РЕФЕРАТ

Пояснювальна записка: ___ с., ___ рис., ___ табл., __ дод., ___ джерел.

Об'єкт розробки: формування маршрутного листа на автотранспортному підприємстві з пошуком найкоротшого шляху.

Мета кваліфікаційної роботи: розробка інформаційної системи, що забезпечуватиме формування маршрутного листа на автотранспортному підприємстві з можливістю прокладання оптимального маршруту до місця слідування водія.

У вступі розглядається аналіз та сучасний стан проблеми, конкретизується мета кваліфікаційної роботи та галузь її застосування, наведено обґрунтування актуальності теми та уточнюється постановка завдання.

У першому розділі проведено аналіз предметної галузі, визначено актуальність завдання та призначення розробки, розроблена постановка завдання, задані вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі виконано аналіз існуючих рішень, обрано вибір платформи для розробки, виконано проектування і розробка програми, наведено опис алгоритму і структури функціонування програми, визначені вхідні і вихідні дані, наведені характеристики складу параметрів технічних засобів, описаний виклик та завантаження програми, описана робота програми.

В економічному розділі визначено трудомісткість розробленої інформаційної системи, проведений підрахунок вартості роботи по створенню програми та розраховано час на його створення.

Практичне значення роботи полягає в тому, що ефективна організація обліку за допомогою нових технологій в автотранспорті сприяє оптимальному використанню ресурсів, мінімізує витрати виробництва. Організація облікової справи може впливати на розвиток і прибутковість підприємства настільки ж сильно, як і будь-яке нововведення в технічному обладнанні підприємств.

Актуальність інформаційної системи визначається великим попитом на подібні розробки, що оптимізують та спрощують дії щодо ведення бази даних системи, підвищення ефективності діяльності підприємства за рахунок скорочення затрачених ресурсів.

Список ключових слів: ІНФОРМАЦІЙНА СИСТЕМА, АЛГОРИТМ ДЕЙКСТРИ, АВТОПЕРЕВІЗНИК, МАРШРУТ, ПРОЕКТУВАННЯ, МЕНЮ, ВКЛАДКА, ГРАФ.

ABSTRACT

Explanatory note: ___ pp., ___ fig., ___ table, __ appendix, ___ sources.

Object of development: formation of the route list at the motor transport enterprise with search of the shortest way.

The purpose of the qualification work: development of an information system that will ensure the formation of the route list at the trucking company with the possibility of laying the optimal route to the driver's place of departure.

The introduction considers the analysis and the current state of the problem, specifies the purpose of the qualification work and the scope of its application, provides a justification for the relevance of the topic and clarifies the problem.

In the first section the analysis of the subject area is carried out, the urgency of the task and purpose of development are defined, the statement of the task is developed, requirements to software realization, technologies and software are set.

The second section analyzes existing solutions, selects a platform for development, performs design and development of the program, describes the algorithm and structure of the program, determines the input and output data, provides characteristics of the parameters of hardware, describes the call and download of the program, describes the program .

The economic section determines the complexity of the developed information system, calculates the cost of work to create a program and calculates the time for its creation.

The practical significance of the work is that the effective organization of accounting with the help of new technologies in motor transport promotes the optimal use of resources, minimizes production costs. The organization of accounting can affect the development and profitability of the enterprise as much as any innovation in the technical equipment of enterprises.

The relevance of the information system is determined by the high demand for such developments that optimize and simplify actions to maintain a database of the system, increase the efficiency of the enterprise by reducing resources.

List of keywords: INFORMATION SYSTEM, DICKSTER ALGORITHM, CARRIER, ROUTE, DESIGN, MENU, TAB, GRAPH.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ІС – інформаційна система;

ОС – операційна система;

ПЗ – програмне забезпечення;

ПК – персональний комп’ютер.

ЗМІСТ

ВСТУП

Маршрутний лист - це документ, за яким обліковують роботу локомотивів, автотранспорту, витрати палива, електроенергії, матеріалів, робочого часу (нормативні й фактичні дані), використовують при нарахуванні заробітної плати локомотивним бригадам, водіям та ін.

В маршрутний лист на підприємствах автотранспорту вносяться наступні дані: дані водія, марка автомобіля, підстава, місце прибуття, відмітка про прибуття, підтверджуючий документ та рух пального. Раніше всі ці пункти заповнювалися в рукописному вигляді, що займало багато часу і перетворювалося на рутинну роботу.

З появою персональних комп'ютерів на підприємствах процес оформлення маршрутного листа став значно легшим, а саме це досягається шляхом створення комп'ютерних програм, які дозволять заповнювати ці пункти за декілька хвилин та формувати автоматично маршрутний лист.

Створення комп'ютеризованої системи обліку автотранспорту потребує глибокого дослідження теоретичних основ і організаційних особливостей обліку в умовах його комп'ютеризації.

З впровадженням сучасної обчислювальної техніки можливості обліку в автотранспорті значно розширюються. Застосування комп'ютерів змінює зміст та організацію праці облікового персоналу: зменшується кількість рутинних операцій з обробки первинних документів, систематизації облікових показників, заповнення регістрів та звітних форм.

Таким чином ефективна організація обліку за допомогою нових технологій в автотранспорті сприяє оптимальному використанню ресурсів, мінімізує витрати виробництва. Організація облікової справи може впливати на розвиток і прибутковість підприємства настільки ж сильно, як і будь-яке нововведення в технічному обладнанні підприємств.

Оптимізація - процес надання будь-чому найвигідніших характеристик, співвідношень (наприклад, оптимізація виробничих процесів і виробництва). Задача оптимізації сформульована, якщо заданий критерій оптимальності.

Методи оптимізації поділяються на прямі та ітераційні. Оптимізація полягає в знаходженні найкращого варіанта. Методи оптимізації застосовуються до пошуку розрахунку оптимальної технології, оптимальної геометричної конструкції, найкращого часу та шляху для технологічних процесів і подібних задач.

Метою даної кваліфікаційної роботи є розробка інформаційної системи, що забезпечуватиме формування маршрутного листа на автотранспортному підприємстві з можливістю прокладання оптимального маршруту до місця слідування водія.

Програмний додаток може бути застосований на невеликих автотранспортних підприємствах в якості допоміжного інструмента у роботі бухгалтера та диспетчера.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1. Загальні відомості з предметної галузі

1.1.1. Специфічна термінологія предметної галузі

Задачею даної кваліфікаційної роботи є розробка інформаційної системи, що забезпечуватиме формування маршрутного листа на автотранспортному підприємстві з можливістю прокладання оптимального маршруту до місця слідування водія.

В основі вирішення оптимізаційних задач є використання теорії графів.

Граф – сукупність об’єктів (вершин) і зв’язків між ними (ребер для неорієнтованого графу та дул для орієнтованого).

Вершини (або вузли) – об`єкти з визначення графу.

Ребра (дуги) – відрізки між вершинами. Ребро – неорієнтований відрізок (аналог дороги з двостороннім рухом), дуга – орієнтований відрізок (аналог дороги, де дозволено рух тільки в один бік).

Степінь вершини – кількість ребер, інцидентних вершині.

Зважений граф – граф, де перехід по ребру коштує певну суму – вагу ребра.

Незважений граф – граф, де ваги всіх ребер однакові. Як правило, неорієнтований. Найчастіше описується матрицею суміжності. Якщо між вершинами є ребро, відповідний елемент матриці – одиниця, інакше він є нулем.

Орієнтований граф – граф, де вершини сполучені тільки дугами.

Неорієнтований граф – граф, де вершини сполучено тільки ребрами.

Джерело – вершина з нульовим степенем заходу.

Стік – вершина з нульовим степенем виходу.

Залишкова пропускна здатність – пропускна здатність ребра (дуги) після того, як через нього пройшов потік

Залишкова мережа – множина ребер з додатньою залишковою пропускною здатністю

Блокуючий потік – це такий потік, без якого шляху з джерела до стоку не існує.

1.1.2. Способи завдання графів

Задати граф означає задати множини його вершин і ребер, а також відношення інцидентності. Коли граф G – скінченний, для опису його вершин та ребер досить їх занумерувати.

Нехай v_1, v_2, …, v_n – вершини графа G; e_1, e_2, …, e_m – його ребра. Відношення інцидентності можна означити матрицею $E=\|\varepsilon_{ij}\|$, яка має n рядків та m стовпців. Рядки відповідають вершинам графа, а стовпці – його ребрам. Якщо ребро e_j є інцидентним вершині v_i, то $\varepsilon_{ij}=1$, в іншому випадку $\varepsilon_{ij}=0$. Це матриця інцидентності звичайного графа G, яка є одним із способів його визначення. Наприклад, для графа, який зображено на рис. 1.2,а матриця інцидентності має вигляд таблаблиці рис. 1.3,а.

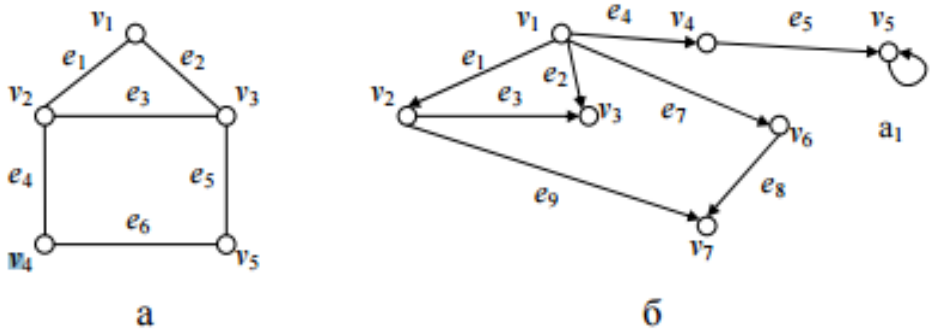


Рис. 1.2. Приклади графів

У матриці інцидентності $\|\varepsilon_{ij}\|$ орієнтованого графа D, якщо вершина v_i – початок дуги e_j, то $\varepsilon_{ij}=-1$, якщо v_i – кінець e_j, то $\varepsilon_{ij}=1$; якщо e_j – петля, а v_i – інцидентна їй вершина, то $\varepsilon_{ij}=2$. Наприклад, для орієнтованого графа, зображеного на рис. 1.2,б, матриця інцидентності наведена у таблиці рис. 1.3,б.

	e_1	e_2	e_3	e_4	e_5	e_6
v_1	1	1	0	0	0	0
v_2	1	0	1	1	0	0
v_3	0	1	1	0	1	0
v_4	0	0	0	1	0	1
v_5	0	0	0	0	1	1

а

	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9
v_1	-1	-1	0	-1	0	0	-1	0	0
v_2	1	0	-1	0	0	0	0	0	-1
v_3	0	1	1	0	0	0	0	0	0
v_4	0	0	0	1	-1	0	0	0	0
v_5	0	0	0	0	1	2	0	0	0
v_6	0	0	0	0	0	0	1	-1	0
v_7	0	0	0	0	0	0	0	1	1

б

Рис. 1.3. Матриці інцидентності

У кожному рядку матриці інцидентності для неорієнтованого або орієнтованого графа тільки два елементи відмінні від 0 (або один, якщо ребро є петлею). Тому такий спосіб завдання графа – не досить економний. Відношення інцидентності можна задати ще списком ребер графа. Кожний рядок цього списку відповідає ребру, в ньому записано номери вершин, інцидентних йому. Для неорієнтованого графа порядок цих вершин у рядку довільний, для орієнтованого першим записується номер або інше найменування початку ребра, а другим – його кінця. У таблиці на рис. 1.4,а та б наведено списки ребер відповідно для графів з рис. 1.1, а та б.

За списком ребер графа можна легко визначити матрицю інцидентності. Справді, кожний рядок цього списку відповідає стовпцю матриці з тим самим номером. Для неорієнтованого графа в рядку списку записуються номери елементів стовпця матриці інцидентності, що дорівнюють 1, а для орієнтованого графа в цьому рядку першим зазначається номер елемента стовпця матриці, який дорівнює -1, другим – номер елемента, що дорівнює 1.

Поняття матриці інцидентності та списку ребер можна легко узагальнити на випадок мультиграфа.

Ребро	Вершини
e_1	v_1, v_2
e_2	v_1, v_3
e_3	v_2, v_3
e_4	v_2, v_4
e_5	v_3, v_5
e_6	v_4, v_5

а

Ребро	Вершини
e_1	v_1, v_2
e_2	v_1, v_3
e_3	v_2, v_3
e_4	v_1, v_4
e_5	v_4, v_5
e_6	v_5, v_5
e_7	v_1, v_6
e_8	v_6, v_7
e_9	v_2, v_7

Б

Рис. 1.4. Списки ребер відповідно для графів з рис. 1.1, а та б.

Розглянемо третій спосіб завдання графів – матриця суміжності – це квадратна матриця $\Delta \| \delta_{ij} \|$, стовпцям і рядкам якої відповідають вершини графа. Для неорієнтованого графа δ_{ij} дорівнює кількості ребер, інцидентних і- та j-й вершинам, для орієнтованого – цей елемент матриці відповідає кількості ребер з початком в і-й вершині й кінцем у j-й вершині. Таким чином, матриця суміжності неорієнтованого графа є симетричною ($\delta_{ij} = \delta_{ji}$), а орієнтованого – необов'язково. Якщо вона все ж симетрична, то для кожного ребра орієнтованого графа існує ребро, яке з'єднує ті самі вершини, але йде у зворотному напрямку. Очевидно, орієнтований граф із симетричною матрицею суміжності відповідає неорієнтованому графу, який має ту саму матрицю суміжності.

Матриці суміжності розглянутих вище графів (див. рис. 1.4) наведено в таблицях на рис. 1.5.

	v_1	v_2	v_3	v_4	v_5
v_1	0	1	1	0	0
v_2	1	0	1	1	0
v_3	1	1	0	0	1
v_4	0	1	0	0	1
v_5	0	0	1	1	0

а

	v_1	v_2	v_3	v_4	v_5	v_6	v_7
v_1	0	1	1	1	0	1	0
v_2	0	0	1	0	0	0	1
v_3	0	0	0	0	0	0	0
v_4	0	0	0	0	1	0	0
v_5	0	0	0	0	1	0	0
v_6	0	0	0	0	0	0	1
v_7	0	0	0	0	0	0	0

б

Рис. 1.5. Матриці суміжності

Між простими (без кратних ребер) орієнтованими графами та бінарними відношеннями існує взаємно однозначне співставлення. Довільний граф з множиною вершин $V=\{v_1,\ldots,v_n\}$ визначає бінарне відношення на множині V – відношення суміжності. Матриця суміжності цього графа – це матриця бінарного відношення суміжності. Справедливо й обернене – довільне бінарне відношення R на довільній множині $A=\{a_1,\ldots,a_n\}$ можна зобразити графом G, вершини якого відповідають елементам A, а ребро (a_i, a_j) в цьому графі існує, тоді й тільки тоді, коли виконується a_iRa_j. Бінарна матриця відношення R одночасно є матрицею суміжності графа G, а сам граф називається графом відношення R.

За матрицею суміжності графа можна визначити властивості відношення R. Граф рефлексивного відношення містить петлі у всіх вершинах і, відповідно, одиниці на всіх елементах головної діагоналі матриці суміжності. Симетричному відношенню відповідає граф із симетричною матрицею суміжності. Як було зазначено вище, такий граф рівнозначний простому неорієнтованому графу. Граф транзитивного відношення має наступні властивості: якщо є ребра (v_i, v_j) і (v_j, v_k), то існує ребро (v_i, v_k).

Оскільки довільний граф представляє певне відношення, можна визначити операції об’єднання та перетину над графами так само, як і над відношеннями. Доповненню R’ відношення R (тобто відношенню, яке істинно, коли R хибне) відповідає доповнення графа G до повного графа, тобто граф G’, в якому є ті й тільки ті дуги, яких немає в G. Оберненому відношенню R^{-1} відповідає граф G^{-1}, який отримується з графа G зміною орієнтації всіх його дуг на протилежні.

Комбінаторика − розділ математики, присвячений вирішенню задач вибору та розташування елементів деякої, зазвичай скінченної множини відповідно до заданих правил.

1.2. Призначення розробки та галузь застосування

Даний програмний додаток призначений для формування маршрутного листа для автотранспортного підприємства з можливістю прокладання найкоротшого шляху до місця слідування водія.

Система створюється для обслуговування таких груп користувачів: адміністратор; бухгалтер; водій.

Ефективна організація обліку за допомогою нових технологій в автотранспорті сприяє оптимальному використанню ресурсів, мінімізує витрати виробництва. Організація облікової справи може впливати на розвиток і прибутковість підприємства настільки ж сильно, як і будь-яке нововведення в технічному обладнанні підприємств.

На даний час є великий попит на подібні розробки, що оптимізують та спрощують дії щодо ведення бази даних системи, підвищують ефективність діяльності підприємства за рахунок скорочення затрачених ресурсів.

Програмний додаток може бути застосований на невеликих автотранспортних підприємствах в якості допоміжного інструмента у роботі бухгалтера, адміністратора, диспетчера та водія.

1.3. Підстава для розробки

Підставою для розробки кваліфікаційної роботи бакалавра на тему «Розробка інформаційної системи для формування маршрутного листа на автотранспортному підприємстві з можливістю прокладання оптимального маршруту в середовищі MS Visual Studio 2019» є наказ по Національному технічному університету «Дніпровська політехніка» від __.__. 2021р. № ____-_.

1.4. Постановка завдання

Метою даної кваліфікаційної роботи є розробка інформаційної системи, що забезпечуватиме формування маршрутного листа на автотранспортному підприємстві з можливістю прокладання оптимального маршруту до місця слідування водія.

Для досягнення мети роботи необхідно спроектувати та розробити програму, яка на основі застосованого алгоритму пошуку оптимального шляху формуватиме маршрутний лист для автотранспортного підприємства. Дана система дозволить повністю автоматизувати обробку документа.

У якості вхідних даних виступають введені у користувачами дані щодо водія (прізвище, ім'я, по-батькові, посада, автомобіль, підстава проїздки) та інформація про маршрут (дата прибуття, місце прибуття, час прибуття та вибуття, підтверджуючий документ, витрати), а також дані для авторизації користувача у системі (логін та пароль).

Вихідними даними програми є прорахований оптимальний маршрут поїздки, сформований та готовий до друку маршрутний лист, який можна буде експортувати у зовнішній файл форматів Word або Excel.

Під час виконання завдання необхідно виконати наступні етапи:

1. Ознайомитися з предметною галуззю.
2. Дослідити та проаналізувати тему «Моделювання графів» та «Пошук оптимального шляху між двома точками».
3. Структурувати матеріал з даної теми.
4. Розробити структуру інформаційної системи.
5. Розробити алгоритм роботи програми.
6. Розробити базу даних проєкту.
7. Розробити алгоритм розрахунку для знаходження оптимального шляху.
8. Визначити мову програмування і середовище розробки.
9. Розробити програму для використання на ПК із зручним інтерфейсом.

Розроблена інформаційна система має бути протестована на конкретних прикладах та з різноманітними даними, що засвідчуватиме про її працеспроможність.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Розроблений програмний додаток призначений для формування маршрутного листа і повинен мати трирівневий доступ, а саме: адміністратор, водій та бухгалтер. В залежності від рівня доступу програма повинна надавати наступні функціональні можливості:

1. Адміністратор повинен мати доступ до всіх функцій програми, а саме редагування бази даних та формування маршрутного листа.
2. Бухгалтер повинен мати доступ до обмеженої кількості функцій, а саме введення даних для формування маршрутного листа та формування маршрутного листа.
3. Водій повинен мати можливість тільки переглядати інформацію у маршрутному листі та переглядати на мапі прокладений найкоротший шлях до адреси місцезнаходження його кінцевої точки слідування.

Інтерфейс програми повинен бути багатовіконним, графічним і повинен складатися з наступних вікон:

– вікно входу, яке буде з’являтися при запуску програми і використовується для авторизації користувача;

– вікно з мапою міста для формування найкоротшого маршруту слідування для водія;

– вікно для формування маршрутного листа;

– вікно для перегляду та редагування інформації в інформаційній системі;

– вікно «Про програму», в якому подано короткий опис створеної програми;

– довідкове вікно, в якому описані вказівки по роботі з програмою. Всі вікна програми повинні мати елементи керування, контекстне меню, рядок стану, набір гарячих клавіш.

1.5.2. Вимоги до інформаційної безпеки

Надійність роботи інформаційної системи залежить від надійності операційної системи, під управлінням якої вона буде функціонувати, а також надійності розроблюваного програмного забезпечення.

Для надійної роботи програмного забезпечення зі сторони операційної системи необхідно дотримуватися таких факторів:

- використання ліцензійного ПЗ;
- захист від зловмисних програм;
- архівація даних на сервері;
- встановлення блоків безперебійного живлення.

Для надійності роботи програмного забезпечення зі сторони розробленого додатку: перевірка введених даних користувачем для виключення можливих зловмисних ситуацій.

1.5.3. Вимоги до складу та параметрів технічних засобів

Для користування створеним додатком потрібен комп’ютер чи ноутбук з встановленою операційною системою Windows та текстовим редактором MS Word або редактор електронних таблиць MS Excel. Мінімальними характеристиками для роботи програмного додатку є Windows xp, яка має наступні мінімальні характеристики:

– процесор з частотою 233 МГц;
– не менше 64 МБ оперативної пам’яті;
– не менше 1,5 ГБ вільного місця на жорсткому диску.

Так як такі характеристики в наш час мають тільки старі комп’ютери, то для більш коректної роботи програми краще використовувати Windows 7 або Windows 10 які мають наступні мінімальні характеристики:

– процесор з частотою 1 ГГц;

– не меше 1 Гбайт для 32 бітної системи або 2 Гбайт для 64 битної системи;

– об’єм жорсткого диску не менше 16 ГБ для 32 бітної системи або 20 Гб для 64 битної системи;

– відеокарта з підтримкою Microsoft DirectX 9.

1.5.4. Вимоги до інформаційної та програмної сумісності

Дана інформаційна система повинна бути розроблена з використанням наступних програмно-апаратних засобів:

– операційна система Microsoft Windows XP/Vista/7;

– за допомогою середовища програмування .NET C# (WPF + Windows Forms).

Для повноцінної роботи розробленого програмного додатку необхідно мати встановлений на комп’ютері текстовий редактор MS Word або електронний редактор таблиць MS Excel так як сформований маршрутний лист зберігається в файлах редактора.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Функціональне призначення системи

В даній роботі необхідно спроектувати та розробити програму, яка на основі застосованого алгоритму пошуку оптимального шляху формуватиме маршрутний лист для автотранспортного підприємства. Дана система дозволить повністю автоматизувати обробку документа.

Розроблений програмний додаток призначений для формування маршрутного листа і повинен мати трирівневий доступ, а саме: адміністратор, водій та бухгалтер. В залежності від рівня доступу програма надає наступні функціональні можливості:

1. Адміністратор має доступ до всіх функцій програми, а саме редагування бази даних та формування маршрутного листа.
2. Бухгалтер має доступ до обмеженої кількості функцій, а саме введення даних для формування маршрутного листа та формування маршрутного листа.
3. Водій має можливість тільки переглядати інформацію у маршрутному листі та переглядати на мапі прокладений найкоротший шлях до адреси місцезнаходження його кінцевої точки слідування.

2.2. Опис застосованих математичних методів

Теорія графів – розділ математики, що вивчає властивості графів. Наочно граф можна уявити, як геометричну конфігурацію, яка складається з точок (вершини), сполучених лініями (ребрами). У строгому визначенні графом називається така пара множин $G = (V, E)$, де V є підмножина будь-якої зліченної множини, а E – підмножина $V \times V$.

Визначення графу є настільки загальним, що цим терміном можна описувати безліч подій та об'єктів повсякденного життя. Високий рівень абстракції та узагальнення дозволяє використовувати типові алгоритми теорії графів для вирішення зовнішньо несхожих задач у транспортних і комп’ютерних мережах, будівельному проектуванні, молекулярному моделюванні тощо [4].

Алгоритм Дейкстри – алгоритм на графах, відкритий Дейкстрою. Знаходить найкоротший шлях від однієї вершини графа до всіх інших вершин.

Алгоритм Дейкстри, на відміну від алгоритму Флойда з допомогою якого можна відшукати найкоротший маршрут між будь-якими двома вершинами графа, призначений для пошуку маршруту найкоротшої довжини від заданої вершини графа до всіх інших. В процесі виконання даного алгоритму при переході від вершини *i* до вершини *j* використовується спеціальна процедура, яка кожній вершині графа присвоює відповідну мітку [4].

Алгоритм Дейкстри дозволяє знайти найкоротшу довжину шляху і визначити маршрт від стартової вершини А до кожної з інших вершин.

Мітки в алгоритмі Дейкстри можуть бути двох типів: тимчасові або постійні. Тимчасова мітка може бути замінена на нову також тимчасову, якщо в процесі виконання алгоритму буде знайдено більш короткий маршрут до даної вершини. Якщо ж в процесі виконання алгоритму виявиться, що більш короткого маршруту від початкової до даної вершини не існує, то змінюємо статус тимчасової мітки на постійну.

Приклад роботи алгоритму Дейкстри зображено на рис. 2.1.

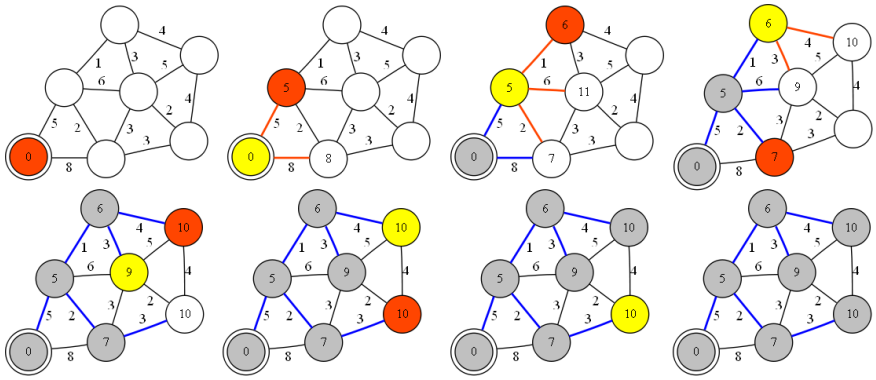


Рис. 2.1. Алгоритм Дейкстри

Розв'язок за алгоритмом Дейкстри складається з наступних етапів:

– початковій вершині (вершина під номером 1) присвоюємо постійну мітку $[0, -]$ і покладаємо $i = 1$;

– для всіх вершин *j*, які поєднані орієнтованим ребром з вершиною *i*, і не мають міток, статус яких становить «постійна», обчислюємо тимчасові мітки $[u_i + d_{ij}, i]$. Якщо для вершини *j* вже існує деяка мітка $[u_j, k]$, то слід провірити виконання умови $u_i + d_{ij} < u_j$. Виконання даної умови говорить про те, що необхідно мітку $[u_j, k]$замінити на $[u_i + d_{ij}, i]$.

Даний процес продовжуємо до тих пір, поки всім вершинам графа не буде присвоєно мітку зі статусом – постійна [3].

2.3. Опис використаних технологій та мов програмування

Для розробки інформаційної системи для автоматизації формування маршрутного листа було використано середовище розробки Microsoft Visual Studio 2019, мову програмування C#, а для реалізації бази даних було обрано систему керування базою даних СКБД MS SQL Server.

Додаток написаний за допомогою методів об’єктно-орієнтованого програмування на мові C# і скомпільований з розширенням exe. Він працює у середовищі операційної системи Windows та Unix.

Windows являє собою ціле сімейство операційних систем, розроблених

компанією Microsoft. Вона виросла з графічної надбудови над системою MS-DOS в самостійну, що стала згодом однією з найпопулярніших і затребуваних по всьому світу.

Вона має ряд особливостей:

– зручний для користувача графічний інтерфейс. Він дає змогу досить просто керувати роботою комп'ютера, використовуючи такі поняття, як «Мій комп'ютер», «Мережне оточення», «Кнопка Пуск», «Панель задач», «Контекстне меню», «Вікно», «Ярлик», технології «Вкажи і вибери» (Point and Click), «Перенеси і відпусти» (Drag and Drop) і т. д.;

– вона є об'єктно-орієнтованою ОС для оброблення документів. В ній використовують такі офісні аналогії, як «робочий стіл», «папка», «документ», «кошик» і т. д.;

– допускає підключення до локальних і глобальних комп'ютерних мереж (електронна пошта, факс, Internet);

– містить довгі імена файлів (до 255 символів);

– допускає роботу з програмами, розробленими в інших ОС (Windows xp, Windows Vista, Windows 7), і виконання їх;

– працює в багатозадачному режимі. Використовує процесну форму (паралельно виконується кілька програм) та потокову форму (паралельно виконуються різні частини однієї програми);

– застосовує пряму адресацію оперативної пам'яті, завдяки чому у програмах (додатках) можуть використовуватися до 4 Гбайт віртуальної пам'яті (оперативна пам'ять і пам'ять на жорсткому диску);

– підтримує обмін даними між додатками за допомогою OLE-технології (Object Linking and Embedding - зв'язування та вбудовування об'єктів). Наприклад, таблиці, а також діаграми, побудовані в табличному процесорі Excel, можуть використовуватися в документі, створеному в текстовому редакторі Word;

– містить ряд стандартних програм (Блокнот, Графічний редактор Paint, Калькулятор).

