**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:«Кафе»

*Виконано згідно методичних матеріалів*

**Виконала:**

ст. гр. КІ-15

Романішина А.Ю.

**Прийняв:**

асистент

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.

# Завдання на курсову роботу

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

**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.

**Вступ**

**Актуальність проблеми:** заклади громадського харчування, до яких належать кафе є невід’ємною частиною цивілізованого світу уже протягом кількох десятків століть. В умовах сучасного науково-технічного прогресу з’являються методи автоматизації рутинної роботи, тому я зауважив доцільним створити за їхньою допомогою програму для управління господарством, послугами, інвентарем та працівниками типового кафетерію.

**Мета роботи:** написати програмне забезпечення для адміністрування і керування роботою кафе.

**Об’єкт дослідження:** надання харчових послуг, реалізація асортименту продукції, фінансова політика і управління персоналом.

**Предмет дослідження:** загальноприйняті парадигми побудови робочого процессу відповідних структур закладів громадського харчування.

**Засоби виконання курсової роботи:** пошук та аргументація вибору технологій об’єктно-орієнтованого програмування, аналіз та розробка алгоритму, програмна реалізація алгоритму, моделювання, відлагодження та тестування реалізованої програми, аналіз продуктивності (часу роботи) реалізованої програми.

1. **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».

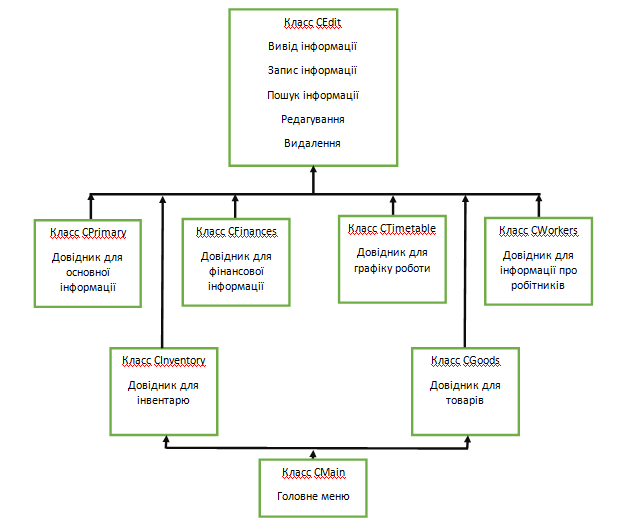
**Множинне спадкування**

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++.

