**Міністерство освітиі 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-орієнтованогопрограмування»**

Індивідуальне завдання: Магазин електроніки

Виконав:

студент гр. КІ-15

Колосенко Д.О.

Прийняв:

викладач каф. ЕОМ

Козак Н.Б.

Львів-2020

**Анотація**

Одна з основних завдань програміста - це боротьба зі складністю, набагато зручніше при описі можливостей майбутньої системи говорити в термінах існуючих речей, а не просто вирваними з контексту шматками коду.

ООП надає новий рівень абстракції, що дозволяє оперувати поняттями з реального світу.

Для розв’язання багатьох практичних задач з різних галузей, необхідно використовувати базові принципи об’єктно-орієнтованого програмування, оскільки вони можуть адаптувати вирішення поставлених задач до їх швидкої зміни в реальному часі. Одним з шляхів забезпечення таких вимог є організація об’єктно-орієнтованих процесів і відповідних технічних засобів їх реалізації.

Причому, ефективність застосуванняоб’єктно-орієнтованого програмування у вирішенні поставленого завдання залежить, як від продуктивності комп’ютерів, так і від розмірів і структури пам’яті, пропускної здатності каналів зв’язку, використанихоб’єктно-орієнтованихмов програмування, компіляторів, операційних систем, чисельних методів та інших математичних досліджень. Такий широкий обсяг параметрів вимагає проведення досліджень на різних рівнях: на рівні опису алгоритмів, створення спеціальних мов програмування, компіляторів, багатопроцесорних систем, неоднорідних систем, кластерів.

Для скорочення термінів розробки об’єктно-орієнтованих програм та дослідження їх роботи використовується моделювання.

Метою виконання курсової роботи є засвоєння основних методів та алгоритмів об’єктно-орієнтованихструктур, принципів побудови відповідних структур, набуття початкових практичних навиків проектування таких засобів.

**Зміст**

[1. Огляд та обґрунтовування вибору технологій об’єктно-орієнтованого програмування 4](#_Toc515704929)

[2. Аналіз та розробка алгоритму згідно індивідуального завдання 7](#_Toc515704930)

[2.1 Опис алгоритму та оцінка його складності. 7](#_Toc515704931)

[2.2 Аналіз особливостей алгоритму. 8](#_Toc515704932)

[2.3 Опис алгоритму у псевдокоді. 8](#_Toc515704933)

[2.4 Розробка блок-схеми алгоритму. 10](#_Toc515704934)

[3. Програмна реалізація алгоритму згідно індивідуального завдання 11](#_Toc515704935)

[3.1 Опис послідосної версіїї програмної реалізії 11](#_Toc515704936)

[3.1.1 Опис структур даних програми 11](#_Toc515704937)

[3.1.2 Опис реалізації основних функцій/методів 12](#_Toc515704938)

[3.2 Опис програмної реалізації алгоритму 13](#_Toc515704939)

[3.2.1 Діаграма класів і її опис 13](#_Toc515704940)

[3.2.2 Схема викликів функцій. 14](#_Toc515704941)

[4. Відлагодження та тестування реалізованої програми згідно індивідуального завдання 16](#_Toc515704942)

[4.1 Опис структури програми 16](#_Toc515704943)

[4.2 Відлагодження та тестування окремих частин програми 16](#_Toc515704944)

[4.3 Відлагодження та тестування програми в цілому 19](#_Toc515704945)

[5. Аналіз продуктивності (часу роботи) та структурної складності реалізованої прграми згідно індивідуального завдання 20](#_Toc515704946)

[5.1 Аналіз структурної складності виконання програми. 20](#_Toc515704947)

[5.2 Аналіз часу виконання програми в залежності від об'єму вхідних даних 21](#_Toc515704948)

[6. Висновки 24](#_Toc515704949)

[8. Додатки 26](#_Toc515704950)

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

В основі об'єктно-орієнтованої мови програмування лежать два основних поняття: клас та об'єкт. Об'єкт – це базове поняття в ООП, це конкретна реалізація, екземпляр класу. Об'єкт складається з трьох частин: стан (змінні стану або влаcтивості), методи (функції або операції), ім'я об'єкта. Клас - це група даних і методів або функцій для роботи з цими даними. Об'єкти з однаковими наборами змінних стану і методів, утворюють клас. Якщо об'єкти мають реалізацію з конкретного світу, то класи є абстракціями. Трохи більш складні об'єкти можуть взагалі не містити даних, а представляти процес і містити тільки функції, які реалізують цей процес. Для формування реального об'єкта необхідно мати шаблон, по прикладу якого і будується даний об'єкт.

Поняттю "об'єкт" зіставляють ряд доповнюючих один одного визначень:

· це сукупність змінних стану і пов'язаних з ними методів (операцій). Ці методи визначають як об'єкт взаємодіє з оточуючим його світом;

· це конкретна реалізація, екземпляр класу. У програмуванні відносини об'єкта і класу можна порівняти з описом змінної, де сама змінна (об'єкт) є екземпляром якого-небудь типу даних (класу);

· це відчутна реальність, що характеризується чітко визначеною поведінкою;

· особливий упізнаваний предмет, блок або сутність (реальний чи абстрактний), що має важливе функціональне призначення в даній предметній області.

Є й інша класифікація методів об'єкта, коли виділяють функції управління, реалізації, доступу та допоміжні функції. Індивідуальність об'єкта - це властивість об'єкта, що відрізняє цей об'єкт від усіх інших об'єктів. Об'єкти можуть перебувати в певних відносинах один до одного, відносини можуть бути ієрархічними.

Існує 3 основні парадигми об’єктно-орієнтованого програмування: інкапсуляція, поліморфізм та спадкування. Зазвичай вважається, що без інкапсуляції неможливо уявити собі об’єктно-орієнтоване програмування. Інкапсуляція це здатність об'єктів скривати деякі способи обробки даних (методи) та самі дані від навколишнього цифрового середовища. Історія розвитку методологій програмування спонукувана боротьбою зі складністю розробки програмного забезпечення. Складність великих програмних систем, у створенні яких бере участь відразу велика кількість розробників, зменшується, якщо на верхньому рівні не видно деталей реалізації нижніх рівнів. Власне, процедурний підхід був першим кроком на цьому шляху. Під інкапсуляцією (encapsulation, що можна перекласти по-різному, але на потрібні асоціації добре наводить слово "обволікання") розуміється приховування інформації про внутрішній устрій об'єкта, при якому робота з об'єктом може вестися тільки через його загальнодоступний (public) інтерфейс. Або це механізм програмування, об'єднуючий разом код і дані, якими він маніпулює, виключаючи як втручання ззовні, так і неправильне використання даних. В об'єктно-орієнтованій мові програмування дані і код можуть бути об'єднані в абсолютно автономний чорний ящик. Усередині такого ящика знаходяться всі необхідні дані і код. Коли код і дані зв'язуються разом подібним чином, створюється об'єкт. Іншими словами, об'єкт - це елемент, що підтримує інкапсуляцію.

Поліморфізм - також одна з трьох основних парадигм ООП. Якщо говорити коротко, поліморфізм - це здатність об'єкта використовувати методи похідного класу, який не існує на момент створення базового. Слово «поліморфізм» можна перекласти як «багато форм». Тобто це можливість використання одного і того ж імені операції або методу до об'єктів різних класів, при цьому дії, що здійснюються з об'єктами, можуть істотно різнитися. Тому можна сказати, що в одного і того ж слова багато форм.

І остання парадигма спадкування (inheritance) - стосується здатності мови дозволяти будувати нові визначення класів на основі визначень існуючих класів. По суті, спадкування дозволяє розширювати поведінку базового класу, наслідуючи основну функціональність у похідному підкласі.Спадкування являє собою процес, в ході якого один об'єкт набуває властивостей іншого об'єкта. Це дуже важливий процес, оскільки він забезпечує принцип ієрархічної класифікації. Якщо вдуматися, то велика частина знань піддається систематизації завдяки ієрархічної класифікації по низхідній.

Інформацію взято з сайту https://www.ua5.org/oop/384-osnovn-ponjattja-obktno-orntovanogo.html

# Аналіз та розробка алгоритму згідно індивідуального завдання

Індивідуальне завдання: «Магазин електроніки».

Вхідні дані взяті з Інтернет а саме: відстань між містами, відсоток припустимих витрат на працівника.

