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**»

Завдання роботи: «ІТ-компанія»

Виконала:
ст.гр. КІ-110
Герега Р. О.
Прийняла:
Асистент кафедри ЕОМ
Гузинець Н. В.

Львів-2022

Анотація

Під час виконання курсової роботи було виконано написання програмного забезпечення на мові програмування C++ з використанням основних принципів об’єктно-орієнтованого програмування, таких як одинарне та множинне наслідування, абстрагування, інкапсуляція. Програма являє собою спрощену модель IT-компанії та включає у себе роботу з текстовими файлами, а саме: запис, зчитування, пошук, видалення та редагування вибраних даних. Інтерфейс програми виконаний графічно у програмі QT Creator. У реалізації програми було використано реальні та вигадані назви компаній, працівників тощо.

Основна мета виконання курсової роботи з дисципліни «Об'єктно-орієнтоване програмування» — закріплення на практиці вміння використовувати основні концепції об'єктно-орієнтованого підходу (ООП) — класи, інкапсуляцію, успадкування, поліморфізм, перевантаження методів і операцій, шаблони методів і класів.

Зміст

Завдання на курсову роботу

У даній курсовій роботі мені потрібно було реалізувати ІТ-компанію. У ній повинні бути використані основні принципи об’єктно-орієнтованого програмування. Усі дані було вписано випадково для демонстрації роботи програми.

Вступ

Об'єктно-орієнтоване програмування (ООП) — одна з парадигм програмування, яка розглядає програму як множину «об'єктів», що взаємодіють між собою. Основу ООП складають три основні концепції: інкапсуляція, успадкування та поліморфізм. Одною з переваг ООП є краща модульність програмного забезпечення (тисячу функцій процедурної мови, в ООП можна замінити кількома десятками класів із своїми методами). Попри те, що ця парадигма з'явилась в 1960-тих роках, вона не мала широкого застосування до 1990-тих, коли розвиток комп'ютерів та комп'ютерних мереж дозволив писати надзвичайно об'ємне і складне програмне забезпечення, що змусило переглянути підходи до написання програм. Однією з таких мов є C++. C++ (Сі-плюс-плюс) — мова програмування високого рівня з підтримкою декількох парадигм програмування: об'єктно-орієнтованої, узагальненої та процедурної. Розроблена Б'ярном Страуструпом (англ. Bjarne Stroustrup) в AT&T Bell Laboratories (Мюррей-Хілл, Нью-Джерсі) 1979 року та початково отримала назву «Сі з класами». Згодом Страуструп перейменував мову на C++ у 1983 р. Базується на мові C. Вперше описана стандартом ISO/IEC 14882:1998, найбільш актуальним же є стандарт ISO/IEC 14882:2014. У 1990-х роках C++ стала однією з найуживаніших мов програмування загального призначення. Мову використовують для системного програмування, розробки програмного забезпечення, написання драйверів, потужних серверних та клієнтських програм, а також для розробки розважальних програм таких як відеоігри. C++ суттєво вплинула на інші, популярні сьогодні, мови програмування: C# та Java.

Метою виконання цієї курсової роботи є вивчення об’єктно-орієнтованої мови C++ та практичне застосування здобутих моїх знань у цій сфері.

1.Огляд та обгрунтування вибору технологій об'єктно-орієнтованого програмування.

1.1Визначення ООП

Об’єктно-орієнтоване програмування (ООП) – це модель програмування що базується на стверджені, що програма це сукупність об’єктів які взаємодіють між собою. Кожен об’єкт в цій моделі є незалежним, і він здатний отримувати та обробляти дані, також об’єкт може відправляти дані іншим об’єктам. В ООП використано моделі успадкування, модульності, поліморфізму та інкапсуляції.

Основним поняттям в ООП є об’єкт. **Об’єкт** – це певна сукупність даних(характеристик об’єкта) та методів роботи з ними. **Клас** – це зразок об’єкту для створення об’єктуБ тобто об’єкт є копією класу.

Кожен об’єкт має процедури і функції, які служать для роботи з даними об’єкта. Ці процедури і функції називають **методами**.

Три основні парадигми на яких базується ООП:

-**Інкапсуляція**. Зміст інкапсуляції полягає в приховуванні деталей реалізації об’єкта від зовнішнього користувача, замість чого надаючи інтерфейс для взаємодії з ним.

-**Успадкування**. Це означає, що об’єкти можуть переймати деякі властивості у своїх прабатьків, це призводить до повторного використання вже написаного коду. Підкласи успадковують атрибути своїх батьківських класів, і можуть мати нові власні атрибути. Тобто утворюють ієрархію з класів, у якій від основного класу походять усі інші класи.

-**Поліморфізм**. Означає залежність поведінки від класу, в якому ця поведінка викликається, тобто, два або більше класів можуть по різному реагувати на однакові повідомлення. Це спричинене зміною в одного з класів якогось методу, шляхом запису іншого алгоритму.

1.2Фундаментальні поняття

В результаті дослідження Дебори Дж. Армстронг (англ. Deborah J. Armstrong) комп'ютерної літератури, що була видана протягом останніх 40 років, вдалось відокремити фундаментальні поняття (принципи), використані у переважній більшості визначень об'єктно-орієнтованого програмування. До них належить:

Клас

Клас визначає абстрактні характеристики деякої сутності, включаючи характеристики самої сутності та дії, які вона здатна виконувати. Наприклад, клас Кішка може характеризуватись рисами, притаманними всім Кішкам, зокрема: порода, колір хутра, здатність м’явкати. Класи вносять модульність та структурованість в об'єктно-орієнтовану програму. Також, код реалізації класу має бути досить самодостатнім. Властивості та методи класу, разом називаються його членами.

Об'єкт

Об'єкт - це програмний компонент, який зберігає дані та функції для роботи з цими даними. Об'єкти можуть відповідати об'єктам реального світу (товари у інтернет-магазині) або бути віртуальними об'єктами (для збереження тексту створюється об'єкт). Об'єктами можуть бути масиви, числа, логічні значення.

Метод

Можливості об'єкта. Оскільки Сірко — Собака, він може гавкати. Тому гавкати() є одним із методів об'єкта Сірко. Він може мати й інші методи, зокрема: місце(), або їсти(). В межах програми, використання методу має впливати лише на один об'єкт; всі Собаки можуть гавкати, але треба щоб гавкав лише один окремий собака.

Обмін повідомленнями

«Передача даних від одного процесу іншому, або надсилання викликів методів.»

Абстрагування

Спрощення складної дійсності шляхом моделювання класів, що відповідають проблемі, та використання найприйнятнішого рівня деталізації окремих аспектів проблеми. Наприклад Собака Сірко більшу частину часу може розглядатись як Собака, а коли потрібно отримати доступ до інформації специфічної для собак породи коллі — як Коллі і як Тварина (можливо, батьківський клас Собака) при підрахунку тварин Петра.

1.3 Об’єктно-орієнтовані особливості С++

Сі++ додає до Сі об'єктно-орієнтовані можливості. Він вводить класи, які забезпечують три найважливіші властивості ООП: інкапсуляція, поліморфізм, успадкування.

Переваги ООП:

- Основною перевагою ООП в порівнянні з програмування мовою С є «більш природна» декомпозиція програмного забезпечення, яка істотно полегшує його розробку.
- Крім цього, підхід в ООП пропонує нові способи організації програм, засновані на оновних його парадигмах успадкування, поліморфізму, композиції, наповнення.
- Ці механізми дозволяють створювати складні об'єкти з порівняно простих.

В результаті істотно збільшується показник повторного використання кодів і з'являється можливість створення бібліотек класів для різних застосувань.

В мові С основним способом організації даних були структури. Структура складається з набору полів, які ніяк не захищені. Якщо елементи структури мають змінну довжину, їх представляють у вигляді вказівників. Виділення і звільнення пам'яті під ці вказівники робляться вручну.

Така реалізація небезпечна і неефективна з багатьох причин:

- Необхідно виділяти та очищати пам'ять для творення масиву. Програміст може забути викликати одну з цих функцій, або викликати її дуже рано/запізно, або двічі, або з вказівником на неправильний масив. Все це приводить до помилок, що важко виявити.
- Немає ніякого способу перешкодити програмістам створювати і інші функції для роботи із структурою.
- Немає ніякого способу перешкодити програмістам безпосередньо міняти значення масиву або його довжину.

Мова Сі++, використовуючи ООП, усуває всі ці проблеми.

Інкапсуляція

Завдяки тому що у С++ можна додавати змінними параметри доступу такі, як public, private, protected, на відміну від структур мови програмування С, класи можуть мати методи за допомогою яких автор класу може сам вирішувати чи давати доступ користувачу до змінної чи ні.

Агрегація

Агрегація (агрегування за посиланням) — відношення «частина ціле» між двома рівноправними об'єктами, коли один об'єкт (контейнер) має посилання на інший об'єкт. Обидва об'єкти можуть існувати незалежно: якщо контейнер буде знищено, його вміст — ні.

Композиція

Композиція — більш суворий варіант агрегації. Відома також як агрегація за значенням.

Композиція має жорстку залежність часу існування екземплярів класу контейнера та примірників містяться класів. Якщо контейнер буде знищений, то весь його вміст буде також знищено.

Графічно представляється як і агрегація, але з зафарбованим ромбиком.

Відмінності між композицією і агрегацією

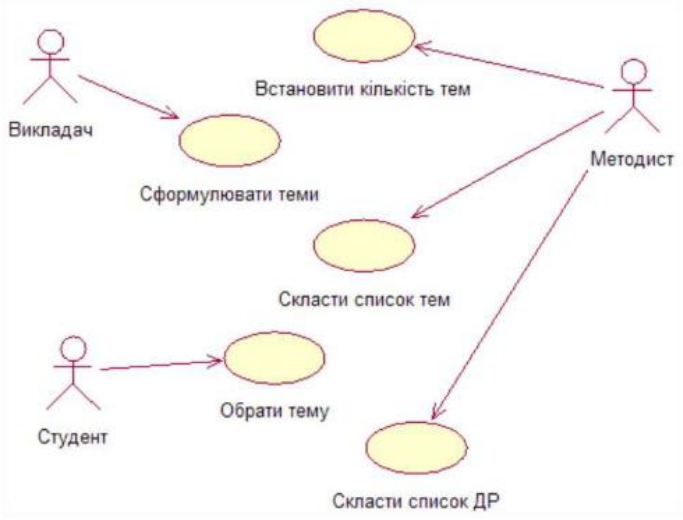
Агрегація - це тип асоціації, який описує тип відносин **«має»** між об'єктами. Наприклад, *автомобіль «має» коробку передач, а автомобіль «має» двигун* . Для відносин один-до-одного, прикладом є *автомобіль, який має багато коліс* .

Композиція ж описує **“частину”** відносин. Наприклад, *лист є частиною дерева*, якщо дерево руйнується, то листя повинні бути знищені.

*1.4*Теорія про діаграми

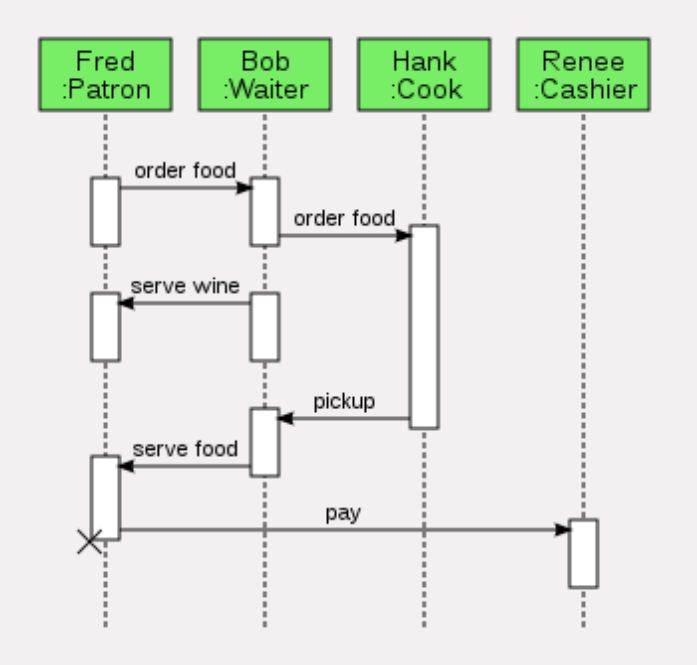
Діаграма прецедентів

Діаграма прецедентів — в UML, діаграма, на якій зображено відношення між акторами та прецедентами в системі. Також, перекладається як діаграма варіантів використання.



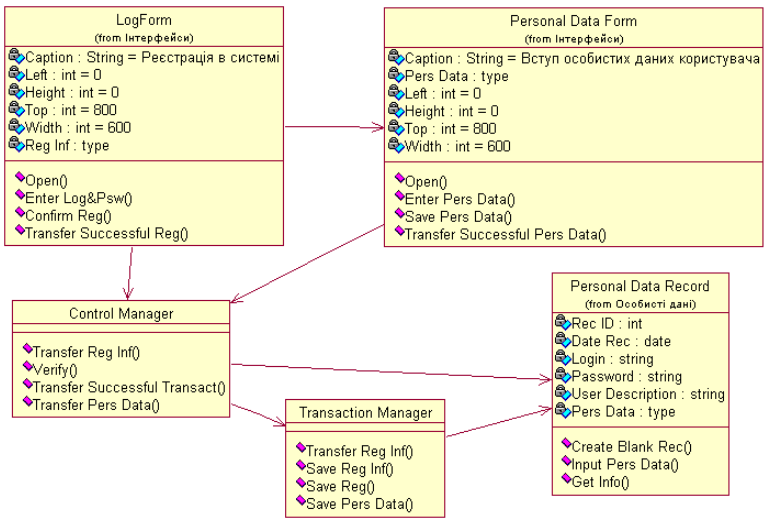
Діагама послідовності

Діаграма послідовності (англ. *sequence diagram*) — різновид діаграми в UML. Діаграма послідовності відображає взаємодії об'єктів впорядкованих за часом. Зокрема, такі діаграми відображають задіяні об'єкти та послідовність відправлених повідомлень.