Linux - загальна назва unix-подібніх операційних систем на основі однойменного ядра і зібраних для нього бібліотек і системних програм.

Основна відмінність Linux полягає в можливості адаптувати цю ОС під абсолютно різні завдання. Вихідний код в Linux можна модифікувати, а також використовувати за своїм бажанням. При необхідності в будь-який момент можна виправити помилки системи, розширити її функціональність.

Основні особливості Linux:

– linux часто використовують як базову операційну систему для серверів. Операційна система Linux - багатокористувацька система, всі користувачі, поділяються на 2 групи: звичайні користувачі та адміністратори;

– для операційної системи Linux вірусів майже не існує. Сама побудова операційної системи виключає роботу шкідливих програм. А відтак можна обійтися без антивірусних програм, що гальмують комп'ютер і заважають працювати. Не потрібно весь час оновлювати антивірусні бази і перевіряти жорсткий диск на віруси, втрачаючи дорогоцінний час;

– всі необхідні драйвери знаходяться на диску з дистрибутивом операційної системи. Тому немає необхідності інсталювати драйвери на невизначені обладнання. Після інсталяції Linux на комп'ютер, він буде повністю працездатний. У випадку коли якийсь пристрій буде не встановлено, то це легко виправляється шляхом пошуку його в інтернеті;

– з будь-якого дистрибутива Linux можна зробити як серверну операційну систему, так і звичайну робочу станцію. В операційній системі Linux, можна легко встановити і налаштувати як web сервер, так і сервер електронної пошти[7].

Для повноцінної роботи розробленого програмного додатку необхідно мати встановлений на комп’ютері текстовий редактор MS Word або електронний редактор таблиць MS Excel так як сформований маршрутний лист зберігається в файлах редактора.

Microsoft Word - потужний інтелектуальний текстовий процесор. Це обумовлено насамперед його численними перевагами, до яких належать широкі

функціональні можливості. Важко знайти таке завдання під час роботи з текстами, в процесі створення професійно оформлених документів, яке не можна було б виконати засобами Microsoft Word, які до того ж є зручними та простими у використанні.

До традиційних засобів форматування процесора Word належать: шрифтове оформлення документів, управління вирівнюванням текстів на смузі набору, управління режимами обтікання вбудованих об’єктів.

У питаннях форматування особливої уваги заслуговують унікальні можливості останніх версій Word у роботі з таблицями.

Існує кілька версій Word для Windows, кожна наступна версія сумісна, як правило, з попередніми версіями і має додаткові можливості[8].

MS Excel - засіб для роботи з електронними таблицями, який набагато перевищує за своїми можливостями існуючі редактори таблиць.

Ключові переваги MS Excel:

– ефективний аналіз і обробка даних;

– багаті засоби форматування та відображення даних; - наочний друк;

– спільне використання даних і робота над документами;

– обмін даними та інформацією через Internet і внутрішні Intranet-мережі.

MS Excel надає:

– швидкий і ефективний аналіз, зручні засоби для роботи з даними (майстер зведених таблиць дозволяє швидко обробляти великі масиви даних і одержувати підсумкові результати в зручному вигляді);

– механізм автокорекції формул автоматично розпізнає і виправляє помилки при введенні формул. Microsoft Excel уміє розпізнавати 15 найбільш поширених помилок, які допускаються користувачами при введенні формул в комірку. Наприклад, автоматично виправляються помилки, пов'язані з неправильними посиланнями, отриманими в результаті переміщення комірок. Введений помилково символ "х" автоматично перетворюється на знак множення і т.д. Природно, при цьому Excel спочатку запитує користувача, чи

потрібно проводити виправлення;

– використання природної мови при написанні формул;

– проведення різних обчислень з використанням потужного апарату функцій і формул;

– дослідження впливу різних факторів на дані;

– рішення задач оптимізації;

– отримання вибірки даних, які відповідають певним критеріям;

– побудова графіків і діаграм;

– статистичний аналіз даних. У Excel для Windows є настільки потужний апарат математичної статистики, що можна займатися статистичним моделюванням[9].

Visual Studio представляє собою повністю інтегровану середу розробки. Вона спроектована таким чином, щоб робити процес написання коду, його налагодження та компіляції в збірку для поставки кінцевим споживачам якомога простішим. На практиці це означає, що Visual Studio є дуже складним додатком з багатодокументним інтерфейсом, в якому можна робити практично все, що стосується розробки коду.

За допомогою цього редактора можна створювати тексти програм на мові C#. Текстовий редактор має досить потужні можливості. Наприклад, при введенні тексту програми він автоматично компонує його на сторінці, створюючи між рядками необхідні відступи, вирівнюючи відкриває та закриває фігурні дужки блоків коду і виділяє ключові слова кольором. Крім того, по мірі введення коду він виконує його перевірку на предмет синтаксичних помилок і підкреслює фрагменти, які будуть викликати помилки при компіляції, що також називається налагодженням на стадії проектування. У редакторі реалізовано засіб IntelliSense, який забезпечує автоматичне відображення імен класів, полів або методів при початку їх введення, а також списки параметрів, які підтримують всі доступні перевантажені версії методів при початку введення параметрів для методів.

Також редактор дозволяє розміщувати бажані елементи управління

призначеного для користувацького інтерфейсу і доступу до даних в проекті, а Visual Studio потім автоматично додає в вихідні файли код на мові C#, який необхідний для створення екземплярів цих елементів в проекті.

В Visual Studio замість того щоб виконувати компіляцію проекту, запускаючи компілятор C# з командного рядка, можна вибрати відповідний пункт меню в середовищі розробки. Visual Studio самостійно викликає компілятор і передає йому всі необхідні параметри командного рядка, які вказують, на які збірки повинен посилатися код і який вид повинна мати збірка на виході (наприклад, виконуваний файл або бібліотека * .dll). При бажанні Visual Studio може також автоматично запускати скомпільований виконуваний файл на виконання, дозволяючи перевірити його роботу.

Visual Studio має можливість компіляції прямо в середовищі розробки. Замість того щоб виконувати компіляцію проекту, запускаючи компілятор C# з командного рядка, можна вибрати відповідний пункт меню в середовищі розробки. Visual Studio самостійно викликає компілятор і передає йому всі необхідні параметри командного рядка, які вказують, на які повинен посилатися код і який вид повинна мати збірка на виході. При бажанні Visual Studio може також автоматично запускати скомпільований виконуваний файл на виконання, дозволяючи перевірити його роботу.

Visual Studio надає доступ до цілого ряду інших утиліт, які дозволяють переглядати і змінювати різні аспекти комп'ютера або мережі, не покидаючи середовища розробки. Завдяки цим інструментам, можна переглядати як виконуються служби та активні сполуки з базами даних, заглядати в таблиці на сервері SQL Server.

Visual Studio дозволяє отримувати доступ до документації MSDN прямо з середовища IDE. У разі, наприклад, виникнення сумнівів з приводу призначення того чи іншого ключового слова під час роботи з текстовим редактором, можна виділити це ключове слово і натиснути клавішу F1, в результаті чого Visual Studio автоматично підключиться до MSDN і відобразить відповідні розділи довідки. Аналогічно, якщо пот-рібно подивитися, що означає

та чи інша помилка компіляції, потрібно виділяти повідомлення з помилкою і натиснути F1.

Також Visual Studio містить графічні редактори і конструктори XML, забезпечує підтримку розробки програм Windows, орієнтованих на мобільні пристрої, підтримку розробки програм Microsoft Office і Windows Workflow Foundation, містить вбудовану підтримку рефакторинга коду і інструменти візуального конструювання класів[1].

Для розробки програмного додатку було обрано мову C# яка дозволяє працювати з візуальними засобами Visual Studio 2019.

C# є об'єктно-орієнтованою мовою, але підтримує також і компонентно-орієнтоване програмування. Розробка сучасних додатків все більше тяжіє до створення програмних компонентів у формі автономних і самоописних пакетів, що реалізують окремі функціональні можливості.

Важлива особливість таких компонентів - це модель програмування на основі властивостей, методів і подій. Кожен компонент має атрибути, які надають декларативні відомості про компоненти, а також вбудовані елементи документації. C# надає мовні конструкції, які безпосередньо підтримують таку концепцію роботи.

У C# існує єдина система типів. Всі типи C#, включаючи типи-примітиви, такі як int і double, успадковують від одного кореневого типу object. Таким чином, всі типи використовують загальний набір операцій, і значення будь-якого типу можна зберігати, передавати і обробляти схожим чином. Крім того, C# підтримує призначені для користувача посилальні типи і типи значень, дозволяючи як динамічно виділяти пам'ять для об'єктів, так і зберігати спрощені структури в стеці. Завдяки цьому C# відмінно підходить для створення і застосування програмних компонентів.

Мова C# і пов'язану з ним середу .NET Framework можна без перебільшення назвати найзначнішою з пропонованих в даний час технологій для розробників. Середа .NET є таким середовищем, яка була створена для того, щоб в ній можна було розробляти практично будь-який додаток для

запуску в Windows, а C# є мовою програмування, яка була спеціально створена для використання в .NET Framework. Наприклад, із застосуванням C# і .NET Framework можна створювати динамічні веб-сторінки, додатки Windows Presentation Foundation, веб-служби XML, компоненти для розподілених додатків, компоненти для доступу до баз даних, класичні настільні додатки Windows і навіть клієнтські програми нового інтелектуального типу, володіють можливостями для роботи в оперативному і автономному режимах.

Призначення .NET Framework - служити середовищем для підтримки розробки та виконання сильно розподілених компонентних додатків. Він забезпечує спільне використання різних мов програмування, а також безпеку, переносимість програм і загальну модель програмування для платформи Windows.

Розглянемо програмні методи по роботі з MS Excel та MS Word.

Для роботи з Excel необхідно додати посилання на бібліотеку Microsoft Excel Object Library.

Сам сервер - об'єкт Application або додаток Excel, може містити одну або більше книг, посилання на які містить властивість Workbooks. Книги – об'єкти Workbook, можуть містити одну або більше сторінок, поси-лання на які містить властивість Worksheets Сторінки – Worksheet, міс-тить об'єкти осередки або групи осередків, посилання на які стають до-ступними через об'єкт Range.

Властивість SheetsInNewWorkbook повертає або встановлює кількість листів, автоматично розміщених в Excel на новій робочій книзі.

Властивість StartupPath повертає шлях до папки, яка містить надбудови, що виконуються під час запуску.

Книги можуть бути не тільки додані, але і закриті.

Метод excelapp.Workbooks.Close() дозволяє закривати всі або конкретні робочі книги.

Документи Excel можна зберегти програмно і звичайним для Excel способом. У будь-якому випадку перед виходом з Excel необхідно викликати метод Quit. Якщо властивість Excel.Application DisplayAlerts має значення true,

Excel запропонує зберегти незбережені дані, якщо після старту в документ були внесені якісь зміни.

Для збереження документів можна використовувати методи Excel.Workbook Save і SaveAs. Метод Save зберігає робочу книгу в папці "Мої документи" з іменами, що привласнюються документу за замовчуванням ("Книга1.xls", "Книга2.xls" ...) або в директорію і з ім'ям, під яким документ вже був збережений.

Метод SaveAs дозволяє зберегти документ із зазначенням імені, формату файлу.

Для доступу до книги використовуються значення AccessMode xlShared – спільна робоча книга, xlExclusive – монопольний доступ або xlNoChange - заборона зміни режиму доступу.

Параметр ConflictResolution – спосіб вирішення конфліктів при одночасному внесенні декількома користувачами змін в один документ – може мати значення: xlUserResolution – відображення діалогового вікна вирішення конфліктів, xlLocalSessionChanges - прийняття змін, внесених користувачем або xlOtherSessionChanges – прийняття змін, внесених іншими користувачами.

Також для збереження документа може бути використаний метод SaveCopyAs, який зберігає копію робочої книги у файлі.

Для відкриття існуючого документа основним методом є метод Open набору Excel.Workbooks. Для відкриття текстових файлів як робочих книг, баз даних, файлів у форматі .XML, використовуються методи OpenText, OpenDatabase або OpenXml.

Для виділення використовується метод get_Range, який дозволяє виділити групу комірок через завдання кутових комірок діапазону. Ще один спосіб визначення обраних осередків - використання методу get_Offset об'єкта Range, що повертає об'єкт Range, віддалений від заданої комірки на задану кількість рядків і стовпців, рахуючи від лівого верхнього кута. Виділені комірки далі можуть бути об'єднані і з ними дії можуть виконуватися як з одною коміркою. Для цього використовується метод Merge.

Для читання інформації з комірки Excel на обраному листі необхідно в обраній книзі вибрати одну клітинку або об'єднану групу комірок - після чого досить перетворити значення в виділених комірках до потрібного типу даних[3].

Всі об'єкти Word мають ієрархічну структуру. Об'єкти Document можуть містити такі об'єкти як Paragraph, Table, Range, Bookmark, Chapter, Word, Sentence, Sections, Headers та ін.

Створення об'єкта Word.Application аналогічно тому, як це робиться в Excel, закриття ж сервера World вимагає використання ряду параметрів з ключовим словом ref, які, як і для багатьох інших методів в Word, повинні бути записані в змінну до передачі в метод. У кожному разі потрібно створювати змінну типу Object, привласнювати їй значення і використовувати цю змінну з ключовим словом ref.

Властивості Content об'єкта Document визначає основну область документа. Тобто його вміст без колонтитулів та інших допоміжних елементів.

Для створення документу використовується Метод Add(). Документи Word можна зберегти програмно і звичайним для Word способом. У будь-якому випадку перед виходом з Word необхідно викликати метод Quit. Якщо властивість Word.Application DisplayAlerts має значення true, Word запропонує зберегти дані в тому випадку, коли після старту в документ були внесені якісь зміни.

Для збереження документа можна використовувати методи Save() і SaveAs(). Метод Save() не має параметрів і при його виклику буде відображено діалогове вікно Word "Збереження документа". Навпаки, метод SaveAs має безліч параметрів, більшість з яких можна не вказувати, а використовувати як параметри за замовчуванням.

Для відкриття існуючого документа основними методом є метод Open().

Для роботи з параграфами використовується метод Word.Paragraph. Виведення тексту виконується не просто в параграф, а в діапазон параграфа - об'єкт Range. Об'єкт Range - це безперервна область документа, що включає

позицію початкового і кінцевого символів. У документі можна визначити діапазон, викликом методу Range з передачею йому початкового і кінцевого значень позицій символів.

Крім об'єкта Range документа кожен параграф також має властивість Range, яка дозволяє виводити текст і встановлювати його характеристики для параграфа.

Всі дії, пов'язані з внесенням в документ будь-яких приватних змін в Word виконуються відносно до виділених фрагментів. Програмно це виконується через об'єкт Selection. При програмному виведення необхідно спочатку виділити фрагмент і, далі, виконати необхідні дії над даними фрагментом.

Об'єкт Selection може представляти блок, рядок або стовпець таблиці, курсор введення, малюнок, виділений текст, або деяку комбінацію об'єктів, які можна визначити через властивість Type об'єкта Selection. Оскільки властивості і методи у різних типів об'єктів Selection різні, то при застосуванні того чи іншого методу або використання властивості рекомендується виконати перевірку типу об'єкта Selection[4].

Для збереження даних маршрутного листа буде використовуватися база даних. Для реалізації бази даних буде використовуватися СКБД SQL Server.

MS SQL Server – це платформа для вирішення критично важливих завдань в масштабі підприємства, що володіє високою доступністю, підвищеною продуктивністю і безпекою. Рішення являє собою добре маштабуючий, повністю реляційний, швидкодіючий сервер, здатний обробляти великі обсяги даних для клієнт-серверних додатків. Рекордна продуктивність MS SQL Server забезпечується новими технологіями роботи з пам'яттю, що допомагає підприємствам прискорити свій бізнес і реалізувати нові сценарії роботи. Крім того, SQL Server дозволяє використовувати нові гібридні хмарні рішення і користуватися новими перевагами хмарних обчислень.

Системи бази даних надають логічну незалежність файлів, тобто, іншими словами, логічну структуру бази даних можна змінювати без необхідності

внесення будь-яких змін в прикладні програми бази даних. Наприклад, додавання атрибута до вже існуючої в системі баз даних структурі об'єкта з ім'ям Person (наприклад, адреса) викликає необхідність модифікувати тільки логічну структуру бази даних, а не існуючі прикладні програми. (однак додатки зажадають модифікування для використання нового стовпчика.).

SQL Server надає можливість створення користувацького інтерфейсу. Більшість баз даних проектуються і реалізовуються для роботи з ними різних типів користувачів, що мають різні рівні знань. З цієї причини система баз даних повинна надавати кілька окремих користувацьких інтерфейсів. Інтерфейс може бути графічним або текстовим.

У графічних інтерфейсах введення здійснюється за допомогою клавіатури або миші, а виведення реалізується в графічному вигляді на монітор. Різновидом текстового інтерфейсу, часто використовуваного в системах баз даних, є інтерфейс командного рядка, за допомогою якого користувач здійснює введення за допомогою набору команд на клавіатурі, а система відображає виведення в текстовому форматі на моніторі.

Фізична незалежність даних SQL Server означає, що прикладні програми бази даних не залежать від фізичної структури даних, що зберігаються в базі даних. Ця важлива особливість дозволяє змінювати збережені дані без необхідності вносити будь-які зміни в прикладні програми баз даних.

Головну роль у створенні додатків відіграє клас Dataset, що дозволяє створювати локальний набір взаємопов'язаних таблиць. Також необхідно відзначити, що за допомогою ADO.NET можна звертатися до різних джерел даних. Роботи з різними джерелами даних реалізована за допомогою керованих провайдерів (managed provider). Керований провайдер - набір класів, які реалізують інтерфейси IDbConnection, IDbCommand, IDataReader, IDbDataAdapter і призначений для роботи з певним джерелом даних. Для роботи з Microsoft SQL Server використовуються провайдери SQL, оптимізовані для роботи саме з цими СУБД. Крім того, до складу ADO.NET включений провайдер OLEDb, що дозволяє працювати з будь-якими джерелами даних, які

підтримують протокол OLE DB.

При створенні з'єднання з Microsoft SQL Server необхідно вказати значення для наступних параметрів: server - ім'я комп'ютера (або його IP адреса), на якому встановлено Microsoft SQL Server; uid - обліковий запис користувача Microsoft SQL Server, від імені якого створюється підключення; pwd - пароль облікового запису користувача Microsoft SQL Server, від імені якого створюється підключення; database - ім'я бази даних, з якої необхідно створити з'єднання. Рядок з'єднання може бути вказаний при натисненні конструктора об'єкта класу SqlConnection, або через присвоєння значення властивості ConnectionString інтерфейсу ID-bConnection.

Для відкриття з'єднання з базою даних використовується метод void Open() інтерфейсу IDbConnection. Для закриття з'єднання з базою даних використовується метод void Close() інтерфейсу IDbConnection[5].

2.4. Опис структури системи та алгоритмів її функціонування

2.4.1. Опис структури додатку

User-case – це діаграма, яка описує сценарій взаємодії учасників. User-case діаграма автотранспортного підприємства описує взаємодію трьох ролей (адміністратор, бухгалтер, водій) в предметній області.

Розглянемо функціонал програмного додатку за допомогою User-case діаграми (рис.2.2).

Рис. 2.2. User-case діаграма автотранспортного підприємства

Кожен користувач при вході в систему повинен пройти авторизацію. На рис. 2.3 наведено блок-схему вікна авторизації.

Після проходження авторизації користувачу надається роль:

- Адміністратор - має повний набір можливостей по роботі з даними в базі даних (додавання, редагування, видалення). Також йому надається можливість видавати роль користувачам та формувати маршрутний лист;
- Бухгалтер - може формувати та переглядати маршрутний лист;
- Водій - має можливість перегляду маршрутного листа та мапи з маршрутом слідування до місця призначення.

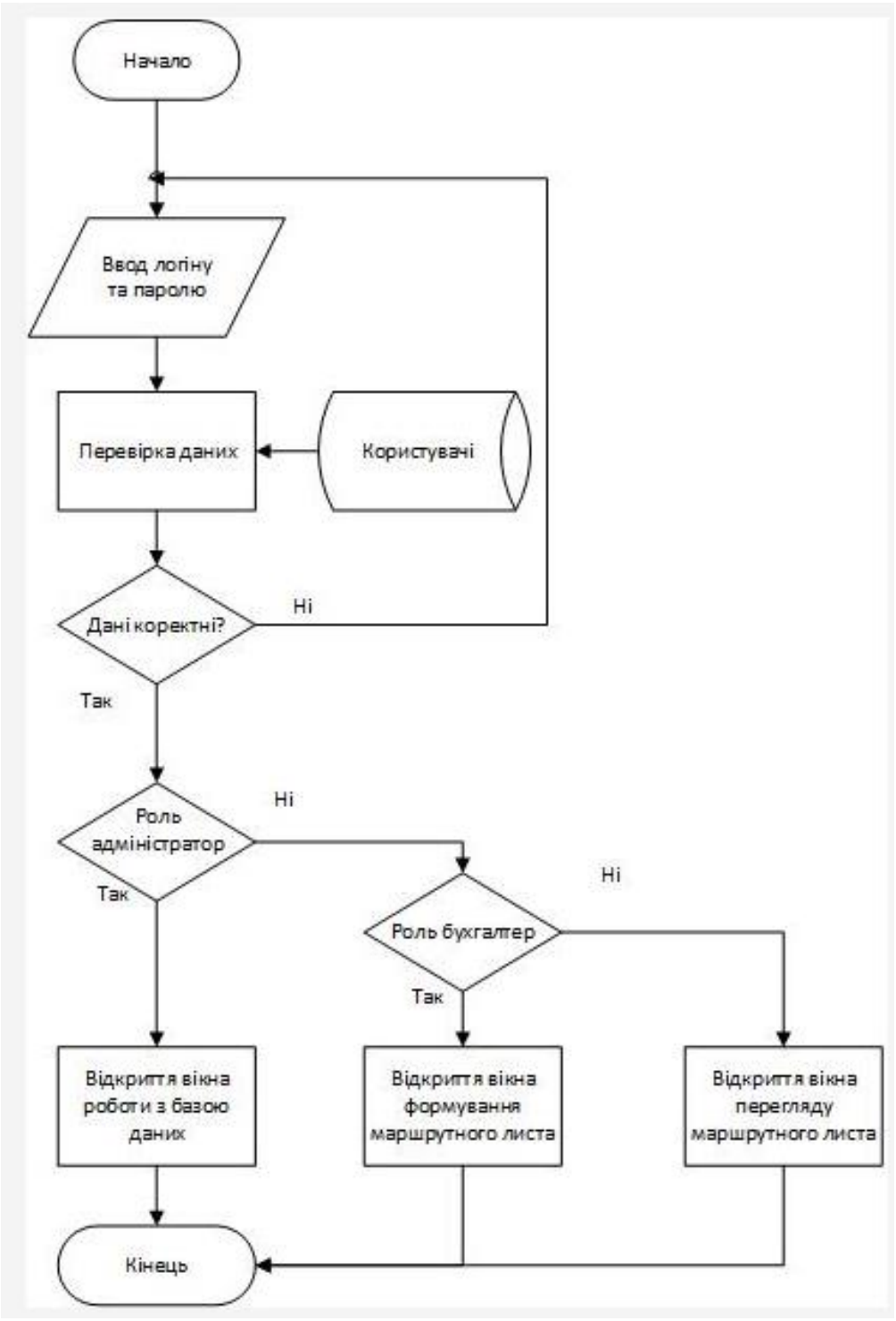


Рис. 2.3. Блок-схема вікна авторизації

Діаграма послідовності - це діаграма яка відображає взаємодії об’єктів впорядкованих за часом[11]. Діаграма послідовності автотранспортного підприємства покроково описує взаємодію кожної ролі (адміністратор, бухгалтер, водій) в предметній області.

Функціонал програмного додатку можна розглянути за допомогою діаграми послідовності Адміністратора, Бухгалтера та Водія (рис. 2.4 – 2.6).

