

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ
МЕТОДІВ РОБОТИ З МАКРОСАМИ
RESEARCH AND SOFTWARE IMPLEMENTATION OF
METHODS FOR WORKING WITH MACROS

Виконав: студент(ка) 2 курсу, групи КНТ-223м
Спеціальності 122 Комп'ютерні науки
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Системи штучного інтелекту

ГУЛАК В.М.

(ПРИЗВИЩЕ та ініціали)

Керівник

ШИТІКОВА О.В.

(ПРИЗВИЩЕ та ініціали)

Рецензент

КОЦУР М.І.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут, факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 122 Комп'ютерні науки
(код і найменування)
Освітня програма (спеціалізація) Системи штучного інтелекту
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
“ ” 2024 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ГУЛАКА Владислава Максимовича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів роботи з макросами.

Research and Software Implementation of Methods for Working with Macros.

керівник проєкту (роботи) к.т.н., ШИТІКОВА Олена Вікторівна,

(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “ 07 ” жовтня 2024 року № 398

2. Строк подання студентом проєкту (роботи) 02 грудня 2024 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3. Основні рішення щодо реалізації програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ШИТІКОВА О.В., доцент		
Нормоконтролер	БЄЛОВА А.В., асистент		

7. Дата видачі завдання “07” жовтня 2024 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	2–3 тижні	Розділ 1
3	Розробка та удосконалення методів, моделей й алгоритмів вирішення задачі.	4–5 тижні	Розділ 2
4	Розробка архітектури програми.	6 тиждень	Розділ 3
5	Розробка програми.	7–8 тижні	Розділ 4
6	Тестування та експериментальне дослідження програмного забезпечення.	9 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	10-11 тижні	Додатки
8	Нормоконтроль та рецензування.	12 тиждень	
9	Захист роботи.	12 тиждень	

Студент(ка)

Владислав ГУЛАК
(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

Олена ШИТІКОВА
(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
92 с., 7 табл., 37 рис., 3 дод., 19 джерел.

МАКРОС, КОМП'ЮТЕРНА ПРОГРАМА, КЛАВІАТУРА,
ПЕРСОНАЛЬНИЙ КОМП'ЮТЕР, C#, .NET, VISUAL STUDIO.

Об'єкт дослідження — програмні методи роботи з макросами клавіатури.

Предмет дослідження — програмне забезпечення для запису та відтворення клавіатурних макросів.

Мета роботи — створення програмного застосунку для роботи з клавіатурними макросами.

Матеріали, методи та технічні засоби: структурне та об'єктно-орієнтоване програмування, мови програмування C# з використанням .NET та Windows Forms для створення графічного інтерфейсу. Для реалізації функцій інтегровані бібліотеки WindowsInput (емуляція натискань клавіш), MouseKeyHook (перехоплення подій клавіатури та миші) та Microsoft.VisualBasic (додаткові функції). Розробка проводилася в Microsoft Visual Studio., персональний комп'ютер з процесором Intel Core i3 під управлінням операційної системи Microsoft Windows 10/11.

Результати — програмне забезпечення для запису та виконання клавіатурних макросів із підтримкою користувацьких профілів.

Висновки — розроблено програмне забезпечення для роботи з макросами, яке дозволяє значно економити час та підвищує продуктивність роботи користувача, за рахунок зменшення кількості рутинних задач.

Галузь використання — автоматизація повторюваних дій у офісних додатках, програмуванні, геймінгу та інших сферах, де потрібна швидка та точна обробка команд.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 92 pages, 7 tables, 37 figures, 3 appendixes, 19 sources.

MACROS, PERSONAL COMPUTER, KEYBOARD, COMPUTER PROGRAM, C#, .NET, VISUAL STUDIO.

Object of study is a software methods for working with keyboard macros.

Subject of study are software for recording and replaying keyboard macros.

The purpose of the work is to create a software system for for working with keyboard macros.

Materials, methods, and technical tools: structured and object-oriented programming, C# programming language with the use of .NET and Windows Forms for creating the graphical user interface. To implement specific functions, external libraries were integrated, including WindowsInput (key press simulation), MouseKeyHook (keyboard and mouse event interception), and Microsoft.VisualBasic (additional functions). The development was carried out in Microsoft Visual Studio, using a personal computer with an Intel Core i3 processor running the Microsoft Windows 10/11 operating system.

Results. A solution for recording and executing keyboard macros with support for user profiles.

Conclusions. Software has been developed for working with macros, enabling significant time savings and enhancing user productivity by minimizing the number of routine tasks.

The field of use is a automation of repetitive tasks in office applications, programming, gaming, and other areas where quick and precise command processing is required.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	9
Вступ.....	10
1 Аналіз предметної області.....	12
1.1 Дослідження використання макросів клавіатури	12
1.2 Дослідження існуючих аналогів.....	14
1.2.1 AutoHotKey	14
1.2.2 JitBit Macro Recorder.....	15
1.2.3 TinyTask	16
1.2.4 Macro Recorder.....	17
1.2.5 Порівняння програмних рішень	18
1.3 Постанова завдання.....	19
1.4 Функціональні вимоги	21
1.5 Вимоги до інтерфейсу.....	21
1.6 Експлуатаційні вимоги	22
1.7 Висновки за розділом 1	22
2 Матеріали і методи.....	24
2.1 Вибір мови програмування	24
2.2 Вибір фреймворків та бібліотек для розробки.....	26
2.2.1 .NET Framework.....	26
2.2.2 Windows Forms	28
2.2.3 MouseKeyHook	28
2.2.4 WindowsInput.....	29
2.2.5 Microsoft.VisualBasic.....	30
2.3 Вибір середовища розробки.....	30
2.4 Висновки за розділом 2	33
3 Основні рішення щодо реалізації програми.....	34
3.1 Структура програмного забезпечення	34

3.1.1 MainForm.....	35
3.1.2 IKeyboardMouseEvents.....	35
3.1.3 InputSimulator.....	36
3.1.4 KeyPromptForm.....	36
3.1.5 Profile Manager.....	37
3.1.6 JsonSerializer	37
3.2 Функціонування програмного забезпечення.....	37
3.3 Алгоритм роботи.....	39
3.4 Реалізація методів роботи	42
3.4.1 Ініціалізація програми при завантаженні	42
3.4.2 Створення нового профілю	43
3.4.3 Підписка на глобальний хук	44
3.4.4 Запис натиснутих клавіш для макросу	44
3.4.5 Прив'язка макросу до клавіші	45
3.4.6 Визначення натискання і виконання макросу.....	46
3.4.7 Видалення прив'язки клавіші до макроса	47
3.5 Розробка користувацького інтерфейсу	50
3.6 Висновки за розділом 3	52
4 Експлуатація, тестування та експериментальне дослідження програми	53
4.1 Умови застосування та призначення програмного продукту	53
4.2 Комплект постачання програмного застосунку.....	54
4.3 Характеристика програми	54
4.4 Інструкція з експлуатації програми	55
4.4.1 Встановлення програми.....	55
4.4.2 Робота з застосунком	56
4.5 Тестування програми.....	62
4.6 Висновки за розділом	68
Висновки	70
Перелік джерел посилання	71
Додаток А Технічне завдання	73

Додаток Б Текст програми	77
Додаток В Слайди презентації.....	87

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

MMORPG	— Massively Multiplayer Online Role-Playing Game;
MOBA	— Multiplayer Online Battle Arena;
MS	— Microsoft;
VB	— Visual Basic;
VS	— Visual Studio;
ОС	— операційна система;
ПЗ	— програмний застосунок;
ПК	— персональний комп'ютер.

ВСТУП

На сьогоднішній день в світі налічуються мільйони користувачів персональних комп'ютерів, і майже кожен користувач використовує клавіатуру та мишу як основні маніпулятори для роботи з ним.

В сучасному світі у багатьох компаніях та підприємствах використовуються комп'ютери і спеціалізовані програмні засоби для виконання поставлених задач, багато з яких мають у своєму функціоналі використання макросів для автоматизації рутинних процесів в роботі, наприклад, виконання порядку натискань клавіш у текстових редакторах для роботи з документами.

Якщо дивитись з боку простих користувачів то макроси дуже поширені, завдяки тому що в багатьох програмах та навіть в деяких браузерах є можливість запису макросів для використання. Їх використання дуже поширене серед геймерів у комп'ютерних іграх. Виробники комп'ютерної периферії нерідко випускають спеціалізоване програмне забезпечення під свої продукти.

Використання макросів дає змогу автоматизувати, або прискорити виконання одноманітних дій при користуванні програмами, тому не дивно, що для багатьох вони стали невід'ємною частиною повсякденного життя.

Людям потрібні прості програмні рішення для створення макросів і використання їх у своїй роботі з персональним комп'ютером. Особливо на українському ринку на якому на даний момент не представлено подібних програм.

Об'єкт дослідження – програмні методи роботи з макросами клавіатури.

Предмет дослідження – програмне забезпечення для запису та відтворення клавіатурних макросів.

Тому метою проєкту є створення застосунку, що забезпечує роботу з макросами для клавіатури.

Програма буде виконувати наступні функції:

- запис макросів для послідовності натискань клавіш;
- прив'язка записаного макроса до потрібної клавіші;
- збереження даних про створені макроси з прив'язками на носії;
- робота з профілями їх створення, видалення і вибір активного профіля;
- збереження записаних макросів у профіль користувача, наприклад, для виконання різних макросів однією і тою ж клавішою.

Для досягнення мети роботи необхідно вирішити наступні завдання:

- дослідити вже існуючі аналоги;
- визначити необхідний інструментарій для розробки;
- провести проектування та програмну реалізацію прикладної програми;
- протестувати розроблений застосунок.

Було проведено дослідження методів роботи з макросами, їх переваги та недоліки, на основі чого і було створено програмний продукт для роботи з макросами клавіатури.

Практична значимість розробленого ПЗ – це економія часу та підвищення продуктивності роботи користувача, за рахунок зменшення кількості рутинних задач.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Дослідження використання макросів клавіатури

Макроси для клавіатур дуже широко використовуються для полегшення роботи та зменшення використання рутинних або складних комбінацій клавіш, наприклад, ми можемо розглянути будь який редактор тексту та навіть простий чат, в який використовуються такі комбінації клавіш як «Ctrl+C» для копіювання виділеного тексту, та «Ctrl+V» для його вставки. Завдяки макросам ми можемо значно пришвидшити використання навіть таких простих комбінацій.

Макроси також активно використовуються в комп'ютерних іграх. Найпоширеніші жанри, в яких використовуються макроси, включають файтинги, MMORPG, МОБА. Розглянемо кожен з цих жанрів. Файтинги, такі як Street Fighter і Tekken, як відомо, неймовірно складні. Для виконання складних комбо та потужних атак потрібно кілька натискань кнопок. Завдяки функції запису макросів геймери можуть спростити ці складні комбінації одним натисканням однієї кнопки. Крім того, оскільки макроси записані точно, вони будуть виконувати команди без людської помилки. MMORPG – ще один сумнозвісний жанр за кількістю кнопок, які потрібно натиснути. Як і в файтингах, макроси неймовірно корисні для виконання команд у MMORPG. А завдяки таким функціям, як затримка, користувачі можуть встановлювати конкретні комбо, розраховуючи час введення команд залежно від того, як довго триває кожне заклинання. Нарешті, макроси також можуть бути дуже корисними для МОБА. Звичайно, цей жанр не такий складний, як два інших, про які ми говорили. Однак вони все одно можуть стати в нагоді для швидкого виконання комбо або використання предметів у певному порядку [1].

Хоча використання макросів у змагальних іграх — це дуже суперечлива тема, яка постійно обговорюється з точки зору чесності гри,

тому в деяких іграх існують обмеження або навіть програмне блокування макросів [1].

Використання макросів для клавіатур виконується зазвичай у двох варіантах — це використання спеціального програмного забезпечення і вбудовані в саму клавіатуру спеціальні програмовані кнопки на які можна записувати макроси на апаратному рівні. У другому варіанті, з першого погляду це здається дуже зручним, але тут свою роль відіграє висока ціна на пристрої з таким функціоналом, підвищення складності будови такого девайсу, а з цим і зменшення надійності, збільшення розмірів самої клавіатури, що для багатьох користувачів може бути тим фактором, який скоріш за все відмовить його від покупки. І найголовніша проблема це обмеження кількості одночасно доступних для роботи макросів, адже для їх виклику будуть використовуватися лише спеціальні клавіші, яких на клавіатурі небагато [1].

Використання ж програмних рішень нівелює більшість таких проблем, адже користувач не обмежений невеликим набором клавіш для відтворення і в залежності від того яке програмне забезпечення він буде використовувати, йому не буде потрібна спеціальна периферія під яку розроблений цей застосунок, адже багато виробників створюють програмне забезпечення під свої девайси.

Головна з переваг апаратних макросів перед програмними — це те, що система сприймає ніби користувач сам натискає ці послідовності клавіш, бо клавіатура симулює натискання, не всі програмні рішення так вміють. Якщо для роботи з програмами це не має значення, крім займання робочої клавіші у випадку використання програми, а ось зі сторони комп'ютерних ігор можуть бути проблеми з програмними блокуваннями макросів для чесної гри, але на сьогоднішній день апаратні рішення теж не є панацеєю, все залежить від реалізацію функціоналу від конкретного виробника, а вони є пропрієтарними тому їх дослідження є складним адже потребує специфічних інженерних навичок та реверс інжинірингу.

Тому для роботи будуть розглядатися лише програмні методи роботи з макросами.

1.2 Дослідження існуючих аналогів

1.2.1 AutoHotkey

AutoHotkey — це вільний відкритий засіб створення макросів і автоматизації, що дозволяє користувачам автоматизувати повторювані завдання у Microsoft Windows. AutoHotkey дає можливість змінити користувацький інтерфейс будь-якої програми, завдяки власній скриптовій мові, розробленої для роботи з клавіатурними скороченнями і гарячими клавішами [2].

Інтерфейс програми представлено на рисунку 1.1.

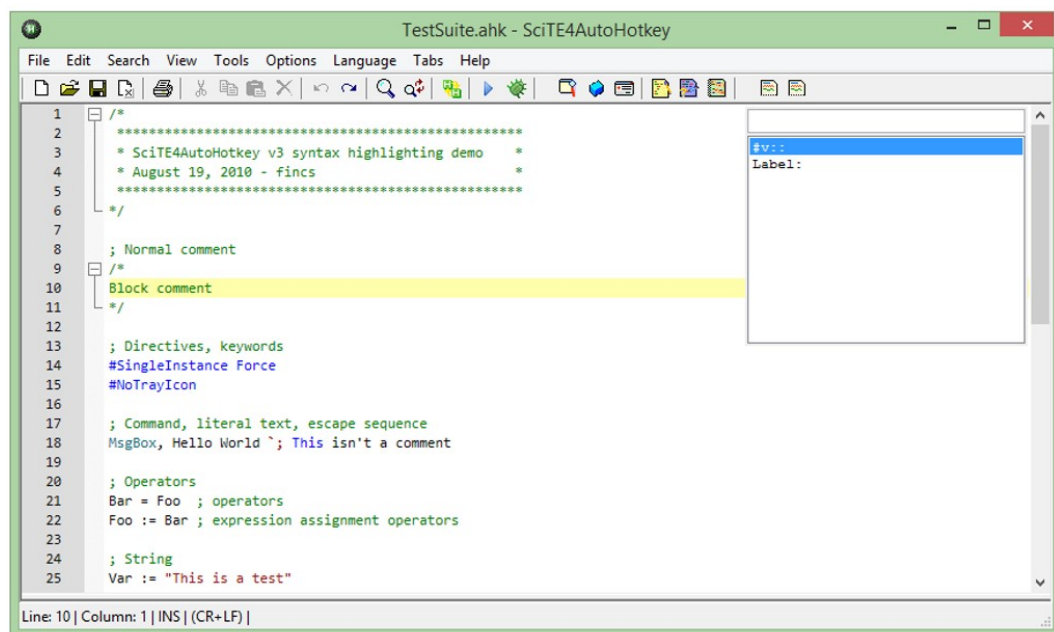


Рисунок 1.1 — Інтерфейс програми AutoHotkey

Переваги:

— це потужний інструмент, який дозволяє створювати складні макроси за допомогою використання скриптової мови. Вона дозволяє автоматизувати будь — які дії, від натискання клавіш до роботи з ПЗ;

— вона має відкритий код, і розповсюджується безкоштовно тому користувачу легко налаштувати програму під себе, або використовувати в своїх проєктах [2].

Недоліки:

— програма не орієнтована на новачків, або людей без досвіду у програмуванні, адже робота у програмі потребує знання мови написання скриптів [2].

1.2.2 JitBit Macro Recorder

Проста у використанні програма яка дозволяє записувати дії на клавіатурі та миші та відтворення у подальшому. Також підтримуються користувацькі гарячі клавіші. Інтерфейс застосунку представлено на рисунку 1.2.