# 2.1 Опис алгоритму та оцінка його складності.

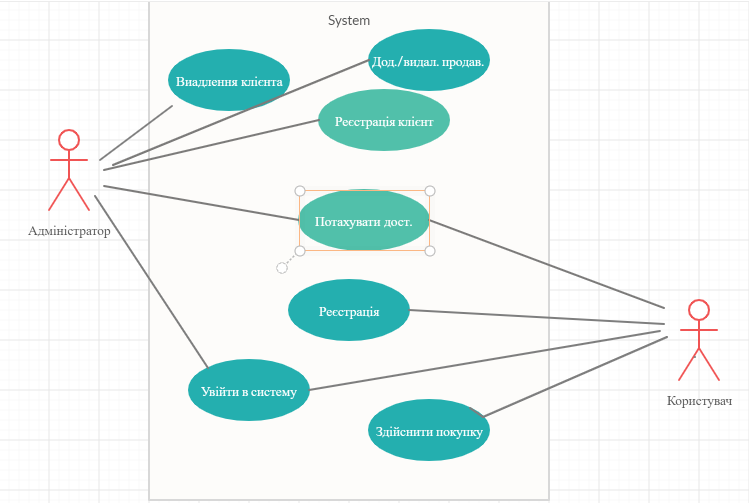
****

Рис. 2.1 Діаграма прецедентів

Діаграма включає 2 актори : користувач , адміністратор; і 7 прецедентів (варіантів використання). Користувачу доступні прецеденти: увійти в систему, здійснити покупку, зареєструватися як постійний покупець. Адміністратор володіє прецедентами : увійти в систему ,додати/звільнити працівника, порахувати доставку, реєстрація клієнта, видалення клієнта.

# 2.2 Аналіз особливостей алгоритму.

В алгоритмі наявні конструкції switch що утворюють меню консольної програми. При запуску програми буде виводитись основна інформація про радіостанцію (якщо це не перший запуск програми ) , та очікування введення користувачем чи адміністратором логіну та паролю ( інформація про логін та пароль зберігатиметься в файлах usersLogin.txt , adminLogin.txt .

При першому запуску програми адміністратор вводить інформацію про телепрограму та максимальну кількість каналів які можуть бути збережені чи працювати в радіостанції (ця інформація записується в текстовий файл ElectronicStoreInfo.txt ).

Перша конструкція switch очікуватиме вибору користувача чи адміністратора : увійти в систему чи зареєструватись. Після цього будуть оновлюватись властивості об’єктів класів Calculation.txt та Employee.txt з текстових документів Workers.txtта EmployeeList.txt за допомогою методів RefreshList(обєкти класів потрібно оновити оскільки при виході з програми викликаються деструктори класів і вся інформація записана у пам’яті змінних втрачається ).

В залежності від того хто увійшов в систему адміністратор чи користувач програма викликає різні конструктори switch, тому користувачу будуть доступні лише методи перегляду списку програми та каналів , а адміністратору додатково методи додавання та видалення програми і видалення каналу.

Обєкти класів будуть створюватись динамічно в залежності від того яка максимальна кількість композицій та працівників буде введена адміністратором ( інформація про адміністратора його логін та пароль буде введена в текстовий документ перед першим запуском програми).

# 2.3 Опис алгоритму у псевдокоді.

Програма міститиме 3головні класи: (і похідні від них)

1. Clients
2. Calculation
3. Employment

Клас Calculations міститиме всю інформацію про об’єкт композицій теоепрограм , він включатиме ID, назва телепрограми , а також міститиме методи Показати композицію (showCaltulation), додати композицію (Workers.), видалити композицію(deleteCalculation) , перезаписати інформацію (RefreshList), очистити список (CleanList), встановити ID (SetID ).

Клас Client міститиме всю інформацію про єдиний обєкт клієнта який зберігатиме в собі інформацію а саме: ім’я, к-сь покупок, сува здійснених покупок, та володітиме методами : обрахувати вартість доставки(existing ) , та здійснити покупку(showInfo ).

Клас Employment міститиме інформацію про персонал : ім’я працівника , прізвище , посада , ID. І володітиме методами : видалити працівника(deleteUser) ,оновити інформацію про працівника (refreshList) , очистити список працівників ( cleanList), показати список працівників (showEmployment).

# 2.4 Розробка блок-схеми алгоритму.

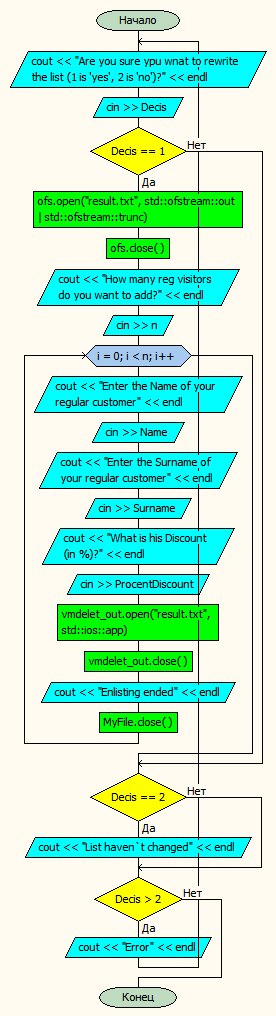
****

Рис. 2.1 Схема алгоритму роботи програми.

# Програмна реалізація алгоритму згідно індивідуального завдання

# 3.1 Опис послідосної версіїї програмної реалізії

# 3.1.1 Опис структур даних програми

Програма взаэмодіє з 7 текстовими документами: usersLogin.txt ,Сalculation.txt ,Clients.txt , Employee.txt .

Текстовий докумен CalculationInfo.txt відкривається при першому запуску програми і в подальшому на початку виконання програми , якщо цей файл пустий то користувач записує в файл кількість працівників та розмір магазину, максимальне число програм що може додати адміністратор ,суму за кілограм за метр та діаметр кабеля. Якщо ця інформація вже записана у файл то на початку запуску програми виводиться приблизні обрахунки.

Бінарний документ adminLogins.binзберігає пароль та логін адміністратора , вважається що пароль та логін записані до першого запуску програми і новий адміністратор не може зареєструватися.

Текстовий документ usersLogin.txtмістить інформацію про користувачів , а саме: ідентифікаційний номер (ID) , логін , пароль , та посаду користувача.

Текстовий документ EmployeetList.txt зберігає інформацію про добавлені адміністратором композиції : ID , Назва товару, вартість, к-сть покупок.

Текстовий документ Client.txt зберігає масив розмір якого такий як і значення максимальної кількості користувачів , цей масив показує які користувачі були видалені а які мають доступ до програми за своїм логіном та паролем. Цей тестовий документ взаємодіє з документом userLogin.txt.

# 3.1.2 Опис реалізації основних функцій/методів

1. Функція виводу інформації про радіостанції.

voidRadiostation::ShowInfo();

1. Функція зареєструватись користувачу.

void Password::registrate();

1. Функція залогуватись (увійти в систему : користувачу, адміністратору).

void Password::userLogin(int localEmployee);

void Password::adminLogin();

1. Функція показати список композицій.

void Compositions::ShowList();

1. Функція показати список користувачів.

void Employee::ShowEmployee( int numberEmployee);

1. Функція добавити композицію в плейлист.

void Employee::addTrackToUser( int userID);

1. Функція видалити композицію.(Видалення клієнта відбувається в 3 етапи : 1 – очищення файлуMusicList.txt, 2 - присвоєння треку , що потрібно видалити , інакшого ідентифікаційного номера , 3 - перезапис текстового документу з врахуванням пункту 2)

void Client::CleanList();

void Client::setID( int trekID );

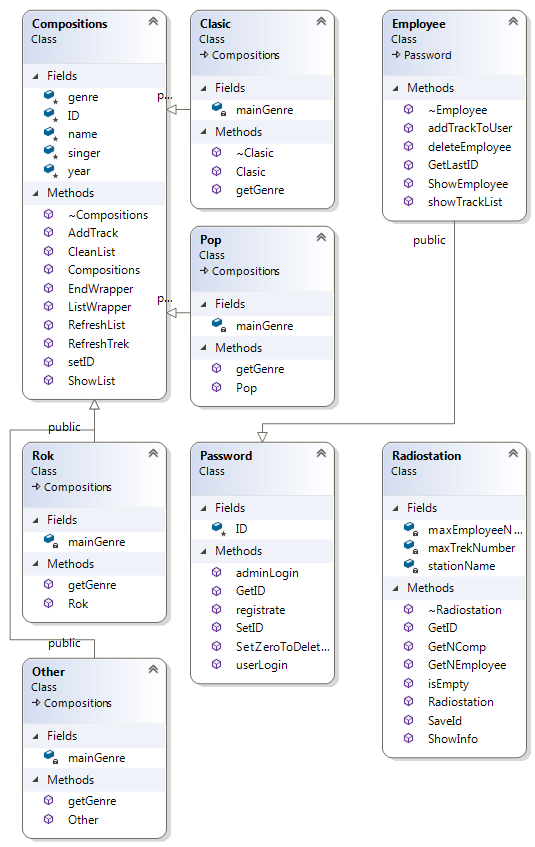
void Client::RefreshTrek();

1. Функція видалення користувача.

void Employee::deleteEmployee(int numberEmplo);

# 3.2 Опис програмної реалізації алгоритму

# 3.2.1 Діаграма класів і її опис



*Рис. 3.2.1 Діаграма класів*

На діаграмі зображено 8 класів : клас Compositionsта 4 похідні від нього класи (Pop, Rok, Other, Clasic), клас Employeeта похідний від нього класPassword, а також клас Radiostation.