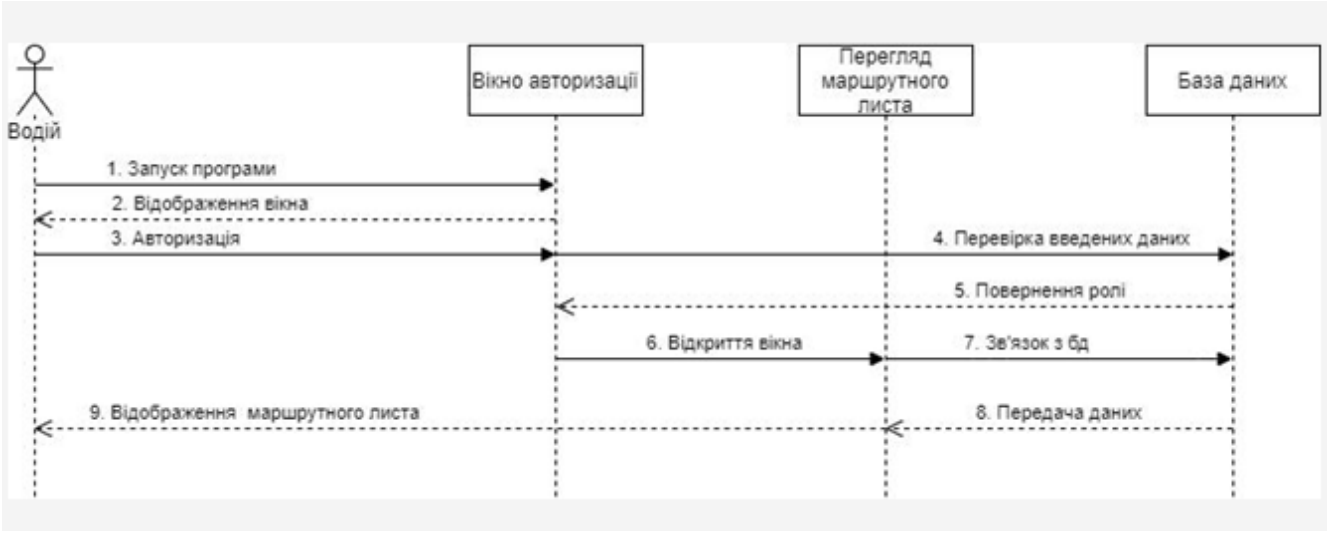


Рис. 2.4. Діаграма послідовності водія

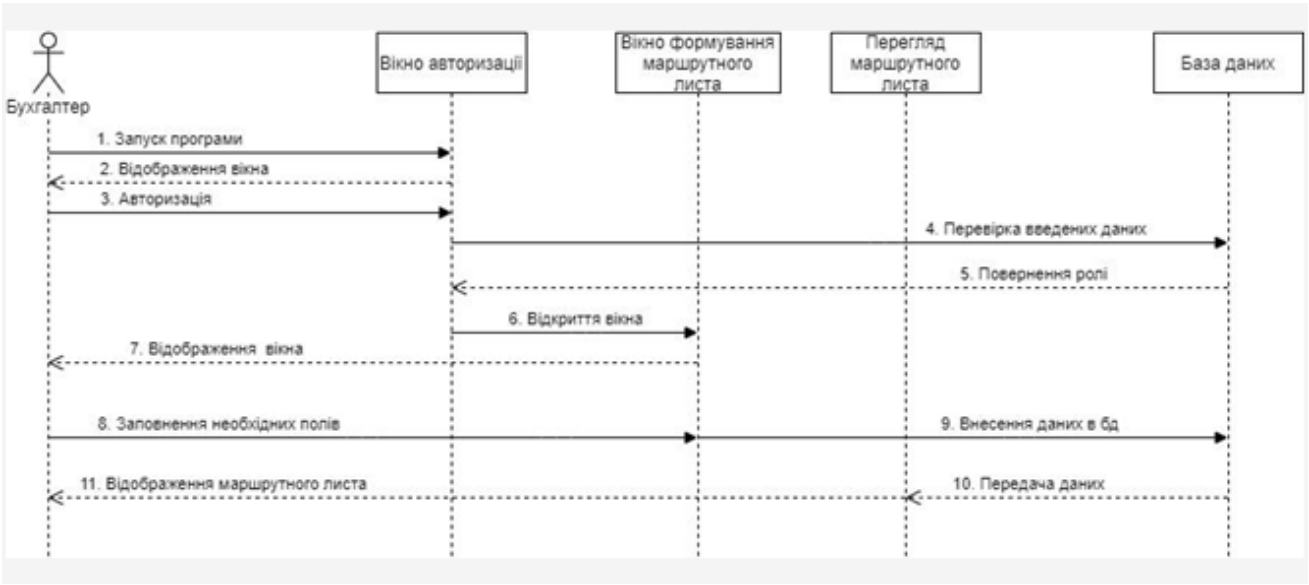


Рис. 2.5. Діаграма послідовності бухгалтера

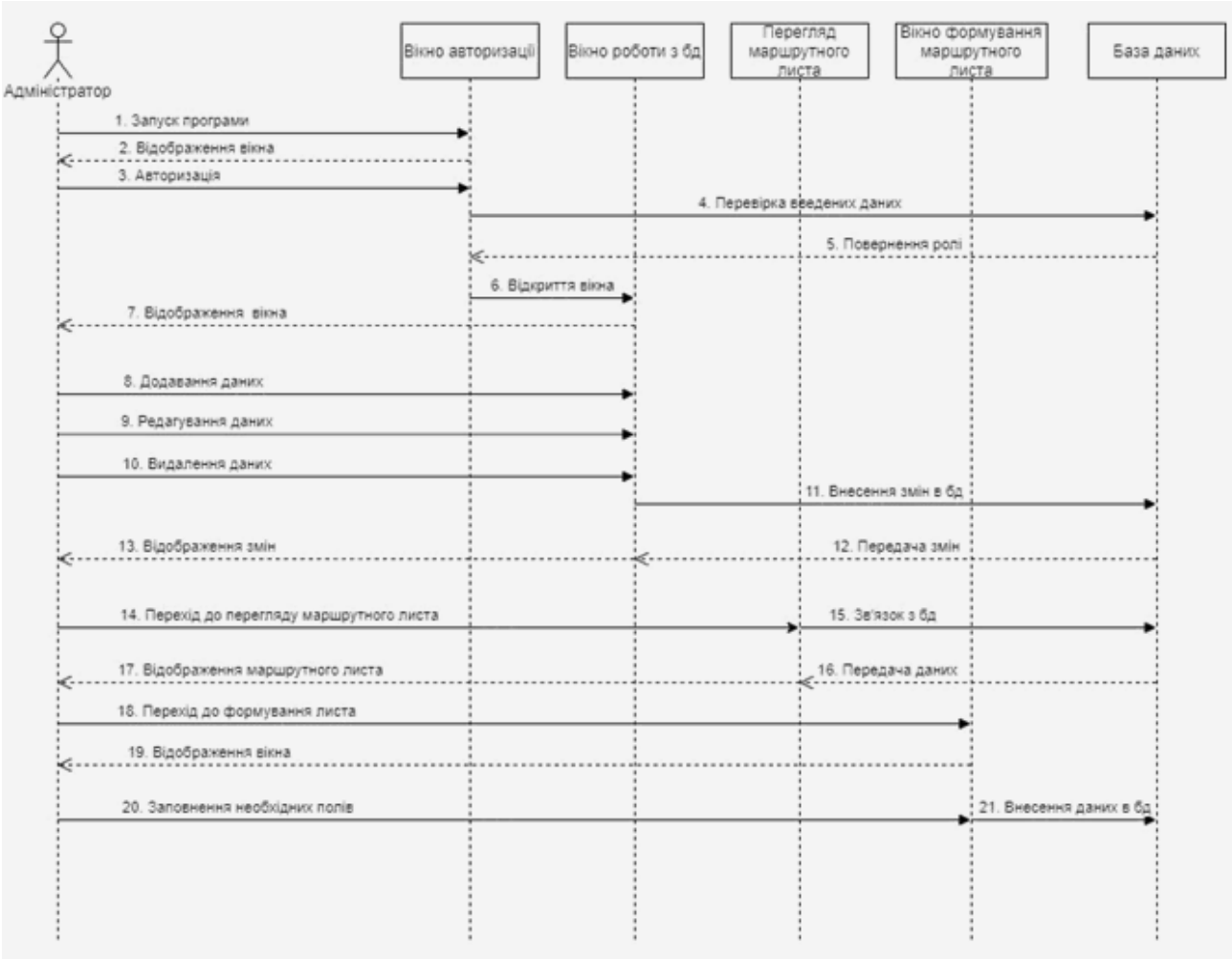


Рис. 2.6. Діаграма послідовності адміністратора

Маршрутний лист формується на основі даних про водія, автомобіль, підстави поїздки, дати, місця призначення, маршруту та обрахованих витрат. На рис. 2.7. наведено блок-схему вікна формування маршрутного листа.

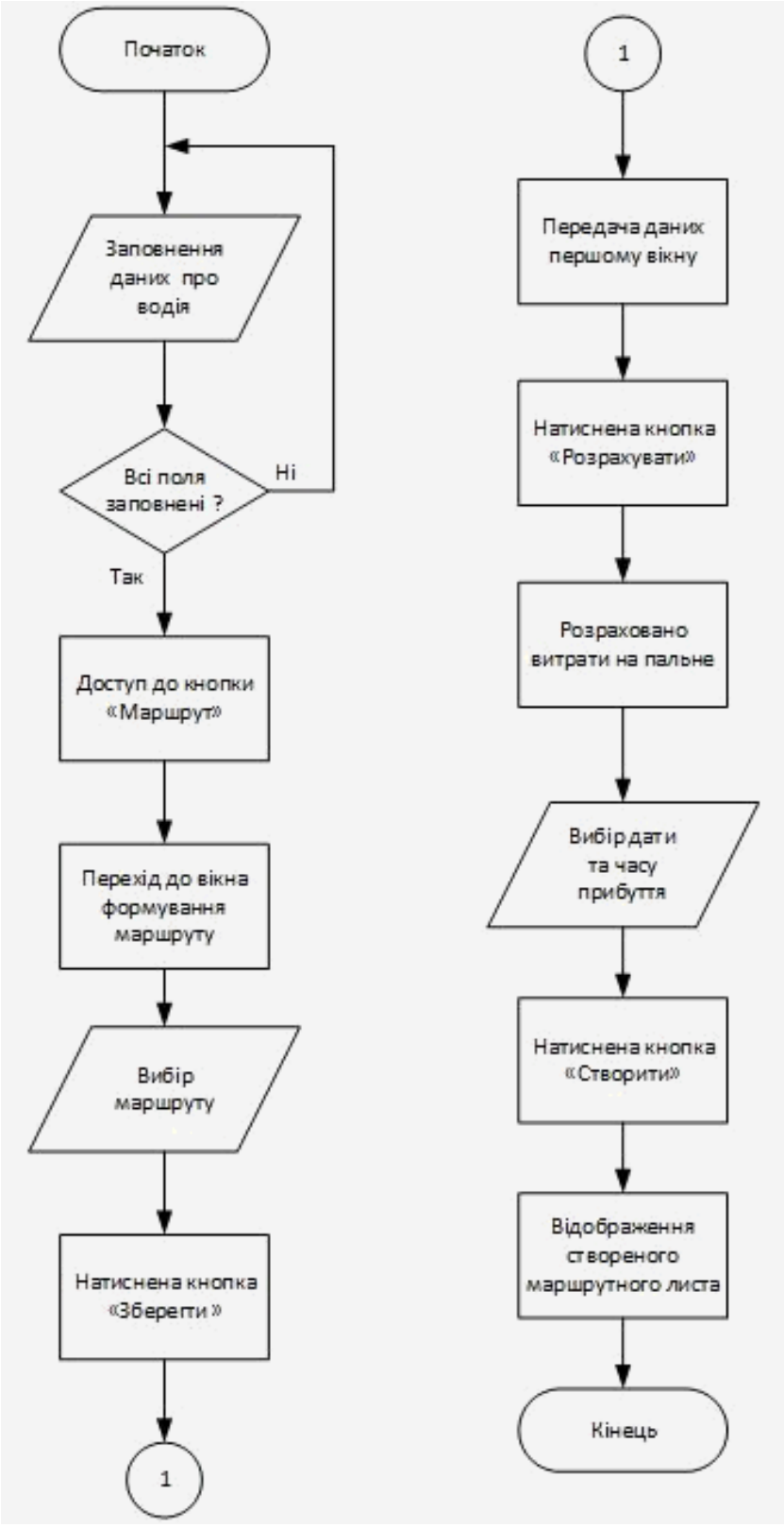


Рис. 2.7. Блок-схема вікна формування маршрутного листа

2.4.2. Реалізація алгоритму Дейкстри

Для формування найкоротшого маршруту слідування водія до місця призначений у програмному додатку використовується алгоритм Дейкстри.

Алгоритм Дейкстри дозволяє виконувати пошук найкоротших шляхів з однієї вершини до всіх інших вершин у графі. Даний алгоритм знаходить найкоротші шляхи до вершин графа в порядку їх віддаленості від вихідної вершини, тобто спочатку знаходиться найкоротший шлях від вихідної вершини до найближчої, потім до другої найближчої тощо. Таким чином, перед початком і-ої ітерації алгоритм визначає найкоротші шляхи до (і – 1)- ої вершин, найближчих до вихідної. Дані вершини, вихідна вершина та ребра найкоротших шляхів утворюють піддерево даного графа. Чергова найближча до вихідної вершина може бути знайдена серед вершин, суміжних з отриманим піддеревом: для кожної суміжної вершини обчислюється сума відстаней до найближчої вершини дерева, після чого обирається вершина з найменшою сумою.

Розглянемо роботу алгоритму візуально (рис. 2.8 - 2.14).

Нехай необхідно знайти найкоротший шлях від першої вершини до всіх інших.

Ініціалізація.

Мітка самої вершини 1 покладається рівною 0, мітки інших вершин - недосяжно велике число (в ідеалі - нескінченність). Це відображає те, що відстані від вершини 1 до інших вершин поки невідомі. Всі вершини графа позначаються як невідвідані.

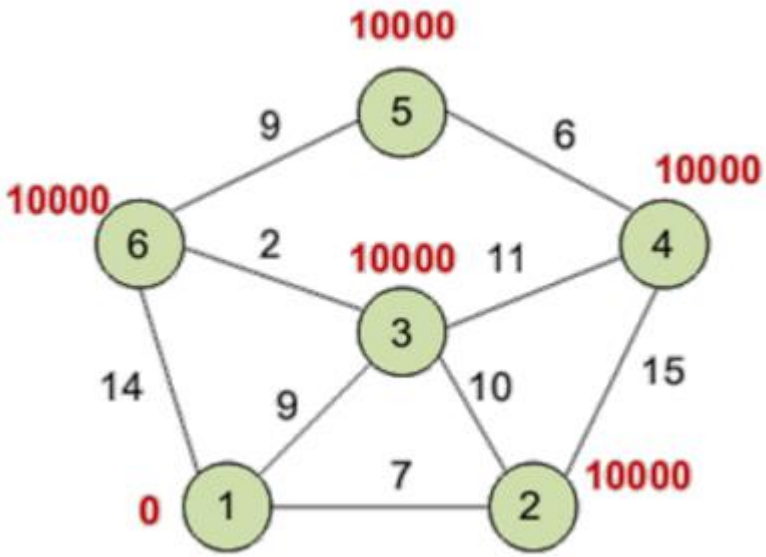


Рис. 2.8. Ініціалізація графів

Мінімальну мітку має вершина 1. Її сусідами є вершини 2, 3 і 6. Обходимо сусідів вершини по черзі.

Перший сусід вершини 1 – вершина 2, тому що довжина шляху до неї мінімальна. Довжина шляху в неї через вершину 1 дорівнює сумі найкоротшої відстані до вершини 1 (значенням її мітки) і довжини ребра, що йде з 1-ї у 2-у, тобто 0 + 7 = 7. Це менше поточної мітки вершини 2 (10000) , тому нова мітка 2-й вершини дорівнює 7.

Аналогічно знаходимо довжини шляху для всіх інших сусідів (вершини 3 і 6).

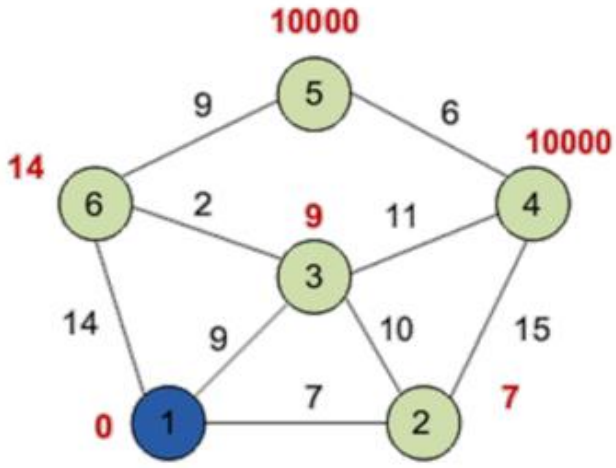


Рис. 2.9. Крок перший

Перший крок алгоритму повторюється. Знову знаходимо «найближчу» з невідвіданих вершин. Це вершина 2 з міткою 7.Знову намагаємося зменшити мітки сусідів обраної вершини, намагаючись пройти в них через 2 вершину. Сусідами вершини 2 є вершини 1, 3 і 4.

Вершина 1 вже переглянута. Наступний сусід вершини 2 - вершина 3, так як має мінімальну позначку з вершин, позначених як не відвідані.

Ще один сусід вершини 2 - вершина 4. Якщо йти в неї через 2, то довжина такого шляху буде дорівнює 22

Всі сусіди вершини 2 переглянуті, помічаємо її як відвідану.

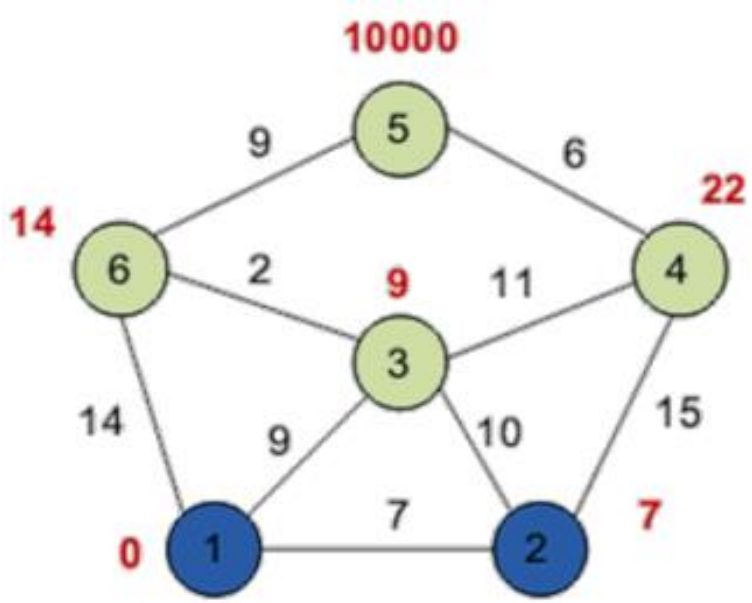


Рис. 2.10. Крок другий

Третій крок

Повторюємо крок алгоритму, вибравши вершину 3. Після її «обробки» отримаємо наступні результати.

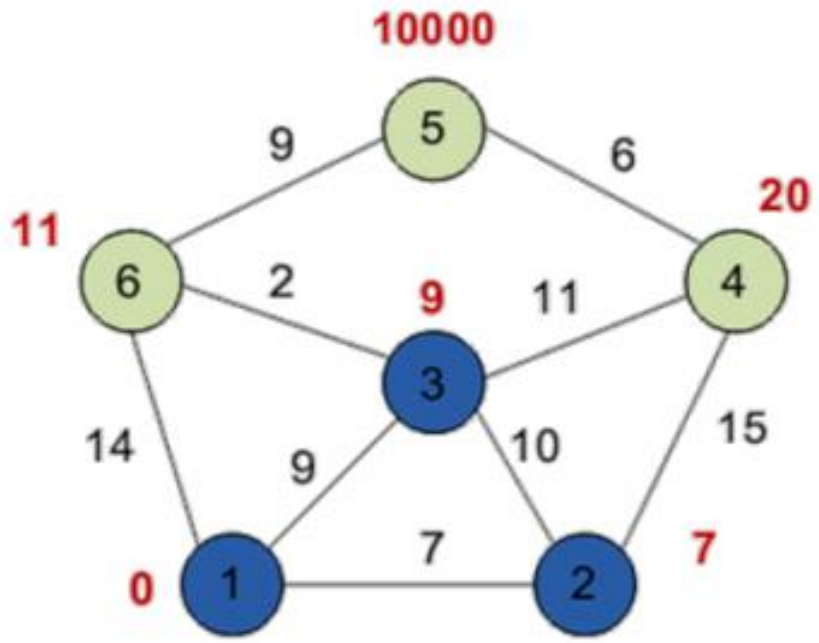


Рис. 2.11. Крок третій

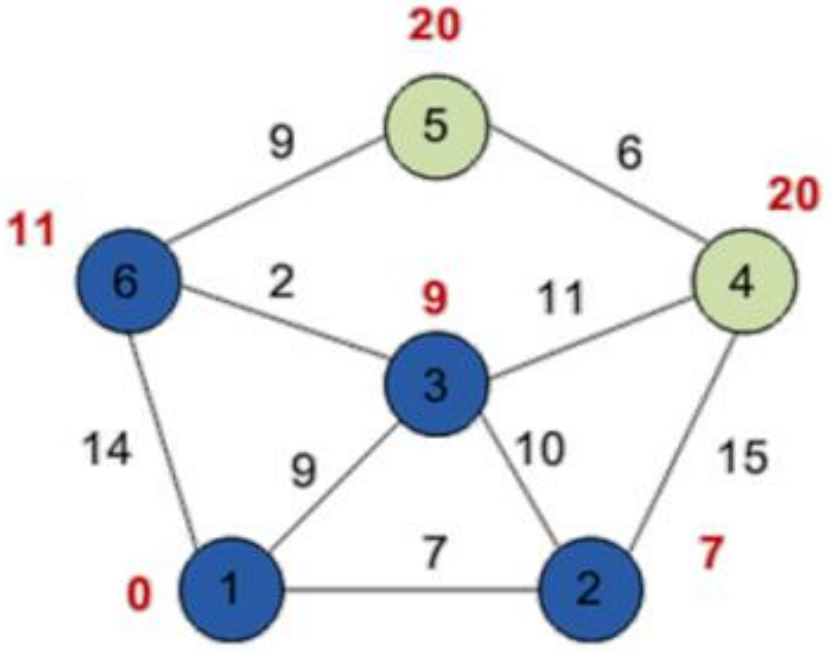


Рис. 2. 12. Крок четвертий

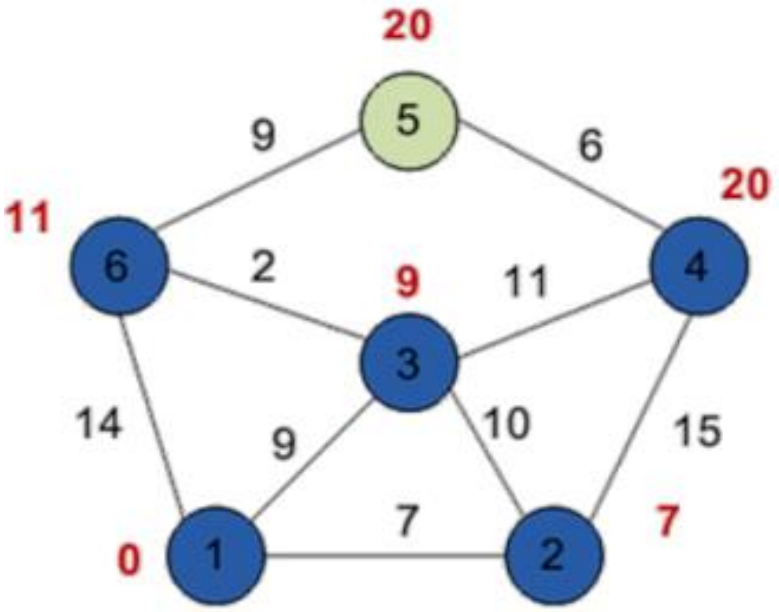


Рис. 2.13. Крок п’ятий

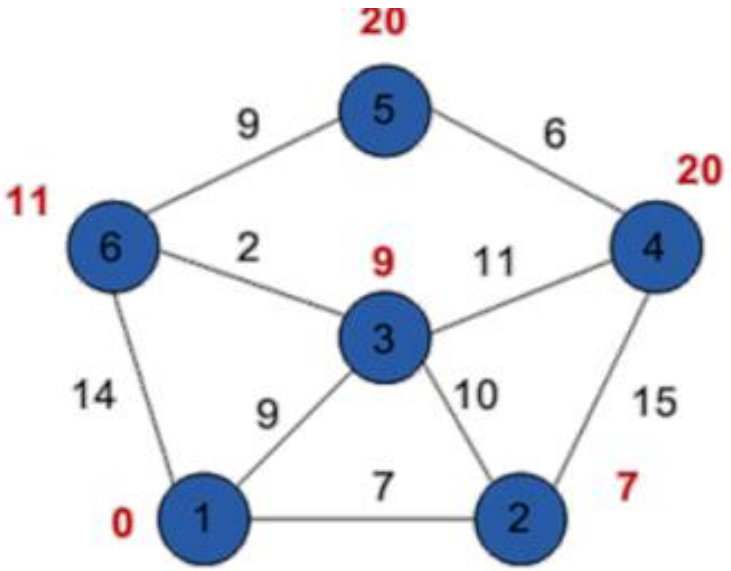


Рис. 2.14. Крок шостий

Таким чином, найкоротшим шляхом з вершини 1 у вершину 5 буде шлях через вершини 1 - 3 - 6 - 5, оскільки таким шляхом набирається мінімальна вага, рівна 20[6].

Блок-схема алгоритму Дейкстри наведена на рис. 2.8.

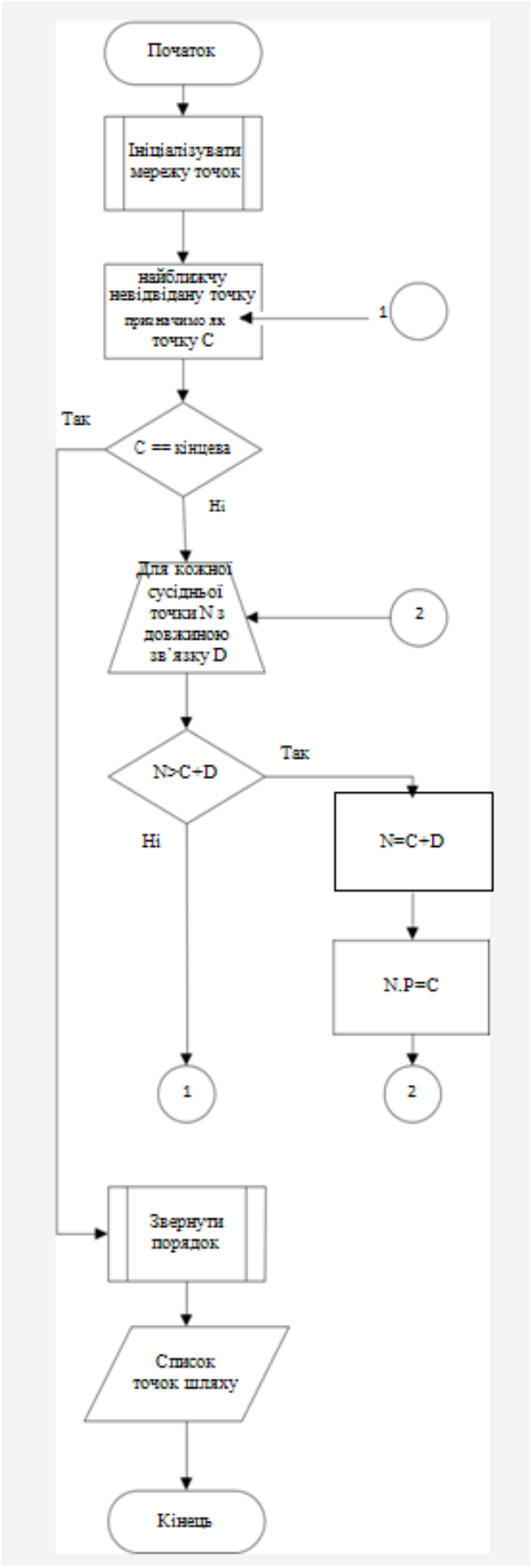


Рис. 2.8. Блок- схема алгоритму Дейкстри

2.4.3. Розробка бази даних системи

Для роботи програмного продукту використовується база даних (далі - БД). Щоб створити базу даних було використано систему керування базою даних Microsoft SQL Server.

База даних створюється для інформаційного обслуговування працівників. БД повинна містити дані про користувачів програми, водіїв, автомобілі та маршрутний лист.

Відповідно до предметної області система будується з урахуванням особливості, що за водієм може бути не закріплений один автомобіль.

Виділимо базові сутності цієї предметної області:

– Водії. Атрибути Водіїв – код водія, Прізвище Ім’я По батькові, посада;

– Автотранспорт. Атрибути автотранспорту – код транспорту, марка, витрата палива на 100 км;

– Автотранспорт-водій. Атрибути автотранспорт-водій - код водія, код транспорту;

– Користувач. Атрибути користувача – код користувача, логін, пароль, роль;

– Маршрутний лист. Атрибути листа – код листа, номер листа, Прізвище, Ім'я, По батькові, посада, автомобіль, підстава, дата, місце прибуття, відмітка про прибуття, відмітка про вибуття, підтверджуючий документ, витрати.

ER – діаграма автотранспортного підприємства приведена на рис. 2.15.

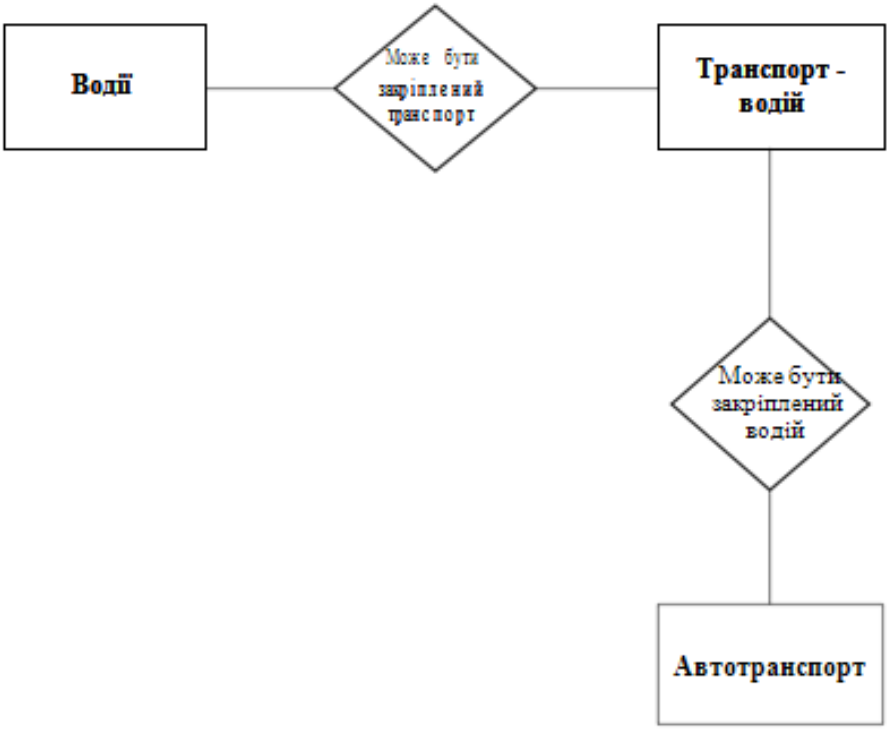


Рис. 2.15. ER - діаграма автотранспортного підприємства

Система створюється для обслуговування таких груп користувачів: адміністратор; бухгалтер; водій.

Визначимо межі інформаційної підтримки користувачів.