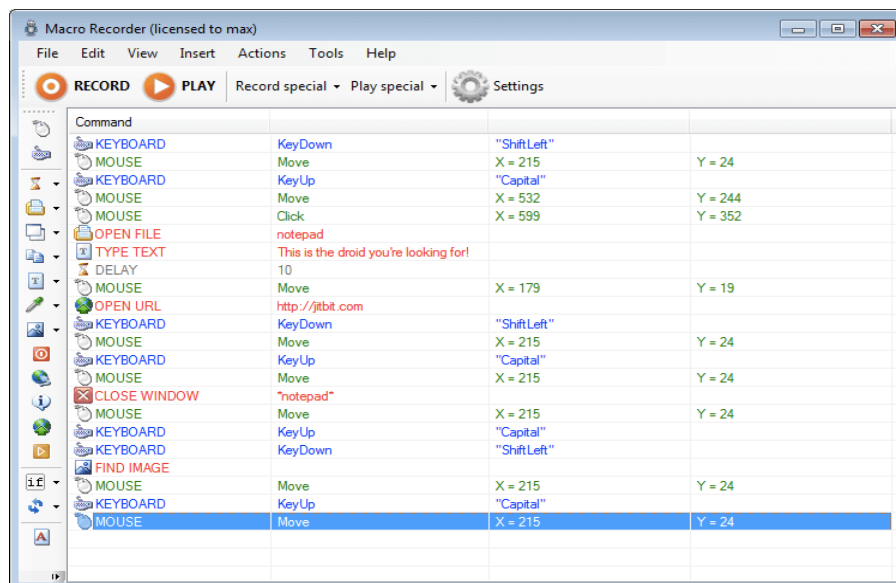


Рисунок 1.2 — Інтерфейс програми JitBit Macro Recorder

Переваги:

— програма має зручний інтерфейс і дружній до користувача функціонал, що дає змогу створювати макроси без додаткових знань у програмуванні;

— можна налаштовувати гарячі клавіші для активації записаних макросів, що дуже пришвидшує роботу користувачу, також вона підтримує планування макросів.

Недоліки:

— програма платна, тому для використання повного функціоналу необхідно купувати ліцензію;

— якщо порівнювати з попередньою програмою то вона має значно менший функціонал і не дозволяє виконувати складні макроси.

1.2.3 TinyTask

TinyTask — це безкоштовна мінімалістична програма для автоматизації ПК, яку можна використовувати для запису та повторення дій. Як випливає з назви, він неймовірно компактний [3].

Основна суть програми у компактності та простоті використанні, застосунок важить всього 36 КБ. Вона створена для користувачів яким не потрібний великий функціонал, і які не хочуть поглиблюватись у програмування. Проект має дуже гарну підтримку з боку розробника який розповсюджує програму безкоштовно, за бажанням можна залишити донат розробнику. Інтерфейс представлено на рисунку 1.3 [3].

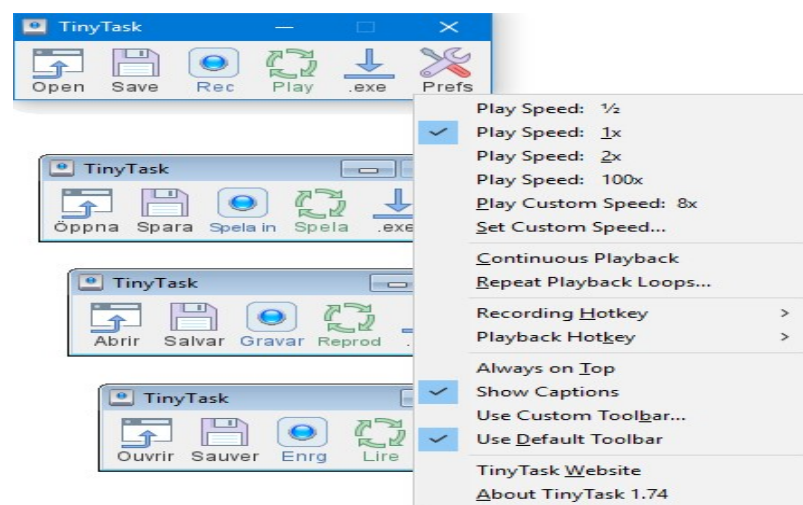


Рисунок 1.3 — Інтерфейс програми TinyTask

Переваги:

- це дуже легкий та простий у використанні інструмент, який займає лише 36 КБ і не вимагає інсталяції;
- програма повністю безкоштовна, що робить її доступною для всіх;
- відмінно підходить для автоматизації простих рутинних завдань [3].

Недоліки:

- TinyTask призначений для дуже базових макросів і не підтримує складніші сценарії або налаштування;
- відсутня можливість редагувати макроси або створювати більш складні варіанти автоматизації [3].

1.2.4 Macro Recorder

Macro Recorder — це чудовий інструмент для тих, хто хоче автоматизувати повсякденні завдання з мінімальними зусиллями. Хоча він не підтримує складні скрипти, його простота та функціональність роблять його ідеальним для базової автоматизації [4].

Інтерфейс програми представлено на рисунку 1.4.

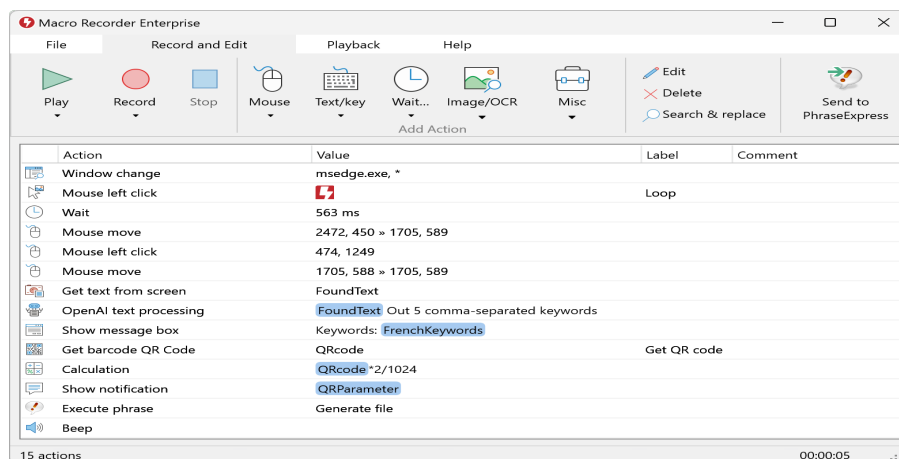


Рисунок 1.4 — Інтерфейс програми Macro Recorder

1.2.5 Порівняння програмних рішень

AutoHotkey дає можливість створювати макроси за допомогою скриптів, це дає користувачам дуже багато можливостей для автоматизації. Завдяки цьому є можливість створення дуже складних і вузьконаправлених скриптів, що дає змогу використовувати її не тільки для відтворення клавіш, а й для інтеграції з різними програмами. JitBit Macro Recorder спеціалізується на записі і відтворенні макросів клавіатури, але він більш доступний для користувачів, які не хочуть, або не вміють програмувати. TinyTask та Macro Recorder також виконують схожий функціонал, але вони обмежені лише простими завданнями з автоматизації, бо вони не підтримують складних сценаріїв або інтеграцій [3], [4].

Всі програми, крім TinyTask, підтримують гарячі клавіші для запуску макросів. Серед них AutoHotkey має найбільшу гнучкість та рівень кастомізації, адже дозволяє створювати не тільки макроси за допомогою гарячих клавіш, а й дозволяє налаштовувати їх для складних послідовностей дій [3], [4].

Більш детальне та наглядне порівняння цих програм наводиться у таблиці 1.1.

Таблиця 1.1 — Порівняння розглянутих програм

Характеристика	AutoHotkey	JitBit Macro Recorder	TinyTask	Macro Recorder
Тип макросів	Скрипти	Запис і відтворення	Запис і відтворення	Запис і відтворення
Гарячі клавіші	+ (скрипти)	+	—	+
Гнучкість	Висока	Середня	Низька	Низька
Підтримка планування	+ (скрипти)	+	—	+

Кінець таблиці 1.1

Характеристика	AutoHotkey	JitBit Macro Recorder	TinyTask	Macro Recorder
Доступність	Безкоштовна (відкритий код)	Платна (пробна версія)	Безкоштовна	Платна (доступна безкоштовна обмежена версія)
Простота використання	Складна (знання скриптів)	Легка (простий інтерфейс)	Дуже проста (тільки базові задачі)	Легка
Додаткові функції	Складні сценарії	Обмежені функції для простих задач	Мінімальні можливості	Просте планування та гарячі клавіші
Системні вимоги	Помірні (наявність Python)	Низькі	Дуже низькі	Низькі

У висновку ми бачимо, що в цілому всі ці програми так чи інакше підтримують запис та виконання макросів. Кожна програма має свої особливості чи обмеження, але на кожне рішення знайдеться свій користувач. З усього вищевипаного ми можемо сформулювати вимоги до свого програмного продукту.

1.3 Постановка завдання

Головним завданням роботи є створення програмного застосунку для роботи з клавіатурними макросами, орієнтовану на простоту до користувача але з достатнім функціоналом.

1.4 Функціональні вимоги

Функціональна вимога — це формулювання, яке визначає, що саме система повинна робити, аби задовольнити потреби або очікування користувачів. Вона описує зовнішню поведінку системи, тобто ті її функції, які безпосередньо доступні для використання користувачами або іншими системами. Такі вимоги є ключовими для розробки, оскільки вони визначають, які дії система повинна виконувати для досягнення своїх цілей [5].

Функціональні вимоги складаються з двох основних аспектів: функції та поведінки [5].

Функція описує, що саме виконує система. Наприклад, це може бути розрахунок податку на продаж, генерація звітів або обробка запитів користувача [5].

Поведінка вказує, як саме система виконує цю функцію. Наприклад, розрахунок податку може виконуватися шляхом множення вартості товару на встановлену ставку податку [5].

Ці вимоги спрямовані на те, щоб система забезпечила необхідний набір дій для виконання своїх завдань. При цьому функціональні вимоги відрізняються від нефункціональних, які фокусуються на внутрішніх характеристиках системи, таких як її продуктивність, надійність, безпека, масштабованість чи інші аспекти, що стосуються якості роботи [5].

Наприклад, функціональна вимога "Система повинна забезпечувати можливість авторизації користувача за логіном і паролем".

Функція – надання доступу до системи.

Поведінка – після введення логіна й пароля система перевіряє їх у базі даних та дозволяє доступ тільки при збігу [5].

Таким чином, функціональні вимоги є основою для визначення і проектування ключових функцій системи та їхнього впливу на кінцевого користувача [5].

До застосунку, що розробляється будуть висунуті наступні вимоги з функціональності:

- запис послідовностей натискань клавіш клавіатури;
- прив'язка записаної послідовності (макроса) до потрібної користувачу клавіші;
- відтворення створеного макроса по натисканню на обрану клавішу;
- макроси повинні виконуватись у будь якій програмі, а не тільки у власному вікні;
- користувач повинен мати можливість видалити прив'язку за потреби;
- користувач повинен мати можливість створювати профілі для збереження та групування макросів в залежності від його робочих задач;
- користувач повинен мати можливість зберегти записані макроси у свій профіль і мати можливість швидко перемикатись між своїми профілями;
- користувач повинен мати можливість видаляти профілі, разом з записаними до них макросами, без залишків в системі;
- створені та збережені макроси повинні зберігатись на носії і завантажуватись при запуску програми.

1.5 Вимоги до інтерфейсу

Висунуті наступні вимоги до інтерфейсу користувача:

- інтерфейс повинен бути простим, та інтуїтивно зрозумілим;
- дружнім до нового користувача;
- не допускається перевантаження інтерфейсу;
- забезпечити компактність інтерфейсу для зручної роботи в комбінації з цільовими програмами, і продуктивності використання простору екрану.

1.6 Експлуатаційні вимоги

Експлуатаційні вимоги частково сформовані з вимог до цільових операційних систем, а саме MS Windows 10 та MS Windows 11. Вимоги будуть наведені у мінімальному та оптимальному форматі.

Мінімальні вимоги до програмно – технічних засобів:

- ОС: Windows 10/11 (32-біт або 64-біт);
- процесор: Intel Core i3 (2.0 ГГц) або еквівалент;
- оперативна пам'ять: 4 ГБ;
- місце на диску: 500 МБ;
- .NET Framework: 4.7.2 або новіша;
- екран: роздільна здатність 1280x720 пікселів;
- клавіатура та миша.

Оптимальні вимоги:

- ОС: Windows 10/11 (64-біт);
- процесор: Intel Core i5 (3.0 ГГц) або еквівалент;
- оперативна пам'ять: 8 ГБ;
- місце на диску: 1 ГБ;
- .NET Framework: 4.7.2 або новіша;
- екран: роздільна здатність 1920x1080 пікселів;
- клавіатура та миша.

Вимоги до місця на диску зумовлені запасом місця на майбутнє, для запису великої кількості макросів та профілів.

1.7 Висновки за розділом 1

З метою розробки програмного продукту для роботи з макросами був проведений детальний аналіз предметної області, зокрема були розглянуті існуючі на сьогоднішній день програмні аналоги до нашої програми. Було

розглянуто їх переваги та недоліки, а також порівняли їх між собою як у тексті так і за допомогою порівняльної таблиці.

На основі розглянутого матеріалу були сформовані функціональні вимоги, вимоги до інтерфейсу та експлуатаційні вимоги (включаючи мінімальні та оптимальні) до розроблюваного продукту. З цих сформованих вимог ми можемо перейти до вибору методів реалізації які будуть описані у наступному пункті.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Вибір мови програмування

Для розробки програмного застосунку було зупинено вибір на мові програмування C#.

C# є мовою, яка створена для роботи на платформі .NET. Вона має дуже гарну інтеграцію з цільовою ОС. Це дозволяє розробляти додати, які набагато простіше можуть взаємодіяти з операційною системою, особливо коли йдеться мова про задачі з перехоплення подій клавіатури чи миші. Window Forms являє собою дуже потужний інструмент для побудови графічного інтерфейсу який у моєму випадку є невід'ємною частиною розроблюваного додатку [6], [7].

C# підтримує велику кількість як офіційних так і сторонніх бібліотек, таких як MouseKeyHook для роботи з подіями клавіатури та миші, а також WindowsInput, яка здатна емулювати натискання клавіш. Ці бібліотеки спрощують реалізацію функціональних вимог, щодо запису та відтворення макросів [7], [8].

C# забезпечує дуже високу продуктивність розробки, завдяки підтримці інструментів, таких як IntelliSense у Visual Studio. VS Дозволяє швидко створювати програмні застосунки, проводити відладку, працювати з базами даних і виконувати локалізацію. Для винонання вимог, щодо інтерфейсу, а саме для простоти та зручності, це дуже важливі фактори [9].

У таблиці 2.1 проведено порівняння мови C# з іншими схожими мовами.

Таблиця 2.1 — Порівняння мов програмування

Критерій	C#	Java	C++
Інтеграція з Windows	Глибока інтеграція через .NET	Потребує JNI для інтеграції	Пряма, але складна робота з WinAPI

Кінець таблиці 2.1

Критерій	C#	Java	C++
Розробка GUI	Легко через Windows Forms, WPF	JavaFX, але складніше у використанні	Потребує зовнішніх бібліотек (Qt, wxWidgets)
Продуктивність	Висока, оптимізація .NET	Вища, ніж Python, але більші ресурси JVM	Найвища завдяки низькорівневій оптимізації
Управління пам'яттю	Автоматичне через збирач сміття	Автоматичне через збирач сміття	Ручне управління пам'яттю
Кросплатформність	Обмежена до Windows, підтримка через .NET Core	Підтримка всіх ОС через JVM	Кросплатформність через бібліотеки
Складність розробки	Легка завдяки Visual Studio	Складніша через менш інтуїтивні інструменти	Висока через низькорівневу розробку

З усього перерахованого видно, що C# є найкращим вибором для розробки Windows-додатка завдяки глибокій інтеграції з ОС, легкості створення GUI та високій продуктивності при автоматичному управлінні пам'яттю.

2.2 Вибір фреймворків та бібліотек для розробки

Фреймворк — інфраструктура програмних рішень, що полегшує розробку складних систем. Спрощено дану інфраструктуру можна вважати своєрідною комплексною бібліотекою, але при цьому вона має ряд обмежень, що задають правила створення структури проєкту та написання коду. Одна з головних переваг, при використанні каркасних застосунків, полягає в тому, що такі програми мають стандартну структуру. Фреймворки стали популярними з появою елементів інтерфейсу, які мали тенденцію до реалізації стандартної структури для додатків. З їх використанням стало набагато простіше створювати засоби для автоматичного створення графічних інтерфейсів, оскільки структура внутрішньої реалізації коду програми стала відома заздалегідь. Для забезпечення каркаса, зазвичай, використовують підходи об'єктно-орієнтованого програмування [10].

Для виконання роботи були використані наступні фреймворки та бібліотеки:

- .NET Framework;
- Windows Forms;
- Gma.System.MouseKeyHook;
- WindowsInput;
- Microsoft.VisualBasic.

2.2.1 .NET Framework

.NET Framework — це незалежне середовище розробки програмного забезпечення. Воно надає набір бібліотек для роботи з даними, мережами, графічними інтерфейсами та багато іншого. Для .NET C# є основною мовою програмування, а Windows Forms є частиною системи створення десктоп застосунків [11].

Переваги використання:

- сумісність з Windows (особливо для старих версій Windows, таких як Windows 7 та 8);
- велика кількість бібліотек для роботи з базами даних, файловими системами, тощо;
- простота розробки при використанні IDE Visual Studio [11].

Це основний фреймворк для розробки проекту, він найкраще підходить для створення Windows-додатків завдяки високій продуктивності, відмінній підтримці UI і глибокій інтеграції з Windows API. Порівняння з іншими фреймворками наведено у таблиці 2.2.

