Зміст

ПЕРЛІК СКОРОЧЕНЬ

ООП – Об’єктно орієнтоване програмування

БД – База даних

АРМ – Автоматизоване робоче місце

ПЗ – Програмне забезпечення

ТЗ – Технічне завдання

ВСТУП

За останні декілька десятиліть, у зв’язку з розвитком інформаційних технологій, програмне забезпечення охоплює майже всі сфери діяльності людини. ПЗ дозволяє значно спростити діяльність людей і надає їм можливість оптимізації рутинної роботи і також знижує кількість помилок до мінімуму, зокрема повністю виключаючи помилки зв’язані з людським фактором. Створення замовлень, додавання клієнтів та їх даних в списки, розробка звітів та пошук потрібної інформації – це доволі складний та довготривалий процес на рівні звичайних текстових редакторів або електронних таблиць. Для пришвидшення даного процесу доцільно автоматизувати його за допомогою ПЗ.

Під час розробки програми основною метою є автоматизація занесення інформації та реалізація пошуку її за певними атрибутами. Користувачем цієї програми має бути працівник, робота якого має бути автоматизована.

Головною ціллю будь якої АРМ полягає в спрощенні роботи працівника в реальному світі. В цій роботі ми будемо спрощувати роботу працівника, який належить до нашої предметної області.

1 АНАЛІЗ ТЕХНІЧНОГО ЗАВДАННЯ

1.1 Визначення необхідності створення ПЗ

Дана АРМ дозволить менеджеру СТО вести зручний облік клієнтів, працівників та замовлень. Також дана АРМ забезпечить зберігання даних в окремих файлах, що дозволить робити резервні копії інформації.

1.2 Постановка задачі

Основним і єдиним користувачем програми буде менеджер станції технічного обслуговування. Для захисту в нього буде авторизація в систему згідно логіну та паролю, який він здатен змінити. Також менеджер може додати клієнта і його дані, дані автомобіля клієнта, працівника і також – створити замовлення на ремонт.

В кожного клієнта є дані які необхідно зберігати. Такими даними є:

- Прізвище
- Ім’я
- Номер телефону
- Автомобіль

Відповідно в автомобіля також є характеристики, якими автомобілі можна відрізнити один від одного:

- Марка
- Модель
- Колір
- Номерний знак

Ще в системі передбачено зберігання даних працівника:

- Прізвище
- Ім’я
- Спеціалізація
- Зарплата

Замовлення – найважливіша структура яку доведеться створювати та зберігати менеджерові. В неї такі атрибути:

- Авто
- Працівник
- Клієнт
- Вартість
- Тип поломки
- Короткий опис поломки
- Дата завершення ремонту
- Стан виконання

2 ОБҐРУНТУВАННЯ АЛГОРИТМУ Й СТРУКТУРИ ПРОГРАМИ

Розробка програми буде на мові C++ в середовищі CodeBlocks.

Проаналізувавши технічне завдання можна скласти алгоритм та структуру програми. Роботу програми буде реалізовано за допомогою циклу з умовою, за допомогою якого можна забезпечити роботу програми до того моменту, доки цього не захоче користувач. Основні функції, які повинна буде виконувати програма будуть в меню програми, з якого менеджер зможе переміщатися між ними.

Такими основними функціями є:

- Додати замовлення
- Змінити/Видалити замовлення
- Додати/Видалити клієнта
- Додати/Звільнити працівника
- Змінити логін та пароль
- Вихід

Структура програми здійснюється з дотриманням усіх концепцій ООП.

В процесі розробки доведеться використовувати різноманітні структури даних для ефективної роботи з даними. В програмі будуть застосовані вектори для збереження об’єктів.

Вектор – це структура, яка зберігає елементи в динамічний масив даних. Для взаємодії з вектором даних в програмі є бібліотека з додатковими функціями, які спрощують роботу з даною структурою.

Також в роботі використав ітератор для зручного обходу векторів в циклах для зчитування та запису даних.

Ітератор – це спеціальна структура даних, яка дозволяє легко обходити масив даних.

3 РОЗРОБЛЕННЯ ПРОГРАМИ

3.1 Розроблення системи класів

Клас – це спеціальна конструкція, яка пов’язує об’єкти за спільними характеристиками та функціями. Класи в ООП є невід’ємною частиною програми.

Проаналізувавши попередню інформацію можна сформувати такі класи як:

- Авто
- Клієнт
- Працівник
- Замовлення
- Менеджер

Також у кожного класу будуть конструктори, за допомогою яких буде зручніше створювати об’єкти класу.

В класі замовлення будуть методи за допомогою яких можна буде змінити певні параметри атрибутів, такі як:

- Зміна вартості
- Зміна короткого опису поломки
- Позначення виконання роботи
- Зміна працівника

Між класами є різні зв’язки:

- Асоціація – зв’язок між класами
- Агрегація – коли один клас має вищий ранг ніж інший
- Композиція – строга агрегація, менший за рангом клас не може існувати без вищого за рангом

Продемонструвати класи, їх атрибути, методи та зв’язки між ними найкраще в діаграмі класів(Рис 3.1.1).