Функціональні можливості:

- ведення БД (запис, читання, модифікація);
- забезпечення логічної несуперечності БД;
- забезпечення захисту даних від несанкціонованого або випадкового доступу (визначення прав доступу);

База даних створюється на основі схеми бази даних. Для перетворення ER-діаграми в схему БД наведемо уточнену ER-діаграму, яка містить атрибути сутностей (рис. 2.16).

Рис. 2.16. Уточнена ER - діаграма автотранспортного підприємства

Кожне реляційне відношення відповідає однієї сутності і в нього вносяться всі атрибути сутності. Для кожного відношення необхідно визначити первинний ключ і зовнішні ключі (якщо вони є). У тому випадку, якщо базове відношення не має потенційних ключів, вводиться сурогатний первинний ключ, який не несе смислового навантаження і служить тільки для ідентифікації записів.

Відношення наведені в таблицях 2.1 – 2.5. Для кожного відношення вказані атрибути з їх внутрішнім назвою, типом і довжиною. Типи даних позначаються так: N - числовий, C – символьний.

Таблиця 2.1

Схема відношень Водії (Driver)

Зміст поля	Ім’я поля	Тип, довжина	Примітки
Код водія	DriveCode	N(4)	Первинний ключ
Прізвище, Ім’я, По батькові	DriveFIO	C(50)	Обов’язкове поле
Посада	DrivePosotion	C(20)	Обов’язкове поле

Таблиця 2.2

Схема відношень Автотранспорт (Car)

Зміст поля	Ім’я поля	Тип, довжина	Примітки
Код транспорту	CarCode	N(4)	Первинний ключ
Марка	CarName	C(20)	Обов’язкове поле
Витрати на 100км	CarAVG	C(20)	Обов’язкове поле

Таблиця 2.3

Схема відношень Автотранспорт-водій (DriverCar)

Зміст поля	Ім’я поля	Тип, довжина	Примітки
Код водія	DriveCode	N(4)	Зовнішній код
Код транспорту	CarCode	N(4)	Зовнішній код

Таблиця 2.4

Схема відношень Користувачі (Users)

Зміст поля	Ім’я поля	Тип, довжина	Примітки
Код користувача	UserCode	N(4)	Первинний ключ
Логін	UserLogin	C(13)	Обов’язкове поле
Пароль	UserPassword	C(13)	Обов’язкове поле
Роль	UserRole	C(9)	Обов’язкове поле

Таблиця 2.5

Схема відношень Маршрутний лист (LastList)

Зміст поля	Ім’я поля	Тип, довжина	Примітки
Код листа	TestCode	N(4)	Первинний ключ
Номер листа	TestNumber	N(4)	Обов’язкове поле
Прізвище, Ім'я, По батькові	TestFio	C(50)	Обов’язкове поле
Посада	TestClass	C(20)	Обов’язкове поле
Автомобіль	TestAuto	C(20)	Обов’язкове поле
Підстава	TestOsnova	C(30)	Обов’язкове поле
Дата	TestData	C(10)	Обов’язкове поле
Місце прибуття	TestCity	C(40)	Обов’язкове поле
Відмітка про прибуття	TestUp	C(20)	Обов’язкове поле
Відмітка про вибуття	TestDown	C(20)	Обов’язкове поле
Документ	TestDoc	C(40)	Обов’язкове поле
Витрати	TestPrice	C(10)	Обов’язкове поле

Наведемо для кожної групи користувачів права доступу до кожної таблиці і до кожного поля (атрибуту).

– Адміністратор БД: має доступ до всіх даних, може редагувати дані, встановлює права доступу для інших груп користувачів;

– Бухгалтер: мають доступ до формування маршрутного листа;

– Водій: має доступ до перегляду маршрутного листа та мапи.

Для роботи з БД необхідно підключити бібліотеку Sys-tem.Data.SqlClient, яка надає всі необхідні інструменти для роботи з нею.

2.4.4. Реалізація екранного інтерфейсу

Для кожної програми C# Windows Form створює один файл проекту та один файл форми. Файл проекту генерується при виборі пункту меню Файл/Створити проект/Додаток Windows Forms. Якщо в процесі розробки програми додаються форми і модулі, Visual Studio оновлює файли проекту.

Форма є одним з найважливіших елементів Windows Form. Процес редагування форми відбувається при додаванні до форми компонентів, зміні їх властивостей, створенні обробників подій. Файл форми як і файл коду (.cs) - містить вихідний код на мові програмування C# в який записуются всі класи і функції проекту.

Для реалізації описаного функціоналу було використано 10 форм. Розроблені форми мають візуальні та невізуальні компоненти. Розглянемо проектування кожної з цих форм:

Форма Login призначена для авторизації користувача у системі. Ескіз форми наведений на рис. 2.17, події та компоненти форми наведені в таблицях 2.6 – 2.7.

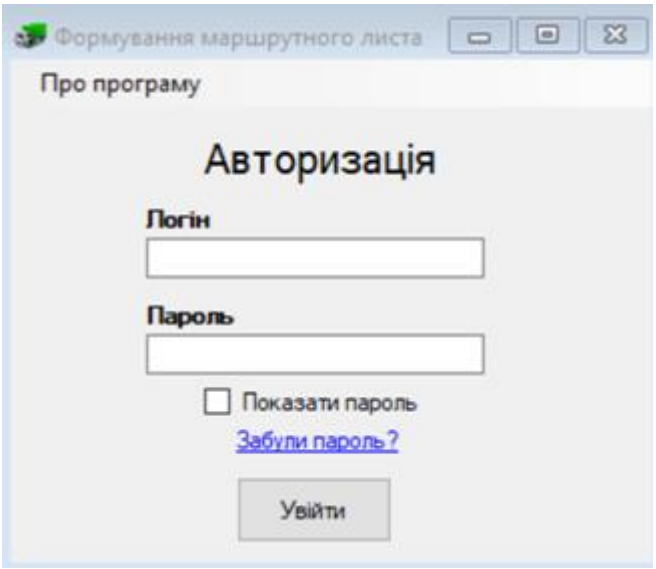


Рис. 2.17. Ескіз форми Login

Таблиця 2.6

Компоненти форми Login

Тип компоненту	Назва компоненту	Призначення
Button	Button1	Кнопка авторизації
TextBox	TextBox1	Текстове поле для вводу логіну
TextBox	TextBox2	Текстове поле для вводу па роля
menuStrip	menuStrip1	Головне меню
ToolStripMenuItem	проПрограму ToolStripMenuItem	Кнопка перехода до форми«Про програму»
Label	Label1	Напис «Логін»
Label	Label2	Напис «Пароль»
Label	Label3	Напис «Авторизація»
Label	Label4	Напис «Забули пароль»
CheckBox	CheckBox1	Відображення паролю
ToolTip	ToolTip1	Відображення підказки

Таблиця 2.7

Події компонентів форми Login

Назва компоненту	Призначення
checkBox1_CheckedChanged	Ініціалізація компонентів
button1_Click	Авторизація користувача
проПрограмуToolStripMenuItem_click	перехіда до форми «Про програму»

Функції - члени даної форми:

- private void loginscreen() – виконує перевірку на користувача та заносить його роль до класу Role.

Форма AddCar призначена для додавання в базу даних інформацію про автомобіль. Ескіз форми наведений на рис. 2.18, події та компоненти форми наведені в таблицях 2.8 – 2.9.

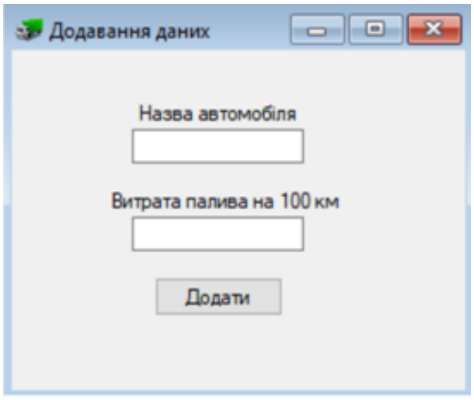


Рис. 2.18. Ескіз форми AddCar

Таблиця 2.8

Компоненти форми AddCar

Тип компоненту	Назва компоненту	Призначення
Button	Button1	Кнопка додавання нового авто мобіля
TextBox	TextBox1	Текстове поле для вводу назви автомобіля
TextBox	TextBox2	Текстове поле для вводу витрат палива
Label	Label1	Напис «Назва автомобіля»
Label	Label2	Напис «Витрата палива 100км
ToolTip	ToolTip1	Підказка

Таблиця 2.9

Події компонентів форми AddCar

Назва компоненту	Призначення
button1_Click	Додає новий запис в таблицю «Транспорт

Форма ListInfo призначена для редагування інформації в базі даних Ескіз форми наведений на рисунку 2.19, події та компоненти форми наведені в таблицях 2.10 – 2.11.

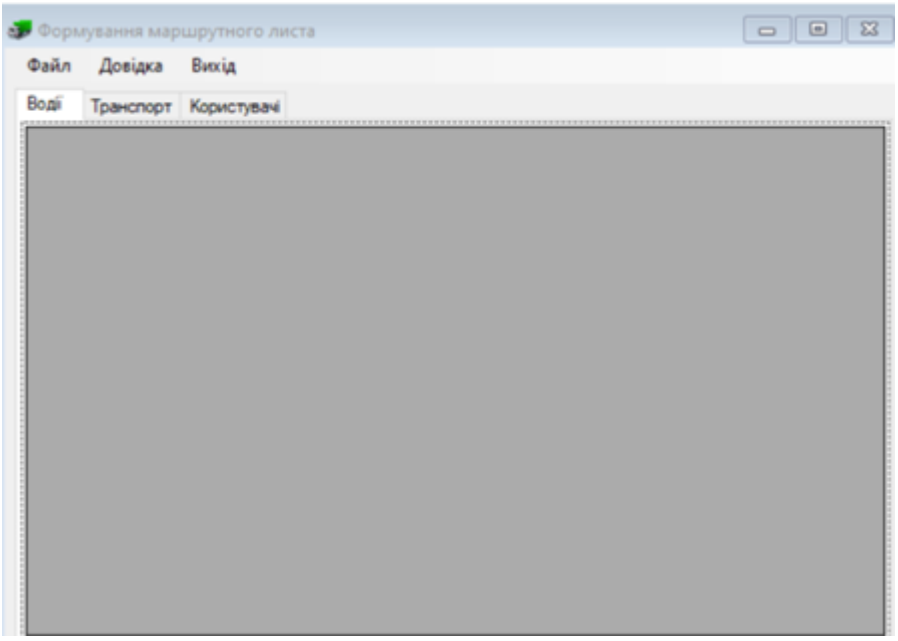


Рис. 2.19. Ескіз форми ListInfo

Таблиця 2.10

Компоненти форми ListInfo

Тип компоненту	Назва компоненту	Призначення
TabControl	TabControl1	Панель з вкладками
TabPage	TabPage1	Сторінка з таблицею водіїв
TabPage	TabPage2	Сторінка з таблицею автомобілів
TabPage	TabPage3	Сторінка таблиці користува чів
menustrip	Menustrip1	Функціональне меню
ToolStripMenuItem	файлToolStripMenu	кнопка яка відображає випада юче меню -
ToolStripMenuItem	довідкаToolS- рMenuItem	Кнопка переходу до форми «Довідка»-

Продовж. табл. 2.10

Тип компоненту	Назва компоненту	Призначення
ToolStripMenuItem	вихідToolS рMenuItem -	кнопка яка відображає випада юче меню -
ToolStripMenuItem	вихід- olStripMenuItem	Кнопка переходу до форми «ав торизації
ToolStripMenuItem	вToolStripMenuItem	Кнопка виходу з програми
ToolStripMenuItem	ствМаршЛистToolS рMenuItem	Кнопка переходу до форми «Формування маршрутного листа
ToolStripMenuItem	переОстанЛс oolStripMenuItem -	Кнопка переходу до форми «Ві дображення маршутного листа»-
dataGridView	dataGridView1	Відображає таблицю «Водії»
dataGridView	dataGridView2	Відображає таблицю «Транс порт»-
dataGridView	dataGridView3	Відображає таблицю «Користу вачі»-
ContexMenuStrip	ContexMenuStrip1	Контекстне меню
ToolStripMenuItem	додатиToolStrip enuItem1-	Кнопка переходу до форми «До давання -
ToolStripMenuItem	видали- ToolStripMenuIte m1	Кнопка видалення
ToolStripMenuItem	редагува ToolStripMenuIte m1-	Кнопка переходу до форми «Ре дагування -

Таблиця 2.11

Події компонентів форми ListInfo

Назва компоненту	Призначення
ListInfo_Load	Заповнення таблиць даними
редагуватиToolStripMenuItem1	Перехід до форми «Редагування»
видалитиToolStripMenuItem1	Видалення рядка
додатиToolStripMenuItem1	Перехід до форми «Додавання»
переглянутиОстанній Маршрут-нийЛистToolStripMenuItem_	Перехід до форми «Перегляду маршруту»
створитиМаршртуний-ЛистToolStripMenuItem	Перехід до форми «Формування марш-рутного листа»
вToolStripMenuItem	Вихід з програми
вихідЗАккаунтуToolStripMenuItem	Перехід до форми авторизації
довідкаToolStripMenuItem	Перехід до форми «Довідка»

Функції - члени даної форми:

- void datagrid() – заповнення таблиць даними.

Форма AddDriver призначена для додавання в базу даних інформацію про водія. Ескіз форми наведений на рис. 2.20, події та компоненти компоненти форми наведені в таблицях 2.12 – 2.13.

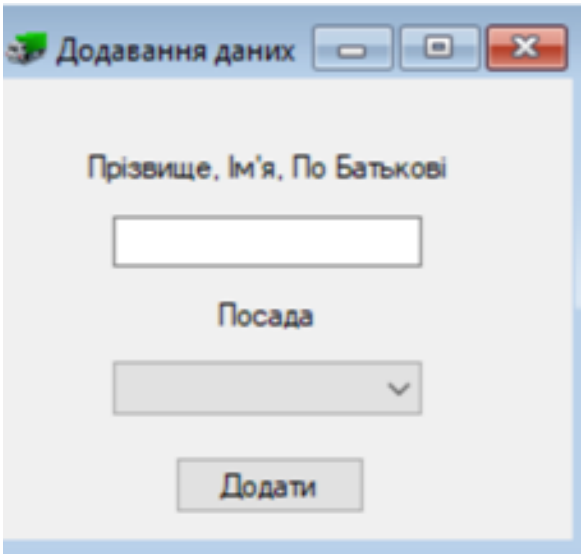


Рис. 2.20. Ескіз форми AddDriver

Таблиця 2.12

Компоненти форми AddDriver

Тип компоненту	Назва компоненту	Призначення
Button	Button1	Кнопка додавання нового водія
TextBox	TextBox1	Текстове поле для вводу Прізвища, Ім’я, По Батькові водія
Label	Label1	Напис«Прізвище, Ім’я, По Батькові»
Label	Label2	Напис «Посада»
ToolTip	ToolTip1	Підказка
combobox	Combobox1	Випадаюче текстове поле для вибору посади водія

Таблиця 2.13

Події компонентів форми AddCar

Назва компоненту	Призначення
button1_Click	Додає новий запис в таблицю «Водії»

Форма AddUser призначена для додавання в базу даних інформацію про користувача. Ескіз форми наведений на рис. 2.21, події та компоненти форми наведені в таблицях 2.14 – 2.15.

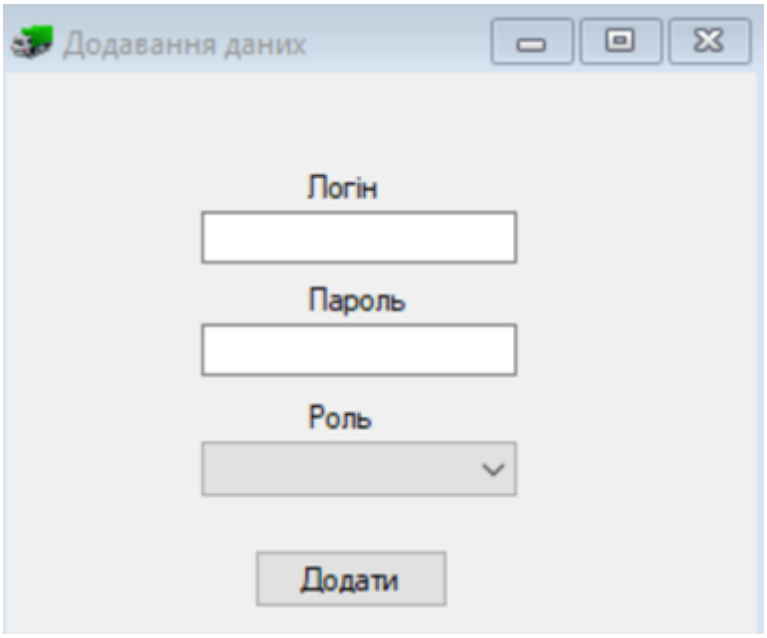


Рис. 2.21. Ескіз форми AddUser

Таблиця 2.14

Компоненти форми AddUser

Тип компоненту	Назва компоненту	Призначення
Button	Button1	Кнопка додавання нового користувача
TextBox	TextBox1	Текстове поле для вводу логіну
TextBox	TextBox2	Текстове поле для вводу ролю
Combobox	Combobox1	Випадаюче текстове поле для вибору ролі користувача
Label	Label1	Напис «Логін»
Label	Label2	Напис «Пароль»
Label	Label3	Напис «Роль»

Таблиця 2.15

Події компонентів форми AddUser

Назва компоненту	Призначення
button1_Click	Додає новий запис в таблицю «Користувачі»

Форма CreateList призначена для формування маршрутного листа. Ескіз форми наведений на рис. 2.22, події та компоненти форми наведені в таблицях 2.16 – 2.17.

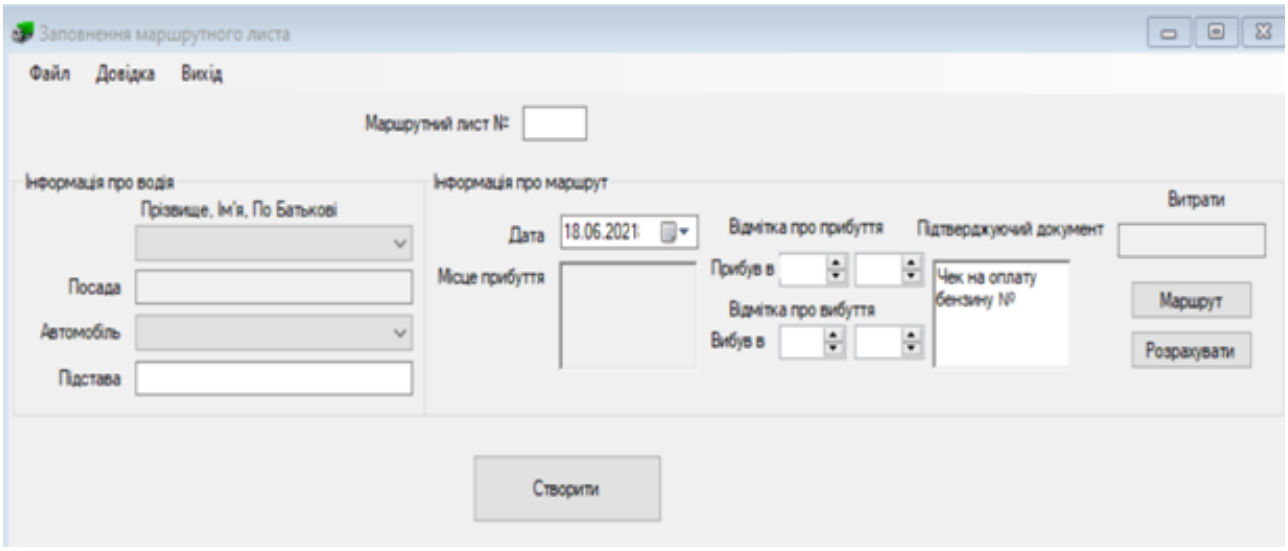


Рис. 2.22. Ескіз форми CreateList

Таблиця 2.16

Компоненти форми CreateList

Тип компоненту	Назва компоненту	Призначення
Button	Button1	Кнопка переходу до форми «Мапа»
Button	Button2	Кнопка для розрахунку маршруту

Продовж. табл. 2.16

Тип компоненту	Назва компоненту	Призначення
Button	Button3	Кнопка створення маршрутного листа
TextBox	TextBox1	Текстове поле для вводу посади
TextBox	TextBox2	Текстове поле для вводу підстави
TextBox	TextBox3	Текстове поле для вводу номера
TextBox	TextBox4	Текстове поле для відображення витрат
combobox	Combobox1	Випадаюче текстове поле для вибору водія
combobox	Combobox2	Випадаюче текстове поле для вибору автомобіля
RichTextBox	RichTextBox1	Текстове поле для відображення маршруту
RichTextBox	RichTextBox1	Текстове поле для вводу документа
DataTimePicker	DataTimePicker1	Випадаюче текстове поле для вибору дати
domainUpDown	domainUpDown1	Текстове поле для вибору часу
domainUpDown	domainUpDown2	Текстове поле для вибору часу
domainUpDown	domainUpDown3	Текстове поле для вибору часу
domainUpDown	domainUpDown4	Текстове поле для вибору часу
menustrip	Menustrip1	Функціональне меню
ToolStripMenuItem	BxToolStripMenuItem	Кнопка виходу з програми

Закінч. табл. 2.16

Тип компоненту	Назва компоненту	Призначення
ToolStripMenuItem	ДкаToolStripMenuItem	Кнопка відкриття довідки
GroupBox	GroupBox1	Контейнер для компонентів «інформація про водія»
GroupBox	GroupBox2	Контейнер для компонентів «інформація про маршрут»
Timer	Timer1	Виводить поточний час
Label	Label3	Напис «Посада»
Label	Label4	Напис «Автомобіль»
Label	Label5	Напис «Підстава»
Label	Label6	Напис «Дата»
Label	Label7	Напис «Місце прибуття»
Label	Label8	Напис «Відмітка про прибуття»
Label	Label9	Напис «Відмітка про вибуття»
Label	Label10	Напис «Підтверджуючий документ»
Label	Label11	Напис «Витрати»
Label	Label12	Напис «Прибув в»
Label	Label13	Напис «Вибув в»
ToolStripMenuItem	створитиМаршрт-ЛистToolStripMenuItem	Кнопка переходу до форми «Формування маршрутного листа»
ToolStripMenuItem	перегляну Маршрут-нийЛистToolStripMenuItem	Кнопка Відображення маршутного листа

Таблиця 2.17

Події компонентів форми CreateList

Назва компоненту	Призначення
Button1_Click	Перехід до форми «Мапа»
Button2_Click	Підрахунок витрат і вивід назви маршруту
Button3_Click	Створення маршрутного листа
comboBox1_SelectedIndexChange	Автоматичне заповненя текстового поля
comboBox2_SelectedIndexChange	Автоматичне заповненя даних
Timer1_tick	Запуск таймера для виводу поточного часу
вихідToolStripMenuItem_Click	Вихід з програми
отсаToolStripMenuItem_Click	Перехід до форми перегляду останього маршрутного листа
новийМаршрутний-ЛистToolStripMenuItem_Click	Оновлення вікна формування маршрутного листа

Функції - члени даної форми:

- void driverLoad() – завантажує дані водія в combobox1;
- void carload() – завантажує дані автомобіля combobox2;
- void statustime() – завантажує поля в menustrip.

Форма MapForm призначена для формування маршруту. Ескіз форми наведений на рис. 2.23, події та компоненти форми наведені в таблицях 2.18 – 2.19.

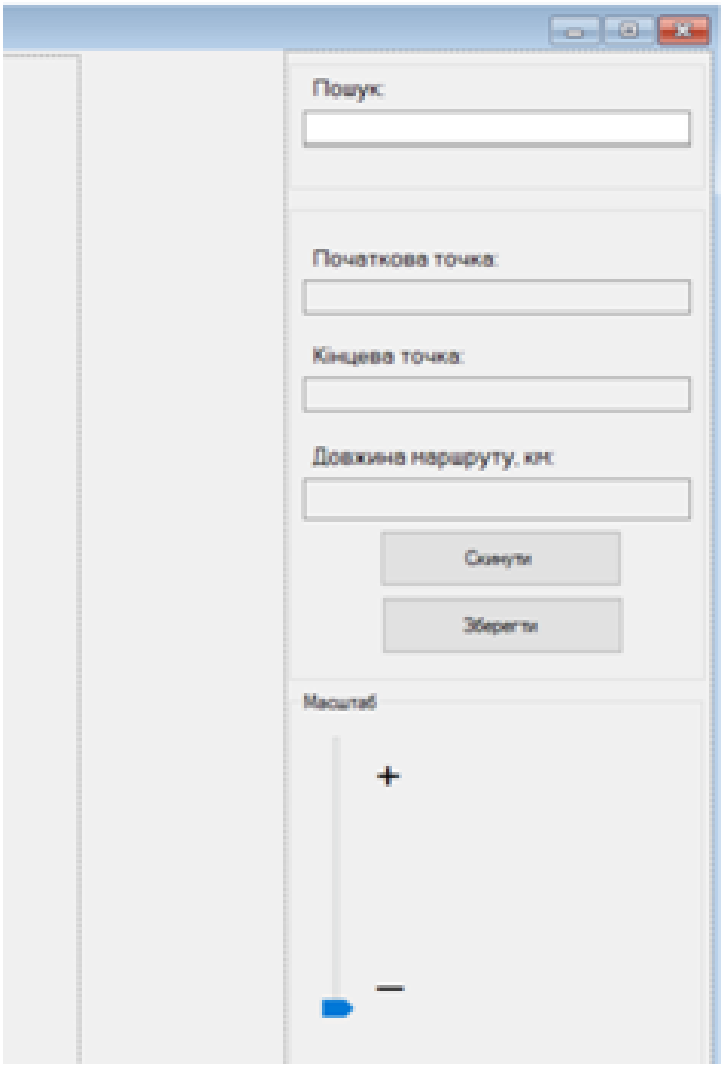


Рис. 2.23. Ескіз форми MapForm

Таблиця 2.18

Компоненти форми MapForm

Тип компоненту	Назва компоненту	Призначення
Button	btnReset	Кнопка для скидання маршруту
Button	Button1	Кнопка для зберігання даних маршруту
Panel	canvas	Панель для відображення мапи
Panel	PanelBottom	Панель для обмеження переміщення мапи

Продовж. табл. 2.18

Тип компоненту	Назва компоненту	Призначення
Panel	PanelRight	Панель для обмеження переміщення мапи
TextBox	TextBoxfind	Текстове поле для пошуку точки
Label	Label1	Напис «Початкова точка»
TextBox	textBoxStartaAdress	Текстове поле для відображення назви першої точки
TextBox	textBoxEndAdress	Текстове поле для відображення назви другої точки
TextBox	Textbox1	Текстове поле для відображення довжини маршруту
ComboBox	ComboBox1	Випадаюче текстове поле для вибору точки
GroupBox	GroupBox1	Контейнер для виводу даних про точки
GroupBox	GroupBox2	Контейнер для масштабування мапи
GroupBox	GroupBox3	Контейнер для пошуку точки
TrackBar	TrackBar1	При переміщенні повзунка змінює розмір мапи
Label	Label2	Напис «Кінцева точка»
Label	Label3	Напис «Довжина маршруту, км»
Label	Label4	Напис «Пошук»
Label	Label5	Напис «+»
Label	Label6	Напис «-»

Таблиця 2.19

Події компонентів форми MapForm

Назва компоненту	Призначення
TextBoxFind_TextChange	Пошук точки
trackBar1_Scroll	Зміна приближення мапи
btnReset_Click	Скидання маршруту
button1_Click	Збереження даних маршруту та закриття форми
Form1_Resize	Обмеження переміщення панелі для відтворення карти
panel1_Paint	Обробляє малювання на панелі для відтворення карти
panel1_MouseClick	Відмічання початкової та кінцевої точки
panel1_MouseDown	Запам'ятовується положення курсора
panel1_MouseMove	обмеження переміщення за край
panel1_MouseUp	При відпусканні перестати перетягувати панель
comboBox1_SelectedIndexChanged	При виборі точки відображає її на мапі

Функції - члени даної форми:

- void Reset() - обнулення значень;
- void LoadGraphFromFile() – завантаження графу з файлу;
- void DijkstrasTest() – реалізація алгоритму Дейкстри;
- void CreateShortestRouteTo() – формування шляху;

private void FindItemContaining() – вибір елементу combobox.

Форма ListReport призначена для відображення маршрутного листа. Ескіз форми наведений на рис. 2.24, компоненти форми наведені в таблиці 2.20.

