Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська кваліфікаційна робота

на тему: «Розробка програмного застосунку для аналізу та підвищення продуктивності роботи комп'ютера»

Виконав: студент IV курсу
групи 1ПІ-21б
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Ведельський В. Ю.
(прізвище та ініціали)

Керівник: к.т.н., доц. каф. ПЗ Ткаченко О. М.
(прізвище та ініціали)

Рецензент:. к.т.н., доц. каф ОТ Кадук О. В.
(прізвище та ініціали)

Допущено до захисту

Зав. кафедри ПЗ Романюк О. Н.
«09» 06 2025 р.

ВНТУ – 2025

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший (бакалаврський)
Галузь знань 12 – Інформаційні технології
Спеціальність 121 – Інженерія програмного забезпечення
Освітньо-професійна програма – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Директор ТОВ «Агро Поділля і К.»
Червяков В.В.
«21» березня 2025 року

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
д.т.н., професор Романюк О.Н.
«21» березня 2025 року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Ведельському Володимиру Юрійовичу

1. Тема роботи – Розробка програмного застосунку для аналізу та підвищення продуктивності роботи комп'ютера.

Керівник роботи: Ткаченко Олександр Миколайович, к.т.н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від « 20 » березня 2025 р. № 97.

2. Строк подання студентом роботи: 2 червня 2025р.

3. Вихідні дані до роботи: методи перегляду дерева реєстру, методи редагування дерева реєстру, методи моніторингу за навантаженням центрального процесора та оперативної пам'яті, методи менеджера процесів, методи менеджера автозапуску, методи очищення дискового простору, методи резервного копіювання реєстру.

4. Зміст текстової частини: алгоритми перегляду та редагування дерева реєстру, алгоритми слідкування за навантаженням на комплектуючі, алгоритм пошуку програм у автозапуску, алгоритм відслідковування процесів, алгоритм очищення дискового простору, алгоритм створення резервних копій, тестування системи, розробка інструкції користувача.

5. Перелік ілюстративного матеріалу: графічні інтерфейси аналогів програмного застосунку, основні етапи алгоритму для відображення дерева реєстру, ключові функції та методи модулів програмного застосунку, графічний інтерфейс.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Ткаченко О.М., к.т.н., доцент кафедри ПЗ	24.03.2025	01.06.2025

7. Дата видачі завдання 24 березня 2025р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи	Примі
1.	Аналіз розвитку систем моніторингу та процесів підвищення продуктивності роботи комп'ютера	25.03.25-05.04.25	Викон
2.	Розробка моделі та методів програмного забезпечення	06.04.25-13.04.25	Викон
3.	Розробка програмних модулів	14.04.25-07.05.25	Виконан
4.	Тестування та практичне застосування програмного забезпечення	08.05.25-20.05.25	Виконан
5.	Оформлення матеріалів до захисту БКР	21.05.25-30.05.25	Виконан

Студент (підпис) **Ведельський В** (прізвище та ініціали)

Керівник бакалаврської кваліфікаційної роботи (підпис) **Ткаченко О.М.** (прізвище та ініціали)

АНОТАЦІЯ

УДК 004.912.032.26

Ведельського В.Ю. Розробка програмного застосунку для аналізу та підвищення продуктивності роботи комп’ютера: бакалаврська кваліфікаційна робота зі спеціальності 121 – інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця: ВНТУ, 2025. 116 с.

На укр. мові. Бібліогр.: 26 назв; рис.: 38; табл.: 2.

У бакалаврській кваліфікаційній роботі було розроблено програмний засіб для моніторингу та підвищення продуктивності роботи комп’ютера. Проєкт включає розробку модулів для визначення невалідних записів у реєстрі, аналізу продуктивності комп’ютера та графічного інтерфейсу. Додаток призначений для підвищення продуктивності роботи комп’ютера а також моніторингу задіяних ресурсів. Програмні модулі, розроблені в роботі, можуть бути використані в побутових умовах користування комп’ютером. Додаток розроблено з використанням мов програмування C# та середовища Visual Studio Code. Для здійснення аналізу параметрів у реєстрі були використані засоби Win32.

ABSTRACT

Vedelskiy V.Y. Development of a software application for analysis and increasing computer productivity. Bachelor's thesis in specialty 121 Software Engineering, educational program - software engineering. Vinnytsia: VNTU, 2025. 116p.

In Ukrainian. Bibliography: 26 titles; figures: 38; tables: 2.

In this project, a software tool for monitoring and improving computer performance has been developed. The project includes the development of modules for detecting invalid registry entries, analyzing computer performance, and designing a graphical user interface. The application is designed to enhance computer performance and monitor resource usage. The software modules developed in this project can be used in everyday computer operation. The application was developed using the C# programming language and the Visual Studio Code environment. Win32 tools were used to analyze registry parameters.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Актуальність створення програмного застосунку для аналізу та підвищення продуктивності роботи комп’ютера обумовлена необхідністю виявлення потенційних проблем безпеки та потребою в оптимізації ресурсів реєстру, що значно підвищує безпеку, надійність та продуктивність операційної системи, а також надає можливість моніторингу задіяних операційною системою ресурсів [1].

Сучасні комп’ютери використовуються для виконання широкого спектра завдань – від офісної роботи до складних обчислень, 3D-моделювання та комп’ютерних ігор. Однак з часом продуктивність системи може знижуватися через різні причини, такі як накопичення непотрібних файлів, перевантаження оперативної пам’яті, конфлікти в реєстрі та неефективне управління ресурсами [2].

Для забезпечення стабільної та ефективної роботи комп’ютера необхідно проводити регулярний моніторинг використання ресурсів і своєчасно вживати заходи щодо їх оптимізації. Саме тому виникає потреба у створенні програмного застосунку, який дозволить аналізувати продуктивність системи та пропонувати шляхи її покращення [3].

Наявні аналогічні програмні засоби не поєднують у собі одночасний моніторинг та процеси підвищення продуктивності операційної системи. До того ж, більшість наявних засобів є платними, а у безкоштовній версії мають значно зменшений функціонал, тому розробка програмних засобів для моніторингу та підвищення продуктивності роботи комп’ютера є актуальною.

Зв’язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження. Метою бакалаврської кваліфікаційної роботи є підвищення продуктивності роботи комп’ютера та збільшення вільного дискового простору шляхом розробки програмного застосунку, який дозволяє моніторити використання ресурсів системи та здійснювати їх оптимізацію. Робота

передбачає створення різних модулів, які будуть інтегровані в програмний комплекс. Ці модулі включають модуль аналізу продуктивності системи, модуль виявлення та оптимізації споживання ресурсів, а також модуль графічного інтерфейсу для зручної взаємодії з користувачем [4].

Для досягнення поставленої мети необхідно розв’язати такі задачі:

- розробити алгоритм аналізу продуктивності, який забезпечать оцінку навантаження на систему та ефективність оптимізації використання ресурсів;
- розробити модуль моніторингу системних ресурсів, який буде відстежувати, аналізувати та відображати основні параметри продуктивності комп’ютера;
- розробити модуль графічного інтерфейсу, який буде зручним у використанні для користувачів, забезпечуючи наочне відображення стану системи;
- провести інтеграцію різних модулів в єдиний програмний засіб, який дозволить проводити комплексний аналіз продуктивності комп’ютера.

Об'єктом дослідження є процес аналізу системних ресурсів і підвищення продуктивності роботи комп’ютера.

Предметом дослідження є методи та засоби аналізу та підвищення продуктивності роботи комп’ютера.

Методи дослідження. У процесі дослідження застосовувались: методи об’єктно-орієнтованого програмування для створення модульної архітектури, методи системного аналізу для оцінки впливу програми на продуктивність і стабільність системи; методи комп’ютерного моделювання для перевірки ефективності запропонованих рішень.

Наукова новизна отриманих результатів:

1. Отримав подальшого розвитку метод очищення сміття у дисковому просторі, в якому, на відміну від існуючих, диск очищується не лише від залишкових файлів програмного забезпечення, а додатково і від файлів, тимчасово створених у системі стороннім програним забезпеченням, що дозволило збільшити

розмір вільного дискового простору та підвищити продуктивність роботи SSD або HDD дисків.

2. Отримав подальшого розвитку метод створення резервних копій основних розділів реєстру, який відрізняється від існуючих можливістю вибіркового копіювання основних ключів реєстру, що дозволило зменшити час очікування створення файлу резервної копії.

Практичне значення роботи полягає у створенні програмного забезпечення, яке дозволить моніторити завантаженість операційної системи, а також вживати заходів щодо підвищення її продуктивності. Розроблені програмні модулі можуть бути використані як у вигляді окремої утиліти, так і у складі інших системних застосунків [5].

Особистий внесок здобувача. Усі наукові результати, викладені у бакалаврській кваліфікаційній роботі, отримані автором особисто. Наукові праці опубліковано одноосібно.

Апробація матеріалів бакалаврської кваліфікаційної роботи. Результати роботи було представлено на LIV Всеукраїнській науково-технічній конференції підрозділів Вінницького національного технічного університету (Вінниця, 2025 р.) і XIII Міжнародній науково-практичній конференції з інформаційних систем та технологій Infocom Advanced Solutions 2025 (Київ, 2025 р.).

Публікації. Результати роботи опубліковано в матеріалах LIV Всеукраїнської науково-технічної конференції підрозділів Вінницького національного технічного університету і XIII Міжнародної науково-практичної конференції з інформаційних систем та технологій Infocom Advanced Solutions 2025 [6, 7].

1 АНАЛІЗ РОЗВИТКУ ТЕХНОЛОГІЙ МОНІТОРИНГУ ТА ПРОЦЕСУ ПІДВИЩЕННЯ ПРОДУКТИВНОСТІ РОБОТИ КОМП'ЮТЕРА

1.1 Аналіз стану технологій моніторингу та процесу підвищення продуктивності роботи комп'ютера

Сучасний розвиток інформаційних технологій вимагає постійного вдосконалення методів аналізу та оптимізації продуктивності комп'ютерних систем. Незважаючи на зростаючу потужність апаратного забезпечення, проблеми ефективного використання ресурсів залишаються актуальними. Зниження швидкодії комп'ютера може бути викликане перевантаженням процесора, надмірним використанням оперативної пам'яті, наявністю помилкових записів у системному реєстрі та іншими факторами [8].

Одним із ключових аспектів розвитку технологій моніторингу продуктивності є зростання швидкості та потужності обчислювальних систем, що дозволяє ефективно аналізувати роботу комп'ютера в режимі реального часу. Завдяки сучасним програмним засобам можна відстежувати навантаження на процесор, оперативну пам'ять, дискову систему та графічний адаптер, а також визначати джерела можливих проблем [9].

Використання програмного забезпечення для моніторингу та оптимізації продуктивності надає значні переваги порівняно з традиційними методами ручного налаштування. Головною перевагою таких систем є автоматизований підхід до аналізу продуктивності, що дозволяє виявляти та усувати вузькі місця в роботі комп'ютера без втручання користувача [10].

Також сучасні програмні засоби можуть включати різні типи аналізу: виявлення перевантажених процесів, оптимізацію споживання пам'яті, очищення реєстру від непотрібних записів тощо. Це дозволяє отримувати комплексну інформацію про стан системи та своєчасно реагувати на зниження продуктивності.

Ще однією важливою перевагою спеціалізованого програмного забезпечення є підвищення стабільності роботи системи. Оптимізація використання ресурсів

допомагає запобігти зависанню програм, надмірному споживанню енергії та передчасному зносу апаратних компонентів.