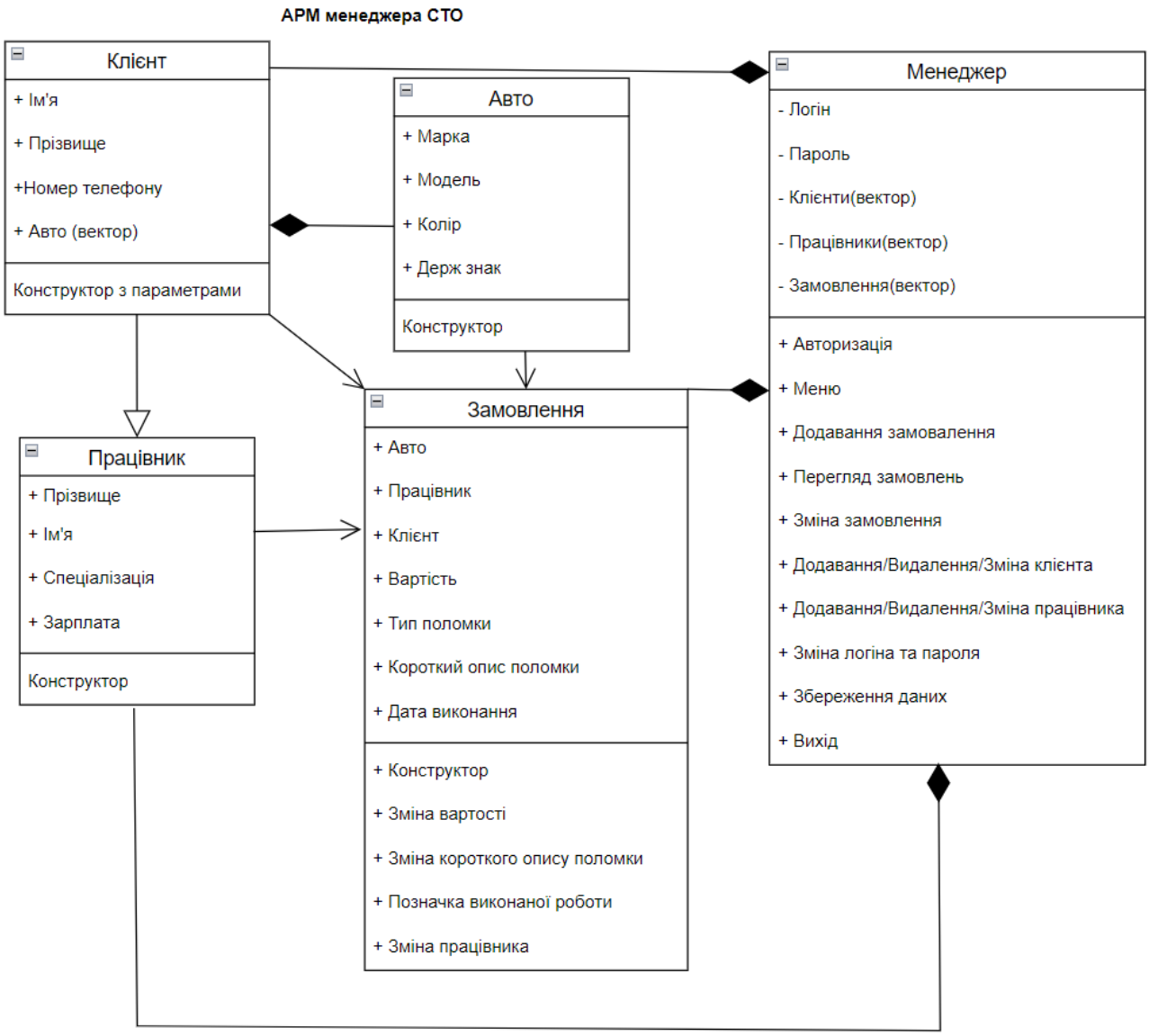

Рис. 3.1.1 Діаграма класів

Для демонстрації можливостей користувача була створена діаграма прецедентів(UML-діаграма)(Рис 4.1.2) в якій відображається сутності і їх прецеденти.

Сутність(актор) – роль, яка виконується сутністю яка взаємодіє з системою. В сутностей є прецеденти.

Прецедент – функціональна вимога, яку може виконати актор, простими словами функція.

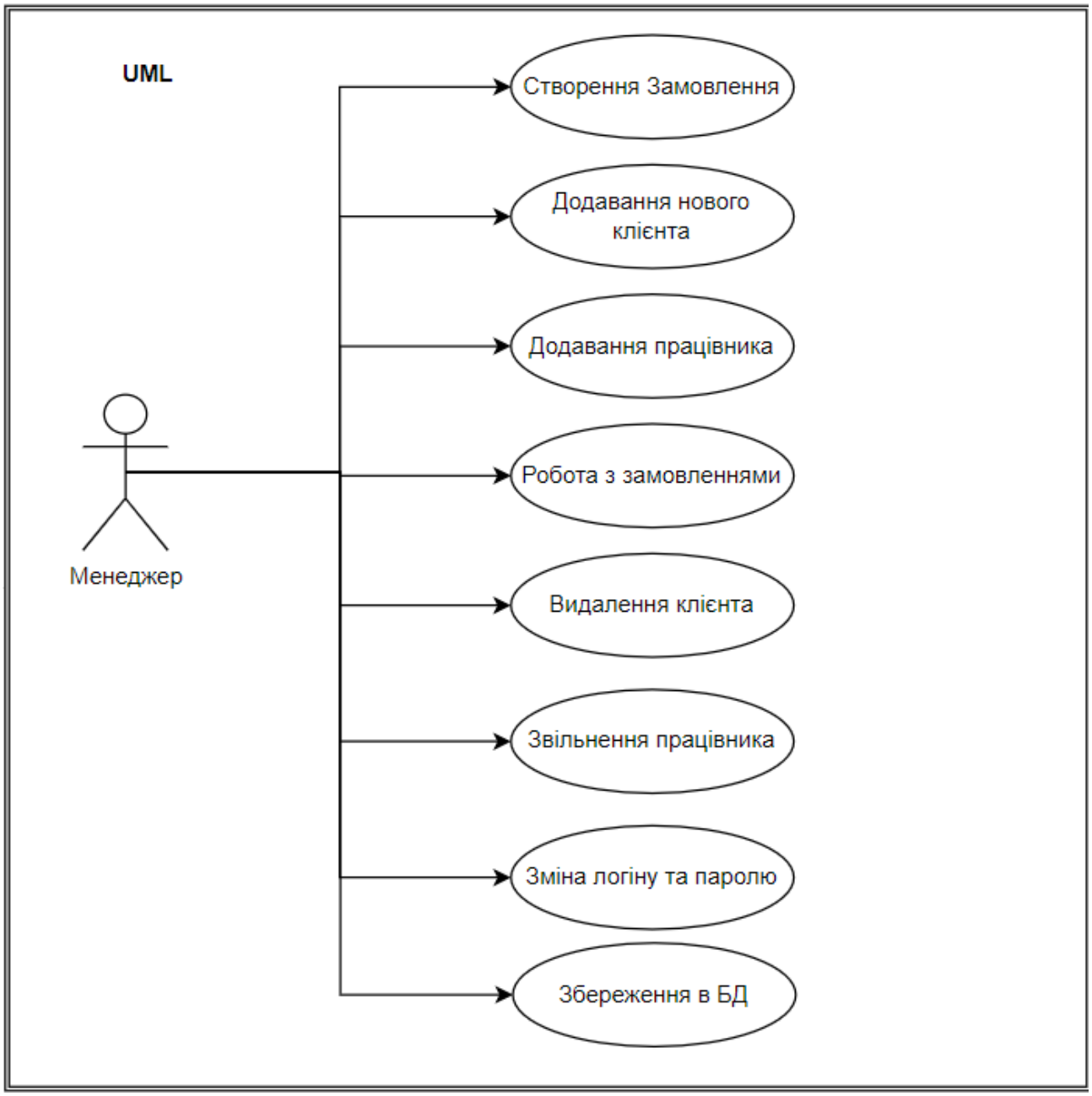

Рис. 4.1.2 UML діаграма.

3.2 Розробка методів

Розглянемо декілька методів в класі Manager.

Метод авторизація приймає з консолі дві рядкові змінні та порівнює їх з логіном та паролем, якщо вони співпадають, тоді користувач потрапляє в меню, якщо ні, то програма просить ввести логін та пароль ще раз.

```
    void Manager::autorithation(){
        cout<<"================================================"<<endl;
        cout<<"|\t\t  Авторизація  \t\t\t|\n=====================================
==========\n\n";
        string l,p;
        do{
        cout<<"  Введіть логін:  ";
        cin>>l;
        cout<<"  Введіть пароль: ";
        cin>>p;
        clearConsole();
        if(l!=login||p!=password){
            cout<<"================================================"<<endl;
            cout<<"|\tНеправильно введено логін або пароль\t|\n";
            cout<<"================================================\n\n";
        }
        }while(l!=login||p!=password);
        cout<<"================================================"<<endl;
        cout<<"|\t\tВітаю вас в системі\t\t|"<<endl;
        cout<<"================================================"<<endl<<endl<<endl
;
    }
```