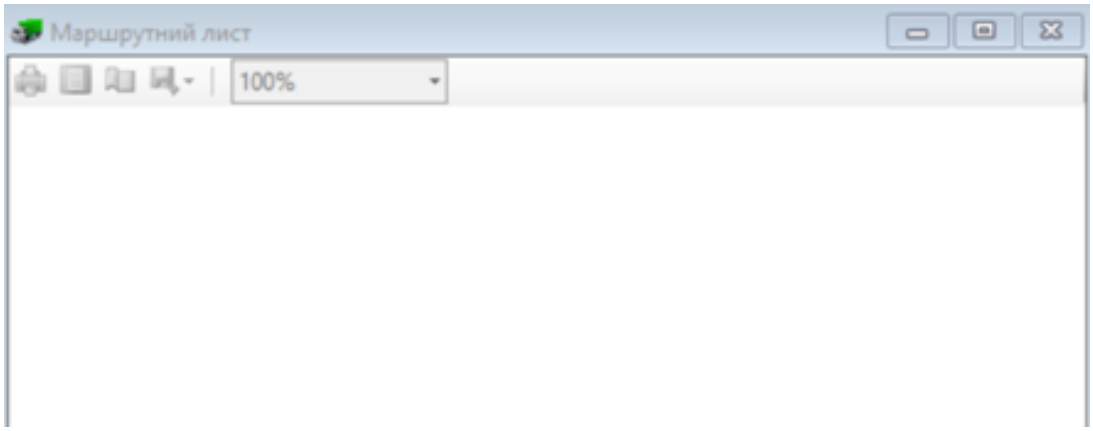


Рис. 2.24. Ескіз форми ListReport

Таблиця 2.20

Компоненти форми ListReport

Тип компоненту	Назва компоненту	Призначення
ReportViewer	ReportViewer1	Відображення маршрутного листа

Форма ProgramInfo призначена для відображення короткої інформації про програму. Ескіз форми наведений на рис. 2.25, компоненти форми наведені в таблиці 2.21.

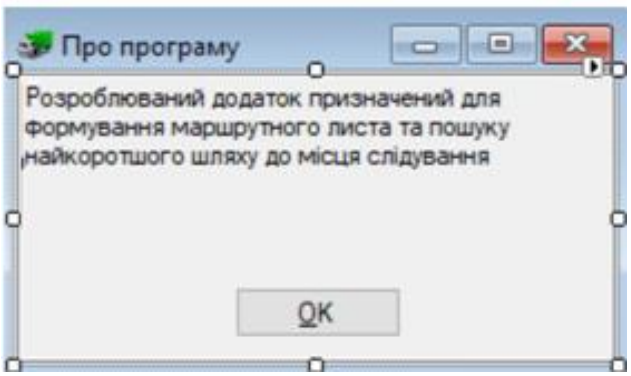


Рис. 2.25. Ескіз форми ProgramInfo

Таблиця 2.21

Компоненти форми ProgramInfo

Тип компоненту	Назва компоненту	Призначення
RichTextBox	RichTextBox1	Текстове поле для відображення інформації про програму
Button	Button1	Закриває форму

Форма Sparvka призначена для відображення інформації по роботі з віками. Ескіз форми наведений на рис. 2.26, компоненти форми наведені в таблиці 2.22.

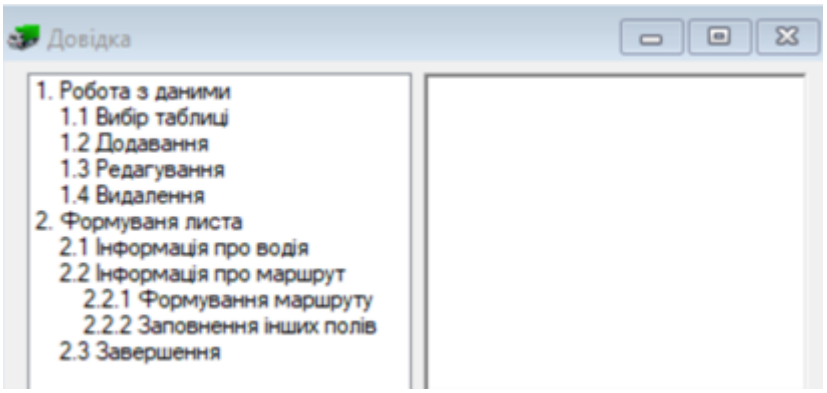


Рис. 2.26. Ескіз форми Sparvka

Таблиця 2.22

Компоненти форми Sparvka

Тип компоненту	Назва компоненту	Призначення
RichTextBox	RichTextBox1	Текстове поле для відображення інформації
Listbox	Listbox1	Текстовий список з вибором

2.4.5. Опис файлової структури програми

У файлів проекту є певний набір ключових елементів:

– бібліотека System – це простір імен, що включає класи, що представляють собою базові типи даних, що використовуються всіма додатками;

– бібліотека System.Collections.Generic містить інтерфейси і класи, які визначають різні колекції об'єктів, такі як списки, черги, виконавчі масиви, хеш-таблиці і словники;

– бібліотека System.ComponentModel надає класи, використовувані для реалізації поведінки компонентів і елементів управління під час розробки та виконання;

– бібліотека System.Data надає доступ до класів, які представляють ADO.NET архітектуру;

– бібліотека System.Drawing забезпечує доступ до функціональних можливостей графічного інтерфейсу;

– бібліотека System.Linq містить класи та інтерфейси, які підтримують LINQ;

– бібліотека System.Text містить класи, що представляють кодування символів ASCII і Unicode;

– бібліотека System.Threading.Tasks надає типи, які спрощують роботу з написання паралельного і асинхронного коду;

– бібліотека System.Windows.Forms представляє вікно або діалогове вікно, яке становить для користувача інтерфейс програми;

– простір імені namespace в якому зберігається весь код програми;

– основний клас програми class Program[10].

Програмний продукт для формування маршрутного листа складається з наступних файлів:

– Properties - файл, у якому зберігаються всі налаштування для роботи програми;

– References – файл, який містить посилання на завантаженні бібліотеки;

– папка bin, в ній зберігаються всі встановлені плагіни та дані для роботи мапи;

– папка SqlServerTypes, в ній зберігаються необхідні компоненти для роботи звіту;

– AddCar.cs - файл форми додавання транспорту до бази даних, в якому описуються функції та класи для роботи з нею;

– AddDriver.cs - файл форми додавання даних щодо водія до бази даних, в якому описуються функції та класи для роботи з нею;

– AddUser.cs - файл форми додання даних користувача до бази даних, в якому описуються функції та класи для роботи з нею;

– CreateList.cs - файл форми створення маршрутного листа, в якому описуються функції та класи для роботи з нею;

– Form1.cs - файл форми формування маршруту, в якому описуються функції та класи для роботи з нею;

– MainForm.cs - файл основної форми, в якому описуються функції та класи для роботи з нею;

– Login.cs. - файл форми авторизації, в якому описуються функції та класи для роботи з нею;

– ProgramInfo.cs. - файл форми для відображення інформації про програму, в якому описуються функції та класи для роботи з нею;

– spravka.cs. - файл форми з поясненнями до заповнення листа, в якому описуються функції та класи для роботи з нею;

– ListReport.cs - файл форми для перегляду звіту, в якому описуються функції та класи для роботи з нею;

– TransportTest.mdg. - файл бази даних, в якому зберігаються дані всіх таблиць бази даних;

– DataBank.cs - клас для передачі даних між формами;

– Edge.cs - клас ребер графа;

– Graph.cs - клас графа;
– Line.cs - клас з’єднувальної лінії;
– Node.cs - клас вершини графа;
– Role.cs - клас для збереження ролі користувача.

2.5. Обґрунтування та організація вхідних та вихідних даних програми

У якості вхідних даних виступають введені у користувачами дані щодо водія (прізвище, ім’я, по-батькові, посада, автомобіль, підстава проїздки) та інформація про маршрут (дата прибуття, місце прибуття, час прибуття та вибуття, підтверджуючий документ, витрати), а також дані для авторизації користувача у системі (логін та пароль).

Вихідними даними програми є прорахований оптимальний маршрут поїздки, сформований та готовий до друку маршрутний лист, який можна буде експортувати у зовнішній файл форматів Word або Excel.

2.6. Опис роботи розробленої системи

2.6.1. Використані технічні засоби

Для розробки ІС використовувався ПК з наступними системними вимогами:

– 1 GB RAM;
– 2 GB вільного простору на жорсткому диску;
– стандартний GPU з підтримкою DirectX 9.0;
– роздільна здатність екрану 1024x768;
– Intel® 1.4 GHz ;
– переферійні пристрої введення-виведення: клавіатура, миша.

2.6.2. Використані програмні засоби

Для розробки інформаційної системи для автоматизації формування маршрутного листа було використано середовище розробки Microsoft Visual Studio 2019, мову програмування C#, а для реалізації бази даних було обрано систему керування базою даних СКБД MS SQL Server.

Додаток написаний за допомогою методів об'єктно-орієнтованого програмування на мові C# і скомпільований з розширенням exe. Він працює у середовищі операційної системи Windows та Unix.

Для повноцінної роботи розробленого програмного додатку необхідно мати встановлений на комп'ютері текстовий редактор MS Word або електронний редактор таблиць MS Excel так як сформований маршрутний лист зберігається в файлах редактора.

2.6.3. Виклик та завантаження програми

Для запуску програмного додатку необхідно відкрити WindowsFormDijkstra.exe файл в папці програми

2.6.4. Опис інтерфейсу користувача

Після запуску програмного додатку користувач потрапляє у вікно авторизації, в якому йому необхідно ввести свій логін та пароль. Також у вікні авторизації розташовані перемикач для відображення паролю та напис «Забули пароль ?», який при наведені відображає підказку (рис. 2.27).

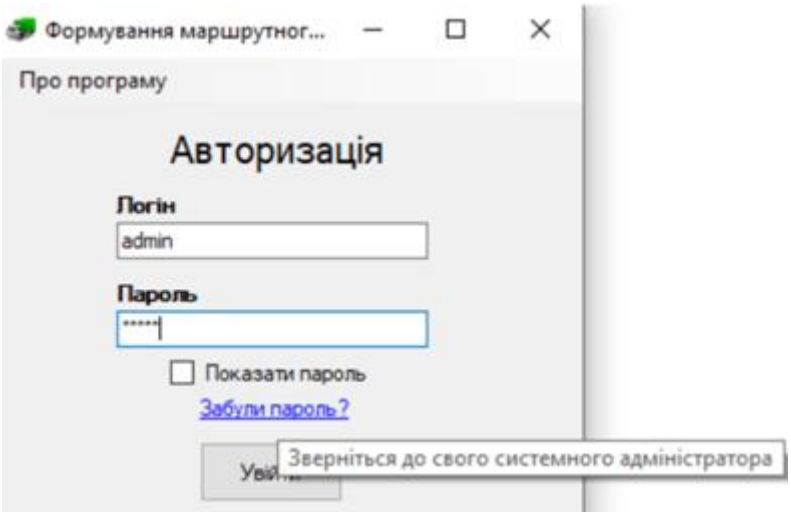


Рис. 2.27. Вікно авторизації користувача у системі

При натисканні кнопки головного меню «Про програму» користувач потрапляє у вікно з коротким описом програми (рис. 2.28)

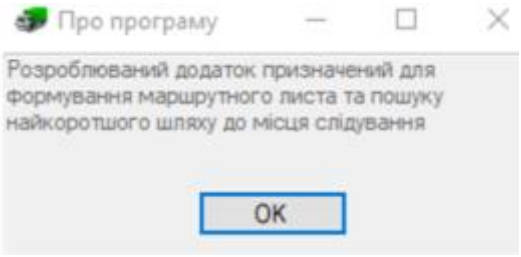


Рис. 2.28. Вікно «Про програму»

Після проходження авторизації користувач, в залежності від ролі, потрапляє у робоче вікно.

Адміністратору відкриється вікно роботи з усіма даними автотранспортного підприємства. При відкриті цього вікна будуть відображені вкладки з відповідними назвами таблиць (рис. 2.29 – 2.31).

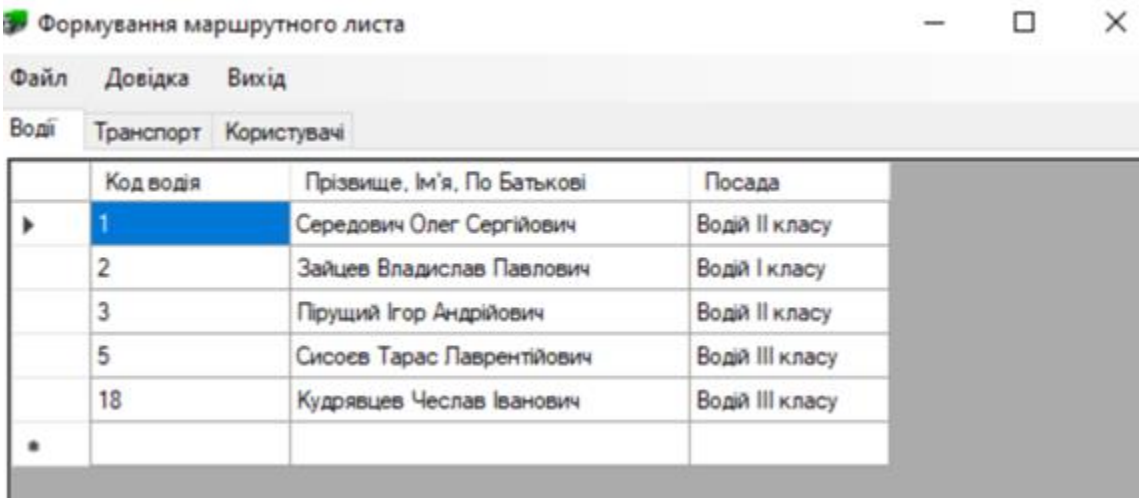


Рис. 2.29. Вікно роботи з даними програми щодо водіїв

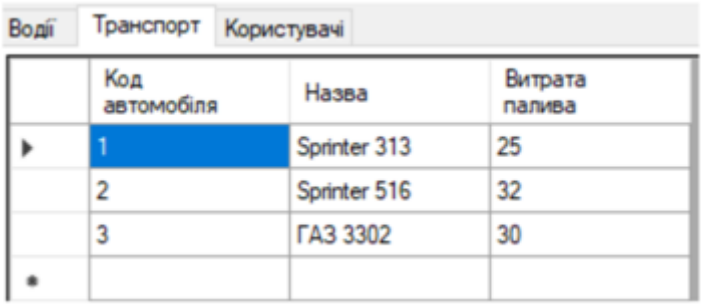


Рис. 2.30. Вікно роботи з даними програми щодо транспортного засобу

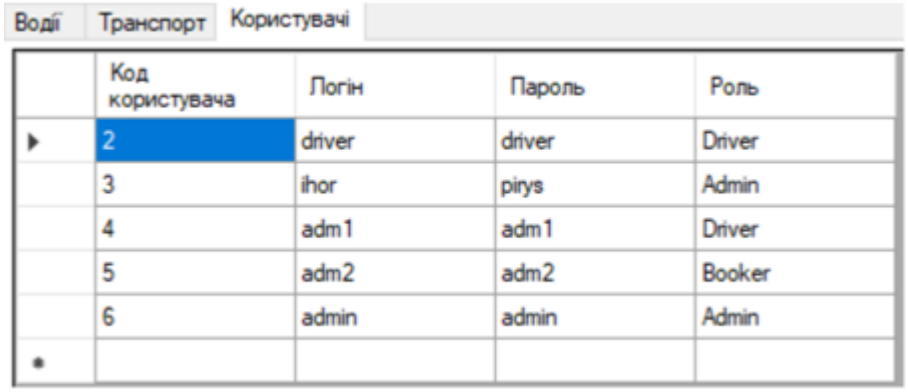


Рис. 2.31. Вікно роботи з даними програми щодо користувачів

Для роботи з даними використовується контекстне меню, яке з'являється при натисканні правої кнопки миші на відповідну область таблиці (рис 2.32).

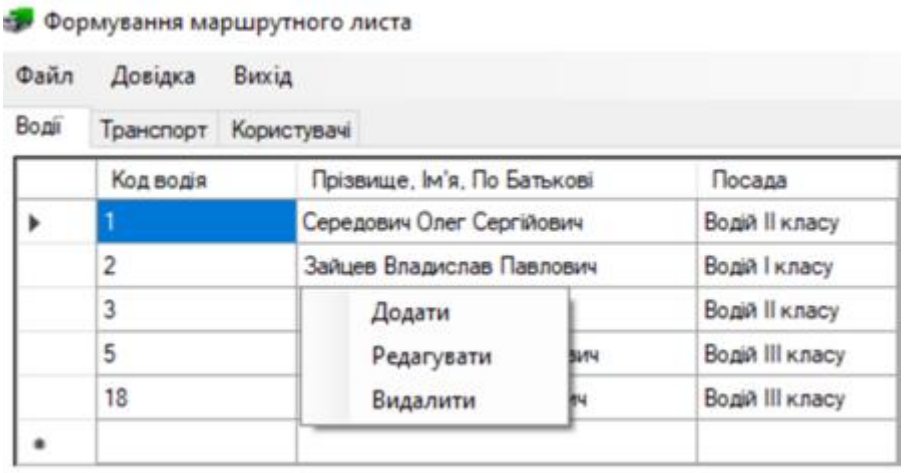


Рис. 2.32. Активоване контекстне меню для роботи з даними у програмі

Після натискання пункту «Додати» контекстного меню, з’явиться вікно «Додавання даних», яке містить текстові поля для вводу прізвища, ім’я, по-батькові та випадаючий список для вибору класу водія. При спробі ввести цифри в текстове поле спрацює виключення, яке відобразить повідомлення про заборону на введення цифр (рис 2.33). Після натискання кнопки «Додати» вікно закриється та внесуться зміни в базу даних. Приклад додавання даних у систему щодо нового водія наведений на рис. 2.34-2.35.

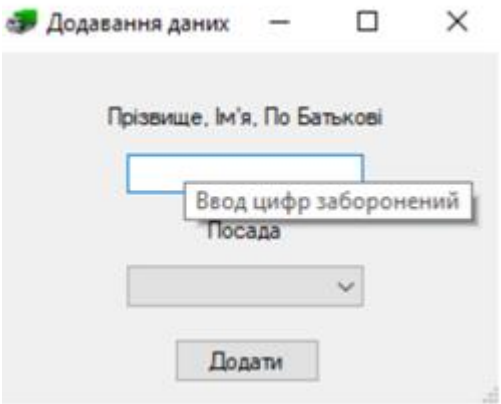


Рис. 2.33. Вікно додавання/редагування даних щодо водія

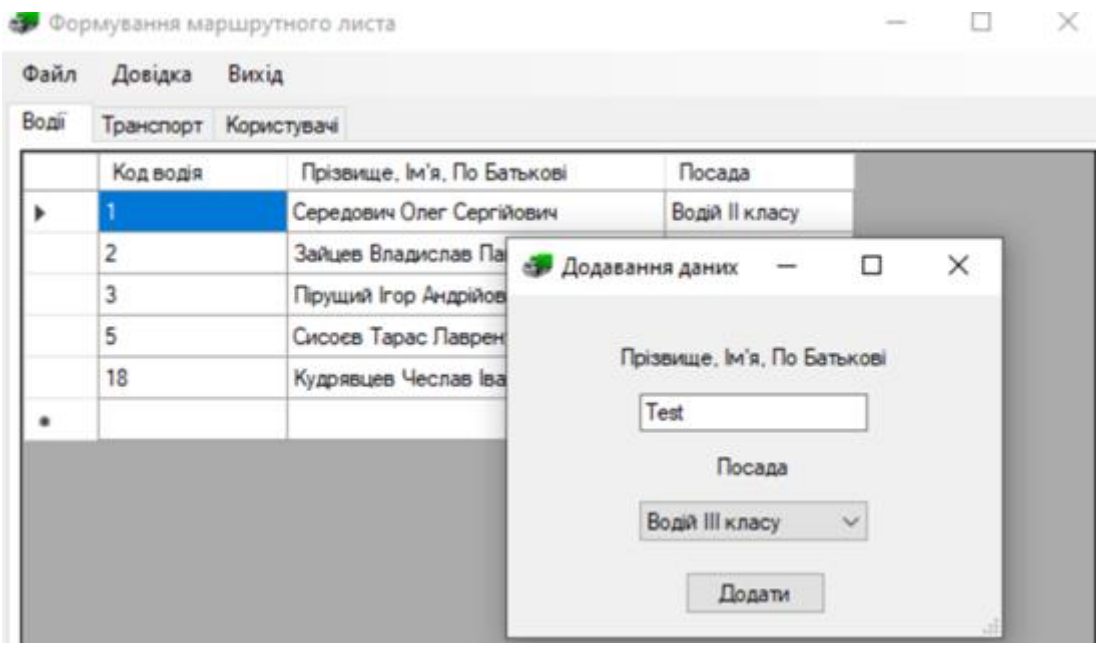
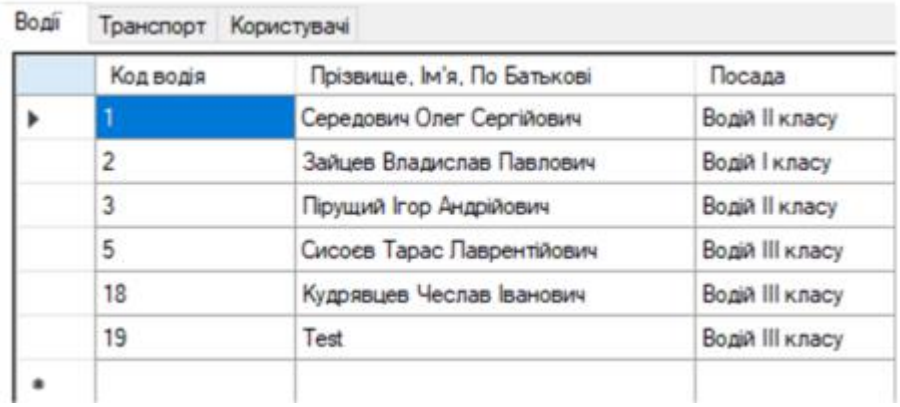


Рис. 2.34. Вікно додавання даних про нового водія до бази даних



Код водія	Прізвище, Ім'я, По Батькові	Посада
1	Середович Олег Сергійович	Водій II класу
2	Зайцев Владислав Павлович	Водій I класу
3	Піруший Ігор Андрійович	Водій II класу
5	Сисоєв Тарас Лаврентійович	Водій III класу
18	Кудрявцев Чеслав Іванович	Водій III класу
19	Test	Водій III класу

Рис. 2.35. Оновлена таблиця водіїв додатка

Для редагування інформації у системі треба вибрати необхідний рядок у таблиці та обрати пункт «Редагувати» в контекстному меню. Приклад редагування інформації про водія наведений на рис. 2.36 – 2.37.

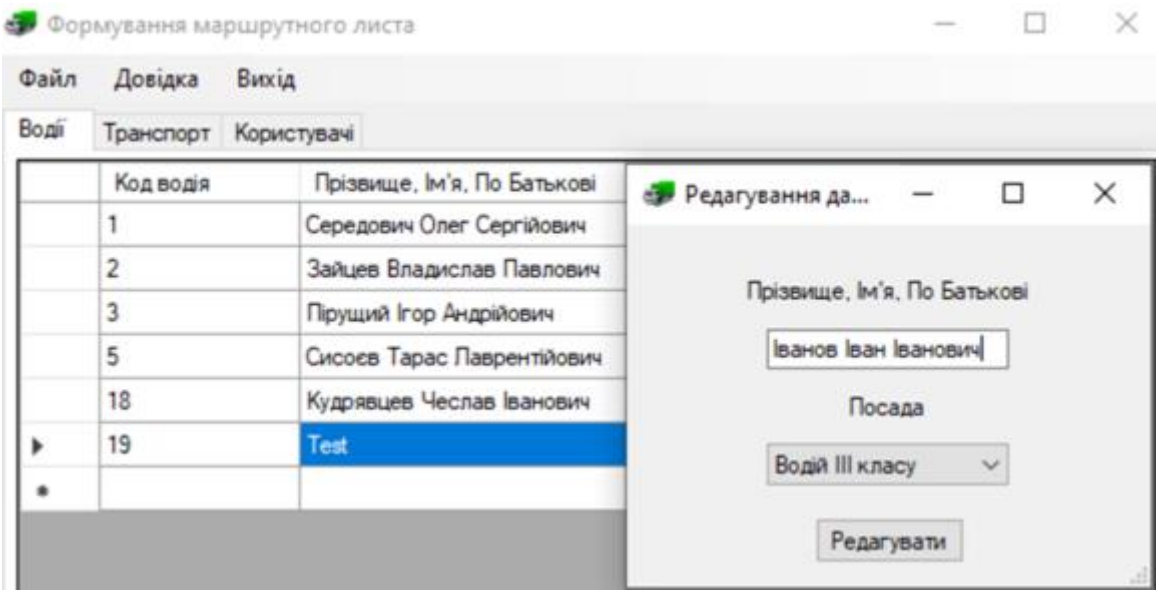


Рис. 2.36. Вікно «Редагування даних» щодо водія

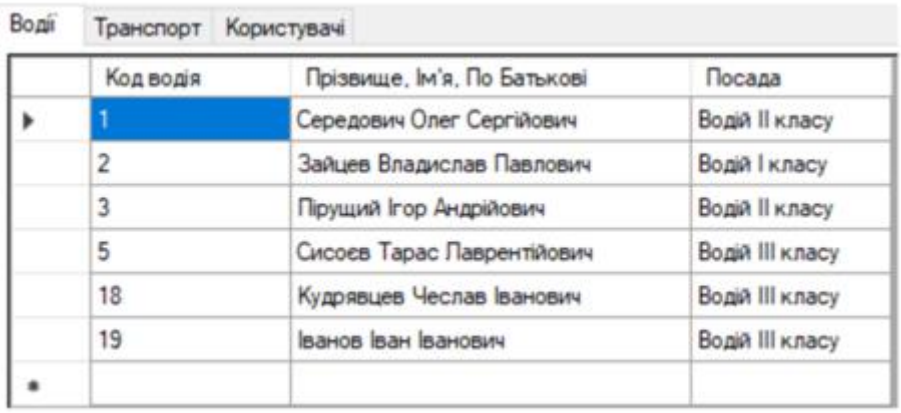


Рис. 2.37. Таблиця «Водії» після редагування інформації

Для видалення інформації з системи треба вибрати необхідний рядок у таблиці даних та обрати пункт «Видалити» в контекстному меню. Після натиснення пункту меню «Видалити» перед користувачем з’явиться діалогове вікно для підтвердження дій користувача. При натисканні кнопки «Ок» вибраний рядок буде видалений з бази даних. Приклад видалення інформації із системи щодо водія наведений на рис. 2.38 – 2.39.