Таблиця 2.2 — Порівняння фреймворків

Критерій	.NET	Qt	Electron
Підтримка Windows	Оптимізовано для Windows (Windows Forms, WPF)	Підтримка Windows є, але не така нативна	Підтримка Windows, але через браузерний движок
Продуктивність	Висока, нативна продуктивність	Висока, але може бути залежною від конкретної реалізації	Помірна, через використання вебтехнологій
Підтримка GUI	Відмінна підтримка UI через Windows Forms і WPF	Потужна підтримка UI, але з більш складною архітектурою	Використовує вебтехнології (HTML, CSS, JS)
Кросплатформеність	Кросплатформене через .NET Core	Кросплатформене, але з меншою оптимізацією	Кросплатформене завдяки Chromium

Кінець таблиці 2.2

Критерій	.NET	Qt	Electron
Інтеграція з апаратним забезпеченням	Легко інтегрується з Windows API	Може вимагати додаткових налаштувань	Обмежена інтеграція через JavaScript API

2.2.2 Windows Forms

Windows Forms — це фреймворк інтерфейсу користувача для створення настільних програм Windows. Він надає один із найпродуктивніших способів створення настільних програм на основі візуального дизайнера, наданого у Visual Studio. Такі функції, як перетягування, розміщення візуальних елементів керування, дозволяє легко створювати програми для настільних комп'ютерів. За допомогою Windows Forms можна розробляти графічно багаті програми, які легко розгортати, оновлювати та працювати в автономному режимі або при підключенні до Інтернету. Програми Windows Forms можуть отримувати доступ до локального устаткування та файлової системи комп'ютера, на якому запущено програму [12].

Переваги:

- легкий у використанні та вивченні для нових розробників;
- інтеграція з .NET, що дозволяє використовувати широкий спектр можливостей;
- відомість серед розробників, багато документації та прикладів [11].

2.2.3 MouseKeyHook

Ця бібліотека прикріплюється до глобальних процедур Windows, відстежує клацання та рухи клавіатури та миші, а також викликає поширені

події .NET за допомогою KeyEventArgs та MouseEventArgs, тому ми маємо можливість легко отримати потрібну інформацію, наприклад:

- координати миші;
- натиснуті кнопки миші;
- прокручування коліщатка миші;
- натискання та відпускання клавіш;
- спеціальні ключові стани;
- комбінації клавіш і послідовності [13].

Переваги використання MouseKeyHook:

- легко інтегрується з .NET-додатками для відстеження подій клавіатури та миші;
- підтримка різних типів подій (натискання клавіші, відпускання, рух миші);
- робота з усіма програмами, навіть коли ваша програма не знаходиться в фокусі [13].

2.2.4 WindowsInput

WindowsInput — це бібліотека для емуляції натискання клавіш та подій миші, що дає змогу автоматично взаємодіяти з іншими програмами чи операційною системою. Вона особливо корисна для автоматизації процесів і виконання макросів. WindowsInput надає прості класи .NET (C#) для захоплення та імітації введення з клавіатури та миші за допомогою SetWindowsHook та SendInput Win32. Весь інтероп зроблено за вас, і для всього існує проста модель програмування [14].

Переваги використання:

- підтримка емуляції подій клавіатури та миші;
- легке використання для автоматизації та тестування;
- може імітувати натискання клавіші без фізичного введення користувачем [14].

2.2.5 Microsoft.VisualBasic

Бібліотека `Microsoft.VisualBasic` надає інтерфейси для функцій, типових для Visual Basic, таких як `InputBox`, `MsgBox` та інші функції для взаємодії з користувачем через діалогові вікна. Хоча бібліотека асоціюється з Visual Basic, вона також доступна у C# через .NET Framework, що дозволяє використовувати ці функції в C#-додатках [15].

2.3 Вибір середовища розробки

Для розробки програмного застосунку було використано середовище розробки Microsoft Visual Studio 2022.

Visual Studio — це інтегроване середовище розробки (IDE), розроблене компанією Microsoft, яке призначене для створення широкого спектра програм, включаючи вебсайти, вебдодатки, вебслужби та мобільні додатки. Воно підтримує роботу з різними платформами для розробки програмного забезпечення Microsoft, такими як Windows API, Windows Forms, Windows Presentation Foundation (WPF), Microsoft Store та Microsoft Silverlight. Visual Studio дає можливість розробляти як нативний, так і керований код, що робить його універсальним інструментом для програмування на різних рівнях [16].

Visual Studio включає потужний редактор коду з підтримкою IntelliSense, що допомагає автоматично доповнювати код, а також інструменти для рефакторингу. Вбудований налагоджувач дозволяє працювати як на рівні вихідного коду, так і на рівні машинного коду. До інших корисних функцій належать профайлер коду, візуальний дизайнер для створення графічних інтерфейсів користувача, вебдизайнер, а також інструменти для проєктування класів і схем баз даних. Visual Studio також підтримує плагіни, які можуть розширювати функціональність на різних етапах розробки, додаючи підтримку систем контролю версій (наприклад, Git

або Subversion) та інші корисні інструменти, такі як спеціалізовані редактори чи інструменти для інтеграції з Azure DevOps [16].

Microsoft Visual C# — це реалізація мови програмування C# від Microsoft, орієнтована на використання в середовищі .NET Framework. Вона включає мовні служби, які дають можливість IDE Visual Studio підтримувати проєкти на C#. Хоча ці мовні служби є частиною Visual Studio, компілятор Visual C# доступний окремо, як частина .NET Framework. Компілятори для версій Visual C# 2008, 2010 та 2012 підтримують специфікації мови C# для версій 3.0, 4.0 та 5.0 відповідно. Visual C# також включає підтримку дизайнерів класів, форм та даних у Visual Studio, що полегшує процес розробки програм [16].

У таблиці 2.3 наведено порівняння середовищ розробки.

Таблиця 2.3 — Порівняння середовищ розробки

Критерій	Visual Studio	JetBrains Rider	NetBeans
Підтримка C#	Найкраща підтримка, інтеграція з .NET Framework	Підтримка через плагіни, але залежить від окремих налаштувань	Не підтримується
Інтеграція з .NET	Глибока інтеграція, повний набір інструментів для розробки на платформі .NET	Доступно, але не настільки оптимально, як у Visual Studio	Не підтримується
Інструменти розробки UI	Графічний дизайнер для Windows Forms і WPF	Відсутній вбудований графічний дизайнер	Потребує зовнішніх плагінів

Кінець таблиці 2.3

Критерій	Visual Studio	JetBrains Rider	NetBeans
Інструменти для тестування	Вбудований тестовий фреймворк MSTest, інтеграція з NUnit та іншими	Підтримка через додаткові модулі	Обмежені можливості
Продуктивність	Оптимізоване для роботи з великими проєктами, підтримка багатозадачності	Висока продуктивність, але потребує більше ресурсів	Аналогічно до Eclipse
Доступність	Безкоштовна Community-версія для індивідуальних розробників	Платне середовище	Безкоштовне

З усього вищезазначеного приходимо до висновку, що Visual Studio є найкращим вибором для цього проєкту. Воно забезпечує відмінну підтримку мови C#, має глибоку інтеграцію з платформою .NET Framework і пропонує широкий набір вбудованих інструментів для створення користувацького інтерфейсу, тестування та оптимізації коду. Окрім цього, Visual Studio демонструє високу продуктивність при роботі з великими проєктами, що робить його ідеальним рішенням для даного випадку.

2.4 Висновки за розділом 2

У другому розділі були вирішені питання обґрунтування та вибору інструментів для розробки програмного застосунку. Для створення

застосунку було проведено аналіз доступних технологій, після порівняння яких був сформований обґрунтований вибір: для програмної реалізації застосунку буде використовуватись мова програмування C#, з використанням фреймворків .NET та Windows Forms. Були обрані необхідні додаткові бібліотеки такі як WindowsInput, MouseKeyHook та Microsoft.VisualBasic в середовищі розробки Microsoft Visual Studio.

Обраний інструментарій розробки дозволить комфортно та якісно виконати реалізацію програмного застосунку.

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ ПРОГРАМИ

3.1 Структура програмного забезпечення

Була розроблена структура програмного застосунку для забезпечення реалізації методів роботи з макросами, що представлена на рисунку 3.1.

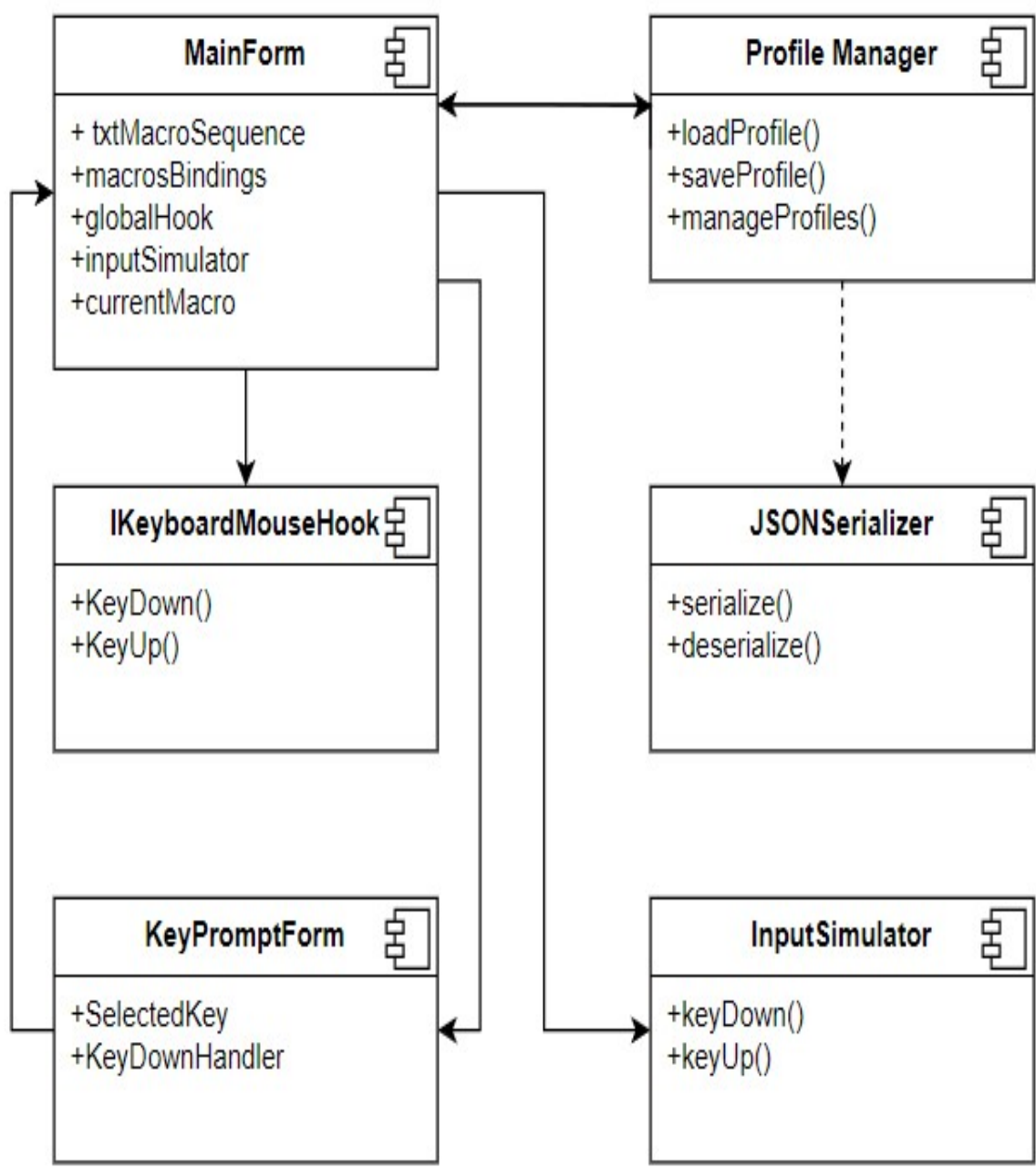


Рисунок 3.1 — Діаграма програмних компонентів реалізації методів роботи з макросами

3.1.1 MainForm

MainForm — це основний компонент програми, що забезпечує інтерфейс для взаємодії з макросами. Вся головна логіка, включаючи запис макросів, їх прив'язку до клавіші, збереження профілів та інші операції, реалізована тут. MainForm взаємодіє з іншими компонентами і управляє їх функціональністю через відповідні обробники подій.

Функції:

- запис макросів: користувач може записувати послідовність натискання клавіш за допомогою глобального хука, який перехоплює натискання;
- прив'язка клавіші до макросу: після запису макросу, користувач може прив'язати його до певної клавіші;
- управління профілями: користувач може створювати, завантажувати та зберігати профілі, що містять налаштування макросів;
- виконання макросів: програма може симулювати натискання клавіші через InputSimulator, виконуючи макроси при натисканні відповідної клавіші.

3.1.2 IKeyboardMouseEvents

IKeyboardMouseEvents — це інтерфейс для обробки подій, що стосуються натискання клавіш. Він використовується для перехоплення натискань клавіш через глобальний хук, що дозволяє програмі зчитувати всі клавіатурні події в системі.

Функції:

- перехоплення натискання клавіші: зчитує натискання клавіші і передає ці події для подальшої обробки в MainForm;

— обробка подій: інтерфейс дозволяє реагувати на події натискання клавіш і виконувати відповідні дії, наприклад, запускати макрос при натисканні конкретної клавіші.

3.1.3 InputSimulator

InputSimulator — це бібліотека для симуляції натискання клавіші на клавіатурі. Вона дозволяє програмно імітувати натискання і відпускання клавіш, що необхідно для виконання макросів.

Функції:

— симуляція натискання клавіші: InputSimulator надає методи для імітації натискання клавіші (наприклад, KeyDown), що дозволяє автоматично виконувати макроси, що складаються з натискань клавіш;

— симуляція відпускання клавіші: бібліотека також дозволяє симулювати відпускання клавіші, забезпечуючи повну симуляцію натискання.

3.1.4 KeyPromptForm

KeyPromptForm — це окрема форма, яка дає користувачу можливість вибрати клавішу для прив'язки до записаного макросу. Вона дозволяє вибрати певну клавішу з клавіатури для подальшого зв'язування з макросом.

Функції:

— вибір клавіші: при натисканні клавіші в формі, вона відображається на екрані, щоб користувач бачив, яку клавішу він обрав;

— передача вибору в MainForm: після вибору клавіші, форма передає інформацію в MainForm, де ця клавіша може бути прив'язана до макросу.

3.1.5 Profile Manager

Profile Manager — компонент відповідає за керування профілями користувача. Профіль зберігає налаштування, такі як прив'язка макросів до клавіш, і зберігається у вигляді JSON-файлів. Компонент дозволяє створювати нові профілі, завантажувати існуючі та зберігати зміни.

Функції:

- створення профілю: користувач може створювати новий профіль для збереження налаштувань макросів;
- завантаження профілю: профілі можна завантажити в програму, щоб користувач міг вибирати налаштування для макросів;
- збереження профілю: після внесення змін у макроси, користувач може зберегти їх у профілі.

3.1.6 JsonSerializer

JsonSerializer — це інструмент для серіалізації та десеріалізації об'єктів у формат JSON. У програмі він використовується для серіалізації макросів і прив'язок клавіш у JSON-формат, а також для десеріалізації профілів користувачів із JSON-файлів.

Функції:

- серіалізація: перетворює об'єкти (наприклад, словники макросів) у формат JSON для збереження на диску.
- десеріалізація: перетворює дані у форматі JSON назад в об'єкти, які використовуються програмою.

3.2 Функціонування програмного забезпечення

Схему функціонування програмного забезпечення можна представити у вигляді алгоритму, який представлено на рисунку 3.2.



Рисунок 3.2 — Схема функціонування програмного забезпечення

Після запуску програми вона завантажує профілі користувачів, якщо їх не було то користувач створює новий профіль.

Після створення або завантаження профілю користувач може записати послідовність натискань.

Наступним кроком користувач прив'язує необхідну йому клавішу.

Після цих кроків користувач може користуватись записаним макросом, за бажанням він може зберегти нові записані макроси у профіль.

3.3 Алгоритм роботи

Загальний алгоритм роботи програмного застосунку був реалізований наступним чином:

- після запуску програми, вона перевіряє чи існують збережені профілі, якщо ні то вона пропонує користувачу створити новий профіль, якщо профілі вже існують, тоді відкривається головне вікно MainForm, в якому користувач бачить елементи інтерфейсу для роботи з макросами та профілями;

- користувач натискає кнопку для початку запису нового макросу. Програма підключає глобальний хук IKeyboardMouseEvents, щоб перехоплювати натискання клавіш. Кожне натискання клавіші додається до списку макросу, а також відображається у текстовому полі на екрані. Користувач може записувати кілька натискань клавіші до завершення запису макросу;

- після запису макросу користувач натискає кнопку для прив'язки цього макросу до конкретної клавіші. Для цього програма відкриває форму KeyPromptForm, в якій користувач вибирає потрібну клавішу. Після вибору клавіші форма передає вибір назад в MainForm, де цей вибір зберігається, і клавіша прив'язується до записаного макросу. Тепер при натисканні на цю клавішу буде виконуватись записаний макрос;

— коли користувач натискає клавішу, яка була прив'язана до макросу, глобальний хук перехоплює натискання цієї клавіші та перевіряє, чи є для неї прив'язаний макрос. Якщо макрос знайдений, програма через `InputSimulator` імітує натискання клавіші відповідно до записаного макросу. Кожна клавіша в макросі буде натискатися та відпускатися згідно з вказівками;

— користувач може створювати нові профілі, завантажувати існуючі або зберігати поточні налаштування макросів в профіль. Для цього користувач натискає відповідну кнопку для створення профілю або завантажує профіль обравши його з випадаючого списку. При завантаженні програма зчитує дані з файлу профілю (формат JSON) і відновлює налаштування макросів. Для збереження та завантаження профілів використовуються функції `JsonSerializer`. Користувач може зберегти поточні налаштування макросів у профіль для майбутнього використання;

— після завершення роботи з програмою користувач може закрити програму. Програма звільняє ресурси, закриває всі підключення, наприклад, глобальний хук, і завершує роботу.

