Міністерство освіти і науки України
Університет митної справи та фінансів

Факультет інноваційних технологій
Кафедра комп'ютерних наук та інженерії програмного забезпечення

Кваліфікаційна робота бакалавра

на тему «Розроблення програмного забезпечення для взаємодії з електронними таблицями»

Виконав: студент групи ІПЗ19-2
Спеціальність 121 Інженерія програмного забезпечення
Щербаткін Владислав Олексійович

Керівник к.е.н., доц. Яковенко Т. Ю.

Рецензент Університет митної справи та фінансів

В.о. завідувача кафедри кібербезпеки та інформаційних технологій

к.т.н., доц. Прокопович-Ткаченко Д.І.

Дніпро – 2023

АНОТАЦІЯ

Щербаткін В.О. Розроблення програмного забезпечення для взаємодії з електронними таблицями.

Кваліфікаційна робота на здобуття освітнього ступеня бакалавр за спеціальністю 121 «Інженерія програмного забезпечення». – Університет митної справи та фінансів, Дніпро, 2023.

Кваліфікаційна робота присвячена розробці програмного забезпечення взаємодії з електронними таблицями.

По-перше, були досліджені основи роботи з електронними таблицями. Це включало розуміння структури даних у таблицях, форматів файлів (XLSX, CSV), основних операцій (створення, читання, запис, редагування) й специфічних функцій, доступних у конкретних додатках для роботи з таблицями.

По-друге, був проведений аналіз вимог користувачів. Було важливо зрозуміти, які функції та можливості очікують користувачі від програмного забезпечення, що розробляється. Це включає функції автоматизації, аналізу даних, генерації звітів, імпорту та експорту даних, фільтрації та сортування, а також підтримку специфічних форматів та розрахунків.

В кваліфікаційній роботі була реалізована функціональність програмного забезпечення. В процесі роботи створювався код, що дозволяє програмному забезпеченню взаємодіяти з електронними таблицями. Після реалізації програмного забезпечення було проведено тестування та налагодження для перевірки його функціональності, стабільності та відповідності вимогам. Було також важливо виявити та виправити можливі помилки та недоліки.

Ключові слова: електронні таблиці, структура даних, формати файлів, аналіз даних, імпорт та експорт даних, мова програмування C++, Qt Creator, Qt Framework.

ABSTRACT

Shcherbatkin V.A. Development of software for interaction with spreadsheets.

Qualification work for a bachelor's degree in specialty 121 «Software Engineering». – University of Customs and Finance, Dnipro, 2023.

The qualification work is devoted to the development of software for interacting with spreadsheets.

First, the basics of working with spreadsheets were studied. This included an understanding of the data structure in tables, file formats (XLSX, CSV), basic operations (create, read, write, edit), and specific functions available in specific spreadsheet applications.

Second, we analyzed user requirements. It was important to understand what features and capabilities users expect from the software being developed. This includes automation functions, data analysis, report generation, data import and export, filtering and sorting, as well as support for specific formats and calculations.

In the qualification work, the software functionality was implemented. In the process, the code was created to allow the software to interact with spreadsheets. After the software was implemented, testing and debugging was performed to verify its functionality, stability, and compliance with the requirements. It was also important to identify and correct possible errors and shortcomings.

Keywords: spreadsheets, data structure, file formats, data analysis, data import and export, C++ programming language, Qt Creator, Qt Framework.

ЗМІСТ

ВСТУП

У сучасному інформаційному суспільстві електронні таблиці стали невід'ємною частиною нашого повсякденного життя. Вони є ефективним інструментом для організації, аналізу та візуалізації даних у різних сферах діяльності, чи то бізнес, чи то фінанси, чи то наукові дослідження, чи то особисте управління завданнями.

Розробка програмного забезпечення для взаємодії з електронними таблицями має величезне значення, оскільки дає змогу автоматизувати і спростити безліч завдань, пов'язаних з обробкою даних. Це включає в себе створення, редагування, аналіз, фільтрацію, сортування та візуалізацію інформації, а також проведення складних обчислювальних операцій і створення звітів.

Актуальність роботи з розроблення програмного забезпечення для взаємодії з електронними таблицями є досить значущою в сучасному інформаційному суспільстві. Ось кілька основних причин, чому ця робота є актуальною:

1) широке застосування електронних таблиць. Електронні таблиці широко використовуються в різних сферах діяльності, включно з бізнесом, фінансами, освітою, дослідженнями та багатьма іншими. Вони надають зручний і ефективний спосіб організації та аналізу даних. Розробка програмного забезпечення, здатного ефективно взаємодіяти з електронними таблицями, дає змогу автоматизувати безліч завдань і підвищити продуктивність роботи;

2) зростаючий обсяг даних. Нині обсяги даних, з якими стикаються, постійно зростають. Електронні таблиці дають змогу зберігати й обробляти великі обсяги даних, і розробка програмного забезпечення для роботи з ними стає дедалі актуальнішою. Таке програмне забезпечення може запропонувати інструменти для ефективної фільтрації, сортування, аналізу та візуалізації даних;

3) підвищення точності та мінімізація помилок. Ручне опрацювання даних в електронних таблицях схильне до людських помилок, особливо під час роботи з великим обсягом інформації. Розробка програмного забезпечення, яке може виконувати автоматичну обробку даних, дає змогу мінімізувати помилки і підвищує точність результатів. Це особливо важливо для бізнесу та фінансових операцій, де навіть невеликі помилки можуть мати серйозні наслідки;

4) інтеграція з іншими системами. Програмне забезпечення для взаємодії з електронними таблицями може бути інтегроване з іншими інформаційними системами, такими як бази даних, CRM-системи, системи управління проєктами та інші. Це забезпечує гнучкіше й ефективніше керування даними та обмін інформацією між різними системами;

5) розвиток нових технологій і можливостей. З розвитком технологій і появою нових підходів до опрацювання нових підходів до опрацювання даних, таких як штучний інтелект, машинне навчання та аналіз великих даних, розробка програмного забезпечення для взаємодії з електронними таблицями стає ще більш актуальною. Використання таких технологій може значно поліпшити процеси аналізу даних і прийняття рішень.

Таким чином, робота з розробки програмного забезпечення для взаємодії з електронними таблицями є актуальною і має широкий спектр застосувань у різних сферах діяльності. Вона дає змогу підвищити ефективність роботи з даними, поліпшити точність результатів і використовувати новітні технології для обробки інформації.

Метою роботи є розроблення програмного забезпечення для взаємодії з електронними таблицями.

Для досягнення поставленої мети з розроблення програмного забезпечення для взаємодії з електронними таблицями можуть бути визначені такі завдання:

1) вивчення основ роботи з електронними таблицями. Це включає розуміння структури даних у таблицях, форматів файлів (наприклад, XLSX,

CSV), основних операцій (створення, читання, запис, редагування) і специфічних функцій, доступних у конкретних додатках для роботи з таблицями;

2) аналіз вимог користувачів. Важливо зрозуміти, які функції та можливості користувачі очікують від розроблюваного програмного забезпечення. Це може включати функції автоматизації, аналізу даних, генерації звітів, імпорту та експорту даних, фільтрації та сортування, а також підтримку специфічних форматів і розрахунків;

3) вивчення наявних інструментів і технологій. Існує безліч мов програмування, фреймворків і бібліотек, призначених для роботи з електронними таблицями. Необхідно провести дослідження та обрати найбільш підходящі інструменти для розроблення, з огляду на вимоги та цілі проєкту;

4) розробка архітектури програмного забезпечення. Це включає визначення модулів, компонентів і інтерфейсів програми, які забезпечуватимуть необхідну функціональність. Важливо також врахувати аспекти безпеки, продуктивності та масштабованості розроблюваного програмного забезпечення;

5) реалізація функціональності програмного забезпечення. У цьому завданні розробники створюють код, який дозволяє програмному забезпеченню взаємодіяти з електронними таблицями. Це може включати читання і запис даних, застосування формул і функцій, маніпуляції з комірками, форматування і створення звітів;

6) тестування та налагодження. Після реалізації програмного забезпечення необхідно провести тестування для перевірки його функціональності, стабільності та відповідності вимогам. Важливо також виявити і виправити можливі помилки та недоліки;

7) оцінка та оптимізація продуктивності. Якщо програма працює з великим обсягом даних, може знадобитися оптимізація продуктивності;

8) продуктивності. Це може включати оптимізацію запитів до таблиць, поліпшення алгоритмів обробки даних і реалізацію багатопоточності або розподілених обчислень;

9) документування та підтримка. Не менш важливим завданням є створення документації, яка описує функціональність програмного забезпечення та інструкції з його використання. Також може знадобитися підтримка і супровід програмного продукту, включно з виправленням помилок, додаванням нових функцій і оновленнями відповідно до змін у стандартах і вимог користувачів.

Розв'язання цих завдань допоможе досягти поставленої мети розроблення програмного забезпечення для взаємодії з електронними таблицями та створити повнофункціональне, надійне й ефективне програмне рішення для роботи з даними.

Об'єктом дослідження є процеси роботи програмного забезпечення для взаємодії з електронними таблицями.

Предметом дослідження є апаратно-програмне забезпечення для розроблення програмного забезпечення для взаємодії з електронними таблицями.

Практичне значення одержаних результатів полягає у підвищенні якості взаємодії з електронними таблицями.

Результатами роботи є програмне забезпечення для взаємодії з електронними таблицями.

Кваліфікаційна робота складається зі вступу, 3-х розділів, висновків, використаних джерел з 20 найменувань, 2-х додатків.

Обсяг роботи складається з 53 сторінок основного тексту з 70 сторінок кваліфікаційної роботи та 20 рисунків.

РОЗДІЛ 1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ. ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Огляд поняття електронних таблиць

Електронна таблиця – це структуроване уявлення даних у вигляді сітки, що складається з осередків, які можуть містити текст, числа, формули та інші значення. Вона являє собою електронний документ, де дані організовані у вигляді таблиці з рядами та стовпцями. Електронні таблиці широко застосовуються для зручної організації, зберігання та обробки даних.

Особливості електронних таблиць:

1) осередки. Основний елемент електронної таблиці це комірка. Осередки утворюють сітку, яка розділена на рядки та стовпці. Кожна комірка має унікальну адресу, яка визначається номером рядка та стовпця (наприклад, A1, B2 тощо);

2) формати файлів. Електронні таблиці можуть бути збережені у різних форматах файлів, таких як XLSX, CSV, ODS та інші. Кожен формат має свої особливості та можливості, і вибір формату залежить від конкретних потреб та вимог;

3) функції та формули. Однією з основних особливостей електронних таблиць є можливість використання функцій та формул для автоматичного виконання обчислень та обробки даних. Функції можуть виконувати математичні операції, аналізувати дані, фільтрувати, сортувати та багато іншого;

4) посилання та залежності. Електронні таблиці дозволяють встановлювати посилання між осередками, аркушами і навіть різними файлами. Це дозволяє створювати зв'язки між даними та оновлювати їх автоматично під час зміни вихідних даних. Залежності між осередками можуть бути використані для створення складних моделей та аналізу даних;

5) форматування та стилізація. Електронні таблиці дозволяють форматувати дані та застосовувати стилі до осередків, рядів та стовпців. Це включає зміну шрифту, кольору, вирівнювання, застосування умовного форматування та багато інших можливостей візуального представлення даних;

6) сортування та фільтрація. Електронні таблиці надають функціональність сортування даних за різними критеріями та фільтрації даних на основі заданих умов. Це дозволяє швидко знаходити, відображати та аналізувати потрібні дані в таблиці;

7) графіки та діаграми. Електронні таблиці також підтримують створення графіків та діаграм для візуалізації даних. Графіки дозволяють наочно відобразити зв'язки та тренди даних, що спрощує їх аналіз та розуміння;

8) імпорт та експорт даних. Електронні таблиці забезпечують можливість імпорту та експорту даних з інших джерел або в інші формати. Це дозволяє обмінюватись даними з іншими додатками та системами, такими як бази даних, текстові файли тощо;

9) спільна робота та доступ до даних. Електронні таблиці можуть використовуватися для спільної роботи, де кілька користувачів можуть одночасно редагувати таблицю та бачити зміни у режимі реального часу. Також передбачені різні рівні доступу до даних, що забезпечує контроль за конфіденційністю та безпекою інформації;

10) макроси та автоматизація. Деякі електронні таблиці підтримують макроси та автоматизацію операцій. Макроси дозволяють записувати і відтворювати послідовність дій, що спрощує повторення рутиних операцій або створення функцій користувача.

Ці особливості електронних таблиць роблять їх потужним інструментом для управління, аналізу та обробки даних у різних сферах, включаючи бізнес, фінанси, науку, освіту та багато інших.

1.2 Огляд форматів файлів електронних таблиць

Формати файлів електронних таблиць – це спеціальні стандарти та способи організації даних у файлі, які визначають структуру, типи даних, форматування, функції та інші характеристики таблиці. Кожен формат має свої особливості та може бути підтримуваний різними програмами для роботи з електронними таблицями.

Деякі з найпоширеніших форматів файлів електронних таблиць наведені нижче.

XLSX (Excel XML Spreadsheet Format):

– це основний формат файлів Microsoft Excel, який використовує розширення xlsx;

– XLSX заснований на XML і є бінарним форматом, в якому дані зберігаються у стислому вигляді;

– підтримує багато функцій, формул, форматування, графіки, макроси та інші можливості Excel;

– має гарну сумісність з різними програмами для роботи з електронними таблицями.

CSV (Comma-Separated Values):

– CSV є текстовим форматом, де дані таблиці розділені комами (або іншим роздільником, таким як точка з комою або табуляція);

– CSV-файли не підтримують формули, макроси та складну структуру, але є простим і широко використовуваним форматом для обміну даними між різними програмами;

– підходить для великих обсягів даних та простих таблиць без складних обчислень чи форматування.

ODS (OpenDocument Spreadsheet):

– ODS – це формат файлів електронних таблиць у рамках стандарту OASIS OpenDocument;

– розроблений як відкритий формат, ODS забезпечує сумісність із різними програмами для роботи з електронними таблицями, такими як LibreOffice, OpenOffice та іншими;

– підтримує функції, формули, графіки, макроси та інші можливості роботи з даними.

XLS (Excel Binary Workbook Format):

– це бінарний формат файлів Excel, який використовує розширення .xls;

– XLS був попередником формату XLSX, і хоча він все ще використовується, його використання стає менш поширеним;

– він підтримує безліч функцій та можливостей Excel, але може бути менш ефективним та менш сумісним у порівнянні з форматом XLSX.

Google Sheets – це онлайн-додаток для створення та роботи з електронними таблицями, розроблений Google.

Google Sheets:

– формат файлів Google Sheets зазвичай має розширення .gsheet або .xlsx (при експорті);

– цей формат призначений для роботи в хмарному оточенні та забезпечує спільне редагування, зберігання даних та інтеграцію з іншими сервісами Google.

HTML (HyperText Markup Language):

– HTML є мовою розмітки, яка використовується для створення веб-сторінок, але також може використовуватися для представлення простих електронних таблиць;

– HTML-файли містять теги та структуру, що описує таблицю, і можуть бути переглянуті веб-браузерами.

Кожен із цих форматів має свої переваги та недоліки, і вибір формату залежить від конкретних потреб та вимог, а також від програмного забезпечення, яке використовуватиметься для роботи з електронними таблицями.