Діаграма класів

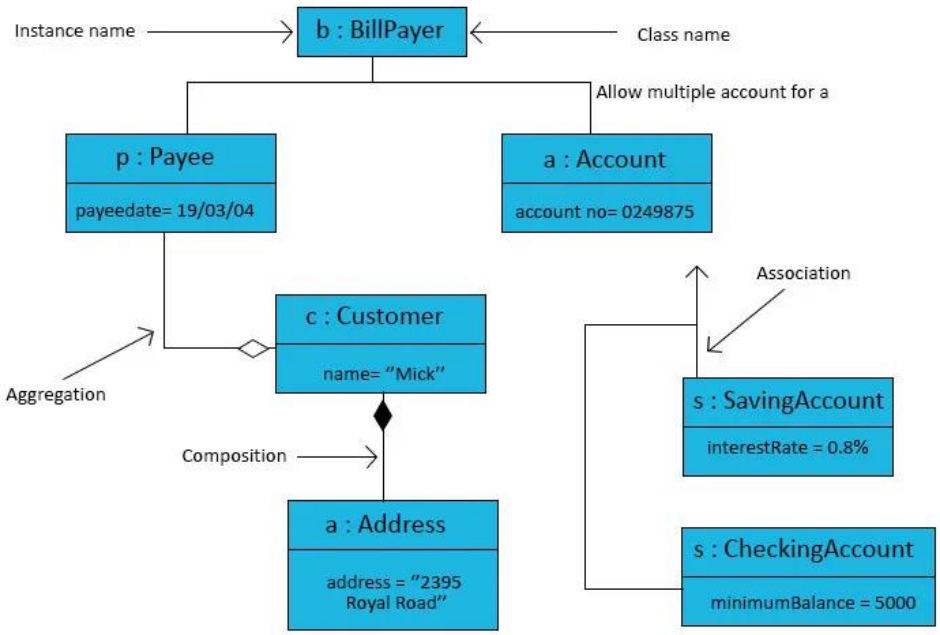
Діагра́ма кла́сів (англ. *class diagram*) — статичне представлення структури моделі. Відображає статичні (декларативні) елементи, такі як: класи, типи даних, їх зміст та відношення. Діаграма класів, також, може містити позначення для пакетів та може містити позначення для вкладених пакетів. Також, діаграма класів може містити позначення деяких елементів поведінки, однак їх динаміка розкривається в інших типах діаграм.[1] Діаграма класів служить для представлення статичної структури моделі системи в термінології класів об'єктно-орієнтованого програмування. На цій діаграмі показують класи, інтерфейси, об'єкти й кооперації, а також їхні відносини.



Діаграми об’єктів

Діаграма об'єктів — в UML, діаграма, що відображає об'єкти та їх зв'язки в певний момент часу. Діаграма об'єктів може розглядатись як окремий випадок діаграми класів, на якій можуть бути представлені як класи, так і екземпляри (об'єкти) класів. Схожою за змістом є діаграма взаємодії (англ. collaboration diagram).

Діаграми об'єктів не мають власної нотації. Оскільки діаграми класів можуть відображати об'єкти, то діаграма класів, на якій відображено лише об'єкти, та не відображено класи, може вважатись діаграмою об'єктів.



2. Аналіз та розробка алгоритму

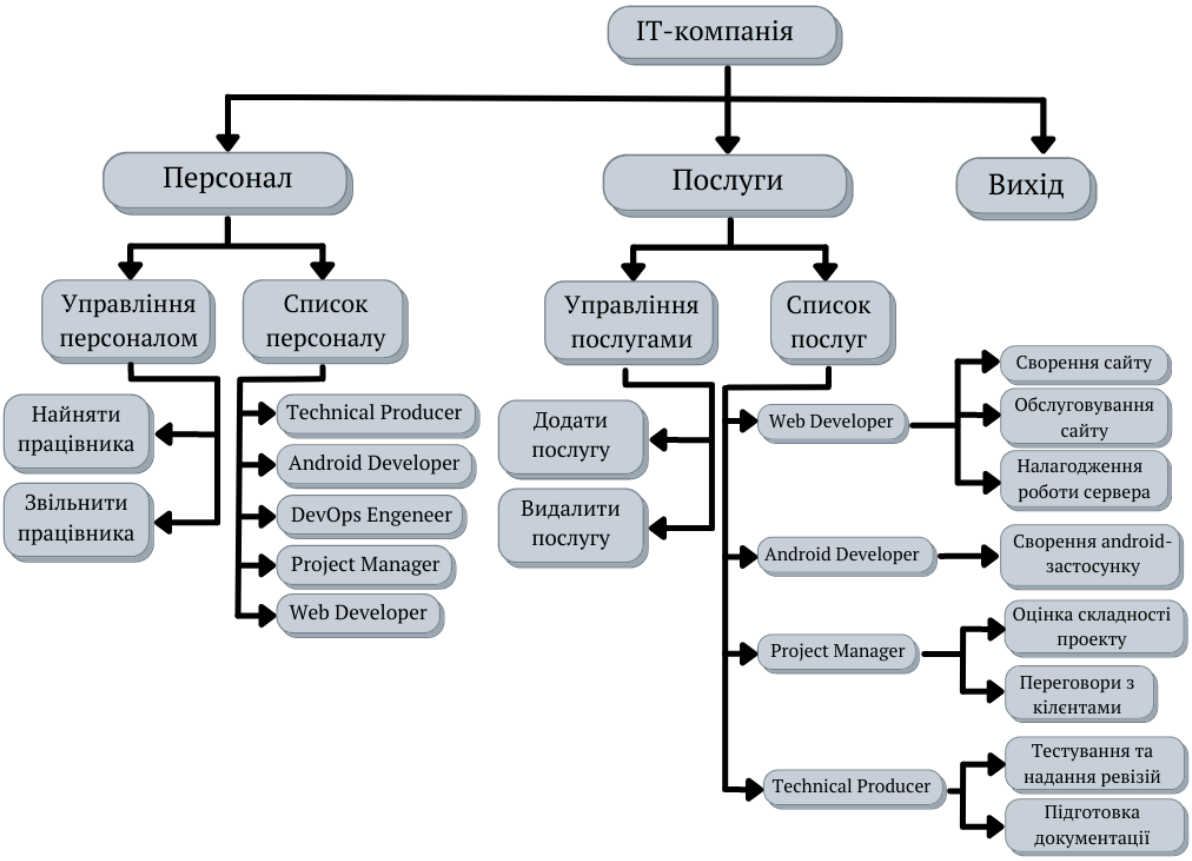


Рис.1 Опис предметної області

2.1 Опис алгоритму та його складності

Рис.2 Діаграма прецендентів

При використанні цієї програми у користувача буде два варіанти входу вб перший – це як адміністраторб другий – чк кілєнт. Та в залежності від того який варіант доступу ви виберете у вас будуть різні можливості у програмі.

Якщо ви увійдете як адміністратор тоді у вас буде доступ до таких функцій:

- Доступ до іформації про ІТ компанію.
- Список працівників.
- Можливість найму та звільнення працівників.
- Доступ до даних працівників.
- Список проєктів
- Управління проектами
- Можливість дати проєкт на опрацювання працівнику.

А якщо увійдете як клієнт, то у вас буде доступ до:

- Перегляд інформації про ІТ компанію
- Можливість переглянути список послуг та ціни
- Можливість замовити власний проект
- Отримати контактні дані для зв’язку з менеджером

Якщо ви зайшли як адміністратор то побачите кнопки:

1.Працівники.
2.Проєкти.
3.Додати.
4.Відняти.
5.Показати інформацію

Також побачите список у який виведеться список працівників якщо нажати на кнопку 1, або список проєктів якщо нажати на кнопку 2. Після виведення списку працівників/проектів у вас з’явиться можливість вибрати працівника/проект зі списку та звільнити/видалити цей об’єкт на кнопку 4, також у вас буде можливість додати новго працівника нажавши на кнопку 3 та ввівши його ім’я, професію та заробітню плату або нажавши на ту саму кнопку при вибрних проєктах, ви зможете додати новий проєкт ввівши його назву.

Якщо ви зайшли як клієнт то побачите кнопки:

1.Інформація про ІТ-компанію
2.Список послуг
3.Індивідуальне замовлення
4.Контактні данні

Отже при вході як клієнт у вас є тільки 4 варіанти вибору, якщо ви нажали на кнопку 1 то вам виведеться уся інформація про ІТ-компанію, якщо нажали на кнопку 2 то вам висвітиться список послуг у якому ви зможете нажати на потрібну вам послугу і вона додасть в чергу замовлень, Якщо ви нажмете на кнопку 3 ви побачите віконце у яке зможете ввести своє замовлення і пізніше адмін зможе його переглянути та дати на виконання працівнику, та якщо ви

нажмете на кнопку 4 вам висвітиться контактний номер телефону, телеграм тег, та електрона адреса для зв’язку з адміністратором.

2.2 Аналіз особливостей алгориму

В данному алгоритмі пристуні декілька властивостей:

1. Використання типів даних для роботи з QT Creator.
2. Наявні переходи між вікнами за допомогою QPushButton.
3. Наявні конструкції if else, for та while для перевірки умов та прокручування циклів.
4. Наявні алгоритми зчитування/запису даних з файлів.

При першому заході у програму ви побачите перед собою два поля вводу, для логіну та паролю, та дві кнопки для входу та реєстрації, вам потрібно буде нажати на кнопку для регістрації після чого з’явиться вікно куди потрібно буде ввести дані та підтвердити їх, після чого за допомогою алгоритму запису даних у файл вас буде добавлено у список юзерів та під час настпуної спроби ходу програма спочатку за допомогою конструкції if else перевірить чи ви не є адміном та за допомогою циклу for та алгоритму зчитування з фалу провірить чи ви правильно ввели дані та якщо все правильно загрузить вас в меню прорами.

2.3 Опис алгоритму у псевдокоді

У цій програмі є три основних класи на прикладі яких я зараз покажу вам роботу програми

CCompany – основний клас.

CWorkers – клас працівника.

CProject -Клас проекту.

В процесі виконання програми дуже часто потрібно отримати якусь інформацію від працівника знаючи тільки його id тому в класі CCompany є створений масив працівників WorkersList[] наприклад функції:

GetWorkerName(int id);

GetWorkerProf(int id);

GetWorkerProj(int id);

SetProjectToWorker(int id);

В усіх цих функціях використовуються масив WorkersList створений класом CCompany яка отримала дані з файлу та має доступ до всіх працівників.

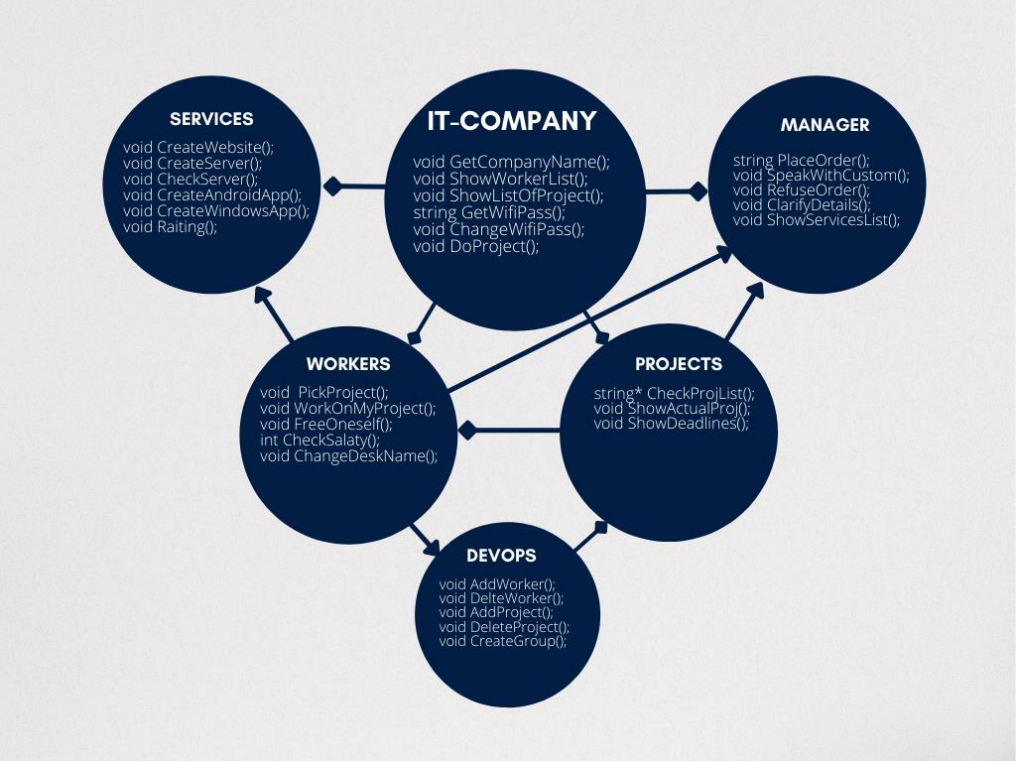
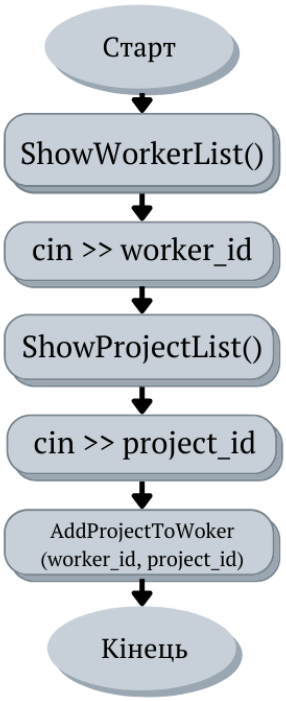


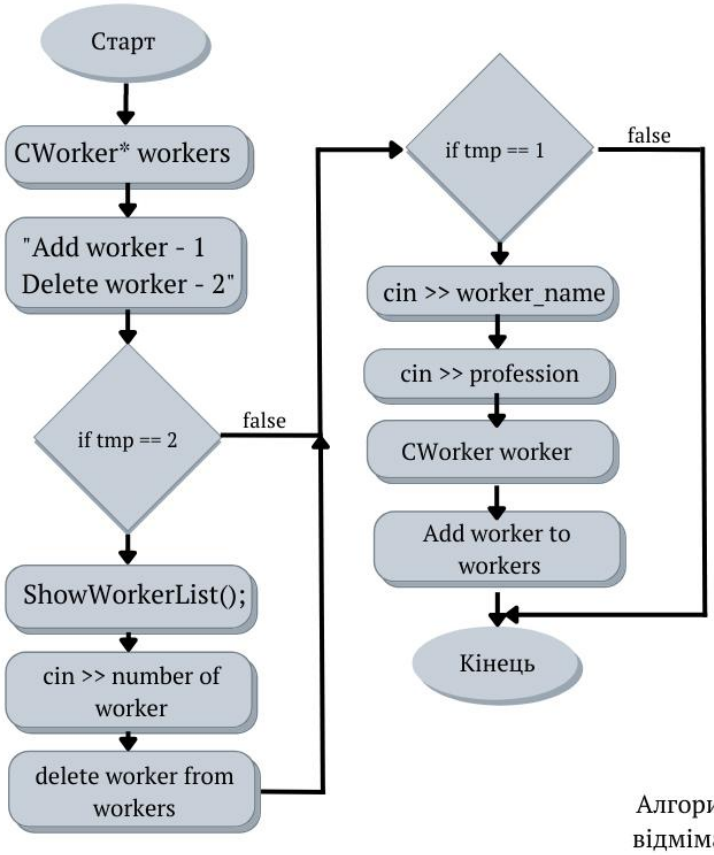
Рис.3 Діаграма класів

2.4 Розробка блок-схеми алгоритму



Алгоритм вибору проекту для працівників

Рис.4 Блок-схема алгоритму метода AddProjectToWorker();



Алгоритм додавання та відмімання працівників

Рис.5 Блок-схема алгоритму метода AddWorker() та DeleteWorker()

3 Програмна реалізація проекту

Для коректної роботи програми та збереження даних я використав аналог файлової змінної QFile яка використовується виключно в середовищі QT Creator та задопомогою цієї бібліотеки створив 5 текстових файлів:

name.txt

salary.txt

workers.txt

project.txt

users.txt

Та записував та зчитував з них дані під час виконання програми.