Отже, аналіз стану технологій моніторингу та підвищення продуктивності роботи комп'ютера підтверджує важливість використання спеціалізованого програмного забезпечення для автоматизованого управління ресурсами системи. Розробка таких програмних засобів дозволяє не лише покращити швидкодію комп'ютерів, а й підвищити їхню енергоефективність, стабільність та довговічність. Враховуючи постійний розвиток технологій у цій сфері, важливо використовувати сучасні методи аналізу та оптимізації для забезпечення максимальної продуктивності та зручності користувачів.

1.2 Порівняльний аналіз аналогів

Найбільш схожим аналогом застосунку є вбудована програма у Windows, що називається Scanreg [11]. При успішному запуску комп'ютера, програма створює резервну копію системних файлів і налаштувань реєстру. Основною функцією програми є перевірка при запуску комп'ютера недійсних записів в системному реєстрі. У разі знаходження недійсних записів програма перевірки реєстру автоматично відновлює зроблену в минулий день резервну копію. За допомогою цієї програми також можна відновлювати окремі файли, а не систему в цілому.

Функціонал програми є досить зручним і корисним, оскільки дозволяє оптимізувати і у разі непередбачуваного збою в системі автоматично відновити її використовуючи резервну копію.

Проте, програма містить і ряд недоліків. Наприклад, якщо в реєстрі присутній параметр, який посилається на неіснуючий файл, то такий запис неможливо виправити за допомогою Scanreg, тобто система буде уповільнюватись з часом, накопичуючи все більше і більше параметрів у реєстрі, що с часом негативно вплине на продуктивність комп'ютера. Також програма не має функціоналу моніторингу використання ресурсів системою.

Наступним найбільш популярним аналогом є програма CCleaner. Це популярний інструмент для очищення та оптимізації системи Windows. Окрім

перевірки валідності записів у реєстрі, також має функцію аналізу та виправлення проблем у реєстрі, включаючи виявлення та видалення непотрібних або пошкоджених параметрів. Також застосунок пропонує можливість створення резервних копій реєстру, щоб уникнути можливих проблем.

Серед недоліків програми варто зазначити, що вона не містить функціоналу для моніторингу за використанням ресурсів системою, а також те, що в минулому у цьому програмному забезпеченні були виявлені певні вразливості у безпеці. На рисунку 1.1 показано графічний інтерфейс CCleaner [12].

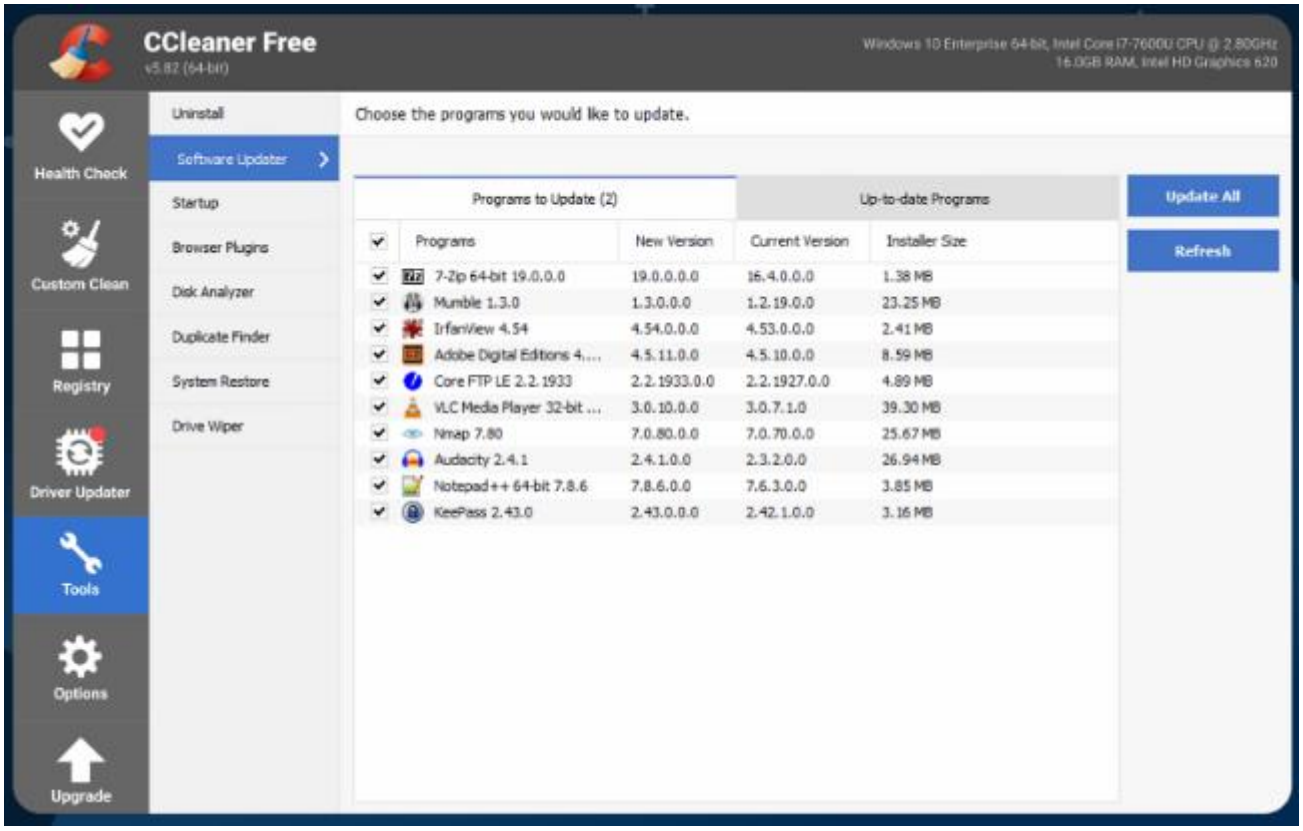


Рисунок 1.1 – Графічний інтерфейс CCleaner.

Також, досить популярним є застосунок Auslogics Registry Cleaner, основним функціоналом програми є сканування реєстру. Проте, деякі користувачі програми зазначають, що програма «завищує» значення виявлених проблем для стимулюванню користувачів користуванням її послугами [13]. Також, ця програма

не має потрібного функціоналу для моніторингу за використанням ресурсів системи. Графічний інтерфейс Auslogics Registry Cleaner показано на рисунку 1.2.

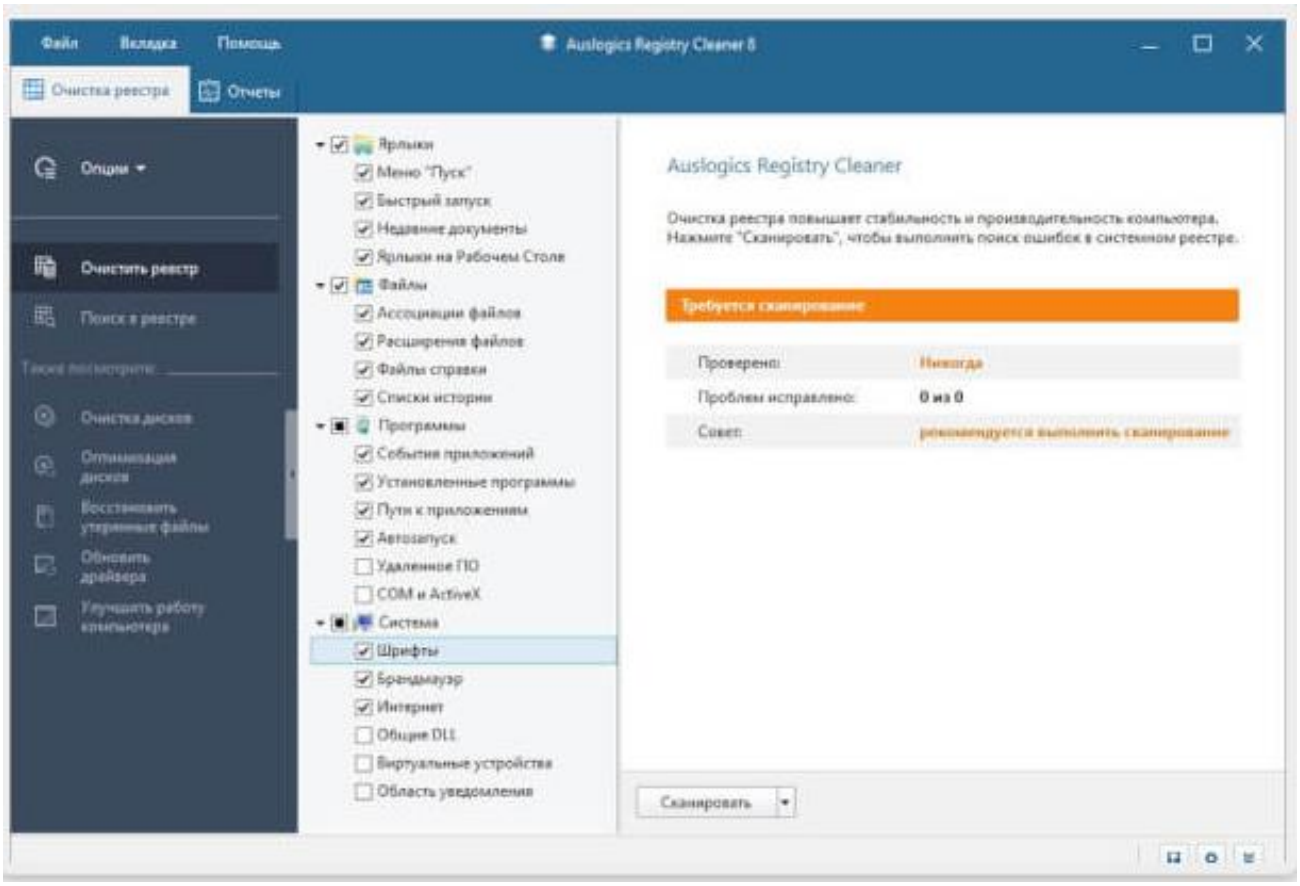


Рисунок 1.2 – Графічний інтерфейс Auslogics Registry Cleaner

Також ще одним популярним аналогом програмного застосунку є програма Reg Organizer. Програмний застосунок орієнтований на очистку реєстру, тобто він дозволяє видаляти застарілі та непотрібні записи з реєстру, виправлення помилок реєстру, аналіз реєстру та деякі інструменти оптимізації. Також, за допомогою застосунку можна видаляти повністю всі сліди програм, після їх звичайного видалення. Застосунок дозволяє керувати настройками встановлених програм, що є досить зручним [14].

Недоліками програмного забезпечення є відсутність функціоналу для аналізу використання ресурсів системи. Графічний інтерфейс Reg Organizer показано на рисунку 1.3.