Алгоритм роботи представлено на рисунку 3.3.

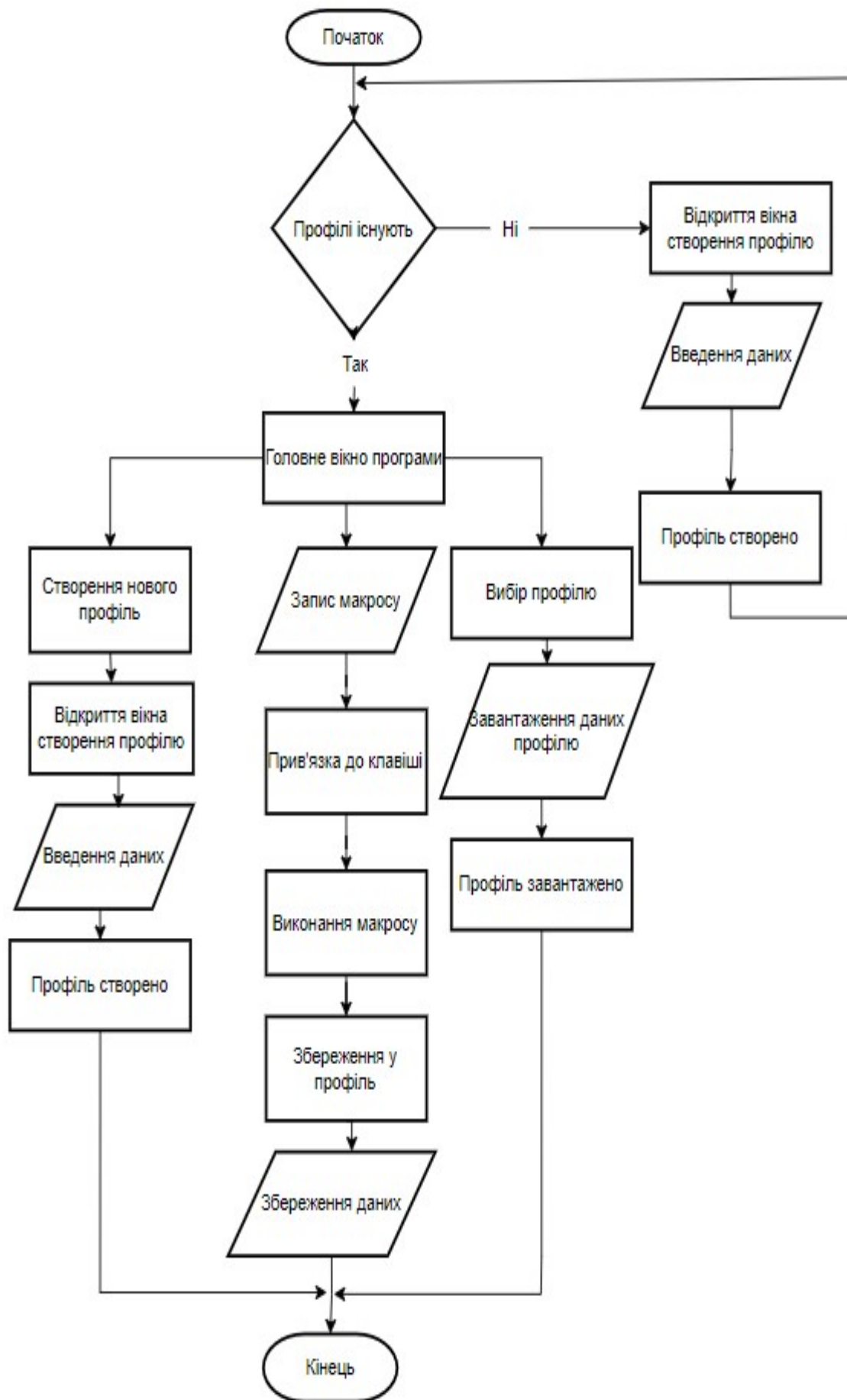


Рисунок 3.3 — Алгоритм роботи програми

3.4 Реалізація методів роботи

У ході роботи методи роботи з макросами були реалізовані за допомогою функцій. Далі будуть наведені основні функції програми.

3.4.1 Ініціалізація програми при завантаженні

Функція представлена на рисунку 3.4.

```
private void MainForm_Load(object sender, EventArgs e)
{
    InitializeMacroGrid();
    SubscribeGlobalHook();
    txtMacroSequence.Visible = false;
    string profilesDirectory = Path.Combine(Application.StartupPath, "Profiles");
    if (!Directory.Exists(profilesDirectory))
    {
        Directory.CreateDirectory(profilesDirectory);
    }

    string[] profileFiles = Directory.GetFiles(profilesDirectory, "*.json");
    if (profileFiles.Length == 0)
    {
        MessageBox.Show("Жодного профілю не знайдено. Зараз буде створено новий користувацький профіль.");
        CreateNewProfile();
    }
    else
    {
        LoadProfilesIntoComboBox();
    }
}
```

Рисунок 3.4 — Ініціалізація програми

Дана функція ініціалізує таблицю для відображення макросів. Підключає глобальний хук для перехоплення натискання клавіш. Перевіряє чи існують профілі і директорія збереження профілів, якщо папки не існує, вона автоматично створиться, якщо не знайдено жодного збереженого

профіля викликається функція створення профілю. Після вона завантажує всі існуючі профілі до випадаючого списку.

3.4.2 Створення нового профілю

Функція представлена на рисунку 3.5.

```
private void CreateNewProfile()
{
    string newProfileName = Microsoft.VisualBasic.Interaction.InputBox("Введіть назву нового профілю:", "Нове ім'я профілю", "Новий профіль");

    if (!string.IsNullOrEmpty(newProfileName))
    {
        string profilesDirectory = Path.Combine(Application.StartupPath, "Profiles");

        if (!Directory.Exists(profilesDirectory))
        {
            Directory.CreateDirectory(profilesDirectory);
        }

        string newProfilePath = Path.Combine(profilesDirectory, newProfileName + ".json");
        if (!File.Exists(newProfilePath))
        {
            File.WriteAllText(newProfilePath, JsonSerializer.Serialize(new Dictionary<string, List<VirtualKeyCode>>()));
            LoadProfiles();
            MessageBox.Show("Новий профіль створено!");
        }
        else
        {
            MessageBox.Show("Профіль з таким іменем вже існує!");
        }
    }
}
```

Рисунок 3.5 — Створення профілю

Ця функція дозволяє користувачу створити новий профіль, запитуючи ім'я профілю. Після введення імені вона перевіряє, чи існує профіль з таким ім'ям, і якщо ні, зберігає новий порожній профіль у форматі JSON.

3.4.3 Підписка на глобальний хук

Функція представлена на рисунку 3.6.

```
private void SubscribeGlobalHook()  
{  
    globalHook = Hook.GlobalEvents();  
    globalHook.KeyDown += GlobalHook_KeyDown;  
}
```

Рисунок 3.6 — Підписка на глобальний хук

Функція підключає глобальний хук для перехоплення натискання клавіші на рівні операційної системи. Після підключення, вона підписується на подію натискання клавіші, щоб відреагувати на натискання.

3.4.4 Запис натиснутих клавіш для макросу

Функція представлена на рисунку 3.7.

```
private void RecordMacroKey(object sender, KeyEventArgs e)  
{  
    if (!pressedKeys.Contains(e.KeyCode))  
    {  
        pressedKeys.Add(e.KeyCode);  
        VirtualKeyCode keyCode = (VirtualKeyCode)e.KeyValue;  
        currentMacro.Add(keyCode);  
        txtMacroSequence.Text = string.Join(" + ", currentMacro.Select(key => key.ToString()));  
    }  
}
```

Рисунок 3.7 — Запис натиснутих клавіш для макросу

Функція `RecordMacroKey` відповідає за запис натискання клавіші під час створення макросу. Вона перевіряє, чи була клавіша вже натиснута, і якщо ні, додає її до списку натиснутих клавіш. Потім перетворює код клавіші на об'єкт типу `VirtualKeyCode` і додає до поточної послідовності макросу. Оновлюється текстове поле, яке відображає поточний макрос у вигляді послідовності натиснутих клавіш, з'єднаних символом `+`.

3.4.5 Прив'язка макросу до клавіші

Функція представлена на рисунку 3.8.

```
private void btnBindKey_Click(object sender, EventArgs e)
{
    if (currentMacro.Count == 0)
    {
        MessageBox.Show("Спочатку запишіть макрос.");
        return;
    }

    string targetKey = KeyPromptForm();
    if (!string.IsNullOrEmpty(targetKey))
    {
        selectedKeyForBinding = targetKey;
        currentMacro = currentMacro
            .Where(key => key.ToString() != selectedKeyForBinding)
            .ToList();
        macroBindings[selectedKeyForBinding] = new List<VirtualKeyCode>(currentMacro);

        UpdateMacroGrid();
        currentMacro.Clear();
        txtMacroSequence.Clear();
        txtMacroSequence.Visible = false;
        globalHook.KeyDown -= RecordMacroKey;
        globalHook.KeyUp -= KeyUpHandler;
    }
    else
    {
        MessageBox.Show("Невірна клавіша для прив'язки.");
    }
}
```

Рисунок 3.8 — Прив'язка макросу до клавіші

Функція `btnBindKey_Click` відповідає за прив'язку записаного макросу до клавіші. Спочатку перевіряється, чи є хоча б один записаний макрос. Якщо немає, з'являється повідомлення з проханням спочатку записати макрос. Далі відкривається форма `KeyPromptForm` для вибору клавіші. Якщо клавіша вибрана, вона додається до словника `macroBindings` як ключ, а відповідний макрос — як значення. Оновлюється таблиця макросів через метод `UpdateMacroGrid`, очищується текстове поле, і відписується від подій для запису макросів. Це дозволяє легко прив'язати макрос до клавіші і редагувати їх.

3.4.6 Визначення натискання і виконання макросу

Реалізовано двома функціями `GlobalHook_KeyDown` та `ExecuteMacro`, які зображені на рисунку 3.9.

```
private void GlobalHook_KeyDown(object sender, KeyEventArgs e)
{
    string pressedKey = e.KeyCode.ToString();

    if (macroBindings.ContainsKey(pressedKey) && !isMacroRunning)
    {
        ExecuteMacro(macroBindings[pressedKey]);
        e.Handled = true;
    }
}

Ссылка: 1
private void ExecuteMacro(List<VirtualKeyCode> macro)
{
    isMacroRunning = true;

    foreach (var key in macro)
    {
        inputSimulator.Keyboard.KeyDown(key);
    }

    System.Threading.Thread.Sleep(50);
    foreach (var key in macro)
    {
        inputSimulator.Keyboard.KeyUp(key);
    }

    isMacroRunning = false;
}
```

Рисунок 3.9 — Визначення натискання і виконання макросу

GlobalHook_KeyDown викликається, коли користувач натискає клавішу. Якщо для натиснутої клавіші є прив'язаний макрос, і макрос зараз не виконується (перевірка через isMacroRunning), викликається функція ExecuteMacro, щоб виконати пов'язану з цією клавішею послідовність клавіш.

ExecuteMacro приймає список клавіш, що складають макрос, і виконує їх по черзі через inputSimulator. Спочатку натискаються всі клавіші, потім вони відпускаються.

3.4.7 Видалення прив'язки клавіші до макроса

Функція представлена на рисунку 3.10.

```
private void btnClearBinding_Click(object sender, EventArgs e)
{
    if (!string.IsNullOrEmpty(selectedKeyForBinding))
    {
        if (macroBindings.ContainsKey(selectedKeyForBinding))
        {
            macroBindings.Remove(selectedKeyForBinding);
            UpdateMacroGrid();
            MessageBox.Show($"Макрос для клавіші {selectedKeyForBinding} видалено.");
            selectedKeyForBinding = null;
        }
        else
        {
            MessageBox.Show("Макрос для цієї клавіші не знайдено.");
        }
    }
    else
    {
        MessageBox.Show("Будь ласка, виберіть макрос для видалення.");
    }
}
```

Рисунок 3.10 — Видалення прив'язки

Функція `btnClearBinding_Click` використовується для видалення прив'язки макросу до певної клавіші. Якщо вибрана клавіша вже має прив'язаний макрос, він видаляється зі словника `macroBindings`, що зберігає всі прив'язки. Після видалення оновлюється таблиця макросів за допомогою функції `UpdateMacroGrid`, а також відображається повідомлення, яке підтверджує успішне видалення. Якщо макрос для вибраної клавіші не знайдено або жодна клавіша не була обрана, користувач отримує відповідне сповіщення. Ця функція забезпечує гнучке управління прив'язками макросів і дозволяє легко їх видаляти за потреби.

Це основні функції які забезпечують реалізацію методів роботи з макросами. Нижче буде наведена таблиця 3.1 з переліком всіх функцій з короткими описами.

Таблиця 3.1 — Перелік функцій програми

Назва функції	Призначення
<code>MainForm_Load</code>	Виконує ініціалізацію інтерфейсу, підключення глобального хука, завантаження або створення профілів.
<code>CreateNewProfile</code>	Створює новий профіль і зберігає його у вигляді файлу JSON.
<code>LoadProfilesIntoComboBox</code>	Завантажує список доступних профілів у випадаючий список.
<code>LoadProfiles</code>	Завантажує профілі з диску і додає їх до випадаючого списку.
<code>InitializeMacroGrid</code>	Відображення макросів і прив'язаних клавіш.
<code>SubscribeGlobalHook</code>	Підключає глобальний хук для перехоплення натискання клавіш.

Продовження таблиці 3.1

Назва функції	Призначення
GlobalHook_KeyDown	Викликається при натисканні клавіші; перевіряє, чи є для неї макрос, і запускає його.
ExecuteMacro	Виконує макрос, симулюючи натискання та відпускання клавіш.
KeyUpHandler	Обробляє відпускання клавіш, очищуючи відповідні записи.
btnRecordMacro_Click	Починає запис нового макросу, дозволяючи користувачу вводити послідовність клавіш.
RecordMacroKey	Реєструє натискання клавіш під час запису макросу.
btnBindKey_Click	Прив'язує записаний макрос до вибраної клавіші.
KeyPromptForm	Викликає діалогове вікно для вибору клавіші, до якої буде прив'язаний макрос.
btnClearBinding_Click	Видаляє прив'язку макросу від вибраної клавіші.
SaveProfile	Зберігає поточні прив'язки макросів у вибраний профіль.
UpdateMacroGrid	Оновлює відображення таблиці макросів відповідно до актуальних прив'язок.
btnSaveProfile_Click	Зберігає зміни у вибраному профілі.
btnDeleteProfile_Click	Видаляє вибраний профіль з диску після підтвердження.

Кінець таблиці 3.1

Назва функції	Призначення
MainForm_FormClosing	Завершує підключення глобального хука при закритті програми.
macroGrid_CellClick	Обробляє вибір рядка таблиці, встановлюючи вибрану клавішу для подальшої дії.
profileComboBox_SelectedIndexChanged	Завантажує прив'язки макросів із вибраного профілю.
btnNewProfile_Click	Створює новий профіль через введення імені користувачем.

3.5 Розробка користувацького інтерфейсу

Інтерфейс користувача (UI) — це середовище, яке забезпечує взаємодію між користувачем і програмним забезпеченням або цифровим пристроєм. Його можна уявити як механізм, що дозволяє людині або автоматизованій системі, наприклад, роботу, управляти пристроєм, машиною чи програмою [16].

Існують різні типи користувацьких інтерфейсів, які відповідають різним потребам користувачів:

- інтерфейс командного рядка (CLI) – взаємодія через введення текстових команд, популярний серед розробників і IT-фахівців;
- меню-орієнтований інтерфейс – користувачі працюють через меню, як у банкоматах;
- формовий інтерфейс – взаємодія через заповнення форм, використовується в додатках для введення даних;
- сенсорний інтерфейс – управління за допомогою дотиків, поширений у смартфонах;

- голосовий інтерфейс (VUI) – голосові команди, застосовується у віртуальних помічниках;
- інтерфейс віртуальної реальності (VRI) – взаємодія у віртуальному середовищі через VR-гарнітури;
- жестовий інтерфейс – управління через фізичні жести, наприклад, у ігрових консолях;
- графічний інтерфейс (GUI) – візуальні елементи, як кнопки й піктограми, що спрощують взаємодію [17].

GUI є найпоширенішим і найзручнішим типом інтерфейсу для користувача, тому для розробки дружнього та простого способу взаємодії користувача з програмою він пріоритетний.

Для виконання поставлених задач з створення інтерфейсу, необхідно побудувати простий, дружній до користувача та ергономічний інтерфейс. Програма повинна раціонально використовувати простір екрану комп'ютера, тому не допускається громіздкість візуальної частини. Не допускається використання кольорів — подразників та малопоеднаних кольорів, адже це може погіршувати самопочуття користувача та негативно впливати на зір.