1.3 Огляд основних операцій та функцій в контексті роботи з електронними таблицями

Операції та функції, що використовуються під час роботи з електронними таблицями, дозволяють виконувати різні дії з даними, обчислювати значення, аналізувати інформацію та візуалізувати результати. Деякі з основних операцій та функцій, які широко застосовуються під час роботи з електронними таблицями, наведені нижче.

Вставка та видалення даних:

– вставлення даних у комірки. Операція вставки дозволяє додати дані у обрані комірки таблиці;

– видалення даних із осередків. Операція видалення дозволяє видалити дані з обраних осередків, звільняючи їх для інших даних.

Копіювання та переміщення даних:

– копіювання даних. Операція копіювання дозволяє створити дублікати обраних осередків або діапазонів даних;

– переміщення даних. Операція переміщення дозволяє перемістити обрані осередки або діапазони даних з одного місця до іншого.

Форматування даних:

– форматування чисел та тексту. Операції форматування дозволяють змінювати зовнішній вигляд чисел та тексту, включаючи стиль шрифту, розмір, колір, вирівнювання та ін;

– умовне форматування. Функціональність умовного форматування дозволяє автоматично застосовувати певне форматування даних на основі заданих умов;

– форматування дат та часу. Операції форматування також дозволяють задавати формати відображення дати та часу, включаючи формати дат, часу, дати та часу разом тощо.

Обчислення та формули:

– арифметичні операції. Електронні таблиці підтримують основні арифметичні операції, такі як додавання, віднімання, множення та розподіл чисел;

– формули. Формули дозволяють виконувати складні обчислення та операції над даними. Формули можуть включати математичні функції (наприклад, сума, середня), логічні операції (наприклад, умовні вирази), посилання на комірки та інші функції;

– автозаповнення. Операція автозаповнення дозволяє швидко заповнити осередки або ряди даними на основі певних шаблонів або послідовностей (наприклад, числові ряди, дати, текстові шаблони тощо);

– посилання на комірки. Функціональність посилань на комірки дозволяє використовувати значення з одного комірки чи діапазону в інших формулах чи обчисленнях.

Сортування та фільтрація даних:

– сортування. Операція сортування дозволяє впорядкувати дані в таблиці за заданим критерієм, наприклад, за зростанням або зменшенням числових значень, алфавітним порядком або датами;

– фільтрування. Фільтрування даних дозволяє відображати лише ті рядки, які відповідають певним умовам, таким як значення стовпця, текстові фільтри, числові діапазони тощо.

Графіки та діаграми:

– створення графіків. Електронні таблиці пропонують можливість створення різних типів графіків та діаграм для візуалізації даних. Це включає стовпчасті, кругові, лінійні, точкові та інші графіки;

– налаштування графіків. Операції налаштування дозволяють змінювати зовнішній вигляд графіків, включаючи кольори, шрифти, масштабування осей, додавання заголовків та підписів.

Спільна робота:

– онлайн-спільна робота. Деякі програми для роботи з електронними таблицями підтримують спільне редагування, дозволяючи кільком

користувачам одночасно працювати над таблицею та бачити зміни у режимі реального часу;

– керування доступом. Функції керування доступом дозволяють встановлювати різні рівні доступу до даних, визначати права на читання, запис та редагування таблиці для різних користувачів чи груп користувачів.

Це лише деякі з багатьох операцій та функцій, доступних в електронних таблицях. Конкретний перелік операцій та функцій може змінюватись в залежності від програмного забезпечення, яке використовується, та його можливостей.

1.4 Огляд існуючих електронних таблиць

На сьогоднішній день існує кілька популярних програмних програм та сервісів для роботи з електронними таблицями.

Microsoft Excel (рис. 1.1):

– Microsoft Excel є одним з найбільш відомих та широко використовуваних програм для роботи з електронними таблицями;

– він пропонує безліч функцій та можливостей, включаючи форматування даних, обчислення та формули, графіки, сортування та фільтрацію, макроси, спільну роботу та багато іншого;

– Excel має широку підтримку та сумісність із різними операційними системами, включаючи Windows та macOS.

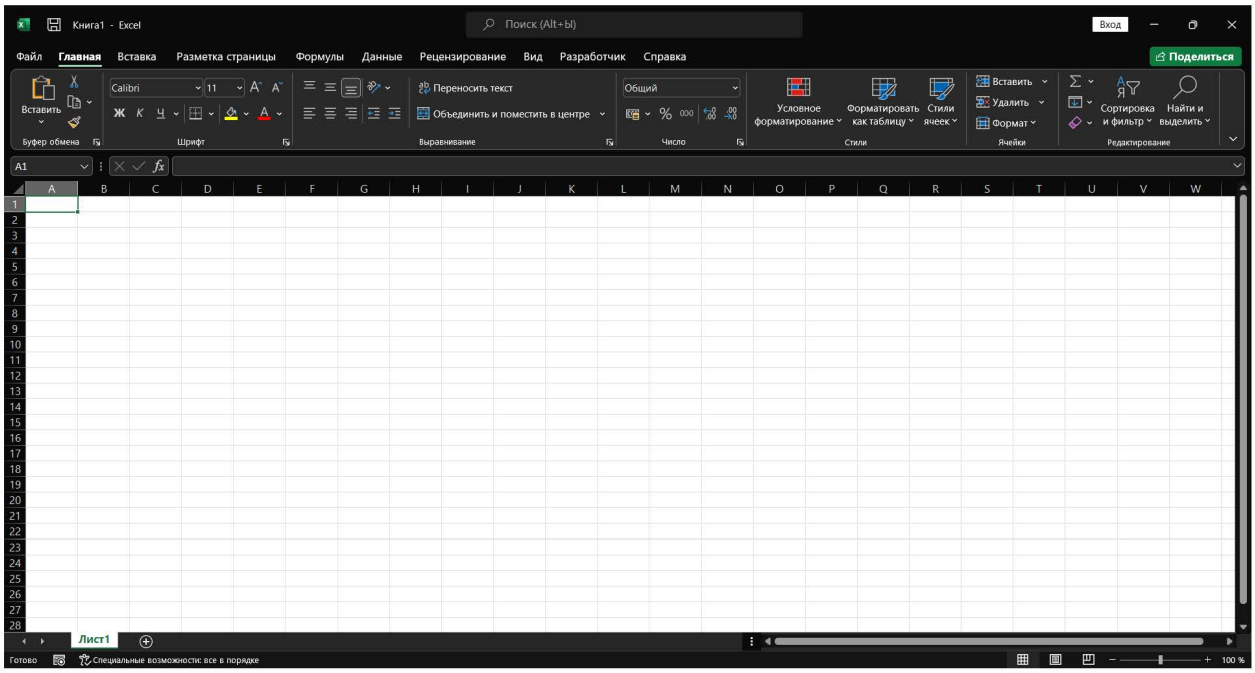


Рисунок 1.1 – Інтерфейс Excel

Google Sheets (рис. 1.2):

– Google Sheets – це безкоштовний онлайн-сервіс електронних таблиць, що надається Google;

– він пропонує безліч функцій для роботи з даними, аналогічних Excel, і має перевагу спільної роботи в реальному часі та доступу до таблиць з будь-якого пристрою з підключенням до інтернету. - Google Sheets інтегрується з іншими сервісами Google, такими як Google Документи та Google Презентації, та підтримує розширення та доповнення для розширення його функціональності.

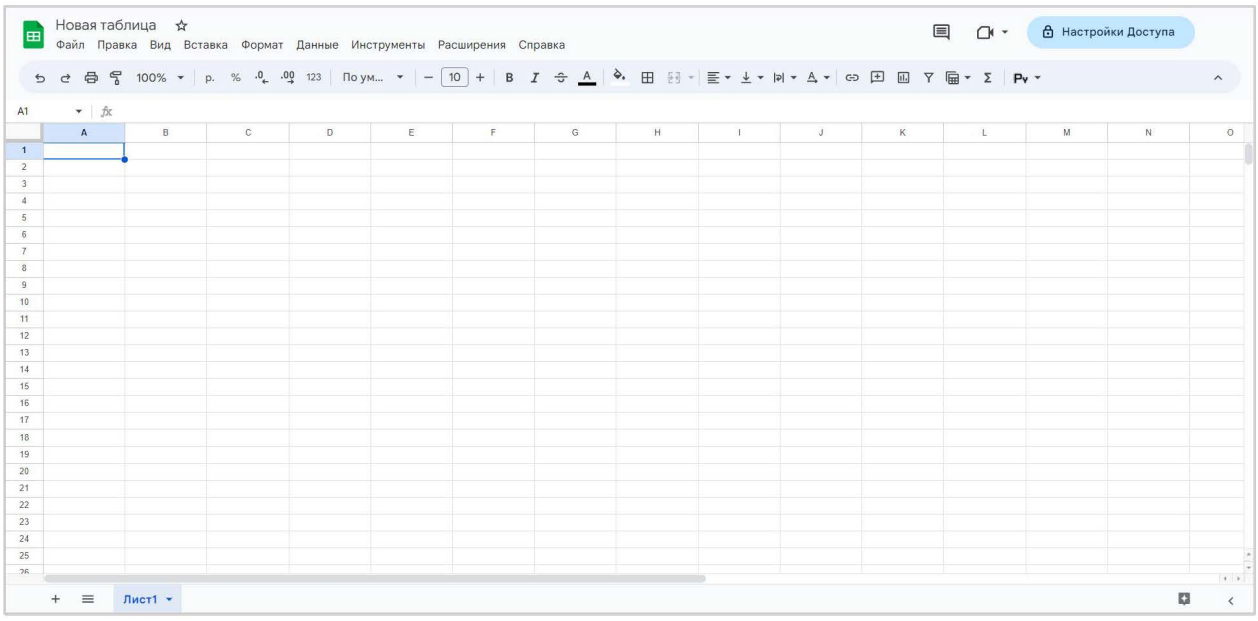


Рисунок 1.2 – Інтерфейс Google Sheets

LibreOffice Calc (рис. 1.3):

– LibreOffice Calc це безкоштовна програма для роботи з електронними таблицями, що входить до складу пакету офісних програм LibreOffice;

– Calc надає безліч функцій для створення та редагування таблиць, включаючи формули, сортування, фільтрацію, графіки, імпорт та експорт у різні формати та інші можливості;

– LibreOffice Calc є крос-платформною програмою і може використовуватися на Windows, macOS та Linux.

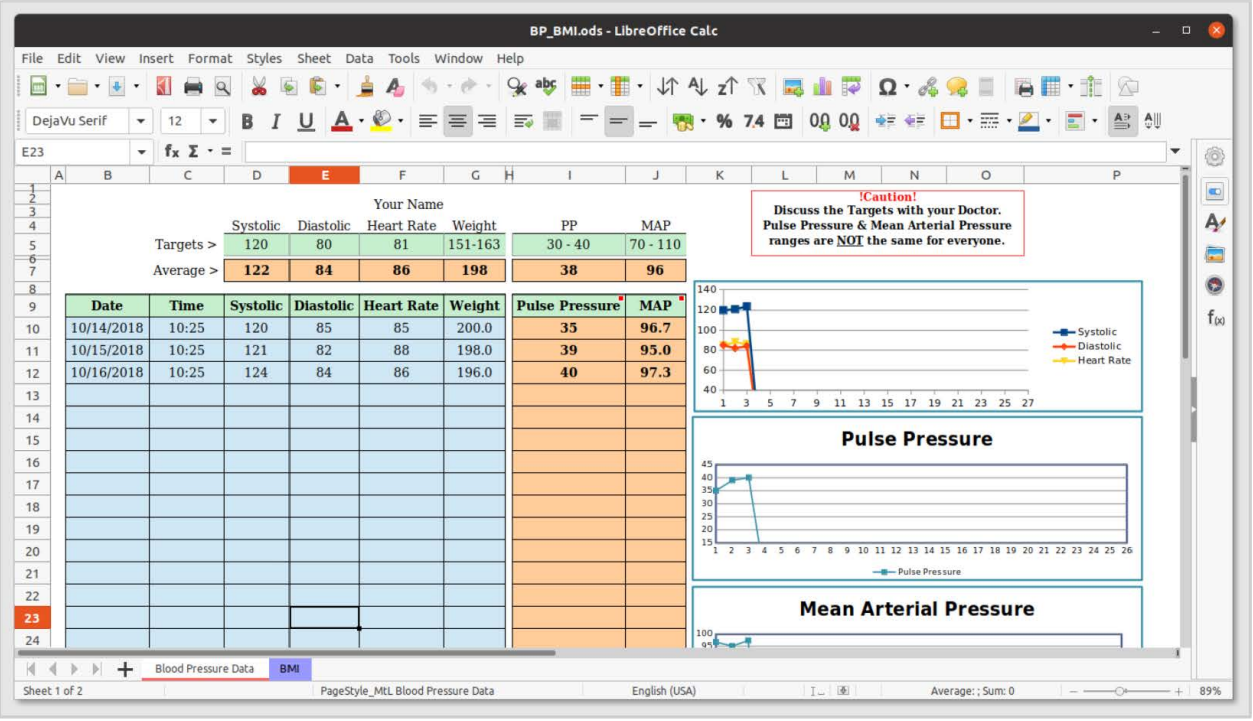


Рисунок 1.3 – Інтерфейс LibreOffice Calc

Apple Numbers:

– Apple Numbers це додаток для роботи з електронними таблицями, розроблений спеціально для операційної системи macOS та пристроїв Apple;

– воно пропонує інтуїтивний інтерфейс користувача і безліч функцій для створення і форматування таблиць, додавання графіків, використання формул і сортування даних;

– Apple Numbers має інтеграцію з іншими програмами Apple, такими як Pages (для обробки текстових документів) та Keynote (для створення презентацій).

Zoho Sheet (рис. 1.4):

– Zoho Sheet це онлайн-додаток для роботи з електронними таблицями, запропоноване компанією Zoho;

– воно надає широкий набір функцій, включаючи формули, фільтрацію, сортування, графіки, спільну роботу та інтеграцію з іншими програмами Zoho;

– Zoho Sheet також підтримує імпорт та експорт файлів у різних форматах, таких як Excel, CSV та інші.