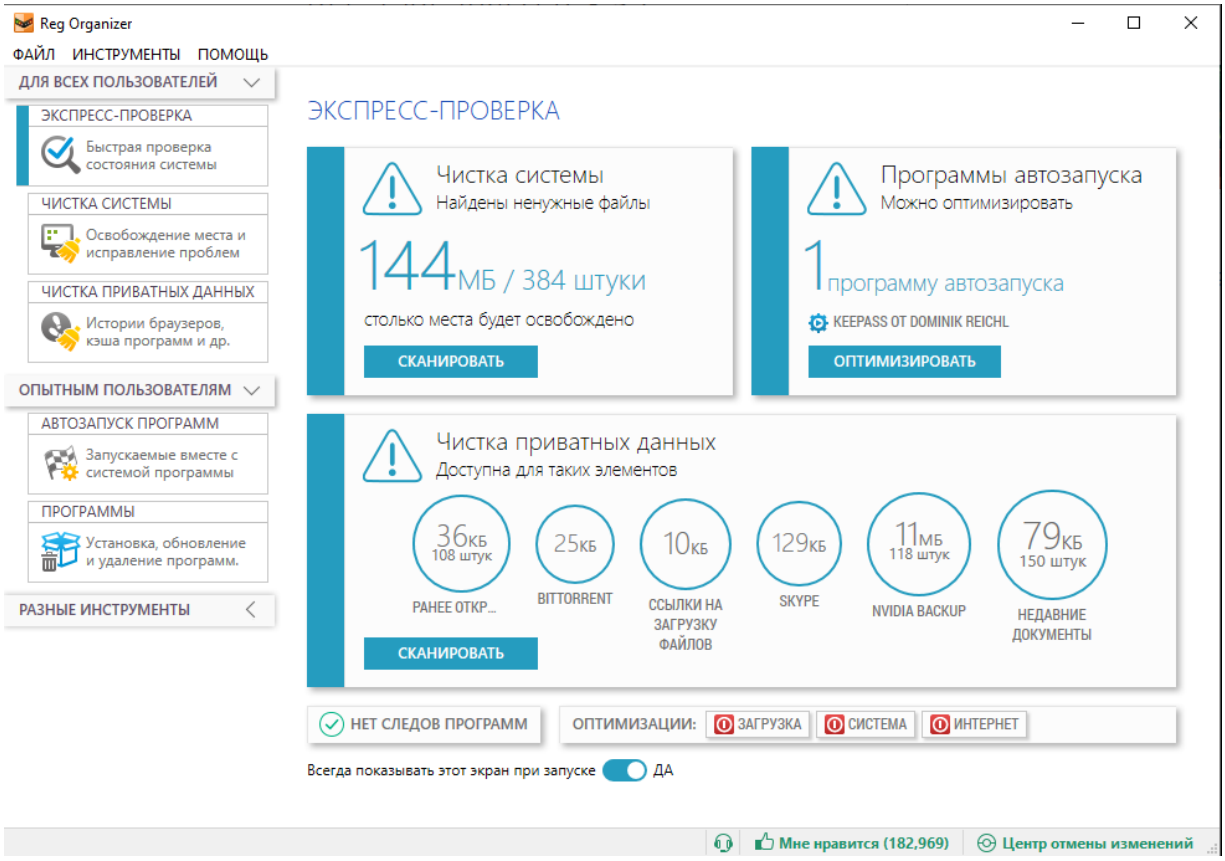


Рисунок 1.3 – Графічний інтерфейс Reg Organizer

У таблиці 1.1 було проведено порівняльний аналіз функціональності аналогів розроблюваного програмного застосунку задля більш зрозумілого порівняння застосунків.

Таблиця 1.1 – Порівняльний аналіз функціональності

Критерій	CCleaner	Auslogics Registry Cleaner	Windows Scanreg	Reg Organizer	Власна розробка
Резервна копія	1	1	1	0	1
Очистка дискового простору	1	0	0	1	1

Продовження таблиці 1.1

Критерій	Ccleaner	Auslogics Registry Cleaner	Windows Scanreg	Reg Organizer	Власна розробка
Моніторинг ресурсів	0	0	1	0	1
Перевірка програм у автозапуску	1	1	0	1	1
Менеджер процесів	1	1	0	0	1
Сумарний коефіцієнт	4	3	1	2	5

Отже, порівнявши розроблюваний застосунок із його аналогами, можна зробити висновок, що він позбавлений деяких недоліків аналогів, а також має додатковий розширений функціонал.

1.3 Аналіз методів розв’язання задачі

Розробка програмного засобу для моніторингу та підвищення продуктивності роботи комп’ютера вимагає комплексного підходу до вирішення завдань, пов’язаних із аналізом продуктивності системи, виявленням проблемних місць та їх усуненням. Для цього існує низка методів, кожен із яких має свої переваги та недоліки. У цьому розділі проаналізовано найбільш ефективні підходи до розробки таких програмних рішень.

Одним із ключових аспектів оптимізації продуктивності є моніторинг використання ресурсів у реальному часі. Для цього можна використовувати вбудовані інструменти операційної системи, такі як Windows Performance Monitor, який надає детальну інформацію про завантаження процесора, оперативної пам’яті та дискової підсистеми. Однак цей метод має досить обмежений функціонал.

Інший підхід – використання алгоритмів штучного інтелекту та машинного навчання для аналізу поведінки системи. Алгоритми можуть навчатися на основі зібраних даних про роботу комп’ютера та прогнозувати можливі падіння продуктивності. Такий метод використовується в комерційних продуктах, таких як Windows Defender Performance Analyzer та антивірусні програми з функціями оптимізації. Однак, використання штучного інтелекту потребує значних обчислювальних ресурсів, що може навпаки вплинути на продуктивність системи.

Найбільш ефективним підходом є комбіноване використання аналізу системних ресурсів, визначення проблем у реєстрі та оптимізації процесів у реальному часі.

Розглянувши переваги та недоліки кожного підходу, було вирішено використовувати поєднання аналізу продуктивності на рівні системних процесів та реєстру разом із методами оптимізації роботи ресурсів у реальному часі. Такий підхід дозволить створити ефективний інструмент для підвищення продуктивності комп’ютера, що працюватиме без значного навантаження на систему та забезпечуватиме стабільну роботу користувацького середовища.

1.4 Постановка задач бакалаврської кваліфікаційної роботи

Проаналізувавши переваги та недоліки існуючих програмних засобів для моніторингу та підвищення продуктивності роботи комп’ютера, було визначено такі основні завдання, необхідні для успішної реалізації програмного забезпечення:

- розробити алгоритм аналізу продуктивності, який забезпечать оцінку навантаження на систему та ефективність оптимізації використання ресурсів;
- розробити модуль моніторингу системних ресурсів, який буде відстежувати, аналізувати та відображати основні параметри продуктивності комп’ютера;
- розробити модуль графічного інтерфейсу, який буде зручним у використанні для користувачів, забезпечуючи наочне відображення стану системи;

- провести інтеграцію різних модулів в єдиний програмний засіб, який дозволить проводити комплексний аналіз продуктивності комп’ютера;

1.5 Висновки

У першому розділі було розглянуто стан програмних засобів для моніторингу та підвищення продуктивності комп’ютера. Були розглянуті програмні засоби такі як Windows Scanreg, CCleaner, Auslogics Registry Cleaner, Reg Organizer. Кожен з цих засобів має свої переваги і недоліки, що були проаналізовані.

Проаналізувавши переваги та недоліки існуючих програмних засобів, було сформульовано завдання для подальшої розробки власного програмного забезпечення. Ці завдання включають розробку алгоритму аналізу стану реєстру та виявлення невалідних записів, які можуть впливати на продуктивність системи, оцінки продуктивності комп’ютера, що включатимуть аналіз використання процесора, оперативної пам’яті, дискової підсистеми та інших ресурсів, розробку графічного інтерфейсу. Таким чином було доведено доцільність розробки власного програмного рішення. На основі отриманої інформації було сформовано задачі, які необхідно виконати для розробки власних програмних засобів.

2 РОЗРОБКА МОДЕЛІ ТА МЕТОДІВ ПРОГРАМНОГО ПРОДУКТУ

2.1 Аналіз вхідних даних системи

Для реалізації програмного застосунку для моніторингу та підвищення продуктивності роботи системи буде використано мову програмування C# у поєднанні з технологією Windows Forms, а також вбудовані засоби Windows для роботи з системними ресурсами, такими як реєстр, процеси та файлова система. C# – це сучасна об’єктно-орієнтована мова програмування, розроблена Microsoft, яка використовується для створення надійних і продуктивних додатків для платформи Windows [15], завдяки своїй інтеграції з .NET Framework. Windows Forms – це бібліотека для створення графічних інтерфейсів користувача, що забезпечує зручний і швидкий спосіб розробки настільних додатків із підтримкою широкого набору компонентів для взаємодії з користувачем [16]. Поєднання C# та Windows Forms дозволить створити програмний засіб, який забезпечить ефективний моніторинг системних ресурсів і підвищення продуктивності роботи операційної системи.

Розробка програмного застосунку для моніторингу та підвищення продуктивності системи передбачає кілька ключових етапів. На першому етапі буде створено модуль для зручної та інтуїтивно зрозумілої взаємодії користувача із системою. Цей модуль відповідатиме за навігацію між основними функціями застосунку, такими як перегляд системного реєстру, управління автозапуском, моніторинг процесів, створення резервних копій і очищення тимчасових файлів.

Далі буде розроблено модуль моніторингу системних ресурсів, який відповідатиме за відстеження використання центрального процесора та оперативної пам’яті. Цей модуль використовуватиме вбудовані класи .NET, такі як PerformanceCounter для збору даних про завантаженість CPU та ComputerInfo для оцінки використання RAM, із подальшим відображенням інформації у вигляді прогрес-барів для зручного аналізу користувачем. Наступним етапом стане створення модуля управління автозапуском, який дозволить користувачу

переглядати та редагувати програми, що запускаються разом із системою, використовуючи доступ до відповідних ключів системного реєстру [17].

Ще одним важливим модулем стане модуль управління процесами, який надасть користувачу можливість переглядати активні процеси, групувати їх за назвою, завершувати окремі процеси або цілі дерева процесів, а також змінювати їх пріоритет для оптимізації роботи системи. Для реалізації цього модуля буде використано клас Process із .NET Framework, що забезпечує доступ до інформації про запущені процеси та їхні характеристики, такі як обсяг використаної пам’яті та ідентифікатор процесу.

Додатково буде створено модуль для створення резервних копій системного реєстру та очищення тимчасових файлів. Модуль резервного копіювання використовуватиме утиліту reg.exe для експорту ключів реєстру у файли, а модуль очищення дозволить видаляти тимчасові файли з комп’ютера звільняючи місце на диску та підвищуючи продуктивність системи. Для візуалізації структури реєстру та значень буде розроблено модуль редактора реєстру, який дозволить переглядати та редагувати ключі та значення в зручному вигляді.

Розробка цих програмних модулів потребує глибокого розуміння системного програмування, роботи з операційною системою Windows та програмування на C#. Для реалізації будуть використані інструменти та бібліотеки, зокрема Visual Studio як основне середовище розробки, а також вбудовані класи .NET Framework для роботи з системними ресурсами. Програмні модулі будуть ретельно протестовані для забезпечення їхньої стабільності, швидкості роботи та точності обробки даних [18].

Отже, розробка програмного застосунку для моніторингу та підвищення продуктивності роботи системи на основі C# і Windows Forms є комплексним завданням, яке включає кілька етапів. Модулі мають бути ретельно спроєктовані, реалізовані та протестовані, щоб забезпечити користувачам зручний і ефективний інструмент для управління системними ресурсами. Завдяки використанню сучасних технологій цей програмний продукт зможе задовольнити потреби

користувачів, допомагаючи оптимізувати продуктивність їхніх систем і відкриваючи нові можливості для ефективного управління Windows.

2.2 Розробка моделі системи

Для ефективного створення програмного застосунку для моніторингу та підвищення продуктивності роботи системи необхідно чітко визначити його структуру, функціональні можливості та взаємодію між компонентами. З цією метою на початковому етапі розробки було проведено аналіз вимог до системи, що дозволило сформувати основні завдання, які має виконувати програмне забезпечення. До ключових вимог відносяться: забезпечення зручного та інтуїтивно зрозумілого інтерфейсу для користувача, можливість моніторингу системних ресурсів у реальному часі, управління автозапуском і процесами, створення резервних копій системного реєстру, очищення тимчасових файлів, а також перегляд і редагування структури реєстру. Ці вимоги стали основою для проєктування архітектури програми.