Метод меню. Він передбачає виведення меню та можливість переходу між різними пунктами меню за допомогою введення цифри пункту меню. Даний метод реалізовується за допомогою оператора switch і повертає значення bool для циклу з умовою в головні й функції.

```
   bool Manager::main(){

        if(auth==0){
            sqlite3 *db;
            sqlite3_open("database.db", &db);
            readDataFromDatabase(db, clients, employeers, orders, login, password);
            autorithation();
            sqlite3_close(db);

            auth++;
        }
        int a;
        cout<<"================================================"<<endl;
```

```
            cout<<"|\t\t\tМеню\t\t\t|\n";
            cout<<"================================================"<<endl;
            cout<<"================================================\n";
            cout<<"|  1. Додати замовлення.\t\t\t|\n";
            cout<<"|  2. Змінити/Переглянути замовлення.\t\t|\n";
            cout<<"|  3. Додати/Звільнити працівника.\t\t|\n";
            cout<<"|  4. Додати/Видалити клієнта.\t\t\t|\n";
            cout<<"|  5. Змінтити логін та пароль.\t\t\t|\n";
            cout<<"|  6. Зберегти дані в БД.\t\t\t|\n";
            cout<<"|  0. Вихід.\t\t\t\t|\n";
            cout<<"================================================\n";
            cout<<"\n  Виберіть пункт меню: ";
            cin>>a;
            clearConsole();
            sqlite3 *db;
            switch(a){
                case 1:
                    addOrder();
                    clearConsole();
                    return false;break;
                case 2:
                    checkO();
                    clearConsole();
                    return false;break;
                case 3:
                    add_del_emp();
                    clearConsole();
                    return false;break;
                case 4:
                    add_del_client();
                    clearConsole();
                    return false;break;
                case 5:
                    changePass();
                    return false;break;
                case 6:

                    sqlite3_open("database.db", &db);
                    writeDataToDatabase(db, clients,
                              employeers, orders, login, password);
                    sqlite3_close(db);
                    clearConsole();
                    cout<<"================================================"<<endl;
                    cout<<"|\tІнформацію успішно збережено!\t\t|"<<endl;
                    cout<<"================================================"<<endl<<en
dl<<endl;
                    return false;break;
                case 0:
                    clearConsole();
```

```
                cout<<endl<<endl<<endl<<"Чи хочете зберегти незбережені дані(1-так,
0-ні): ";
                cin>>auth;
                if(auth==1){
                sqlite3_open("database.db", &db);
                writeDataToDatabase(db, clients,
                          employeers, orders, login, password);
                sqlite3_close(db);}
                return true;
            default:
                clearConsole();
                cout<<"================================================="<<endl;
                cout<<"|\tТакого пункту меню не існує\t\t|\n";
                cout<<"================================================="<<endl;
                return false;
                }
```

Метод додавання замовлення. В методі вибираються клієнт, його авто, працівник за допомогою методів, додаткові дані вводяться з консолі. Після збору всієї інформації замовлення додається в вектор за допомогою функції push_back().

```
        void Manager::addOrder(){
            cout<<"================================================="<<endl;
            cout<<"|\t\tДодавання замовлення\t\t|\n";
            cout<<"================================================="<<endl<<endl;
            cout<<"================================================="<<endl;
            cout<<"|\t\tВиберіть клієнта\t\t|\n";
            cout<<"================================================="<<endl;
            int c=client_chose();
            int a;
            clearConsole();
            cout<<"================================================="<<endl;
            cout<<"|\t\tДодавання замовлення\t\t|\n";
            cout<<"================================================="<<endl<<endl;
            cout<<"================================================="<<endl;
            cout<<"|\t\tВиберіть Авто клієнта\t\t|\n";
            cout<<"================================================="<<endl;
            a=auto_chose(c);
            clearConsole();
            cout<<"================================================="<<endl;
            cout<<"|\t\tДодавання замовлення\t\t|\n";
            cout<<"================================================="<<endl<<endl;
            cout<<"================================================="<<endl;
            cout<<"|\t\tВиберіть працівника\t\t|\n";
            cout<<"================================================="<<endl;
            int e=employee_chose();
            clearConsole();
            cout<<"================================================="<<endl;
            cout<<"|\t\tДодавання замовлення\t\t|\n";
            cout<<"================================================="<<endl<<endl;
```

```
            cout<<"  Введіть вартість ремонту:      ";
            double p;
            cin>>p;
            cout<<"  Введіть тип поломки:           ";
            string t;
            cin>>t;
            cout<<"  Введіть короткий опис поломки: ";
            string prob;
            getline(cin>>ws, prob);
            cout<<endl;
            orders.push_back(Order(clients[c].automobile[a],employeers[e],clients[c
],p,t,prob));
        }
```

Метод зміни логіну та паролю. Приймає з консолі дві змінні і записує їх в логін та пароль.

```
        void Manager::changePass(){
            cout<<"================================================"<<endl;
            cout<<"|\t\tЗміна логіна та пароля:\t\t|"<<endl;
            cout<<"================================================"<<endl<<endl;
               string l,p;
               cout<<"  Введіть новий Логін:  ";
               cin>>l;
               cout<<"  Введіть новий Пароль: ";
               cin>>p;
               login=l;
               password=p;
               clearConsole();
               cout<<"================================================"<<endl;
               cout<<"|\tЛогін та пароль успішно змінений\t|";
               cout<<endl<<"================================================"<<en
dl<<endl;
            }
```

Метод вибір клієнта. Якщо в векторі немає клієнта то автоматично запускається метод на додавання клієнта. Потім відображається список клієнтів і після введення номеру клієнта він повертається. Також можна обрати цифру 0 і додати ще клієнтів.

```
int Manager::client_chose(){
            if(clients.size()==0){
                cout<<"================================================"<<endl;
                cout<<"|Список клієнтів порожній, додайте нового клієнта|\n";
                cout<<"================================================"<<endl<<en
dl;
                cout<<"Натисніть Enter, щоб продовжити >>";
                getch();
                clearConsole();
```

```
            addClient();
            clearConsole();
            Auto A1=addAuto();
            clients[0].automobile.push_back(A1);
            clearConsole();
            }
        long long unsigned int i;
        cout<<"================================================="<<endl;
        cout<<"|\t\tСписок клієнтів\t\t\t|\n";
        cout<<"================================================="<<endl<<endl;
        cout<<"-------------------------------------------------";
        cout<<"\n  0. Додати нового клієнта.\n";
        for(i=0;i<clients.size();i++){
            cout<<"-------------------------------------------------"<<endl;
            cout<<"  "<<i+1<<". Ім'я:\t\t"<<clients[i].name
            <<"\n     Прізвище:\t\t"<<clients[i].surname
            <<"\n     Номер телефону:\t"<<clients[i].phone<<endl;
            cout<<"-------------------------------------------------"<<endl;
        }
        cout<<endl<<endl<<"  Введіть номер клієнта: ";
        cin>>i;
        if(i==0){
            clearConsole();
            addClient();
            i=clients.size()-1;
            clearConsole();
            Auto A1=addAuto();
            clients[i].automobile.push_back(A1);
            clearConsole();
            return client_chose();
        }
        return i-1;
    }
```

3.3 Створення об’єктів і розроблення головної програми

Дана АРМ розроблена як багатофайловий проект, складається з наступних файлів:

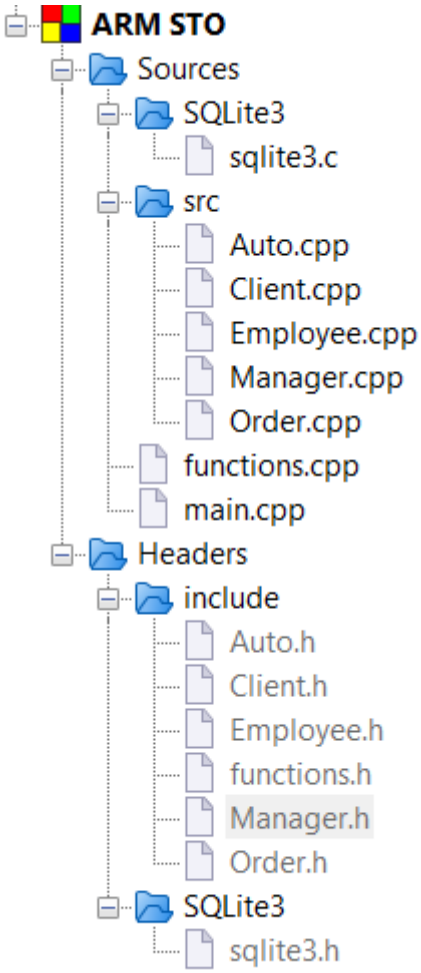

Рис. 3.3.1

Заголовкові файли:

- Auto.h – заголовковий файл класу «Авто»
- Client.h – заголовковий файл класу «Клієнт»
- Employee.h – заголовковий файл класу «Працівник»
- Manger.h – заголовковий файл класу «Менеджер»
- Order.h – заголовковий файл класу «Замовлення»
- functions.h – заголовковий файл для додаткових функцій
- sqlite3.h – заголовковий файл для БД

Файли вихідного коду:

- Auto.cpp – файл вихідного коду класу «Авто»

- Client.cpp – файл вихідного коду класу «Клієнт»
- Employee.cpp – файл вихідного коду класу «Працівник»
- Manger.cpp – файл вихідного коду класу «Менеджер»
- Order.cpp – файл вихідного коду класу «Замовлення»
- Sqlite3.c – файл для БД
- Functions.cpp – файл з кодом додаткових функцій
- Main.cpp – файл головної функції програми

Використані бібліотеки:

- iostream – для виведення і введення інформації в консолі
- vector – для роботи з веторами
- string – для рядкових змінних
- windows.h – для підключення відображення кирилиці в консолі

3.4 Опис бази даних та інтерфейсу програми

В даному ПЗ використовується БД SQLite за допомогою якої реалізується збереження і зчитування інформації в додатку. Для підключення цієї БД було створено додатковий файл, в якому було прописано основні функції для роботи з БД.

Щоб БД працювала обов’язково підключити такі файли:

- sqlite3.c
- sqlite.h

Для запису, зчитування створено файл database.db та в класі створено методи запису і зчитування і для них було створено дві функції для обробки запитів.

Функція запису у БД видаляє існуючі таблиці з даними з файлу, створює заново таблиці і заносить в них дані, які потрібно зберігати. Створюються такі таблиці:

- Авто(ID клієнта, марка, модель, колір, державний знак)
- Клієнти(Ім’я, Прізвище, Номер телефону)

- Замовлення(ID клієнта, ID авто, ID працівника, вартість ремонту, короткий опис поломки, тип поломки, дата виконання, статус виконання)
- Працівники(Ім’я, Прізвище, Спеціалізація, Зарплатня)
- Авторизаційні дані(логін, пароль)

Код функції запису:

```
void writeDataToDatabase(sqlite3* db, const vector<Client>& clients,
                         const vector<Employee>& employeers, const vector<Order>&
orders, string login, string password) {
    // Видалення таблиць, якщо вони існують
    executeQueryForCreateAndDrop(db, "DROP TABLE IF EXISTS Auto;");
    executeQueryForCreateAndDrop(db, "DROP TABLE IF EXISTS LogPass;");
    executeQueryForCreateAndDrop(db, "DROP TABLE IF EXISTS Clients;");
    executeQueryForCreateAndDrop(db, "DROP TABLE IF EXISTS Employees;");
    executeQueryForCreateAndDrop(db, "DROP TABLE IF EXISTS Orders;");
    // Створення таблиць Auto, Clients, Employees, Orders
    executeQueryForCreateAndDrop(db, "CREATE TABLE Auto (client_id INT, marka TEXT,
model TEXT, color TEXT, znak TEXT);");
    executeQueryForCreateAndDrop(db, "CREATE TABLE Clients (name TEXT, surname
TEXT, phone TEXT);");
    executeQueryForCreateAndDrop(db, "CREATE TABLE LogPass (login TEXT, password
TEXT);");
    executeQueryForCreateAndDrop(db, "CREATE TABLE IF NOT EXISTS Employees (name
TEXT, surname TEXT, specialization TEXT, salary REAL);"
        "CREATE TABLE Orders (auto_id INTEGER, employee_id INTEGER, client_id
INTEGER, price REAL, "
        "type_of_repair TEXT, short_problem TEXT, execution_date_day INTEGER,
execution_date_month INTEGER, execution_date_year INTEGER, complete INTEGER);");
    //Запис логіну та пароля
    string LogPassQuery="INSERT INTO LogPass (login, password) VALUES( "+ login +
", " + password + ");";
    executeQueryForCreateAndDrop(db, LogPassQuery);
    // Запис даних в таблицю Auto
    for(int c=0;c<clients.size();c++){
    for (const auto& aut : clients[c].automobile) {
        string query = "INSERT INTO Auto (client_id ,marka, model, color, znak)
VALUES ( "+ to_string(c) +",'" + aut.marka + "', '" +
                       aut.model + "', '" + aut.color + "', '" + aut.znak + "');";
        executeQueryForCreateAndDrop(db, query);
    }}
    // Запис даних в таблицю Clients
    for (const auto& client : clients) {
        string query = "INSERT INTO Clients (name, surname, phone) VALUES ('" +
client.name + "', '" + client.surname +
                       "', '" + client.phone + "');";
        executeQueryForCreateAndDrop(db, query);
    }
```

```
    // Запис даних в таблицю Employees
    for (const auto& employee : employeers) {
        string query = "INSERT INTO Employees (name, surname, specialization,
salary) VALUES ('" + employee.name +
                       "', '" + employee.surname + "', '" + employee.specialization
+ "', " +
                       to_string(employee.salary) + ");";
        executeQueryForCreateAndDrop(db, query);
    }

    // Запис даних в таблицю Orders
    for (const auto& order : orders) {
        // Отримання ідентифікаторів об'єктів з інших таблиць
        int autoId = executeQueryForCreateAndDrop(db, "SELECT rowid FROM Auto WHERE
marka='" + order.automobile.marka +
                                                  "' AND model='" + order.automobile.model
+ "' AND color='" +
                                                  order.automobile.color + "' AND znak='" +
order.automobile.znak + "';");

        int employeeId = executeQueryForCreateAndDrop(db, "SELECT rowid FROM
Employees WHERE name='" + order.employee.name +
                                                   "' AND surname='" +
order.employee.surname + "' AND specialization='" +
                                                   order.employee.specialization + "'
AND salary=" +
                                                   to_string(order.employee.salary) +
";");

        int clientId = executeQueryForCreateAndDrop(db, "SELECT rowid FROM Clients
WHERE name='" + order.client.name +
                                                 "' AND surname='" +
order.client.surname + "' AND phone='" +
                                                 order.client.phone + "';");

        string query =
            "INSERT INTO Orders (auto_id, employee_id, client_id, price,
type_of_repair, short_problem, "
            "execution_date_day, execution_date_month, execution_date_year,
complete) VALUES ("+ to_string(autoId) +
            ", " + to_string(employeeId) + ", " + to_string(clientId) + ", " +
            to_string(order.price) + ", '" + order.type_of_repair + "', '" +
order.short_problem + "', '" +
            to_string(order.execution_date.day) + "', '" +
to_string(order.execution_date.month) + "', '" +
to_string(order.execution_date.year) + "', " + (order.complete ? "1" : "0") + ");";
        executeQueryForCreateAndDrop(db, query);
    }
}
```

Для обробки даних з БД також потрібна функція зчитування. Вона зчитує всі дані з таблиць і напряму записує їх у вектори та змінні.