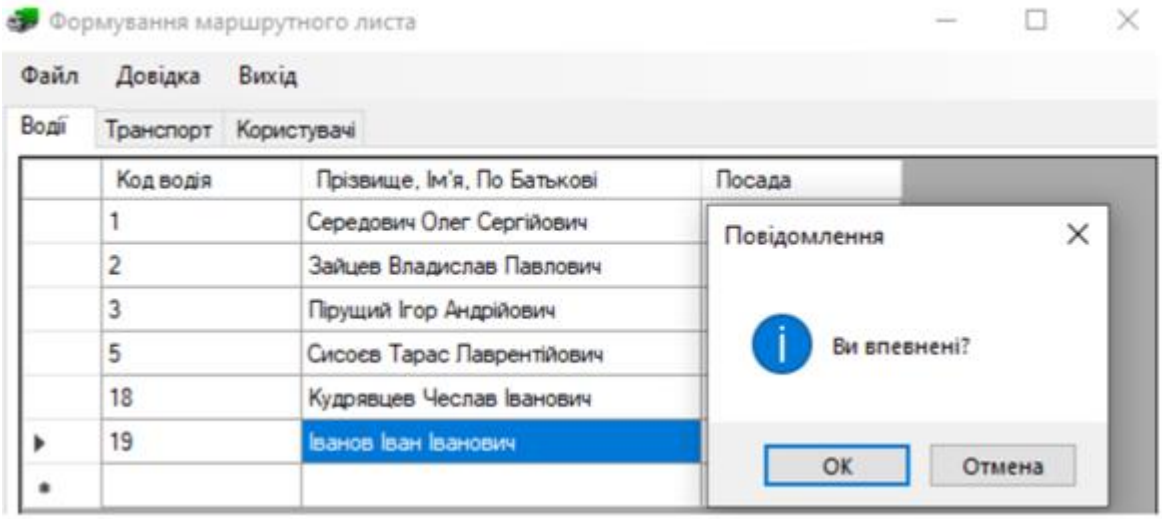


Рис. 2.38. Вікно з активованою функцією для видалення інформації про водія

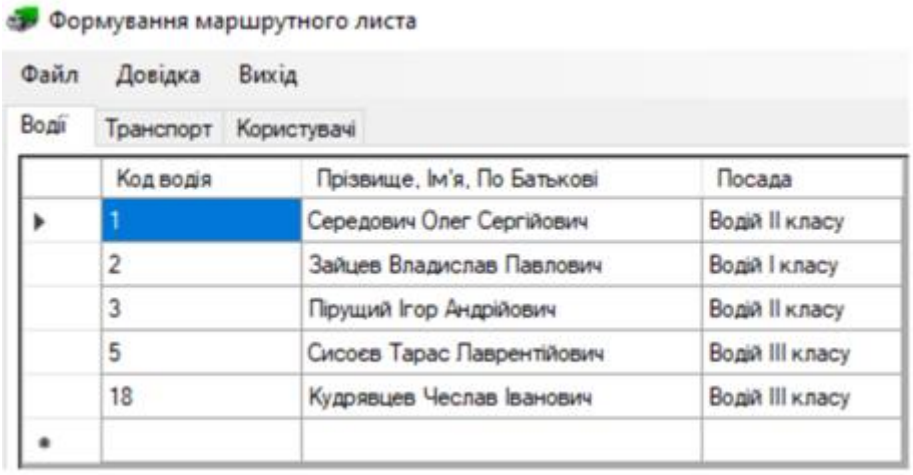


Рис. 2.39. Таблиця «Водії» після видалення даних

Розглянемо головне меню головного вікна програми.

Випадаючий пункт меню «Вихід» (рис. 2.40) складається з двох підпунктів «Вихід з акаунту» та «Вихід з програми». При виборі підпункту «Вихід з акаунту» користувач потрапляє у вікно авторизації, яке розглядалося раніше (рис. 2.40). Пункт «Вихід з програми» закриває додаток.

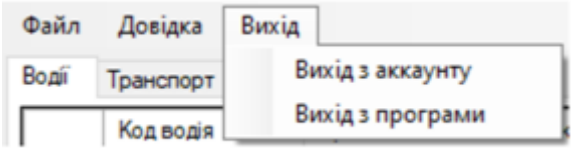


Рис. 2.40. Активований пункт меню «Вихід» головного вікна програми

Пункт «Довідка» відкриває вікно з поясненнями по роботі з додатком (рис. 2.41).

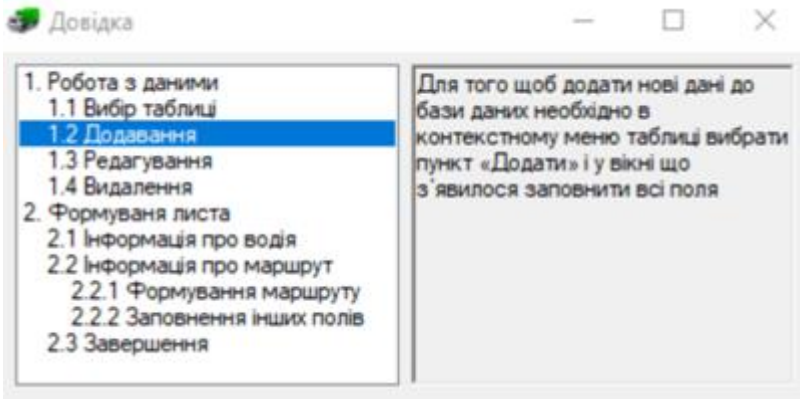


Рис. 2.41. Вікно «Довідка» додатка

Пункт головного меню «Файл» (рис. 2.42) складається з двох підпунктів «Новий маршрутний лист» та «Останній маршрутний лист». При виборі підпункту «Останній маршрутний лист» користувач потрапляє у вікно перегляду останнього сформованого маршрутного листа (рис. 2.43). Підпункт «Новий маршрутний лист» відкриває вікно для створення нового маршрутного листа (рис. 2.44).

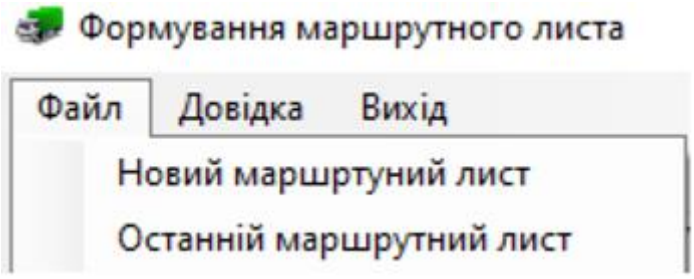


Рис. 2.42. Активований пункт меню «Файл» головного вікна додатка

Маршрутний лист № 432

Прізвище, Ім'я, По батькові: Зайцев Владислав Павлович

Посада: Водій I класу

Автомобіль: ГАЗ 3302

Підстава: Службова поїздка

Дата	Місце прибуття	Відмітка про прибуття	Відмітка про вибуття	Підтверджуючий документ	Витрати
06.06.2021	вул. Січови Стрільців 89	Прибув в 7 год 0 хв	Вибув в 9 год 0 хв	Чек на оплату бензину №43	24.102

Рис. 2.43. Вікно для перегляду останнього створеного у системі маршрутного листа

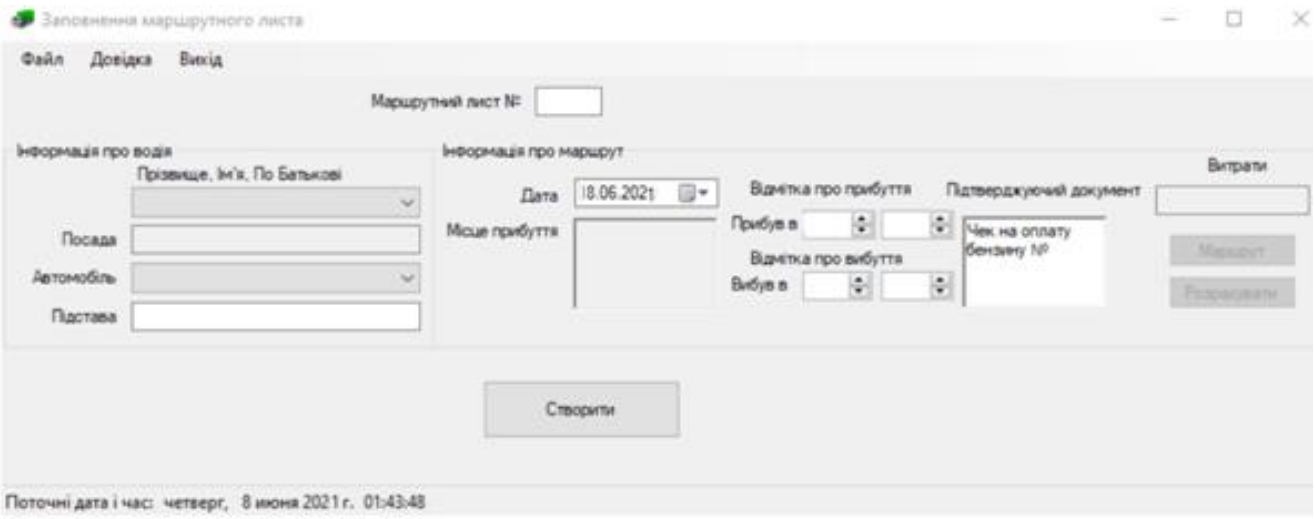


Рис. 2.44. Вікно формування нового маршрутного листа

Розглянемо процес створення нового маршрутного листа.

Для заповнення даних про водія необхідно в випадаючих текстових полях «Прізвище», «Ім’я», «По-батькові» та «Автомобіль» вибрати відповідні дані про водія і транспортний засіб та заповнити підставу поїздки.

Після заповнення інформації про водія користувачу відкриється можливість натиснути кнопку «Маршрут» (рис. 2.45). Після натискання цієї кнопки відкриється вікно формування маршруту на мапі (рис. 2.46).

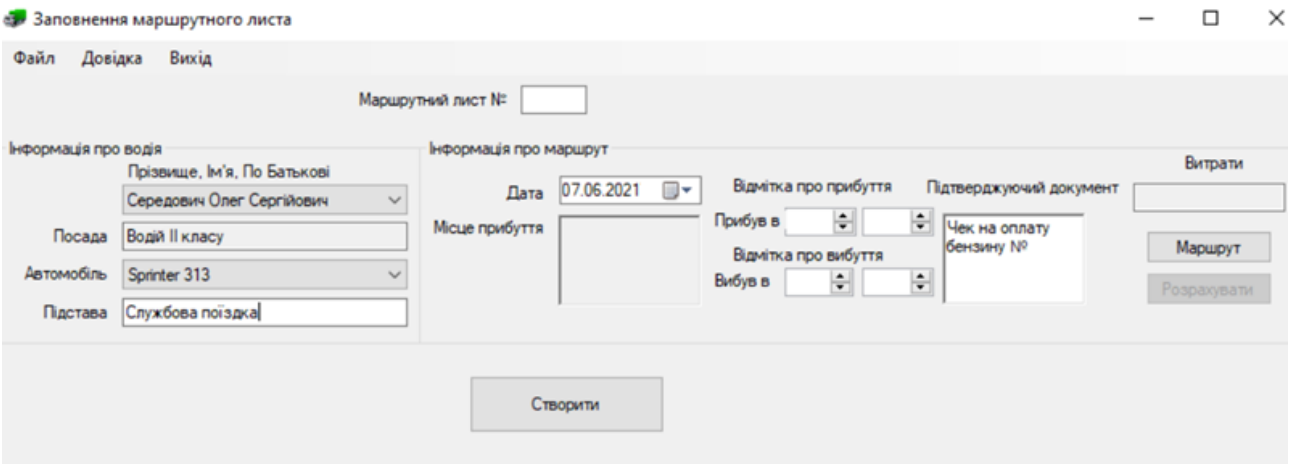


Рис. 2.45. Головне вікно додатка з заповненою інформацією для маршрутного листа

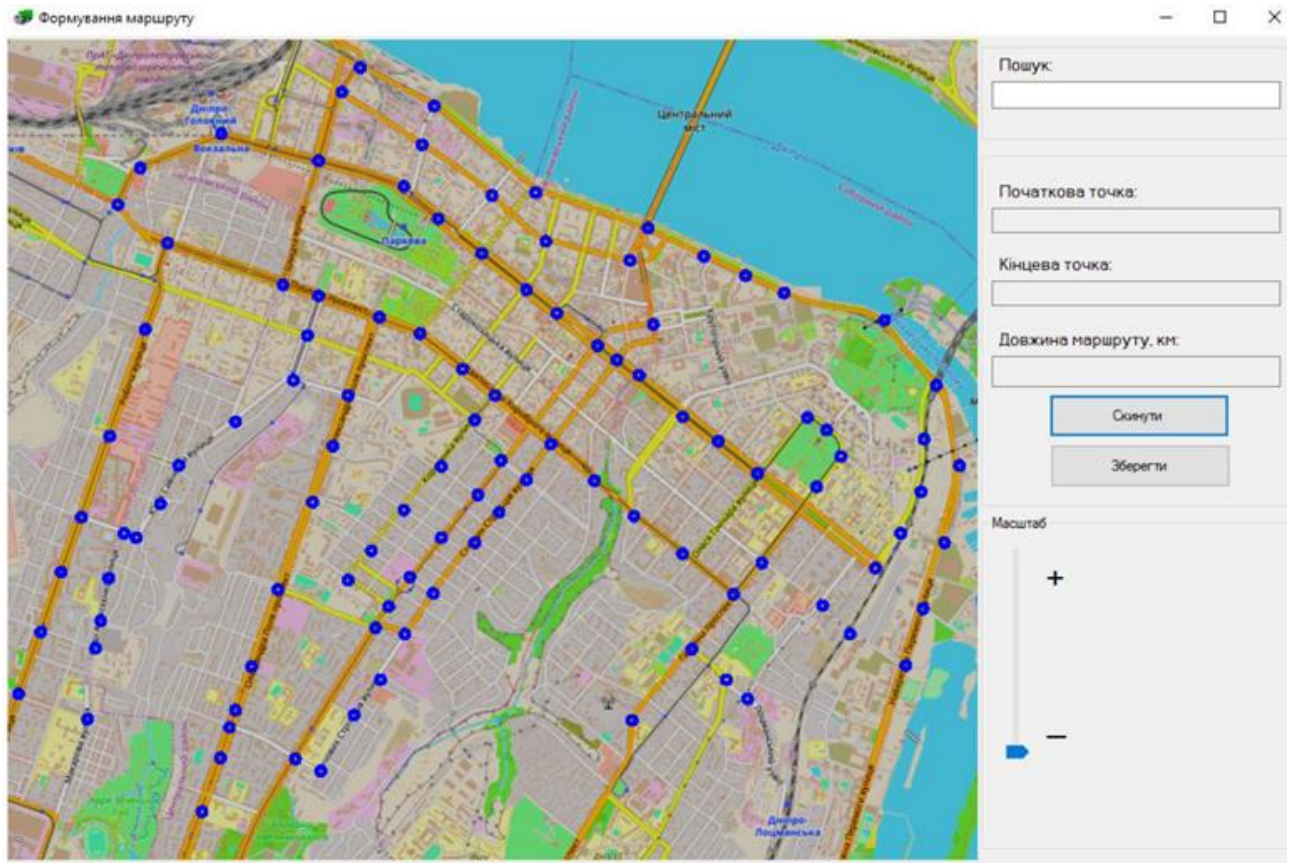


Рис. 2.46. Вікно «Формування маршруту» додатка

В цьому вікні користувачу надається можливість формування маршрута. Для цього необхідно вибрати дві точки на мапі або скористатися пошуком (рис. 2.47 - 2.48). Для більш детального огляду мапи у вікні є стрічка для зміни масштабу (рис. 2.49). Для видалення маршруту необхідно натиснути на кнопку «Скинути».

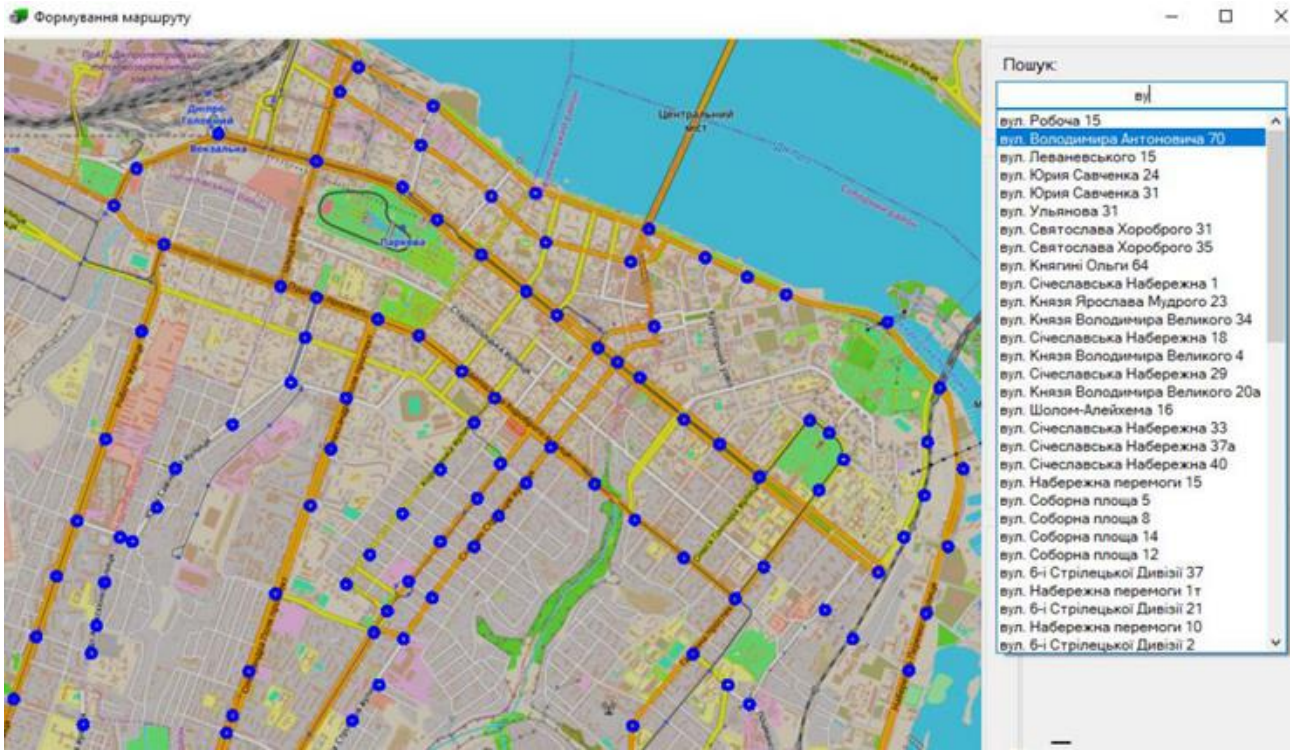


Рис. 2.47. Пошук адреси на мапі

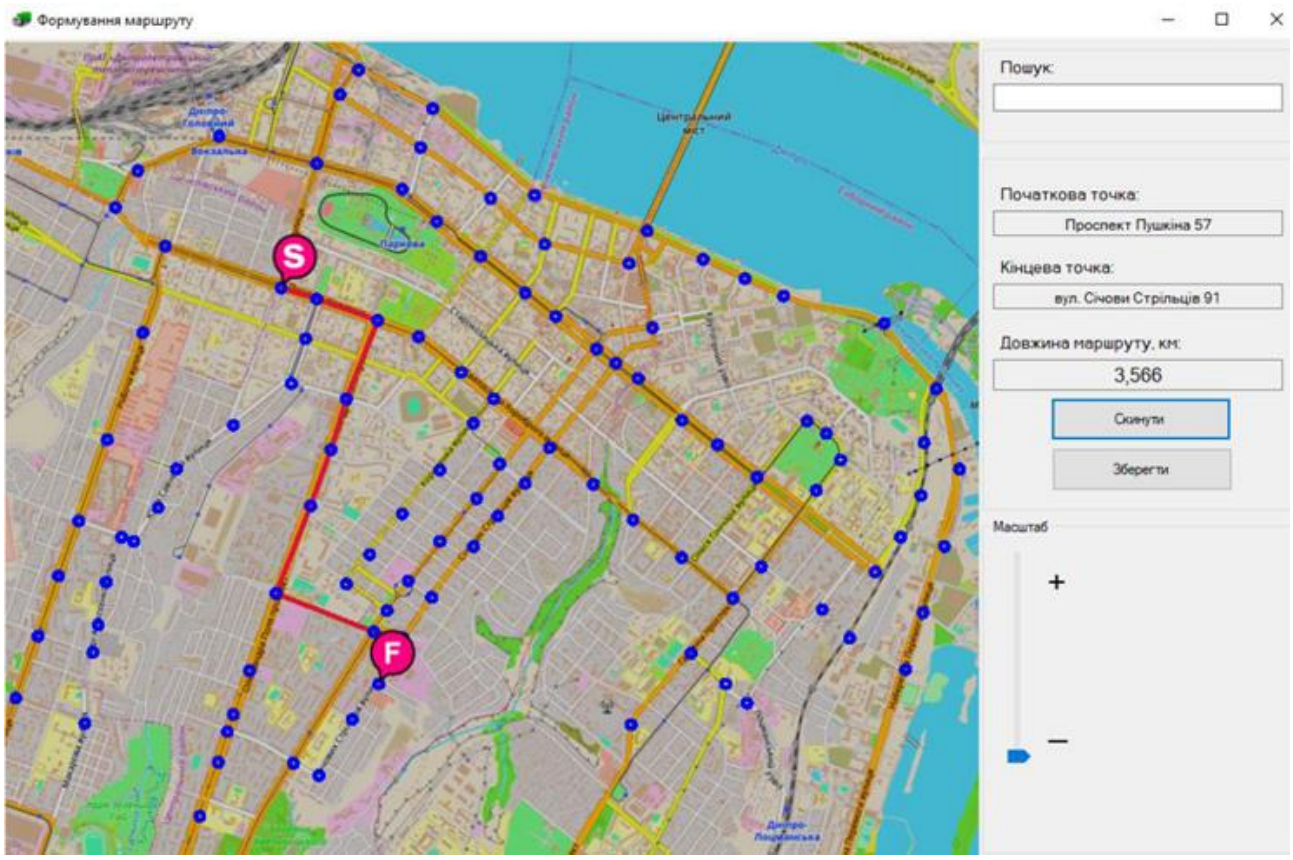


Рис. 2.48. Вибір двох точок на мапі

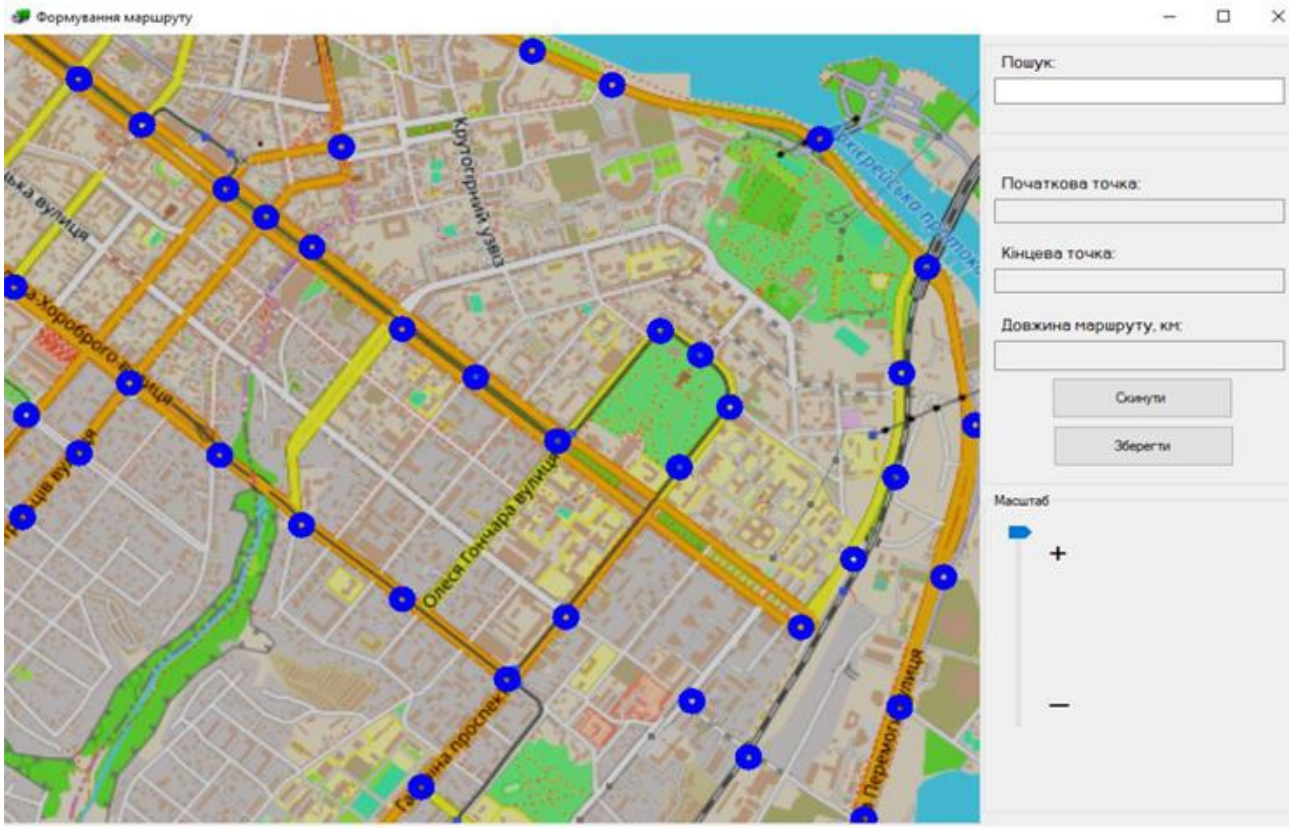


Рис. 2.49. Зміна масштабу мапи

Після вибору маршруту і натисканні на кнопки «Зберегти» вікно буде закрито і користувачу розблокується кнопка «Розрахувати» у головному вікні для створення маршрутного листа для підрахування витрат на поїздку і заповнення місця прибуття (рис. 2.50). Після цього залишиться заповнити «Відмітку про прибуття» за допомогою перимикачів зі стрілками та завершити формувати документ. Після натискання кнопки «Створити» відкриється вікно щойно створеного маршрутного листа (рис. 2.51).

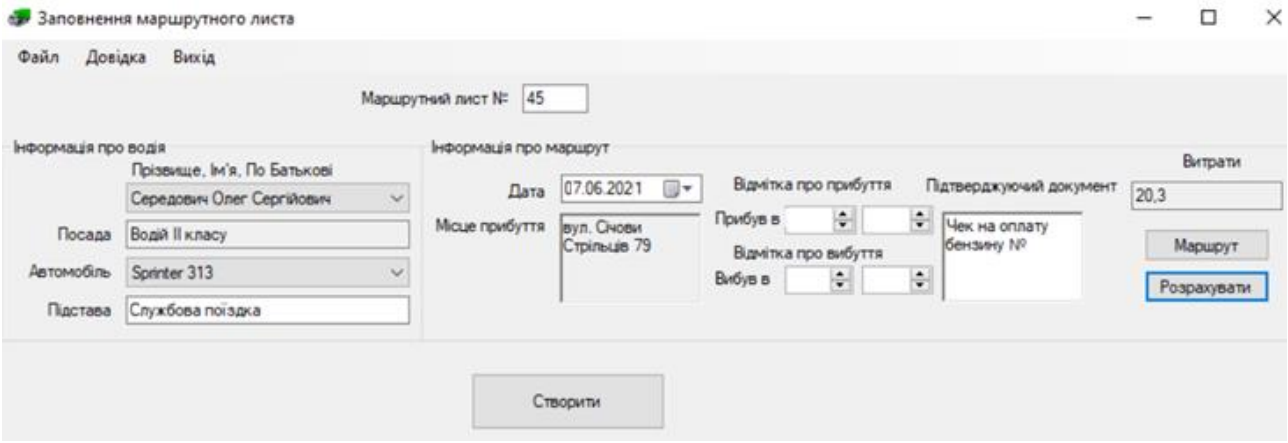


Рис. 2.50. Остаточний вигляд головного вікна для формування маршрутного листа

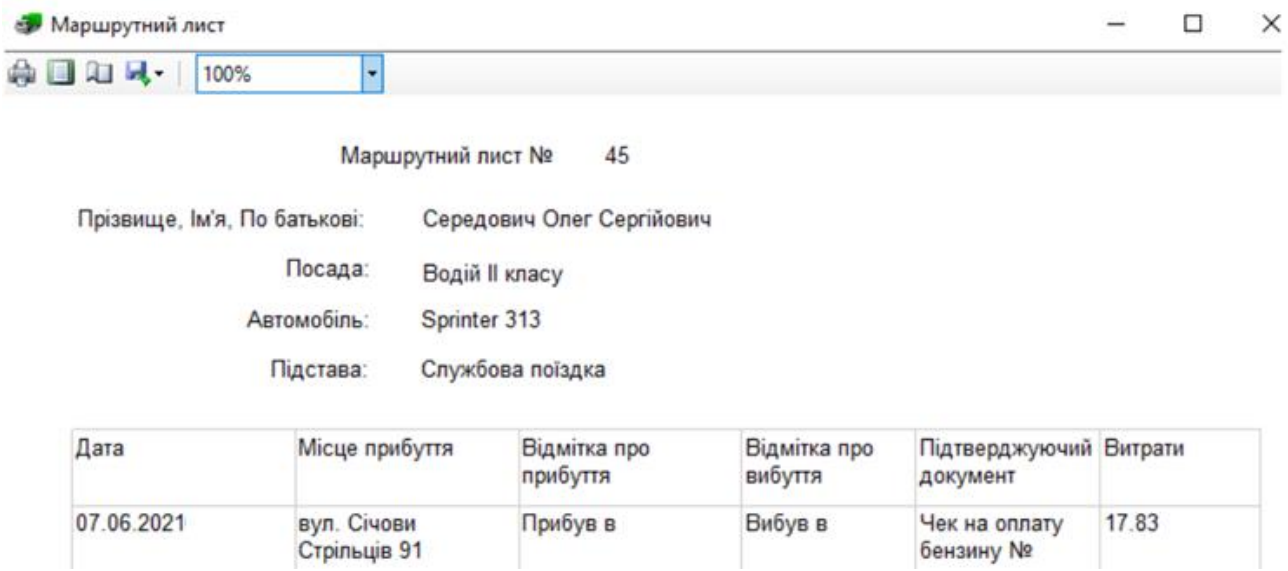
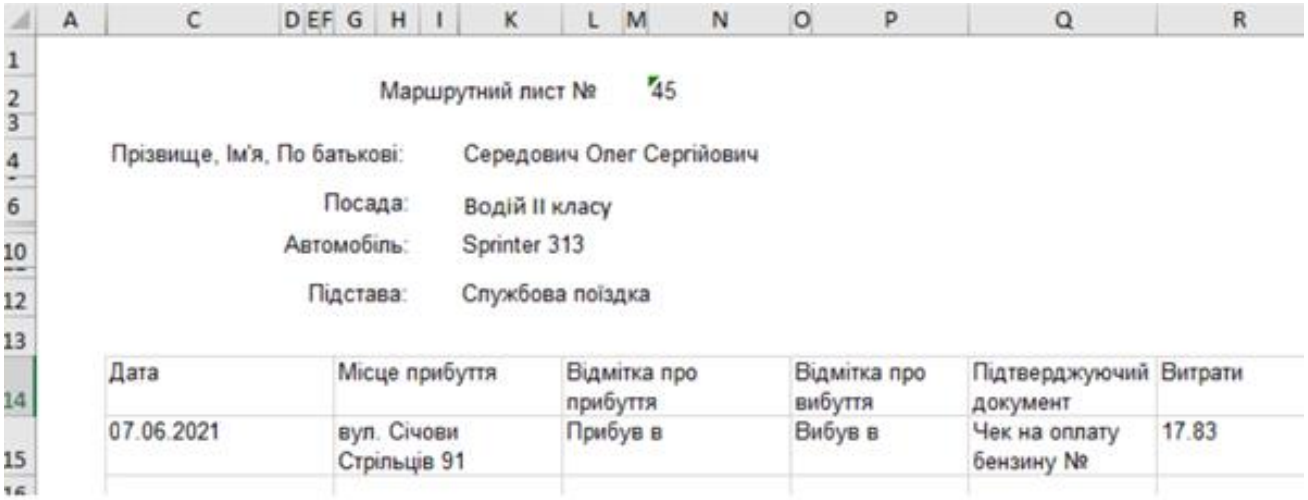


Рис. 2.51. Сформований маршрутний лист

В меню перегляду користувачу надається також можливість експорту документа в формати MS Excel або MSWord (рис. 2.52 та 2.53).