Програмний застосунок розроблено з використанням модульного підходу, що забезпечує гнучкість, легкість у підтримці та масштабуванні. Кожен модуль відповідає за окремий функціонал: моніторинг ресурсів, управління автозапуском, контроль процесів, створення резервних копій, очищення системи та редагування реєстру. Для реалізації було обрано мову програмування C# у поєднанні з технологією Windows Forms, що дозволяє створювати оптимально адаптовані застосунки для роботи в операційній системі Windows. Використання C# забезпечує доступ до широкого набору вбудованих класів .NET Framework, таких як PerformanceCounter, Process, Registry та ComputerInfo, які необхідні для взаємодії з системними ресурсами.

Для більш детального розуміння роботи модулів програмного застосунку для аналізу та підвищення продуктивності роботи комп’ютера було вирішено розробити модель системи у вигляді UML діаграми діяльності, яку показано на рисунку 2.1.

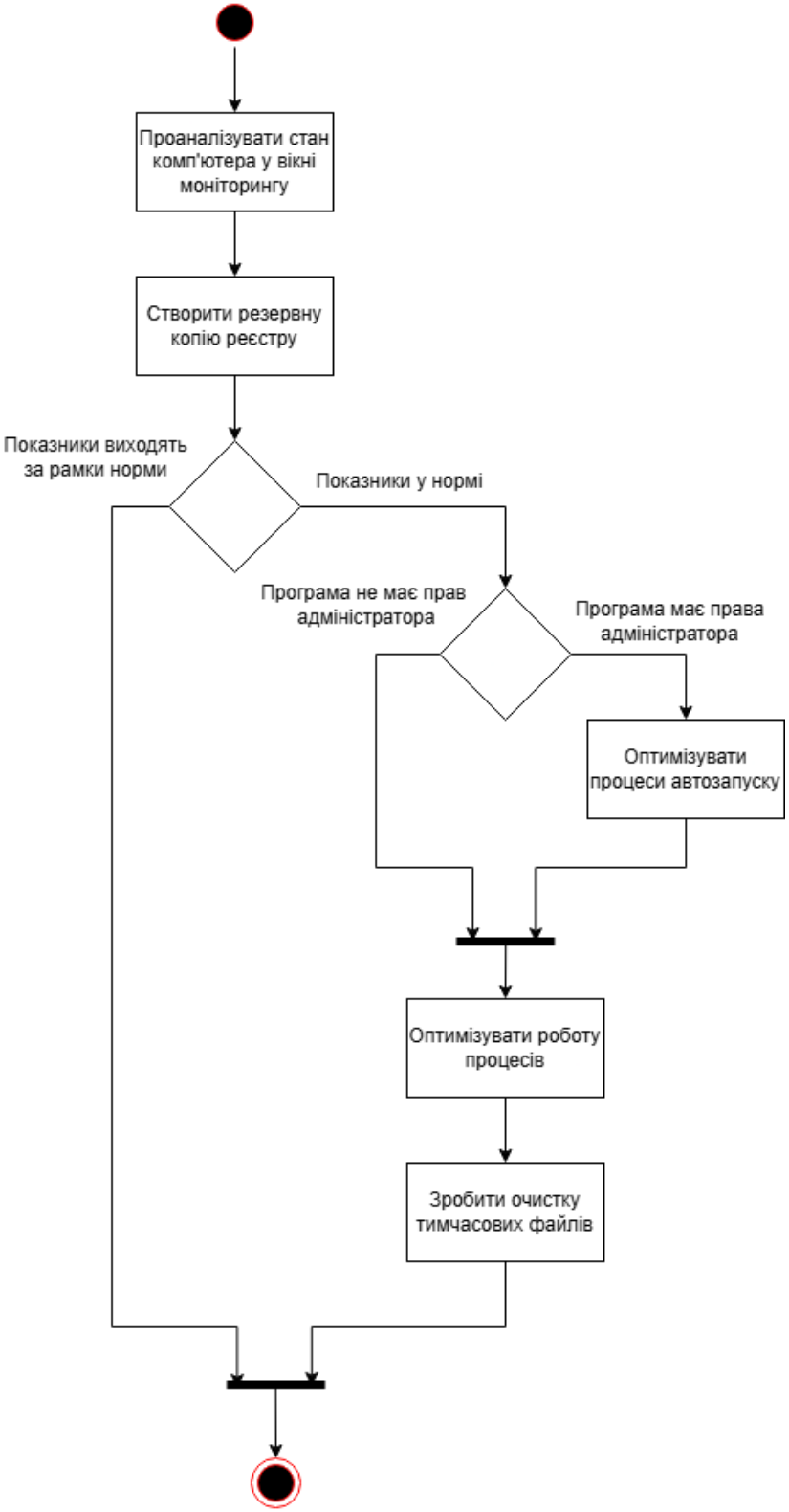


Рисунок 2.1 – UML діаграма діяльності

Таким чином, розробка моделі системи передбачала комплексний підхід до аналізу вимог, проєктування архітектури та створення зручного інтерфейсу.

2.3 Розробка методу очищення сміття

Метод очищення сміття є частиною форми CleanupForm, яка призначена для видалення непотрібних файлів із системи, зокрема тимчасових файлів, що накопичуються в директорії %TEMP%. Це дозволяє користувачам оптимізувати використання дискового простору без необхідності вручну шукати та видаляти файли. У контексті програми RegCheck, яка фокусується на управлінні системними ресурсами, функція очищення сміття доповнює її, надаючи комплексний інструмент для підтримки продуктивності системи.

Метод призначений для:

1. Видалення тимчасових файлів і директорій, створених програмами.

2. Надання зрозумілого інтерфейсу для вибору та підтвердження дії.

3. Звітування про результати очищення, включаючи кількість видалених файлів і звільнений простір.

Метод CleanButton_Click є основним обробником події, який виконує очищення після натискання кнопки "Очистити". Він включає кілька етапів, кожен із яких має чітку роль у процесі.

Етапи роботи методу:

1. Перевірка вибору категорії

- Перевіряє, чи позначений чекбокс для очищення тимчасових файлів. Якщо чекбокс не активний, відображається діалогове вікно з повідомленням про необхідність вибору хоча б однієї опції.
- Запобігає запуску процесу очищення без явного вибору користувача, що забезпечує контроль і безпеку.

2. Підтвердження дії

- Відображає діалогове вікно, яке попереджає, що дія незворотна, і просить користувача підтвердити її вибором "Так" або "Ні".
- Захищає від випадкового видалення файлів, що є важливим для системних утиліт, де помилки можуть мати наслідки.

3. Ініціалізація змінних для звітування

- Створює дві змінні: одну для підрахунку кількості видалених файлів і другу для обсягу звільненого простору в байтах.
- Дозволяє відстежувати результати очищення для подальшого звітування користувачу.

4. Процес очищення тимчасових файлів

- Отримує шлях до директорії %TEMP% за допомогою системного методу.
- Перебирає всі файли в цій директорії, видаляючи кожен і додаючи його розмір до загального обсягу звільненого простору, а також збільшуючи лічильник файлів.
- Перебирає всі піддиректорії в %TEMP%, видаляючи їх разом із вмістом.

5. Звіт про результати

- Відображає діалогове вікно з підсумками очищення, включаючи кількість видалених файлів і звільнене місце, переведене в мегабайти з двома знаками після коми.

Функція очищення є частиною ширшої екосистеми програми і доступна через кнопку "Очищення" на головній формі. Ця кнопка розташована на панелі інструментів разом із кнопками для інших функцій, таких як редагування реєстру чи управління процесами. При натисканні кнопки створюється і відкривається екземпляр CleanupForm, що дозволяє користувачу перейти до процесу очищення.

2.4 Розробка методу створення резервної копії

Метод створення резервних копій у розроблюваному програмному застосунку дозволяє створювати резервні копії вибраних основних ключів системного реєстру Windows у форматі .reg файлів. Цей функціонал реалізовано через методи BackupButton_Click і CreateRegistryBackups у класі RegistryEditorForm, а також за допомогою форми RegistryBackupSelectionForm, яка дозволяє користувачеві вибирати основні ключі для подальшого резервного копіювання.

Послідовність дій методу.

1. Відкриття діалогового вікна вибору ключів. При натисканні на відповідну кнопку, відкривається форма RegistryBackupSelectionForm. Форма містить прапорці для кожного основного ключа реєстру, дозволяючи користувачеві вибрати, які ключі потрібно зберегти. За замовчуванням усі прапорці активовані.

2. Вибір папки для збереження. Після підтвердження вибору ключів відкривається діалог FolderBrowserDialog для вибору папки, куди будуть збережені резервні копії.

3. Для кожного вибраного ключа реєстру метод CreateRegistryBackups викликає утиліту reg.exe із командою export, яка експортує ключ у файл із розширенням .reg.

На відміну від багатьох стандартних інструментів, наприклад, вбудованого редактора реєстру Windows, які дозволяють створювати резервні копії або всіх ключів, або лише одного вручну, розроблюване програмне забезпечення дає змогу вибрати довільну комбінацію основних ключів.

Функція резервного копіювання є частиною комплексного застосунку, який також включає редактор реєстру, диспетчер завдань, менеджер автозапуску, менеджер процесів і очищення системи. Автоматичне формування імен файлів із датою спрощує організацію резервних копій.

Метод створення резервних копій у програмному застосунку є зручним, гнучким і користувацько-орієнтованим рішенням для створення резервних копій ключів реєстру. Його ключова перевага – можливість вибору потрібних ключів через графічний інтерфейс.

2.5 Розробка алгоритму роботи системи

Для розробки програмного застосунку для аналізу та підвищення продуктивності роботи комп’ютера було розроблено загальний алгоритм роботи застосунку, блок-схему якого показано на рисунку 2.2.