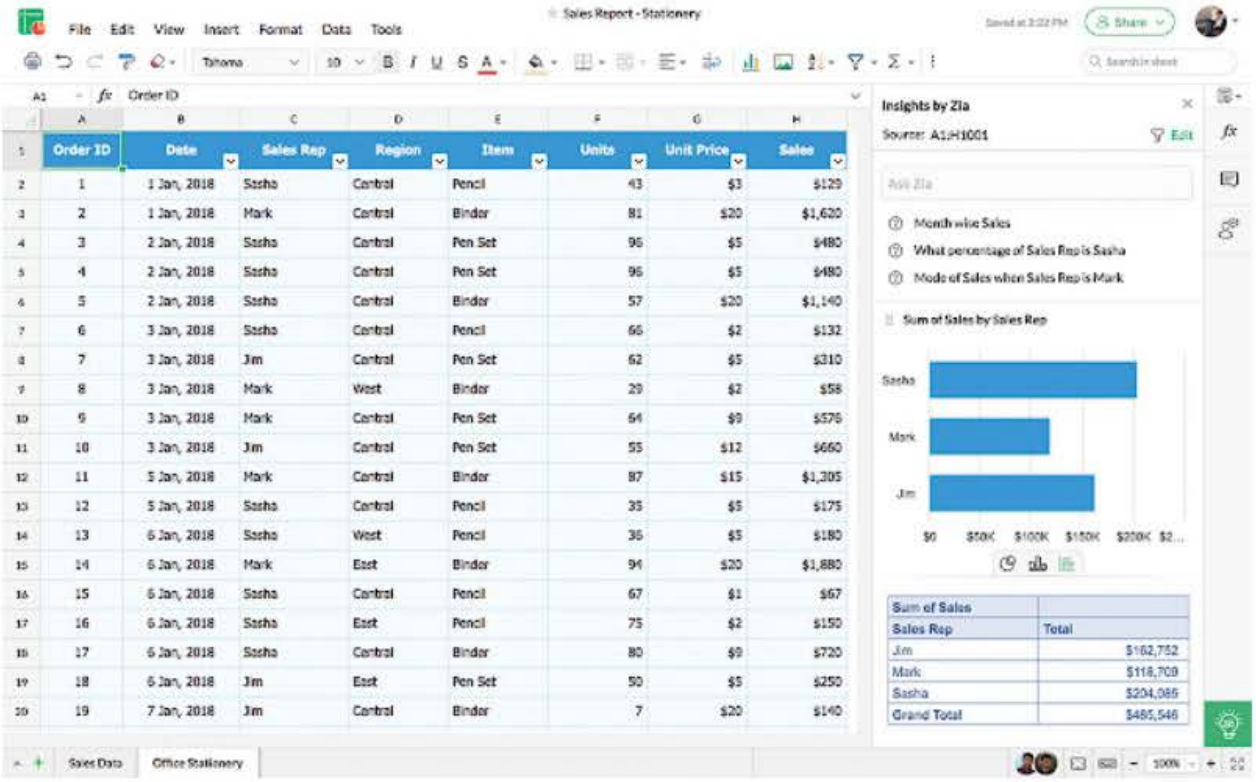


Рисунок 1.4 – Інтерфейс Zoho Sheet

Кожна з цих програм має свої особливості, функції та можливості, і вибір конкретної програми залежить від потреб, переваг та доступності на використовуваній платформі або в мережі.

1.5 Огляд технологій та засобів для розроблення програмного забезпечення для взаємодії з електронними таблицями

Для розробки програмного забезпечення, що взаємодіє з електронними таблицями, існує низка технологій та засобів, які полегшують цей процес.

API та бібліотеки для роботи з електронними таблицями:

– Google Sheets API. Google Sheets API надає можливість взаємодії з Google Sheets, дозволяючи створювати, змінювати та зчитувати дані з таблиць. Він надає широкий набір методів та функцій для роботи з осередками,

діапазонами даних, формулами, форматуванням та іншими аспектами таблиць;

– Microsoft Office Interop API. Для роботи з електронними таблицями Microsoft Excel можна використовувати Microsoft Office Interop API. Це набір бібліотек та інтерфейсів, який дозволяє взаємодіяти з Excel із додатків, написаних на платформі .NET;

– Apache POI. Apache POI це бібліотека Java, яка надає можливості для роботи з форматами файлів Microsoft Office, включаючи Excel. Вона дозволяє створювати, модифікувати та зчитувати дані з електронних таблиць у форматі XLS та XLSX.

Мови програмування:

– Python. Python є популярною мовою програмування для розробки програм, включаючи ті, що працюють з електронними таблицями. Існують різні бібліотеки, такі як pandas, openpyxl, xlrd, які полегшують взаємодію Космосу з таблицями у форматах Excel та інших;

– JavaScript. JavaScript широко використовується для розробки веб-застосунків, і його можна використовувати для взаємодії з електронними таблицями в браузері. Деякі бібліотеки, такі як SheetJS та Handsontable, надають функції для читання, запису та обробки даних таблиць;

– якщо розробляються програми для платформи .NET, то мова C# та пов'язані бібліотеки, такі як EPPlus, ClosedXML, дозволяють працювати з електронними таблицями формату Excel.

Фреймворки для веб-розробки:

– Flask. Flask – це легкий фреймворк для розробки веб-додатків мовою Python. Він може бути використаний для створення веб-застосунків, які взаємодіють з електронними таблицями, наприклад, для імпорту, експорту чи аналізу даних;

– Node.js. Node.js – це платформа для виконання JavaScript-коду на сервері. За допомогою фреймворків, таких як Express.js або Koa.js, можна

створювати веб-програми, які взаємодіють з електронними таблицями та надають API для роботи з даними.

Інструменти розробки:

– Visual Studio. Visual Studio – це інтегроване середовище розробки (IDE) від Microsoft, яке надає широкий набір інструментів для розробки програм на різних платформах, включаючи програми для роботи з електронними таблицями;

– Jupyter Notebook. Jupyter Notebook це інтерактивне середовище розробки, що дозволяє створювати та виконувати код у блокноті. Він широко використовується для аналізу даних та дослідницького програмування, включаючи роботу з електронними таблицями.

Хмарні послуги:

– Amazon Web Services (AWS). AWS пропонує ряд сервісів для зберігання та обробки даних, включаючи Amazon S3 для зберігання файлів та Amazon Athena для виконання SQL-запитів до даних, включаючи електронні таблиці;

– Google Cloud Platform (GCP). GCP також пропонує хмарні послуги для роботи з даними, такі як Google Cloud Storage для зберігання файлів та BigQuery для виконання аналітичних запитів до даних.

Залежно від потреб та переваг, можна обирати поєднання різних технологій, мов програмування, фреймворків та інструментів розробки для створення програмного забезпечення, яке ефективно взаємодіятиме з електронними таблицями.

1.6 Висновки до першого розділу. Постановка задач дослідження

В результаті вивчення предметної галузі, постановки завдань та проведення огляду різних аспектів роботи з електронними таблицями можна зробити такі загальні висновки:

1) електронні таблиці є зручним і широко використовуваним інструментом для організації та аналізу даних. Вони дозволяють зберігати та структурувати інформацію у вигляді таблиць, а також виконувати різні операції та обчислення над даними;

2) формати файлів електронних таблиць включають такі стандартні розширення, як XLS, XLSX і CSV. Кожен формат має свої особливості та можливості, і вибір формату залежить від конкретних вимог та сумісності з програмними засобами;

3) у роботі з електронними таблицями використовуються різні операції та функції. Вони включають форматування даних, обчислення та формули, сортування та фільтрацію, створення графіків, спільну роботу та інші операції, які полегшують маніпуляцію та аналіз даних;

4) на ринку існує кілька популярних електронних таблиць, таких як Microsoft Excel, Google Sheets, LibreOffice Calc, Apple Numbers та Zoho Sheet. Кожна з них має свої особливості, функціональність та підтримку різних платформ, що дозволяє обрати відповідний варіант залежно від потреб користувача;

5) для розробки програмного забезпечення, яке взаємодіє з електронними таблицями, є безліч технологій і засобів. Це включає API та бібліотеки, такі як Google Sheets API, Microsoft Office Interop API та Apache POI, а також мови програмування, такі як Python, JavaScript та C#. Крім того, є фреймворки для веб-розробки та хмарні сервіси, які полегшують роботу з даними таблиць в онлайн-середовищі.

Виходячи з проведеного огляду, можна зробити висновок про значущість електронних таблиць та необхідність розробки програмного забезпечення для їхньої взаємодії. Таке програмне забезпечення може полегшити роботу з даними, підвищити продуктивність та ефективність аналізу інформації, а також покращити можливості спільної роботи над таблицями.

Метою роботи є розроблення програмного забезпечення для взаємодії з електронними таблицями.

Для досягнення поставленої мети з розроблення програмного забезпечення для взаємодії з електронними таблицями можуть бути визначені такі завдання:

1) вивчення основ роботи з електронними таблицями. Це включає розуміння структури даних у таблицях, форматів файлів (наприклад, XLSX, CSV), основних операцій (створення, читання, запис, редагування) і специфічних функцій, доступних у конкретних додатках для роботи з таблицями;

2) аналіз вимог користувачів. Важливо зрозуміти, які функції та можливості користувачі очікують від розроблюваного програмного забезпечення. Це може включати функції автоматизації, аналізу даних, генерації звітів, імпорту та експорту даних, фільтрації та сортування, а також підтримку специфічних форматів і розрахунків;

3) вивчення наявних інструментів і технологій. Існує безліч мов програмування, фреймворків і бібліотек, призначених для роботи з електронними таблицями. Необхідно провести дослідження і обрати найбільш підходящі інструменти для розроблення, з огляду на вимоги та цілі проєкту;

4) розробка архітектури програмного забезпечення. Це включає визначення модулів, компонентів і інтерфейсів програми, які забезпечуватимуть необхідну функціональність. Важливо також врахувати аспекти безпеки, продуктивності та масштабованості розроблюваного програмного забезпечення;

5) реалізація функціональності програмного забезпечення. У цьому завданні розробники створюють код, який дозволяє програмному забезпеченню взаємодіяти з електронними таблицями. Це може включати читання і запис даних, застосування формул і функцій, маніпуляції з комірками, форматування і створення звітів;

6) тестування та налагодження. Після реалізації програмного забезпечення необхідно провести тестування для перевірки його функціональності, стабільності та відповідності вимогам. Важливо також виявити і виправити можливі помилки та недоліки;

7) оцінка та оптимізація продуктивності. Якщо програма працює з великим обсягом даних, може знадобитися оптимізація продуктивності;

8) продуктивності. Це може включати оптимізацію запитів до таблиць, поліпшення алгоритмів обробки даних і реалізацію багатопоточності або розподілених обчислень;

9) документування та підтримка. Не менш важливим завданням є створення документації, яка описує функціональність програмного забезпечення та інструкції з його використання. Також може знадобитися підтримка і супровід програмного продукту, включно з виправленням помилок, додаванням нових функцій і оновленнями відповідно до змін у стандартах і вимог користувачів.

Розв'язання цих завдань допоможе досягти поставленої мети розроблення програмного забезпечення для взаємодії з електронними таблицями та створити повнофункціональне, надійне й ефективне програмне рішення для роботи з даними.

РОЗДІЛ 2 АНАЛІЗ ЗАСОБІВ РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ВЗАЄМОДІЇ З ЕЛЕКТРОННИМИ ТАБЛИЦЯМИ

2.1 Вибір програмних засобів для розроблення програмного забезпечення для взаємодії з електронними таблицями

Додаток доцільно розробити з використанням наступних технологій:

1) Qt Framework. Програма написана мовою програмування C++ з використанням фреймворку Qt. Qt надає різні інструменти та бібліотеки для розробки крос-платформних додатків із графічним інтерфейсом;

2) Qt Widgets. Графічний інтерфейс програми створений з використанням Qt Widgets, який надає набір готових елементів управління (кнопки, поля введення, таблиці та ін), які можуть бути використані для побудови інтерфейсу користувача;

3) QFile та QTextStream. Для роботи з файлами використовуються класи QFile та QTextStream з Qt. QFile надає функціональність для роботи з файлами, включаючи читання та запис. QTextStream надає зручний спосіб читання та записування текстових даних з файлу;

4) QStandardItemModel. Для представлення даних у таблиці використовується клас QStandardItemModel. Він надає модель даних для відображення у вигляді таблиці та дозволяє додавати, видаляти та змінювати елементи даних;

5) QMessageBox. Клас QMessageBox використовується для відображення діалогових вікон із повідомленнями про помилки або інформаційними повідомленнями;

6) QString. Для роботи з рядками використовується клас QString, який надає різні функції маніпуляції рядками.

Таким чином, основні технології, використані для створення цієї програми, включають Qt Framework, Qt Widgets, QFile, QTextStream, QStandardItemModel, QMessageBox і QString.

2.2 Qt Framework

Qt – це кроссплатформенний фреймворк для розробки програм з відкритим вихідним кодом. Він надає набір інструментів і бібліотек, які дозволяють розробникам створювати потужні та інтуїтивно зрозумілі графічні інтерфейси користувача, а також реалізовувати функціональність, включаючи взаємодію з електронними таблицями.

Деякі ключові особливості та можливості Qt Framework:

1) кросплатформенність. Однією з головних переваг Qt є його здатність працювати на різних операційних системах, таких як Windows, macOS, Linux, Android та iOS. Це дозволяє розробникам створювати програми, які можуть бути запущені на різних платформах, без необхідності писати вихідний код заново;

2) графічні інтерфейси користувача. Qt надає потужні інструменти для створення привабливих і функціональних графічних інтерфейсів (GUI). За допомогою Qt можна легко створювати та налаштовувати різні елементи інтерфейсу, такі як кнопки, поля введення, таблиці та графіки. Він також пропонує широкий вибір стилів та тем оформлення, щоб адаптувати інтерфейс під вимоги проекту;

3) модульність та розширюваність. Qt має модульну структуру, яка дозволяє розробникам вибирати тільки ті компоненти та функціональність, які їм необхідні. Фреймворк включає безліч модулів, таких як Qt Core, Qt GUI, Qt Widgets, Qt Network та інші, які надають різні можливості для роботи з даними, мережею, мультимедіа та іншими аспектами додатків;

4) обробка подій. Qt надає механізм обробки подій, який дозволяє відстежувати та реагувати на різні події, такі як клацання миші, натискання

клавіш, зміна даних та інші. Це робить взаємодію з електронними таблицями зручнішою та гнучкішою;

5) підтримка багатьох мов програмування. Qt підтримує кілька мов програмування, включаючи C++, Python, JavaScript та QML. При цьому можна обирати найбільш зручну для вас мову для розробки програм з використанням Qt;

6) інтеграція з електронними таблицями. Qt надає різні засоби та API для взаємодії з електронними таблицями. Можна використовувати функції та класи Qt для читання, запису та обробки даних у форматі електронних таблиць. Крім того, існують сторонні бібліотеки та плагіни, які розширюють можливості Qt для роботи з різними форматами електронних таблиць;

7) широке співтовариство та документація. Qt має активну спільноту розробників, яка надає підтримку, навчальні матеріали та безліч ресурсів для допомоги у розробці програм з використанням фреймворку. Офіційна документація Qt є вичерпним джерелом інформації про функції, класи та методи, що полегшує вивчення та використання фреймворку.

Qt Framework надає потужні та гнучкі інструменти для розробки програм з інтерфейсом та функціональністю, пов'язаною з електронними таблицями. Завдяки своїй кросплатформенності, модульності та широким можливостям, Qt є популярним вибором для розробників, які прагнуть створити якісні програми, що взаємодіють із даними таблиць.

2.3 Qt Creator C++

Qt Creator – це інтегроване середовище розробки (IDE) для створення програм на основі Qt та мови програмування C++. Воно надає розробникам зручний і потужний набір інструментів для створення графічних інтерфейсів, написання коду, налагодження та складання додатків.

Інтерфейс Qt Creator зображено на рисунку 2.1.