1. **Аналіз та розробка алгоритму згідно індивідуального завдання.**

Рис. 2.1 Схема наслідування класів

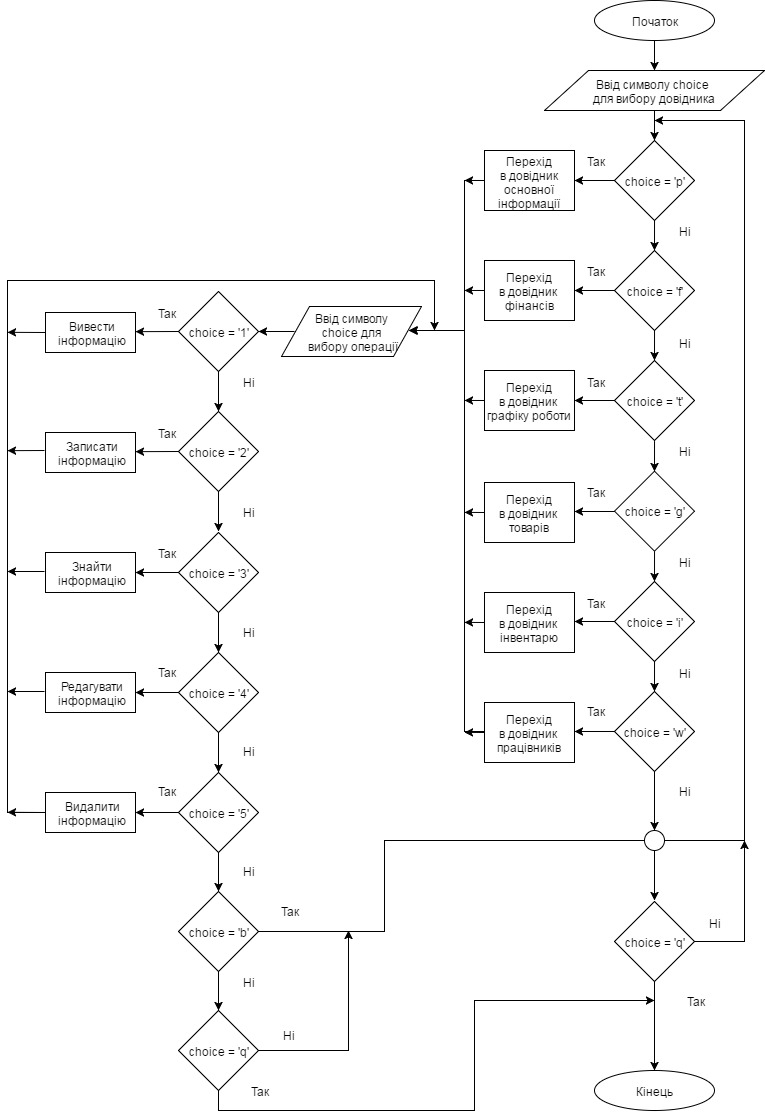


Рис. 2.2 Блок-схема алгоритму роботи програми

### Програма, записана псевдокодом:

* Підключення необхідних класів і заготовочних файлів.
* Створення об’єкту класу CMain.
* 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.

**3.** **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 **CPrimary, CFinances, CTimetable, CGoods, 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 CMain множинно наслідує класи **CPrimary, CFinances, CTimetable, CGoods, CInventory, CWorkers** і неявно **CEdit**.

Метод **void Menu0()**

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

Метод **void Menu1()**

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

Метод **void Menu2()**

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

Метод **void Menu3()**

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 Menu0(),void Menu1(),void Menu2(),void Menu3().**

**4. Відлагодження та тестування реалізованої програми згідно індивідуального завдання.**

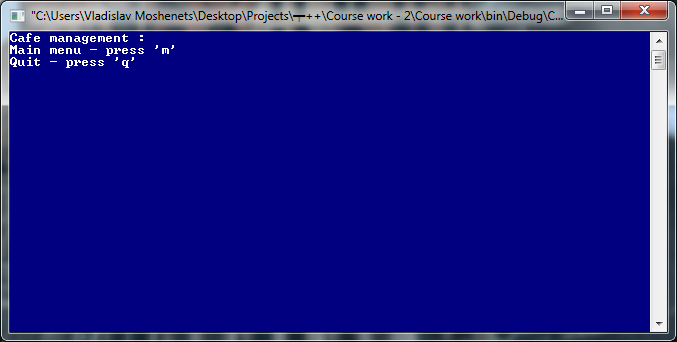
**** Технологія відлагодження програми - це послідовність дій та засоби виявлення, аналізу та виправлення помилок у програмі. Описуються використані засоби автоматизованого відлагодження та оптимізації програми (автономні та вбудовані відлагоджувачі, профайлери). Проводиться аналіз помилок, допущених в ході програмування, спосіб їх виявлення та усунення.