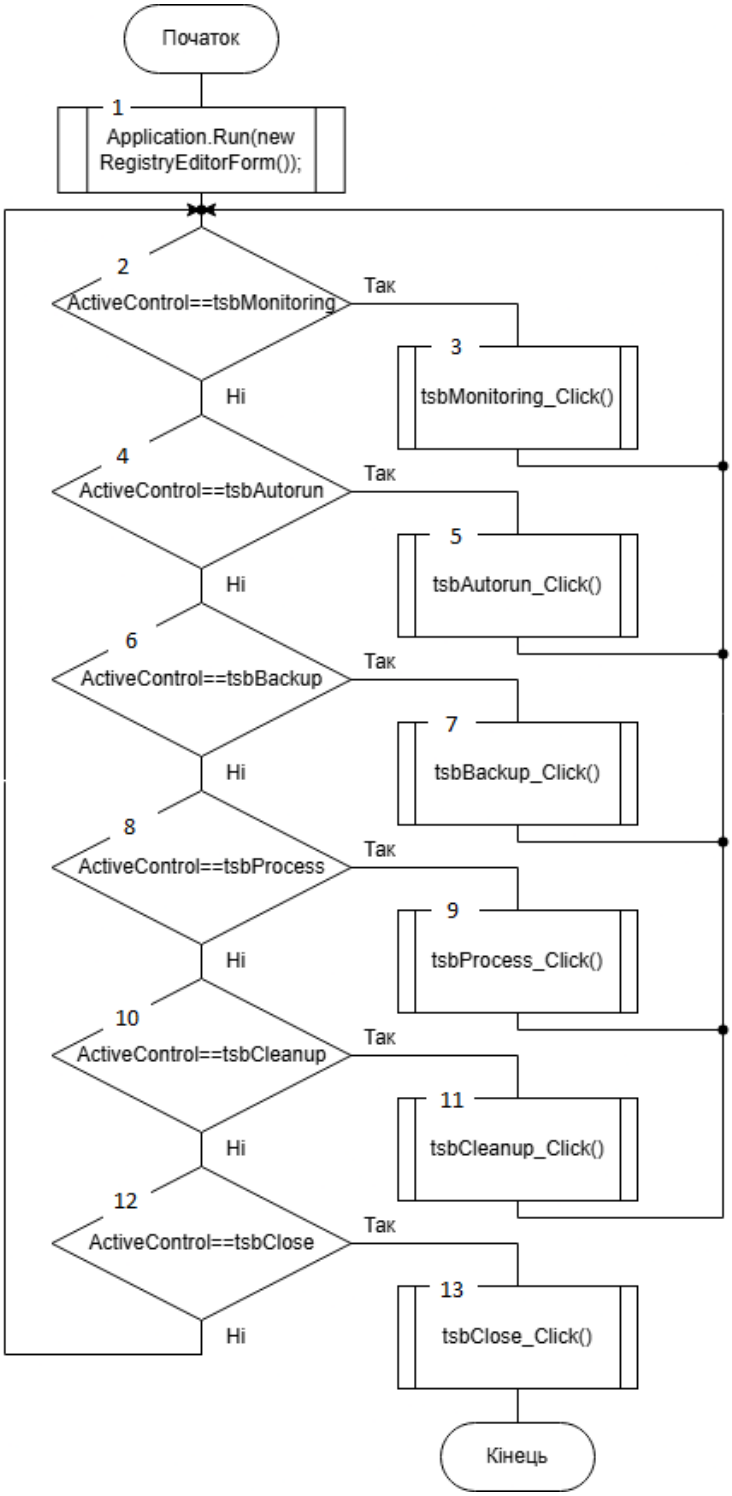


Рисунок 2.2 – Блок-схема загального алгоритму роботи застосунку

Згідно алгоритму роботи програмного застосунку, після запуску форми користувач має можливість обрати потрібний йому функціонал, тобто визвати відповідні вікна натисканням на кнопки, можливо відкрити вікна моніторингу системний ресурсів, вікно автозапуску програм, вікно створення резервної копії

системного реєстру, вікно очистки тимчасових файлів операційної системи, а також закриття програми.

2.6 Висновки

У другому розділі було проведено аналіз вхідних та вихідних даних програмних засобів. Розроблено модель системи та створено UML-діаграму діяльності. Вдосконалено метод очищення сміття у дисковому просторі, в якому, на відміну від існуючих, диск очищується не лише від залишкових файлів програмного забезпечення, а додатково і від файлів, тимчасово створених у системі стороннім програним забезпеченням, що дозволило збільшити розмір вільного дискового простору та підвищити продуктивність роботи SSD або HDD дисків. Вдосконалено метод створення резервних копій основних розділів реєстру, який відрізняється від існуючих можливістю вибіркового копіювання основних ключів реєстру, що дозволило зменшити час очікування створення файлу резервної копії. Розроблено основний алгоритм роботи системи.

3 РОЗРОБКА ПРОГРАМНИХ МОДУЛІВ ДЛЯ АНАЛІЗУ ТА ПІДВИЩЕННЯ ПРОДУКТИВНОСТІ РОБОТИ КОМП'ЮТЕРА

3.1 Варіантний аналіз і обгрунтування вибору засобів для реалізації програми

При розробці програмного застосунку для моніторингу та підвищення продуктивності роботи системи важливо обрати найбільш оптимальні технології для реалізації кожного модуля. C# у поєднанні з технологією Windows Forms є досить зручним інструментом для створення настільних додатків із графічним інтерфейсом, адаптованим для операційної системи Windows. Використання C# забезпечує доступ до широкого набору вбудованих класів .NET Framework, таких як PerformanceCounter, Process, Registry і ComputerInfo, що дозволяють ефективно взаємодіяти з системними ресурсами, моніторити їхній стан і управляти ними. Windows Forms надає зручний набір компонентів для створення інтуїтивно зрозумілого інтерфейсу користувача, включаючи кнопки, панелі, TreeView, ListView і прогрес-бари, які використовуються для відображення даних у реальному часі.

Visual Studio – це інтегроване середовище розробки від Microsoft, яке забезпечує великий набір інструментів для створення, налагодження та тестування програмного забезпечення на C#. Воно підтримує Windows Forms і дозволяє розробникам швидко створювати графічні інтерфейси шляхом перетягування та розміщення візуальних елементів, таких як кнопки, панелі, списки та прогрес-бари, на робоче полотно. Visual Studio також надає розширені можливості для роботи з кодом, включаючи автодоповнення, рефакторинг і вбудовані інструменти для тестування, що значно прискорює процес розробки [19].

Клас PerformanceCounter із .NET Framework є ключовим інструментом для моніторингу продуктивності системи. Він дозволяє отримувати дані про завантаженість центрального процесора у реальному часі, що використовується в модулі моніторингу ресурсів для відображення інформації у вигляді прогрес-бару. PerformanceCounter забезпечує високу точність і швидкість збору даних, що є критично важливим для створення ефективного інструменту моніторингу.

Аналогічно, клас ComputerInfo використовується для оцінки використання оперативної пам’яті, дозволяючи обчислювати обсяг використаної та доступної пам’яті для подальшого відображення у програмі.

Клас Process із .NET Framework відіграє важливу роль у модулі управління процесами. Він дозволяє отримувати інформацію про всі активні процеси, включаючи їхні назви, ідентифікатори та обсяг використаної пам’яті, а також виконувати дії, такі як завершення процесу або зміна пріоритету. Цей клас забезпечує гнучкість і контроль над системними процесами, що є необхідним для підвищення продуктивності системи. Для роботи з системним реєстром використовується клас Registry, який надає доступ до ключів і значень реєстру, дозволяючи переглядати, редагувати та створювати резервні копії ключів у форматі .reg за допомогою утиліти reg.exe.

Компоненти TreeView і ListView із Windows Forms використовуються для створення зручного інтерфейсу для перегляду структури реєстру та списку процесів. TreeView дозволяє відображати ієрархічну структуру ключів реєстру або груп процесів, а ListView забезпечує деталізоване відображення значень реєстру чи характеристик процесів у табличному вигляді [20].

Поєднання цих технологій дозволяє розробити програмний застосунок для моніторингу та підвищення продуктивності роботи системи. C# і .NET Framework забезпечують ефективну кодову базу для взаємодії з системними ресурсами, а Windows Forms і його компоненти, надають зручні інструменти для створення інтуїтивного інтерфейсу. Додаткові класи, такі як PerformanceCounter, Process і Registry, забезпечують точний моніторинг і управління системою, а Visual Studio спрощує процес розробки завдяки своїм зручним інструментам.

Отже, вибір технологій для розробки програмного застосунку для моніторингу та підвищення продуктивності роботи системи, що базується на C#, Windows Forms, Visual Studio і вбудованих класах .NET Framework, має вирішальне значення для створення ефективного та зручного інструменту. Поєднання цих засобів забезпечує комплексний підхід до розробки, дозволяючи створити продукт із високим рівнем функціональності, стабільності та зручності використання.

3.2 Розробка інтерфейсу програмного застосунку

При розробці інтерфейсу програмного забезпечення було приділено особливу увагу його зрозумілості та зручності для користувача.

Основним тематичним кольором програмному застосунку було обрано сучасний сірий, а також використано різні його відтінки.

При відкритті програмного застосунку, користувач побачить головне вікно програми, яке показано на рисунку 3.1.

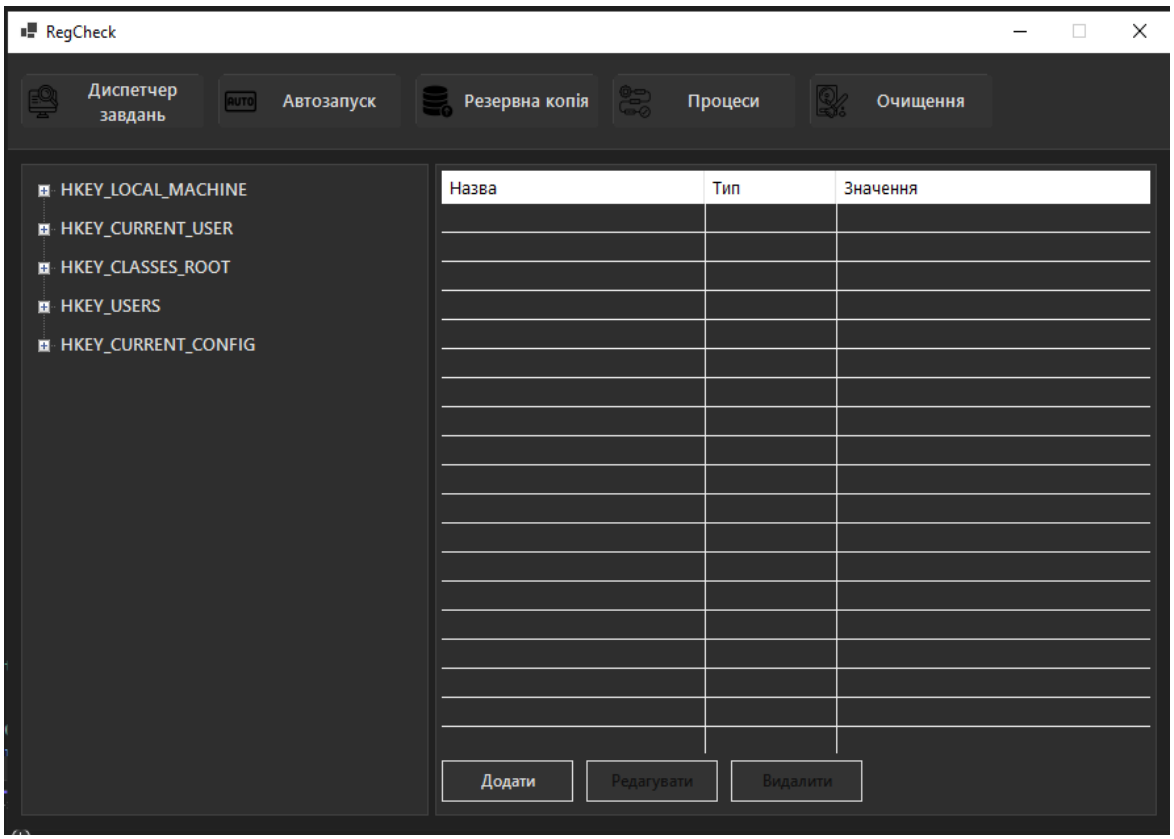


Рисунок 3.1 – Головне вікно програми

Оскільки програмний засіб передбачає роботу з різними аспектами операційної системи, було вирішено створити окремі вікна під відповідний функціонал програмного застосунку. На рисунку 3.2 показано вікно моніторингу навантаження процесора та використання оперативної пам’яті, за допомогою якого можна відстежувати стабільність показників системи.

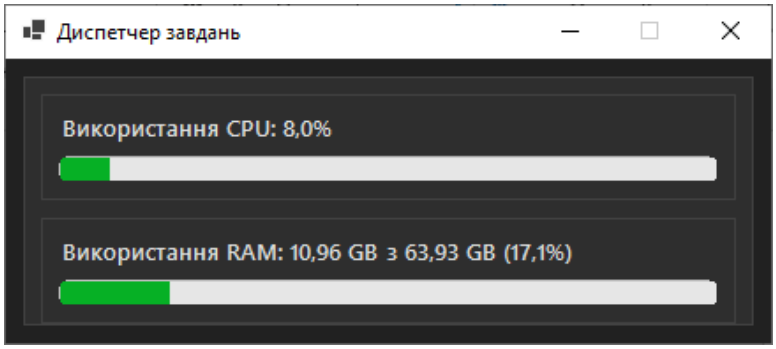


Рисунок 3.2 – Вікно диспетчера завдань

Також, на рисунку 3.3 показано вікно менеджера автозапуску, в якому можна додавати або видаляти програми з автозапуску, тобто роботи так, щоб програма запускалась при ввімкненні комп’ютера.