Код функції зчитування:

```
void readDataFromDatabase(sqlite3* db, vector<Client>& clients,
                          vector<Employee>& employeers, vector<Order>& orders,
string& login, string& password) {

    // Зчитування логіна та пароля

    string selectLPQuery="SELECT login, password FROM LogPass;";
    vector<vector<string>> LogPassResults;

    if(executeQuery(db, selectLPQuery, LogPassResults)== SQLITE_OK){
    login=LogPassResults[0][0];
    password=LogPassResults[0][1];
    }
    // Зчитування таблиці Clients
    string selectClientsQuery = "SELECT name, surname, phone FROM Clients;";
    vector<vector<string>> clientsResults;
    executeQuery(db, selectClientsQuery, clientsResults);
    for (const auto& row : clientsResults) {
        Client client;
        client.name = row[0];
        client.surname = row[1];
        client.phone = row[2];
        clients.push_back(client);
    }

    // Зчитування таблиці Auto
    string selectAutoQuery = "SELECT client_id, marka, model, color, znak FROM
Auto;";
    vector<vector<string>> autoResults;
    executeQuery(db, selectAutoQuery, autoResults);
    for (const auto& row : autoResults) {
        Auto aut;
        int c=stoi(row[0]);
        aut.marka = row[1];
        aut.model = row[2];
        aut.color = row[3];
        aut.znak = row[4];
        clients[c].automobile.push_back(aut);
    }

    // Зчитування таблиці Employees
    string selectEmployeesQuery = "SELECT name, surname, specialization, salary
FROM Employees;";
    vector<vector<string>> employeesResults;
    executeQuery(db, selectEmployeesQuery, employeesResults);
    for (const auto& row : employeesResults) {
```

```
        Employee employee;
        employee.name = row[0];
        employee.surname = row[1];
        employee.specialization = row[2];
        employee.salary = stod(row[3]);
        employeers.push_back(employee);
    }

// Зчитування таблиці Orders

    string selectOrdersQuery =
        "SELECT auto_id, employee_id,client_id, price, type_of_repair,
short_problem, execution_date_day, execution_date_month, execution_date_year,
complete "
        "FROM Orders;";
    vector<vector<string>> ordersResults;
    executeQuery(db, selectOrdersQuery, ordersResults);
    for (const auto& row : ordersResults) {
        int a=stoi(row[0]);
        int e=stoi(row[1]);
        int c=stoi(row[2]);

        Order order;
        order.automobile = clients[c].automobile[a];
        order.employee = employeers[e];
        order.client = clients[c];
        order.price = stod(row[3]);
        order.type_of_repair = row[4];
        order.short_problem = row[5];
        order.execution_date.day = stoi(row[6]);
        order.execution_date.month = stoi(row[7]);
        order.execution_date.year = stoi(row[8]);
        order.complete = (row[9] == "1");
        orders.push_back(order);
    }
}
```

Інтерфейс програми реалізований в консолі.

4 ТЕСТУВАННЯ ПРОГРАМИ І РЕЗУЛЬТАТИ ЇЇ ВИКОНАННЯ

Запуск програми, авторизація(Рис. 4.1).

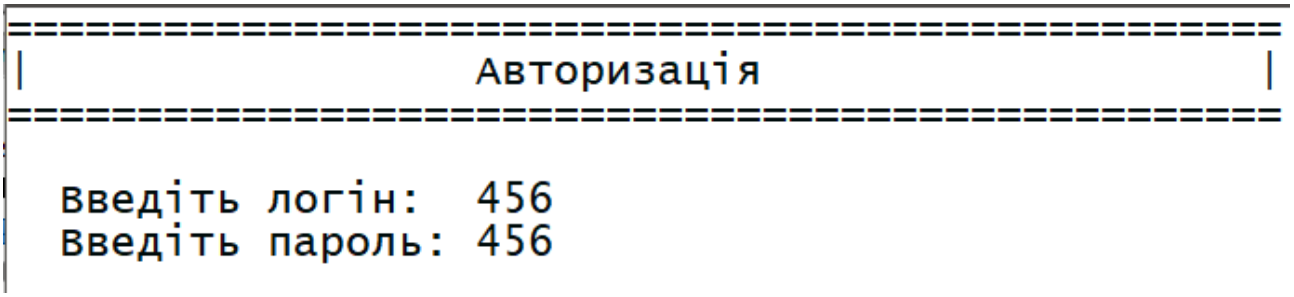

Рис. 4.1 Авторизація

Меню програми з основними функціями(Рис. 4.2).

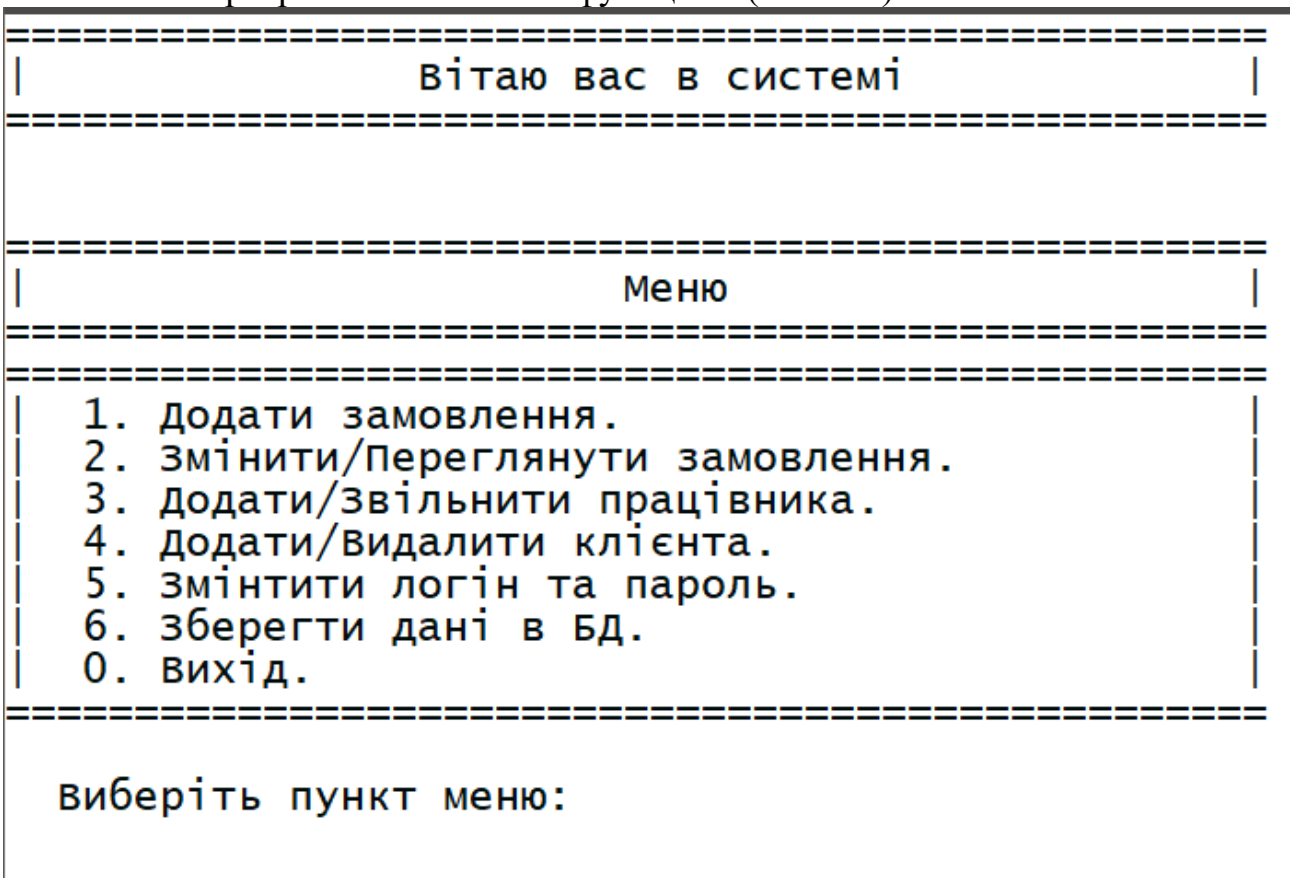

Рис. 4.2 Меню

Додати замовлення: Вибір клієнта(Рис. 4.3).

```
================================================
|              Додавання замовлення            |
================================================

================================================
|              Виберіть клієнта                |
================================================
================================================
|              Список клієнтів                 |
================================================

------------------------------------------------
  0. Додати нового клієнта.
------------------------------------------------
  1. Ім'я:              Володимир
     Прізвище:          Онищук
     Номер телефону:    +380672505390
------------------------------------------------
------------------------------------------------
  2. Ім'я:              Володимир
     Прізвище:          Зеленський
     Номер телефону:    +38000000000
------------------------------------------------


  Введіть номер клієнта: 2
```