Рис. 1 Перше текстове меню

### 

### 

Рис. 2 Меню авторизації

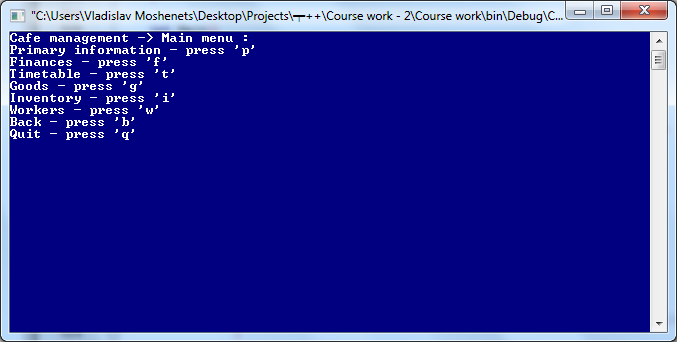


Рис. 3 Головне текстове меню

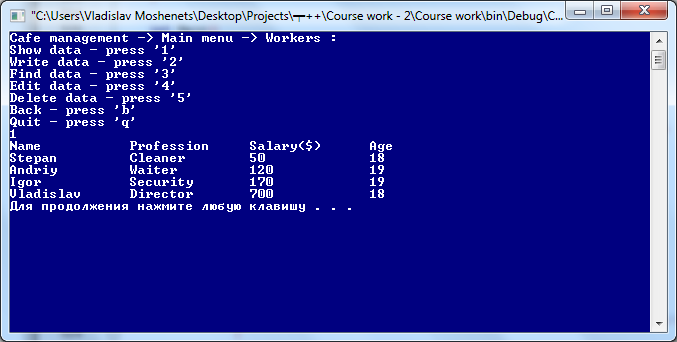


Рис. 4 Вивід інформації

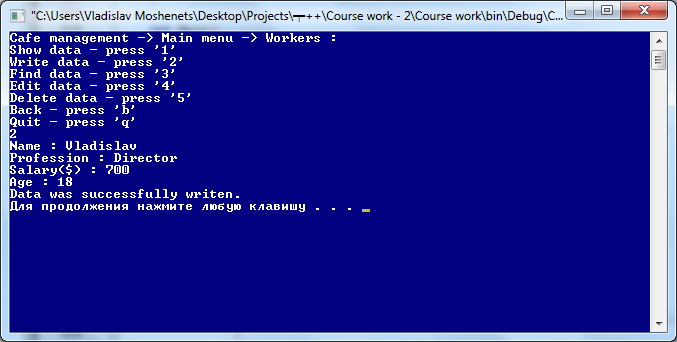


Рис. 4 Запис інформації

### 

Рис. 5 Пошук інформації

### 

Рис. 6 Редагування інформації

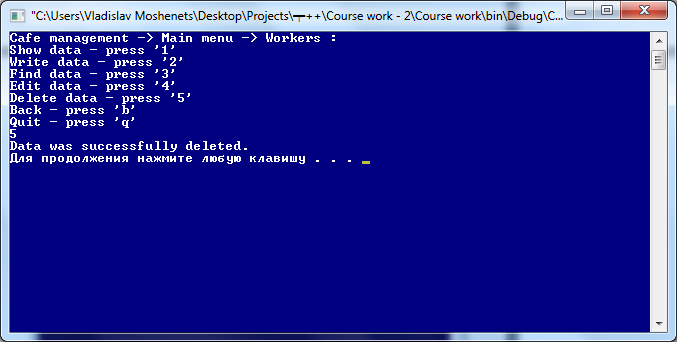


Рис. 7 Видалення інформації

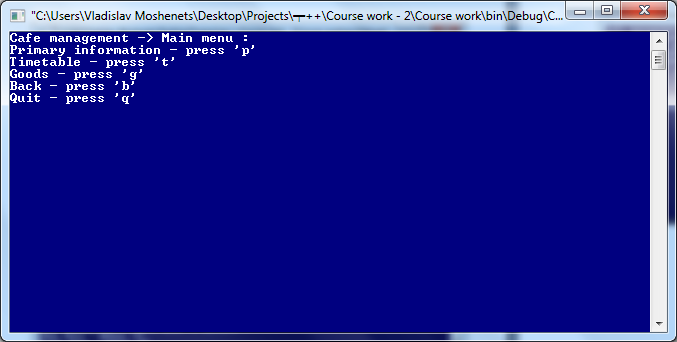


Рис. 8 Доступні довідники для користувача

### Висновок

### В ході виконання курсової роботи я написала програмне забезпечення для адміністрування і керування роботою кафе, дослідила загальноприйняті парадигми побудови робочого процесу відповідних структур закладів громадського харчування, засвоїв головні доктрини і особливості об’єктно-орієнтованого програмування : інкапсуляцію, наслідування, поліморфізм. Вдосконалила свої навички написання та відлагодження складних програм, отримала 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.