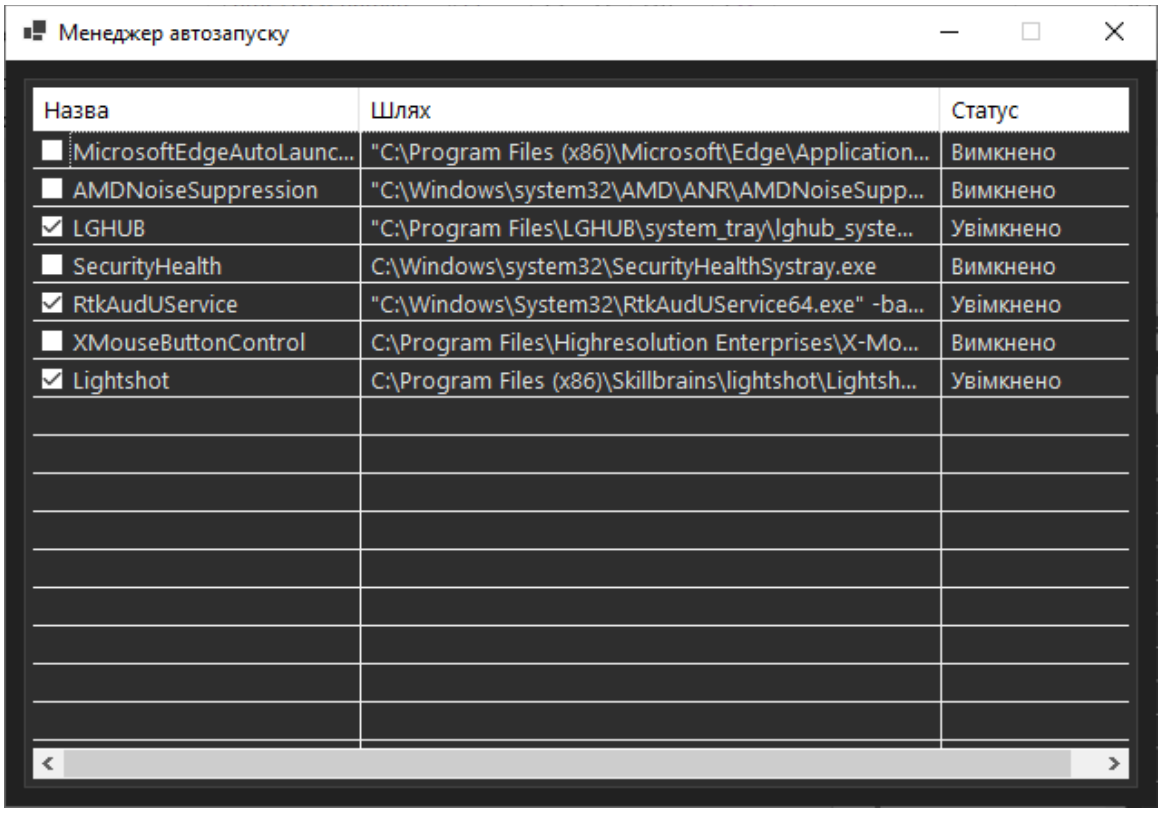


Рисунок 3.3 – Вікно менеджера автозапуску

Також, на рисунку 3.4 показано вікно менеджера процесів, за допомогою якого можна керувати процесами операційної системи, тобто завершувати процеси, задавати їм певний пріоритет.

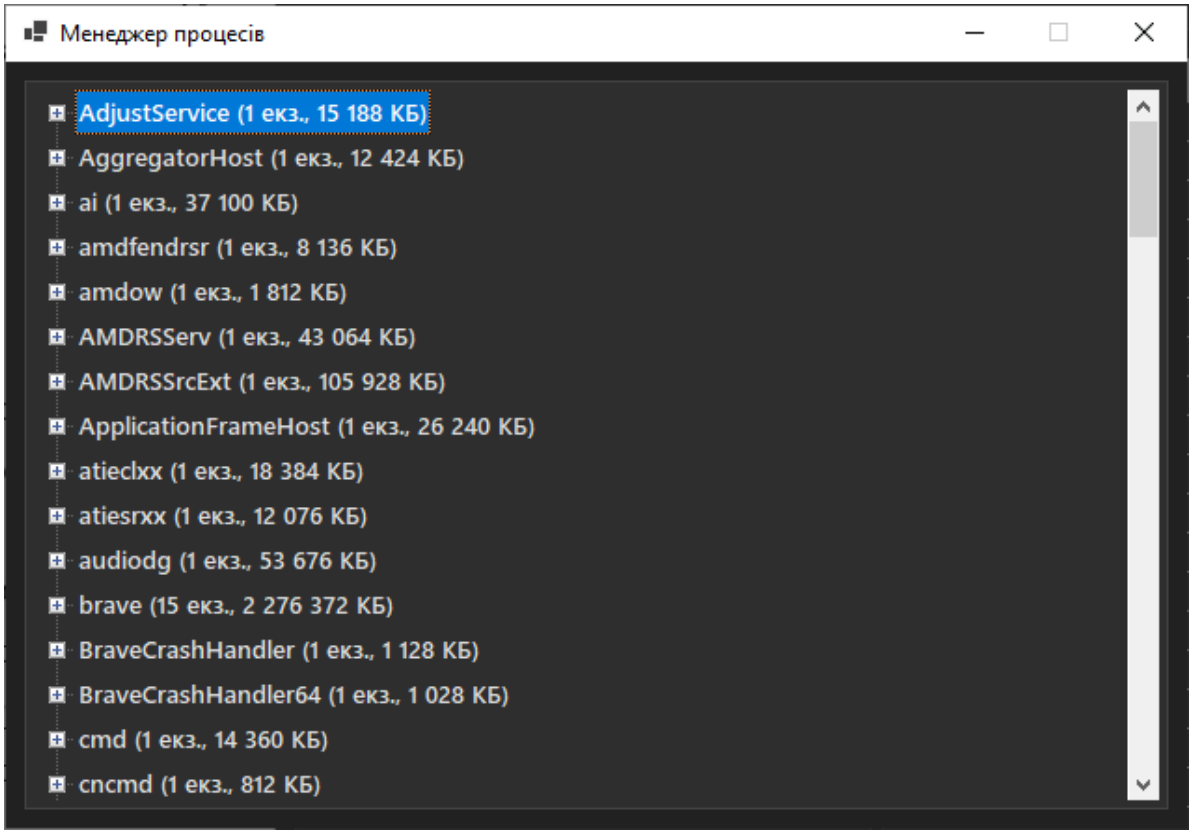


Рисунок 3.4 – Вікно візуалізації результатів вимірювання

На рисунку 3.5 показано вікно очищення, яке дозволяє очищати тимчасові файли на комп’ютері.

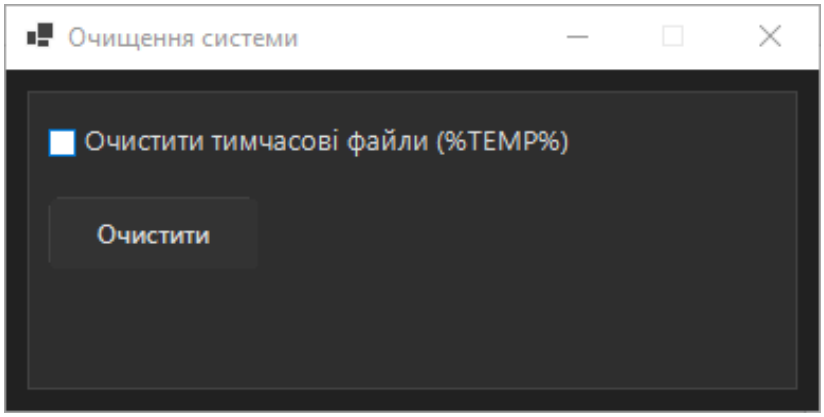


Рисунок 3.5 – Вікно очищення

Розробка графічного інтерфейсу програмного застосунку для моніторингу та підвищення продуктивності роботи системи була проведена у середовищі розробки

Visual Studio з використанням технології Windows Forms та мови програмування C#.

3.3 Розробка модуля перегляду дерева реєстру

Модуль перегляду дерева реєстру призначений для зручного перегляду та навігації по структурі системного реєстру Windows. Користувач може бачити ієрархію ключів реєстру, розгортати їх для перегляду підключів, а також переглядати значення, пов’язані з обраним ключем, у табличному вигляді. Цей модуль корисний для аналізу та редагування системного реєстру, що дозволяє користувачу оптимізувати роботу системи шляхом внесення змін до налаштувань.

Модуль складається з двох основних частин: панелі з TreeView для відображення ключів і панелі з ListView для відображення значень. При запуску програми завантажуються кореневі ключі реєстру, а при розгортанні гілки підключі реєстру завантажуються динамічно за допомогою методу LoadSubKeys.

Також, було розроблено метод RegistryTree_AfterSelect, який відповідає за заповнення ключів реєстру параметрами і їх значеннями. На рисунку 3.2 показано фрагмент коду цього методу.

Отже, розробка цього модулю була важливим етапом створення програмного застосунку для аналізу та підвищення продуктивності роботи комп’ютера оскільки за його допомогою користувач зможе детально переглядати ключі та параметри у реєстрі і в подальшому використовувати їх для оптимізації роботи системи.

3.4 Розробка модуля моніторингу за навантаженням на процесор та пам’ять

Модуль моніторингу призначений для відстеження в реальному часі завантаженості центрального процесора та оперативної пам’яті. Користувач може бачити відсоток використання процесора а також обсяг використаної та доступної оперативної пам’яті у числовому вигляді, що допомагає оцінити продуктивність системи та виявити потенційні проблеми, такі як надмірне навантаження.

Модуль оновлює дані кожну секунду за допомогою таймера. Дані про процесор отримуються через PerformanceCounter, а про RAM – через ComputerInfo [21]. Інформація відображається у вигляді прогрес-барів і текстових міток.

Розробка модуля моніторингу за навантаженням на процесор та пам’ять є важливим етапом розробки програмного застосунку, оскільки цей модуль напряму впливає на аналіз навантаженності на систему та дозволяє порівнювати наявні показники із прийнятними нормами.

3.5 Розробка модуля менеджера автозапуску

Модуль менеджера автозапуску дозволяє користувачу переглядати та редагувати програми, які автоматично запускаються під час старту системи. Це допомагає оптимізувати продуктивність, вимикаючи непотрібні програми, що споживають ресурси.

Модуль завантажує програми з ключів автозапуску, відображає їх у ListView із колонками "Назва", "Шлях" і "Статус". Користувач може вмикати або вимикати програми, змінюючи значення в реєстрі.

Отже, розробка модуля менеджеру автозапуску є важливим етапом створення програмного застосунку, оскільки функціонал увімкнення або вимкнення програми у автозапуску системи може суттєво впливати на продуктивність роботи комп’ютера.