Рис. 4.3 Вибір клієнта

Далі відбувається вибір авто клієнта(Рис. 4.4).

```
================================================
|              Додавання замовлення            |
================================================

================================================
|              Виберіть Авто клієнта           |
================================================
================================================
|              Список автомобілів              |
================================================

------------------------------------------------
  0. Додати новий автомобіль клієнту.
------------------------------------------------
  1. Марка:             Тесла
     Модель:            МодельX
     Колір:             Білий
     Державний знак:    AT7887YE
------------------------------------------------
------------------------------------------------
  2. Марка:             Ауді
     Модель:            TT
     Колір:             Сірий
     Державний знак:    AA7777AA
------------------------------------------------


  Введіть номер авто: 2
```

Рис. 4.4 Вибір авто клієнта

Після цього відбувається вибір працівника(Рис 4.5).

```
================================================
|              Додавання замовлення            |
================================================

================================================
|              Виберіть працівника             |
================================================
================================================
|              Список працівників              |
================================================

------------------------------------------------
  0. Додати нового працівника.
------------------------------------------------
  1. Ім'я:              Назар
     Прізвище:          Коваль
     Спеціалізація:     Слюсар
     Зарплата:          15000
------------------------------------------------
------------------------------------------------
  2. Ім'я:              Костянтин
     Прізвище:          Чоботар
     Спеціалізація:     Електрик
     Зарплата:          25000
------------------------------------------------

  Введіть номер працівника: 1
```

Рис. 4.5 Вибір працівника

І далі вводяться вартість ремонту, тип поломки та короткий опис поломки(Рис. 4.6).

```
================================================
|              Додавання замовлення            |
================================================

  Введіть вартість ремонту:      5000
  Введіть тип поломки:           Кузов
  Введіть короткий опис поломки: Ремонт переднього бампера
```

Рис. 4.6 Додавання замовлення

Додавання/Видалення/Зміна даних клієнта(Рис. 4.7).

```
================================================
|       Додати/Видалити/Змінити клієнта        |
================================================

================================================
|  1. Додати клієнта.                          |
|  2. Видалити клієнта.                        |
|  3. Змінити дані клієнта.                    |
|  0. Меню                                     |
================================================


  Виберіть пункт меню:  1
```

Рис. 4.7 Додавання/Видалення/Зміна даних клієнта

Додавання клієнта, внесення ім’я, прізвища, номеру телефону(Рис. 4.8).

```
================================================
|                Додавання клієнта             |
================================================

Введіть ім'я:            Максим
Введіть прізвище:        Залізняк
Введіть номер телефону: +380987456123
```

Рис. 4.8 Додавання клієнта

Додавання авто клієнту(Рис. 4.9).

```
================================================
|                Додавання Авто                |
================================================

  Введіть марку авто:          Порше
  Введіть модель авто:         Тайкан
  Введіть колір авто:          Оранжевий
  Введіть номерний знак авто: СЕ1234НТ
```

Рис. 4.9 Додавання Авто

Зміна логіна та пароля(Рис. 4.10).

```
"C:\Users\sahar\Desktop\=ютр яряър (2)\╣єЁёютр ╫╫╣\project\ARM STO\bin\Debug\ARM STO.exe"
================================================
|               Зміна логіна та пароля:        |
================================================

  Введіть новий Логін:  456
  Введіть новий Пароль: 456
```

Рис. 4.10 Зміна логіна та пароля

Меню для зміни даних працівника(Рис. 4.11).

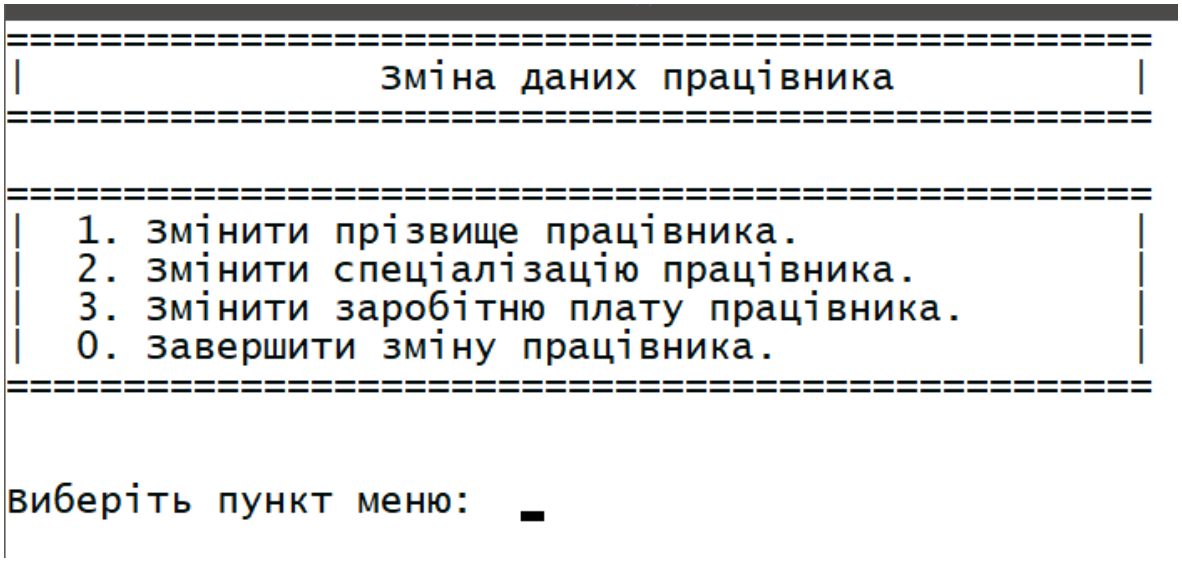

Рис. 4.11 Зміна даних працівника

Меню зміни замовлень та перегляду виконаних та не виконаних замовлень(Рис. 4.12).

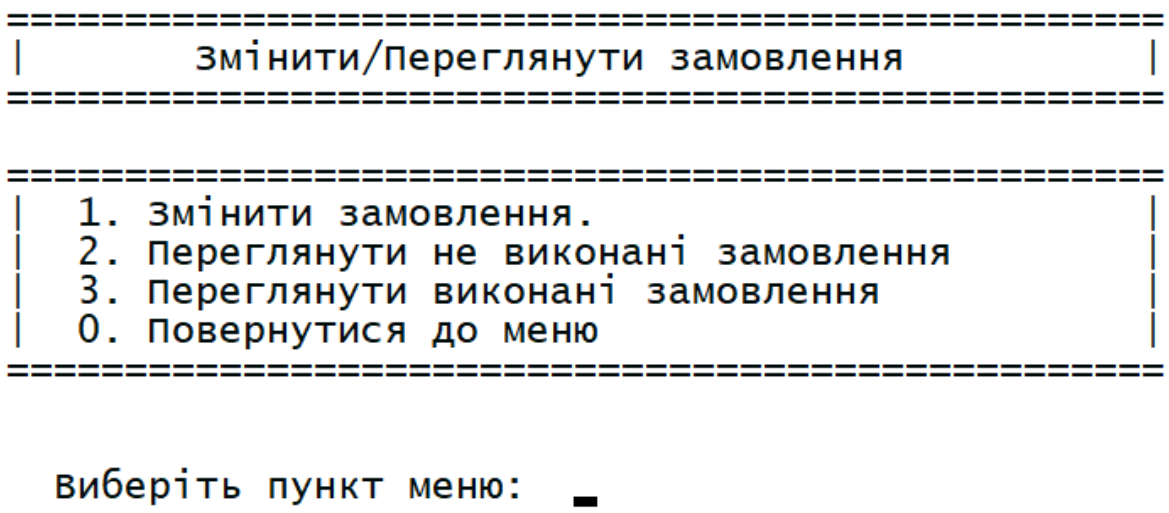

Рис. 4.12 Змінити/Переглянути замовлення

Вихід з програми(Рис. 4.13).