3.1 Програмна реалізація об’єктно-орієнтованої моделі проекту

Клас MainWindow наслідують класи QMainWindow який дає мені змогу використувати усі функції класу QMainWindow для роботи з графічним інтерфейсом.

mainwindow.h

```
class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

private slots:
    void on_pushButton_clicked();

    void on_pushButton_2_clicked();

private:
    Ui::MainWindow *ui;
};
```

3.2 Програмна реалізація алгоритмів функціонування проекту

Клас CCompany основний клас прорами у якому є всі потрібні методи для управління компанією та коректної роботи запису та зчитування з файлу та у цьому класі є 3 основних методи :

AddProjectToWorker(); - функція яка додає проект до працівнкика для його обробки;

AddWorker() – функція за допомогою якої ви можете добавити працівнка у компанію;

DeleteWorker() - функція за допомогою якої ви можете звільнити працівнка узкомпанію;

ccompany.h

```
#ifndef CCOMPANY_H
#define CCOMPANY_H
#include "cgroup.h"
#include <string>
#include "cworkers.h"
#include "ui_mainwindow.h"

class CCompany
{
public:
    int w_info;
    CCompany();
    CCompany(QString cname);
    CCompany(const CCompany& other);
    ~CCompany();
    string ShowCName();
    void AddWorker(QString profession, QString name, int salary, int save);
    void DeleteWorker(int id);
    void AddProject(QString pname, int save);
    void DeleteProject(int id);
    void AddProjectToWorker(int pid, int wid);
    bool SingIn(QString log, QString pass);
    void AddUser(QString log, QString pass);
    void Register(QString log, QString pass);
    bool GetAdmin();
    int GetNWorkers();
    QString GetWorkerName(int wid);
    QString GetWorkerSalary(int wid);
    QString GetWorkerProf(int wid);
    QString GetProjectName(int pid);
    QString GetWorkerProjectName(int wid);
    QString GetCompanyName();
    void ChangeWorkerName(QString name, int wid);
    void ChangeWorkerSalary(QString sal, int wid);


    int GetNProjects();


private:
    bool isAdmin = false;
    QString* logins;
    QString* passwords;
```

```
    int number_of_users = 1;
    QString c_name;
    int number_of_workers = 0;
    CWorkers* WorkersList;
    int number_of_project = 0;
    CProject* ProjectsList;

};

#endif // CCOMPANY_H
```

ccompany.cpp

```
#include "ccompany.h"
#include <fstream>
#include <iostream>
#include <QFile>

using namespace std;

CCompany::CCompany() {
    c_name = "None";
    WorkersList = new CWorkers[number_of_workers];
    CWorkers W1("ADMIN", "Rostyk", 5000, 0);
    WorkersList[0] = W1;
    logins = new QString[1];
    logins[0] = "ADMIN";
    passwords = new QString[1];
    passwords[0] = "1234";
}

CCompany::CCompany(QString cname) {
    c_name = cname;

    size_t linenum = 0;

    QString p_line;
    QFile p_file("C:\\Users\\rostyk\\Documents\\kursova2\\projects.txt");
    if(p_file.open(QIODevice::ReadOnly)){
        QTextStream in(&p_file);
        do{
            p_line = in.readLine();
             if(!p_line.isNull()) { linenum++; }
        } while (!p_line.isNull());
        p_file.close();
    }
    QString *project = new QString[linenum];

    if(p_file.open(QIODevice::ReadOnly)){
        QTextStream in(&p_file);
        for (size_t i = 0; i < linenum; i++){
            p_line = in.readLine();
            AddProject(p_line, 0);
        }
        p_file.close();
    }

    linenum = 0;
    QString w_line;
    QFile w_file("C:\\Users\\rostyk\\Documents\\kursova2\\workers.txt");
    if(w_file.open(QIODevice::ReadOnly)){
        QTextStream in(&w_file);
        do{
            w_line = in.readLine();
             if(!w_line.isNull()) { linenum++; }
        } while (!w_line.isNull());
```

```
    w_file.close();
}
QString* prof = new QString[linenum];
QString* proj = new QString[linenum];
if(w_file.open(QIODevice::ReadOnly)){
    QTextStream in(&w_file);
    for (size_t i = 0; i < linenum; i++){
        w_line = in.readLine();
        if(i % 2 == 0 || i == 0){
            prof[i/2] = w_line;
        }else if(i % 2 != 0 || i ==1){
            proj[i/2] = w_line;
        }
    }
    for(int i = 0; i < linenum/2; i++){
        AddWorker(prof[i],"None", 500, 0);
    }
    for(int i = 0; i < linenum/2; i++){
        int tmp = proj[i].toInt();
        if(tmp != -1){
             WorkersList[i].SetProject(ProjectsList[tmp]);
        }

    }
    w_file.close();
}

QFile n_file("C:\\Users\\rostyk\\Documents\\kursova2\\name.txt");
QString n_line;
if(n_file.open(QIODevice::ReadOnly)){
    QTextStream in(&n_file);
    for (size_t i = 0; i < linenum/2; i++){
        n_line = in.readLine();
        WorkersList[i].SetName(n_line);
    }
    n_file.close();
}


QFile s_file("C:\\Users\\rostyk\\Documents\\kursova2\\salary.txt");
QString s_line;
if(s_file.open(QIODevice::ReadOnly)){
    QTextStream in(&s_file);
    for (size_t i = 0; i < linenum/2; i++){
        s_line = in.readLine();
        WorkersList[i].SetSalary(s_line.toInt());
    }
    s_file.close();
}


logins = new QString[1];
logins[0] = "admin";
passwords = new QString[1];
passwords[0] = "1234";

linenum = 0;
QString line;
QFile file("C:\\Users\\rostyk\\Documents\\kursova2\\users.txt");
if(file.open(QIODevice::ReadOnly)){
    QTextStream in(&file);
    do{
        line = in.readLine();
        if(!line.isNull()) { linenum++; }
    } while (!line.isNull());
```

```
        file.close();
    }
    QString* log = new QString[linenum];
    QString* pass = new QString[linenum];
    if(file.open(QIODevice::ReadOnly)){
        QTextStream in(&file);
        for (size_t i = 0; i < linenum; i++){
            line = in.readLine();
            if(i % 2 == 0 || i == 0){
                log[i/2] = line;
            }else if(i % 2 != 0 || i ==1){
                pass[i/2] = line;
            }
        }
        for (int i = 0; i < linenum/2; i++){
            AddUser(log[i], pass[i]);
        }
        file.close();
    }



}

CCompany::CCompany(const CCompany& other)
{
    if (this != &other) {
        c_name = other.c_name;
        number_of_workers = other.number_of_workers;
        number_of_project = other.number_of_project;
        WorkersList = new CWorkers[number_of_workers];
        WorkersList = other.WorkersList;
        ProjectsList = new CProject[number_of_project];
        ProjectsList = other.ProjectsList;
    }
}



void CCompany::AddWorker(QString profession, QString name, int salary, int
save){
    CWorkers W(profession, name, salary, number_of_workers);
    CWorkers* WorkersListTmp = new CWorkers[number_of_workers];
    WorkersListTmp = WorkersList;
    if(save == 1){
        QFile file("C:\\Users\\rostyk\\Documents\\kursova2\\workers.txt");
        if(file.open(QIODevice::Append | QIODevice::Text)){
            QTextStream writeStream(&file);
            writeStream << profession  << "\n" << W.GetProjId()  << "\n";
            file.close();
        }
        QFile n_file("C:\\Users\\rostyk\\Documents\\kursova2\\name.txt");
        if( n_file.open(QIODevice::Append | QIODevice::Text)){
            QTextStream writeStream(& n_file);
            writeStream << name << "\n";
            n_file.close();
        }
        QFile s_file("C:\\Users\\rostyk\\Documents\\kursova2\\salary.txt");
        if( s_file.open(QIODevice::Append | QIODevice::Text)){
            QTextStream writeStream(& s_file);
            writeStream << salary << "\n";
            s_file.close();
        }
    }
```

```
    number_of_workers++;
    WorkersList = new CWorkers[number_of_workers];
    for (int i = 0; i < number_of_workers; i++) {
        if (i != number_of_workers - 1) {
            WorkersList[i] = WorkersListTmp[i];
        }
        else {


            WorkersList[i] = W;

        }
    }
}

void CCompany::DeleteWorker(int id) {
    CWorkers* WorkersListTmp = new CWorkers[number_of_workers];
    WorkersListTmp = WorkersList;
    number_of_workers--;
    WorkersList = new CWorkers[number_of_workers];
    for (int i = 0; i < number_of_workers; i++) {
        if (i < id) {
            WorkersList[i] = WorkersListTmp[i];
        }
        else if (i >= id) {
            WorkersList[i] = WorkersListTmp[i + 1];
            WorkersList[i].SetId(i);
        }

    }
    delete[] WorkersListTmp;

    QFile file("C:\\Users\\rostyk\\Documents\\kursova2\\workers.txt");
    if(file.open(QIODevice::WriteOnly | QIODevice::Text)){
        file.close();

    }
    for(int i = 0; i < number_of_workers; i++){
        if(file.open(QIODevice::Append | QIODevice::Text)){
            QTextStream writeStream(&file);
            writeStream << WorkersList[i].GetProfession() << "\n" <<
WorkersList[i].GetProjId() << "\n";
            file.close();
        }
    }

    QFile n_file("C:\\Users\\rostyk\\Documents\\kursova2\\name.txt");
    if(n_file.open(QIODevice::WriteOnly | QIODevice::Text)){
        n_file.close();

    }
    for(int i = 0; i < number_of_workers; i++){
        if(n_file.open(QIODevice::Append | QIODevice::Text)){
            QTextStream writeStream(&n_file);
            writeStream << WorkersList[i].GetName() << "\n";
            n_file.close();
        }
    }

    QFile s_file("C:\\Users\\rostyk\\Documents\\kursova2\\salary.txt");
    if(s_file.open(QIODevice::WriteOnly | QIODevice::Text)){
        s_file.close();

    }
    for(int i = 0; i < number_of_workers; i++){
```

```
        if(s_file.open(QIODevice::Append | QIODevice::Text)){
            QTextStream writeStream(&s_file);
            writeStream << WorkersList[i].GetSalary() << "\n";
            s_file.close();
        }
    }

}



void CCompany::AddProject(QString pname, int save) {
    if(save == 1){
        QFile file("C:\\Users\\rostyk\\Documents\\kursova2\\projects.txt");
        if(file.open(QIODevice::Append | QIODevice::Text)){
            QTextStream writeStream(&file);
            writeStream << pname << "\n";
            file.close();
        }
    }
    if(number_of_project != 0 ) {
        CProject* ProjectsListTmp = new CProject[number_of_project];
        ProjectsListTmp = ProjectsList;
        number_of_project++;
        ProjectsList = new CProject[number_of_project];
        for (int i = 0; i < number_of_project; i++) {
            if (i != number_of_project - 1) {
                ProjectsList[i] = ProjectsListTmp[i];
            }
            else {

                CProject P(pname, i);
                ProjectsList[i] = P;

            }
        }
        delete[] ProjectsListTmp;
    }
    else {
        number_of_project++;
        ProjectsList = new CProject[1];
        CProject P(pname, 0);
        ProjectsList[0] = P;
    }



}

void CCompany::DeleteProject(int id) {
    if (number_of_project != 0) {
        CProject* ProjectsListTmp = new CProject[number_of_workers];
        ProjectsListTmp = ProjectsList;
        number_of_project--;
        ProjectsList = new CProject[number_of_project];
        for (int i = 0; i < number_of_project; i++) {
            if (i < id) {
                ProjectsList[i] = ProjectsListTmp[i];
            }
            else if (i >= id) {
                ProjectsList[i] = ProjectsListTmp[i + 1];
                ProjectsList[i].SetId(i);
            }

        }
```

```
        delete[] ProjectsListTmp;
    }
    else {
        cout << "Company hasnt have project" << endl;
    }
    QFile file("C:\\Users\\rostyk\\Documents\\kursova2\\projects.txt");
    if(file.open(QIODevice::WriteOnly | QIODevice::Text)){
        file.close();

    }
    for(int i = 0; i < number_of_project; i++){
        if(file.open(QIODevice::Append | QIODevice::Text)){
            QTextStream writeStream(&file);
            writeStream << ProjectsList[i].GetPName() << "\n";
            file.close();
        }
    }

}



void CCompany::AddProjectToWorker(int pid, int wid) {
    WorkersList[wid].SetProject(ProjectsList[pid]);
    QFile file("C:\\Users\\rostyk\\Documents\\kursova2\\workers.txt");
    if(file.open(QIODevice::WriteOnly | QIODevice::Text)){
        file.close();

    }
    for(int i = 0; i < number_of_workers; i++){
        if(file.open(QIODevice::Append | QIODevice::Text)){
            QTextStream writeStream(&file);
            writeStream << WorkersList[i].GetProfession() << "\n" <<
WorkersList[i].GetProjId() << "\n";
            file.close();
        }
    }
}


bool CCompany::SingIn(QString log, QString pass){
    for(int i =0; i< number_of_users; i++){
        if(log == logins[i]){
            if(pass == passwords[i]){
                if(i == 0){
                    isAdmin = true;
                }
                return true;
            }else{
                continue;
            }
        }else{
            continue;
        }
    }
}

void CCompany::AddUser(QString log, QString pass){
    number_of_users++;
    QString* l_tmp = new QString[number_of_users-1];
    l_tmp = logins;
    logins = new QString[number_of_users];
    for(int i =0 ; i < number_of_users ;i++){
        if(i != number_of_users - 1){
            logins[i] = l_tmp[i];
```

```
        }else{
            logins[i] = log;
        }
    }
    QString* p_tmp = new QString[number_of_users-1];
    p_tmp = passwords;
    passwords = new QString[number_of_users];
    for(int i =0 ; i < number_of_users ;i++){
        if(i != number_of_users - 1){
            passwords[i] = p_tmp[i];
        }else{
            passwords[i] = pass;
        }
    }
    delete[] l_tmp;
    delete[] p_tmp;
}

void CCompany::Register(QString log, QString pass){
    QFile file("C:\\Users\\rostyk\\Documents\\kursova2\\users.txt");
    if(file.open(QIODevice::Append | QIODevice::Text)){
        QTextStream writeStream(&file);
        writeStream << log << "\n" << pass;
        file.close();
    }
}

QString CCompany::GetWorkerProf(int wid){
    return WorkersList[wid].GetProfession();
}

QString CCompany::GetWorkerName(int wid){
    return WorkersList[wid].GetName();
}

QString CCompany::GetWorkerSalary(int wid){
    QString s = QString::number(WorkersList[wid].GetSalary());
    return s;
}

QString CCompany::GetProjectName(int pid){
    return ProjectsList[pid].GetPName();
}

bool CCompany::GetAdmin(){
    return isAdmin;
}

int CCompany::GetNWorkers(){
    return number_of_workers;
}

QString CCompany::GetWorkerProjectName(int wid){
   return WorkersList[wid].GetProjName();
}

int CCompany::GetNProjects(){
    return number_of_project;
}

void CCompany::ChangeWorkerName(QString name, int wid){
    WorkersList[wid].SetName(name);
    QFile file("C:\\Users\\rostyk\\Documents\\kursova2\\name.txt");
    if(file.open(QIODevice::WriteOnly | QIODevice::Text)){
        file.close();
```

```
    }
    for(int i = 0; i < number_of_workers; i++){
        if(file.open(QIODevice::Append | QIODevice::Text)){
            QTextStream writeStream(&file);
            writeStream << WorkersList[i].GetName() << "\n";
            file.close();
        }
    }
}

void CCompany::ChangeWorkerSalary(QString sal, int wid){
    WorkersList[wid].SetSalary(sal.toInt());
    QFile file("C:\\Users\\rostyk\\Documents\\kursova2\\salary.txt");
    if(file.open(QIODevice::WriteOnly | QIODevice::Text)){
        file.close();

    }
    for(int i = 0; i < number_of_workers; i++){
        if(file.open(QIODevice::Append | QIODevice::Text)){
            QTextStream writeStream(&file);
            writeStream << WorkersList[i].GetSalary() << "\n";
            file.close();
        }
    }
}

QString CCompany::GetCompanyName(){
    return c_name;
}


CCompany::~CCompany() {

}
```