Елементи керування необхідно групувати для того, щоб користувач наглядно бачив за що відповідає та чи інша група елементів.

Реалізація цих вимог представлена на рисунку 3.11.

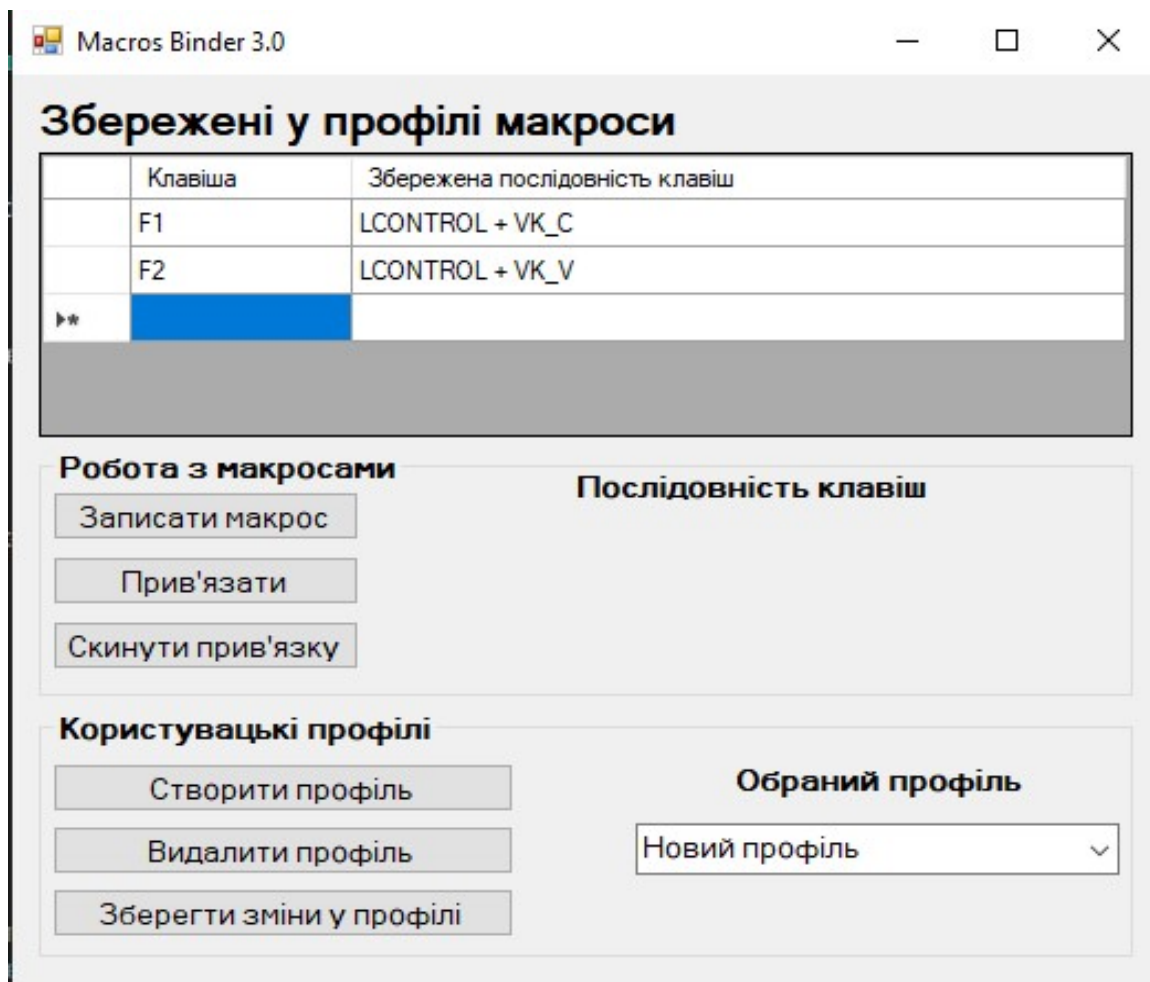


Рисунок 3.11 — Вигляд інтерфейсу програми

3.6 Висновки за розділом 3

У третьому розділі був сформований принцип функціонування програмного забезпечення на основі якого були сформовані методи для виконання поставлених задач. Було показано як програмний застосунок отримує інформацію про записаний макрос і як він прив'язується до потрібної клавіші. Були наведені приклади роботи з профілями.

Було розглянуто питання створення користувацького інтерфейсу. Для розробки був обраний дружній до користувача, інтуїтивно зрозумілий графічний інтерфейс. Саме ці рішення дозволять розробити якісний та доступний користувачу програмний продукт.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Умови застосування та призначення програмного продукту

Програму створено для запису та відтворення клавіатурних макросів, для персональних комп'ютерів на базі ОС MS Windows 10/11. Основною перевагою є простота використання та можливість роботи з профілями, що значно збільшує продуктивність роботи користувача з необхідними йому програмами.

Програма виконує наступні функції:

- запис макросів, що складаються з послідовності натискань клавіш;
- прив'язка створеного макросу до вибраної клавіші;
- збереження інформації про макроси та їх прив'язки на носії;
- управління профілями: створення, видалення та вибір активного профілю;
- збереження макросів у профілях користувача, що дозволяє використовувати одну клавішу для виконання різних макросів залежно від обраного профілю.

Програмний продукт потребує ПК з наступними системними вимогами.

Мінімальні вимоги:

- операційна система: Windows 10/11 (32-біт або 64-біт);
- процесор: Intel Core i3 з частотою 2.0 ГГц або аналогічний;
- оперативна пам'ять: 4 ГБ;
- місце на диску: щонайменше 500 МБ;
- платформа: .NET Framework версії 4.7.2 або новіша;
- дисплей: мінімальна роздільна здатність 1280x720 пікселів;
- пристрої введення: клавіатура та миша.

Оптимальні вимоги:

- операційна система: Windows 10/11 (64-біт);
- процесор: Intel Core i5 із частотою 3.0 ГГц або аналогічний;
- оперативна пам'ять: 8 ГБ;
- місце на диску: щонайменше 1 ГБ;
- платформа: .NET Framework версії 4.7.2 або новіша;
- дисплей: рекомендована роздільна здатність 1920x1080 пікселів;
- пристрої введення: клавіатура та миша.

4.2 Комплект постачання програмного застосунку

Програма надається безкоштовно і може поширюватися через хмарне сховище або на фізичних носіях, залежно від уподобань користувачів.

До складу пакета входить саморозпаковуваний архів, що містить виконуваний файл програми (.exe) разом із необхідними бібліотеками для роботи, а також повну технічну документацію до застосунку.

4.3 Характеристика програми

Продукт розроблено мовою програмування C# із використанням фреймворку .NET та технології Windows Forms для створення графічного інтерфейсу користувача. У процесі розробки було залучено кілька додаткових бібліотек, зокрема WindowsInput для емуляції натискань клавіш, MouseKeyHook для перехоплення та обробки подій клавіатури та миші, а також Microsoft.VisualBasic для підтримки додаткових функцій і зручностей. Проєкт створено в інтегрованому середовищі розробки Microsoft Visual Studio.

4.4 Інструкція з експлуатації програми

4.4.1 Встановлення програми

Програмний продукт постачається у формі саморозпаковуючого архіву. Для встановлення користувачу потрібно двічі клацнути на значок архіву, вибрати бажану директорію для розпаковки файлів і натиснути кнопку «Ок» для завершення процесу. По завершенню ви отримаєте папку з структурою файлів, що зображена на рисунку 4.1.

Ім'я	Тип	Розмір
app.publish	Папка файлів	
WindowsInput.dll	Розширення заст...	22 КБ
WindowsInput.xml	Microsoft Edge H...	121 КБ
System.Numerics.Vectors.dll	Розширення заст...	114 КБ
System.Numerics.Vectors.xml	Microsoft Edge H...	180 КБ
System.ValueTuple.dll	Розширення заст...	25 КБ
System.ValueTuple.xml	Microsoft Edge H...	1 КБ
System.Buffers.dll	Розширення заст...	21 КБ
System.Buffers.xml	Microsoft Edge H...	4 КБ
System.Threading.Tasks.Extensions.dll	Розширення заст...	26 КБ
System.Threading.Tasks.Extensions.xml	Microsoft Edge H...	10 КБ
System.Runtime.CompilerServices.Unsaf...	Microsoft Edge H...	21 КБ
System.Runtime.CompilerServices.Unsaf...	Розширення заст...	18 КБ
System.Memory.dll	Розширення заст...	139 КБ
System.Memory.xml	Microsoft Edge H...	14 КБ
Gma.System.MouseKeyHook.dll	Розширення заст...	56 КБ
System.Text.Encodings.Web.dll	Розширення заст...	78 КБ
System.Text.Encodings.Web.xml	Microsoft Edge H...	62 КБ
Microsoft.Bcl.AsyncInterfaces.dll	Розширення заст...	26 КБ
Microsoft.Bcl.AsyncInterfaces.xml	Microsoft Edge H...	31 КБ
System.IO.Pipelines.dll	Розширення заст...	83 КБ
System.IO.Pipelines.xml	Microsoft Edge H...	41 КБ
System.Text.Json.dll	Розширення заст...	710 КБ
System.Text.Json.xml	Microsoft Edge H...	558 КБ
MACROS3_0.exe.config	Исходный файл ...	1 КБ
MACROS3_0.application	Application Manif...	2 КБ
MACROS3_0.exe	Застосунок	20 КБ
MACROS3_0.exe.manifest	Файл MANIFEST	12 КБ
MACROS3_0.pdb	Program Debug D...	48 КБ

Рисунок 4.1 — Структура файлів встановленої програми

4.4.2 Робота з застосунком

Для запуску програми виконайте подвійний клік по файлу MACROS3_0.exe. Після цього у випадку першого запуску ви отримаєте повідомлення про створення нового профілю, яке представлене на рисунку 4.2. Після цього натисніть «Ок» і ви перейдете до вікна створення нового профілю, яке зображене на рисунку 4.3. Введіть назву профіля у текстовий рядок та натисніть «Ок». Після цього буде створено новий профіль, ви отримаєте повідомлення про успішне створення нового профілю, яке зображене на рисунку 4.4, далі ви перейдете до головного вікна програми, яке представлене на рисунку 4.5.

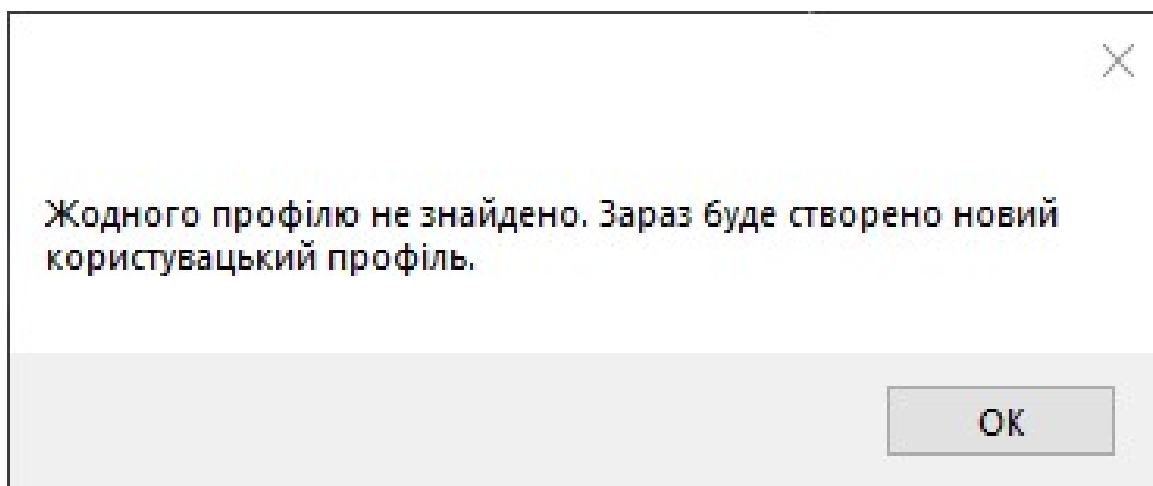


Рисунок 4.2 — Повідомлення про відсутність профілю

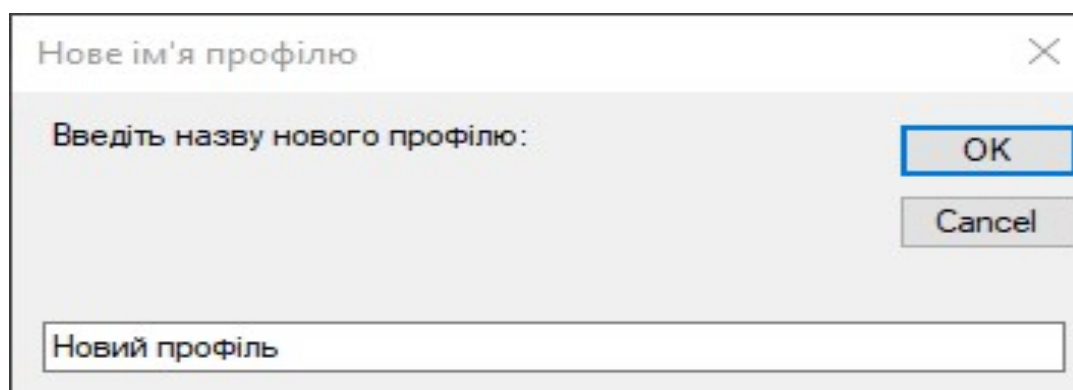


Рисунок 4.3 — Вікно створення нового профілю

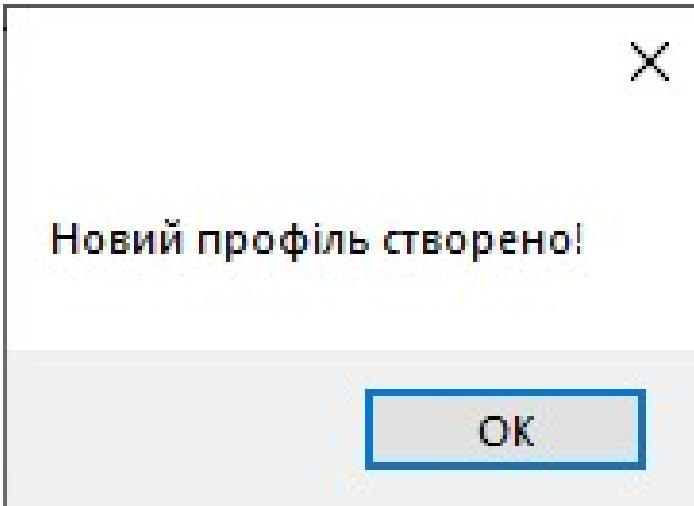


Рисунок 4.4 — Успішне створення профілю

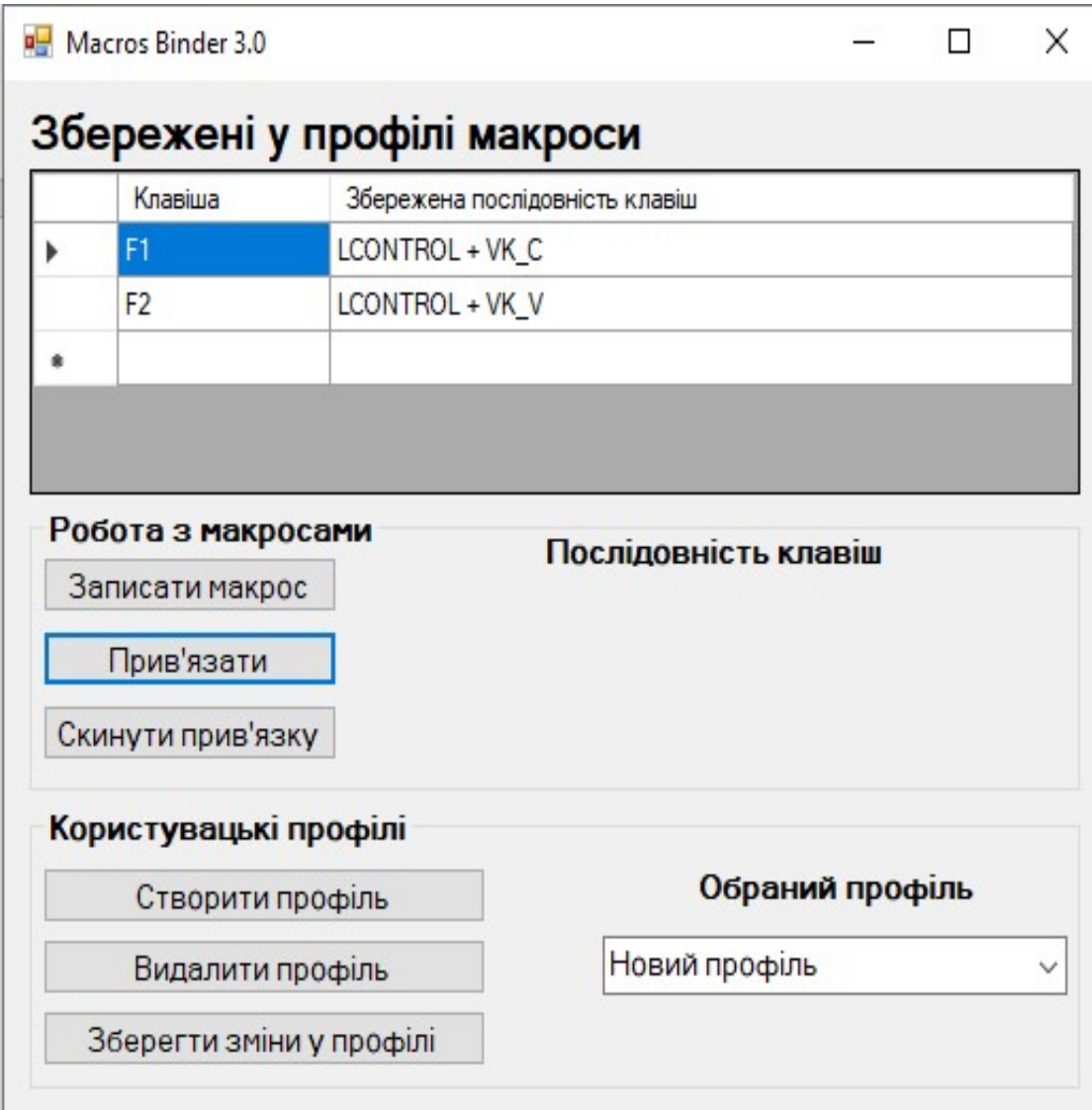


Рисунок 4.5 — Головне вікно програми

На головному вікні представлена таблиця яка показує макроси, що вже записані з відповідними до них клавішами, група елементів керування макросами та профілями.