3.6 Розробка модуля менеджера процесів

Модуль менеджера процесів дозволяє користувачу переглядати активні процеси, групувати їх за назвою, завершувати окремі процеси або цілі дерева процесів, а також змінювати пріоритет процесів для оптимізації роботи системи. Це допомагає зменшити навантаження на систему, завершуючи непотрібні процеси або підвищуючи пріоритет важливих.

Модуль групує процеси за назвою у TreeView, відображаючи кількість екземплярів і загальний обсяг використаної пам’яті. Користувач може вибрати процес або групу, відкрити контекстне меню та виконати дії, такі як завершення

процесу або зміна пріоритету. На рисунку 3.6 показано фрагмент коду завантаження всіх процесів на комп’ютері.

```
Ссылок: 3
private void LoadProcesses()
{
    processTree.Nodes.Clear();

    var processGroups = Process.GetProcesses()
        .GroupBy(p => p.ProcessName)
        .OrderBy(g => g.Key);

    foreach (var group in processGroups)
    {
        try
        {
            long totalMemory = group.Sum(p => p.WorkingSet64) / 1024;
            int processCount = group.Count();
            TreeNode groupNode = new TreeNode($"{group.Key} ({processCount} екз., {totalMemory:N0} КБ)")
            {
                ForeColor = Color.FromArgb(224, 224, 224)
            };
            groupNode.Tag = group.ToList();

            foreach (var process in group.OrderBy(p => p.Id))
            {
                try
                {
                    TreeNode processNode = new TreeNode($"ID: {process.Id}, Пам'ять: {(process.WorkingSet64 / 1024):N0} КБ")
                    {
                        ForeColor = Color.FromArgb(224, 224, 224)
                    };
                    processNode.Tag = process;
                    groupNode.Nodes.Add(processNode);
                }
                catch
                {
                    continue;
                }
            }

            processTree.Nodes.Add(groupNode);
        }
        catch
        {
            continue;
        }
    }
}
```

Рисунок 3.6 – Фрагмент коду завантаження всіх процесів на комп’ютері.

Важливою складовою цього модуля є можливість завершувати процеси або дерева процесів на комп'ютері оскільки це напряму впливає на продуктивність роботи комп'ютера, можливість завершувати дерева процесів також є корисною складовою оскільки більшість сучасних програм використовують декілька процесів у одному дереві і завершувати їх окремо досить незручно.

Отже, розробка модуля менеджера процесів є важливою складовою розробки програмного застосунку для аналізу та підвищення продуктивності роботи комп’ютера оскільки функціонал моніторингу за процесами та можливість завершувати їх істотно впливає на загальну продуктивність роботи комп’ютера.

3.7 Розробка модуля очищення

Модуль очищення дозволяє користувачу видаляти тимчасові файли з папки %TEMP%, звільняючи місце на диску та підвищуючи продуктивність системи. Користувач може вибрати опцію очищення та отримати звіт про обсяг звільненого місця [22]. Результат відображається у вигляді повідомлення. Також, на рисунку 3.7 показано фрагмент коду методу очищення тимчасових файлів на комп'ютері.

```
Ссылок: 1
private void CleanButton_Click(object sender, EventArgs e)
{
    if (!tempFilesCheckBox.Checked)
    {
        MessageBox.Show("Виберіть хоча б одну опцію для очищення.", "Попередження", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return;
    }

    if (MessageBox.Show("Ви впевнені, що хочете очистити вибрані файли? Цю дію неможливо скасувати.",
        "Підтвердження", MessageBoxButtons.YesNo, MessageBoxIcon.Warning) == DialogResult.Yes)
    {
        long deletedSize = 0;
        int deletedFiles = 0;

        if (tempFilesCheckBox.Checked)
        {
            string tempPath = Path.GetTempPath();
            try
            {
                foreach (string file in Directory.GetFiles(tempPath))
                {
                    try
                    {
                        FileInfo fileInfo = new FileInfo(file);
                        deletedSize += fileInfo.Length;
                        File.Delete(file);
                        deletedFiles++;
                    }
                    catch
                    {
                        continue;
                    }
                }

                foreach (string dir in Directory.GetDirectories(tempPath))
                {
                    try
                    {
                        Directory.Delete(dir, true);
                        deletedFiles++;
                    }
                    catch
                    {
                        continue;
                    }
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Помилка при очищенні тимчасових файлів: {ex.Message}", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }
        }

        MessageBox.Show($"Очищення завершено!\nВидалено файлів: {deletedFiles}\nЗвільнено місця: {(deletedSize / 1024f / 1024f):F2} МБ",
            "Успіх", MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
}
```

Рисунок 3.7 – Фрагмент коду очищення тимчасових файлів

Модуль перевіряє, чи вибрано опцію очищення, після чого видаляє файли та папки з %TEMP%, підраховуючи обсяг звільненого місця.

3.8 Розробка модуля резервного копіювання

Модуль створення резервних копій призначений для збереження ключів системного реєстру у вигляді файлів у форматі .reg, що дозволяє користувачу захистити важливі системні налаштування перед внесенням змін. Це особливо корисно для запобігання втраті даних у разі помилок під час редагування реєстру або збоїв системи. Користувач може обрати папку для збереження резервних копій, після чого програма автоматично створює файли для всіх основних гілок реєстру додаючи до імені файлу дату створення для зручності ідентифікації.

На рисунку 3.8 показано фрагмент коду створення резервних копій.

```
private void CreateRegistryBackups(string backupPath)
{
    string[] hiveNames = { "HKEY_CLASSES_ROOT", "HKEY_CURRENT_USER", "HKEY_LOCAL_MACHINE", "HKEY_USERS", "HKEY_CURRENT_CONFIG" };
    string dateStamp = DateTime.Now.ToString("dd_MM_yyyy");

    foreach (string hive in hiveNames)
    {
        string fileName = Path.Combine(backupPath, $"{hive}_{dateStamp}.reg");
        try
        {
            ProcessStartInfo processInfo = new ProcessStartInfo
            {
                FileName = "reg.exe",
                Arguments = $"export \"{hive}\" \"{fileName}\" /y",
                UseShellExecute = false,
                RedirectStandardOutput = true,
                RedirectStandardError = true,
                CreateNoWindow = true
            };

            using (Process process = Process.Start(processInfo))
            {
                process.WaitForExit();
                if (process.ExitCode == 0)
                {
                    MessageBox.Show($"Резервна копія для {hive} успішно створена: {fileName}", "Успіх", MessageBoxButtons.OK, MessageBoxIcon.Information);
                }
                else
                {
                    string error = process.StandardError.ReadToEnd();
                    MessageBox.Show($"Помилка при створенні резервної копії для {hive}: {error}", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Помилка при створенні резервної копії для {hive}: {ex.Message}", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
}
```

Рисунок 3.8 – Фрагмент коду створення резервних копій

Модуль викликається при натисканні кнопки "Резервна копія" на головній панелі. Після цього відкривається діалогове вікно FolderBrowserDialog, де користувач обирає папку для збереження файлів. Програма послідовно створює резервні копії для кожної основної гілки реєстру, використовуючи утиліту reg.exe через ProcessStartInfo. Для кожної гілки створюється окремий файл із назвою, що

включає ім’я гілки та поточну дату. У разі успіху користувач отримує повідомлення про створення файлу, а у разі помилки – деталізоване повідомлення про проблему.

Отже, розробка модуля створення резервних копій є важливою складовою розробки програмного застосунку для аналізу та підвищення продуктивності роботи комп’ютера оскільки даний модуль дозволяє безпечно виконувати оптимізацію системи і, якщо у разі оптимізації випливуть різні помилки та поломки, дозволяє повернутись до минулих налаштувань системи.

3.9 Висновки

Було проведено аналіз інструментів, бібліотек та розширень, які можуть допомогти у створенні програмного застосунку для аналізу та підвищення продуктивності роботи комп’ютера та обрано найбільш зручні та ефективні для розробки програмного застосунку.

Кожен модуль програми розроблено з урахуванням специфіки його функціоналу та потреб користувача. Використання C# і Windows Forms, а також вбудованих класів .NET Framework, дозволило створити ефективні інструменти для моніторингу та управління системними ресурсами а також створення резервних копій основних ключів реєстру. Сучасний дизайн забезпечує зручність використання, а модульна структура програми спрощує її підтримку та масштабування.

4 ТЕСТУВАННЯ ТА ПРАКТИЧНЕ ЗАСТОСУВАННЯ ПРОГРАМИ

4.1 Тестування системи

Тестування системи – це важливий етап у процесі розробки програмного забезпечення, який має на меті перевірити функціональність, продуктивність, стабільність і зручність використання програмного продукту. Для програмного застосунку, призначеного для моніторингу та підвищення продуктивності роботи системи, тестування дозволяє переконатися, що всі модулі працюють коректно, не викликають збоїв у системі та відповідають вимогам користувача. Тестування також допомагає виявити помилки, такі як винятки при доступі до системних ресурсів, некоректне відображення даних або проблеми з інтерфейсом, і виправити їх до випуску продукту.

Тестування системи включає кілька підходів: функціональне тестування, тобто перевірка основних функцій, нефункціональне тестування, тобто оцінка продуктивності, зручності використання, безпеки, а також тестування на різних конфігураціях обладнання та операційних систем для забезпечення сумісності. Для даного програмного застосунку особливу увагу приділено перевірці коректності роботи з системними ресурсами, оскільки помилки в таких операціях можуть призвести до серйозних проблем, наприклад, випадкового видалення важливих даних у реєстрі або завершення критичних системних процесів [23].

Під час підготовки до тестування програмного застосунку для аналізу та підвищення продуктивності роботи комп’ютера було виділено такі етапи функціонального тестування [24]:

1. Модуль перегляду дерева реєстру: перевірити, чи коректно відображаються кореневі ключі, чи завантажуються підключі при розгортанні вузла, чи відображаються значення у ListView при виборі ключа.

2. Модуль моніторингу за навантаженням: перевірити точність відображення завантаженості центрального процесора та оперативної пам’яті, чи оновлюються дані кожну секунду, чи коректно відображаються прогрес-бари.

3. Модуль менеджера автозапуску: перевірити, чи відображаються всі програми автозапуску, чи змінюється їхній статус при перемиканні CheckBox, чи зберігаються зміни в реєстрі.

4. Модуль менеджера процесів: перевірити відображення списку процесів, групування за назвою, завершення процесу та зміну пріоритету через контекстне меню, а також оновлення списку після завершення процесу.

5. Модуль очищення: перевірити видалення файлів із %TEMP%, підрахунок звільненого місця та відображення результату.

6. Модуль резервного копіювання: перевірити створення резервних копій ключів реєстру у форматі .reg, а також обробку помилок, наприклад, якщо диск переповнений.

Також, під час підготовки до тестування програмного застосунку для аналізу та підвищення продуктивності роботи комп'ютера було виділено такі етапи нефункціонального тестування [25]:

1. Продуктивність: оцінити, як програма впливає на системні ресурси. Наприклад, перевірити, чи не викликає модуль моніторингу надмірного навантаження на центральний процесор під час оновлення даних, чи швидко завантажуються підключі реєстру у TreeView.

2. Зручність використання: перевірити, чи є інтерфейс інтуїтивно зрозумілим, чи зрозуміло відображаються дані у TreeView і ListView, чи достатньо інформативні повідомлення про помилки.

3. Безпека: перевірити, як програма обробляє помилки доступу до системних ресурсів. Наприклад, якщо користувач запускає програму без прав адміністратора, чи видається відповідне повідомлення, чи не викликає програма збоїв при спробі завершити критичний системний процес.

Також, окремими етапи тестування можуть бути:

1. Тестування на сумісність, тобто перевірка роботи програми на різних версіях Windows а також різних розрядностях.