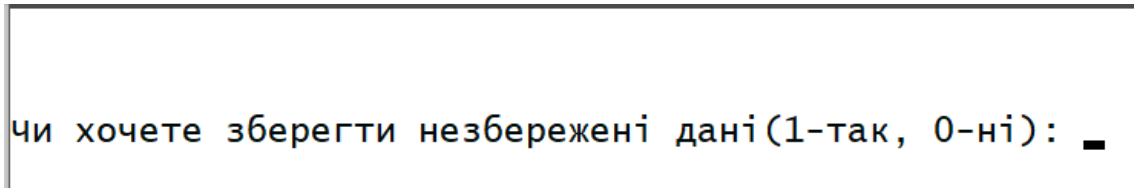

Рис. 4.13 Вихід з програми

Виявлені помилки при тестуванні програми:

- Не передбачено перевірку на введення інформації в консолі.
- Можливі викиди з програми при довгому використанні.
- Потреба у використанні програми кваліфікованим працівником.

ВИСНОВОК

Виконуючи дану курсову роботу ознайомився з базою даних SQLite та навчився використовувати її в проекті. Виконав динамічне збереження інформації в програмі, навчився працювати з методами консольного очищення.

Розробив ПЗ для АРМ менеджера СТО, яке здатне спростити ведення «паперової» роботи менеджерові та забезпечить його від помилок.

Дане ПЗ не є досконале і навпаки потребує вдосконалення, можна додати ще безліч методів, які забезпечуватимуть збереження більшої кількості інформації та дозволятимуть ще більше спростити роботу. Ще можна додати графічний дизайн.

ЛІТЕРАТУРА

1. Страуструп Б. Мова програмування C++ / Б'ярн Страуструп. – Мельбурн: Addison-Wesley, 1991. – 686 с.
2. Страуструп Б. Програмування: принципи та практика використання C++ / Б'ярн Страуструп. – Мельбурн: Addison-Wesley, 2014. – 1274 с.
3. Лафоре Р. Объектно-ориентированное программирование в C++ / Роберт Лафоре. – Санкт-Петербург: "Питер", 2013. – 926 с.

Інформаційні джерела:

1. Документація для роботи з векторами URL: https://en.cppreference.com/w/cpp/container/vector
2. Інструкція з підключення БД до програми URL: https://www.tutorialspoint.com/sqlite/sqlite_c_cpp.htm

ДОДАТОК А

Main.cpp

```
#include <windows.h>
#include "Manager.h"
using namespace std;

int main(){
    system("chcp 1251 > nul");
    Manager m1;
    bool exit=false;
    while(!exit){
        exit=m1.main();
    }
    return 0;
}
```

