (UserMenuChoice == '3') //Знайти інформацію

{

Find\_Info("Generall\_Info\_information.txt");

return UserMenuChoice = 'p';

}

if ((UserMenuChoice == '4') && (name == "admin")) //Редагувати інформацію, доступно тільки адміністратору

{

Edit\_Info("Generall\_Info\_information.txt", "Generall\_Info\_information1.txt");

return UserMenuChoice = 'p';

}

if ((UserMenuChoice == '5') && (name == "admin")) //Видалити інформацію, доступно тільки адміністратору

{

Delete\_Info("Generall\_Info\_information.txt");

return UserMenuChoice = 'p';

}

if ((UserMenuChoice == 'b') || (UserMenuChoice == 'B')) //Повернення до попереднього меню

{

return UserMenuChoice = 'm';

}

if ((UserMenuChoice == 'q') || (UserMenuChoice == 'Q')) //Вихід з програми

exit(0);

else

return UserMenuChoice = 'p';

};

//CInventory.h

#pragma once

#include <iomanip>

#include <string>

#include "CEdit.h"

class CInventory : public CEdit

{

public:

char Inventory(char UserMenuChoice, string name);

};

//CInventory.cpp

#include "CInventory.h"

using namespace std;

char CInventory::Inventory(char UserMenuChoice, string name)

{

if (UserMenuChoice == '1') //Прочитати інформацію з файлу

{

Show\_Info("Inventory.txt", "Object", "Count", "Price per 1", "ID");

return UserMenuChoice = 'i';

}

if ((UserMenuChoice == '2') && (name == "admin")) //Записати інформацію у файл, доступно тільки адміністратору

{

Write\_Info("Inventory.txt", "Object : ", "Count : ", "Price per 1 : ", "ID : ");

return UserMenuChoice = 'i';

}

if (UserMenuChoice == '3') //Знайти інформацію

{

Find\_Info("Inventory.txt");

return UserMenuChoice = 'i';

}

if ((UserMenuChoice == '4') && (name == "admin")) //Редагувати інформацію, доступно тільки адміністратору

{

Edit\_Info("Inventory.txt", "Inventory1.txt");

return UserMenuChoice = 'i';

}

if ((UserMenuChoice == '5') && (name == "admin")) //Видалити інформацію, доступно тільки адміністратору

{

Delete\_Info("Inventory.txt");

return UserMenuChoice = 'i';

}

if ((UserMenuChoice == 'b') || (UserMenuChoice == 'B')) //Повернення до попереднього меню

{

return UserMenuChoice = 'm';

}

if ((UserMenuChoice == 'q') || (UserMenuChoice == 'Q')) //Вихід з програми

exit(0);

else

return UserMenuChoice = 'i';

}

//CMerchendise.h

#pragma once

#include <iomanip>

#include <string>

#include "CEdit.h"

class CMerchendise : public CEdit

{

protected:

bool back;

public:

char Merchendise(char UserMenuChoice, string name);

};

//CMerchendise.cpp

#include "CMerchendise.h"

using namespace std;

char CMerchendise::Merchendise(char UserMenuChoice, string name)

{

if (UserMenuChoice == '1') //Прочитати інформацію з файлу

{

Show\_Info("Merchendise.txt", "Merchendise", "Count", "Price per 1", "ID");

return UserMenuChoice = 'm';

}

if ((UserMenuChoice == '2') && (name == "admin")) //Записати інформацію у файл, доступно тільки адміністратору

{

Write\_Info("Merchendise.txt", "Merchendise : ", "Count : ", "Price per 1 : ", "ID : ");

return UserMenuChoice = 'm';

}

if (UserMenuChoice == '3') //Знайти інформацію

{

Find\_Info("Merchendise.txt");

return UserMenuChoice = 'm';

}

if ((UserMenuChoice == '4') && (name == "admin")) //Редагувати інформацію, доступно тільки адміністратору

{

Edit\_Info("Merchendise.txt", "Merchendise1.txt");

return UserMenuChoice = 'm';

}

if ((UserMenuChoice == '5') && (name == "admin")) //Видалити інформацію, доступно тільки адміністратору

{

Delete\_Info("Merchendise.txt");

return UserMenuChoice = 'm';

}

if ((UserMenuChoice == 'b') || (UserMenuChoice == 'B')) //Повернення до попереднього меню

{

back = true;

return UserMenuChoice = 'm';

}

if ((UserMenuChoice == 'q') || (UserMenuChoice == 'Q')) //Вихід з програми

exit(0);

else

return UserMenuChoice = 'm';

}

//CFinances.h

#pragma once

#include <iomanip>

#include <string>

#include "CEdit.h"

class CFinances : public CEdit

{

public:

char Finances(char UserMenuChoice, string name);

};

//CFinances.cpp

#include "CFinances.h"

using namespace std;

char CFinances::Finances(char UserMenuChoice, string name)