# 3.2.2Схема викликів функцій.

1. Клас Calculation містить поля для зберігання інформації про придутки та збитки. Цей клас зберігає такі методи: SetPrice –встановлення цін, CLeanList– повністю очищує файл з програмами, EndWraper та ListWrapper– виводять в консоль перший та отанній рядок таблиці програм, RefreshList–обновляє дані композиції при запуску програми , ShowList–виводить на екран інформацію про обрахунки.
2. Класс Employee володіє такими функціями: addSell–додає покупку в інформацію про користувача , deleteEmployee–функція яка доступна тільки адміністратору , і дозволяє видалити користувача за його ідентифікаційним номером, ShowEmployee– показує список користувачів, GetLastID– зчитує з файлу usersLogin.txtостанній записаний ідентифікатор для того щоб знати який ідентифікатор присвоїти новому користувачеві.
3. Клас Client містить змінні які зберігають інформацію про імя клієнтів а також число покупок та їхню середню вартість. Клас містить такі методи: ShowInfo – показує інформацію про клієнта, SaveId– функція викликається в кінці програми перед виходом і зберігає в текстовий , isEmpty– перевіряє чи інформація про клієнта введена , GetNEmployeeта GetNEmployee повертають значення змінних maxEmployeeNumberта maxTrekNumber відповідно.
4. Класс Password, клас зберігає лише 1 зміннуID . Клас місить такі функції: adminLogin–функція для входу в систему адміністратору , registrar– функція для реєстрації користувача , userLogin– вхід користувача в сиситему , SetZeroToDeleteUser– функція взаємодіє з текстовим файлом deleteCounter.txtі присвоює 0 для тих користувачів які були видалені.

# Відлагодження та тестування реалізованої програми згідно індивідуального завдання

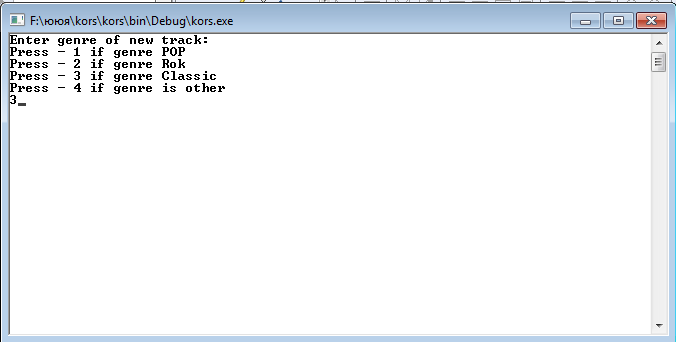
# 4.1 Опис структури програми

Для розроблення unitтестів було використано бібліотеку Boost– це бібліотека яка розширює функціональність C++. Дана бібліотека включає багато компонентів для роботи з алгоритмами контейнерами стркутурами даних а також з юніт тестами. Перевагами цієї бібліотеки є можливість запуску на будь-яких платформах та в будь-яких середовищах програмування.

Boost має помітну спрямованість на дослідження і розширюваність (метапрограмування і узагальнене програмування з активним використанням шаблонів). Завдяки ретельному підбору і контролю якості бібліотеки, включені в Boost, мають високу надійністю і продуктивність. Думки щодо використання різняться. Деякі вважають його стандартом де-факто і необхідним доповненням до STL. Деякі, навпаки, уникають всякого використання бібліотеки в проектах, оскільки це зайва залежність в проекті і для використання цих бібліотек програмістові необхідно добре знати C++, оскільки деякі частини Boost вимагають досить хорошої підготовки програміста і є вельми складними.

# 4.2Відлагодження та тестування окремих частин програми

Для коректної роботи програми необхідна наявність усіх текстових документів. Основними вузлами програми є класи Calculation та Employeeсаме їх методи опрацьовують найбільшу кількість інформації. Обєкти цих класів реалізовані по різному в програмі. Обєкти класу Clients створюються динамічно і кожен клієнт є окремим обєктом.. Основною проблемою класу Calculation це знищення його об’єктів під час виклику деструктора ( тобто після завершення програми ) , тому для оновлення інформації обєктів використовується функція RefreshList яка викликається на початку і зчитує дані з документу CalculationInfo.txt в змінні обєктів.



*Рис. 4.2.1 Меню – додати композицію.*

Клас Employeeреалізований по-іншому. В програмі створюється лише один об’єкт класу Employeeякий взаємодіє зі всіма користувачами , тому немає необхідності оновлювати інформацію про користувачів.

Юніт тести реалізовані за допомогою бібліотеки Boost, для її підключення використовую команду:

#include <boost/test/auto\_unit\_test.hpp>

Код Юніт тестів :

#define BOOST\_TEST\_LOG\_LEVEL test\_suite

#define BOOST\_TEST\_DYN\_LINK

#define BOOST\_TEST\_MODULE Test\_Storage

#include <boost/test/auto\_unit\_test.hpp>

#include "Compositions.h"

#include <Employee.h>

#include <Password.h>

#include <fstream>

#include <string>

using namespace std;

BOOST\_AUTO\_TEST\_SUITE(Test\_Storage)

BOOST\_AUTO\_TEST\_CASE(Test\_AddTrack) {

Compositions comp;

comp.CleanList();

fstream f("TelecastList.txt");

f.seekp(0, ios::end);

size\_t size = f.tellg();

BOOST\_CHECK\_EQUAL(size, 0);

}

BOOST\_AUTO\_TEST\_CASE(Test\_GetLastID) {

Password pass;

pass.registrate();

Employee emp;

int id = emp.GetLastID();

ifstream f("usersLogin.txt");

int id\_check = 0;

string name, \_pass, position;

while (f >> id\_check >> name >> \_pass >> position);

id\_check++;

BOOST\_CHECK\_EQUAL(id, id\_check);

system("pause");

}

BOOST\_AUTO\_TEST\_CASE(Test\_SeekEmployer) {

Password pass;

pass.registrate();

Employee emplo;

int id = emplo.GetLastID();

int result = emplo.SeekEmployer(id)+1;

BOOST\_CHECK\_EQUAL(result, id);

system("pause");

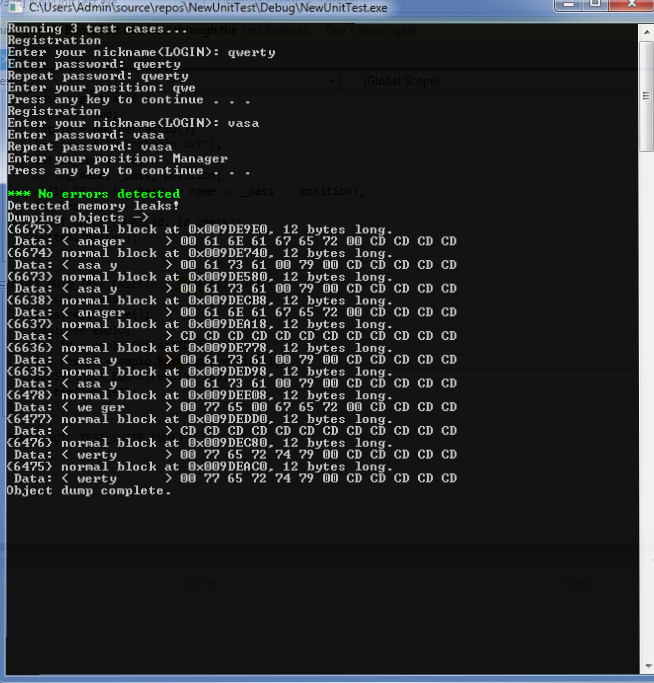
}

BOOST\_AUTO\_TEST\_SUITE\_END()

Оскільки в програмі майже всі функції реалізовані як функції типу void, тестую лише ті функції які повертають якісь значення.

# 4.3Відлагодження та тестування програми в цілому

Результати роботи Юніт тестів:



*Рис. 4.3.1 Результати роботи Юніт тестів*

Команда – “Noerrorsdetected” вказує на правильність виконання юніт- тестів.