3.3 Програмна реалізація інтерфейсу проекту та його пунктів меню

mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

private slots:
    void on_pushButton_clicked();

    void on_pushButton_2_clicked();
```

```
private:
    Ui::MainWindow *ui;
};
#endif // MAINWINDOW_H
```

main.cpp

```
#include "mainwindow.h"
#include <QApplication>
#include <QLocale>
#include <QTranslator>


int main(int argc, char *argv[])
{
    QApplication a(argc, argv);

    QTranslator translator;
    const QStringList uiLanguages = QLocale::system().uiLanguages();
    for (const QString &locale : uiLanguages) {
        const QString baseName = "kursova2_" + QLocale(locale).name();
        if (translator.load(":/i18n/" + baseName)) {
            a.installTranslator(&translator);
            break;
        }
    }
    MainWindow w;
    w.setWindowFlags(Qt::Window | Qt::FramelessWindowHint);
    w.show();
    return a.exec();
}
```

mainwindow.cpp

```
#include "mainwindow.h"
#include "registerwindow.h"
#include "menuwindow.h"
#include "ui_mainwindow.h"
#include "ccompany.h"
#include <QMessageBox>
#include "usermenuwindow.h"

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    CCompany C("GEREGA CORPORATION");
    ui->tittle->setText(C.GetCompanyName());
}

MainWindow::~MainWindow()
{
    delete ui;
}


void MainWindow::on_pushButton_clicked()
{
    QString login = ui->loginInput->text();
    QString pass = ui->passInput->text();
    CCompany C("Gerega Corporation");
    bool res = C.SingIn(login, pass);
    if(res == true){
        if(C.GetAdmin()){
            this->close();
```

```
            QMessageBox::information(this, "Успіх!", "Ви успішно
авторизувались");
            MenuWindow menu;
            menu.setWindowFlags(Qt::Window | Qt::FramelessWindowHint);
            menu.setModal(true);
            menu.exec();
         }else{
            this->close();
            QMessageBox::information(this, "Успіх!", "Ви успішно
авторизувались");
            UserMenuWindow menu;
            menu.setWindowFlags(Qt::Window | Qt::FramelessWindowHint);
            menu.setModal(true);
            menu.exec();
        }

    }else{
        QMessageBox::information(this, "Невдача(", "Логін або пароль введено не
правильно");
    }
}

void MainWindow::on_pushButton_2_clicked()
{
    RegisterWindow reg;
    reg.setModal(true);
    reg.exec();
}
```

menuwindow.h

```
#ifndef MENUWINDOW_H
#define MENUWINDOW_H

#include <QDialog>
#include "mainwindow.h"

namespace Ui {
class MenuWindow;
}

class MenuWindow : public QDialog
{
    Q_OBJECT

public:
    explicit MenuWindow(QWidget *parent = nullptr);
    ~MenuWindow();

private slots:
    void on_pushButton_4_clicked();

    void on_workButton_clicked();

    void on_projButton_clicked();

    void on_pushButton_3_clicked();

    void on_addButton_clicked();

    void on_infoButton_clicked();

    void on_subButton_clicked();

private:
```

```
    Ui::MenuWindow *ui;
    MainWindow *mainWindow;
};
```

#endif *// MENUWINDOW_H*

menuwindow.cpp

```
#include "ccompany.h"
#include "mainwindow.h"
#include "addworkerwindow.h"
#include "addprojectwindow.h"
#include "workerinfowindow.h"
#include <QFile>

int IsWorker = -1;

MenuWindow::MenuWindow(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::MenuWindow)
{
    IsWorker = -1;
    ui->setupUi(this);
}

MenuWindow::~MenuWindow()
{
    delete ui;
}


void MenuWindow::on_workButton_clicked()
{
    IsWorker = 1;
    ui->addButton->setText("Найняти працівника");
    ui->subButton->setText("Звільнити працівника");
    while(ui->listWidget->count()>0)
    {
        ui->listWidget->takeItem(0);
    }
    CCompany C("GEREGACORPORATION");
    int N = C.GetNWorkers();
    for(int i = 0; i < N; i++ ){
        QString S;
        if(C.GetWorkerProjectName(i) != ""){
            S = C.GetWorkerProf(i) + "->" +  C.GetWorkerProjectName(i);
        }else{
            S = C.GetWorkerProf(i);
        }
        ui->listWidget->addItem(S);

    }
}

void MenuWindow::on_projButton_clicked()
{
    IsWorker = 0;
    ui->addButton->setText("Додати проєкт");
    ui->subButton->setText("Видалити проєкт");
    while(ui->listWidget->count()>0)
    {
        ui->listWidget->takeItem(0);
    }
    CCompany C("GEREGACORPORATION");
    int N = C.GetNProjects();
```

```
    for(int i = 0; i < N; i++ ){
        ui->listWidget->addItem(C.GetProjectName(i));
    }
}


void MenuWindow::on_pushButton_3_clicked()
{
    this->close();
}


void MenuWindow::on_addButton_clicked()
{
    CCompany C("GEREGA CORPORATION");
    if(IsWorker == 1){
        AddWorkerWindow ww;
        ww.setModal(true);
        ww.exec();
    }else if(IsWorker == 0){
        AddProjectWindow pw;
        pw.setModal(true);
        pw.exec();
    }

}

void MenuWindow::on_subButton_clicked()
{
    QListWidgetItem *id= ui->listWidget->currentItem();
    CCompany C("GEREGA CORPORATION");
    if(IsWorker == 1){
        C.DeleteWorker(ui->listWidget->row(id));
    }else if(IsWorker == 0){
        C.DeleteProject(ui->listWidget->row(id));
    }
}

void MenuWindow::on_infoButton_clicked()
{
    CCompany C("GEREGA CORPORATION");
    QListWidgetItem *id= ui->listWidget->currentItem();
    if(id){
        if(IsWorker == 1){
            QFile file("C:\\Users\\rostyk\\Documents\\kursova2\\tmp.txt");
            if(file.open(QIODevice::WriteOnly | QIODevice::Text)){
                QTextStream writeStream(&file);
                writeStream << ui->listWidget->row(id) << "\n";
                file.close();
            }
            this->close();
            WorkerInfoWindow w;
            w.setModal(true);
            w.exec();

        }
   }
}
```

registerwindow.h

```
#ifndef REGISTERWINDOW_H
#define REGISTERWINDOW_H

#include <QDialog>
```

```
namespace Ui {
class RegisterWindow;
}

class RegisterWindow : public QDialog
{
    Q_OBJECT

public:
    explicit RegisterWindow(QWidget *parent = nullptr);
    ~RegisterWindow();

private slots:
    void on_pushButton_clicked();

private:
    Ui::RegisterWindow *ui;
};
```

#endif *// REGISTERWINDOW_H*

registerwindow.h

```
#include "mainwindow.h"
#include "registerwindow.h"
#include "ui_registerwindow.h"
#include "ccompany.h"
#include <QMessageBox>

RegisterWindow::RegisterWindow(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::RegisterWindow)
{
    ui->setupUi(this);
}

RegisterWindow::~RegisterWindow()
{
    delete ui;
}

void RegisterWindow::on_pushButton_clicked()
{
    QString login = ui->loginInput->text();
    QString pass = ui->passInput->text();
    CCompany C("Gerega Corporation");
    C.Register(login, pass);
    C.AddUser(login, pass);
    QMessageBox::information(this, "Успіх!", "Ви успішно зареєструвались");
    this->close();
}
```

usermenuwindow.h

```
#ifndef USERMENUWINDOW_H
#define USERMENUWINDOW_H

#include <QDialog>

namespace Ui {
class UserMenuWindow;
}
```

```
class UserMenuWindow : public QDialog
{
    Q_OBJECT

public:
    explicit UserMenuWindow(QWidget *parent = nullptr);
    ~UserMenuWindow();

private slots:
    void on_pushButton_clicked();

    void on_pushButton_4_clicked();

    void on_pushButton_2_clicked();

    void on_pushButton_3_clicked();

    void on_pushButton_5_clicked();

private:
    Ui::UserMenuWindow *ui;
};
```

#endif *// USERMENUWINDOW_H*

usermenuwindow.cpp

```
#include "usermenuwindow.h"
#include "ui_usermenuwindow.h"
#include "ccompany.h"
#include "userinfowindow.h"
#include "userpickwindow.h"
#include "addprojectwindow.h"
#include "contactwindow.h"


UserMenuWindow::UserMenuWindow(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::UserMenuWindow)
{
    ui->setupUi(this);
    CCompany C("GEREGA CORPORATION");
    ui->tittle->setText(C.GetCompanyName());
}

UserMenuWindow::~UserMenuWindow()
{
    delete ui;
}

void UserMenuWindow::on_pushButton_clicked()
{
    //info
    this->close();
    UserinfoWindow w;
    w.setWindowFlags(Qt::Window | Qt::FramelessWindowHint);
    w.setModal(true);
    w.exec();
}


void UserMenuWindow::on_pushButton_4_clicked()
{
```

```
    //menu
    this->close();
    UserPickWindow w;
    w.setWindowFlags(Qt::Window | Qt::FramelessWindowHint);
    w.setModal(true);
    w.exec();
}


void UserMenuWindow::on_pushButton_2_clicked()
{
    //invidual
    AddProjectWindow w;
    w.setModal(true);
    w.exec();
}


void UserMenuWindow::on_pushButton_3_clicked()
{
    //contact
    ContactWindow w;
    w.setWindowFlags(Qt::Window | Qt::FramelessWindowHint);
    w.setModal(true);
    w.exec();
}


void UserMenuWindow::on_pushButton_5_clicked()
{
    this->close();

}
```

userpickwindow.h

```
#ifndef USERPICKWINDOW_H
#define USERPICKWINDOW_H

#include <QDialog>

namespace Ui {
class UserPickWindow;
}

class UserPickWindow : public QDialog
{
    Q_OBJECT

public:
    explicit UserPickWindow(QWidget *parent = nullptr);
    ~UserPickWindow();

private slots:
    void on_webButton_clicked();

    void on_androidButton_clicked();

    void on_windowsButton_clicked();

    void on_analButton_clicked();

    void on_server1Button_clicked();

    void on_server2Button_clicked();
```

```
    void on_server3Button_clicked();

    void on_pushButton_clicked();

private:
    Ui::UserPickWindow *ui;
};

#endif // USERPICKWINDOW_H
```

userpickwindow.cpp

```
#include "userpickwindow.h"
#include "ui_userpickwindow.h"
#include <QMessageBox>
#include "ccompany.h"
#include "usermenuwindow.h"


UserPickWindow::UserPickWindow(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::UserPickWindow)
{
    ui->setupUi(this);
}