Маршрутний лист № 45

Прізвище, Ім'я, По батькові: Середович Олег Сергійович

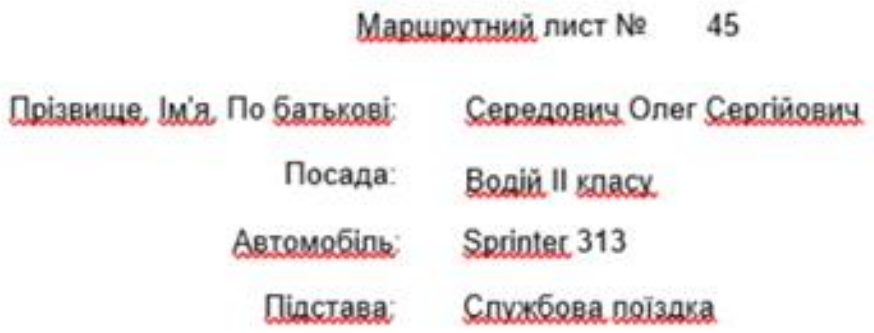
Посада: Водій II класу

Автомобіль: Sprinter 313

Підстава: Службова поїздка

Дата	Місце прибуття	Відмітка про прибуття	Відмітка про вибуття	Підтверджуючий документ	Витрати
07.06.2021	вул. Січови Стрільців 91	Прибув в	Вибув в	Чек на оплату бензину №	17.83

Рис. 2.52. Маршрутний лист експортований в MS Excel



Маршрутний лист № 45

Прізвище, Ім'я, По батькові: Середович Олег Сергійович

Посада: Водій II класу

Автомобіль: Sprinter 313

Підстава: Службова поїздка

Рис. 2.53. Маршрутний лист експортований в MS Word

Якщо після проходження авторизації користувачу надається роль бухгалтера, то він потрапляє у вікно формування маршрутного листа, який розглядався раніше (рис. 2.44).

Якщо після проходження авторизації користувачу надається роль водій, то він потрапляє у вікно перегляду маршрутного листа з можливістю перегляду мапи з сформованим найкоротшим маршрутом до місця слідування, який розглядався раніше (рис. 2.43).

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Вхідні дані:

- передбачуване число операторів 1000;
- коефіцієнт складності програми - 2;
- коефіцієнт корекції програми в ході її розробки - 0,08;
- годинна заробітна плата програміста, грн / год - 40.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_u + t_a + t_n + t_{отл} + t_д, \text{ людино-годин,} \quad (3.1)$$

де to - витрати праці на підготовку й опис поставленої задачі (приймається 50),

ти - витрати праці на дослідження алгоритму рішення задачі,

та - витрати праці на розробку блок-схеми алгоритму,

тп - витрати праці на програмування по готовій блок-схемі,

тотл - витрати праці на налагодження програми на ЕОМ,

тд - витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.:

- умовне число операторів (підпрограм):

$$Q = q \times C \times (1+ p) \text{ , людино-годин,} \qquad (3.2)$$

де q - передбачуване число операторів,

С - коефіцієнт складності програми,

р - коефіцієнт кореляції програми в ході її розробки.

$$Q = 1000 \cdot 2 \cdot (1 + 0{,}08) = 2808 \text{ людино-годин.}$$

Витрати праці на вивчення опису задачі ti визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{QB}{(75...85)K} \text{ , людино-годин,} \qquad (3.3)$$

де В - коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі; В=1.2 ... 1.5,

k - коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності.

Витрати праці на розробку алгоритму рішення задачі:

$$t_u = \frac{2808 \cdot 1{,}2}{80 \cdot 0{,}8} = 52 \text{ , людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t\alpha = \frac{Q}{(20...25)K} \quad \text{людино-годин.} \qquad (3.4)$$

$$t_a = \frac{2808}{20 \cdot 0{,}8} = 175 \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20...25)K} \quad \text{людино-годин.} \tag{3.5}$$

$$t_n = \frac{2808}{25 \cdot 0{,}8} = 140 \text{ людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

$$t_{отл} = \frac{Q}{(4...5)K} \quad \text{людино-годин.} \tag{3.6}$$

$$t_{отл} = \frac{2808}{4 \cdot 0{,}8} = 877 \text{ людино-годин.}$$

- за умови комплексного налагодження завдання:

$$t_{отл}^{к} = 1{,}2 \cdot t_{отл}; \tag{3.7}$$

$$t_{отл} = 877 * 1{,}2 = 1053$$

Витрати праці на підготовку документації:

$$t_д = t_{др} + t_{до}, \text{ людино-годин,} \tag{3.8}$$

де tдр - трудомісткість підготовки матеріалів і рукопису.

$$t_{др} = \frac{Q}{(15...20)K}, \text{ людино-годин.} \tag{3.9}$$

$$t_{др} = \frac{2808}{15 \cdot 0{,}8} = 234 \text{ людино-годин,}$$

tдо – трудомісткість редагування, печатки й оформлення документації

$$t_{до} = 0{,}75 \cdot t_{др}, \text{ людино-годин.} \tag{3.10}$$

$$tdo = 0{,}75 \cdot 234 = 175$$

$$t_д = 234 + 175 = 409 \text{ людино-годин.}$$

Отримуємо трудомісткість розробки програмного забезпечення:

$$t = 50 + 52 + 175 + 140 + 1053 + 409 = 1861 \text{ людино-годин.}$$

3.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ Кпо включають витрати на заробітну плату виконавця програми Зз/п і витрат машинного часу, необхідного на налагодження програми на ЕОМ:

$$K_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн,} \quad (3.11)$$

Заробітна плата виконавців визначається за формулою:

$$З_{ЗП} = t \cdot C_{ПР}, \text{ грн,} \quad (3.12)$$

де t - загальна трудомісткість, людино-годин,

Спр - середня годинна заробітна плата програміста, грн/година

$$З_{ЗП} = 1861 \cdot 30 = 43755 \text{ грн.}$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{МВ} = t_{отл} \times C_М, \text{ грн,} \quad (3.13)$$

де tотл - трудомісткість налагодження програми на ЕОМ, год.

Смч - вартість машино-години ЕОМ, 5 грн/год.

Визначені в такий спосіб витрати на створення програмного забезпечення є частиною одноразових капітальних витрат на створення АСУП.

$$З_{МВ} = 810 \times 5 = 4050 \text{ грн.}$$

$$Кпо = 43755 + 4050 = 47805 \text{ грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p} \text{ міс.} \qquad (3.13)$$

де Bk - число виконавців,

Fp - місячний фонд робочого часу (при 40 годинному робочому тижні Fp=176 годин).

$$T = \frac{2160}{1 \cdot 176} = 8{,}3 \text{ міс.}$$

Висновок: були проведені обчислення трудомісткості та витрат для розробки програмного забезпечення. Для створення даної інформаційної системи знадобиться 8,3 місяця, трудомісткість розробки ПЗ – 1861 людино-годин, а витрати на створення програмного забезпечення – 47805 грн.

ВИСНОВКИ

В ході роботи над кваліфікаційної роботи був розроблений програмний додаток призначений для формування маршрутного листа для автотранспортного підприємства з можливістю прокладання найкоротшого шляху до місця слідування водія.

Для розробки інформаційної системи для автоматизації формування маршрутного листа було використано середовище розробки Microsoft Visual Studio 2019, мову програмування C#, а для реалізації бази даних було обрано систему керування базою даних СКБД MS SQL Server.

Додаток написаний за допомогою методів об'єктно-орієнтованого програмування на мові C# і скомпільований з розширенням exe. Він працює у середовищі операційної системи Windows та Unix.

Для повноцінної роботи розробленого програмного додатку необхідно мати встановлений на комп'ютері текстовий редактор MS Word або електронний редактор таблиць MS Excel так як сформований маршрутний лист зберігається в файлах редактора.

Для формування найкоротшого маршруту слідування водія до місця призначений у програмному додатку використовується алгоритм Дейкстри.

Алгоритм Дейкстри дозволяє виконувати пошук найкоротших шляхів з однієї вершини до всіх інших вершин у графі. Даний алгоритм знаходить найкоротші шляхи до вершин графа в порядку їх віддаленості від вихідної вершини, тобто спочатку знаходиться найкоротший шлях від вихідної вершини до найближчої, потім до другої найближчої тощо.

Додаток має багатовіконний графічний інтерфейс та багаторівневий доступ (адміністратор, бухгалтер, водій). Водію доступна функція перегляду сформованого маршрутного листа та перегляду мапи з сформованим найкоротшим маршрутом слідування до визначеної адреси. Бухгалтеру надається можливість формувати маршрутний лист. Адміністратор має доступ до всіх функцій додатку, а саме робота з базою даних автотранспортного

підприємства, реєстрація нових користувачів та формування маршрутного листа.

Програмний додаток може бути застосований на невеликих автотранспортних підприємствах в якості допоміжного інструмента у роботі бухгалтера.

В економічному розділі були проведені обчислення трудомісткості та витрат для розробки програмного забезпечення. Для створення даної інформаційної системи знадобиться 8,3 місяця, трудомісткість розробки ПЗ – 1861 людино-годин, а витрати на її створення програмного забезпечення - 47805 грн.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Сафонов В.О. Возможности Visual Studio 2013, 2015 - 379с.
2. Ендрю Стіллмен Вивчаємо C#, 3-е видання, 2014 - 816с.
3. Робота з Excel в C# URL: http://wladm.narod.ru/C-Sharp/comexcel.html. дата звернення: 5.05.2021.
4. Робота з Word в C# URL: http://wladm.narod.ru/C-Sharp/comword.html#2. дата звернення: 5.05.2021.
5. Іцик Бен-Ган Microsoft SQL Server 2012. Основы T-SQL, 2015 - 400с.
6. Алгоритм Дейкстри URL: https://prog-cpp.ru/ дата звернення: 5.05.2021.
7. Ендрю Таненбаум Сучасні операційні системи, 4-е видання, 2014 - 1120с.
8. Системи робки тексту URL: http://www.shevch-enkove.org.ua/ дата звернення: 5.05.2021.
9. Особливості Microsoft Excel URL https://studo-pedia.su/542296osoblivosti-Microsoft--EXCEL.html.
10. Бібліотеки C# URL: https://docs.microsoft.com/ru-ru/dotnet/api/ дата звернення: 5.05.2021.
11. Діаграма послідовності URL: http://flash.retejo.inf/ дата звернення: 5.05.2021.
12. Бібліографічний запис. Бібліографічний опис. Загальні вимоги та правила складання : (ГОСТ 7.1-2003, IDT) : ДСТУ ГОСТ 7.1:2006. – Чинний з 2007–07–01. – К. : Держспоживстандарт України, 2007. – 47 с. – (Система стандартів з інформації, бібліотечної та видавничої справи) (Національний стандарт України).
13. Бусигін Б.С., Коротенко Г.М., Коротенко Л.М. Прикладна інформатика. Підручник для студентів комп'ютерних спеціальностей. –

Дніпропетровськ: Видавництво НГУ, 2004. – 559 с. URL: http://www.programmer.dp.ua/book-ua-k01.php. дата звернення: 15.03.2021.

14. Бусыгин Б.С., Дивизинюк М.М., Коротенко Г.М., Коротенко Л.М. Введение в современную информатику. Учебник. – Севастополь: Издательство СНУЯЭиП, 2005. – 644 с. / URL: http://www.programmer.dp.ua/book-ru-k02.php. дата звернення: 15.01.2018.

15. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення : ДСТУ 3008-95. – Чинний від 1996–01–01. – К. : Держстандарт України, 1996. – 39 с.

16. ДСТУ 2394-94 Інформація та документація. Комплектування фонду, бібліографічний опис, аналіз документів. Терміни та визначення. – Чинний від 01.01.1995. - Київ : Держстандарт України, 1994. – 88 с.

17. Методичні вказівки з виконання економічного розділу в дипломних проектах студентів спеціальності “Комп’ютерні системи ” / Укладачі О.Г. Вагонова, Нікітіна О.Б. Н.Н. Романюк – Дніпропетровськ: Національний гірничий університет. – 2013. – 23с.

18. Методичні рекомендації до виконання кваліфікаційних робіт бакалаврів напряму підготовки 122 «Комп’ютерні науки» галузі знань 12 Інформаційні технології/, Л.М. Коротенко , О.С. Шевцова; Нац. гірн. ун-т. – Д : ДВНЗ НГУ, 2019. – 65 с.

19. Методичні рекомендації щодо написання, оформлення та представлення учнівських науково-дослідницьких робіт учнів – членів Малої академії наук України / Г.Г. Півняк, Л.М. Коротенко, І.М. Удовик, Є.М. Головня – Д.: ДВНЗ «Національний гірничий університет», 2017. – 24 с.

20. Скорочення слів в українській мові у бібліографічному описі. Загальні правила та вимоги : ДСТУ 3582-97. – Чинний від 1998–07–01. – К. : Держстандарт України, 1998. – 24 с. – (Державний стандарт України).

ДОДАТОК А

КОД ПРОГРАМИ

Login.cs

```
public partial class Login : Form
{
const string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;At-
tachDbFilename=C:\Deicsra\WindowsFormDijkstra\TransportTest.mdf;Integrated Secu-rity =
True;";
SqlDataAdapter adapter = new SqlDataAdapter(); DataTable table = new DataTable(); public
Login()
{
InitializeComponent();
}
private void loginscreen()
{
if (textBox1.Text != "" && textBox2.Text != "")
{

string sql = "SELECT * FROM Users WHERE UserLogin=@UserLogin AND
UserPassword=@UserPassword";
try
{
using (SqlConnection connection = new SqlConnection(connection-
String))
{
SqlCommand cp = new SqlCommand(sql, connection);
cp.Parameters.AddWithValue("@UserLogin", textBox1.Text);
cp.Parameters.AddWithValue("@UserPassword",  text-
Box2.Text);
connection.Open();
SqlDataReader reader = cp.ExecuteReader();
if (reader.HasRows)
{
reader.Read();
if (reader.GetValue(3).ToString() == "Admin          ")
{
ListInfo mainf = new ListInfo();
Role.thisrole = "Admin";
connection.Close();
this.Hide();
mainf.ShowDialog();
}
else if (reader.GetValue(3).ToString() == "Driver ")
{
ListReport listrep = new ListReport();
Role.thisrole = "Driver";
connection.Close();
this.Hide();
listrep.ShowDialog();
}
```

```
else if (reader.GetValue(3).ToString() == "Booker  ")
{
CreateList mainf = new CreateList();
Role.thisrole = "Booker";
connection.Close();
this.Hide();
mainf.ShowDialog();
}

}
}
}
catch (Exception ex)
{
MessageBox.Show(ex.Message);
}
}
else MessageBox.Show("Заповніть всі поля");
}
private void checkBox1_CheckedChanged(object sender, EventArgs e)
{
if (checkBox1.Checked == true)
{
textBox2.PasswordChar = '\0';
}
else if (checkBox1.Checked == false)
{
textBox2.PasswordChar = '*';
}
}
private void button1_Click(object sender, EventArgs e)
{
loginscreen();
}
private void проПрограмуToolStripMenuItem_Click(object sender, EventArgs e) {
ProgramInfo prinf = new ProgramInfo();
prinf.ShowDialog();
}
private void label4_MouseHover(object sender, EventArgs e)
{
toolTip1.SetToolTip(label4, "Зверніться до свого системного адміністра-
тора");
}
private void Login_FormClosing(object sender, FormClosingEventArgs e) {
Application.Exit();
}
```

ListInfo.cs

```
public partial class ListInfo : Form
{
SqlConnection con = new SqlConnection(@"Data Source=(LocalDB)\MSSQLLo-
```

```
calDB; AttachDbFilename=C:\Deicsra\WindowsFormDijkstra\TransportTest.mdf;Inte-grated
Security=True;");
DataTable dtdriver;
SqlDataAdapter adptdriver;
DataTable dtcar;
SqlDataAdapter adptcar;
DataTable dtuser;
SqlDataAdapter adptuser;
public ListInfo()
{
InitializeComponent();
}
private void ListInfo_Load(object sender, EventArgs e)
{
datagrid();
}
void datagrid()
{
adptdriver = new SqlDataAdapter("select * from Driver", con); dtdriver = new DataTable();
adptcar = new SqlDataAdapter("select * from Car", con); dtcar = new DataTable();
adptuser = new SqlDataAdapter("select * from Users", con);
dtuser = new DataTable();
con.Open();
adptdriver.Fill(dtdriver);
dataGridView1.DataSource = dtdriver;
adptcar.Fill(dtcar);
dataGridView2.DataSource = dtcar;
adptuser.Fill(dtuser);
dataGridView3.DataSource = dtuser;
con.Close();
dataGridView1.Columns[0].HeaderText = "Код водія";
dataGridView1.Columns[1].HeaderText = "Прізвище, Ім'я, По Батькові";
dataGridView1.Columns[1].Width = 200;
dataGridView1.Columns[2].HeaderText = "Посада";
dataGridView2.Columns[0].HeaderText = "Код автомобіля";
dataGridView2.Columns[1].HeaderText = "Назва"; dataGridView2.Columns[2].HeaderText =
"Витрата палива";
dataGridView3.Columns[0].HeaderText = "Код користувача";
dataGridView3.Columns[1].HeaderText = "Логін"; dataGridView3.Columns[2].HeaderText =
"Пароль"; dataGridView3.Columns[3].HeaderText = "Роль";
}
private void видалитиToolStripMenuItem_Click(object sender, EventArgs e) {
String connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB; At-
tachDbFilename=C:\Deicsra\WindowsFormDijkstra\TransportTest.mdf;Integrated Secu-
rity=True;Connect Timeout=30";
using (SqlConnection conn = new SqlConnection(connectionString))
{
switch (tabControl1.SelectedIndex)
{
case 0:
SqlCommand com1 = new SqlCommand("DELETE FROM Driver where
```

```
DriverCode=@id", conn);
int id1 = int.Parse(dataGridView1.Curren-tRow.Cells[0].Value.ToString());
com1.Parameters.AddWithValue("@id", id1);
conn.Open(); //Открываем подключение
try
{
DialogResult result = MessageBox.Show("Ви впевнені?",
"Повідомлення",
MessageBoxButtons.OKCancel, MessageBoxIcon.Information)
if (result == DialogResult.OK)
{
com1.ExecuteNonQuery();
datagrid();
}
conn.Close();
}
catch
{
MessageBox.Show("Удалить не удалось!");
con.Close();
}
break;
case 1:
SqlCommand com2 = new SqlCommand("DELETE FROM Car where CarCode=@id", conn);
int id2 = int.Parse(dataGridView2.Curren-tRow.Cells[0].Value.ToString());
com2.Parameters.AddWithValue("@id", id2);
conn.Open(); //Открываем подключение
try
{
DialogResult result = MessageBox.Show("Ви впевнені?",
"Повідомлення",
MessageBoxButtons.OKCancel, MessageBoxIcon.Information); if (result == DialogResult.OK)
{
com2.ExecuteNonQuery();
datagrid();
}
conn.Close();
}
catch
{
MessageBox.Show("Удалить не удалось!");
con.Close();
}
break;
case 2:
SqlCommand com3 = new SqlCommand("DELETE FROM Users where
UserCode=@id", conn);
int id3 = int.Parse(dataGridView3.Curren-tRow.Cells[0].Value.ToString());
com3.Parameters.AddWithValue("@id", id3);
conn.Open(); //Открываем подключение
try
{
```

```
DialogResult result = MessageBox.Show(
"Ви впевнені?",
"Повідомлення",
MessageBoxButtons.OKCancel, MessageBoxIcon.Information);
if (result == DialogResult.OK)
{
com3.ExecuteNonQuery();
datagrid();
}
conn.Close();
}
catch
{
MessageBox.Show("Удалить не удалось!");
con.Close();
}
break;
}
}
}
private void додатиToolStripMenuItem1_Click(object sender, EventArgs e) {
switch (tabControl1.SelectedIndex)
{
case 0:
UpdateBd.add = true;
AddDriver addrive = new AddDriver();
addrive.ShowDialog(); break;
case 1:
UpdateBd.add = true;
AddCar adcar = new AddCar();
adcar.ShowDialog(); break;
case 2:
UpdateBd.add = true;
AddUser aduser = new AddUser();
aduser.ShowDialog(); break;
}
}
private void вToolStripMenuItem_Click(object sender, EventArgs e)
{
Application.Exit();
}
private void вихідЗАккаунтуToolStripMenuItem_Click(object sender, EventArgs
e)
{
this.Hide();
Login lo = new Login();
lo.Show();
}
private void створитиМаршртунийЛистToolStripMenuItem_Click(object sender, EventArgs e)
{
this.Hide();
CreateList crelist = new CreateList();
```

```
crelist.Show();
}
private void переглянутиОстанніймаршрутнийЛистToolStripMenuItem_Click(ob-ject sender,
EventArgs e)
{
ListReport lisrep = new ListReport();
lisrep.ShowDialog();
}
private void редагуватиToolStripMenuItem_Click(object sender, EventArgs e) {
string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;At-
tachDbFilename=C:\Deicsra\WindowsFormDijkstra\TransportTest.mdf;Integrated Secu-
rity=True;Connect Timeout=30";
using (SqlConnection conn = new SqlConnection(connectionString))
{
switch (tabControl1.SelectedIndex)
{
case 0:
BdData.id = int.Parse(dataGridView1.Curren-tRow.Cells[0].Value.ToString());
BdData.pole1 = dataGridView1.Curren-tRow.Cells[1].Value.ToString();
BdData.pole2 = dataGridView1.Curren-tRow.Cells[2].Value.ToString();
UpdateBd.add = false;
AddDriver addrive = new AddDriver();
addrive.ShowDialog();
break;
case 1:
BdData.id = int.Parse(dataGridView2.Curren-tRow.Cells[0].Value.ToString());
BdData.pole1 = dataGridView2.Curren-tRow.Cells[1].Value.ToString();
BdData.pole2 = dataGridView2.Curren-tRow.Cells[2].Value.ToString();
UpdateBd.add = false;
AddCar adcar = new AddCar();
adcar.ShowDialog();
break;
case 2:
BdData.id = int.Parse(dataGridView3.Curren-tRow.Cells[0].Value.ToString());
BdData.pole1 = dataGridView3.Curren-tRow.Cells[1].Value.ToString();
BdData.pole2 = dataGridView3.Curren-tRow.Cells[2].Value.ToString();
BdData.pole3 = dataGridView3.Curren-tRow.Cells[3].Value.ToString();
UpdateBd.add = false;
AddUser aduser = new AddUser();
aduser.ShowDialog();
break;
}
}
}
private void ListInfo_Activated(object sender, EventArgs e)
{
datagrid();
}
private void проПрограмуToolStripMenuItem_Click(object sender, EventArgs e) {
ProgramInfo proginf = new ProgramInfo();
proginf.ShowDialog();
}
```

```
private void довідкаToolStripMenuItem_Click(object sender, EventArgs e) {
Spravka sprv = new Spravka();
sprv.ShowDialog();
}
}

AddCar.cs
public partial class AddCar : Form
{
SqlConnection con = new SqlConnection(@"Data Source=(LocalDB)\MSSQLLo-
calDB;AttachDbFilename=C:\Deicsra\WindowsFormDijkstra\TransportTest.mdf;Integrated
Security=True;Connect Timeout=30");
public AddCar()
{
InitializeComponent();
}
private void button1_Click(object sender, EventArgs e)
{
switch (UpdateBd.add)
{
case true: //добавление
con.Open();
if (textBox1.Text != "" || textBox2.Text != "")
{
try
{
using (SqlCommand command = new SqlCommand( "INSERT INTO Car VALUES(@Carname,
@Caravg)", con)) {
command.Parameters.Add(new SqlParameter("Carname", text-
Box1.Text));
command.Parameters.Add(new SqlParameter("Caravg", text-
Box2.Text));
command.ExecuteNonQuery();
this.Close();
}
}
catch
{
Console.WriteLine("Count not insert.");
}
else if (textBox1.Text == "" || textBox2.Text == "")
{
MessageBox.Show("Заполните все поля");
}
con.Close(); break;
case false: //обновление
con.Open();
if (textBox1.Text != "" || textBox2.Text != "")
{
using (SqlCommand command = new SqlCommand(
"Update Car SET CarName = @Carname, CarAVG = @Caravg Where CarCode = @Carcode",
con))
```

```
{
command.Parameters.AddWithValue("@Carcode",
BdData.id);
command.Parameters.AddWithValue("@Carname",          text-
Box1.Text);
command.Parameters.AddWithValue("@Caravg",  int.Parse(text-
Box2.Text));
command.ExecuteNonQuery();
this.Close();
}
}
else if (textBox1.Text == "" || textBox2.Text == "")
{
MessageBox.Show("Заполните все поля");
}
con.Close(); break;
}
}
private void AddCar_Load(object sender, EventArgs e)
{
switch (UpdateBd.add)
{
case true: button1.Text = "Додати";
this.Text = "Додавання даних";
break;
case false:
this.Text = "Редагування даних";
button1.Text = "Редагувати";
textBox1.Text = BdData.pole1;
textBox2.Text = BdData.pole2;
break;
}
}
private void textBox1_KeyPress(object sender, KeyPressEventArgs e)
{
if (!Char.IsDigit(e.KeyChar))
{
return;
}
else
{
toolTip1.Show("Ввод цифр заборонений", textBox1); e.Handled = true;
}
}
private void textBox2_KeyPress(object sender, KeyPressEventArgs e)
{
// ввод в texBox только цифр и кнопки Backspace char ch = e.KeyChar;
if (!Char.IsDigit(ch) && ch != 8)
{
toolTip1.Show("Ввод букв заборонений", textBox2); e.Handled = true;
}
}
```

```
}

AddDriver.cs
public partial class AddDriver : Form
{
SqlConnection con = new SqlConnection(@"Data Source=(LocalDB)\MSSQLLo-
calDB;AttachDbFilename=C:\Deicsra\WindowsFormDijkstra\TransportTest.mdf;Integrated
Security=True;Connect Timeout=30");
public AddDriver()
{
InitializeComponent();
}
private void button1_Click(object sender, EventArgs e)
{
switch (UpdateBd.add)
{
case true: //добавление
con.Open();
if (textBox1.Text != "" || comboBox1.Text != "")
{
try
{
using (SqlCommand command = new SqlCommand(
"INSERT INTO Driver VALUES(@DriverFi, @Driverpos)",
con))
{
command.Parameters.Add(new SqlParameter("Driv-
erFi", textBox1.Text));
command.Parameters.Add(new SqlParameter("Driv-
erpos", comboBox1.Text));
command.ExecuteNonQuery();
}
this.Close();
}
catch
{
Console.WriteLine("Count not insert.");
}
}
else if (textBox1.Text == "" || comboBox1.Text == "")
{
MessageBox.Show("Заполните все поля");
}
con.Close(); break;
case false: //обновление
con.Open();
if (textBox1.Text != "" || comboBox1.Text != "")
{
try
{
using (SqlCommand command = new SqlCommand(
"Update Driver SET DriverFIO = @driverfio, DriverPosition
```

```
= @driverpos Where DriverCode = @drivecod", con))
{
command.Parameters.AddWithValue("@drivecod",
BdData.id);
command.Parameters.AddWithValue("@driverfio",          text-
Box1.Text);
command.Parameters.AddWithValue("@driverpos",          com-
boBox1.Text);
command.ExecuteNonQuery();
this.Close();
}
}
catch
{
Console.WriteLine("Count not insert.");
}
}
else if (textBox1.Text == "" || comboBox1.Text == "")
{
MessageBox.Show("Заполните все поля");
}
con.Close(); break;
}
}
private void AddDriver_Load(object sender, EventArgs e)
{
switch (UpdateBd.add)
{
case true: button1.Text = "Додати";
this.Text = "Додавання даних";
break;
case false:
this.Text = "Редагування даних";
button1.Text = "Редагувати";
textBox1.Text = BdData.pole1;
comboBox1.Text = BdData.pole2;
break;
}
}
private void textBox1_KeyPress(object sender, KeyPressEventArgs e)
{
if (!Char.IsDigit(e.KeyChar))
{
return;
}
else
{

toolTip1.Show("Ввод цифр заборонений", textBox1); e.Handled = true;
}
}
}
```