# Аналіз продуктивності (часу роботи) та структурної складності реалізованої прграми згідно індивідуального завдання

# 5.1 Аналіз структурної складності виконання програми.

У контексті теоретичного аналізу центральним для інформатики є поняття складності. Неформально складність - це "інформаційна опірність середовища", тобто те, що заважає вирішувати завдання, принципово забороняє автоматизувати ті чи інші речі. Тому в інформатиці під поняттям структурної складності виконання програми зазвичай ховається алгоритмічна складність , тобто складність розроблення алгоритму для автоматизації деякого процесу чи задачі. Формально складність алгоритму це кількість елементарних операцій, що витрачаються алгоритмом для вирішення конкретного завдання. Складність залежить не тільки від розмірності вхідних даних, але і від самих даних. Очевидно, що чим складніше алгоритм в обчислювальному плані, тим більше часу і обчислювальних ресурсів вимагатиме його виконання.

Розрізняють тимчасову і просторову складність. Перша визначає час, необхідний на вирішення завдання заданої розмірності за допомогою даного алгоритму, а друга - кількість необхідних ресурсів (пам'яті) при тих же умовах.

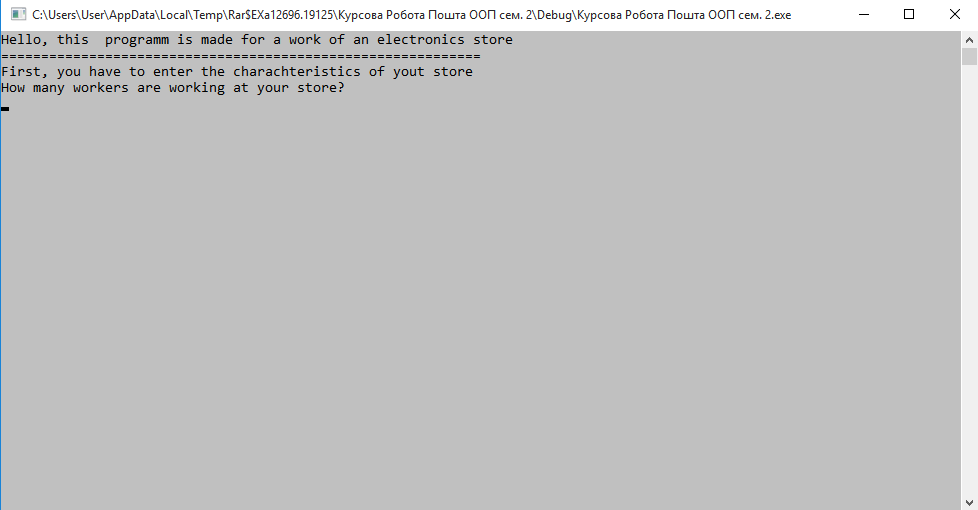
Кожен обчислювальний алгоритм може бути віднесений до одного з двох класів складності. В даному випадку це безліч завдань, для вирішення яких відомі алгоритми, схожі за трудомісткістю. У класі P обчислювальні витрати лінійно зростають зі збільшенням розмірності. Наприклад, час, необхідний на прибирання снігу, прямо пропорційно площі. Якщо її збільшити вдвічі, то і тимчасові витрати також зростуть в два рази. Клас NP включає завдання, для вирішення яких відомі тільки алгоритми, складність яких експоненціально залежить від розмірності даних. Тому вони, як правило, неефективні при роботі з великими множинами. Прикладом є задача пошуку виходу з лабіринту, тимчасові витрати на який експоненціально зростають зі збільшенням числа розгалужень.