UserPickWindow::~UserPickWindow()
{
    delete ui;
}

void UserPickWindow::on_webButton_clicked()
{
    QMessageBox::StandardButton reply = QMessageBox::question(this,
"Підтвердження",
                                                              "Ви впевнені що
хочете замовити \"" + ui->webButton->text() +"\"",
                                                              QMessageBox::Yes |
QMessageBox::No);
    if(reply ==  QMessageBox::Yes){
        CCompany C("GEREGA CORPORATION");
        C.AddProject("WebSite", 1);
    }
}


void UserPickWindow::on_androidButton_clicked()
{
    QMessageBox::StandardButton reply = QMessageBox::question(this,
"Підтвердження",
                                                              "Ви впевнені що
хочете замовити \"" + ui->androidButton->text()+"\"",
                                                              QMessageBox::Yes |
QMessageBox::No);
    if(reply ==  QMessageBox::Yes){
        CCompany C("GEREGA CORPORATION");
        C.AddProject("AndroidApp", 1);
    }
}


void UserPickWindow::on_windowsButton_clicked()
{
```

```
    QMessageBox::StandardButton reply = QMessageBox::question(this,
"Підтвердження",
                                                              "Ви впевнені що
хочете замовити \"" + ui->windowsButton->text()+"\"",
                                                              QMessageBox::Yes |
QMessageBox::No);
    if(reply ==  QMessageBox::Yes){
        CCompany C("GEREGA CORPORATION");
        C.AddProject("WindowsApp", 1);
    }
}


void UserPickWindow::on_analButton_clicked()
{
    QMessageBox::StandardButton reply = QMessageBox::question(this,
"Підтвердження",
                                                              "Ви впевнені що
хочете замовити \"" + ui->analButton->text()+"\"",
                                                              QMessageBox::Yes |
QMessageBox::No);
    if(reply ==  QMessageBox::Yes){
        CCompany C("GEREGA CORPORATION");
        C.AddProject("Analize", 1);
    }
}


void UserPickWindow::on_server1Button_clicked()
{
    QMessageBox::StandardButton reply = QMessageBox::question(this,
"Підтвердження",
                                                              "Ви впевнені що
хочете замовити \"" + ui->server1Button->text()+"\"",
                                                              QMessageBox::Yes |
QMessageBox::No);
    if(reply ==  QMessageBox::Yes){
        CCompany C("GEREGA CORPORATION");
        C.AddProject("ServerCreation", 1);
    }
}


void UserPickWindow::on_server2Button_clicked()
{
    QMessageBox::StandardButton reply = QMessageBox::question(this,
"Підтвердження",
                                                              "Ви впевнені що
хочете замовити \"" + ui->server2Button->text()+"\"",
                                                              QMessageBox::Yes |
QMessageBox::No);
    if(reply ==  QMessageBox::Yes){
        CCompany C("GEREGA CORPORATION");
        C.AddProject("ServerAdjustment", 1);
    }
}


void UserPickWindow::on_server3Button_clicked()
{
    QMessageBox::StandardButton reply = QMessageBox::question(this,
"Підтвердження",
                                                              "Ви впевнені що
хочете замовити \"" + ui->server3Button->text()+"\"",
```

```
                                                    QMessageBox::Yes |
QMessageBox::No);
    if(reply ==  QMessageBox::Yes){
        CCompany C("GEREGA CORPORATION");
        C.AddProject("ServerRent", 1);
    }
}


void UserPickWindow::on_pushButton_clicked()
{
    this->close();
    UserMenuWindow w;
    w.setWindowFlags(Qt::Window | Qt::FramelessWindowHint);
    w.setModal(true);
    w.exec();
}
```

4.Відлагодження

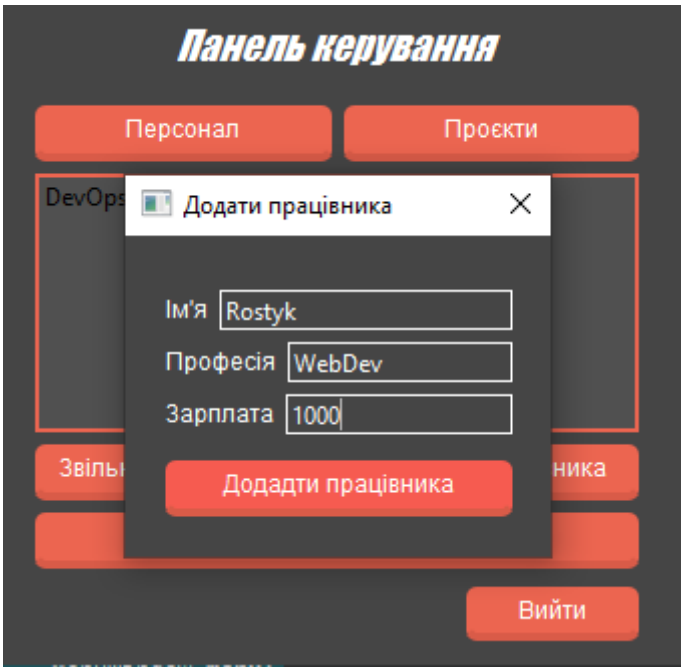


Рис 4.1 Введення інформації

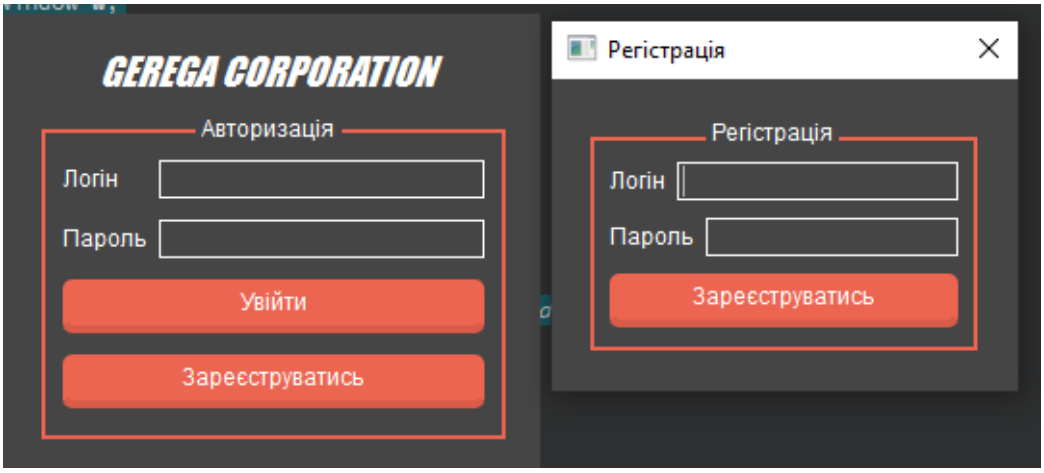


Рис 3.2 Меню авторизації/регістрації

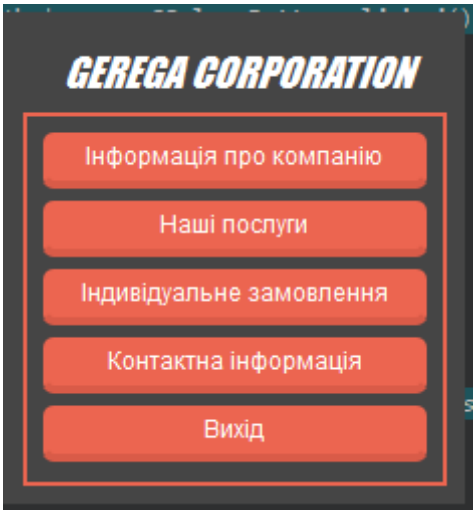


Рис 3.3 Головне меню клієнта

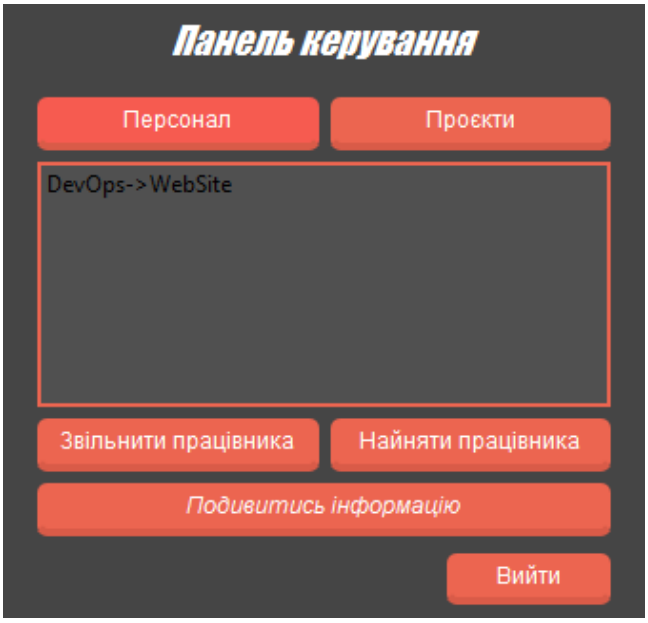


Рис 3.4 Головне меню адміністратора

Рис 3.5 Список послуг

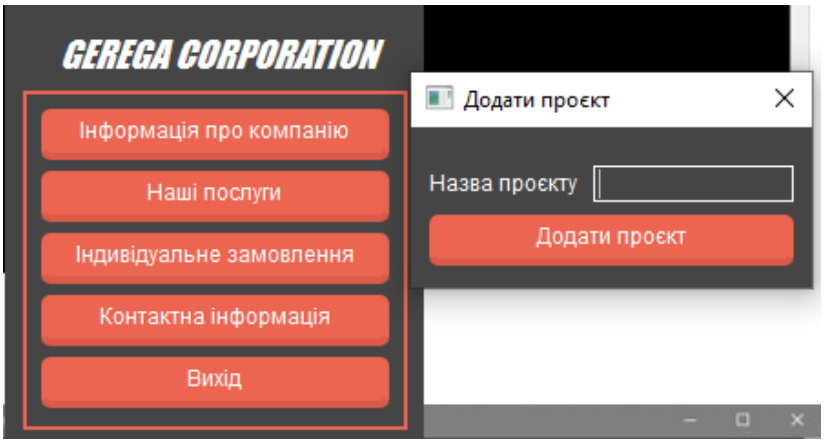


Рис 3.6 Меню додавання індивідуального замовлення

4.1 Аналіз структурної складності виконання програми

Якщо оцінювати структуру програми написану у QT Creator то не можливо не сказати про зручність використання нового класу для кожного нового вікна, крім того оскільки всі ці класи знаходяться у різних файлах це дуже зручно розділяє код та робить його структуру кращою для розуміння. Тому на мою думку хоть і в моїй програмі багато коду та файлі її структура не є фажкою для сприйняття.

Під час написання програми я зіткнувся в основному тільки з однією складністю це робота з типами даних у QT Creator, тому що для виводу інформації у графічне меню нене можна використати тип даних String, а потрібно використовувати QString, але ця проблема була вирішена шляхом підключення бібліотеки QString:

#include <QString> та переписом функція які використовують String у функції які використовують QString.

Другою проблемою була зв’язана з записом та зчитуванням даних у файл, сама проблема полягає у тому що QT Creator не підтримую роботу типу даних FILE. Рішенням цієї пробелми заключалось у підключені біліотеки

QFile:

#include<QFile>

Та переписом функції запису/зчитування у функції які використовують QFile замість FILE.

5 Аналіз часу виконання програми в залежності від об’єму вхідних даних

Хоч у моєї програми і графічний інтерфейс, працює вона достатньо швидко з будь-яким об’ємом даних. Це через те що під час написання інтерфейсу я не використовував велику кількість картинок у програмі а старався максимально стисло в текстовому фортматі подати інформацію користувачу і на мою думку це сильно вплинуо на бистродію програми.

Усі методи зчитування та запису у файл виконують за короткий період часу.

В основному файл середнього розміру(500 кілобайт) записується у файл за 0.3 секунди.

Та той самий об’єм інформація зчитується з файлу за 0.17 секунди.

Вікна інтерфейсу відкриваються без затримо тому можу сказати що у цієї програми доволі хороші показники бистродії і функції запису/зчитування та відображення меню.

5.1 Повний код програми

ccompany.h

```
#ifndef CCOMPANY_H
#define CCOMPANY_H
#include "cgroup.h"
#include <string>
#include "cworkers.h"
#include "ui_mainwindow.h"

class CCompany
{
public:
    int w_info;
    CCompany();
    CCompany(QString cname);
```

```
    CCompany(const CCompany& other);
    ~CCompany();
    string ShowCName();
    void AddWorker(QString profession, QString name, int salary, int save);
    void DeleteWorker(int id);
    void AddProject(QString pname, int save);
    void DeleteProject(int id);
    void AddProjectToWorker(int pid, int wid);
    bool SingIn(QString log, QString pass);
    void AddUser(QString log, QString pass);
    void Register(QString log, QString pass);
    bool GetAdmin();
    int GetNWorkers();
    QString GetWorkerName(int wid);
    QString GetWorkerSalary(int wid);
    QString GetWorkerProf(int wid);
    QString GetProjectName(int pid);
    QString GetWorkerProjectName(int wid);
    QString GetCompanyName();
    void ChangeWorkerName(QString name, int wid);
    void ChangeWorkerSalary(QString sal, int wid);


    int GetNProjects();


private:
    bool isAdmin = false;
    QString* logins;
    QString* passwords;
    int number_of_users = 1;
    QString c_name;
    int number_of_workers = 0;
    CWorkers* WorkersList;
    int number_of_project = 0;
    CProject* ProjectsList;

};

#endif // CCOMPANY_H
```

ccompany.cpp

```
#include "ccompany.h"
#include <fstream>
#include <iostream>
#include <QFile>

using namespace std;

CCompany::CCompany() {
    c_name = "None";
    WorkersList = new CWorkers[number_of_workers];
    CWorkers W1("ADMIN", "Rostyk", 5000, 0);
    WorkersList[0] = W1;
    logins = new QString[1];
    logins[0] = "ADMIN";
    passwords = new QString[1];
    passwords[0] = "1234";
}