{

if (UserMenuChoice == '1') //Прочитати інформацію з файлу

{

Show\_Info("Finances.txt", "Date", "Income($)", "Expense($)", "Total($)");

return UserMenuChoice = 'f';

}

if ((UserMenuChoice == '2') && (name == "admin")) //Записати інформацію у файл, доступно тільки адміністратору

{

Write\_Info("Finances.txt", "Date : ", "Income($) : ", "Expense($) : ", "Total($) : ");

return UserMenuChoice = 'f';

}

if (UserMenuChoice == '3') //Знайти інформацію

{

Find\_Info("Finances.txt");

return UserMenuChoice = 'f';

}

if ((UserMenuChoice == '4') && (name == "admin")) //Редагувати інформацію, доступно тільки адміністратору

{

Edit\_Info("Finances.txt", "Finances1.txt");

return UserMenuChoice = 'f';

}

if ((UserMenuChoice == '5') && (name == "admin")) //Видалити інформацію, доступно тільки адміністратору

{

Delete\_Info("Finances.txt");

return UserMenuChoice = 'f';

}

if ((UserMenuChoice == 'b') || (UserMenuChoice == 'B')) //Повернення до попереднього меню

{

return UserMenuChoice = 'm';

}

if ((UserMenuChoice == 'q') || (UserMenuChoice == 'Q')) //Вихід з програми

exit(0);

else

return UserMenuChoice = 'f';

}

// CWorkers.h

#pragma once

#include <iomanip>

#include <string>

#include "CEdit.h"

using namespace std;

class CWorkers : public CEdit

{

public:

char Workers(char UserMenuChoice, string name);

};

// CWorkers.cpp

#include "CWorkers.h"

#include "CEdit.h"

char CWorkers::Workers(char UserMenuChoice, string name)

{

if (UserMenuChoice == '1') //Прочитати інформацію з файлу

{

Show\_Info("Workers.txt", "Name", "Profession", "Salary($)", "Age");

return UserMenuChoice = 'w';

}

if ((UserMenuChoice == '2') && (name == "admin")) //Записати інформацію у файл, доступно тільки адміністратору

{

Write\_Info("Workers.txt", "Name : ", "Profession : ", "Salary($) : ", "Age : ");

return UserMenuChoice = 'w';

}

if (UserMenuChoice == '3') //Знайти інформацію

{

Find\_Info("Workers.txt");

return UserMenuChoice = 'w';

}

if ((UserMenuChoice == '4') && (name == "admin")) //Редагувати інформацію, доступно тільки адміністратору

{

Edit\_Info("Workers.txt", "Workers1.txt");

return UserMenuChoice = 'w';

}

if ((UserMenuChoice == '5') && (name == "admin")) //Видалити інформацію, доступно тільки адміністратору

{

Delete\_Info("Workers.txt");

return UserMenuChoice = 'w';

}

if ((UserMenuChoice == 'b') || (UserMenuChoice == 'B')) //Повернення до попереднього меню

{

return UserMenuChoice = 'm';

}

if ((UserMenuChoice == 'q') || (UserMenuChoice == 'Q')) //Вихід з програми

exit(0);

else

return UserMenuChoice = 'w';

}

//CShop.h

#pragma once

#include <iostream>

#include <fstream>

#include <string>

#include <iomanip>

#include "CMerchendise.h"

extern bool shop\_process;

extern int buy\_count\_max;

class CShop : public CMerchendise

{

public:

void Buy();

void Basket();

void ShopMenu();

};////

//CShop.cpp

#include "CShop.h"

using namespace std;

void CShop::Buy()

{

string name;

int count = 0, price = 0, id\_f = 0, count\_o = 0, id = 0;

fstream fin("Merchendise.txt");//Відкриття файла для зчитування

fstream fout("Basket.txt", ios::app); //Відкрити файл для запису

while (!fin)

{

cout << "Cannot open file!\n";

exit(0);

}

if (!fout)

{

cout << "Cannot open file!\n";

exit(0);

}

cout << "#========================================#" << endl;

cout << "# " << setw(38) << "Shop menu: " << " #" << "\n"; //Меню shop

::CMerchendise::Merchendise('1', "name");

cout << "# Enter id of product or '0' for exit:" << endl << "# ";

cin >> (id);

if (id == 0) ::shop\_process = false;

while (!fin.eof() && (id) != 0)

{

fin >> name >> count >> price >> id\_f;

if ((id) == id\_f)

{

cout << endl << "# Enter count: ";

cin >> count\_o;

fout << setw(9) << left << name << ' ' << setw(9) << price << ' ' << setw(9) << id\_f << ' ' << setw(9) << count\_o;//Запис

if (count\_o > count)