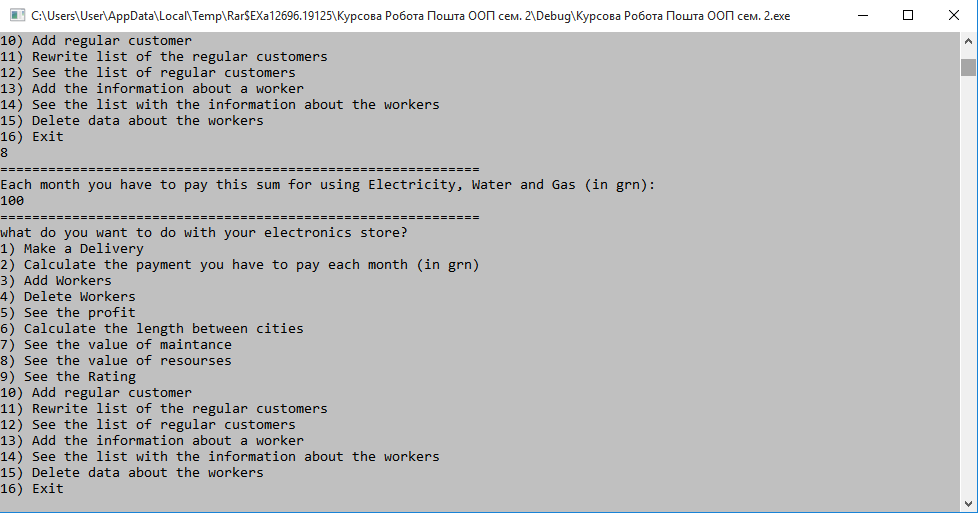
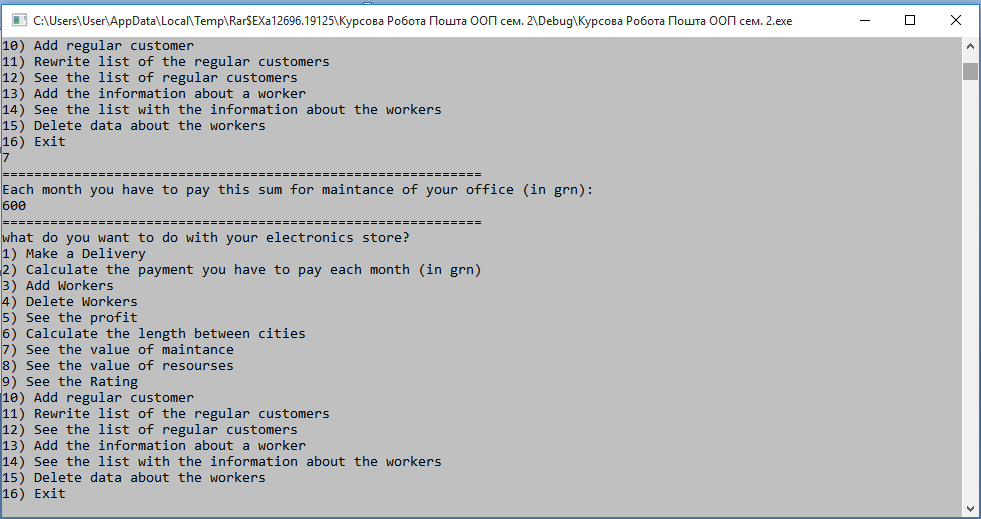
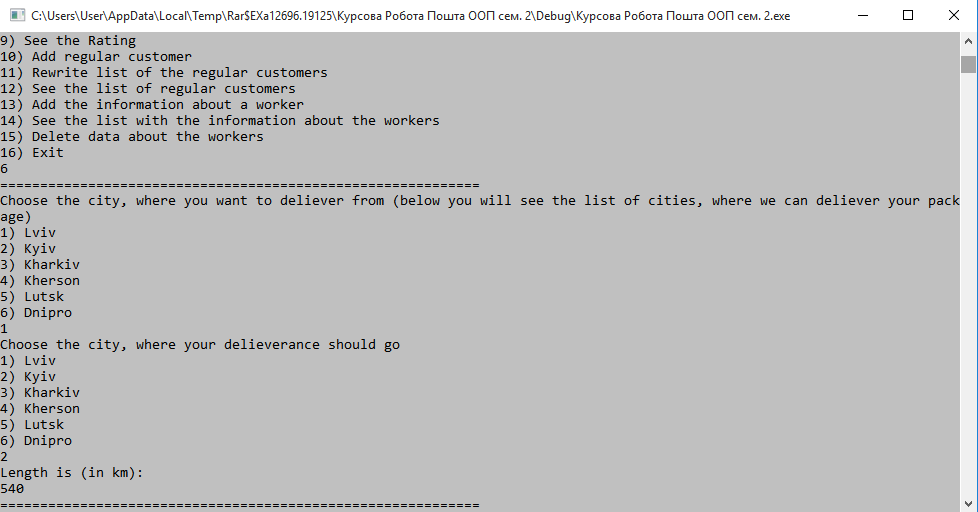
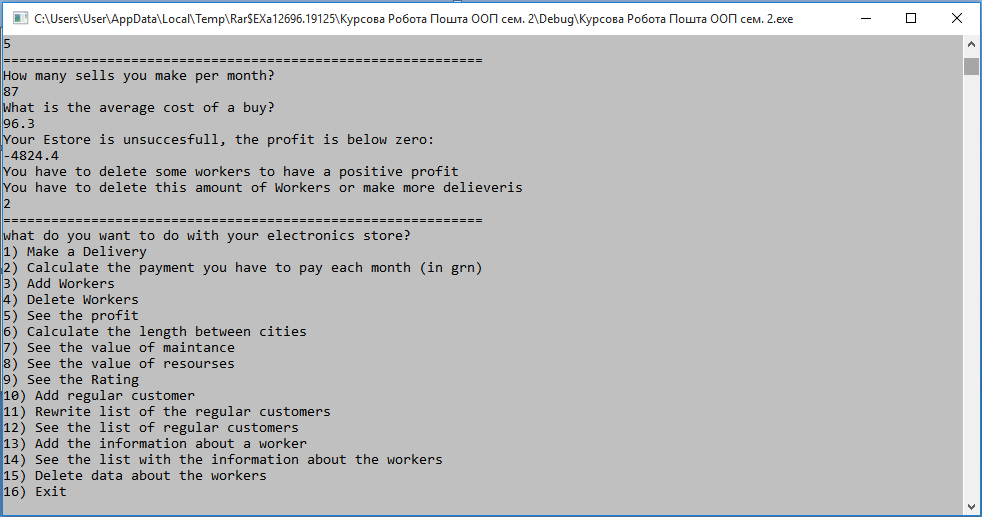
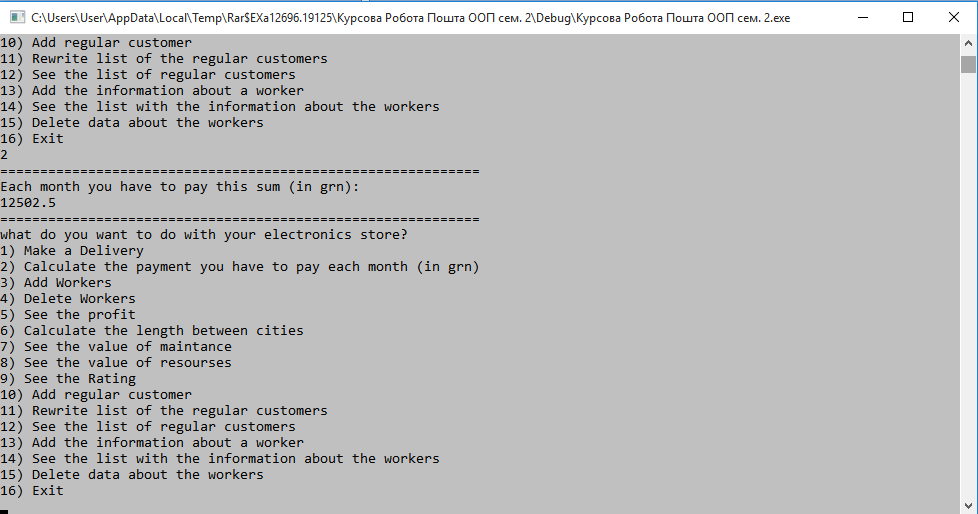
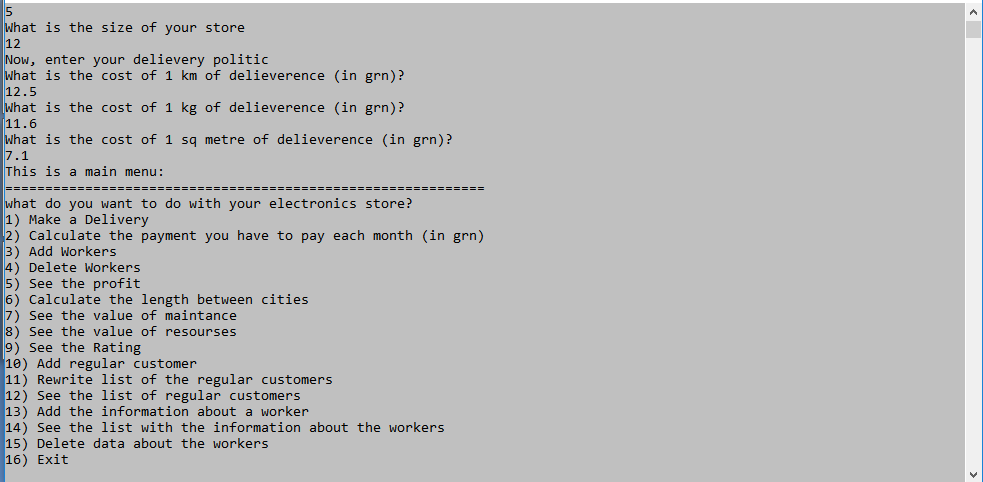
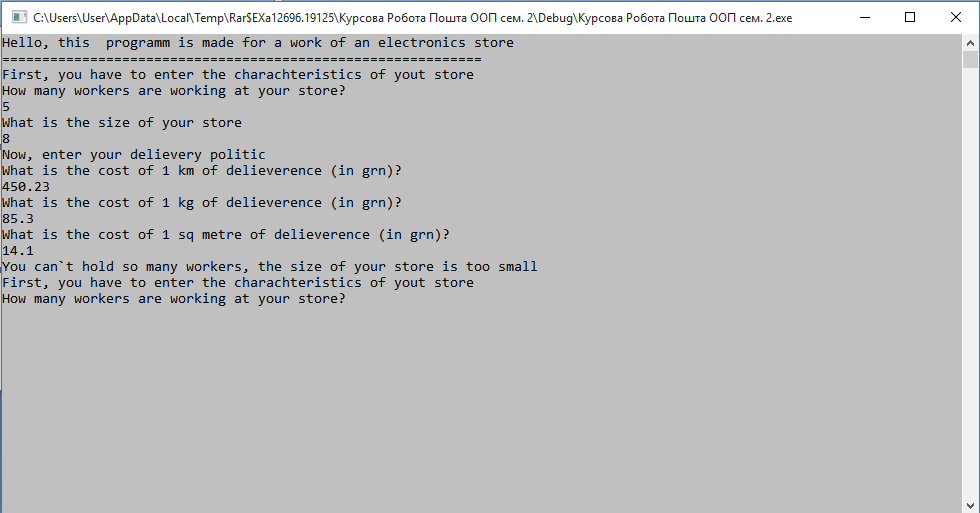
Отже структурну складність виконання програми можна вимірювати за часом виконання алгоритму та за кількістю необхідної пам’яті для збереження даних . Відповідно якщо складність велика то це означає що алгоритм потребує багато часу та великого обсягу пам’яті для вирішення задачі.

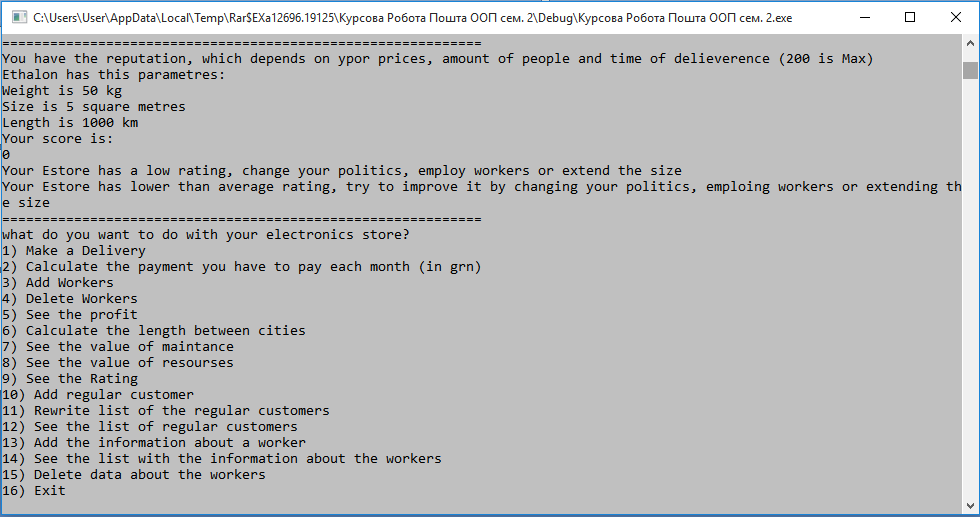
# 5.2 Аналіз часу виконання програми в залежності від об'єму вхідних даних

Всі операції всередині програми виконуються миттєво, немає затримки між переходами по меню та виведенням інформації на консоль. Читання даних з файлу та виконання операцій з числами виконується бездоганно. Є незначні затримки виконання програми під час виконання функції RefreshListякщо додано велику кількість композицій.