### 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

#include <iostream>

#include <fstream>

#include <string>

#include <iomanip>

#include <cstdlib>

using namespace std;

class CEdit

{

protected :

string info1, info2, info3, info4, findInfo, info, repInfo, edInfo;

bool isFound = false;

public :

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

{

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

while (!infile)

{

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 (!infile.eof())

{

infile>>info1>>info2>>info3>>info4; //Зчитування даних з файлу для виводу

if(!infile.eof())

cout<<setw(15)<<left<<info1<<setw(15)<<info2<<setw(15)<<info3<<setw(15)<<info4<<'\n'; //Вивід даних

}

system("pause");

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

}

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

{

cout << header1;

cin >> info1;

cout << header2;

cin >> info2;

cout << header3;

cin >> info3;

cout << header4;

cin >> info4;

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

if (!outfile)

{

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

exit(0);

}

outfile<<setw(9)<<left<<info1<<' '<<setw(9)<<info2<<' '<<setw(9)<<info3<<' '<<setw(9)<<info4<<'\n'; //Запис

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

system("pause");

outfile.close();

}

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

{

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

cin >> findInfo;

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

while (!infile)

{

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

exit(0);

}

while (!infile.eof())

{

infile >> info;

if (findInfo == info)

{

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

isFound = true;

}

}

if(!isFound)

cout << "Not found!\n";

system("pause");

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

}

void Edit\_Info(char fileName[], 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 infile(fileName); //Відкрити файл для зчитування

while (!infile)

{

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

exit(0);

}

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

if (!outfile)

{

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

exit(0);

}

while (!infile.eof())

{

infile>>info1>>info2>>info3>>info4; //Зчитування даних з файлу для виводу

if (edInfo == info1)

{

info1 = repInfo;

isFound = true;

}

if (edInfo == info2)

{

info2 = repInfo;

isFound = true;

}

if (edInfo == info3)

{

info3 = repInfo;

isFound = true;

}

if (edInfo == info4)

{

info4 = repInfo;

isFound = true;

}

if (!infile.eof())

outfile<<setw(9)<<left<<info1<<' '<<setw(9)<<info2<<' '<<setw(9)<<info3<<' '<<setw(9)<<info4<<'\n'; //Перезапис

}

if(!isFound)

cout << "Not found!\n";

else

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

system("pause");

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

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

remove(fileName);

rename(fileName1, fileName);

}

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

{

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

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

system("pause");

outfile.close();

}

};

class CPrimary : public CEdit

{

public :

char Primary(char choice, string name)

{

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

{

Show\_Info("Primary information.txt", "Cafe name", "Director", "Opening date", "Price($)");

return choice = 'p';

}

if ((choice == '2')&&(name == "admin")) //Записати інформацію, операція доступна тільки для адміністратора

{

Write\_Info("Primary information.txt", "Cafe name : ", "Director : ", "Opening date : ", "Price($) : ");

return choice = 'p';

}

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

{

Find\_Info("Primary information.txt");

return choice = 'p';

}

if ((choice == '4')&&(name == "admin")) //Редагувати інформацію, операція доступна тільки для адміністратора

{

Edit\_Info("Primary information.txt", "Primary information1.txt");

return choice = 'p';

}

if ((choice == '5')&&(name == "admin")) //Видалити інформацію, операція доступна тільки для адміністратора

{

Delete\_Info("Primary information.txt");

return choice = 'p';

}

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

{

return choice = 'm';

}

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

exit(0);

else

return choice = 'p';

}

};

class CFinances : public CEdit

{

public :

char Finances(char choice, string name)

{

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

{

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

return choice = 'f';

}

if ((choice == '2')&&(name == "admin")) //Записати інформацію у файл, операція доступна тільки для адміністратора

{

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

return choice = 'f';

}

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

{

Find\_Info("Finances.txt");

return choice = 'f';