functions.cpp

```
#include <iostream>
#include <windows.h>
#include <vector>
#include <string>
#include <conio.h>
#include "sqlite3.h"
#include "Auto.h"
#include "Employee.h"
#include "Client.h"
#include "Order.h"
#include "Manager.h"

using namespace std;

    void clearConsole() {
    HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
    COORD coordScreen = {0, 0};
    DWORD cCharsWritten;
    CONSOLE_SCREEN_BUFFER_INFO csbi;
    DWORD dwConSize;

    GetConsoleScreenBufferInfo(hConsole, &csbi);
    dwConSize = csbi.dwSize.X * csbi.dwSize.Y;
    FillConsoleOutputCharacter(hConsole, ' ', dwConSize, coordScreen,
&cCharsWritten);
    GetConsoleScreenBufferInfo(hConsole, &csbi);
    FillConsoleOutputAttribute(hConsole, csbi.wAttributes, dwConSize, coordScreen,
&cCharsWritten);
    SetConsoleCursorPosition(hConsole, coordScreen);
}
```

```
int executeQueryForCreateAndDrop(sqlite3* db, const string& query) {
    char* errorMessage = nullptr;
    int result = sqlite3_exec(db, query.c_str(), nullptr, nullptr, &errorMessage);
    if (result != SQLITE_OK) {
        cerr << "Error executing query: " << errorMessage << endl;
        sqlite3_free(errorMessage);
    }
    return result;
}



int executeQuery(sqlite3* db, const string& query, vector<vector<string>>& results)
{
    char* errorMessage = nullptr;
    int result = sqlite3_exec(db, query.c_str(), [](void* data, int argc, char**
argv, char** columnNames) {
        vector<string> row;
        for (int i = 0; i < argc; i++)
            row.push_back(argv[i] ? argv[i] : "");
        vector<vector<string>>& results =
*static_cast<vector<vector<string>>*>(data);
        results.push_back(row);
        return 0;
    }, &results, &errorMessage);

    if (result != SQLITE_OK) {
        cerr << "Error executing query: " << errorMessage << endl;
        sqlite3_free(errorMessage);
    }
    return result;
}


void readDataFromDatabase(sqlite3* db, vector<Client>& clients,
                          vector<Employee>& employeers, vector<Order>& orders,
string& login, string& password) {

    // Зчитування логіна та пароля

    string selectLPQuery="SELECT login, password FROM LogPass;";
    vector<vector<string>> LogPassResults;

    if(executeQuery(db, selectLPQuery, LogPassResults)== SQLITE_OK){
    login=LogPassResults[0][0];
    password=LogPassResults[0][1];
    }
    // Зчитування таблиці Clients
    string selectClientsQuery = "SELECT name, surname, phone FROM Clients;";
    vector<vector<string>> clientsResults;
```

```
    executeQuery(db, selectClientsQuery, clientsResults);
    for (const auto& row : clientsResults) {
        Client client;
        client.name = row[0];
        client.surname = row[1];
        client.phone = row[2];
        clients.push_back(client);
    }

        // Зчитування таблиці Auto
    string selectAutoQuery = "SELECT client_id, marka, model, color, znak FROM
Auto;";
    vector<vector<string>> autoResults;
    executeQuery(db, selectAutoQuery, autoResults);
    for (const auto& row : autoResults) {
        Auto aut;
        int c=stoi(row[0]);
        aut.marka = row[1];
        aut.model = row[2];
        aut.color = row[3];
        aut.znak = row[4];
        clients[c].automobile.push_back(aut);
    }

    // Зчитування таблиці Employees
    string selectEmployeesQuery = "SELECT name, surname, specialization, salary
FROM Employees;";
    vector<vector<string>> employeesResults;
    executeQuery(db, selectEmployeesQuery, employeesResults);
    for (const auto& row : employeesResults) {
        Employee employee;
        employee.name = row[0];
        employee.surname = row[1];
        employee.specialization = row[2];
        employee.salary = stod(row[3]);
        employeers.push_back(employee);
    }



// Зчитування таблиці Orders

    string selectOrdersQuery =
        "SELECT auto_id, employee_id,client_id, price, type_of_repair,
short_problem, execution_date_day, execution_date_month, execution_date_year,
complete "
        "FROM Orders;";
    vector<vector<string>> ordersResults;
    executeQuery(db, selectOrdersQuery, ordersResults);
```

```
    for (const auto& row : ordersResults) {
        int a=stoi(row[0]);
        int e=stoi(row[1]);
        int c=stoi(row[2]);

        Order order;
        order.automobile = clients[c].automobile[a];
        order.employee = employeers[e];
        order.client = clients[c];
        order.price = stod(row[3]);
        order.type_of_repair = row[4];
        order.short_problem = row[5];
        order.execution_date.day = stoi(row[6]);
        order.execution_date.month = stoi(row[7]);
        order.execution_date.year = stoi(row[8]);
        order.complete = (row[9] == "1");
        orders.push_back(order);
    }
    }

void writeDataToDatabase(sqlite3* db, const vector<Client>& clients,
                         const vector<Employee>& employeers, const vector<Order>&
orders, string login, string password) {
    // Видалення таблиць, якщо вони існують
    executeQueryForCreateAndDrop(db, "DROP TABLE IF EXISTS Auto;");
    executeQueryForCreateAndDrop(db, "DROP TABLE IF EXISTS LogPass;");
    executeQueryForCreateAndDrop(db, "DROP TABLE IF EXISTS Clients;");
    executeQueryForCreateAndDrop(db, "DROP TABLE IF EXISTS Employees;");
    executeQueryForCreateAndDrop(db, "DROP TABLE IF EXISTS Orders;");

    // Створення таблиць Auto, Clients, Employees, Orders
    executeQueryForCreateAndDrop(db, "CREATE TABLE Auto (client_id INT, marka TEXT,
model TEXT, color TEXT, znak TEXT);");
    executeQueryForCreateAndDrop(db, "CREATE TABLE Clients (name TEXT, surname
TEXT, phone TEXT);");
    executeQueryForCreateAndDrop(db, "CREATE TABLE LogPass (login TEXT, password
TEXT);");
    executeQueryForCreateAndDrop(db, "CREATE TABLE IF NOT EXISTS Employees (name
TEXT, surname TEXT, specialization TEXT, salary REAL);"
        "CREATE TABLE Orders (auto_id INTEGER, employee_id INTEGER, client_id
INTEGER, price REAL, "
        "type_of_repair TEXT, short_problem TEXT, execution_date_day INTEGER,
execution_date_month INTEGER, execution_date_year INTEGER, complete INTEGER);");

    //Запис логіну та пароля
    string LogPassQuery="INSERT INTO LogPass (login, password) VALUES( "+ login +
", " + password + ");";
    executeQueryForCreateAndDrop(db, LogPassQuery);

    // Запис даних в таблицю Auto
```

```
    for(int c=0;c<clients.size();c++){
    for (const auto& aut : clients[c].automobile) {
        string query = "INSERT INTO Auto (client_id ,marka, model, color, znak)
VALUES ( "+ to_string(c) +",'" + aut.marka + "', '" +
                       aut.model + "', '" + aut.color + "', '" + aut.znak + "');";
        executeQueryForCreateAndDrop(db, query);
    }}

    // Запис даних в таблицю Clients
    for (const auto& client : clients) {
        string query = "INSERT INTO Clients (name, surname, phone) VALUES ('" +
client.name + "', '" + client.surname +
                       "', '" + client.phone + "');";
        executeQueryForCreateAndDrop(db, query);
    }

    // Запис даних в таблицю Employees
    for (const auto& employee : employeers) {
        string query = "INSERT INTO Employees (name, surname, specialization,
salary) VALUES ('" + employee.name +
                       "', '" + employee.surname + "', '" + employee.specialization
+ "', " +
                       to_string(employee.salary) + ");";
        executeQueryForCreateAndDrop(db, query);
    }

    // Запис даних в таблицю Orders
    for (const auto& order : orders) {
        // Отримання ідентифікаторів об'єктів з інших таблиць
        int autoId = executeQueryForCreateAndDrop(db, "SELECT rowid FROM Auto WHERE
marka='" + order.automobile.marka +
                                                  "' AND model='" + order.automobile.model
+ "' AND color='" +
                                                  order.automobile.color + "' AND znak='" +
order.automobile.znak + "';");

        int employeeId = executeQueryForCreateAndDrop(db, "SELECT rowid FROM
Employees WHERE name='" + order.employee.name +
                                                    "' AND surname='" +
order.employee.surname + "' AND specialization='" +
                                                    order.employee.specialization + "'
AND salary=" +
                                                    to_string(order.employee.salary) +
";");

        int clientId = executeQueryForCreateAndDrop(db, "SELECT rowid FROM Clients
WHERE name='" + order.client.name +
                                                "' AND surname='" +
order.client.surname + "' AND phone='" +
                                                order.client.phone + "';");
```

```
        string query =
            "INSERT INTO Orders (auto_id, employee_id, client_id, price,
type_of_repair, short_problem, "
            "execution_date_day, execution_date_month, execution_date_year,
complete) VALUES ("+ to_string(autoId) +
            ", " + to_string(employeeId) + ", " + to_string(clientId) + ", " +
            to_string(order.price) + ", '" + order.type_of_repair + "', '" +
order.short_problem + "', '" +
            to_string(order.execution_date.day) + "', '" +
to_string(order.execution_date.month) + "', '" +
to_string(order.execution_date.year) + "', " + (order.complete ? "1" : "0") + ");";
        executeQueryForCreateAndDrop(db, query);
    }
}
```

Meneger.h

```
#ifndef MANAGER_H
#define MANAGER_H
#include <iostream>
#include <string.h>
#include "functions.h"
#include "Auto.h"
#include "Employee.h"
#include "Client.h"
#include "Order.h"
#include "sqlite3.h"

using namespace std;

class Manager
{
    public:
        Manager();
        virtual ~Manager();
        void autorithation();
        bool main();
        void addOrder();
        void checkO();
        void changePass();
        int client_chose();
        int employee_chose();
        int auto_chose(int c);
        Auto addAuto();
        void addClient();
        void addEmployee();
        void changeOrder();
        void showOrders(bool type);
        void add_del_emp();
        void add_del_client();
```

```
        void changeEmployee();
        void changeClient();

    protected:

    private:
            string login="123";
            string password="123";
            Auto auto_for_changes=Auto{};
            Client client_for_changes=Client{};
            Employee employee_for_changes=Employee{};
            Order order_for_changes=Order{};
            vector<Order> orders;
            vector<Employee> employeers;
            vector<Client> clients;
};

#endif // MANAGER_H
```

Auto.h

```
#ifndef AUTO_H
#define AUTO_H
#include <string>
using namespace std;

class Auto
{
public:
    string marka;
    string model;
    string color;
    string znak;

    Auto(string mar="Null", string mod="Null", string c="Null", string z="Null");
};

#endif // AUTO_H
```

Employee.h

```
#ifndef EMPLOYEE_H
#define EMPLOYEE_H
#include <string>
using namespace std;

class Employee
{
public:
    string name;
    string surname;
```

```
    string specialization;
    double salary;
    Employee(string n="Null", string sur="Null", string s="Null", double sal=0);
};

#endif // EMPLOYEE_H
```

Client.h

```
#ifndef CLIENT_H
#define CLIENT_H
#include <string>
#include <vector>
#include "Auto.h"
using namespace std;
class Client
{
public:
    string name;
    string surname;
    string phone;
    vector<Auto> automobile;

    Client( string n="Null", string s="Null", string p="0");
};

#endif // CLIENT_H
```

Order.h

```
#ifndef ORDER_H
#define ORDER_H
#include <string>
#include "Auto.h"
#include "Employee.h"
#include "Client.h"
using namespace std;



struct Date{
    int day;
    int month;
    int year;
};

class Order
{
public:
    Auto automobile;
    Employee employee;
    Client client;
    double price;
```

```
    string type_of_repair;
    string short_problem;
    Date execution_date;
    bool complete=false;

    Order(Auto A=Auto{}, Employee E=Employee{}, Client C=Client{}, double p=0,
string type="", string problem="");
    void change_price();
    void change_SP();
    void change_Emp(Employee emp);
    void change_complete();


};

#endif // ORDER_H
```