{

fout << setw(9) << 0 << '\n'; //Запис

}

else

{

fout << setw(9) << (price \* count\_o) << '\n'; //Запис

}

::buy\_count\_max++;

break;

}

}

fout.close();

fin.close();

}

void CShop::Basket()

{

string name;

int price = 0, id = 0, count = 0, price\_b = 0, g\_price = 0, buy\_count = ::buy\_count\_max;

system("cls"); //Очистити екран

cout << "#========================================#" << endl;

cout << left << "# " << setw(38) << "Shop menu: Basket: " << " #" << "\n"; //Меню корзини

fstream fin("Basket.txt"); //Відкрити файл для зчитування

while (!fin)

{

cout << "Cannot open file!\n";

exit(0);

}

fin.seekg(ios::beg, 0);

cout << setw(20) << left << "Merchendise" << setw(15) << "Price per 1" << setw(15) << "ID" << setw(20) << "You order" << setw(20) << "Order price" << '\n'; //Вивід шапки таблиці

while (!fin.eof() && buy\_count > 0)

{

fin >> name >> price >> id >> count >> price\_b; //Зчитування даних з файлу

cout << '#' << setw(20) << left << name << setw(15) << price << setw(15) << id << setw(20) << count << setw(20) << price\_b << '#' << '\n';

g\_price += price\_b;

buy\_count--;

}

cout << right << setfill('=') << '#' << setw(69) << "= You must pay: " << setfill(' ') << left << setw(20) << g\_price << '#' << endl;

system("pause");

fin.close(); //Закрити файл для зчитування

}

void CShop::ShopMenu()

{

system("cls"); //Очистити екран

int arr\_count\_max = 0, arr\_count = 0, id = -1;

char choice;

cout << left;

cout << "#========================================#" << endl;

cout << "# " << setw(38) << "Do you want to continue shopping or order everything in the cart? (Y/N)" << " #" << "\n" << "# "; //Меню shop

cin >> choice;

system("cls"); //Очистити екран

if ((choice == 'y') || (choice == 'Y'))

{

::shop\_process = true;

while (::shop\_process)

{

Buy();

system("cls"); //Очистити екран

}

}

Basket();

}

//CMain.h

#pragma once

#include <string>

#include <fstream>

#include <iostream>

#include <iomanip>

#include "CGenerall\_Info.h"

#include "CFinances.h"

#include "CInventory.h"

#include "CWorkers.h"

#include "CShop.h"

using namespace std;

class CMain : public CGenerall\_Info, public CFinances, public CInventory, public CWorkers, public CShop

{

private:

char UserMenuChoice;

string name;

float money;

public:

void GetChoiceMenu();

void GetUserName();

void GetUserMoney();

void StartMenu();

void AuthorizationMenu();

void MainMenu();

void info\_Menu(string dirName, string name);

int Main();

};

// CMain.cpp

#include "CMain.h"

void CMain::GetChoiceMenu()

{

cout << "# ";

cin >> UserMenuChoice;

}

void CMain::GetUserName()

{

cout << "# Enter your name : "; //Ввід ім'я користувача

cin >> name;

}

void CMain::GetUserMoney()

{

cout << "# Enter your money ($) : "; //Ввід балансу юзера

cin >> money;

}

void CMain::StartMenu()

{

system("cls"); //Очищення екрану

//Стартову меню:

cout << left;

cout << "#========================================#" << endl;

cout << "# " << setw(38) << "Pharmaceutical company management : " << " #" << "\n";

cout << "# " << setw(38) << "Main menu - press 'm'" << " #" << "\n"; //Перехід в головне меню

cout << "# " << setw(38) << "Shopping menu - press 's'" << " #" << "\n"; //Перехід в меню покупок

cout << "# " << setw(38) << "Quit - press 'q'" << " #" << "\n"; //Вихід з програми кнопкою

cout << "#========================================#" << endl << endl;

}

void CMain::AuthorizationMenu()

{

system("cls"); //Очистити екран

cout << left;

cout << "#========================================#" << endl;

cout << "# " << setw(38) << "Authorization menu: " << " #" << "\n"; //Меню авторизації

}

void CMain::MainMenu()

{

system("cls"); //Очистити екран

cout << left;

cout << "#======================================================#" << endl;

cout << "# " << setw(52) << "Pharmaceutical company management -> Main menu : " << " #" << "\n";

cout << "#======================================================#" << endl;

cout << "# " << setw(52) << "Generall\_Info\_information - press 'p'" << " #" << "\n"; //Перехід в розділ основної інформації

if (name == "admin") //Доступно тільки адміністратору

cout << "# " << setw(52) << "Finances - press 'f'" << " #" << "\n"; //Перехід в розділ фінансів

cout << "# " << setw(52) << "Merchendise - press 'm'" << " #" << "\n"; //Перехід в меню товарів

if (name == "admin") //Доступно тільки адміністратору

cout << "# " << setw(52) << "Inventory - press 'i'" << " #" << "\n"; //Перехід в меню інвентарю

if (name == "admin") //Доступно тільки адміністратору

cout << "# " << setw(52) << "Workers - press 'w'" << " #" << "\n"; //Перехід в меню працівників

cout << "# " << setw(52) << "Back - press 'b'" << " #" << "\n"; //Повернення до попереднього меню

cout << "# " << setw(52) << "Quit - press 'q'" << " #" << "\n"; //Вихід з програми

cout << "#======================================================#" << endl << endl;