Для запису нового макросу натисніть кнопку «Записати макрос», після цього з'явиться поле, яке відображає введену послідовність клавіш для наглядності користувачу. Починайте натискати на клавіші які вам необхідно об'єднати у послідовність, якщо ви помилились при введенні повторно натисніть кнопку «Записати макрос» та введіть послідовність спочатку, процес представлено на рисунку 4.6.

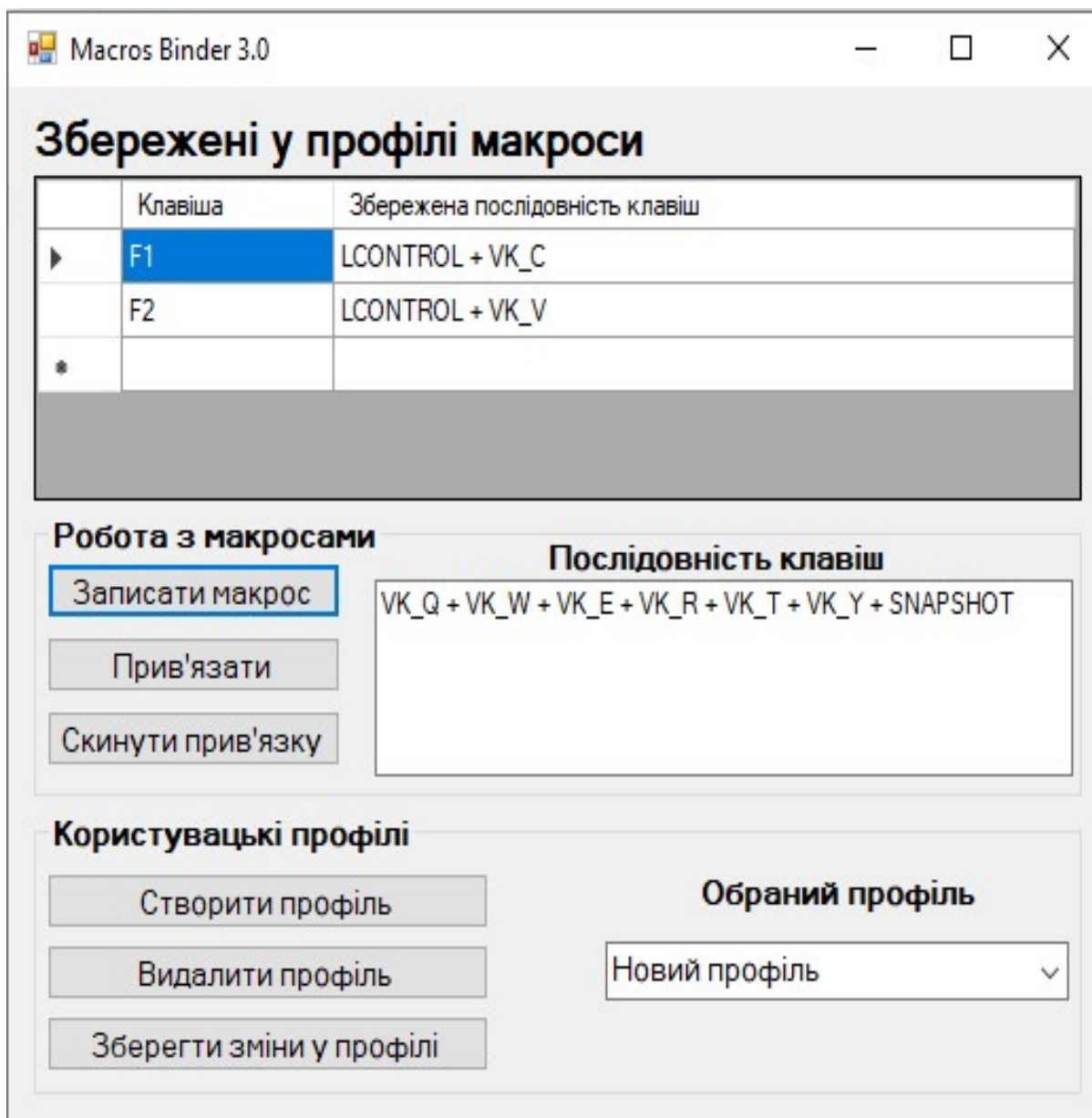


Рисунок 4.6 — Запис макроса

Наступним кроком буде прив'язка необхідної клавіші, для цього після набору послідовності натисніть мишею на кнопку «Прив'язати». Після цього відкриється вікно вибору кнопки для прив'язки, представлене на рисунку 4.7. Натисніть необхідну вам клавішу і натисніть «Ок», після цього вікно закриється, а записаний вами макрос відобразиться у таблиці.

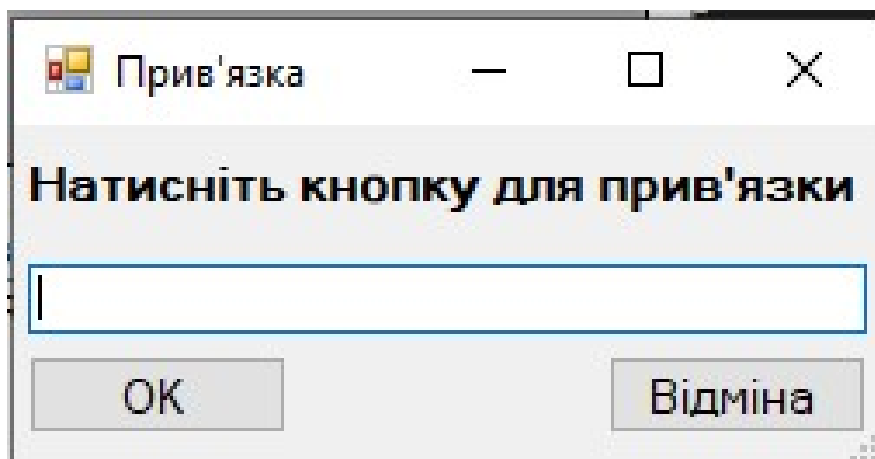


Рисунок 4.7 — Вікно прив'язки клавіші

Після прив'язки кнопка, яку обрав користувач починає відтворювати записані для неї дії.

Для скидання налаштованих макросів з клавіш необхідно мишею обрати потрібний макрос з верхнього списку і натиснути кнопку «Скинути прив'язку», після цього виконається скидання макроса з клавіші і ви отримаєте повідомлення про видалення, що зображене на рисунку 4.8.

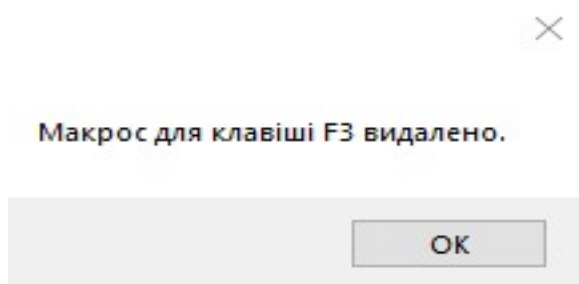


Рисунок 4.8 — Повідомлення про видалення макроса

Для того, щоб зберегти записані макроси у профіль натисніть кнопку «Зберегти зміни у профілі», після цього всі записані в цьому профілі макроси будуть збережені і ви отримаєте повідомлення, зображене на рисунку 4.9.

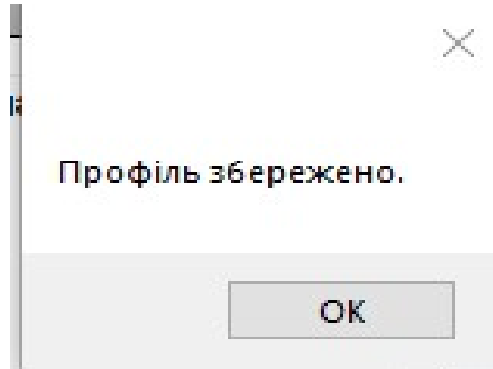


Рисунок 4.9 — Повідомлення про збереження змін у профілі

Для створення нового профілю натисніть на кнопку «Створити профіль», після цього відкриється вікно створення нового профілю, яке вже було зображене на рисунку 4.3, після введення назви профілю натисніть «Ок», після цього новий профіль буде створено і з'явиться повідомлення про успішне створення, що зображене на рисунку 4.4.

Вибір профілю виконується за допомогою випадаючого списку який підписаний як «Обраний профіль», відкрийте випадаючий список і натисніть на необхідний профіль, після цього програма автоматично завантажить дані обраного профілю. Важливо, перед перемиканням між профілями збережіть зміни у відкритому на даний момент профілі. Вибір профілю представлено на рисунку 4.10.

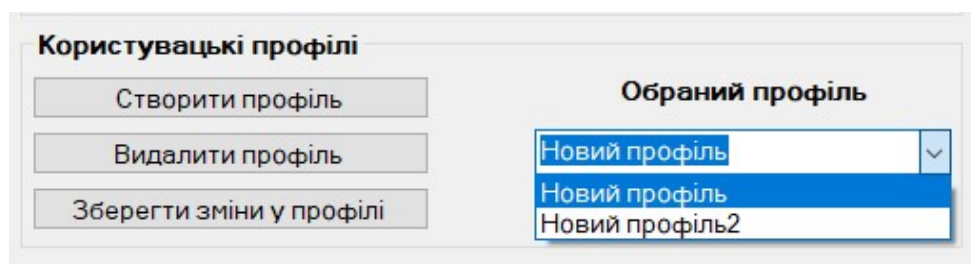


Рисунок 4.10 — Вибір профілю

Для видалення профілю необхідно переключитись на профіль, який потрібно видалити, та натиснути кнопку «Видалити профіль», після цього ви отримаєте попередження про видалення, зображено на рисунку 4.11, натисніть «Так», для видалення, або «Ні», для відміни видалення, після підтвердження видалення ви отримаєте повідомлення про успішне видалення, що зображене на рисунку 4.12, а програма завантажить останній створений профіль.

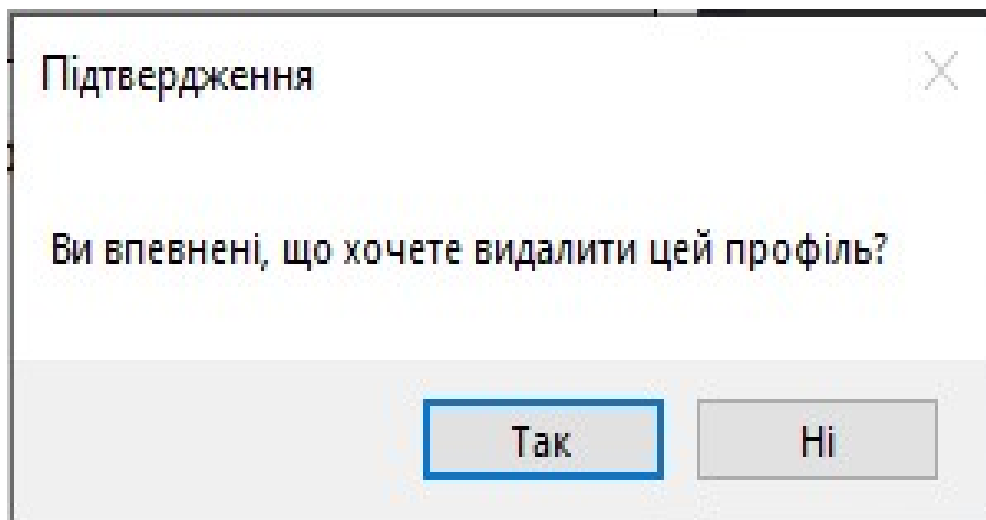


Рисунок 4.11 — Попередження про видалення

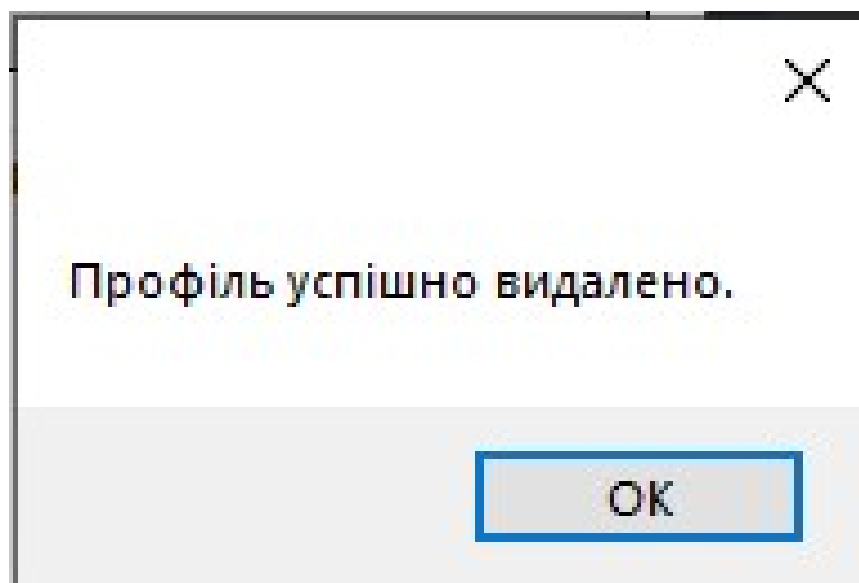


Рисунок 4.12 — Повідомлення про успішне видалення

Після завершення роботи програми всі збережені у профілях зміни будуть доступні при наступних запусках програм.

4.5 Тестування програми

У ході роботи були проведені функціональне тестування та тестування на надійність. Проходження тестування будуть надані у вигляді двох таблиць з тестовими кейсами які були виконані при тестуванні (табл. 4.1 – 4,2).

Таблиця 4.1 — Функціональне тестування

№	Опис тесту	Дія користувача	Очікуваний результат	Фактичний результат
1	Тест запису макросу	Користувач натискає на кнопку «Записати макрос», потім натискає комбінацію клавіш.	Програма повинна записати комбінацію натиснутих клавіш і відобразити її у текстовому полі.	Макрос записаний і відображений у текстовому полі.
2	Тест прив'язки макросу до клавіші	Користувач натискає на кнопку «Прив'язати до клавіші», вибирає клавішу через форму.	Програма повинна прив'язати записаний макрос до вибраної клавіші.	Макрос прив'язаний до вибраної клавіші.

Продовження таблиці 4.1

№	Опис тесту	Дія користувача	Очікуваний результат	Фактичний результат
3	Тест перевірки на відсутність записаних макросів	Користувач намагається прив'язати макрос без попереднього запису.	Програма повинна відобразити повідомлення, що макроси не записані.	Відображення повідомлення: "Будь ласка, запишіть макрос."
4	Тест збереження макросу в профіль	Користувач записує макрос і зберігає його в обраний профіль через меню профілів.	Програма повинна зберегти макрос у відповідному профілі.	Макрос успішно збережений у профілі.
5	Тест збереження профілю	Користувач зберігає поточний профіль після того, як прив'язав макроси до клавіш.	Програма повинна дозволити зберегти профіль з прив'язаними макросами.	Профіль успішно збережено.
6	Тест вибору активного профілю	Користувач обирає активний профіль із списку.	Програма повинна змінити активний профіль на вибраний.	Активний профіль змінено на вибраний.

Продовження таблиці 4.1

№	Опис тесту	Дія користувача	Очікуваний результат	Фактичний результат
7	Тест видалення профілю	Користувач видаляє профіль через відповідну кнопку в інтерфейсі.	Програма повинна видалити вибраний профіль разом з усіма прив'язаними макросами.	Профіль видалено.
8	Тест відображення таблиці макросів	Користувач записує кілька макросів, прив'язує їх до різних клавіш і перевіряє таблицю макросів.	Програма повинна коректно відображати всі макроси в таблиці після запису і прив'язки до клавіш.	У таблиці відображаються усі записані макроси з прив'язками.
9	Тест виклику помилки при некоректному введенні клавіші	Користувач намагається прив'язати вже зайняту клавішу.	Програма повинна відобразити повідомлення про помилку при спробі прив'язати некоректну або вже зайняту клавішу.	Відображення повідомлення про помилку: "Ця клавіша вже прив'язана."