CCompany::CCompany(QString cname) {
    c_name = cname;

    size_t linenum = 0;
```

```
QString p_line;
QFile p_file("C:\\Users\\rostyk\\Documents\\kursova2\\projects.txt");
if(p_file.open(QIODevice::ReadOnly)){
    QTextStream in(&p_file);
    do{
        p_line = in.readLine();
         if(!p_line.isNull()) { linenum++; }
    } while (!p_line.isNull());
    p_file.close();
}
QString *project = new QString[linenum];

if(p_file.open(QIODevice::ReadOnly)){
    QTextStream in(&p_file);
    for (size_t i = 0; i < linenum; i++){
        p_line = in.readLine();
        AddProject(p_line, 0);
    }
    p_file.close();
}

linenum = 0;
QString w_line;
QFile w_file("C:\\Users\\rostyk\\Documents\\kursova2\\workers.txt");
if(w_file.open(QIODevice::ReadOnly)){
    QTextStream in(&w_file);
    do{
        w_line = in.readLine();
         if(!w_line.isNull()) { linenum++; }
    } while (!w_line.isNull());
    w_file.close();
}
QString* prof = new QString[linenum];
QString* proj = new QString[linenum];
if(w_file.open(QIODevice::ReadOnly)){
    QTextStream in(&w_file);
    for (size_t i = 0; i < linenum; i++){
        w_line = in.readLine();
        if(i % 2 == 0 || i == 0){
            prof[i/2] = w_line;
        }else if(i % 2 != 0 || i ==1){
            proj[i/2] = w_line;
        }
    }
    for(int i = 0; i < linenum/2; i++){
        AddWorker(prof[i],"None", 500, 0);
    }
    for(int i = 0; i < linenum/2; i++){
        int tmp = proj[i].toInt();
        if(tmp != -1){
             WorkersList[i].SetProject(ProjectsList[tmp]);
        }

    }
    w_file.close();
}

QFile n_file("C:\\Users\\rostyk\\Documents\\kursova2\\name.txt");
QString n_line;
if(n_file.open(QIODevice::ReadOnly)){
    QTextStream in(&n_file);
    for (size_t i = 0; i < linenum/2; i++){
        n_line = in.readLine();
        WorkersList[i].SetName(n_line);
    }
```

```
        n_file.close();
    }


    QFile s_file("C:\\Users\\rostyk\\Documents\\kursova2\\salary.txt");
    QString s_line;
    if(s_file.open(QIODevice::ReadOnly)){
        QTextStream in(&s_file);
        for (size_t i = 0; i < linenum/2; i++){
            s_line = in.readLine();
            WorkersList[i].SetSalary(s_line.toInt());
        }
        s_file.close();
    }


    logins = new QString[1];
    logins[0] = "admin";
    passwords = new QString[1];
    passwords[0] = "1234";

    linenum = 0;
    QString line;
    QFile file("C:\\Users\\rostyk\\Documents\\kursova2\\users.txt");
    if(file.open(QIODevice::ReadOnly)){
        QTextStream in(&file);
        do{
            line = in.readLine();
            if(!line.isNull()) { linenum++; }
        } while (!line.isNull());
        file.close();
    }
    QString* log = new QString[linenum];
    QString* pass = new QString[linenum];
    if(file.open(QIODevice::ReadOnly)){
        QTextStream in(&file);
        for (size_t i = 0; i < linenum; i++){
            line = in.readLine();
            if(i % 2 == 0 || i == 0){
                log[i/2] = line;
            }else if(i % 2 != 0 || i ==1){
                pass[i/2] = line;
            }
        }
        for (int i = 0; i < linenum/2; i++){
            AddUser(log[i], pass[i]);
        }
        file.close();
    }



}

CCompany::CCompany(const CCompany& other)
{
    if (this != &other) {
        c_name = other.c_name;
        number_of_workers = other.number_of_workers;
        number_of_project = other.number_of_project;
        WorkersList = new CWorkers[number_of_workers];
        WorkersList = other.WorkersList;
        ProjectsList = new CProject[number_of_project];
        ProjectsList = other.ProjectsList;
    }
```

```
}


void CCompany::AddWorker(QString profession, QString name, int salary, int
save){
    CWorkers W(profession, name, salary, number_of_workers);
    CWorkers* WorkersListTmp = new CWorkers[number_of_workers];
    WorkersListTmp = WorkersList;
    if(save == 1){
        QFile file("C:\\Users\\rostyk\\Documents\\kursova2\\workers.txt");
        if(file.open(QIODevice::Append | QIODevice::Text)){
            QTextStream writeStream(&file);
            writeStream << profession  << "\n" << W.GetProjId()  << "\n";
            file.close();
        }
        QFile n_file("C:\\Users\\rostyk\\Documents\\kursova2\\name.txt");
        if( n_file.open(QIODevice::Append | QIODevice::Text)){
            QTextStream writeStream(& n_file);
            writeStream << name << "\n";
            n_file.close();
        }
        QFile s_file("C:\\Users\\rostyk\\Documents\\kursova2\\salary.txt");
        if( s_file.open(QIODevice::Append | QIODevice::Text)){
            QTextStream writeStream(& s_file);
            writeStream << salary << "\n";
            s_file.close();
        }
    }

    number_of_workers++;
    WorkersList = new CWorkers[number_of_workers];
    for (int i = 0; i < number_of_workers; i++) {
        if (i != number_of_workers - 1) {
            WorkersList[i] = WorkersListTmp[i];
        }
        else {


            WorkersList[i] = W;

        }
    }
}

void CCompany::DeleteWorker(int id) {
    CWorkers* WorkersListTmp = new CWorkers[number_of_workers];
    WorkersListTmp = WorkersList;
    number_of_workers--;
    WorkersList = new CWorkers[number_of_workers];
    for (int i = 0; i < number_of_workers; i++) {
        if (i < id) {
            WorkersList[i] = WorkersListTmp[i];
        }
        else if (i >= id) {
            WorkersList[i] = WorkersListTmp[i + 1];
            WorkersList[i].SetId(i);
        }

    }
    delete[] WorkersListTmp;

    QFile file("C:\\Users\\rostyk\\Documents\\kursova2\\workers.txt");
    if(file.open(QIODevice::WriteOnly | QIODevice::Text)){
        file.close();
```

```
    }
    for(int i = 0; i < number_of_workers; i++){
        if(file.open(QIODevice::Append | QIODevice::Text)){
            QTextStream writeStream(&file);
            writeStream << WorkersList[i].GetProfession() << "\n" <<
WorkersList[i].GetProjId() << "\n";
            file.close();
        }
    }

    QFile n_file("C:\\Users\\rostyk\\Documents\\kursova2\\name.txt");
    if(n_file.open(QIODevice::WriteOnly | QIODevice::Text)){
        n_file.close();

    }
    for(int i = 0; i < number_of_workers; i++){
        if(n_file.open(QIODevice::Append | QIODevice::Text)){
            QTextStream writeStream(&n_file);
            writeStream << WorkersList[i].GetName() << "\n";
            n_file.close();
        }
    }

    QFile s_file("C:\\Users\\rostyk\\Documents\\kursova2\\salary.txt");
    if(s_file.open(QIODevice::WriteOnly | QIODevice::Text)){
        s_file.close();

    }
    for(int i = 0; i < number_of_workers; i++){
        if(s_file.open(QIODevice::Append | QIODevice::Text)){
            QTextStream writeStream(&s_file);
            writeStream << WorkersList[i].GetSalary() << "\n";
            s_file.close();
        }
    }

}



void CCompany::AddProject(QString pname, int save) {
    if(save == 1){
        QFile file("C:\\Users\\rostyk\\Documents\\kursova2\\projects.txt");
        if(file.open(QIODevice::Append | QIODevice::Text)){
            QTextStream writeStream(&file);
            writeStream << pname << "\n";
            file.close();
        }
    }
    if(number_of_project != 0 ) {
        CProject* ProjectsListTmp = new CProject[number_of_project];
        ProjectsListTmp = ProjectsList;
        number_of_project++;
        ProjectsList = new CProject[number_of_project];
        for (int i = 0; i < number_of_project; i++) {
            if (i != number_of_project - 1) {
                ProjectsList[i] = ProjectsListTmp[i];
            }
            else {

                CProject P(pname, i);
                ProjectsList[i] = P;

            }
```

```
        }
        delete[] ProjectsListTmp;
    }
    else {
        number_of_project++;
        ProjectsList = new CProject[1];
        CProject P(pname, 0);
        ProjectsList[0] = P;
    }



}

void CCompany::DeleteProject(int id) {
    if (number_of_project != 0) {
        CProject* ProjectsListTmp = new CProject[number_of_workers];
        ProjectsListTmp = ProjectsList;
        number_of_project--;
        ProjectsList = new CProject[number_of_project];
        for (int i = 0; i < number_of_project; i++) {
            if (i < id) {
                ProjectsList[i] = ProjectsListTmp[i];
            }
            else if (i >= id) {
                ProjectsList[i] = ProjectsListTmp[i + 1];
                ProjectsList[i].SetId(i);
            }

        }
        delete[] ProjectsListTmp;
    }
    else {
        cout << "Company hasnt have project" << endl;
    }
    QFile file("C:\\Users\\rostyk\\Documents\\kursova2\\projects.txt");
    if(file.open(QIODevice::WriteOnly | QIODevice::Text)){
        file.close();

    }
    for(int i = 0; i < number_of_project; i++){
        if(file.open(QIODevice::Append | QIODevice::Text)){
            QTextStream writeStream(&file);
            writeStream << ProjectsList[i].GetPName() << "\n";
            file.close();
        }
    }

}



void CCompany::AddProjectToWorker(int pid, int wid) {
    WorkersList[wid].SetProject(ProjectsList[pid]);
    QFile file("C:\\Users\\rostyk\\Documents\\kursova2\\workers.txt");
    if(file.open(QIODevice::WriteOnly | QIODevice::Text)){
        file.close();

    }
    for(int i = 0; i < number_of_workers; i++){
        if(file.open(QIODevice::Append | QIODevice::Text)){
            QTextStream writeStream(&file);
            writeStream << WorkersList[i].GetProfession() << "\n" <<
WorkersList[i].GetProjId() << "\n";
            file.close();
```

```
        }
    }
}


bool CCompany::SingIn(QString log, QString pass){
    for(int i =0; i< number_of_users; i++){
        if(log == logins[i]){
            if(pass == passwords[i]){
                if(i == 0){
                    isAdmin = true;
                }
                return true;
            }else{
                continue;
            }
        }else{
            continue;
        }
    }
}

void CCompany::AddUser(QString log, QString pass){
    number_of_users++;
    QString* l_tmp = new QString[number_of_users-1];
    l_tmp = logins;
    logins = new QString[number_of_users];
    for(int i =0 ; i < number_of_users ;i++){
        if(i != number_of_users - 1){
            logins[i] = l_tmp[i];
        }else{
            logins[i] = log;
        }
    }
    QString* p_tmp = new QString[number_of_users-1];
    p_tmp = passwords;
    passwords = new QString[number_of_users];
    for(int i =0 ; i < number_of_users ;i++){
        if(i != number_of_users - 1){
            passwords[i] = p_tmp[i];
        }else{
            passwords[i] = pass;
        }
    }
    delete[] l_tmp;
    delete[] p_tmp;
}

void CCompany::Register(QString log, QString pass){
    QFile file("C:\\Users\\rostyk\\Documents\\kursova2\\users.txt");
    if(file.open(QIODevice::Append | QIODevice::Text)){
        QTextStream writeStream(&file);
        writeStream << log << "\n" << pass;
        file.close();
    }
}

QString CCompany::GetWorkerProf(int wid){
    return WorkersList[wid].GetProfession();
}

QString CCompany::GetWorkerName(int wid){
    return WorkersList[wid].GetName();
}
```

```
QString CCompany::GetWorkerSalary(int wid){
    QString s = QString::number(WorkersList[wid].GetSalary());
    return s;
}

QString CCompany::GetProjectName(int pid){
    return ProjectsList[pid].GetPName();
}

bool CCompany::GetAdmin(){
    return isAdmin;
}

int CCompany::GetNWorkers(){
    return number_of_workers;
}

QString CCompany::GetWorkerProjectName(int wid){
   return WorkersList[wid].GetProjName();
}

int CCompany::GetNProjects(){
    return number_of_project;
}

void CCompany::ChangeWorkerName(QString name, int wid){
    WorkersList[wid].SetName(name);
    QFile file("C:\\Users\\rostyk\\Documents\\kursova2\\name.txt");
    if(file.open(QIODevice::WriteOnly | QIODevice::Text)){
        file.close();

    }
    for(int i = 0; i < number_of_workers; i++){
        if(file.open(QIODevice::Append | QIODevice::Text)){
            QTextStream writeStream(&file);
            writeStream << WorkersList[i].GetName() << "\n";
            file.close();
        }
    }
}

void CCompany::ChangeWorkerSalary(QString sal, int wid){
    WorkersList[wid].SetSalary(sal.toInt());
    QFile file("C:\\Users\\rostyk\\Documents\\kursova2\\salary.txt");
    if(file.open(QIODevice::WriteOnly | QIODevice::Text)){
        file.close();

    }
    for(int i = 0; i < number_of_workers; i++){
        if(file.open(QIODevice::Append | QIODevice::Text)){
            QTextStream writeStream(&file);
            writeStream << WorkersList[i].GetSalary() << "\n";
            file.close();
        }
    }
}

QString CCompany::GetCompanyName(){
    return c_name;
}


CCompany::~CCompany() {
```

}

cworkers.h

```
#ifndef CWORKERS_H
#define CWORKERS_H

#include <string>
#include <stdlib.h>
#include "cproject.h"
#include "ui_mainwindow.h"


using namespace std;

class CWorkers
{
public:
    CWorkers();
    CWorkers(QString proffesion, QString name, int s,  int id);
    CWorkers(const CWorkers& other);
    CWorkers& operator=(CWorkers& other);
    int GetId();
    int GetSalary();
    int GetProjId();
    QString GetProfession();
    QString GetName();
    QString GetProjName();
    void SetId(int id);
    void SetProject(CProject project);
    void SetName(QString name);
    void SetSalary(int sal);
    ~CWorkers();

private:
    QString w_profession;
    CProject w_project;
    int w_id;
    int p_id = -1;
    int salary;
    QString w_name;

};
```

#endif // CWORKERS_H

cworkers.cpp

```
#include "cworkers.h"
#include <iostream>
#include "workerinfowindow.h"


using namespace std;

CWorkers::CWorkers() {
    w_profession = "None";
    CProject p_null("None", -1);
    w_project = p_null;
    w_id = -1;

}
```

```
CWorkers::CWorkers(QString proffesion, QString name, int s,  int id) {
    w_profession = proffesion;
    w_name = name;
    salary = s;
    CProject p_null("None", -1);
    w_project = p_null;
    w_id = id;
}

CWorkers::CWorkers(const CWorkers& other) {
    if (this != &other) {
        w_profession = other.w_profession;
        CProject P = other.w_project;
        w_project = P;
        w_id = other.w_id;
        w_name = other.w_name;
        salary = other.salary;
    }
}