}

void CMain::info\_Menu(string dirName, string name)

{

system("cls"); //Очистити екран

cout << left;

cout << "#================================================================================#" << endl;

cout << "# Pharmaceutical company management -> Main menu -> " << setw(26) << dirName << " : " << "#" << "\n";

cout << "#================================================================================#" << endl;

cout << "# " << setw(78) << "Show data - press '1'" << " #" << "\n"; //Показати інформацію

if (name == "admin") //Доступно тільки адміністратору

cout << "# " << setw(78) << "Write data - press '2'" << " #" << "\n"; //Записати інформацію

cout << "# " << setw(78) << "Find data - press '3'" << " #" << "\n"; //Знайти інформацію

if (name == "admin") //Доступно тільки адміністратору

cout << "# " << setw(78) << "Edit data - press '4'" << " #" << "\n"; //Знайти інформацію

if (name == "admin") //Доступно тільки адміністратору

cout << "# " << setw(78) << "Delete data - press '5'" << " #" << "\n"; //Видалити інформацію

cout << "# " << setw(78) << "Back - press 'b'" << " #" << "\n"; //Повернутися до попереднього меню

cout << "# " << setw(78) << "Quit - press 'q'" << " #" << "\n"; //Вихід з програми кнопкою 'q'

cout << "#================================================================================#" << endl << endl;

}

int CMain::Main()

{

fstream fout("Basket.txt", ios::out); //Відкрити файл для видалення

fout.close();

::buy\_count\_max = 0;

while (true)

{

StartMenu(); //Стартове меню

GetChoiceMenu();

if ((UserMenuChoice == 's') || (UserMenuChoice == 'S')) //Перехід в меню покупок

{

CShop shop;

shop.ShopMenu();

}

if ((UserMenuChoice == 'm') || (UserMenuChoice == 'M')) //Перехід в меню авторизації

{

AuthorizationMenu(); //Меню авторизації

GetUserName();

GetUserMoney();

}

back = false;

while ((UserMenuChoice == 'm') || (UserMenuChoice == 'M')) //Перехід в головне меню

{

MainMenu();

GetChoiceMenu();

while ((UserMenuChoice == 'p') || (UserMenuChoice == 'P')) //Перехід в розділ основної інформації

{

info\_Menu("Generall\_Info\_information", name);

GetChoiceMenu();

UserMenuChoice = Generall\_Info(UserMenuChoice, name);

}

while (((UserMenuChoice == 'f') || (UserMenuChoice == 'F')) && (name == "admin")) //Перехід в розділ фінансів

{

info\_Menu("Finances", name);

GetChoiceMenu();

UserMenuChoice = Finances(UserMenuChoice, name);

}

while (((UserMenuChoice == 'm') && (!this->back)) || ((UserMenuChoice == 'M') && (!this->back))) //Перехід в меню товарів

{

info\_Menu("Merchendise", name);

GetChoiceMenu();

UserMenuChoice = Merchendise(UserMenuChoice, name);

}

while (((UserMenuChoice == 'i') || (UserMenuChoice == 'I')) && (name == "admin")) //Перехід в меню інвентарю

{

info\_Menu("Inventory", name);

GetChoiceMenu();

UserMenuChoice = Inventory(UserMenuChoice, name);

}

while (((UserMenuChoice == 'w') || (UserMenuChoice == 'W')) && (name == "admin")) //Перехід в меню працівників

{

info\_Menu("Workers", name);

GetChoiceMenu();

UserMenuChoice = Workers(UserMenuChoice, name);

}

if ((UserMenuChoice == 'b') || (UserMenuChoice == 'B')) //Повернення до попереднього меню

break;

if ((UserMenuChoice == 'q') || (UserMenuChoice == 'Q')) //Вихід з програми

exit(0);

else

UserMenuChoice = 'm';

}

if ((UserMenuChoice == 'q') || (UserMenuChoice == 'Q')) //Вихід з програми

exit(0);

}

}

// main.cpp

#include <iostream>

#include <fstream>

#include <string>

#include <iomanip>

#include <cstdlib>

#include <locale>;

#include "windows.h";

#include "CMain.h"

using namespace std;

bool shop\_process = false;

int buy\_count\_max;

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

CMain main;

main.Main();

return 0;

}