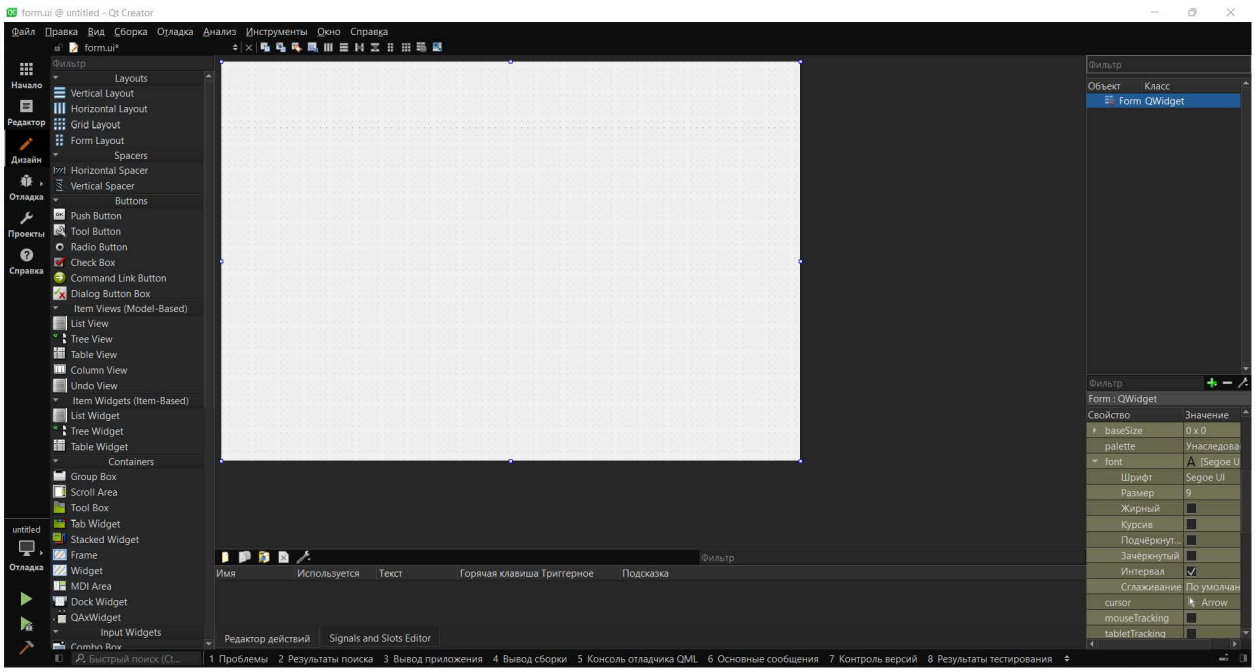


Рисунок 2.1 – Інтерфейс Qt Creator

Опис основних можливостей та функціональності Qt Creator для розробки мовою C++ з використанням Qt:

1) редактор коду. Qt Creator надає потужний та гнучкий редактор коду, який підтримує функції автодоповнення, форматування коду, перевірки синтаксису та підсвічування синтаксису для мови C++. Він забезпечує зручне написання та редагування коду, що підвищує продуктивність розробника;

2) дизайнер графічного інтерфейсу. Qt Creator включає інтуїтивно зрозумілий дизайнер графічного користувальницького інтерфейсу (GUI), який дозволяє створювати і налаштовувати елементи інтерфейсу, такі як кнопки, поля введення, таблиці та інші. Дизайнер дає змогу візуально розробити інтерфейс, а потім згенерувати відповідний код C++;

3) інструменти налагодження. Qt Creator надає потужні інструменти для налагодження програм. Можна використовувати точки зупинки, покрокове виконання коду, перегляд значень змінних та багато іншого для виявлення та усунення помилок у своєму коді. Це дозволяє ефективно налагоджувати програми та підвищувати їх якість;

4) управління проектом. За допомогою Qt Creator можна створювати та керувати проектами на основі Qt та C++. Можна додавати та видаляти файли, налаштовувати залежності, налаштовувати параметри компіляції та лінківки. Це полегшує організацію та структурування проекту;

5) інтеграція з Qt та Qt Libraries. Qt Creator повністю інтегрований з Qt і надає простий доступ до функціональності Qt та Qt Libraries. Можна використовувати готові компоненти та класи Qt для створення графічних інтерфейсів, роботи з даними, мережею, базами даних та іншими аспектами розробки програм;

6) складання та розгортання. Qt Creator забезпечує зручне складання та розгортання додатків. Можна налаштувати параметри складання, обрати цільову платформу і легко створити інсталяційні пакети або файли, що виконуються для своїх додатків. Це дозволяє ефективно поширювати та запускати програму на різних платформах;

7) інтеграція з системами керування версіями. Qt Creator підтримує інтеграцію з популярними системами керування версіями, такими як Git, Subversion та Mercurial. Це дозволяє розробникам легко керувати версіями свого коду, відстежувати зміни, вносити коментарі та взаємодіяти з іншими учасниками проекту;

8) підтримка безлічі інструментів для статичного аналізу. Qt Creator надає можливості для виконання статичного аналізу коду, щоб виявити потенційні проблеми, помилки та покращити якість коду. Він інтегрується з такими інструментами, як Clang Code Model, Cppcheck та іншими, що допомагає розробникам виявити та усунути проблеми до запуску програми;

9) підтримка юніт-тестування. Qt Creator в значній мірі полегшує процес написання та запуску юніт-тестів для коду. Можна створювати тестові сценарії, перевіряти коректність роботи окремих компонентів програми, що розроблюється, та автоматично виконувати тести для забезпечення стабільності та надійності коду;

10) інтегрована документація та довідка. При цьому Qt Creator надає зручний доступ до офіційної документації Qt, а також документації власного проекту. Можна швидко отримати інформацію про класи, функції та їх використання, а також отримати контекстну довідку прямо всередині розробки;

11) підтримка безлічі плагінів. Qt Creator має відкриту архітектуру, яка дозволяє розробникам створювати та інтегрувати плагіни для розширення функціональності середовища. Це означає, що можна налаштувати середовище розробки під потреби, додавши додаткові інструменти та можливості.

Отже, Qt Creator C++ є потужним та інтуїтивно зрозумілим середовищем розробки для створення додатків на основі Qt та мови програмування C++. Його функціональність та інструменти значно полегшують процес розробки, налагодження та складання додатків, що дозволяє розробникам більш ефективно створювати якісне програмне забезпечення.

2.4 Qt Widgets

Qt Widgets – це модуль фреймворку Qt, який надає набір готових до використання віджетів (елементів управління), які можуть бути використані для створення графічного інтерфейсу в додатках. Він надає широкий спектр функціональності та можливостей для створення інтерактивних і чуйних інтерфейсів користувача.

Приклад використання різноманітних віджетів зображено на рисунку 2.2.

```
QStandardItemModel* update_table(QStandardItemModel* csvModel, Ui::MainWindow *ui, QString file_path, QString _split) {
    delete csvModel;
    csvModel = new QStandardItemModel();
    QFile file(file_path);
        if (!file.open(QFile::ReadOnly | QFile::Text)) {
            Show_Error();
        } else {
            QTextStream in(&file);
            while (!in.atEnd())
            {
                QString line = in.readLine();
                QList<QStandardItem *> standardItemsList;
                for (QString item : line.split(_split)) {
                    standardItemsList.append(new QStandardItem(item));
                }
                csvModel->insertRow(csvModel->rowCount(), standardItemsList);
            }
            file.close();
        }
    ui->tableView->setModel(csvModel);
    return csvModel;
}

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    ui->label->setToolTip("Працює як в разі відкриття, так й під час збереження");
    csvModel = new QStandardItemModel(this);
}

MainWindow::~MainWindow() {
    delete ui;
    delete csvModel;
```

Рисунок 2.2 – Приклад використання різноманітних віджетів

Докладний опис основних можливостей та функціональності Qt Widgets:

1) графічні елементи управління. Qt Widgets надає багатий набір графічних елементів управління (віджетів), таких як кнопки, поля введення, прапорці, перемикачі, списки, таблиці, дерева та багато інших. Ці елементи керування дозволяють розробникам створювати інтерфейси з різними функціями та можливостями, відображати та редагувати дані, реагувати на користувацькі дії тощо;

2) макети та компонування. Qt Widgets надає потужні інструменти для керування розташуванням та компонуванням віджетів на екрані. Можна використовувати різні макети, такі як горизонтальний, вертикальний, сітчастий та інші, щоб автоматично розподілити та вирівняти віджети у вікні. Це дозволяє легко створювати гнучкі і адаптивні інтерфейси користувача;

3) подієва модель та сигнали-слоти. Qt Widgets заснований на подієвій моделі, яка дозволяє обробляти події, такі як натискання кнопок, переміщення миші, зміни значень та інші дії користувачів. Крім того, сигнали та слоти

механізм Qt дозволяє пов'язувати події з обробниками сигналів, що забезпечує гнучку та ефективну комунікацію між віджетами та компонентами програми;

4) малювання та графіка. Qt Widgets пропонує можливості для малювання на віджетах, створення графіків, діаграм, анімацій та інших графічних елементів. Можна використовувати класи і функції Qt для створення малюнків, відображення графічних даних і створення кастомних елементів управління за допомогою стандартних засобів малювання;

5) міжпоточність. Qt Widgets надає інструменти та механізми для роботи з багатопоточністю у додатках. Можна створювати окремі потоки для виконання завдання, а також використовувати засоби синхронізації та комунікації між потоками, щоб забезпечити безпеку та ефективність роботи з віджетами у багатопотоковому середовищі;

6) міжплатформність. Qt Widgets є міжплатформним фреймворком, що означає, що можна розробляти програми на одній платформі (наприклад, Windows) і без проблем запускати їх на інших платформах (наприклад, Linux або MacOS). Qt автоматично обробляє відмінності зовнішнього вигляду та поведінки віджетів для кожної платформи, забезпечуючи однаковий досвід користувача;

7) стилі та теми. Qt Widgets дозволяє легко налаштовувати зовнішній вигляд та оформлення віджетів за допомогою стилів та тем. Можна обирати готові стилі Qt або створити свій власний, щоб надати своїй додатку унікального та індивідуального вигляду;

8) міжмовна підтримка. Qt Widgets забезпечує підтримку міжмовних програм, що дозволяє локалізувати інтерфейс різними мовами. Можна використовувати файли перекладу (.ts) та інструменти Qt для локалізації та керування перекладами;

9) інструменти для розробки інтерфейсу користувача. Qt Widgets пропонує інтегрований редактор інтерфейсу користувача, який дозволяє візуально створювати і налаштовувати інтерфейс програми. Можна додавати

віджети, налаштовувати властивості та розміщення, а також встановлювати обробники подій без необхідності написання коду вручну;

10) можливості тестування. Qt Widgets забезпечує інструменти для автоматизованого тестування інтерфейсу користувача. Можна створювати та запускати тестові сценарії, щоб перевірити роботу інтерфейсу та виявити потенційні проблеми;

11) взаємодія з іншими модулями Qt. Qt Widgets інтегрується з іншими модулями Qt, такими як Qt Core, Qt Network, Qt SQL та багатьма іншими. Це дозволяє використовувати функціональність цих модулів разом з віджетами, розширюючи можливості програми.

Отже, Qt Widgets надає розробникам безліч інструментів і можливостей для створення гнучкого, потужного і крос-платформного інтерфейсу користувача. Він є основним модулем для розробки графічного інтерфейсу в додатках на основі Qt і C++, надаючи широкі можливості для створення професійних і сучасних інтерфейсів.

2.5 Робота з файлами в Qt. Клас QFile

Клас QFile є частиною фреймворку Qt і надає зручні засоби роботи з файлами в системі файлів. Він дозволяє відкривати, створювати, записувати, читати та видаляти файли.

Приклад програмного коду для роботи з файлами зображено на рисунку 2.3.

```
void MainWindow::on_pushButton_2_clicked() {
    QString txt, _split = get_split(ui);
    for (int i = 0; i < csvModel->rowCount(); i++) {
        for (int j = 0; j < csvModel->columnCount(); j++) {
            txt += csvModel->data(csvModel->index(i,j)).toString();
            if (j != csvModel->columnCount() - 1)
                txt += _split;
        }
        txt += "\n";
    }
    QFile file(ui->lineEdit->text());
    if (!file.open(QIODevice::WriteOnly))
        Show_Error();
    else {
        QTextStream out(&file);
        out << txt;
        file.close();
        QMessageBox::about(this, "Збереження", "Файл " + ui->lineEdit->text() + " успішно збережений");
    }
}
```

Рисунок 2.3 – Приклад програмного коду для роботи з файлами

Докладний опис основних можливостей класу QFile представлений нижче.

Відкриття файлу:

– для відкриття файлу в класі QFile використовується метод open(). Він приймає прапори, які визначають режим відкриття файлу;

– QIODevice::ReadOnly. відкриває файл лише для читання;

– QIODevice::WriteOnly. відкриває файл лише для запису;

– QIODevice::Append. відкриває файл для додавання даних до кінця;

– QIODevice::ReadWrite. відкриває файл для читання та запису;

– метод open() повертає true, якщо файл успішно відкритий, та false в іншому випадку.

Закриття файлу:

– після завершення роботи з файлом його слід закрити за допомогою методу close();

– метод close() звільняє ресурси, пов'язані з файлом, та завершує операції читання та запису.

Читання даних із файлу:

– для читання даних із файлу можна використовувати методи класу QFile, такі як readAll() і readLine();

– метод readAll() зчитує всі дані з файлу та повертає їх у вигляді екземпляра класу QByteArray;

– метод readLine() зчитує один рядок з файлу і повертає її у вигляді екземпляра класу QByteArray.

Запис даних у файл:

– для запису даних у файл можна використовувати методи класу QFile, такі як write() і writeData();

– метод write() дозволяє записувати дані у файл із екземпляра класу QByteArray або QString;

– метод writeData() дозволяє записувати дані у файл із покажчика на буфер та кількість байт для запису.

Переміщення покажчика читання/запису:

– клас QFile дозволяє переміщати покажчик читання/запису у файлі за допомогою методів seek() та pos();

– метод seek() переміщає покажчик у задану позицію у файлі;

– метод pos() повертає поточну позицію покажчика читання/запису.

Управління файлами:

– клас QFile надає методи для роботи з файлами, такі як rename() для перейменування файлу, remove() для видалення файлу та exists() для перевірки існування файлу.

Обробка помилок:

– клас QFile надає функціональність обробки помилок, що виникають під час роботи з файлами;

– метод error() повертає код помилки, а метод errorString() повертає рядкове подання помилки.

Це лише деякі з основних можливостей класу QFile. Він надає потужний і зручний інтерфейс для роботи з файлами Qt і може бути використаний в різних сценаріях розробки програмного забезпечення.

2.6 QStandardItemModel

Клас QStandardItemModel є частиною фреймворку Qt і представляє модель даних, яка використовується для зберігання та управління ієрархічними або плоскими даними у вигляді стандартних елементів (QStandardItem). Він надає зручний інтерфейс для роботи з даними, а також інтеграцію з Qt-віджетами, такими як QTableView і QTreeView.

Використання QStandardItemModel зображено на рисунку 2.4.

```
QStandardItemModel* update_table(QStandardItemModel* csvModel, Ui::MainWindow *ui, QString file_path, QString _split) {
    delete csvModel;
    csvModel = new QStandardItemModel();
    QFile file(file_path);
        if (!file.open(QFile::ReadOnly | QFile::Text)) {
            Show_Error();
        } else {
            QTextStream in(&file);
            while (!in.atEnd())
            {
                QString line = in.readLine();
                QList<QStandardItem *> standardItemsList;
                for (QString item : line.split(_split)) {
                    standardItemsList.append(new QStandardItem(item));
                }
                csvModel->insertRow(csvModel->rowCount(), standardItemsList);
            }
            file.close();
        }
    ui->tableView->setModel(csvModel);
    return csvModel;
}
```

Рисунок 2.4 – Використання QStandardItemModel

Створення моделі:

– для створення екземпляра QStandardItemModel можна використовувати конструктор без параметрів або конструктор із зазначенням числа рядків та стовпців;

– модель за замовчуванням порожня, і елементи можуть бути додані пізніше за допомогою методів класу.