}

if ((choice == '4')&&(name == "admin")) //Редагувати інформацію, операція доступна тільки для адміністратора

{

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

return choice = 'f';

}

if ((choice == '5')&&(name == "admin")) //Видалити інформацію, операція доступна тільки для адміністратора

{

Delete\_Info("Finances.txt");

return choice = 'f';

}

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

{

return choice = 'm';

}

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

exit(0);

else

return choice = 'f';

}

};

class CTimetable : public CEdit

{

public :

char Timetable(char choice, string name)

{

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

{

Show\_Info("Timetable.txt", "Day", "Date", "Opening", "Closing");

return choice = 't';

}

if ((choice == '2')&&(name == "admin")) //Записати інформацію у файл, операція доступна тільки для адміністратора

{

Write\_Info("Timetable.txt", "Day : ", "Date : ", "Opening : ", "Closing : ");

return choice = 't';

}

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

{

Find\_Info("Timetable.txt");

return choice = 't';

}

if ((choice == '4')&&(name == "admin")) //Редагувати інформацію, операція доступна тільки для адміністратора

{

Edit\_Info("Timetable.txt", "Timetable1.txt");

return choice = 't';

}

if ((choice == '5')&&(name == "admin")) //Видалити інформацію, операція доступна тільки для адміністратора

{

Delete\_Info("Timetable.txt");

return choice = 't';

}

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

{

return choice = 'm';

}

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

exit(0);

else

return choice = 't';

}

};

class CGoods : public CEdit

{

public :

char Goods(char choice, string name)

{

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

{

Show\_Info("Goods.txt", "Goods", "Number(pcs)", "Price($)", "ID");

return choice = 'g';

}

if ((choice == '2')&&(name == "admin")) //Записати інформацію у файл, операція доступна тільки для адміністратора

{

Write\_Info("Goods.txt", "Goods : ", "Number(pcs) : ", "Price($) : ", "ID : ");

return choice = 'g';

}

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

{

Find\_Info("Goods.txt");

return choice = 'g';

}

if ((choice == '4')&&(name == "admin")) //Редагувати інформацію, операція доступна тільки для адміністратора

{

Edit\_Info("Goods.txt", "Goods1.txt");

return choice = 'g';

}

if ((choice == '5')&&(name == "admin")) //Видалити інформацію, операція доступна тільки для адміністратора

{

Delete\_Info("Goods.txt");

return choice = 'g';

}

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

{

return choice = 'm';

}

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

exit(0);

else

return choice = 'g';

}

};

class CInventory : public CEdit

{

public :

char Inventory(char choice, string name)

{

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

{

Show\_Info("Inventory.txt", "Object", "Number(pcs)", "Price($)", "ID");

return choice = 'i';

}

if ((choice == '2')&&(name == "admin")) //Записати інформацію у файл, операція доступна тільки для адміністратора

{

Write\_Info("Inventory.txt", "Object : ", "Number(pcs) : ", "Price($) : ", "ID : ");

return choice = 'i';

}

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

{

Find\_Info("Inventory.txt");

return choice = 'i';

}

if ((choice == '4')&&(name == "admin")) //Редагувати інформацію, операція доступна тільки для адміністратора

{

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

return choice = 'i';

}

if ((choice == '5')&&(name == "admin")) //Видалити інформацію, операція доступна тільки для адміністратора

{

Delete\_Info("Inventory.txt");

return choice = 'i';

}

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

{

return choice = 'm';

}

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

exit(0);

else

return choice = 'i';

}

};

class CWorkers : public CEdit

{

public :

char Workers(char choice, string name)

{

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

{

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

return choice = 'w';

}

if ((choice == '2')&&(name == "admin")) //Записати інформацію у файл, операція доступна тільки для адміністратора

{

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

return choice = 'w';

}

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

{

Find\_Info("Workers.txt");

return choice = 'w';

}

if ((choice == '4')&&(name == "admin")) //Редагувати інформацію, операція доступна тільки для адміністратора

{

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

return choice = 'w';

}

if ((choice == '5')&&(name == "admin")) //Видалити інформацію, операція доступна тільки для адміністратора