Результати роботи програми (скріншоти) :







# Висновки

В цій курсовій роботі я дізнався про основні принципи Об’єктно орієнтованого програмування , та навчився використовувати ці принципи. Я дізнався про основні методи , алгоритми і засоби опрацювання інформації , засоби програмування об’єктно орієнтованих структур , склад апаратних засобів та програмного забезпечення обчислювальних систем.

Навчився писати Unit-тести для перевірки правильності виконання функцій , дізнався про методи тестування і познайомився з бібліотекою Boost.

Також навчився виконувати елементарні вправи з обєктно орієнтованого програмування , описувати алгоритми поставленого перед мною завдання, використовувати класи обєкти , застосовувати модульне програмування. Засвоїв такі принципи як поліморфізм , спадкування та інкапсуляція.

1. **Використана література**
2. Джеремі Сік,The Boost Graph Library: User Guide and Reference Manual, 2006 , с 300.
3. Роберт Лафоре, Об'єктно-орієнтоване програмування в С ++,2013 , с 224.
4. Брюс Еккель, Філософія С ++. Введення в стандартний С ++, 2004 , с 126.
5. Скотт Мейерс, Ефективне використання С ++. 35 нових способів поліпшити стиль програмування, 2006 , с 45.
6. <https://www.ua5.org/oop/384-osnovn-ponjattja-obktno-orntovanogo.html>
7. Макгрегор Дж., Сайкс Д. Тестування об'єктно-орієнтованого програмного обеспеченія.- До: Диасофт, 2002.- С. 432.
8. Тамро Л. Введення в тестування програмного забезпечення.- М .: Видавничий дім-Вільямс‖, 2003.- С. 368.
9. Е. Гамм, Р. Хелм, Р. Джонсон, Д. Вліссідес «Прийоми об'єктно-орієнтованого проектування. Патерни проектування », Санкт-Петербург, Москва, 2007.- С. 366.

# Додатки

**Додаток А (Код програми)**

**//main.cpp**

#include "pch.h"

#include <iostream>

#include <string>

#include <fstream>

using namespace std;

ofstream MyFile;

class Estore

{

public:

Estore(double Size1, int Workers1)

{

double Size;

Size = Size1;

int Workers;

Workers = Workers1;

}

void Cleaning(double Size)

{

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

cout << "Each month you have to pay this sum for maintance of your office (in grn):" << endl;

double Val;

Val = Size \* 50;

cout << Val << endl;

}

void Resources( double Size, int Workers)

{

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

int res;

cout << "Each month you have to pay this sum for using Electricity, Water and Gas (in grn):" << endl;

res = (Size / 3)\*Workers \* 5;

cout << res << endl;

}

void Calculate()

{

string HomeCity;

string CheckPoint;

int Length;

int Decision1, Decision2;

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

cout << "Choose the city, where you want to deliever from (below you will see the list of cities, where we can deliever your package)" << endl;

cout << "1) Lviv" << endl;

cout << "2) Kyiv" << endl;

cout << "3) Kharkiv" << endl;

cout << "4) Kherson" << endl;

cout << "5) Lutsk" << endl;

cout << "6) Dnipro" << endl;

cin >> Decision1;

cout << "Choose the city, where your delieverance should go" << endl;

cout << "1) Lviv" << endl;

cout << "2) Kyiv" << endl;

cout << "3) Kharkiv" << endl;

cout << "4) Kherson" << endl;

cout << "5) Lutsk" << endl;

cout << "6) Dnipro" << endl;

cin >> Decision2;

if (Decision1 == 1 && Decision2 == 1)

{

Length = 0;

}

if (Decision1 == 1 && Decision2 == 2)

{

Length = 540;

}

if (Decision1 == 1 && Decision2 == 3)

{

Length = 1016;

}

if (Decision1 == 1 && Decision2 == 4)

{

Length = 888;

}

if (Decision1 == 1 && Decision2 == 5)

{

Length = 151;

}

if (Decision1 == 1 && Decision2 == 6)

{

Length = 1011;

}

if (Decision1 == 2 && Decision2 == 1)

{

Length = 0;

}

if (Decision1 == 2 && Decision2 == 2)

{

Length = 0;

}

if (Decision1 == 2 && Decision2 == 3)

{

Length = 480;

}

if (Decision1 == 2 && Decision2 == 4)

{

Length = 546;

}

if (Decision1 == 2 && Decision2 == 5)

{

Length = 400;

}

if (Decision1 == 2 && Decision2 == 6)

{

Length = 476;

}

if (Decision1 == 3 && Decision2 == 1)

{

Length = 1016;

}

if (Decision1 == 3 && Decision2 == 2)

{

Length = 480;

}

if (Decision1 == 3 && Decision2 == 3)

{

Length = 0;

}

if (Decision1 == 3 && Decision2 == 4)

{

Length = 515;

}

if (Decision1 == 3 && Decision2 == 5)

{

Length = 875;

}

if (Decision1 == 3 && Decision2 == 6)

{

Length = 217;

}

if (Decision1 == 4 && Decision2 == 1)

{

Length = 888;

}

if (Decision1 == 4 && Decision2 == 2)

{

Length = 546;

}

if (Decision1 == 4 && Decision2 == 3)

{

Length = 515;

}

if (Decision1 == 4 && Decision2 == 4)

{

Length = 0;

}

if (Decision1 == 4 && Decision2 == 5)

{

Length = 845;

}

if (Decision1 == 4 && Decision2 == 6)

{

Length = 329;

}

if (Decision1 == 5 && Decision2 == 1)

{

Length = 151;

}

if (Decision1 == 5 && Decision2 == 2)

{

Length = 400;

}

if (Decision1 == 5 && Decision2 == 3)

{

Length = 875;

}

if (Decision1 == 5 && Decision2 == 4)

{

Length = 845;

}

if (Decision1 == 5 && Decision2 == 5)

{

Length = 0;

}

if (Decision1 == 5 && Decision2 == 6)

{

Length = 871;

}

if (Decision1 == 6 && Decision2 == 1)

{

Length = 151;

}

if (Decision1 == 6 && Decision2 == 2)

{

Length = 400;

}

if (Decision1 == 6 && Decision2 == 3)

{

Length = 875;

}

if (Decision1 == 6 && Decision2 == 4)

{

Length = 845;

}

if (Decision1 == 6 && Decision2 == 5)

{

Length = 0;

}

if (Decision1 == 6 && Decision2 == 6)

{

Length = 871;

}

cout << "Length is (in km):" << endl << Length << endl;

}

void Profit(int Workers, double Size)

{

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

int n, i;

double Profit, Value, Pay;

cout << "How many sells you make per month?" << endl;

cin >> n;

cout << "What is the average cost of a buy?" << endl;

int res;

int Val;

Val = Size \* 50;

cin >> Value;

res = (Size / 3)\*Workers \* 5;

Pay = Workers \* 2500.50;

Profit = (Value\*n) - Pay - res - Val;

if (Profit > 0)

{

cout << "Your Estore is succesfull, and the profit is:" << endl << Profit << endl;

}

if (Profit < 0)

{

cout << "Your Estore is unsuccesfull, the profit is below zero:" << endl << Profit << endl;

cout << "You have to delete some workers to have a positive profit" << endl;

for ( i =0; Profit < 0; i++)

{

Profit = Profit + 2500.50;

}

cout << "You have to delete this amount of Workers or make more delieveris" << endl << i << endl;

}

}

void RegularVisitors()

{

string Name, Surname;

int ProcentDisount;

int Dec;

S:

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

cout << "Do you want to add a regular customer? (1 is 'yes', 2 is 'no')" << endl;

cin >> Dec;

if (Dec == 1)

{

int n;

cout << "How many reg visitors do you want to add?" << endl;

cin >> n;

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

{

cout << "Enter the Name of your regular customer" << endl;

cin >> Name;

cout << "Enter the Surname of your regular customer" << endl;

cin >> Surname;

cout << "What is his Discount (in %)?" << endl;

cin >> ProcentDisount;

std::ofstream vmdelet\_out;

vmdelet\_out.open("result.txt", std::ios::app);

vmdelet\_out << "Name:" << Name << endl << "Surname:" << Surname << endl << "Discount:" << ProcentDisount << endl;

vmdelet\_out.close();

cout << "Enlisting ended" << endl;