Додавання елементів:

– елементи (QStandardItem) додаються в модель за допомогою методів, таких як setItem(), setHorizontalHeaderItem() та setVerticalHeaderItem();

– метод setItem() використовується для встановлення елемента за вказаними індексами рядка та стовпця;

– методи setHorizontalHeaderItem() та setVerticalHeaderItem() використовуються для встановлення елементів заголовків рядків та стовпців відповідно.

Отримання елементів:

– елементи з моделі можна отримати за допомогою методів, таких як item(), horizontalHeaderItem() та verticalHeaderItem();

– метод item() повертає покажчик на елемент за вказаними індексами рядка та стовпця;

– методи horizontalHeaderItem() та verticalHeaderItem() повертають елементи заголовків рядків і стовпців відповідно.

Встановлення даних:

– дані можуть бути встановлені для елементів за допомогою методу setData();

– метод setData() приймає індекс елемента та значення, яке потрібно встановити;

– значення може бути будь-яким типом даних, підтримуваним QVariant.

Отримання даних:

– дані елементів можна одержати за допомогою методу data();

– метод data() приймає індекс елемента та повертає значення елемента у вигляді QVariant.

Редагування елементів:

– методи setData() та data() також можуть бути використані для редагування елементів;

– зміна даних елемента автоматично викликає оновлення пов'язаних із ним віджетів.

Ієрархічні дані:

– qStandardItemModel дозволяє організовувати дані в ієрархічну структуру за допомогою методів appendRow(), insertRow() та removeRow();

– методи appendRow() та insertRow() використовуються для додавання дочірніх елементів до батьківського елементу;

– метод removeRow() видаляє зазначений рядок разом із його дочірніми елементами.

Управління розмірами моделі:

– методи rowCount() та columnCount() повертають кількість рядків та стовпців у моделі;

– методи setRowCount() та setColumnCount() використовуються для встановлення числа рядків та стовпців у моделі.

Інтеграція з віджетами Qt: QStandardItemModel може бути пов'язаний з віджетами Qt, такими як QTableView і QTreeView, для відображення даних і забезпечення взаємодії користувача.

Це лише деякі з основних можливостей класу QStandardItemModel. Він надає потужний і зручний інтерфейс для роботи з даними ієрархічної або плоскої структури та є інструментом, що часто використовується при розробці додатків з використанням Qt.

2.7 QMessageBox

Клас QMessageBox є частиною фреймворку Qt і надає діалогове вікно для відображення повідомлень та запитів користувача. Він використовується для виведення інформаційних повідомлень, попереджень, помилок та запитів підтвердження. QMessageBox надає зручний інтерфейс для створення та налаштування різних типів діалогових вікон.

Приклад використання QMessageBox зображено на рисунку 2.5.

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <QFile>
#include <QMessageBox>
#include <QPlainTextEdit>
#include <QTextStream>
#include <QString>

void Show_Error() {
    QMessageBox().critical(0, "Error", "File not exists");
}
```

Рисунок 2.5 – Приклад використання QMessageBox

На рисунку 2.6 відображається приклад повідомлення, яке було створено за допомогою QMessageBox.

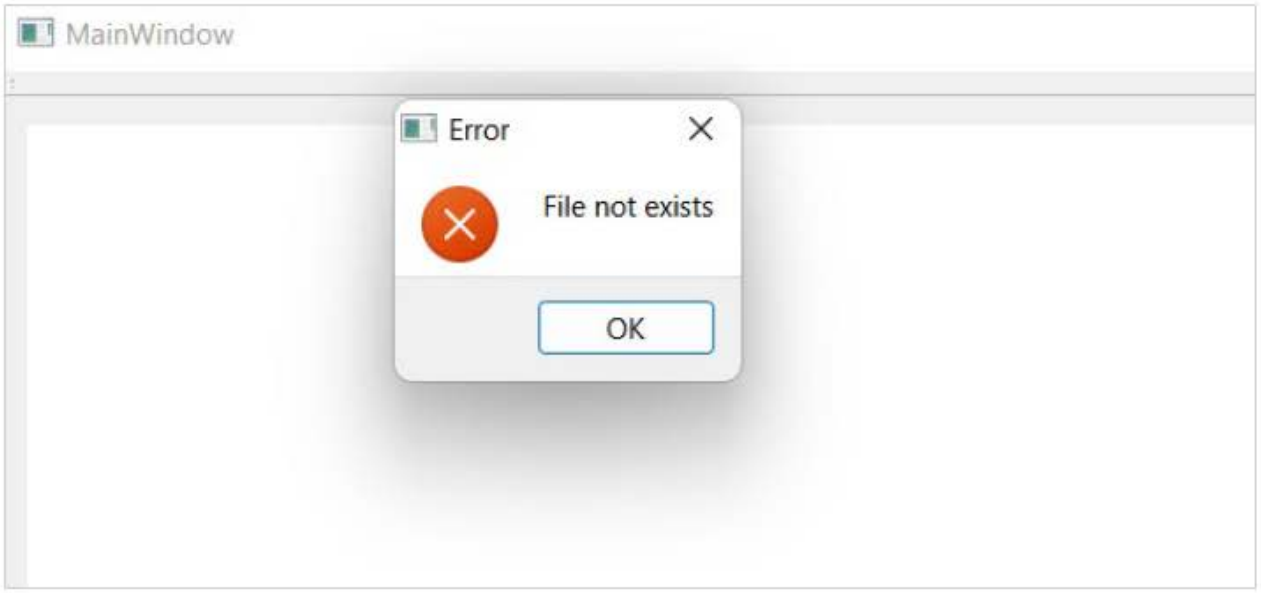


Рисунок 2.6 – Відображення повідомлення з помилкою

Створення та налаштування діалогових вікон:

– для створення екземпляра QMessageBox можна використовувати конструктор без параметрів або конструктор із зазначенням батьківського віджету;

– клас QMessageBox надає різні методи для налаштування діалогових вікон, такі як setText(), setInformativeText(), setDetailedText() тощо;

– методи setText(), setInformativeText() та setDetailedText() дозволяють встановити текст основного повідомлення, додаткової інформації та докладного тексту відповідно.

Типи діалогових вікон:

– QMessageBox підтримує різні типи діалогових вікон, які можуть бути використані в залежності від контексту та намірів розробника. Деякі з типів, що підтримуються, включають;

– інформаційне повідомлення (QMessageBox::Information);

– попередження (QMessageBox::Warning);

– помилка (QMessageBox::Critical);

– питання з вибором (QMessageBox::Question);

– метод setIcon() використовується для встановлення іконки, що відповідає типу діалогового вікна.

Кнопки та дії користувача:

– qMessageBox надає методи для керування кнопками та діями користувача в діалоговому вікні;

– метод addButton() дозволяє додати кнопку користувача;

– метод setStandardButtons() дозволяє встановити стандартні кнопки для діалогового вікна, такі як «ОК», «Скасування» тощо;

– метод exec() запускає діалогове вікно та повертає результат вибору користувача.

Обробка результату вибору користувача:

– після виконання діалогового вікна можна використовувати методи result() або clickedButton() для отримання результату вибору користувача;

– метод result() повертає результат як стандартного коду кнопки;

– метод clickedButton() повертає покажчик на кнопку.

Локалізація та стилізація:

– QMessageBox підтримує можливість локалізації тексту кнопок та повідомлень за допомогою файлів перекладів Qt;

– також можна використовувати стилі для налаштування зовнішнього вигляду діалогових вікон за допомогою таблиць стилів Qt.

Додаткові можливості: клас QMessageBox також надає інші методи налаштування діалогових вікон, такі як setWindowTitle(), setWindowIcon() і setDetailedText().

Це лише деякі з основних можливостей класу QMessageBox. Він надає гнучку та зручну функціональність для створення та керування діалоговими вікнами з повідомленнями та запитами користувачеві у додатках, розроблених з використанням Qt.

2.8 QString

Клас QString є частиною фреймворку Qt і є рядковим типом даних. Він використовується для роботи з текстовими даними та надає безліч функцій та операцій для зручної роботи з рядками.

Приклад використання QString наведено на рисунку 2.7.