{

Delete\_Info("Workers.txt");

return choice = 'w';

}

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

{

return choice = 'm';

}

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

exit(0);

else

return choice = 'w';

}

};

class CMain : public CPrimary, public CFinances, public CTimetable, public CGoods, public CInventory, public CWorkers

{

private :

char choice;

string name;

float money;

public :

void getChoice()

{

cin >> choice;

}

void getName()

{

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

cin >> name;

}

void getMoney()

{

cout << "Enter your money ($) : "; //Ввід грошового балансу користувача

cin >> money;

}

void Menu0()

{

system("cls"); //Очистити екран

system("color 1F");

cout << "Cafe management : \n"; //Стартове меню

cout << "Main menu - press 'm'\n"; //Перехід в головне меню кнопкою 'm'

cout << "Quit - press 'q'\n"; //Вихід з програми кнопкою 'q'

}

void Menu1()

{

system("cls"); //Очистити екран

cout << "Authorization : \n"; //Меню авторизації

}

void Menu2()

{

system("cls"); //Очистити екран

cout << "Cafe management -> Main menu : \n";

cout << "Primary information - press 'p'\n"; //Перехід в розділ основної інформації кнопкою 'p'

if (name == "admin") //Операція доступна тільки для адміністратора

cout << "Finances - press 'f'\n"; //Перехід в розділ фінансів кнопкою 'f'

cout << "Timetable - press 't'\n"; //Перехід в меню графіку роботи кнопкою 't'

cout << "Goods - press 'g'\n"; //Перехід в меню товарів кнопкою 'g'

if (name == "admin") //Операція доступна тільки для адміністратора

cout << "Inventory - press 'i'\n"; //Перехід в меню інвентарю кнопкою 'i'

if (name == "admin") //Операція доступна тільки для адміністратора

cout << "Workers - press 'w'\n"; //Перехід в меню працівників кнопкою 'w'

cout << "Back - press 'b'\n"; //Повернення до попереднього меню кнопкою 'b'

cout << "Quit - press 'q'\n"; //Вихід з програми кнопкою 'q'

}

void Menu3(string dirName, string name)

{

system("cls"); //Очистити екран

cout << "Cafe management -> Main menu -> " << dirName << " : \n";

cout << "Show data - press '1'\n"; //Показати інформацію

if (name == "admin") //Операція доступна тільки для адміністратора

cout << "Write data - press '2'\n"; //Записати інформацію

cout << "Find data - press '3'\n"; //Знайти інформацію

if (name == "admin") //Операція доступна тільки для адміністратора

cout << "Edit data - press '4'\n"; //Знайти інформацію

if (name == "admin") //Операція доступна тільки для адміністратора

cout << "Delete data - press '5'\n"; //Видалити інформацію

cout << "Back - press 'b'\n"; //Повернутися до попереднього меню

cout << "Quit - press 'q'\n"; //Вихід з програми кнопкою 'q'

}

int Main()

{

while (true)

{

Menu0(); //Стартове меню

getChoice();

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

{

Menu1(); //Меню авторизації

getName();

getMoney();

}

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

{

Menu2();

getChoice();

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

{

Menu3("Primary information", name);

getChoice();

choice = Primary(choice, name);

}

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

{

Menu3("Finances", name);

getChoice();

choice = Finances(choice, name);

}

while ((choice == 't')||(choice == 'T')) //Перехід в меню графіку роботи

{

Menu3("Timetable", name);

getChoice();

choice = Timetable(choice, name);

}

while ((choice == 'g')||(choice == 'G')) //Перехід в меню товарів

{

Menu3("Goods", name);

getChoice();

choice = Goods(choice, name);

}

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

{

Menu3("Inventory", name);

getChoice();

choice = Inventory(choice, name);

}

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

{

Menu3("Workers", name);

getChoice();

choice = Workers(choice, name);

}

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

break;

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

exit(0);

else

choice = 'm';

}

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

exit(0);

}

}

};

int main()

{

CMain main;

main.Main();

return 0;

}