Кінець таблиці 4.1

№	Опис тесту	Дія користувача	Очікуваний результат	Фактичний результат
10	Тест збереження змін у профілі після редагування	Користувач редагує профіль, додає нові макроси або змінює прив'язки і зберігає зміни.	Програма повинна зберегти зміни в профілі після редагування (наприклад, після додавання/видалення макросу).	Зміни успішно збережені в профілі.
11	Тест запуску макросу через клавішу	Користувач натискає клавішу, до якої прив'язано макрос.	Програма повинна виконати записаний макрос після натискання відповідної клавіші.	Макрос виконується автоматично.

Таблиця 4.2 — Тестування надійності

№	Опис тесту	Вхідні дані	Фактичний результат
1	Перевірка стабільності програми під час тривалого використання.	Програма працює протягом кількох годин або днів з виконанням різних операцій.	Програма працює без збоїв і зависань навіть після кількох годин роботи.
2	Перевірка роботи програми при великій кількості макросів або профілів.	Створення великої кількості макросів та профілів.	Програма залишається стабільною та не знижує продуктивність.

Кінець таблиці 4.2

№	Опис тесту	Вхідні дані	Фактичний результат
4	Перевірка збереження профілів та макросів після збоїв системи.	Завершення роботи програми без попередження або аварійне вимкнення ПК.	Програма відновила останні збережені макроси та профілі після перезавантаження.
5	Перевірка роботи програми після змін у системних налаштуваннях (наприклад, зміна мови або клавіатурних налаштувань).	Зміна мови або налаштувань клавіатури в системі.	Програма коректно працює навіть після змін у системних налаштуваннях.
6	Перевірка стабільності програми при високих навантаженнях на ресурси (низька кількість пам'яті).	Робота програми при низькому рівні оперативної пам'яті або високому навантаженні на процесор.	Програма залишається стабільною і продовжує працювати без значного зниження продуктивності.

Функціональне тестування — це узагальнюючий термін для різних видів тестів, які спрямовані на перевірку функціональності програмного забезпечення. Мета такого тестування — підтвердити, що програмний продукт виконує свої функції коректно та не має критичних помилок. Для цього тестувальник імітує сценарії використання системи реальними користувачами та порівнює отримані результати з очікуваними, що зафіксовані у специфікації вимог. Функціональне тестування може

проводитися як вручну, так і за допомогою автоматизованих інструментів [18].

Основна увага у функціональному тестуванні приділяється зовнішній поведінці програмного забезпечення. Завдяки цьому воно не вимагає глибокого розуміння внутрішнього коду програми та відноситься до методів тестування «чорного ящика». Зазвичай функціональне тестування виконується раніше, ніж нефункціональне [18].

Існує кілька видів функціонального тестування, кожен із яких має свої особливості, наприклад:

- димове тестування (Smoke Testing): перевірка базової працездатності основних функцій;
- санітарне тестування (Sanity Testing): тестування окремих модулів чи виправлених помилок;
- регресійне тестування (Regression Testing): перевірка, чи не вплинули нові зміни на раніше працюючий функціонал [18].

Функціональне тестування забезпечує перевірку програмного забезпечення на відповідність потребам кінцевих користувачів [18].

Тестування надійності — це вид тестування програмного забезпечення, що оцінює його здатність виконувати безвідмовну роботу у визначених умовах протягом заданого періоду часу. Основною метою такого тестування є виявлення потенційних збоїв або проблем у роботі продукту чи системи та оцінка ймовірності їх виникнення [19].

Складові надійності:

- зрілість (Maturity): рівень відповідності системи чи продукту вимогам надійності під час нормальної роботи;
- готовність (Availability): здатність системи залишатися працездатною та доступною для використання;
- відмовостійкість (Fault Tolerance): здатність системи працювати коректно навіть за наявності дефектів у програмному або апаратному забезпеченні;

— відновлюваність (Recoverability): здатність системи відновити необхідний стан і дані після збою або переривання роботи.

В ході тестування були показані відмінні показники працездатності та надійності, програмний застосунок вирішує поставлені задачі та видає необхідні результати.

В ході тестування надійності було підтверджено працездатність програми в робочих на нестандартних умовах.

Даний продукт відповідає всім стандартам функціональності та надійності.

4.6 Висновки за розділом 4

У розділі 4 було визначено основні умови експлуатації створеного програмного засобу та сформульовано його остаточний функціонал. Окрім того, було проведено всебічний аналіз характеристик програми, таких як системні вимоги, функціональні можливості та зручність використання. Особливу увагу було приділено комплекту постачання програмного продукту, що включає всі необхідні файли та інструкції для користувача.

Для забезпечення максимальної зручності користувачів було розроблено детальне керівництво, яке містить покрокові інструкції з налаштування та використання програми. Це керівництво охоплює всі важливі аспекти роботи з програмним продуктом.

Крім того, було проведено ретельне тестування програмного забезпечення за кількома критеріями: перевірки функціональних можливостей і надійності при високих навантаженнях. Це дозволило виявити та усунути можливі недоліки до завершення розробки.

Як результат, було отримано програмний продукт, який відповідає всім вимогам та готовий до використання в реальних умовах.

ВИСНОВКИ

В процесі виконання дипломної роботи було проведено актуальне дослідження створення програмного забезпечення для роботи з макросами. Основною метою було проаналізувати існуючі методи та реалізувати функціональні можливості для роботи з макросами.

У рамках роботи були виконані наступні кроки:

- здійснено детальний аналіз наявних рішень для роботи з макросами, а також оцінка їх сильних і слабких сторін;
- визначено функціональні вимоги до програмного продукту;
- спроектовано структуру програмного забезпечення;
- розроблено програмний продукт та проведено його тестування.

Програмний продукт розроблений на мові C# з використанням фреймворку .NET та технології Windows Forms для створення графічного інтерфейсу. Для досягнення необхідного функціоналу інтегровані сторонні бібліотеки, зокрема WindowsInput для емуляції натискання клавіш, MouseKeyHook для перехоплення подій клавіатури та миші, а також Microsoft.VisualBasic для розширених функцій. Розробка виконувалася в середовищі Microsoft Visual Studio, що забезпечило високу продуктивність і комфорт під час роботи.

В результаті виконання дипломної роботи був створений програмний продукт для запису та відтворення клавіатурних макросів, який також підтримує користувацькі профілі. Програма є зручною в користуванні і повністю відповідає вимогам технічного завдання.

Тестування показало високу надійність та стабільність продукту.

Усі завдання дипломної кваліфікаційної роботи магіста повністю виконано.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Mechanical Keyboard Macros – Everything You Need To Know [Electronic resource]. — Access mode : <https://keyboardsexpert.com/mechanical-keyboard-macros-everything-you-need-to-know/>.
2. AutoHotKey [Electronic resource]. — Access mode : <https://uk.wikipedia.org/wiki/AutoHotKey>.
3. TinyTask: Automation for Everyone [Electronic resource]. — Access mode : <https://tinytask.net>.
4. Macro Recorder [Electronic resource]. — Access mode : <https://www.macrorecorder.com>.
5. Що таке функціональні вимоги: приклади, визначення, повний посібник [Electronic resource]. — Access mode : <https://visuresolutions.com/uk/блог/функціональні-вимоги/>.
6. A tour of the C# language [Electronic resource]. — Access mode : <https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/overview>.
7. Overview of using controls [Electronic resource]. — Access mode : <https://learn.microsoft.com/en-us/dotnet/desktop/winforms/controls/overview?view=netdesktop-7.0>.
8. globalmousekeyhook [Electronic resource]. — Access mode : <https://github.com/gmamaladze/globalmousekeyhook>.
9. Visual Studio [Electronic resource]. — Access mode : https://en.wikipedia.org/wiki/Visual_Studio.
10. Програмний каркас [Electronic resource]. — Access mode : https://uk.wikipedia.org/wiki/Програмний_каркас.
11. Overview of .NET Framework [Electronic resource]. — Access mode : <https://learn.microsoft.com/en-us/dotnet/framework/get-started/overview>.
12. Desktop Guide (Windows Forms .NET) [Electronic resource]. — Access mode : <https://learn.microsoft.com/en-us/dotnet/desktop/winforms/overview/?view=netdesktop-9.0>.

13. MouseKeyHook [Electronic resource]. — Access mode : <https://www.nuget.org/packages/MouseKeyHook#readme-body-tab>.
14. WindowsInput [Electronic resource]. — Access mode : <https://www.nuget.org/packages/WindowsInput>.
15. Microsoft.VisualBasic Namespace [Electronic resource]. — Access mode : <https://learn.microsoft.com/en-us/dotnet/api/microsoft.visualbasic?view=netframework-4.8>.
16. Microsoft Visual Studio [Electronic resource]. — Access mode : https://uk.wikipedia.org/wiki/Microsoft_Visual_Studio.
17. Інтерфейс користувача [Electronic resource]. — Access mode : <https://www.ranktracker.com/uk/seo/glossary/user-interface/>.
18. Різниця між функціональним і нефункціональним тестуванням [Electronic resource]. — Access mode : <https://training.qatestlab.com/blog/technical-articles/difference-between-functional-and-non-functional-testing/>.
19. Тестування надійності [Electronic resource]. — Access mode : <https://simplentrec.com/reliability-testing/>.

ДОДАТОК А
Технічне завдання

Вступ

Програмний продукт для роботи з макросами знаходить застосування в різних сферах, де необхідно автоматизувати повторювані дії, підвищуючи ефективність і точність роботи. Програма може бути корисним в офісній діяльності для автоматизації процесів у текстових і табличних редакторах, у сфері програмування для прискорення виконання типових операцій, у геймінгу для створення послідовностей дій, а також у будь-якій галузі, де потрібно швидко виконувати складні або тривалі комбінації команд.

Тема для проєктування системи — дослідження та програмна реалізація методів роботи з макросами.

A.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу магістра на тему: «Дослідження та програмна реалізація методів роботи з макросами», затверджене наказом Національного університету «Запорізька політехніка» № 398 від 07 жовтня 2024 р.

A.2 Основні вимоги до програми, що розробляється

Головним завданням роботи є створення програмного застосунку для роботи з клавіатурними макросами.

A.2.1 Вимоги до функціональних характеристик

Функціональні вимоги:

- запис послідовностей натискань клавіш клавіатури;
- прив'язка записаної послідовності (макроса) до потрібної користувачу клавіші;

- відтворення створеного макроса по натисканню на обрану клавішу;
- макроси виконуються у будь якій програмі, а не тільки у власному вікні;
- можливість видалення прив'язки за потреби;
- створення профілів для збереження та групування макросів в залежності робочих задач;
- зберігання записаних макросів у свій профіль і мати можливість швидко перемикатись між своїми профілями;
- можливість видаляти профілі, разом з записаними до них макросами, без залишків в системі;
- створені та збережені макроси зберігаються на носії і завантажуватись при запуску програми.

A.2.2 Експлуатаційні вимоги

Вимоги наведені у мінімальному та оптимальному форматі.

Мінімальні вимоги до програмно – технічних засобів:

- ОС: Windows 10/11 (32-біт або 64-біт);
- процесор: Intel Core i3 (2.0 ГГц) або еквівалент;
- оперативна пам'ять: 4 ГБ;
- місце на диску: 500 МБ;
- .NET Framework: 4.7.2 або новіша;
- екран: роздільна здатність 1280x720 пікселів;
- клавіатура та миша.

Оптимальні вимоги:

- ОС: Windows 10/11 (64-біт);
- процесор: Intel Core i5 (3.0 ГГц) або еквівалент;
- оперативна пам'ять: 8 ГБ;
- місце на диску: 1 ГБ;

- .NET Framework: 4.7.2 або новіша;
- екран: роздільна здатність 1920x1080 пікселів;
- клавіатура та миша.

A.3 Вхідні дані до роботи

Вхідними даними для роботи програми є дії користувача, зокрема натискання клавіш і рухи миші, що записуються як макроси, вибір клавіші для прив'язки макросу, а також завантаження збережених профілів, які містять інформацію про макроси та їхні прив'язки. Крім того, користувач вводить дані через графічний інтерфейс, наприклад, назви профілів або команди для їх редагування чи видалення. Усі ці дані забезпечують виконання основних операцій програми, таких як створення, прив'язка та збереження макросів у профілях.

ДОДАТОК Б
Текст програми

Б.1 Файл MainForm.cs

```

using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text.Json;
using System.Windows.Forms;
using Gma.System.MouseKeyHook;
using MACROS3_0;
using WindowsInput;
using WindowsInput.Native;
using Microsoft.VisualBasic;

namespace MacroManager
{
    public partial class MainForm : Form
    {
        private IKeyboardMouseEvents globalHook;
        private Dictionary<string, List<VirtualKeyCode>> macroBindings = new
Dictionary<string, List<VirtualKeyCode>>();
        private List<VirtualKeyCode> currentMacro = new List<VirtualKeyCode>();
        private InputSimulator inputSimulator = new InputSimulator();
        private bool isMacroRunning = false;
        private string selectedKeyForBinding = null;
        private HashSet<Keys> pressedKeys = new HashSet<Keys>();

        public MainForm()
        {
            InitializeComponent();
            this.txtMacroSequence = new System.Windows.Forms.TextBox();
            this.txtMacroSequence.Multiline = true;
            this.txtMacroSequence.Width = 330;
            this.txtMacroSequence.Height = 80;
            this.txtMacroSequence.Location = new System.Drawing.Point(173, 200);
            this.Controls.Add(this.txtMacroSequence);
        }

        private void MainForm_Load(object sender, EventArgs e)
        {
            InitializeMacroGrid();
            SubscribeGlobalHook();
            txtMacroSequence.Visible = false;
            string profilesDirectory = Path.Combine(Application.StartupPath, "Profiles");
            if (!Directory.Exists(profilesDirectory))
            {
                Directory.CreateDirectory(profilesDirectory);
            }
        }
    }
}

```

```

string[] profileFiles = Directory.GetFiles(profilesDirectory, "*.json");
if (profileFiles.Length == 0)
{
    MessageBox.Show("Жодного профілю не знайдено. Зараз буде створено новий користувачський профіль.");
    CreateNewProfile();
}
else
{
    LoadProfilesIntoComboBox();
}
}
private void CreateNewProfile()
{
    string newProfileName = Microsoft.VisualBasic.Interaction.InputBox("Введіть назву нового профілю:", "Нове ім'я профілю", "Новий профіль");

    if (!string.IsNullOrEmpty(newProfileName))
    {
        string profilesDirectory = Path.Combine(Application.StartupPath, "Profiles");

        if (!Directory.Exists(profilesDirectory))
        {
            Directory.CreateDirectory(profilesDirectory);
        }

        string newProfilePath = Path.Combine(profilesDirectory, newProfileName + ".json");
        if (!File.Exists(newProfilePath))
        {
            File.WriteAllText(newProfilePath, JsonSerializer.Serialize(new Dictionary<string, List<VirtualKeyCode>>()));
            LoadProfiles();
            MessageBox.Show("Новий профіль створено!");
        }
        else
        {
            MessageBox.Show("Профіль з таким іменем вже існує!");
        }
    }
}
private void LoadProfilesIntoComboBox()
{
    string profilesDirectory = Path.Combine(Application.StartupPath, "Profiles");

    if (Directory.Exists(profilesDirectory))
    {
        string[] profileFiles = Directory.GetFiles(profilesDirectory, "*.json");

        profileComboBox.Items.Clear();

        foreach (var file in profileFiles)
        {

```

```

        string profileName = Path.GetFileNameWithoutExtension(file);
        profileComboBox.Items.Add(profileName);
    }

    if (profileComboBox.Items.Count > 0)
    {
        profileComboBox.SelectedIndex = 0;
    }
}
else
{
    MessageBox.Show("Папка профілів не знайдена.");
}
}

private void LoadProfiles()
{
    var profileDirectory = "Profiles";
    if (!Directory.Exists(profileDirectory))
    {
        Directory.CreateDirectory(profileDirectory);
    }

    var profiles = Directory.GetFiles(profileDirectory, "*.json");
    profileComboBox.Items.Clear();
    foreach (var profile in profiles)
    {
        profileComboBox.Items.Add(Path.GetFileNameWithoutExtension(profile));
    }

    if (profileComboBox.Items.Count > 0)
    {
        profileComboBox.SelectedIndex = 0;
    }
}

private void InitializeMacroGrid()
{
    macroGrid.Columns.Add("Key", "Клавіша");
    macroGrid.Columns.Add("Macro", "Збережена послідовність клавіш");
    macroGrid.Columns["Key"].Width = 100;
    macroGrid.Columns["Macro"].Width = 350;
}

private void SubscribeGlobalHook()
{
    globalHook = Hook.GlobalEvents();
    globalHook.KeyDown += GlobalHook_KeyDown;
}

private void GlobalHook_KeyDown(object sender, KeyEventArgs e)
{
    string pressedKey = e.KeyCode.ToString();