QString CWorkers::GetProjName(){
    return w_project.GetPName();
}
int CWorkers::GetId() {
    return w_id;
}



int CWorkers::GetSalary() {
    return salary;
}

int CWorkers::GetProjId() {
    return w_project.GetId();
}


QString CWorkers::GetProfession() {
    return w_profession;
}

QString CWorkers::GetName() {
    return w_name;
}


void CWorkers::SetId(int id) {
    w_id = id;
}

void CWorkers::SetProject(CProject project) {
    w_project = project;
}



CWorkers& CWorkers::operator=(CWorkers& other) {
    if (this != &other) {
        w_profession = other.w_profession;
        w_project = other.w_project;
        w_id = other.w_id;
        w_name = other.w_name;
```

```
        salary = other.salary;

    }
    return *this;
}

void CWorkers::SetName(QString name){
    w_name = name;
}
void CWorkers::SetSalary(int sal){
    salary = sal;
}


CWorkers::~CWorkers() {

}
```

cproject.h

```
#ifndef CPROJECT_H
#define CPROJECT_H
#include <string>
#include <stdlib.h>
#include "ui_mainwindow.h"

using namespace std;

class CProject
{
public:
    CProject();
    CProject(QString pname, int id);
    CProject(const CProject& other);
    CProject& operator=(CProject& other);
    QString GetPName();
    int GetId();
    void SetId(int id);
    ~CProject();

private:
    QString p_name;
    int p_id;
};

#endif // CPROJECT_H
```

cproject.cpp

```
#include "cproject.h"
#include <iostream>


using namespace std;

CProject::CProject() {
    p_name = "None";
    p_id = -1;
}

CProject::CProject(QString pname, int id) {
    p_name = pname;
```

```
    p_id = id;
}

CProject::CProject(const CProject& other) {
    if (this != &other) {
        p_name = other.p_name;
        p_id = other.p_id;
    }
}



int CProject::GetId() {
    return p_id;
}

void CProject::SetId(int id) {
    p_id = id;
}

QString CProject::GetPName() {
    return p_name;
}


CProject& CProject::operator=(CProject& other) {
    if (this != &other) {
        p_name = other.p_name;
        p_id = other.p_id;
    }
    return *this;
}


CProject::~CProject() {

}
```

addprojectwindow.h

```
#ifndef ADDPROJECTWINDOW_H
#define ADDPROJECTWINDOW_H

#include <QDialog>

namespace Ui {
class AddProjectWindow;
}

class AddProjectWindow : public QDialog
{
    Q_OBJECT

public:
    explicit AddProjectWindow(QWidget *parent = nullptr);
    ~AddProjectWindow();

private slots:
    void on_pushButton_clicked();

private:
    Ui::AddProjectWindow *ui;
};
```

#endif // *ADDPROJECTWINDOW_H*

addprojectwindow.cpp

```
#include "addprojectwindow.h"
#include "ui_addprojectwindow.h"
#include "ccompany.h"


AddProjectWindow::AddProjectWindow(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::AddProjectWindow)
{
    ui->setupUi(this);
}

AddProjectWindow::~AddProjectWindow()
{
    delete ui;
}

void AddProjectWindow::on_pushButton_clicked()
{
    CCompany C("GEREGA CORPORATION");
    QString name = ui->name->text();
    C.AddProject(name, 1);
    this->close();
}
```

addworkerwindow.h

```
#ifndef ADDWORKERWINDOW_H
#define ADDWORKERWINDOW_H

#include <QDialog>

namespace Ui {
class AddWorkerWindow;
}

class AddWorkerWindow : public QDialog
{
    Q_OBJECT

public:
    explicit AddWorkerWindow(QWidget *parent = nullptr);
    ~AddWorkerWindow();

private slots:
    void on_pushButton_clicked();

private:
    Ui::AddWorkerWindow *ui;
};
```

#endif // *ADDWORKERWINDOW_H*

addworkerwindow.cpp

```
#include "addworkerwindow.h"
#include "ui_addworkerwindow.h"
#include "ccompany.h"
```

```
AddWorkerWindow::AddWorkerWindow(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::AddWorkerWindow)
{
    ui->setupUi(this);
}

AddWorkerWindow::~AddWorkerWindow()
{
    delete ui;
}

void AddWorkerWindow::on_pushButton_clicked()
{
    CCompany C("GEREGA CORPORATION");
    QString name = ui->name->text();
    QString profession = ui->profession->text();
    QString salary = ui->salary->text();
    C.AddWorker(profession, name, salary.toInt(), 1);
    this->close();
}
```

contactwindow.h

```
#ifndef CONTACTWINDOW_H
#define CONTACTWINDOW_H

#include <QDialog>

namespace Ui {
class ContactWindow;
}

class ContactWindow : public QDialog
{
    Q_OBJECT

public:
    explicit ContactWindow(QWidget *parent = nullptr);
    ~ContactWindow();

private slots:
    void on_pushButton_clicked();

private:
    Ui::ContactWindow *ui;
};
```

#endif // CONTACTWINDOW_H

contactwindow.cpp

```
#include "contactwindow.h"
#include "ui_contactwindow.h"
#include "usermenuwindow.h"

ContactWindow::ContactWindow(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::ContactWindow)
{
    ui->setupUi(this);
}
```

```
ContactWindow::~ContactWindow()
{
    delete ui;
}

void ContactWindow::on_pushButton_clicked()
{
    this->close();
    UserMenuWindow menu;
    menu.setWindowFlags(Qt::Window | Qt::FramelessWindowHint);
    menu.setModal(true);
    menu.exec();
}
```

mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

private slots:
    void on_pushButton_clicked();

    void on_pushButton_2_clicked();

private:
    Ui::MainWindow *ui;
};
#endif // MAINWINDOW_H
```

main.cpp

```
#include "mainwindow.h"
#include <QApplication>
#include <QLocale>
#include <QTranslator>


int main(int argc, char *argv[])
{
    QApplication a(argc, argv);

    QTranslator translator;
    const QStringList uiLanguages = QLocale::system().uiLanguages();
    for (const QString &locale : uiLanguages) {
        const QString baseName = "kursova2_" + QLocale(locale).name();
        if (translator.load(":/i18n/" + baseName)) {
            a.installTranslator(&translator);
            break;
        }
    }
    MainWindow w;
    w.setWindowFlags(Qt::Window | Qt::FramelessWindowHint);
```

```
    w.show();
    return a.exec();
}
```

mainwindow.cpp

```
#include "mainwindow.h"
#include "registerwindow.h"
#include "menuwindow.h"
#include "ui_mainwindow.h"
#include "ccompany.h"
#include <QMessageBox>
#include "usermenuwindow.h"

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    CCompany C("GEREGA CORPORATION");
    ui->tittle->setText(C.GetCompanyName());
}

MainWindow::~MainWindow()
{
    delete ui;
}


void MainWindow::on_pushButton_clicked()
{
    QString login = ui->loginInput->text();
    QString pass = ui->passInput->text();
    CCompany C("Gerega Corporation");
    bool res = C.SingIn(login, pass);
    if(res == true){
        if(C.GetAdmin()){
            this->close();
            QMessageBox::information(this, "Успіх!", "Ви успішно
авторизувались");
            MenuWindow menu;
            menu.setWindowFlags(Qt::Window | Qt::FramelessWindowHint);
            menu.setModal(true);
            menu.exec();
         }else{
            this->close();
            QMessageBox::information(this, "Успіх!", "Ви успішно
авторизувались");
            UserMenuWindow menu;
            menu.setWindowFlags(Qt::Window | Qt::FramelessWindowHint);
            menu.setModal(true);
            menu.exec();
        }

    }else{
        QMessageBox::information(this, "Невдача(", "Логін або пароль введено не
правильно");
    }
}


void MainWindow::on_pushButton_2_clicked()
{
    RegisterWindow reg;
    reg.setModal(true);
    reg.exec();
```

}

menuwindow.h

```
#ifndef MENUWINDOW_H
#define MENUWINDOW_H

#include <QDialog>
#include "mainwindow.h"

namespace Ui {
class MenuWindow;
}

class MenuWindow : public QDialog
{
    Q_OBJECT

public:
    explicit MenuWindow(QWidget *parent = nullptr);
    ~MenuWindow();

private slots:
    void on_pushButton_4_clicked();

    void on_workButton_clicked();

    void on_projButton_clicked();

    void on_pushButton_3_clicked();

    void on_addButton_clicked();

    void on_infoButton_clicked();

    void on_subButton_clicked();

private:
    Ui::MenuWindow *ui;
    MainWindow *mainWindow;
};
```

#endif *// MENUWINDOW_H*

menuwindow.cpp

```
#include "ccompany.h"
#include "mainwindow.h"
#include "addworkerwindow.h"
#include "addprojectwindow.h"
#include "workerinfowindow.h"
#include <QFile>

int IsWorker = -1;


MenuWindow::MenuWindow(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::MenuWindow)
{
    IsWorker = -1;
    ui->setupUi(this);
}
```

```
MenuWindow::~MenuWindow()
{
    delete ui;
}


void MenuWindow::on_workButton_clicked()
{
    IsWorker = 1;
    ui->addButton->setText("Найняти працівника");
    ui->subButton->setText("Звільнити працівника");
    while(ui->listWidget->count()>0)
    {
        ui->listWidget->takeItem(0);
    }
    CCompany C("GEREGACORPORATION");
    int N = C.GetNWorkers();
    for(int i = 0; i < N; i++ ){
        QString S;
        if(C.GetWorkerProjectName(i) != ""){
            S = C.GetWorkerProf(i) + "->" +  C.GetWorkerProjectName(i);
        }else{
            S = C.GetWorkerProf(i);
        }
        ui->listWidget->addItem(S);

    }
}

void MenuWindow::on_projButton_clicked()
{
    IsWorker = 0;
    ui->addButton->setText("Додати проєкт");
    ui->subButton->setText("Видалити проєкт");
    while(ui->listWidget->count()>0)
    {
        ui->listWidget->takeItem(0);
    }
    CCompany C("GEREGACORPORATION");
    int N = C.GetNProjects();
    for(int i = 0; i < N; i++ ){
        ui->listWidget->addItem(C.GetProjectName(i));
    }
}


void MenuWindow::on_pushButton_3_clicked()
{
    this->close();
}


void MenuWindow::on_addButton_clicked()
{
    CCompany C("GEREGA CORPORATION");
    if(IsWorker == 1){
        AddWorkerWindow ww;
        ww.setModal(true);
        ww.exec();
    }else if(IsWorker == 0){
        AddProjectWindow pw;
        pw.setModal(true);
        pw.exec();
    }
```

```
}

void MenuWindow::on_subButton_clicked()
{
    QListWidgetItem *id= ui->listWidget->currentItem();
    CCompany C("GEREGA CORPORATION");
    if(IsWorker == 1){
        C.DeleteWorker(ui->listWidget->row(id));
    }else if(IsWorker == 0){
        C.DeleteProject(ui->listWidget->row(id));
    }
}


void MenuWindow::on_infoButton_clicked()
{
    CCompany C("GEREGA CORPORATION");
    QListWidgetItem *id= ui->listWidget->currentItem();
    if(id){
        if(IsWorker == 1){
            QFile file("C:\\Users\\rostyk\\Documents\\kursova2\\tmp.txt");
            if(file.open(QIODevice::WriteOnly | QIODevice::Text)){
                QTextStream writeStream(&file);
                writeStream << ui->listWidget->row(id) << "\n";
                file.close();
            }
            this->close();
            WorkerInfoWindow w;
            w.setModal(true);
            w.exec();

        }
   }
}
```

registerwindow.h

```
#ifndef REGISTERWINDOW_H
#define REGISTERWINDOW_H

#include <QDialog>

namespace Ui {
class RegisterWindow;
}

class RegisterWindow : public QDialog
{
    Q_OBJECT

public:
    explicit RegisterWindow(QWidget *parent = nullptr);
    ~RegisterWindow();

private slots:
    void on_pushButton_clicked();

private:
    Ui::RegisterWindow *ui;
};
```

#endif // *REGISTERWINDOW_H*

registerwindow.h

```
#include "mainwindow.h"
#include "registerwindow.h"
#include "ui_registerwindow.h"
#include "ccompany.h"
#include <QMessageBox>

RegisterWindow::RegisterWindow(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::RegisterWindow)
{
    ui->setupUi(this);
}

RegisterWindow::~RegisterWindow()
{
    delete ui;
}

void RegisterWindow::on_pushButton_clicked()
{
    QString login = ui->loginInput->text();
    QString pass = ui->passInput->text();
    CCompany C("Gerega Corporation");
    C.Register(login, pass);
    C.AddUser(login, pass);
    QMessageBox::information(this, "Успіх!", "Ви успішно зареєструвались");
    this->close();
}
```

setnamewindow.h

```
#ifndef SETNAMEWINDOW_H
#define SETNAMEWINDOW_H

#include <QDialog>

namespace Ui {
class SetNameWindow;
}

class SetNameWindow : public QDialog
{
    Q_OBJECT

public:
    explicit SetNameWindow(QWidget *parent = nullptr);
    ~SetNameWindow();

private slots:
    void on_pushButton_clicked();

private:
    Ui::SetNameWindow *ui;
};
```

#endif *// SETNAMEWINDOW_H*

setnamewindow.cpp

```
#include "setnamewindow.h"
#include "ui_setnamewindow.h"
#include <QFile>
#include "ccompany.h"
```

```
SetNameWindow::SetNameWindow(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::SetNameWindow)
{
    ui->setupUi(this);
}

SetNameWindow::~SetNameWindow()
{
    delete ui;
}

void SetNameWindow::on_pushButton_clicked()
{
    QString wid;
    QFile file("C:\\Users\\rostyk\\Documents\\kursova2\\tmp.txt");
    if(file.open(QIODevice::ReadOnly)){
        QTextStream in(&file);
        wid = in.readLine();
        file.close();
    }
    QString name = ui->nameInput->text();
    CCompany C("GEREGA CORPORATION");
    C.ChangeWorkerName(name, wid.toInt());
    this->close();
}
```

setprojectwindow.h

```
#ifndef SETPROJECTWINDOW_H
#define SETPROJECTWINDOW_H

#include <QDialog>

namespace Ui {
class SetProjectWindow;
}

class SetProjectWindow : public QDialog
{
    Q_OBJECT

public:
    explicit SetProjectWindow(QWidget *parent = nullptr);
    ~SetProjectWindow();

private slots:
    void on_pushButton_clicked();

private:
    Ui::SetProjectWindow *ui;
};
```