AddUser.cs

```
public partial class AddUser : Form
{
SqlConnection con = new SqlConnection(@"Data Source=(LocalDB)\MSSQLLo-
calDB;AttachDbFilename=C:\Deicsra\WindowsFormDijkstra\TransportTest.mdf;Integrated
Security=True;Connect Timeout=30");
public AddUser()
{
InitializeComponent();
}
private void button1_Click(object sender, EventArgs e)
{
switch (UpdateBd.add)
{
case true: //добавление
con.Open();
if (textBox1.Text != "" || textBox2.Text != "" || comboBox1.Text
!= "")
{
try
{
using (SqlCommand command = new SqlCommand(
"INSERT INTO Users VALUES(@Userlogin, @UserPassword,
@Userrole)", con))
{
command.Parameters.Add(new SqlParameter("Userlogin", textBox1.Text));
command.Parameters.Add(new SqlParameter("UserPassword", textBox2.Text));
command.Parameters.Add(new SqlParameter("Userrole", comboBox1.Text));
command.ExecuteNonQuery();
}
this.Close();
}
catch
{
Console.WriteLine("Count not insert.");
}
}
else if (textBox1.Text == "" || comboBox1.Text == "")
{
MessageBox.Show("Заполните все поля");
}
con.Close(); break;
case false: //обновление
con.Open();
if (textBox1.Text != "" || textBox2.Text != "" || comboBox1.Text
!= "")
{
try
{
using (SqlCommand command = new SqlCommand(
"Update Users SET UserLogin = @userlog, UserPassword =
```

```
@userpass, UserRole = @userrol Where UserCode = @usercode", con))
{
command.Parameters.AddWithValue("@usercode", BdData.id);
command.Parameters.AddWithValue("@userlog",textBox1.Text);
command.Parameters.AddWithValue("@userpass",textBox2.Text);
command.Parameters.AddWithValue("@userrol",comboBox1.Text);
command.ExecuteNonQuery();
this.Close();
}
}
catch
{
Console.WriteLine("Count not insert.");
}
}
else if (textBox1.Text == "" || comboBox1.Text == "")
{
MessageBox.Show("Заполните все поля");
}
con.Close(); break;
}
}
private void AddUser_Load(object sender, EventArgs e)
{
switch (UpdateBd.add)
{
case true:
button1.Text = "Додати";
this.Text = "Додавання даних"; break;
case false:
button1.Text = "Редагувати";
this.Text = "Редагування даних ";
textBox1.Text = BdData.pole1;
textBox2.Text = BdData.pole2;
comboBox1.Text = BdData.pole3;
break;
}
}
}
```

CreateList.cs

```
public partial class CreateList : Form
{
ToolStripLabel dateLabel;
ToolStripLabel timeLabel;
ToolStripLabel infoLabel;
Timer timer;
double Fuel;
double price = 20;
SqlConnection con = new SqlConnection(@"Data Source=(LocalDB)\MSSQLLocalDB
;AttachDbFilename=C:\Deicsra\WindowsFormDijkstra\TransportTest.mdf;Integrated Se-
curity=True;Connect Timeout=30");
```

```
SqlCommand cmdDataBase;
SqlCommand cmdDriver;
public CreateList()
{
InitializeComponent();
}
private void вихідToolStripMenuItem_Click(object sender, EventArgs e) {
Application.Exit();
}
private void button1_Click(object sender, EventArgs e)
{
button3.Enabled = true;
Form1 fm1 = new Form1();

fm1.ShowDialog();
}
void carload()
{
comboBox2.Items.Clear();
con.Open();
cmdDataBase = con.CreateCommand(); cmdDataBase.CommandType = CommandType.Text;
cmdDataBase.CommandText = "SELECT CarName FROM Car";
cmdDataBase.ExecuteNonQuery(); DataTable dt = new DataTable();
SqlDataAdapter da = new SqlDataAdapter(cmdDataBase); da.Fill(dt);
foreach (DataRow dr in dt.Rows)
{
comboBox2.Items.Add(dr["CarName"].ToString());
}
con.Close();
}
void DriverLoad()
{
comboBox1.Items.Clear();
con.Open();
cmdDataBase = con.CreateCommand(); cmdDataBase.CommandType = CommandType.Text;
cmdDataBase.CommandText = "SELECT DriverFIO FROM Driver";
cmdDataBase.ExecuteNonQuery(); DataTable dt = new DataTable();
SqlDataAdapter da = new SqlDataAdapter(cmdDataBase); da.Fill(dt);
foreach (DataRow dr in dt.Rows)
{
comboBox1.Items.Add(dr["DriverFIO"].ToString());
}
con.Close();
}
private void comboBox2_SelectedIndexChanged(object sender, EventArgs e) {

button1.Enabled = true;
cmdDataBase = new SqlCommand("select * from Car where CarName = N'" + comboBox2.Text +
" ' ", con);con.Open();
cmdDataBase.ExecuteNonQuery();
SqlDataReader myReader;
myReader = cmdDataBase.ExecuteReader();
```

```
while (myReader.Read())
{
string sPrice = (string)myReader["CarAVG"].ToString(); Fuel = double.Parse(sPrice);
}
con.Close();
}
private void останнійМаршрутнийЛистToolStripMenuItem_Click(object sender, EventArgs e)
{
ListReport lisrep = new ListReport();
lisrep.ShowDialog();
}
private void timer1_Tick(object sender, EventArgs e)
{
dateLabel.Text = DateTime.Now.ToLongDateString(); timeLabel.Text =
DateTime.Now.ToLongTimeString();
}
private void statustime()
{
infoLabel = new ToolStripLabel();
infoLabel.Text = "Поточні дата і час:";
dateLabel = new ToolStripLabel();
timeLabel = new ToolStripLabel();
statusStrip1.Items.Add(infoLabel);
statusStrip1.Items.Add(dateLabel);
statusStrip1.Items.Add(timeLabel);
timer = new Timer() { Interval = 1000 };
timer.Tick += timer1_Tick;
timer.Start();
}
private void comboBox1_SelectedIndexChanged(object sender, EventArgs e) {
cmdDriver = new SqlCommand("select * from Driver where DriverFIO = N'"
+ comboBox1.Text + " ' ", con); con.Open(); cmdDriver.ExecuteNonQuery(); SqlDataReader
myReader;
myReader = cmdDriver.ExecuteReader(); while (myReader.Read())
{
string typeclass = (string)myReader["DriverPosition"].ToString(); textBox2.Text = typeclass;
}
con.Close();
}
private void CreateList_Load(object sender, EventArgs e)
{
carload();
DriverLoad();
statustime();
}
private void button2_Click(object sender, EventArgs e)
{
string sqlExpression = "DELETE FROM LastList"; string up, down;
up = label12.Text + " " + domainUpDown1.Text + " " + domainUpDown2.Text; down =
label13.Text + " " + domainUpDown4.Text + " " + do-
mainUpDown3.Text;
con.Open();
```

```
SqlCommand commanddel = new SqlCommand(sqlExpression, con);
commanddel.ExecuteNonQuery();
con.Close();
con.Open();
if (textBox1.Text != "" || comboBox1.Text != "" || textBox2.Text != "" || comboBox2.Text != "" ||
textBox4.Text != "" || richTextBox1.Text != "" || richTextBox2.Text != "" || textBox7.Text != "")
{
try
{
using (SqlCommand command = new SqlCommand(
"INSERT INTO LastList VALUES(@number, @fio, @class, @auto, @os-nova, @data, @city,
@up, @down, @doc, @price)", con))
{
command.Parameters.Add(new SqlParameter("number", textBox1.Text));
command.Parameters.Add(new SqlParameter("fio", comboBox1.Text));
command.Parameters.Add(new SqlParameter("class", textBox2.Text));
command.Parameters.Add(new SqlParameter("auto", comboBox2.Text));
command.Parameters.Add(new SqlParameter("osnova", textBox4.Text));
command.Parameters.Add(new SqlParameter("data", dateTimePicker1.Text));
command.Parameters.Add(new SqlParameter("city", richTextBox1.Text));
command.Parameters.Add(new SqlParameter("up", up)); command.Parameters.Add(new
SqlParameter("down", down)); command.Parameters.Add(new SqlParameter("doc",
richTextBox2.Text)); command.Parameters.Add(new SqlParameter("price", textBox7.Text));
command.ExecuteNonQuery();
con.Close();
}
MessageBox.Show("Додано");
}
catch (Exception ex)
{
con.Close();
Console.WriteLine(ex.Message);
}
}
else
{
MessageBox.Show("Заполните все поля");
con.Close();
}
}
private void button3_Click(object sender, EventArgs e)

{
textBox7.Text = Convert.ToString(((Fuel * DataBank.Rout) / 100) *
price);
richTextBox1.Text = DataBank.Street;
button1.Enabled = false;
button2.Enabled = true;
}
private void вихідToolStripMenuItem_Click_1(object sender, EventArgs e) {
}
private void справкаToolStripMenuItem_Click_1(object sender, EventArgs e) {
```

```
Spravka spra = new Spravka();
spra.ShowDialog();
}
private void timer1_Tick_1(object sender, EventArgs e)
{
dateLabel.Text = DateTime.Now.ToLongDateString(); timeLabel.Text =
DateTime.Now.ToLongTimeString();
}
private void вихід3ПрограмиToolStripMenuItem_Click(object sender, EventArgs
e)
{
Application.Exit();
}
private void вихід3АккаунтуToolStripMenuItem_Click(object sender, EventArgs
e)
{
this.Close();
Login lo = new Login();
lo.Show();
}
}
```

MapForm.cs

```
public partial class Form1 : Form
{
public Form1()
{
InitializeComponent();
// textBoxFind.KeyPreview = true;
// включить двойную буфферизацию , чтобы карта не мерцала
typeof(Panel).InvokeMember("DoubleBuffered", System.Reflection.Bind-
ingFlags.SetProperty |
System.Reflection.BindingFlags.Instance | System.Reflection.Bind-ingFlags.NonPublic, null,
canvas,
new object[] { true });
if (!Directory.Exists(@"Data"))
{
MessageBox.Show("Не найдена папка с данными для графа и изображени-
ями",
"Error", MessageBoxButtons.OK, MessageBoxIcon.Error); return;
}
if (!File.Exists(@"Data\mapimage.PNG"))
{
MessageBox.Show("Не найдено изображение для карты", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
bitmap = new Bitmap(50, 100);
}
else
// карта
bitmap = (Bitmap)Image.FromFile(@"Data\mapimage.PNG");
if (!File.Exists(@"Data\toolStart.PNG"))
{
```

```
MessageBox.Show("Не найдено изображение начальной точки", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
startPointBitmap = new Bitmap(50, 100);
}
else
// изображение начальной точки
startPointBitmap = (Bitmap)Image.FromFile(@"Data\toolStart.PNG");
if (!File.Exists(@"Data\toolEnd.PNG"))
{
MessageBox.Show("Не найдено изображение конечной точки", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
endPointBitmap = new Bitmap(50, 100);
}
else
// изображение конечной точки
endPointBitmap = (Bitmap)Image.FromFile(@"Data\toolEnd.PNG");
canvas.BackgroundImage = bitmap; canvas.BackgroundImageLayout = ImageLayout.Zoom;
//
scaleCoef = trackBar1.Value / 100.0f + MAGIC_COEF; canvas.Width = (int)(bitmap.Width *
scaleCoef); canvas.Height = (int)(bitmap.Height * scaleCoef) + 30;
panelRight.Left = (int)(bitmap.Width * START_COEF);
panelRight.Width = this.ClientRectangle.Width - (int)(bitmap.Width *
START_COEF);
panelBottom.Top = (int)(bitmap.Height * START_COEF); panelBottom.Height =
this.ClientRectangle.Height - (int)(bitmap.Height
* START_COEF);
trackBar1_Scroll(null, new EventArgs());
if (!File.Exists(@"Data\graph.txt"))
{
MessageBox.Show("Не найден файл с данными графа", "Error",MessageBoxButtons.OK,
MessageBoxIcon.Error);
}
else
// при открытии окна сразу загружаются данные
LoadGraphFromFile(@"Data\graph.txt");
comboBox1.DisplayMember = "Address";
this.textBoxFind.AutoSize = false;
this.textBoxFind.Height = 23;
}
bool flag = true;
const float START_COEF = 0.85f;
const float MAGIC_COEF = 0.75f;
Bitmap bitmap = null;
Bitmap startPointBitmap = null;
Bitmap endPointBitmap = null;
Point? startPoint = null, endPoint = null;
float scaleCoef = 0;
bool mouseDown = false;
Point mousePos;
Graph graph = new Graph();
List<Line> lines = new List<Line>();
void Reset()
```

```
{
startPoint = null;
endPoint = null;
// обнулить линии пути
lines = new List<Line>();
// очистить текстовые поля textBoxFind.Clear(); textBoxStartAddress.Clear();
textBoxEndAddress.Clear(); textBox1.Clear(); canvas.Invalidate();
}
// загрузить граф из файла
void LoadGraphFromFile(string path)
{
using (StreamReader reader = new StreamReader(path))
{
string line = string.Empty;
// загрузить вершины графа while (true)
{
// чтение строки файла
line = reader.ReadLine();
if (line == null || line == "Edges") break;
// выделить из строки
// индекс вершины и координаты
string[] values = line.Split(new char[] { ' ' }, StringSplitOp-tions.RemoveEmptyEntries);
int index = Int32.Parse(values[0]);
int x = Int32.Parse(values[1]);
int y = Int32.Parse(values[2]);
graph.Nodes.Add(new Node(index, new Point(x, y), string.Join(" ", values.Skip(3))));
//Console.WriteLine(line);
}
if (string.IsNullOrEmpty(line) || line != "Edges")
throw new FileFormatException();
while (true)
{
// чтение строки файла
line = reader.ReadLine();
if (string.IsNullOrEmpty(line)) break;
// выделить из строки индексы первой и второй вершины
// и расстояние между вершинами
string[] values = line.Split(new char[] { ' ', '\t' }, String-SplitOptions.RemoveEmptyEntries);
int a = Int32.Parse(values[0]);
int b = Int32.Parse(values[1]);
double c = double.Parse(values[2], CultureInfo.InvariantCul-
ture);
// получить по индексу вершины из графа
int index1 = graph.Nodes.FindIndex(item => item.Id == a);
if (index1 == -1)
throw new FileFormatException("Node not found.");
int index2 = graph.Nodes.FindIndex(item => item.Id == b);
if (index2 == -1)
throw new FileFormatException("Node not found.");
// добавить ребра вершинам graph.Nodes[index1].AddEdge(graph.Nodes[index2], c);
graph.Nodes[index2].AddEdge(graph.Nodes[index1], c);
//Console.WriteLine(line);
```

```
}
}
}
void DijkstrasTest()
{
for (int i = 0; i < graph.Nodes.Count; i++)
{
// отметить все вершины как непосещенные graph.Nodes[i].Visited = false;
graph.Nodes[i].DistanceFromStart = double.MaxValue;
}
// найти стартовую вершину в графе
int index1 = graph.Nodes.FindIndex(n => n.Point == startPoint); if (index1 == -1)
throw new KeyNotFoundException();
//найти конечную вершину в графе
int index2 = graph.Nodes.FindIndex(n => n.Point == endPoint); if (index2 == -1)
throw new KeyNotFoundException();
Node startNode = graph.Nodes[index1];
Node endNode = graph.Nodes[index2];
graph.SetAllSums(startNode);
textBox1.Text = graph.GetShortestLength(endNode).ToString();
// создание линий для отрисовки кратчайшего пути
CreateShortestRouteTo(startNode, endNode);
}
// Формирование пути
void CreateShortestRouteTo(Node startVertex, Node endVertex)
{
if (startVertex == null || endVertex == null)
return;
lines = new List<Line>();
var prev = endVertex;
Point point2;
while (prev != null && prev != startVertex)
{
Point point = prev.Point;
if (prev.Previous != null)
{
point2 = prev.Previous.Point;
lines.Add(new Line { start = new Point(point.X, point.Y), end = new Point(point2.X, point2.Y) });

}
prev = prev.Previous; // перейти к предыдущему узлу
}
}
// выбор строки в комбобоксе
private void comboBox1_SelectedIndexChanged(object sender, EventArgs e) {
ComboBox cmb = (ComboBox)sender;
flag = false;
if (cmb.SelectedItem != null)
{
// обнулить линии пути
lines = new List<Line>();
// очистить текстовые поля textBoxFind.Clear(); textBox1.Clear(); canvas.Invalidate();
```

```
// Reset();
Node selectedNode = (Node)cmb.SelectedItem; textBoxFind.Text = selectedNode.Address;
if (startPoint == null)
{
startPoint = selectedNode.Point; textBoxStartAddress.Text = selectedNode.Address;
}
else if (endPoint == null && selectedNode.Point != startPoint)
{
endPoint = selectedNode.Point; textBoxEndAddress.Text = selectedNode.Address;
}
if(startPoint != null && endPoint != null)
DijkstrasTest();
canvas.Invalidate();
}
comboBox1.DroppedDown = false;
comboBox1.Items.Clear();
}
private void TextBoxFind_TextChanged(object sender, EventArgs e)
{
if(flag)
FindItemContaining(graph.Nodes, textBoxFind.Text);
}
// Select an item containing the target string.
private void FindItemContaining(List<Node> items, string target)
{
target = target.ToLower().Trim();
List<Node> list = new List<Node>();
foreach (Node item in items)
{
if (target.Length > 1 && item.Address.ToLower().Trim().Con-
tains(target))
list.Add(item);
}
if(list.Count > 0)
{
// показываем комбобокс comboBox1.DroppedDown = true;
}

else // скрываем комбобокс
comboBox1.DroppedDown = false;
comboBox1.Items.Clear();
//Use AddRange to add the list. ToArray() is used to convert List<>
to string[]
comboBox1.Items.AddRange(list.ToArray());
// Return null;
// return null;
}
// обработчик рисования на панели для отрисовки карты private void panel1_Paint(object
sender, PaintEventArgs e)
{
// нарисовать кратчайший путь
using (Pen linePen = new Pen(Color.Crimson, 5 * scaleCoef))
```

```
{
foreach (var item in lines)
{
e.Graphics.DrawLine(linePen, item.start.X * scaleCoef, item.start.Y * scaleCoef, item.end.X *
scaleCoef, item.end.Y * scaleCoef);
}
}
// нарисовать все вершины графа
using (Pen pen = new Pen(Color.Blue, 5 * scaleCoef))
{
foreach (var item in graph.Nodes)
{
Point nodePoint = item.Point;
e.Graphics.DrawEllipse(pen, (nodePoint.X - 4) * scaleCoef, (no-dePoint.Y - 4) * scaleCoef, 8 *
scaleCoef, 8 * scaleCoef);
}
}
// нарисовать начальную точку if (startPoint != null)
{
// смещение координат int xOffset = 8;
int yOffset = 46;
e.Graphics.DrawImage(startPointBitmap, startPoint.Value.X * scaleCoef - xOffset,
startPoint.Value.Y * scaleCoef - yOffset);
}
// нарисовать конечную точку if (endPoint != null)
{
// смещение координат int xOffset = 8;
int yOffset = 46;
e.Graphics.DrawImage(endPointBitmap, endPoint.Value.X * scaleCoef - xOffset,
endPoint.Value.Y * scaleCoef - yOffset);
}
}
private void Form1_Resize(object sender, EventArgs e)
{
// ограничение  перемещения панели для отрисовки карты
// при изменении размеров формы
panelRight.Left = (int)(bitmap.Width * START_COEF);
panelRight.Width = this.ClientRectangle.Width - (int)(bitmap.Width *
START_COEF);

panelBottom.Top = (int)(bitmap.Height * START_COEF); panelBottom.Height =
this.ClientRectangle.Height - (int)(bitmap.Height
* START_COEF);
// обновить изображение на панели canvas.Invalidate();
}
// при клике отмечаются начальная и конечная точки пути
private void panel1_MouseClick(object sender, MouseEventArgs e)
{
// проверить пересечение точек графа и курсора foreach (var node in graph.Nodes)
{
Point p = node.Point;
if (startPoint == null && new RectangleF((p.X - 12) * scaleCoef, (p.Y - 12) * scaleCoef, 30 *
```

```
scaleCoef, 30 * scaleCoef).Contains(e.Location))
{
startPoint = p;
textBoxStartAddress.Text = node.Address;
break;
}
else if (endPoint == null && p != startPoint && new RectangleF((p.X - 12) * scaleCoef, (p.Y - 12)
* scaleCoef, 30 * scaleCoef, 30 * scaleCoef).Con-tains(e.Location))
{
endPoint = p;
textBoxEndAddress.Text = node.Address;
DijkstrasTest();
break;
}
}
// перерисовать карту canvas.Invalidate();
}
// нажатие кнопки Сброс
private void btnReset_Click(object sender, EventArgs e)
{
Reset();
}
// при изменении положения ползунка трекбара
// изменяется коэффициент масштабирования
private void trackBar1_Scroll(object sender, EventArgs e)
{
// мастштабирование
scaleCoef = trackBar1.Value / 100.0f + MAGIC_COEF;
// изменяется размер панели для отрисовки карты
// и карта растягивается

canvas.Width = (int)(bitmap.Width * scaleCoef); canvas.Height = (int)(bitmap.Height * scaleCoef);
// небольшое ограничение
// перемещения панели для отрисовки карты var location = canvas.Location;
if (location.X > 3)
location.X = 3;
if (location.X + canvas.Width < panelRight.Left) location.X = panelRight.Left - canvas.Width;
if (location.Y > 3)
location.Y = 3;
if (location.Y + canvas.Height < panelBottom.Top) location.Y = panelBottom.Top - canvas.Height;
canvas.Location = location;
// перерисовать панель canvas.Invalidate();
}
// при нажатии на панель левой кнопкой мыши
// запоминается положение курсора
// это нужно для правильного перемещения панели
private void panel1_MouseDown(object sender, MouseEventArgs e)
{
if (e.Button == System.Windows.Forms.MouseButtons.Left)
{
mouseDown = true;
mousePos = e.Location;
```

```
}
}
// при отпускании кнопки мыши
// перестать тянуть панель
private void panel1_MouseUp(object sender, MouseEventArgs e)
{
mouseDown = false;
}
private void textBoxFind_KeyPress(object sender, KeyPressEventArgs e) {
if (e.KeyChar == (char)Keys.Enter)
{
flag = false;
if (comboBox1.SelectedItem != null)
{
// обнулить линии пути
lines = new List<Line>();
// очистить текстовые поля textBoxFind.Clear(); textBox1.Clear(); canvas.Invalidate();
// Reset();
Node selectedNode = (Node)comboBox1.SelectedItem; textBoxFind.Text = selectedNode.Address;
if (startPoint == null)
{
startPoint = selectedNode.Point; textBoxStartAddress.Text = selectedNode.Address;
}
else if (endPoint == null && selectedNode.Point != startPoint)
{
endPoint = selectedNode.Point; textBoxEndAddress.Text = selectedNode.Address;
}
if (startPoint != null && endPoint != null)
DijkstrasTest();
canvas.Invalidate();
}
// скрываем комбобокс comboBox1.DroppedDown = false; comboBox1.Items.Clear();
}
flag = true;
}
private void textBoxFind_KeyDown(object sender, KeyEventArgs e)
{
// FindItemContaining(graph.Nodes, textBoxFind.Text);
}
private void button1_Click(object sender, EventArgs e)
{
DataBank.Rout = double.Parse(textBox1.Text); DataBank.Street = textBoxEndAddress.Text;
this.Close();
}
// при перемещении мышью панели для отрисовки карты
// ограничить перемещение за край
private void panel1_MouseMove(object sender, MouseEventArgs e)
{
if (mouseDown)
{
int deltaX = e.X - mousePos.X;
int deltaY = e.Y - mousePos.Y;
```

```
var location = new Point(canvas.Left + deltaX, canvas.Top + deltaY); if (location.X > 3)
location.X = 3;
if (location.X + canvas.Width < panelRight.Left) location.X = panelRight.Left - canvas.Width;
if (location.Y > 3)
location.Y = 3;
if (location.Y + canvas.Height < panelBottom.Top) location.Y = panelBottom.Top - canvas.Height;
canvas.Location = location;
canvas.Invalidate();
}
}

CREATE TABLE [dbo].[Car] (
[CarCode] INT             IDENTITY (1, 1) NOT NULL,
[CarName]    NVARCHAR (30) NOT NULL,
[CarAVG]     INT           NOT NULL,
PRIMARY KEY CLUSTERED     ([CarCode] ASC)
);
CREATE TABLE [dbo].[Driver] (
[DriverCode]  INT                  IDENTITY (1, 1) NOT NULL,
[DriverFIO]   NVARCHAR (50)        NOT NULL,
[DriverPosition] NVARCHAR (30)     NOT NULL,
PRIMARY KEY CLUSTERED     ([DriverCode] ASC)
);
CREATE TABLE [dbo].[LastList] (
[TestCode]          INT           IDENTITY (1, 1) NOT NULL,
[TestNumber] NVARCHAR (10)    NOT   NULL,
[TestFio]           NVARCHAR (50)    NOT   NULL,
[TestClass]    NVARCHAR (20)    NOT   NULL,
[TestAuto]          NVARCHAR (20)    NOT   NULL,
[TestOsnova]  NVARCHAR (50)    NOT   NULL,
[TestData]          NVARCHAR (10)    NOT   NULL,
[TestCity]          NVARCHAR (60)    NOT   NULL,
[TestUp]            NVARCHAR (20)    NOT   NULL,
[TestDown]          NVARCHAR (20)    NOT   NULL,
[TestDoc]           NVARCHAR (40)    NOT   NULL,
[TestPrice]    NVARCHAR (10)    NOT   NULL,
PRIMARY KEY CLUSTERED     ([TestCode] ASC)
);
CREATE TABLE [dbo].[Users] (
[UserCode]          INT           IDENTITY (1, 1) NOT NULL,
[UserLogin]    NCHAR (13) NOT NULL,
[UserPassword] NCHAR (13)        NOT NULL,
[UserRole]          NCHAR (9)    NOT NULL,
PRIMARY KEY CLUSTERED     ([UserCode] A
```

ДОДАТОК Б

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	**Опис**
Пояснювальні документи	
Диплом_ doc	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Диплом_.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
Program.rar	Архів. Містить коди програми і откомпільовану програму
Презентація	
Презентація_.ppt	Презентація кваліфікаційної роботи