```
QStandardItemModel* update_table(QStandardItemModel* csvModel, Ui::MainWindow *ui, QString file_path, QString _split) {
    delete csvModel;
    csvModel = new QStandardItemModel();
    QFile file(file_path);
        if (!file.open(QFile::ReadOnly | QFile::Text)) {
            Show_Error();
        } else {
            QTextStream in(&file);
            while (!in.atEnd())
            {
                QString line = in.readLine();
                QList<QStandardItem *> standardItemsList;
                for (QString item : line.split(_split)) {
                    standardItemsList.append(new QStandardItem(item));
                }
                csvModel->insertRow(csvModel->rowCount(), standardItemsList);
            }
            file.close();
        }
    ui->tableView->setModel(csvModel);
    return csvModel;
}
```

Рисунок 2.7 – Приклад використання QString

Створення та ініціалізація рядків:

– для створення об'єкта QString можна використовувати різноманітні конструктори. Наприклад, конструктор без параметрів створює порожній рядок, а конструктор з параметром типу const char дозволяє ініціалізувати рядок за допомогою літералу або символьного масиву;

– також можна використовувати оператор присвоювання або методи setString() та setChar() для встановлення значення рядка.

Отримання інформації про рядок:

– методи size() та length() повертають довжину рядка;

– методи isEmpty() та isNull() перевіряють, чи є рядок порожнім або нульовим;

– метод toStdString() дозволяє перетворити рядок на стандартний рядок типу std::string.

Маніпуляції з рядками:

– клас QString надає методи для конкатенації рядків, такі як append(), prepend() і operator+=, які дозволяють додавати текст до кінця або початку рядка;

– методи remove() та truncate() використовуються для видалення символів з рядка;

– методи insert() і replace() дозволяють вставляти чи замінювати підрядки у рядку.

Порівняння та пошук у рядках:

– методи compare() та operator== використовуються для порівняння рядків;

– метод contains() дозволяє перевірити, чи містить рядок певний підрядок;

– методи indexOf() та lastIndexOf() використовуються для пошуку позиції першого та останнього входження підрядка в рядок.

Зміна регістру та форматування рядків:

– методи toUpper() та toLower() дозволяють змінювати регістр символів у рядку;

– методи trimmed() та simplified() використовуються для видалення прогалин та спрощення форматування рядка.

Поділ та об'єднання рядків:

– метод split() дозволяє розділити рядок на список підрядків на основі заданого роздільника;

– метод join() використовується для об’єднання списку рядків в один рядок із використанням роздільника.

Форматування та числові операції:

– методи arg() та sprintf() дозволяють форматувати рядки, вставляючи значення змінних у певні позиції;

– клас QString надає методи для конвертації чисел у рядки та навпаки, такі як number(), setNum() та toInt().

Це лише деякі з основних можливостей класу QString. Він має багатий набір методів для роботи з текстовими даними, забезпечуючи зручність та ефективність у розробці додатків, що використовують рядки.

2.9 Висновок до другого розділу

У цьому розділі було проведено аналіз різних засобів розробки програмного забезпечення взаємодії з електронними таблицями.

По-перше, було розглянуто основні програмні інструменти, які можна використовувати розробки такого програмного забезпечення. Було зроблено вибір на основі їхньої функціональності, гнучкості та популярності серед розробників.

Далі були детально розглянуті деякі ключові компоненти Qt Framework, такі як Qt Creator C++ і Qt Widgets. Qt Framework надає потужний набір інструментів та бібліотек для розробки крос-платформних додатків із графічним інтерфейсом користувача.

Потім були розглянуті класи QFile і QStandardItemModel. Клас QFile надає функціональність для роботи з файлами, включаючи читання, запис та керування файловою системою. Клас QStandardItemModel представляє модель даних для використання у віджетах Qt, таких як таблиці, та забезпечує зручні методи для маніпуляції даними всередині моделі.

Також було розглянуто клас QMessageBox, який надає можливість створення діалогових вікон із повідомленнями та запитами користувачеві. Він дозволяє легко взаємодіяти з користувачем, виводячи повідомлення, попередження чи запити та опрацьовуючи результати.

Нарешті, був детально розглянутий клас QString, що представляє рядковий тип даних Qt. Він має широкий набір функцій для роботи з текстовими даними, включаючи операції конкатенації, пошуку, форматування та маніпуляцій з рядками.

Аналіз цих інструментів та класів дозволив визначити основні засоби та технології, які можна використовувати для розробки програмного забезпечення, що взаємодіє з електронними таблицями.

Загальний висновок з цього розділу полягає в тому, що вибір Qt Framework та пов'язаних з ним інструментів, таких як Qt Creator C++, Qt Widgets, QFile, QStandardItemModel, QMessageBox та QString, забезпечує розробникам потужні засоби для створення програмного забезпечення, що взаємодіє з електронними таблицями.

РОЗДІЛ 3 РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ВЗАЄМОДІЇ З ЕЛЕКТРОННИМИ ТАБЛИЦЯМИ

3.1 Архітектура та структура проекту

Структура та архітектура проекту можуть змінюватись в залежності від переваг та практик розробника, а також від специфіки додатка. Однак, на основі наданого коду, виходячи зі стандартних практик розробки на Qt, можна припустити наступну структуру та архітектуру проекту.

– файли проекту;

– mainwindow.h та mainwindow.cpp. Файли, які відповідають за реалізацію головного вікна програми та її логіку;

– main.cpp. Файл, який містить точку входу в програму та функцію main();

– mainwindow.ui. Файл інтерфейсу користувача, створений Qt Designer, який визначає зовнішній вигляд головного вікна програми.

Архітектура проекту:

– model-View-Controller (MVC). Проект може використовувати архітектурний патерн MVC для розділення логіки програми та подання даних. В даному випадку клас QStandardItemModel, що використовується для моделювання даних таблиці, може розглядатися як модель, головне вікно (MainWindow) як подання, а функції обробки подій (on_pushButton_clicked та on_pushButton_2_clicked) як контролери.

Клас MainWindow:

– основний клас, який відповідає за головне вікно програми;

– створюється екземпляр класу Ui::MainWindow, який надає доступ до елементів інтерфейсу, визначених у файлі mainwindow.ui;

– містить методи та обробники подій для взаємодії з користувачем та управління додатком;

– має приватні члени даних, такі як csvModel (QStandardItemModel) та ui (Ui::MainWindow), що використовуються для зберігання моделі таблиці та доступу до елементів інтерфейсу відповідно.

Функції допоміжних операцій:

– show_Error(). Функція, яка відображає діалогове вікно з повідомленням про помилку;

– get_split(). функція, яка визначає роздільник даних залежно від стану радіокнопки;

– update_table(). Функція, яка оновлює модель таблиці (csvModel) на основі даних із файлу CSV.

Інтерфейс користувача (mainwindow.ui):

– визначає зовнішній вигляд головного вікна програми;

– включає віджети, такі як кнопки (QPushButton), текстові поля (QLineEdit), радіокнопки (QRadioButton), мітки (QLabel) та таблицю (QTableView);

– містить розмітку інтерфейсу, властивості та сигнали-слоти для зв'язку з логікою програми.

Файли ресурсів:

– програма може використовувати файли ресурсів (.qrc), у яких зберігаються зовнішні ресурси, такі як іконки, зображення чи шрифти. Однак, у наданому коді файл ресурсів не використовується.

Це лише передбачувана структура та архітектура проекту, заснована на наданому коді та загальноприйнятих практиках розробки на Qt. Реальна структура проекту може відрізнятися залежно від конкретної реалізації та потреби додатку.

Структура проекту зображена на рисунку 3.1.

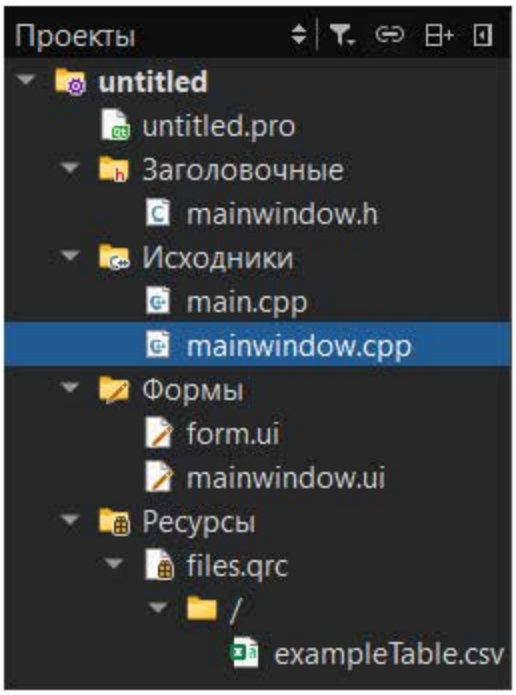


Рисунок 3.1 – Структура проекту

3.2 Опис роботи програмного забезпечення

Процес роботи проекту, заснованого на C++ коді, можна описати так:

Запуск програми:

– при запуску програми створюється екземпляр класу MainWindow, який представляє головне вікно програми;

– створюється екземпляр класу Ui..MainWindow, який надає доступ до елементів інтерфейсу, визначених у файлі mainwindow.ui;

– встановлюється зовнішній вигляд головного вікна, завантажуючи та застосовуючи визначений у файлі mainwindow.ui макет інтерфейсу користувача;

– створюється екземпляр QStandardItemModel (csvModel), який використовуватиметься для моделювання даних таблиці.

Взаємодія з користувачем:

– користувач може взаємодіяти з головним вікном програми, взаємодіючи з елементами інтерфейсу, такими як кнопки, текстові поля, радіокнопки та таблиця;

– при натисканні на кнопку «Відкрити» (pushButton) викликається обробник події on_pushButton_clicked().

Обробка події натискання кнопки "Відкрити" (pushButton):

– виходить стан радіокнопки (radioButton) визначення роздільника даних у файлі CSV;

– викликається функція get_split() для отримання роздільника з урахуванням стану радіокнопки;

– викликається функція update_table(), яка оновлює модель таблиці (csvModel) на основі даних із вказаного CSV файлу;

– всередині функції update_table();

– видалити попередню модель таблиці (csvModel), якщо вона існує;

– створюється новий екземпляр QStandardItemModel (csvModel), який використовуватиметься для нової моделі таблиці;

– відкривається вказаний CSV файл за допомогою QFile;

– якщо файл не може бути відкритий, викликається функція Show_Error(), яка відображає діалогове вікно з повідомленням про помилку;

– якщо файл успішно відкритий, дані з файлу зчитуються за допомогою QTextStream;

– кожен рядок розбивається на елементи з допомогою роздільника, отриманого з функції get_split();

– створюється список елементів QStandardItem для кожного рядка та заповнюється даними з файлу;

– список елементів додається до моделі таблиці (csvModel);

– файл закривається.

Відображення даних у таблиці:

– оновлена модель таблиці (csvModel) встановлюється як модель QTableView, пов’язаного з таблицею в графічному інтерфейсі;

– тепер дані з файлу CSV відображаються у таблиці.

Взаємодія з користувачем та збереження даних:

– користувач може внести зміни до даних у таблиці або виконати інші операції;

– при натисканні на кнопку «Зберегти» (pushButton_2) викликається обробник події on_pushButton_2_clicked().

Обробка події натискання кнопки «Зберегти» (pushButton_2):

– виходить стан радіокнопки (radioButton) визначення роздільника даних у файлі CSV;

– викликається функція get_split() для отримання роздільника з урахуванням стану радіокнопки;

– створюється порожній рядок txt, який міститиме дані для збереження;

– прохід по кожному рядку таблиці та кожному стовпцю;

– дані з комірки виходять із використанням методу data() моделі таблиці (csvModel);

– кожен елемент додається до рядка txt із роздільником;

– перехід на новий рядок після обробки кожного рядка таблиці;

– створюється екземпляр класу QFile із вказаним користувачем ім'ям файлу для збереження;

– якщо файл не може бути відкритим для запису, викликається функція Show_Error();

– якщо файл успішно відкритий, дані з рядка txt записуються у файл за допомогою QTextStream;

– файл закривається;

– спливає діалогове вікно, що повідомляє про успішне збереження файлу.

Таким чином, процес роботи проекту включає запуск програми, взаємодія з користувачем через елементи інтерфейсу, обробку подій, оновлення моделі таблиці, відображення даних в таблиці і збереження даних у файл.

3.3 Процес проектування користувацького інтерфейсу

При проектуванні інтерфейсу користувача в проекті, заснованому на наданому коді, використовувався підхід, характерний для розробки з використанням фреймворку Qt.

Візуальне проектування:

– був використаний інструмент Qt Designer для візуального проектування інтерфейсу користувача;

– в результаті візуального проектування було створено файл mainwindow.ui, в якому визначено елементи інтерфейсу, їх розташування та властивості.

Головне вікно:

– головне вікно програми є екземпляр класу QMainWindow;

– зовнішній вигляд головного вікна визначено у файлі mainwindow.ui та включає заголовок вікна, панель інструментів та область вмісту;

– панель інструментів містить дві кнопки. «Відкрити» (pushButton) та «Зберегти» (pushButton_2);

– область вмісту містить елементи для введення та відображення даних. QLineEdit для введення шляху до файлу та QTableView для відображення таблиці.

Розміщення елементів:

– у файлі mainwindow.ui елементи інтерфейсу розміщені у вигляді віджетів усередині лайаутів;

– для головного вікна використано QVBoxLayout, який вертикально розміщує елементи;

– верхня частина вікна містить QLabel для відображення підказки та QLineEdit для введення шляху до файлу;

– під ними розміщено QTableView для відображення таблиці;

– внизу вікна розміщена панель інструментів із кнопками «Відкрити» та «Зберегти».

Робота з елементами інтерфейсу:

– кожен елемент інтерфейсу має унікальний ідентифікатор (objectName), який дозволяє отримати доступ до програмно;

– у коді проекту використовують клас Ui..MainWindow для доступу до елементів інтерфейсу з файлу mainwindow.ui;

– деякі елементи мають певні властивості, такі як підказка для QLabel (label), стан для радіокнопки (radioButton) та інші.

Обробка подій:

– для елементів, таких як кнопки «Відкрити» та «Зберегти», визначені обробники подій (слоти);

– обробники подій викликаються при натисканні відповідних клавіш і виконують певні дії, такі як оновлення таблиці або збереження даних.