```

```

        if (macroBindings.ContainsKey(pressedKey) && !isMacroRunning)
        {
            ExecuteMacro(macroBindings[pressedKey]);
            e.Handled = true;
        }
    }
    private void ExecuteMacro(List<VirtualKeyCode> macro)
    {
        isMacroRunning = true;

        foreach (var key in macro)
        {
            inputSimulator.Keyboard.KeyDown(key);
        }

        System.Threading.Thread.Sleep(50);
        foreach (var key in macro)
        {
            inputSimulator.Keyboard.KeyUp(key);
        }

        isMacroRunning = false;
    }

    private void KeyUpHandler(object sender, KeyEventArgs e)
    {
        if (pressedKeys.Contains(e.KeyCode))
        {
            pressedKeys.Remove(e.KeyCode);
        }
    }

    private void btnRecordMacro_Click(object sender, EventArgs e)
    {
        currentMacro.Clear();
        pressedKeys.Clear();
        txtMacroSequence.Visible = true;
        txtMacroSequence.BringToFront();
        txtMacroSequence.Clear();
        globalHook.KeyDown += RecordMacroKey;
        globalHook.KeyUp += KeyUpHandler;
    }

    private void RecordMacroKey(object sender, KeyEventArgs e)
    {
        if (!pressedKeys.Contains(e.KeyCode))
        {
            pressedKeys.Add(e.KeyCode);
            VirtualKeyCode keyCode = (VirtualKeyCode)e.KeyValue;
            currentMacro.Add(keyCode);
            txtMacroSequence.Text = string.Join(" + ", currentMacro.Select(key =>
key.ToString()));

```

```

    }
}

private void btnBindKey_Click(object sender, EventArgs e)
{
    if (currentMacro.Count == 0)
    {
        MessageBox.Show("Спочатку запишіть макрос.");
        return;
    }

    string targetKey = KeyPromptForm();
    if (!string.IsNullOrEmpty(targetKey))
    {
        selectedKeyForBinding = targetKey;
        currentMacro = currentMacro
            .Where(key => key.ToString() != selectedKeyForBinding)
            .ToList();
        macroBindings[selectedKeyForBinding] = new List<VirtualKeyCode>(currentMacro);

        UpdateMacroGrid();
        currentMacro.Clear();
        txtMacroSequence.Clear();
        txtMacroSequence.Visible = false;
        globalHook.KeyDown -= RecordMacroKey;
        globalHook.KeyUp -= KeyUpHandler;
    }
    else
    {
        MessageBox.Show("Невірна клавіша для прив'язки.");
    }
}

private string KeyPromptForm()
{
    using (var keyPrompt = new KeyPromptForm())
    {
        if (keyPrompt.ShowDialog() == DialogResult.OK)
        {
            return keyPrompt.SelectedKey;
        }
    }
    return null;
}

private void btnClearBinding_Click(object sender, EventArgs e)
{
    if (!string.IsNullOrEmpty(selectedKeyForBinding))
    {
        if (macroBindings.Contains(selectedKeyForBinding))
        {

```

```

        macroBindings.Remove(selectedKeyForBinding);
        UpdateMacroGrid();
        MessageBox.Show($"Макрос для клавіші {selectedKeyForBinding} видалено.");
        selectedKeyForBinding = null;
    }
    else
    {
        MessageBox.Show("Макрос для цієї клавіші не знайдено.");
    }
}
else
{
    MessageBox.Show("Будь ласка, виберіть макрос для видалення.");
}
}
private void SaveProfile()
{
    string selectedProfile = profileComboBox.SelectedItem.ToString();
    string profilePath = Path.Combine("Profiles", selectedProfile + ".json");
    if (string.IsNullOrEmpty(selectedProfile))
    {
        MessageBox.Show("Будь ласка, виберіть профіль для збереження.");
        return;
    }
    File.WriteAllText(profilePath, JsonSerializer.Serialize(macroBindings));
    MessageBox.Show("Профіль збережено.");
}

private void UpdateMacroGrid()
{
    macroGrid.Rows.Clear();

    foreach (var binding in macroBindings)
    {
        macroGrid.Rows.Add(binding.Key, string.Join(" + ", binding.Value));
    }

    if (macroBindings.Count == 0)
    {
        MessageBox.Show("Ви не маєте жодного збереженого макросу.");
    }
}

private void btnSaveProfile_Click(object sender, EventArgs e)
{
    string selectedProfile = profileComboBox.SelectedItem.ToString();
    string profilePath = Path.Combine("Profiles", selectedProfile + ".json");

    if (string.IsNullOrEmpty(selectedProfile))
    {
        MessageBox.Show("Будь ласка, виберіть профіль для збереження.");
    }
}

```

```

        return;
    }

    File.WriteAllText(profilePath, JsonSerializer.Serialize(macroBindings));
    MessageBox.Show("Профіль збережено.");
}

private void btnDeleteProfile_Click(object sender, EventArgs e)
{
    if (profileComboBox.SelectedIndex == -1)
    {
        MessageBox.Show("Будь ласка, виберіть профіль для видалення.");
        return;
    }

    string selectedProfileName = profileComboBox.SelectedItem.ToString();
    string profilePath = Path.Combine("Profiles", selectedProfileName + ".json");

    if (File.Exists(profilePath))
    {
        var confirmResult = MessageBox.Show("Ви впевнені, що хочете видалити цей профіль?", "Підтвердження", MessageBoxButtons.YesNo);
        if (confirmResult == DialogResult.Yes)
        {
            try
            {
                File.Delete(profilePath);
                MessageBox.Show("Профіль успішно видалено.");
                LoadProfiles();
            }
            catch (Exception ex)
            {
                MessageBox.Show("Сталася помилка при видаленні профілю: " + ex.Message);
            }
        }
        else
        {
            MessageBox.Show("Профіль не знайдено.");
        }
    }
}

private void MainForm_FormClosing(object sender, FormClosingEventArgs e)
{
    globalHook.Dispose();
}

private void macroGrid_CellClick(object sender, DataGridViewCellEventArgs e)
{
    if (e.RowIndex >= 0 && e.ColumnIndex >= 0)
    {

```

```

        string selectedKey = macroGrid.Rows[e.RowIndex].Cells[0].Value?.ToString();
        if (!string.IsNullOrEmpty(selectedKey))
        {
            selectedKeyForBinding = selectedKey;
        }
    }
}

private void profileComboBox_SelectedIndexChanged(object sender, EventArgs e)
{
    string profileName = profileComboBox.SelectedItem.ToString();
    string profilePath = Path.Combine("Profiles", profileName + ".json");
    if (File.Exists(profilePath))
    {
        macroBindings = JsonSerializer.Deserialize<Dictionary<string,
List<VirtualKeyCode>>>>(File.ReadAllText(profilePath));
        UpdateMacroGrid();
    }
    else
    {
        MessageBox.Show("Профіль не знайдено!");
    }
}

private void btnNewProfile_Click(object sender, EventArgs e)
{
    string newProfileName = Microsoft.VisualBasic.Interaction.InputBox("Введіть назву нового профілю:", "Нове ім'я профілю", "Новий профіль");

    if (!string.IsNullOrEmpty(newProfileName))
    {
        string profilesDirectory = Path.Combine(Application.StartupPath, "Profiles");

        if (!Directory.Exists(profilesDirectory))
        {
            Directory.CreateDirectory(profilesDirectory);
        }

        string newProfilePath = Path.Combine(profilesDirectory, newProfileName + ".json");
        if (!File.Exists(newProfilePath))
        {
            File.WriteAllText(newProfilePath, JsonSerializer.Serialize(new Dictionary<string,
List<VirtualKeyCode>>>()));
            LoadProfiles();
            MessageBox.Show("Новий профіль створено!");
        }
        else
        {
            MessageBox.Show("Профіль з таким іменем вже існує!");
        }
    }
}

```

```

    }
}

```

Б.2 Файл KeyPromptForm.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace MACROS3_0
{
    public partial class KeyPromptForm : Form
    {
        public string SelectedKey { get; private set; }

        public KeyPromptForm()
        {
            InitializeComponent();
            SelectedKey = null;
            this.KeyDown += KeyPromptForm_KeyDown;
        }

        private void KeyPromptForm_KeyDown(object sender, KeyEventArgs e)
        {
            SelectedKey = e.KeyCode.ToString();
            txtSelectedKey.Text = SelectedKey;
        }
    }
}

```

ДОДАТОК В
Слайди презентації

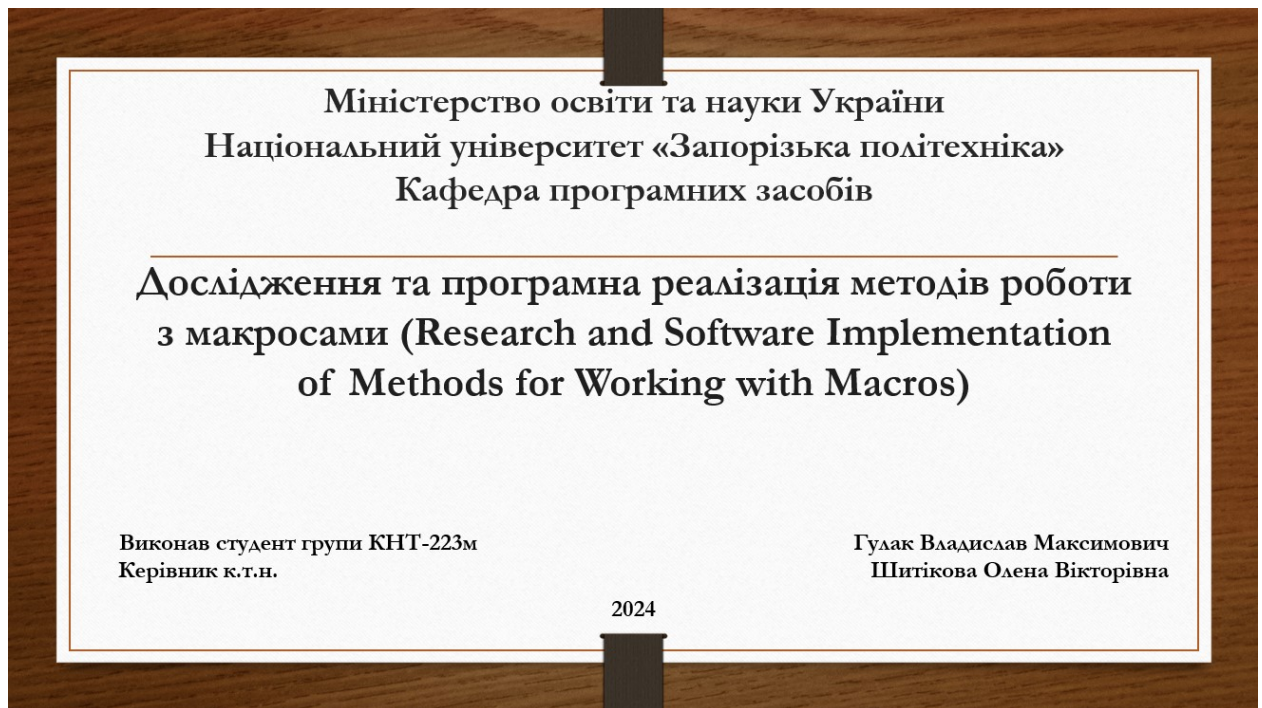


Рисунок В.1 — Слайд 1

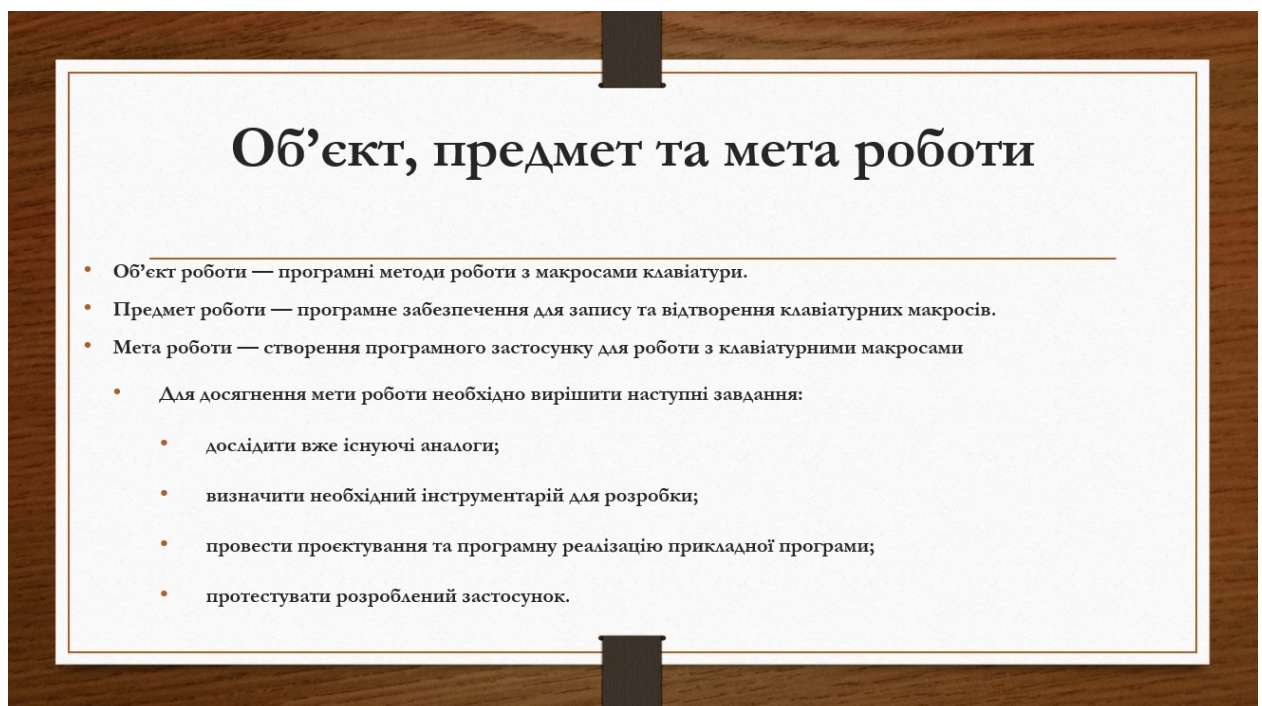


Рисунок В.2 — Слайд 2

Порівняння аналогів

Функція/Характеристика	Розроблений продукт	AutoHotkey	JitBit Macro Recorder	TinyTask	Macro Recorder
Простота використання	+	-	-	+	-
Графічний інтерфейс	+	-	+	-	+
Запис клавіш і миші	+	+	+	+	+
Підтримка профілів	+	-	-	-	-
Легкість у навчанні	+	-	-	+	-
Безкоштовність	+	+	-	+	-

Рисунок В.3 — Слайд 3

Вибір мови програмування

Критерії	C#	C++	Java
Інтеграція з Windows	Висока	Висока	Обмежена
Розробка GUI	Легка	Складна	Середня
Продуктивність	Висока	Дуже висока	Висока
Управління пам'яттю	Автоматичне	Ручне	Автоматичне
Кросплатформність	Висока	Обмежена	Висока
Складність розробки	Середня	Складна	Середня

Рисунок В.4 — Слайд 4

Вибір мови фреймворку			
Критерії	.NET	Qt	Electron
Підтримка Windows	Чудова підтримка через WinForms, WPF, UWP.	Підтримка Windows, але інтеграція з API обмежена.	Підтримка через Chromium та Node.js.
Продуктивність	Висока, оптимізовано для Windows.	Висока, особливо для C++.	Середня, через JavaScript і Chromium.
Підтримка GUI	WinForms, WPF, UWP – потужні інструменти.	Потужна підтримка GUI, зокрема для C++.	Використовує веб-технології (HTML, CSS, JavaScript).
Кросплатформеність	Підтримує Linux, macOS через .NET Core.	Кросплатформенний, підтримує Windows, Linux, macOS.	Повністю кросплатформенний завдяки Chromium та Node.js.
Інтеграція з апаратним забезпеченням	Хороша інтеграція через Windows API.	Доступ до низькорівневих функцій через C++.	Обмежена інтеграція з апаратним забезпеченням.

Рисунок В.5 — Слайд 5



Рисунок В.6 — Слайд 6

Тестування роботи програми

Новий профіль

Жодного профілю не знайдено. Зараз буде створено новий користувацький профіль.

Введіть назву нового профілю:

OK

Cancel

Новий профіль створено!

OK

Macros Binder 3.0

Збережені у профілі макроси

Команда	Збережена послідовність клавіш
F1	LCONTROL + VK_C
F2	LCONTROL + VK_V

Робота з макросами

Послідовність клавіш

Записати макрос

VK_D + VK_W + VK_E + VK_R + VK_I + VK_Y + SNAPSHOT

Прив'язати

Скинути прив'язку

Користувачі профілі

Створити профіль

Видалити профіль

Зберегти зміни у профілі

Обраний профіль

Новий профіль

Прив'язка

Натисніть кнопку для прив'язки

OK

Відміна

Підтвердження

Ви впевнені, що хочете видалити цей профіль?

Так

Ні

Профіль успішно видалено.

OK

Профіль збережено.

OK

Користувачі профілі

Створити профіль

Видалити профіль

Зберегти зміни у профілі

Обраний профіль

Новий профіль

Новий профіль2

Висновки

- В процесі виконання дипломної роботи було проведено актуальне дослідження створення програмного забезпечення для роботи з макросами. Основною метою було проаналізувати існуючі методи та реалізувати функціональні можливості для роботи з макросами.
- В результаті виконання дипломної роботи був створений програмний продукт для запису та відтворення клавіатурних макросів, який також підтримує користувацькі профілі. Програма є зручною в користуванні і повністю відповідає вимогам технічного завдання.
- Тестування показало високу надійність та стабільність продукту.
- Усі завдання дипломної кваліфікаційної роботи магіста повністю виконано.

Рисунок В.9 — Слайд 9

Дякую за увагу

Рисунок В.10 — Слайд 10