MyFile.close();

}

}

if (Dec == 2)

{

cout << "Enlisting ended" << endl;

}

if (Dec > 2)

{

cout << "Error" << endl;

goto S;

}

}

void ReWriteRegV()

{

H:

int Decis;

cout << "Are you sure ypu wnat to rewrite the list (1 is 'yes', 2 is 'no')?" << endl;

cin >> Decis;

if (Decis == 1)

{

ofstream ofs;

ofs.open("result.txt", std::ofstream::out | std::ofstream::trunc);

ofs.close();

cout << "How many reg visitors do you want to add?" << endl;

int n;

string Name, Surname, ProcentDiscount;

cin >> n;

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

{

cout << "Enter the Name of your regular customer" << endl;

cin >> Name;

cout << "Enter the Surname of your regular customer" << endl;

cin >> Surname;

cout << "What is his Discount (in %)?" << endl;

cin >> ProcentDiscount;

std::ofstream vmdelet\_out;

vmdelet\_out.open("result.txt", std::ios::app);

vmdelet\_out << "Name:" << Name << endl << "Surname:" << Surname << endl << "Discount:" << ProcentDiscount << endl;

vmdelet\_out.close();

cout << "Enlisting ended" << endl;

MyFile.close();

}

}

if (Decis == 2)

{

cout << "List haven`t changed" << endl;

}

if (Decis > 2)

{

cout << "Error" << endl;

goto H;

}

}

void PrintRegV()

{

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

ifstream file("result.txt");

cout << file.rdbuf() << endl;

cout << endl;

}

void AmountOfWorkersAdd(int Workers)

{

int WorkAdd;

double Money;

int Decision;

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

cout << "How many workers do you want to add?" << endl;

cin >> WorkAdd;

Workers = Workers + WorkAdd;

Money = Workers \* 2500.50;

cout << "You will have to pay this payment:" << endl << Money << endl;

cout << "Are you sure? (1 is Yes, 2 is No)" << endl;

cin >> Decision;

if (Decision == 2)

{

cout << "The amount of workers haven`t changed" << endl;

Workers = Workers - WorkAdd;

}

if (Decision == 1)

{

cout << "Now, the amount of workers is:" << endl << Workers << endl;

cout << "Now you have to pay this sum each month (in grn):" << endl << Money << endl;

}

Inf();

goto Menu;

}

if (Decision == 16)

{

cout << "Thanks for using, goodbye" << endl;

}

if (Decision > 16)

{

cout << "Error" << endl;

goto Menu;

}

}

}

}

void AmountOfWorkersDelete(int Workers)

{

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

int WorkDel;

double Money;

int Decision;

cout << "How many workers do you want to delete?" << endl;

cin >> WorkDel;

Workers = Workers - WorkDel;

Money = Workers \* 2500.50;

cout << "You will have to pay this payment:" << endl << Money << endl;

cout << "Are you sure? (1 is Yes, 2 is No)" << endl;

cin >> Decision;

if (Decision == 2)

{

cout << "The amount of workers haven`t changed" << endl;

Workers = Workers + WorkDel;

}

if (Decision == 1)

{

cout << "Now, the amount of workers is:" << endl << Workers << endl;

cout << "Now you have to pay this sum each month (in grn):" << endl << Money << endl;

}

}

int TimeOfDelievering(double DelWeig, double Size, double Length)

{

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

int Time;

Time = (Length + DelWeig + Size) / 5;

if (Time < 24)

{

Time = 24;

}

cout << "The probable time of delievering is (in hours):" << endl << Time << endl;

return Time;

}

void AmountOfPaymentForWorkers(int Workers)

{

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

double Money;

Money = Workers \* 2500.50;

cout << "Each month you have to pay this sum (in grn):" << endl << Money << endl;

}

void AmountOfPaymentForDelieveries(double DelWeig, double Size, double Length, double PayOLength, double PayOWeig, double PayOSize)

{

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

double Payment;

Payment = (PayOLength\*Length + PayOSize\*Size + PayOWeig\*DelWeig);

cout << "You have to pay this to deliever your package:" << endl << Payment << endl;

}

friend void Rating(double Size, double Workers, double Pay1, double Pay2, double Pay3);

};

class Human

{

public:

string Name;

string Surname;

int Age;

};

class Worker : public Human

{

public:

int Experience;

string Position;

void RegWorker()

{

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

cout << "Enter the charachteristics of a new worker" << endl;

cout << "Name?" << endl;

cin >> Name;

cout << "Surname" << endl;

cin >> Surname;

cout << "Age?" << endl;

cin >> Age;

cout << "Experience (in years)?" << endl;

cin >> Experience;

cout << "Position?" << endl;

cin >> Position;

std::ofstream vmdelet\_out;

vmdelet\_out.open("Workers.txt", std::ios::app);

vmdelet\_out << "Name:" << Name << endl << "Surname:" << Surname << endl << "Age:" << Age << endl << "Experience:" << Experience << endl << "Position:" << Position << endl;

vmdelet\_out.close();

}

void PrintWorkers()

{

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

ifstream file("Workers.txt");

cout << file.rdbuf() << endl;

cout << endl;

}

void DelInf()

{

int Dec;

G:

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

cout << "Are you sure you want to delete a list (1 is'yes', 2 is 'no')?" << endl;

cin >> Dec;

if (Dec == 1)

{

ofstream ofs;

ofs.open("Workers.txt", std::ofstream::out | std::ofstream::trunc);

ofs.close();

}

if (Dec == 2)

{

cout << "List haven`t changed" << endl;

}

if (Dec > 2)

{

cout << "Error" << endl;

goto G;

}

}

};

void Rating (double Size, double Workers, double Pay1, double Pay2, double Pay3)

{

int Time;

int Ethalon = 1000;

Time = (50 + 1000 + 5) / 5;

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

cout << "You have the reputation, which depends on ypor prices, amount of people and time of delieverence (200 is Max)" << endl;

int Score;

cout << "Ethalon has this parametres:" << endl << "Weight is 50 kg" << endl << "Size is 5 square metres" << endl << "Length is 1000 km" << endl;

double Payment;

Payment = (Pay1\*1000 + Pay3 \* 5 + Pay2 \* 50);

Score =(Ethalon + Size + Workers - (Payment + Time))/2;

if (Score > 200)

{

Score = 200;

}

if (Score < 0)

{

Score = 0;

}

cout << "Your score is:" << endl;

cout << Score << endl;

if (Score == 0)

{

cout << "Your Estore has a low rating, change your politics, employ workers or extend the size" << endl;

}

if (Score < 100)

{

cout << "Your Estore has lower than average rating, try to improve it by changing your politics, emploing workers or extending the size" << endl;

}

if (Score < 200 && Score >= 100)

{

cout << "Your Estore has a good rating, good work" << endl;

}

if (Score == 200)

{

cout << "Your Estore is one of the best, good work" << endl;

}

}

int main()

{

system("color 70");

int Decision;

cout << "Hello, this programm is made for a work of an electronics store" << endl;

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

Start:

cout << "First, you have to enter the charachteristics of yout store" << endl;

Worker Wrkr;

int Workers;

double Size;

cout << "How many workers are working at your store?" << endl;

cin >> Workers;

cout << "What is the size of your store" << endl;

cin >> Size;

cout << "Now, enter your delievery politic" << endl;

double Pay1, Pay2, Pay3;

cout << "What is the cost of 1 km of delieverence (in grn)?" << endl;

cin >> Pay1;

cout << "What is the cost of 1 kg of delieverence (in grn)?" << endl;

cin >> Pay2;

cout << "What is the cost of 1 sq metre of delieverence (in grn)?" << endl;

cin >> Pay3;

Estore PF1(Size, Workers);

if (Size < Workers\*2)

{

cout << "You can`t hold so many workers, the size of your store is too small" << endl;

goto Start;

}

else

{

cout << "This is a main menu:" << endl;

Menu:

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

cout << "what do you want to do with your electronics store?" << endl;

cout << "1) Make a Delivery" << endl;

cout << "2) Calculate the payment you have to pay each month (in grn)" << endl;

cout << "3) Add Workers" << endl;

cout << "4) Delete Workers" << endl;

cout << "5) See the profit" << endl;

cout << "6) Calculate the length between cities" << endl;

cout << "7) See the value of maintance" << endl;

cout << "8) See the value of resourses" << endl;

cout << "9) See the Rating" << endl;

cout << "10) Add regular customer" << endl;

cout << "11) Rewrite list of the regular customers" << endl;

cout << "12) See the list of regular customers" << endl;

cout << "13) Add the information about a worker" << endl;

cout << "14) See the list with the information about the workers" << endl;

cout << "15) Delete data about the workers" << endl;

cout << "16) Exit" << endl;

cin >> Decision;

if (Decision == 1)