В результаті проектування інтерфейсу користувача було створено файл mainwindow.ui, що визначає зовнішній вигляд та розташування елементів інтерфейсу. Код mainwindow.cpp використовує цей файл для доступу до елементів інтерфейсу та реалізації функціональності програми.

Користувацький інтерфейс зображено на рисунку 3.2.

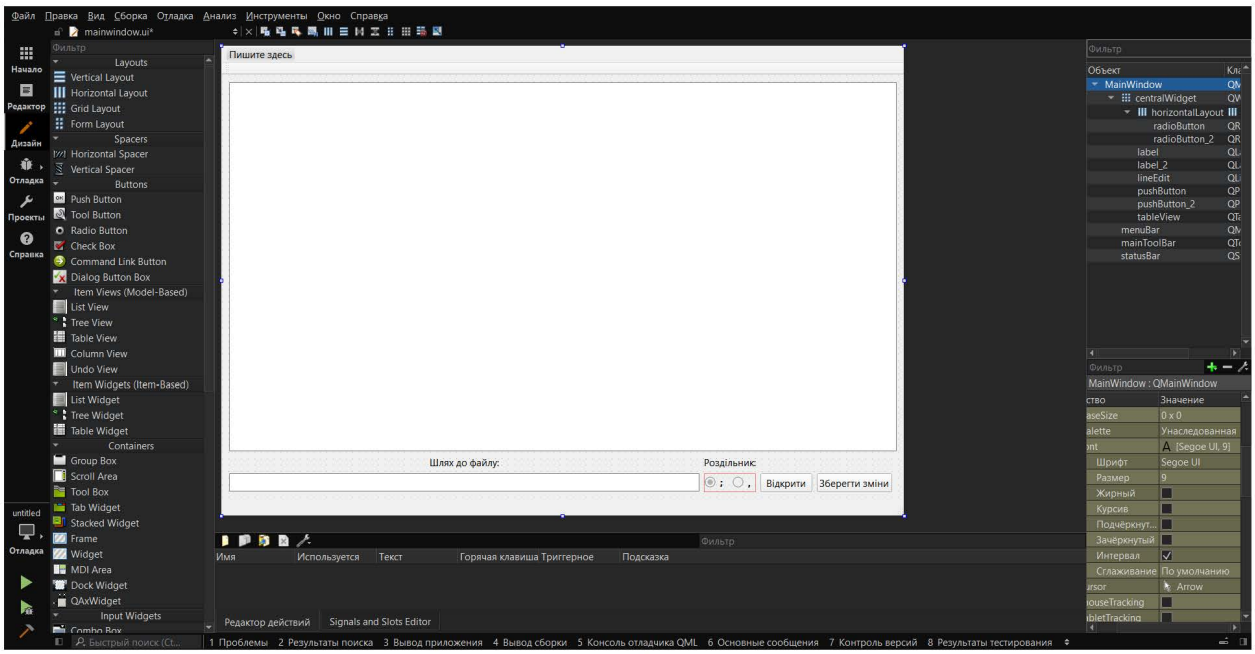


Рисунок 3.2 – Користувацький інтерфейс

3.4 Тестування програмного забезпечення для взаємодії з електронними таблицями

Головне вікно програмного забезпечення зображене на рисунку 3.3.

На рисунку 3.4 відбувається спроба відкрити файл без вказаного шляху, на що користувач отримує помилку.

На рисунку 3.5 відбувається відкриття таблиці з правильно вказаним шляхом.

Можливість редагування даних відображена на рисунку 3.6.

Функціональність збереження оновленої таблиці продемонстрована на рисунку 3.7.

Результати збереження в файловому менеджері зображені на рисунку 3.8.

Результат збереження в таблицях Excel зображено на рисунку 3.9.

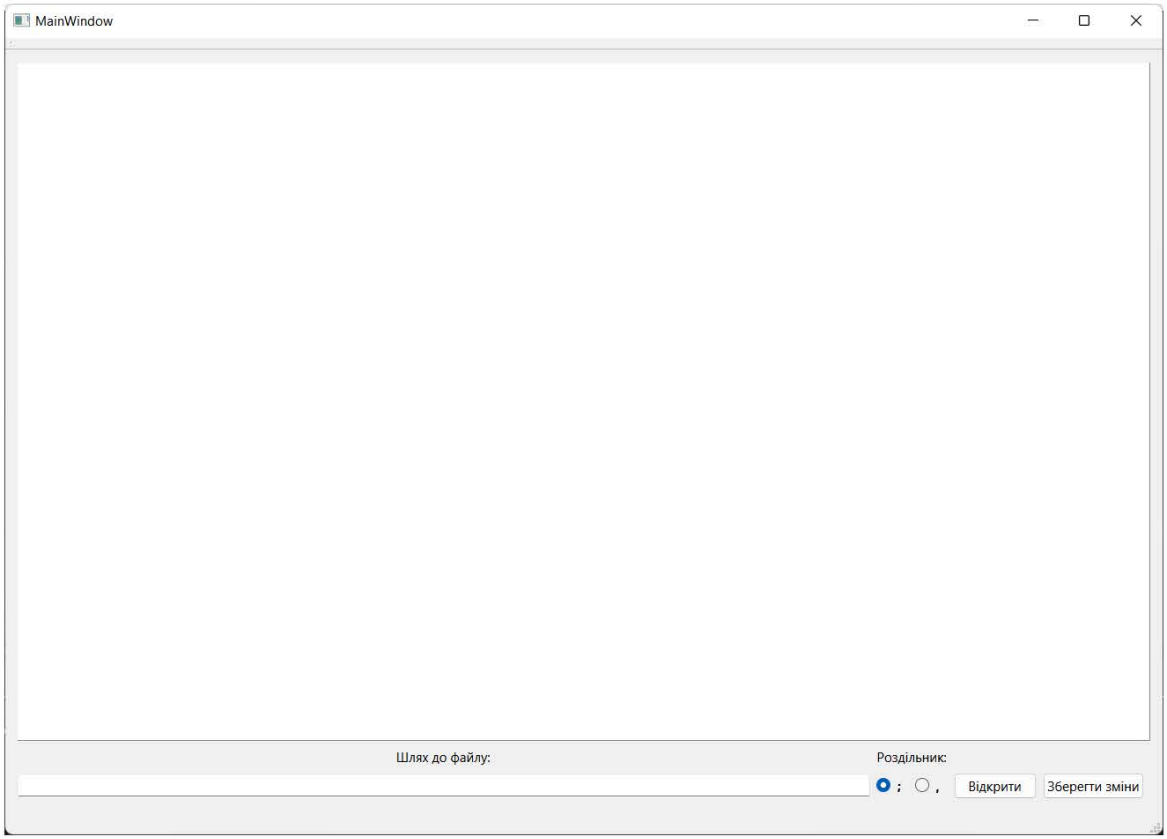


Рисунок 3.3 – Головне вікно програмного забезпечення

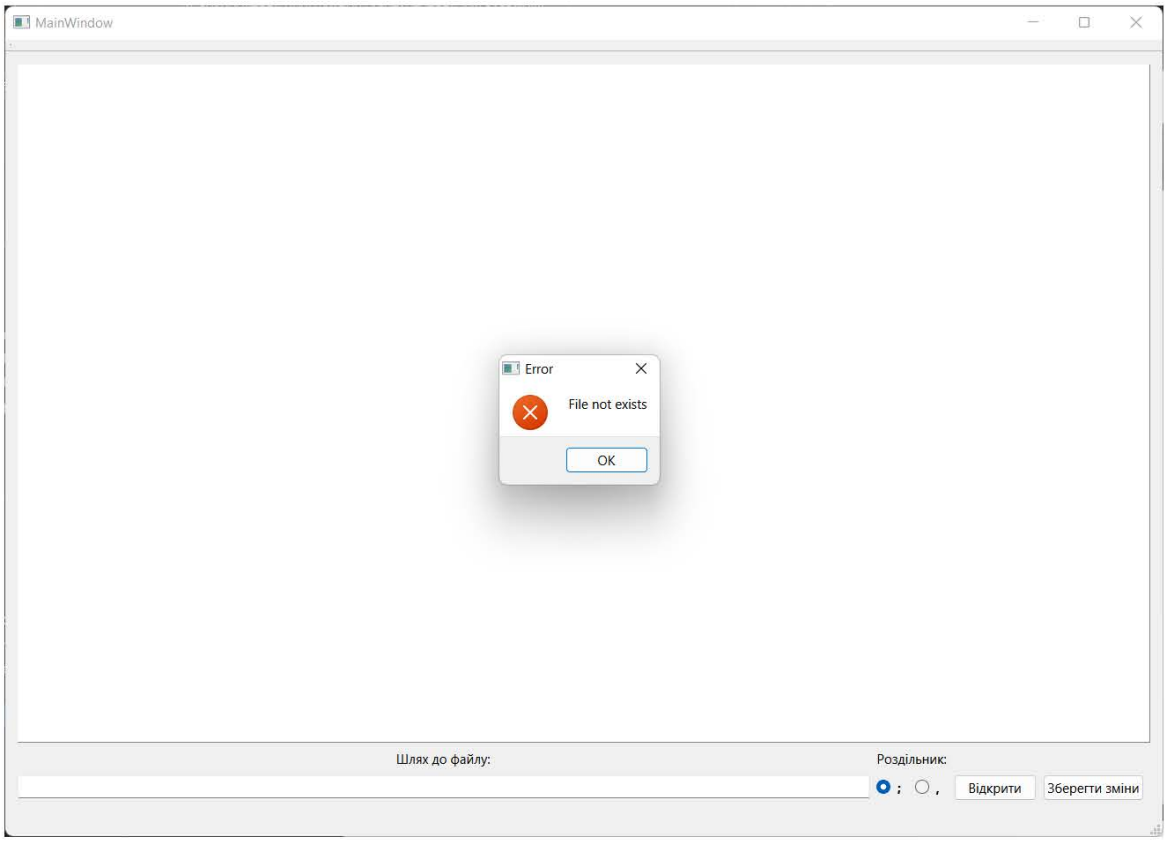
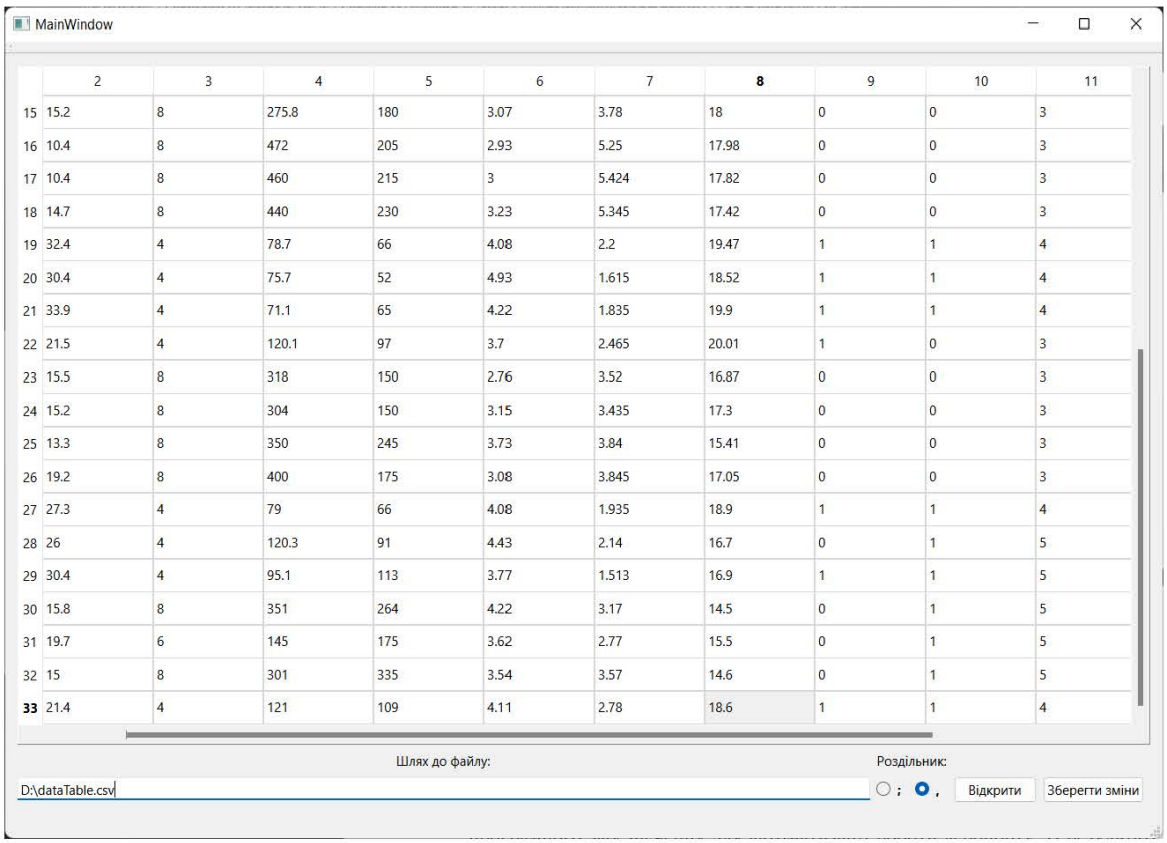
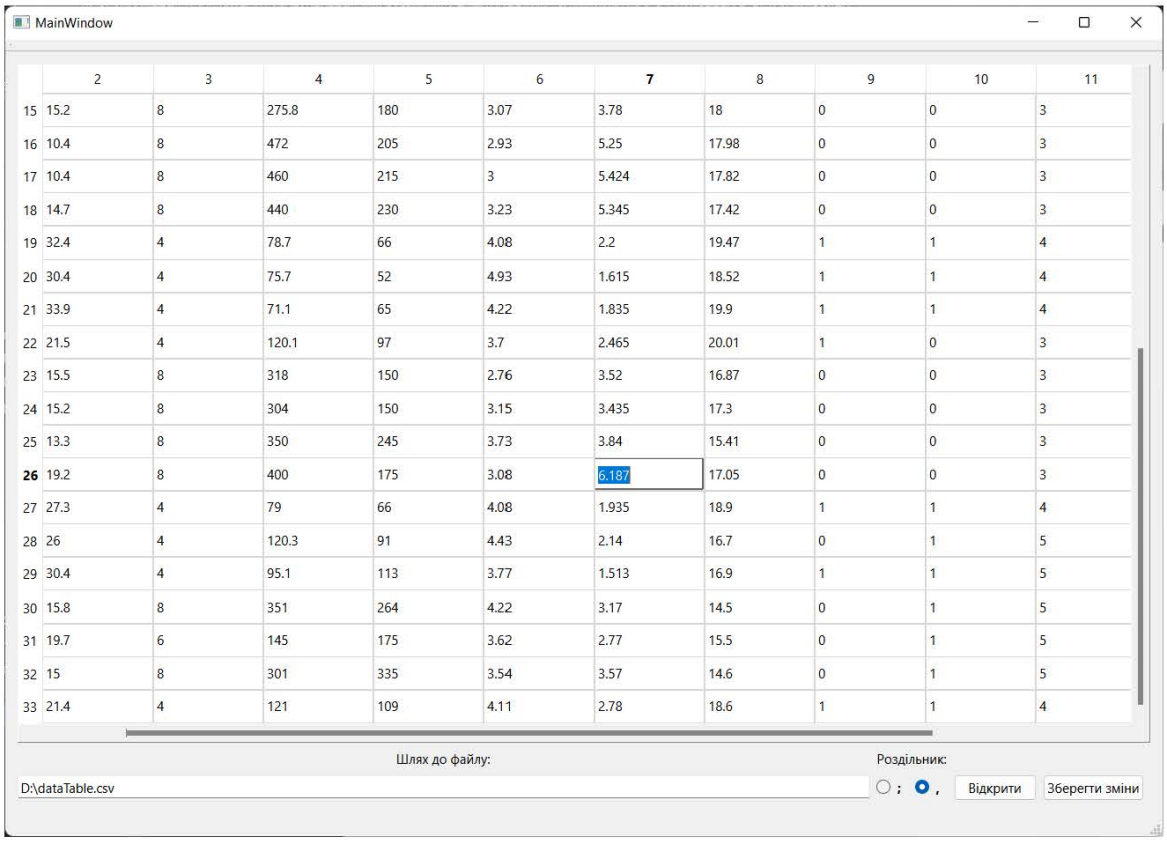


Рисунок 3.4 – Спроба відкрити файл без вказаного шляху



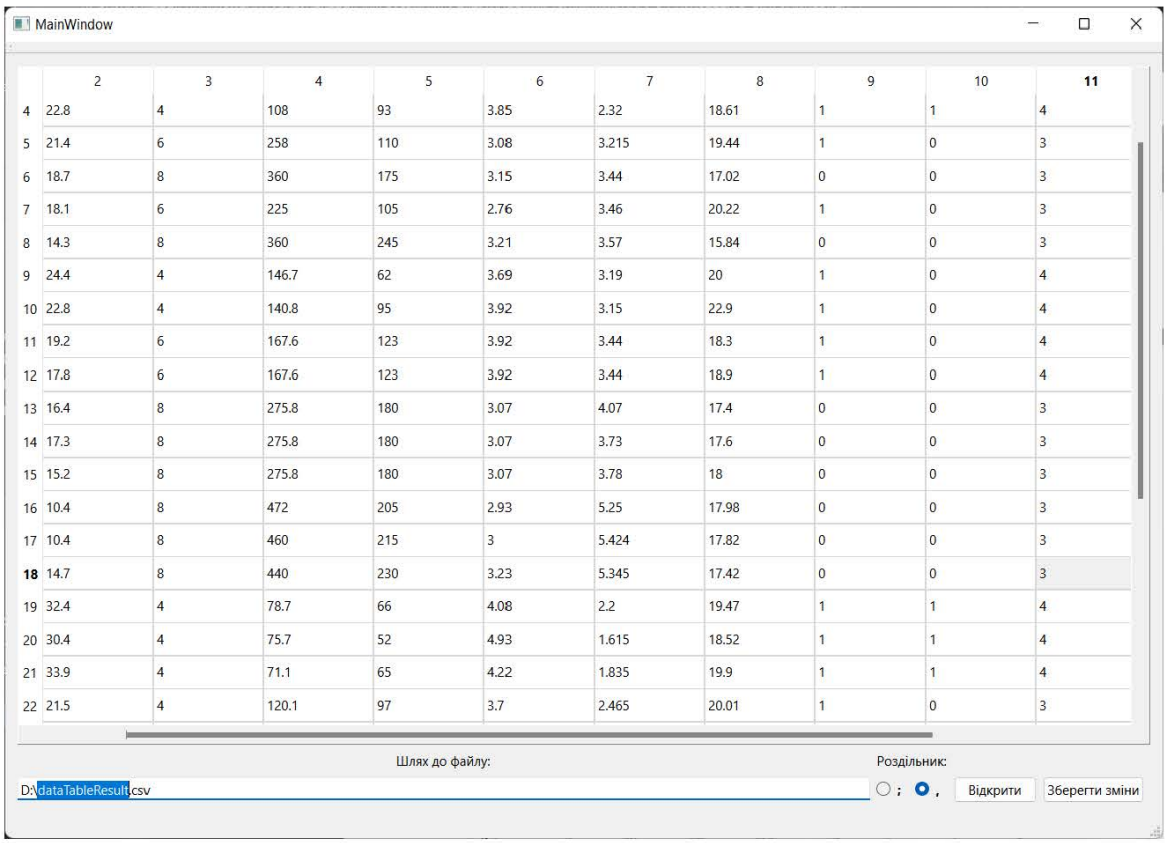
	2	3	4	5	6	7	8	9	10	11
15	15.2	8	275.8	180	3.07	3.78	18	0	0	3
16	10.4	8	472	205	2.93	5.25	17.98	0	0	3
17	10.4	8	460	215	3	5.424	17.82	0	0	3
18	14.7	8	440	230	3.23	5.345	17.42	0	0	3
19	32.4	4	78.7	66	4.08	2.2	19.47	1	1	4
20	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4
21	33.9	4	71.1	65	4.22	1.835	19.9	1	1	4
22	21.5	4	120.1	97	3.7	2.465	20.01	1	0	3
23	15.5	8	318	150	2.76	3.52	16.87	0	0	3
24	15.2	8	304	150	3.15	3.435	17.3	0	0	3
25	13.3	8	350	245	3.73	3.84	15.41	0	0	3
26	19.2	8	400	175	3.08	3.845	17.05	0	0	3
27	27.3	4	79	66	4.08	1.935	18.9	1	1	4
28	26	4	120.3	91	4.43	2.14	16.7	0	1	5
29	30.4	4	95.1	113	3.77	1.513	16.9	1	1	5
30	15.8	8	351	264	4.22	3.17	14.5	0	1	5
31	19.7	6	145	175	3.62	2.77	15.5	0	1	5
32	15	8	301	335	3.54	3.57	14.6	0	1	5
33	21.4	4	121	109	4.11	2.78	18.6	1	1	4

Рисунок 3.5 – Відкриття таблиці



	2	3	4	5	6	7	8	9	10	11
15	15.2	8	275.8	180	3.07	3.78	18	0	0	3
16	10.4	8	472	205	2.93	5.25	17.98	0	0	3
17	10.4	8	460	215	3	5.424	17.82	0	0	3
18	14.7	8	440	230	3.23	5.345	17.42	0	0	3
19	32.4	4	78.7	66	4.08	2.2	19.47	1	1	4
20	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4
21	33.9	4	71.1	65	4.22	1.835	19.9	1	1	4
22	21.5	4	120.1	97	3.7	2.465	20.01	1	0	3
23	15.5	8	318	150	2.76	3.52	16.87	0	0	3
24	15.2	8	304	150	3.15	3.435	17.3	0	0	3
25	13.3	8	350	245	3.73	3.84	15.41	0	0	3
26	19.2	8	400	175	3.08	6.187	17.05	0	0	3
27	27.3	4	79	66	4.08	1.935	18.9	1	1	4
28	26	4	120.3	91	4.43	2.14	16.7	0	1	5
29	30.4	4	95.1	113	3.77	1.513	16.9	1	1	5
30	15.8	8	351	264	4.22	3.17	14.5	0	1	5
31	19.7	6	145	175	3.62	2.77	15.5	0	1	5
32	15	8	301	335	3.54	3.57	14.6	0	1	5
33	21.4	4	121	109	4.11	2.78	18.6	1	1	4

Рисунок 3.6 – Можливість редагування даних



	2	3	4	5	6	7	8	9	10	**11**
4	22.8	4	108	93	3.85	2.32	18.61	1	1	4
5	21.4	6	258	110	3.08	3.215	19.44	1	0	3
6	18.7	8	360	175	3.15	3.44	17.02	0	0	3
7	18.1	6	225	105	2.76	3.46	20.22	1	0	3
8	14.3	8	360	245	3.21	3.57	15.84	0	0	3
9	24.4	4	146.7	62	3.69	3.19	20	1	0	4
10	22.8	4	140.8	95	3.92	3.15	22.9	1	0	4
11	19.2	6	167.6	123	3.92	3.44	18.3	1	0	4
12	17.8	6	167.6	123	3.92	3.44	18.9	1	0	4
13	16.4	8	275.8	180	3.07	4.07	17.4	0	0	3
14	17.3	8	275.8	180	3.07	3.73	17.6	0	0	3
15	15.2	8	275.8	180	3.07	3.78	18	0	0	3
16	10.4	8	472	205	2.93	5.25	17.98	0	0	3
17	10.4	8	460	215	3	5.424	17.82	0	0	3
18	14.7	8	440	230	3.23	5.345	17.42	0	0	3
19	32.4	4	78.7	66	4.08	2.2	19.47	1	1	4
20	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4
21	33.9	4	71.1	65	4.22	1.835	19.9	1	1	4
22	21.5	4	120.1	97	3.7	2.465	20.01	1	0	3

Рисунок 3.7 – Збереження оновленої таблиці

Рисунок 3.8 – Результати збереження

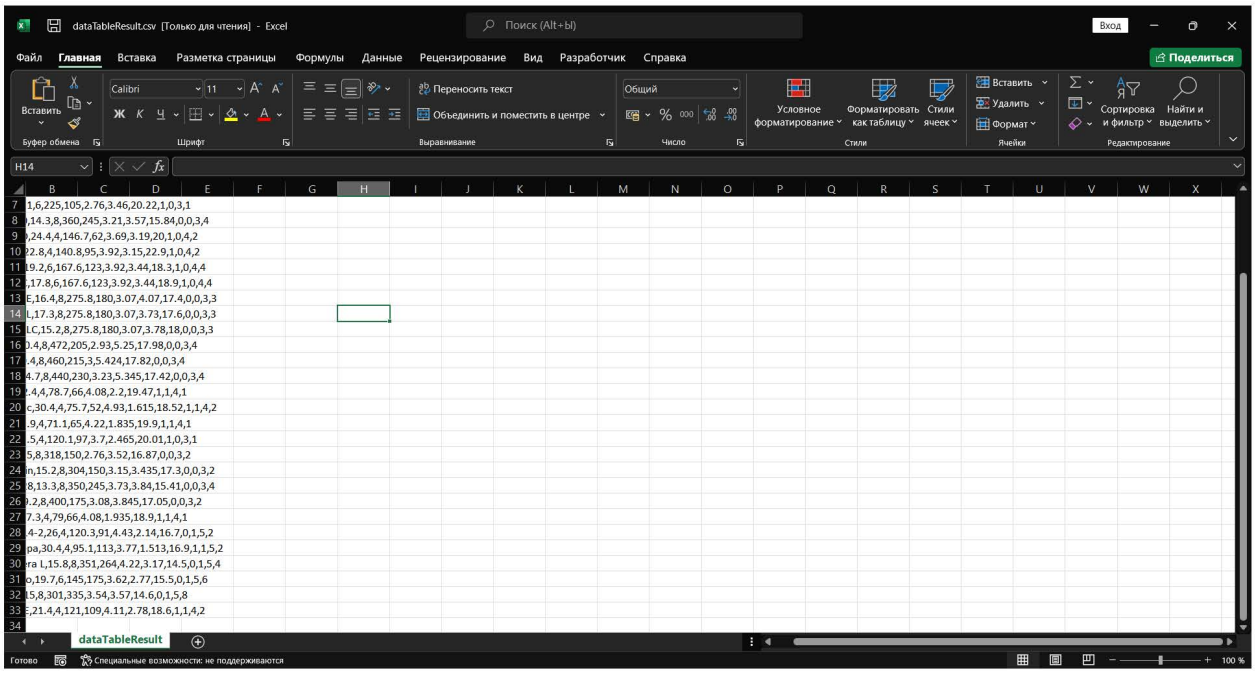


Рисунок 3.9 – Результат збереження в таблицях Excel

3.5 Висновки до третього розділу

У розділі 3 описується розробка програмного забезпечення взаємодії з електронними таблицями.

У ході розробки було розглянуто такі аспекти:

– у розділі представлено структуру проекту, заснованого на коді Qt Creator C++;

– описано основні компоненти проекту, такі як класи MainWindow та Ui_MainWindow;

– докладно описані використовувані бібліотеки, такі як Qt та його модулі QFile, QMessageBox, QPlainTextEdit, QTextStream та інші;

– описано основні функції та методи, що використовуються для обробки подій та взаємодії з користувачем;

– описано функції для відображення даних у таблиці, оновлення таблиці під час відкриття файлу та збереження даних у файлі;

– розглянуто проектування інтерфейсу користувача з використанням інструменту Qt Designer;

– описані елементи інтерфейсу, їх розташування та властивості, визначені у файлі mainwindow.ui;

– описано розміщення елементів з використанням лайаутів та роботу з елементами інтерфейсу;

– ця частина розділу присвячена тестуванню програмного забезпечення для взаємодії з електронними таблицями;

– описано методи та підходи, що використовуються для тестування функціональності програми;

– наприкінці розділу наведено висновки щодо третього розділу;

– зазначається, що розробка програмного забезпечення для взаємодії з електронними таблицями включає проектування інтерфейсу користувача, обробку подій, відображення та редагування даних у таблиці, а також збереження даних у файли;

– наголошується на важливості тестування програмного забезпечення для забезпечення його функціональності та коректної роботи.

Таким чином, розділ 3 надає докладний опис розробки програмного забезпечення для роботи з електронними таблицями, включаючи архітектуру проекту, опис роботи програми, процес проектування інтерфейсу користувача і тестування програмного забезпечення.

ВИСНОВКИ

Метою роботи було розроблення програмного забезпечення для взаємодії з електронними таблицями.

Для досягнення поставленої мети з розроблення програмного забезпечення для взаємодії з електронними таблицями були виконано такі завдання:

1) вивчення основ роботи з електронними таблицями. Це включало розуміння структури даних у таблицях, форматів файлів (XLSX, CSV), основних операцій (створення, читання, запис, редагування) і специфічних функцій, доступних у конкретних додатках для роботи з таблицями;

2) аналіз вимог користувачів. Важливо було зрозуміти, які функції та можливості користувачі очікують від розроблюваного програмного забезпечення. Це включає функції автоматизації, аналізу даних, генерації звітів, імпорту та експорту даних, фільтрації та сортування, а також підтримку специфічних форматів і розрахунків;

3) вивчення наявних інструментів і технологій. Існує безліч мов програмування, фреймворків і бібліотек, призначених для роботи з електронними таблицями. Було проведено дослідження та обрано найбільш підходящі інструменти для розроблення, з огляду на вимоги та цілі проєкту;

4) розробка архітектури програмного забезпечення. Це включало визначення модулів, компонентів і інтерфейсів програми, які забезпечуватимуть необхідну функціональність. Важливо також було врахувати аспекти безпеки, продуктивності та масштабованості розроблюваного програмного забезпечення;

5) реалізація функціональності програмного забезпечення. У цьому завданні створювався код, який дозволяє програмному забезпеченню взаємодіяти з електронними таблицями;

6) тестування та налагодження. Після реалізації програмного забезпечення було проведене тестування для перевірки його

функціональності, стабільності та відповідності вимогам. Важливо було також виявити і виправити можливі помилки та недоліки;

7) оцінка та оптимізація продуктивності. В результаті роботи програма була додатково оптимізована;

8) продуктивності. Це включає оптимізацію запитів до таблиць, поліпшення алгоритмів обробки даних і реалізацію багатопоточності або розподілених обчислень;

9) документування та підтримка. Не менш важливим було завдання зі створення документації, яка описує функціональність програмного забезпечення та інструкції з його використання. Також використовувалась підтримка й супровід програмного продукту, включно з виправленням помилок, додаванням нових функцій та оновленнями відповідно до змін у стандартах й вимогах користувачів.

Розв'язання цих завдань допомогло досягти поставленої мети розроблення програмного забезпечення для взаємодії з електронними таблицями та створити повнофункціональне, надійне й ефективне програмне рішення для роботи з даними.

Об'єктом дослідження були процеси роботи програмного забезпечення для взаємодії з електронними таблицями.

Предметом дослідження було апаратно-програмне забезпечення для розроблення програмного забезпечення для взаємодії з електронними таблицями.

Практичне значення одержаних результатів полягає у підвищенні якості взаємодії з електронними таблицями.

Результатами роботи є програмне забезпечення для взаємодії з електронними таблицями.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Stroustrup B. The C++ Programming Language, 2013. 1368 p.

2. Lippman S. B., Lajoie J., Moo B. E. C++ Primer, 2012. 976 p.

3. Schildt H. C++: The Complete Reference, 2012. 992 p.

4. Qt Documentation, Qt Framework documentation. URL: https://doc.qt.io/

5. Ezust A., Ezust O. An Introduction to Design Patterns in C++ with Qt, 2004. 656 p.

6. Summerfield M. Advanced Qt Programming: Creating Great Software with C++ and Qt 4, 2007. 552 p.

7. Blanchette J., Summerfield M. C++ GUI Programming with Qt 4, 2008. 752 p.

8. Mueller J. P. Mastering C++ Programming, 2018. 864 p.

9. Turner J., Gregoire M. Mastering the C++17 STL: Make full use of the standard library components in C++17, 2017. 432 p.

10. Bryant R. E., O'Hallaron D. R. Computer Systems: A Programmer's Perspective, 2015. 1104 p.

11. Meyers S. Effective C++: 55 Specific Ways to Improve Your Programs and Designs, 2014. 320 p.

12. Alexandrescu A. Modern C++ Design: Generic Programming and Design Patterns Applied, 2001. 352 p.

13. Roberts E. S. The Art and Science of C: A Library-Based Introduction to Computer Science, 1995. 912 p.

14. Koenig A., Moo B. E. Accelerated C++: Practical Programming by Example, 2010. 352 p.

15. Fowler M. Refactoring: Improving the Design of Existing Code, 2018. 448 p.

16. Martin R. C. Clean Code: A Handbook of Agile Software Craftsmanship, 2008. 464 p.

17. McConnell S. Code Complete: A Practical Handbook of Software Construction, 2004. 960 p.

18. Bloch S. Effective Java, 2017. 416 p.

19. Flanagan D. JavaScript: The Definitive Guide, 2020. 706 p.

20. Sierra K., Bates B. Head First Java, 2005. 720 p.

ДОДАТОК А

ЛІСТИНГ ПРОЕКТУВАННЯ ІНТЕРФЕЙСУ

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>MainWindow</class>
 <widget class="QMainWindow" name="MainWindow">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>1049</width>
    <height>722</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>MainWindow</string>
  </property>
  <widget class="QWidget" name="centralWidget">
   <layout class="QGridLayout" name="gridLayout">
    <item row="3" column="3">
     <widget class="QPushButton" name="pushButton">
      <property name="font">
       <font/>
      </property>
      <property name="text">
       <string>Відкрити</string>
      </property>
     </widget>
```

```
   </item>
   <item row="1" column="2">
    <widget class="QLabel" name="label">
     <property name="font">
      <font/>
     </property>
     <property name="text">
      <string>Роздільник:</string>
     </property>
     <property name="alignment">
      <set>Qt::AlignCenter</set>
     </property>
    </widget>
   </item>
   <item row="1" column="1">
    <widget class="QLabel" name="label_2">
     <property name="font">
      <font/>
     </property>
     <property name="text">
      <string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p&gt;Шлях до
файлу:&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>
     </property>
     <property name="alignment">
      <set>Qt::AlignCenter</set>
     </property>
    </widget>
   </item>
   <item row="3" column="1">
    <widget class="QLineEdit" name="lineEdit">
```

```
 <property name="font">
  <font/>
 </property>
</widget>
</item>
<item row="3" column="2">
 <layout class="QHBoxLayout" name="horizontalLayout">
  <item>
   <widget class="QRadioButton" name="radioButton">
    <property name="font">
     <font>
      <bold>true</bold>
     </font>
    </property>
    <property name="text">
     <string>;</string>
    </property>
    <property name="checked">
     <bool>true</bool>
    </property>
   </widget>
  </item>
  <item>
   <widget class="QRadioButton" name="radioButton_2">
    <property name="font">
     <font>
      <bold>true</bold>
     </font>
    </property>
    <property name="text">
```

```
      <string>,</string>
     </property>
    </widget>
   </item>
  </layout>
 </item>
 <item row="3" column="4">
  <widget class="QPushButton" name="pushButton_2">
   <property name="font">
    <font/>
   </property>
   <property name="text">
    <string>Зберегти зміни</string>
   </property>
  </widget>
 </item>
 <item row="0" column="1" colspan="5">
  <widget class="QTableView" name="tableView"/>
 </item>
</layout>
</widget>
<widget class="QMenuBar" name="menuBar">
 <property name="geometry">
  <rect>
   <x>0</x>
   <y>0</y>
   <width>1049</width>
   <height>25</height>
  </rect>
 </property>
```

```
 </widget>
 <widget class="QToolBar" name="mainToolBar">
  <attribute name="toolBarArea">
   <enum>TopToolBarArea</enum>
  </attribute>
  <attribute name="toolBarBreak">
   <bool>false</bool>
  </attribute>
 </widget>
 <widget class="QStatusBar" name="statusBar"/>
</widget>
<layoutdefault spacing="6" margin="11"/>
<resources/>
<connections/>
</ui>
```