#endif *// SETPROJECTWINDOW_H*

setprojectwindow.cpp

```
#include "setprojectwindow.h"
#include "ui_setprojectwindow.h"
#include "ccompany.h"
#include <QFile>

SetProjectWindow::SetProjectWindow(QWidget *parent) :
    QDialog(parent),
```

```
    ui(new Ui::SetProjectWindow)
{
    ui->setupUi(this);
    while(ui->listWidget->count()>0)
    {
        ui->listWidget->takeItem(0);
    }
    CCompany C("GEREGACORPORATION");
    int N = C.GetNProjects();
    for(int i = 0; i < N; i++ ){
        ui->listWidget->addItem(C.GetProjectName(i));
    }
}

SetProjectWindow::~SetProjectWindow()
{
    delete ui;
}

void SetProjectWindow::on_pushButton_clicked()
{
    QListWidgetItem *id= ui->listWidget->currentItem();
    int pid = ui->listWidget->row(id);
    QString wid;
    QFile file("C:\\Users\\rostyk\\Documents\\kursova2\\tmp.txt");
    if(file.open(QIODevice::ReadOnly)){
        QTextStream in(&file);
        wid = in.readLine();
        file.close();
    }
    CCompany C("GEREGA CORPORATION");
    C.AddProjectToWorker(pid, wid.toInt());
    this->close();
}
```

setsalarywindow.h

```
#ifndef SETSALARYWINDOW_H
#define SETSALARYWINDOW_H

#include <QDialog>

namespace Ui {
class SetSalaryWindow;
}

class SetSalaryWindow : public QDialog
{
    Q_OBJECT

public:
    explicit SetSalaryWindow(QWidget *parent = nullptr);
    ~SetSalaryWindow();

private slots:
    void on_pushButton_clicked();

private:
    Ui::SetSalaryWindow *ui;
};

#endif // SETSALARYWINDOW_H
```

setsalarywindow.cpp

```
#include "setsalarywindow.h"
```

```
#include "ui_setsalarywindow.h"
#include <QFile>
#include "ccompany.h"


SetSalaryWindow::SetSalaryWindow(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::SetSalaryWindow)
{
    ui->setupUi(this);
}

SetSalaryWindow::~SetSalaryWindow()
{
    delete ui;
}

void SetSalaryWindow::on_pushButton_clicked()
{
    QString wid;
    QFile file("C:\\Users\\rostyk\\Documents\\kursova2\\tmp.txt");
    if(file.open(QIODevice::ReadOnly)){
        QTextStream in(&file);
        wid = in.readLine();
        file.close();
    }
    QString name = ui->salaryInput->text();
    CCompany C("GEREGA CORPORATION");
    C.ChangeWorkerSalary(name, wid.toInt());
    this->close();
}
```

userinfowindow.h

```
#ifndef USERINFOWINDOW_H
#define USERINFOWINDOW_H

#include <QDialog>

namespace Ui {
class UserinfoWindow;
}

class UserinfoWindow : public QDialog
{
    Q_OBJECT

public:
    explicit UserinfoWindow(QWidget *parent = nullptr);
    ~UserinfoWindow();

private slots:
    void on_pushButton_clicked();

private:
    Ui::UserinfoWindow *ui;
};
```

#endif // *USERINFOWINDOW_H*

userinfowindow.cpp

```
#include "userinfowindow.h"
#include "ui_userinfowindow.h"
#include "ccompany.h"
#include "usermenuwindow.h"


UserinfoWindow::UserinfoWindow(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::UserinfoWindow)
{
    ui->setupUi(this);
    CCompany C("GEREGA CORPORATION");
    QString s = QString::number(C.GetNWorkers());
    ui->nWorkers->setText(s);
}

UserinfoWindow::~UserinfoWindow()
{
    delete ui;
}

void UserinfoWindow::on_pushButton_clicked()
{
    this->close();
    UserMenuWindow w;
    w.setWindowFlags(Qt::Window | Qt::FramelessWindowHint);
    w.setModal(true);
    w.exec();
}
```

usermenuwindow.h

```
#ifndef USERMENUWINDOW_H
#define USERMENUWINDOW_H

#include <QDialog>

namespace Ui {
class UserMenuWindow;
}

class UserMenuWindow : public QDialog
{
    Q_OBJECT

public:
    explicit UserMenuWindow(QWidget *parent = nullptr);
    ~UserMenuWindow();

private slots:
    void on_pushButton_clicked();

    void on_pushButton_4_clicked();

    void on_pushButton_2_clicked();

    void on_pushButton_3_clicked();

    void on_pushButton_5_clicked();

private:
    Ui::UserMenuWindow *ui;
};
```

#endif // *USERMENUWINDOW_H*

usermenuwindow.cpp

```
#include "usermenuwindow.h"
#include "ui_usermenuwindow.h"
#include "ccompany.h"
#include "userinfowindow.h"
#include "userpickwindow.h"
#include "addprojectwindow.h"
#include "contactwindow.h"


UserMenuWindow::UserMenuWindow(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::UserMenuWindow)
{
    ui->setupUi(this);
    CCompany C("GEREGA CORPORATION");
    ui->tittle->setText(C.GetCompanyName());
}

UserMenuWindow::~UserMenuWindow()
{
    delete ui;
}

void UserMenuWindow::on_pushButton_clicked()
{
    //info
    this->close();
    UserinfoWindow w;
    w.setWindowFlags(Qt::Window | Qt::FramelessWindowHint);
    w.setModal(true);
    w.exec();
}


void UserMenuWindow::on_pushButton_4_clicked()
{
    //menu
    this->close();
    UserPickWindow w;
    w.setWindowFlags(Qt::Window | Qt::FramelessWindowHint);
    w.setModal(true);
    w.exec();
}


void UserMenuWindow::on_pushButton_2_clicked()
{
    //invidual
    AddProjectWindow w;
    w.setModal(true);
    w.exec();
}


void UserMenuWindow::on_pushButton_3_clicked()
{
    //contact
    ContactWindow w;
    w.setWindowFlags(Qt::Window | Qt::FramelessWindowHint);
```

```
    w.setModal(true);
    w.exec();
}


void UserMenuWindow::on_pushButton_5_clicked()
{
    this->close();

}
```

userpickwindow.h

```
#ifndef USERPICKWINDOW_H
#define USERPICKWINDOW_H

#include <QDialog>

namespace Ui {
class UserPickWindow;
}

class UserPickWindow : public QDialog
{
    Q_OBJECT

public:
    explicit UserPickWindow(QWidget *parent = nullptr);
    ~UserPickWindow();

private slots:
    void on_webButton_clicked();

    void on_androidButton_clicked();

    void on_windowsButton_clicked();

    void on_analButton_clicked();

    void on_server1Button_clicked();

    void on_server2Button_clicked();

    void on_server3Button_clicked();

    void on_pushButton_clicked();

private:
    Ui::UserPickWindow *ui;
};

#endif // USERPICKWINDOW_H
```

userpickwindow.cpp

```
#include "userpickwindow.h"
#include "ui_userpickwindow.h"
#include <QMessageBox>
#include "ccompany.h"
#include "usermenuwindow.h"


UserPickWindow::UserPickWindow(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::UserPickWindow)
```

```
{
    ui->setupUi(this);
}

UserPickWindow::~UserPickWindow()
{
    delete ui;
}

void UserPickWindow::on_webButton_clicked()
{
    QMessageBox::StandardButton reply = QMessageBox::question(this,
"Підтвердження",
                                                              "Ви впевнені що
хочете замовити \"" + ui->webButton->text() +"\"",
                                                              QMessageBox::Yes |
QMessageBox::No);
    if(reply ==  QMessageBox::Yes){
        CCompany C("GEREGA CORPORATION");
        C.AddProject("WebSite", 1);
    }
}


void UserPickWindow::on_androidButton_clicked()
{
    QMessageBox::StandardButton reply = QMessageBox::question(this,
"Підтвердження",
                                                              "Ви впевнені що
хочете замовити \"" + ui->androidButton->text()+"\"",
                                                              QMessageBox::Yes |
QMessageBox::No);
    if(reply ==  QMessageBox::Yes){
        CCompany C("GEREGA CORPORATION");
        C.AddProject("AndroidApp", 1);
    }
}


void UserPickWindow::on_windowsButton_clicked()
{
    QMessageBox::StandardButton reply = QMessageBox::question(this,
"Підтвердження",
                                                              "Ви впевнені що
хочете замовити \"" + ui->windowsButton->text()+"\"",
                                                              QMessageBox::Yes |
QMessageBox::No);
    if(reply ==  QMessageBox::Yes){
        CCompany C("GEREGA CORPORATION");
        C.AddProject("WindowsApp", 1);
    }
}


void UserPickWindow::on_analButton_clicked()
{
    QMessageBox::StandardButton reply = QMessageBox::question(this,
"Підтвердження",
                                                              "Ви впевнені що
хочете замовити \"" + ui->analButton->text()+"\"",
                                                              QMessageBox::Yes |
QMessageBox::No);
    if(reply ==  QMessageBox::Yes){
        CCompany C("GEREGA CORPORATION");
        C.AddProject("Analize", 1);
```

```
    }
}


void UserPickWindow::on_server1Button_clicked()
{
    QMessageBox::StandardButton reply = QMessageBox::question(this,
"Підтвердження",
                                                              "Ви впевнені що
хочете замовити \"" + ui->server1Button->text()+"\"",
                                                              QMessageBox::Yes |
QMessageBox::No);
    if(reply ==  QMessageBox::Yes){
        CCompany C("GEREGA CORPORATION");
        C.AddProject("ServerCreation", 1);
    }
}


void UserPickWindow::on_server2Button_clicked()
{
    QMessageBox::StandardButton reply = QMessageBox::question(this,
"Підтвердження",
                                                              "Ви впевнені що
хочете замовити \"" + ui->server2Button->text()+"\"",
                                                              QMessageBox::Yes |
QMessageBox::No);
    if(reply ==  QMessageBox::Yes){
        CCompany C("GEREGA CORPORATION");
        C.AddProject("ServerAdjustment", 1);
    }
}


void UserPickWindow::on_server3Button_clicked()
{
    QMessageBox::StandardButton reply = QMessageBox::question(this,
"Підтвердження",
                                                              "Ви впевнені що
хочете замовити \"" + ui->server3Button->text()+"\"",
                                                              QMessageBox::Yes |
QMessageBox::No);
    if(reply ==  QMessageBox::Yes){
        CCompany C("GEREGA CORPORATION");
        C.AddProject("ServerRent", 1);
    }
}


void UserPickWindow::on_pushButton_clicked()
{
    this->close();
    UserMenuWindow w;
    w.setWindowFlags(Qt::Window | Qt::FramelessWindowHint);
    w.setModal(true);
    w.exec();
}
```

workerinfowindow.h

```
#ifndef WORKERINFOWINDOW_H
#define WORKERINFOWINDOW_H

#include <QDialog>
```

```
namespace Ui {
class WorkerInfoWindow;
}

class WorkerInfoWindow : public QDialog
{
    Q_OBJECT

public:
    explicit WorkerInfoWindow(QWidget *parent = nullptr);
    ~WorkerInfoWindow();

private slots:
    void on_CNameButton_clicked();

    void on_CProfButton_clicked();

    void on_CSalaryButton_clicked();

    void on_exitButton_clicked();

private:
    Ui::WorkerInfoWindow *ui;
};
```

#endif // *WORKERINFOWINDOW_H*

workerinfowindow.cpp

```
#include "workerinfowindow.h"
#include "ui_workerinfowindow.h"
#include "ccompany.h"
#include <QFile>
#include "menuwindow.h"
#include "setnamewindow.h"
#include "ccompany.h"
#include "setprojectwindow.h"
#include "setsalarywindow.h"


WorkerInfoWindow::WorkerInfoWindow(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::WorkerInfoWindow)
{
    ui->setupUi(this);
    QString wid;
    QFile file("C:\\Users\\rostyk\\Documents\\kursova2\\tmp.txt");
    if(file.open(QIODevice::ReadOnly)){
        QTextStream in(&file);
        wid = in.readLine();
        file.close();
    }
    CCompany C("GEREGA CORPORATION");
    ui->nameLabel->setText(C.GetWorkerName(wid.toInt()));
    ui->profLabel->setText(C.GetWorkerProf(wid.toInt()));

    ui->projLabel->setText(C.GetWorkerProjectName(wid.toInt()));
    ui->salaryLabel->setText(C.GetWorkerSalary(wid.toInt()));
}

WorkerInfoWindow::~WorkerInfoWindow()
{
```

```
    delete ui;
}

void WorkerInfoWindow::on_CNameButton_clicked()
{
    SetNameWindow w;
    w.setModal(true);
    w.exec();
}


void WorkerInfoWindow::on_CProfButton_clicked()
{
    SetProjectWindow w;
    w.setModal(true);
    w.exec();
}


void WorkerInfoWindow::on_CSalaryButton_clicked()
{
    SetSalaryWindow w;
    w.setModal(true);
    w.exec();
}


void WorkerInfoWindow::on_exitButton_clicked()
{
    this->close();
    MenuWindow menu;
    menu.setWindowFlags(Qt::Window | Qt::FramelessWindowHint);
    menu.setModal(true);
    menu.exec();
}
```

5.2.Висновок

Використання ООП для вирішення подібних задач значно полегшує роботу та робить код більш структурованим та дозволяє легше його читати. Після виконання цієї роботи я зрозумів, що ООП – це гнучкий інструмент який ідеально підходить для написання програм які потребуються описання деяких об’єктів та їх поведінки.

Під час виконання даної курсової роботи я ознайомився з основними принципами об’єктно-орієнтованого програмування та навчився застосовувати їх принципи на практиці.

Навчився правильно використовувати класи та провів детальний аналіх предметної області, також навчився створювати графічний інтерфейс у QT Creator та покращив свої вміння в програмуванні.

Також навчився значно ефективніше знаходити інформацію в інтернеті та читати потрібну та цікаву літературу.

5.3 Список літератури

Використана література : [1] Кнут Д. «Мистецтво програмування». І.Д. Вільямс, 2007.

[2] Є.В. Пишкін. «Основні концепції та механізми ООП». БХВ-Петербург, 2005.

[3] Греді Буч, Роберт А. Максимчук, Майкл У. Енгл, Боббі Дж. Янг, Джим Коналл, Келлі А. Х'юстон. «Об'єктно-орієнтований аналіз та проектування», І.Д. Вільямс, 2010.

[4] Бьярн Страуструп «Мова програмування C++» (четверте видання), Addison–Wesley, 2013