{

double Weight, Size, Length;

cout << "Enter the charachteristics of your package" << endl;

cout << "Weight?" << endl;

cin >> Weight;

cout << "Size?" << endl;

cin >> Size;

cout << "How far are you from the deliever point?" << endl;

cin >> Length;

cout << "Your delievery is taken" << endl;

PF1.AmountOfPaymentForDelieveries(Weight, Size, Length, Pay1, Pay2, Pay3);

PF1.TimeOfDelievering(Weight, Size, Length);

goto Menu;

}

if (Decision == 2)

{

PF1.AmountOfPaymentForWorkers(Workers);

goto Menu;

}

if (Decision == 3)

{

PF1.AmountOfWorkersAdd(Workers);

goto Menu;

}

if (Decision == 4)

{

PF1.AmountOfWorkersDelete(Workers);

goto Menu;

}

if (Decision == 5)

{

PF1.Profit(Workers, Size);

goto Menu;

}

if (Decision == 6)

{

PF1.Calculate();

goto Menu;

}

if (Decision == 7)

{

PF1.Cleaning(Size);

goto Menu;

}

if (Decision == 8)

{

PF1.Resources(Size, Workers);

goto Menu;

}

if (Decision == 9)

{

Rating(Size, Workers, Pay1, Pay2, Pay3);

goto Menu;

}

if (Decision == 10)

{

PF1.RegularVisitors();

goto Menu;

}

if (Decision == 11)

{

PF1.ReWriteRegV();

goto Menu;

}

if (Decision == 12)

{

PF1.PrintRegV();

goto Menu;

}

if (Decision == 13)

{

Wrkr.RegWorker();

goto Menu;

}

if (Decision == 14)

{

Wrkr.PrintWorkers();

goto Menu;

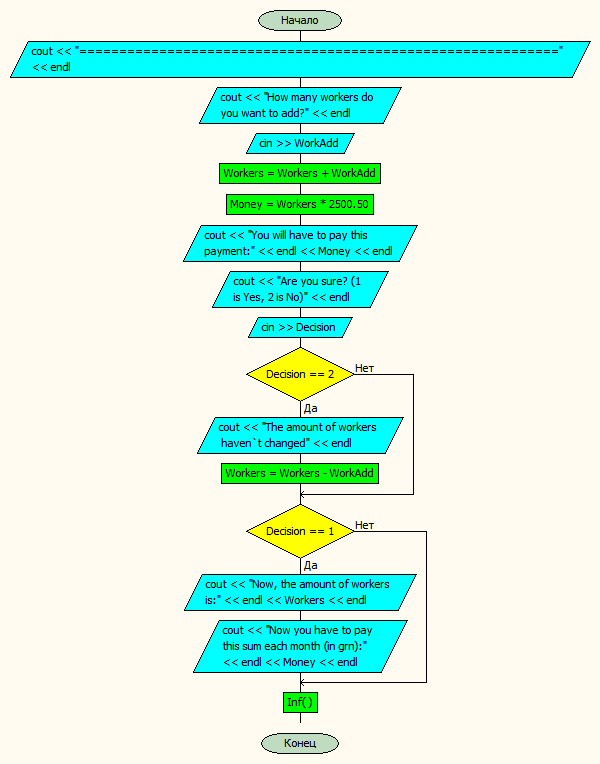
}

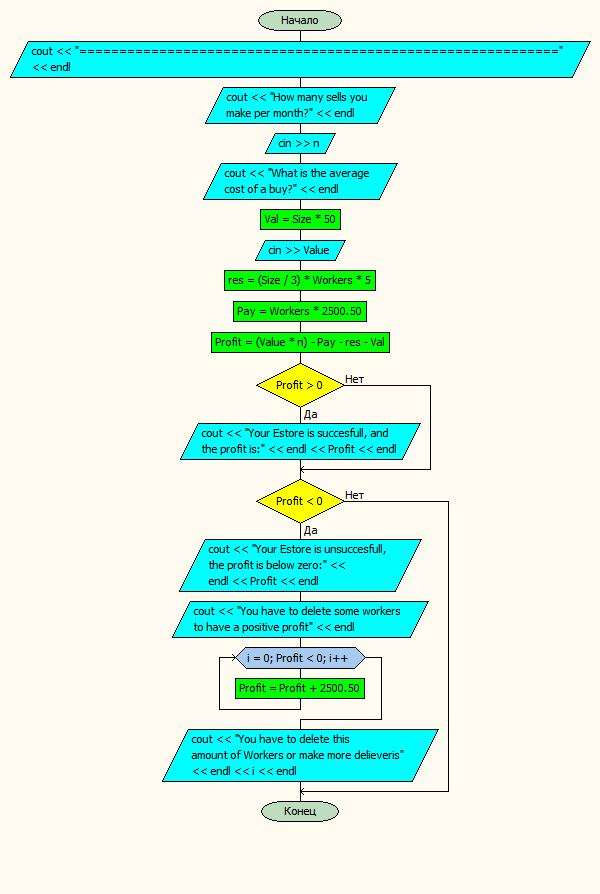
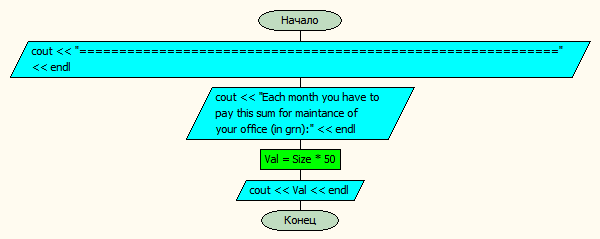
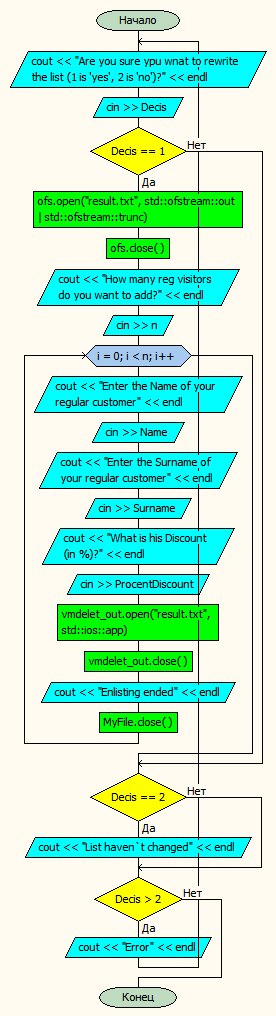
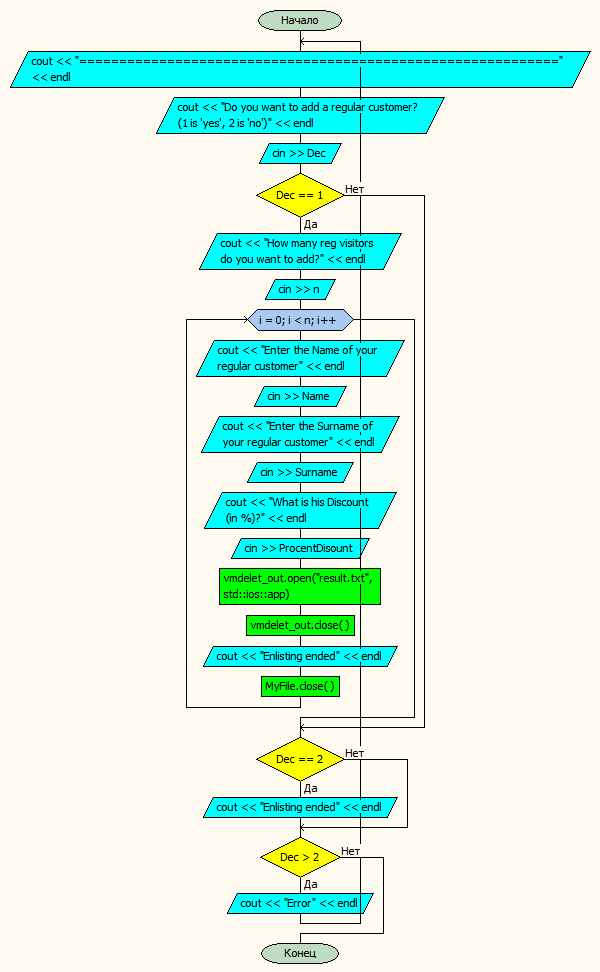
if (Decision == 15)

{

Wrkr.Del

**Додаток Б (Функціонально повна Блок-Схема)**

****

****