ДОДАТОК Б

ЛІСТИНГ ПРОГРАМНОГО КОДУ

```
//main.cpp
//mainwindow.cpp

#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <QFile>
#include <QMessageBox>
#include <QPlainTextEdit>
#include <QTextStream>
#include <QString>

void Show_Error() {
   QMessageBox().critical(0, "Error", "File not exists");
}

QString get_split(Ui::MainWindow *ui) {
   return ui->radioButton->isChecked() ? ";" : ",";
}

QStandardItemModel* update_table(QStandardItemModel* csvModel,
Ui::MainWindow *ui, QString file_path, QString _split) {
   delete csvModel;
   csvModel = new QStandardItemModel();
   QFile file(file_path);
      if (!file.open(QFile::ReadOnly | QFile::Text)) {
         Show_Error();
```

```
    } else {
        QTextStream in(&file);
        while (!in.atEnd())
        {
            QString line = in.readLine();
            QList<QStandardItem *> standardItemsList;
            for (QString item : line.split(_split)) {
                standardItemsList.append(new QStandardItem(item));
            }
            csvModel->insertRow(csvModel->rowCount(), standardItemsList);
        }
        file.close();
    }
    ui->tableView->setModel(csvModel);
    return csvModel;
}

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    ui->label->setToolTip("Працює як в разі відкриття, так Й під час
збереження");
    csvModel = new QStandardItemModel(this);
}

MainWindow::~MainWindow() {
    delete ui;
    delete csvModel;
```

```
}

void MainWindow::on_pushButton_clicked() {
    QString _split = ui->radioButton->isChecked() ? ";" : ",";
    csvModel = update_table(csvModel, ui, ui->lineEdit->text(), _split);
}

void MainWindow::on_pushButton_2_clicked() {
    QString txt, _split = get_split(ui);
    for (int i = 0; i < csvModel->rowCount(); i++) {
        for (int j = 0; j < csvModel->columnCount(); j++) {
            txt += csvModel->data(csvModel->index(i,j)).toString();
            if (j != csvModel->columnCount() - 1)
                txt += _split;
        }
        txt += "\n";
    }
    QFile file(ui->lineEdit->text());
    if (!file.open(QIODevice::WriteOnly))
        Show_Error();
    else {
        QTextStream out(&file);
        out << txt;
        file.close();
        QMessageBox::about(this, "Збереження", "Файл " + ui->lineEdit->text() + "
успішно збережений");
    }
}
```

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

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();

    return a.exec();
}
```