2. Обробка винятків, тобто перевірити, як програма реагує на винятки. Наприклад, якщо доступ до реєстру заборонений, чи відображається повідомлення

про помилку. Перевірити, що станеться, якщо папка %TEMP% порожня або якщо користувач скасовує очищення.

3. Регресійне тестування, тобто після виправлення помилок провести повторне тестування, щоб переконатися, що зміни не вплинули на роботу інших модулів. Наприклад, якщо виправлено помилку в модулі очищення, перевірити, чи не вплинуло це на модуль моніторингу.

Під час користування програмним застосунком, користувач обов'язково повинен запустити програму від імені адміністратора, інакше функціонал програми буде достатньо обмеженим а також користувач, під час роботи у програмі буде отримувати повідомлення про недостатні права. На рисунку 4.1 показано приклад помилки, коли у програми недостатньо прав для виконання певної функції.

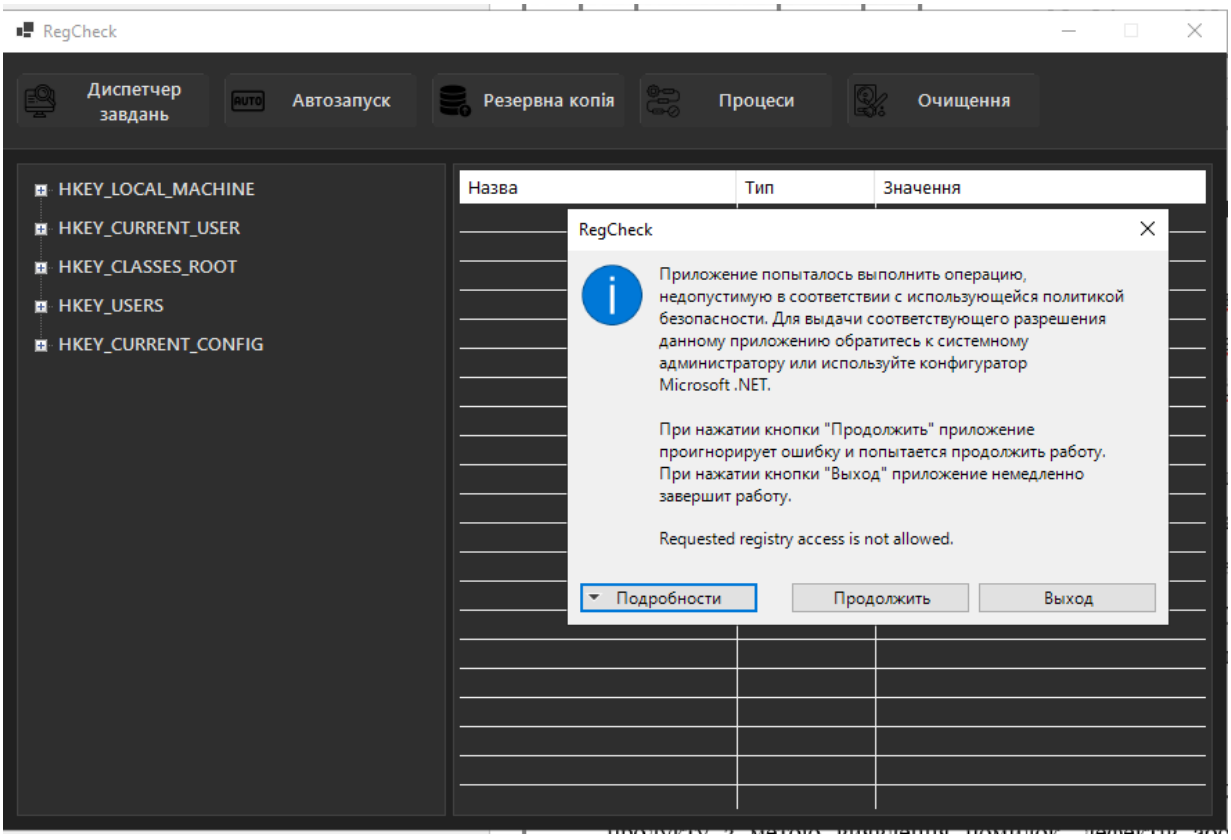


Рисунок 4.1 – Інформаційне вікно про нестачу прав

Користувач може подивитись детальніше про цю помилку, продовжити роботу без надання прав а також вийти з програми.

На рисунку 4.2 показано помилку при редагуванні параметру у ключі реєстру без відповідних прав.

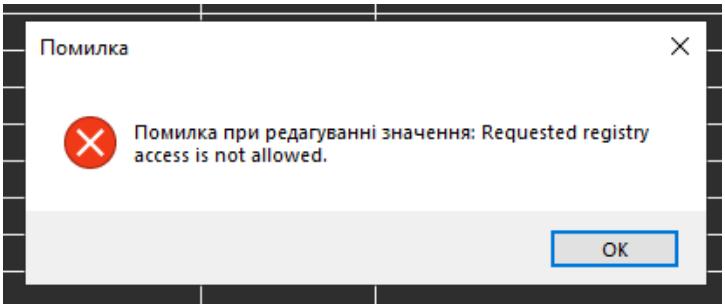


Рисунок 4.2 – Помилка після редагування параметру без необхідних прав

У разі продовження роботи програми без надання їй прав адміністратора, програма може працювати некоректно, обмежить користувача від певного функціоналу або ж повторно буде попереджувати користувача про відсутність прав доступу.

Також, при спробі створити новий параметр, не обравши ключ реєстру, програмний застосунок видасть відповідне повідомлення. На рисунку 4.3 показано приклад помилки.

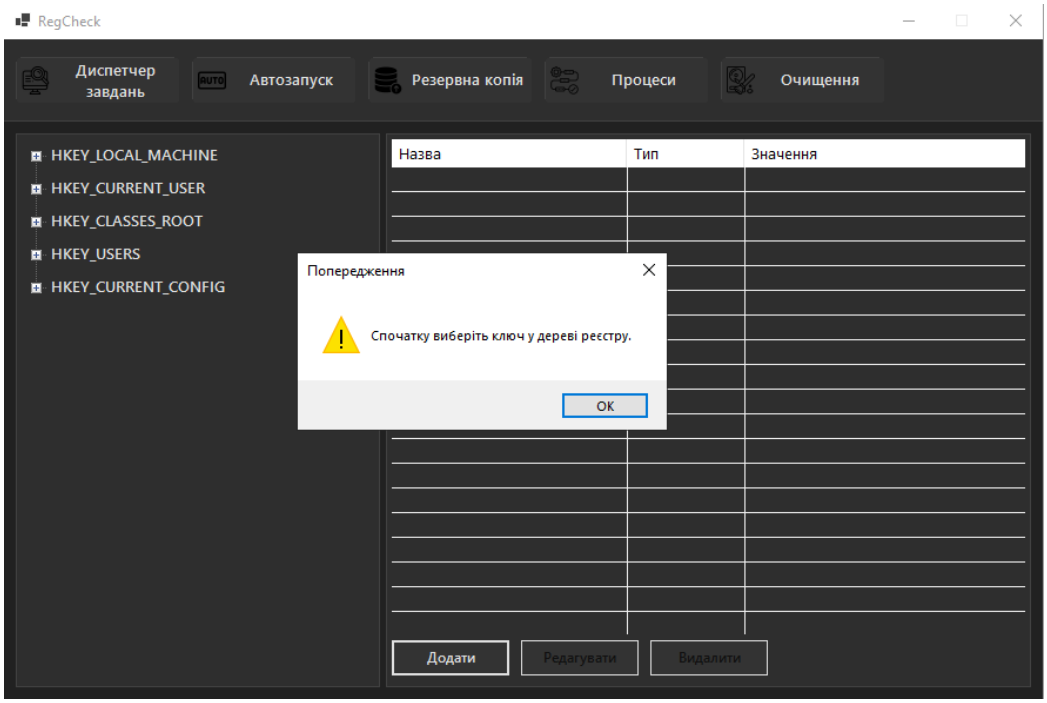


Рисунок 4.3 – Помилка при доданні нового параметру

4.2 Розробка інструкції користувача

Інструкція користувача передбачає визначення технічних вимог для запуску програмного продукту. Деталі конфігурацій показано у таблиці 4.1.

Таблиця 4.1 – Вимоги до апаратного забезпечення

Вимоги до конфігурування	Рекомендовано	Мінімально
Процесор	Intel Core i5	Intel Core i3
Оперативна пам'ять	8 ГБ RAM	4 ГБ RAM
Вільний простір на диску	10 ГБ	5 ГБ
Відеокарта	NVIDIA GeForce GTX 1070	NVIDIA GeForce GTX 1050
Операційна система	Windows 10	Windows 10
Монітор	Роздільна здатність не менше 1920x1080 пікселів, діагональ 21" або більше	Роздільна здатність не менше 1280x720 пікселів, діагональ 21" або більше

Для того, щоб розпочати роботу із програмним застосунком, користувачеві потрібно впевнитись у тому, що набір комплектуючих персонального комп’ютера користувача задовольняє мінімальним вимогам до програмного забезпечення.

Після завантаження програмного застосунку, користувачеві буде представлене головне вікно програми, яке містить у собі навігаційну панель з кнопками, які потрібні для переходу між вікнами програмного застосунку, також показано відображення ключів реєстру у вигляді дерева, а також список, який буде відображати інформацію про параметри у ключах реєстру при натисненні на них, а також можливість редагування, видалення та додавання параметрів у обраних ключах в реєстрі.

На рисунку 4.4 показано головне вікно розроблюваного програмного застосунку.

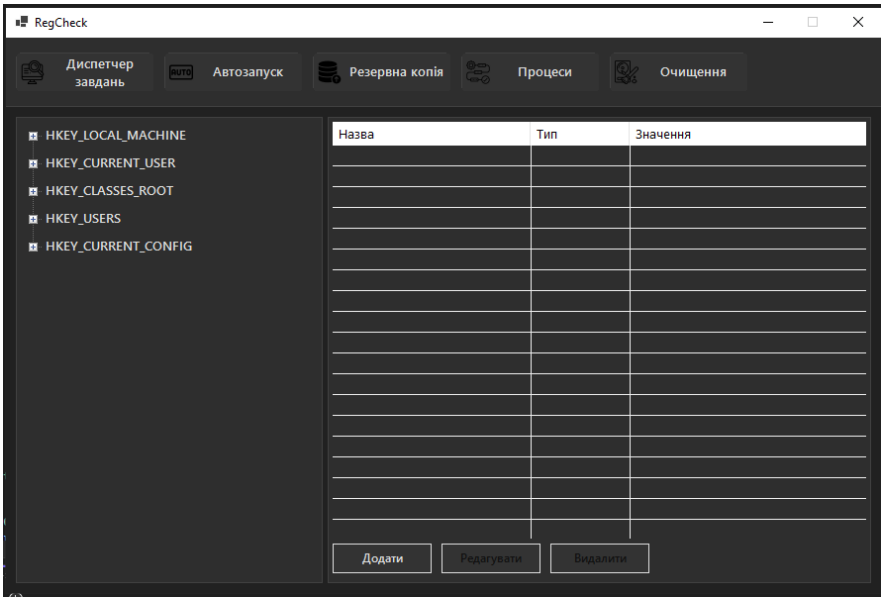


Рисунок 4.4 – Головне вікно застосунку

У головному вікні програмного застосунку користувач може переглядати ключі та підключі дерева реєстру, їх параметри і значення, а також додавати нові параметри, редагувати або видаляти вже існуючі. На рисунку 4.5 показано відкритий розділ реєстру з параметрами та значеннями та меню редагування параметру реєстру.

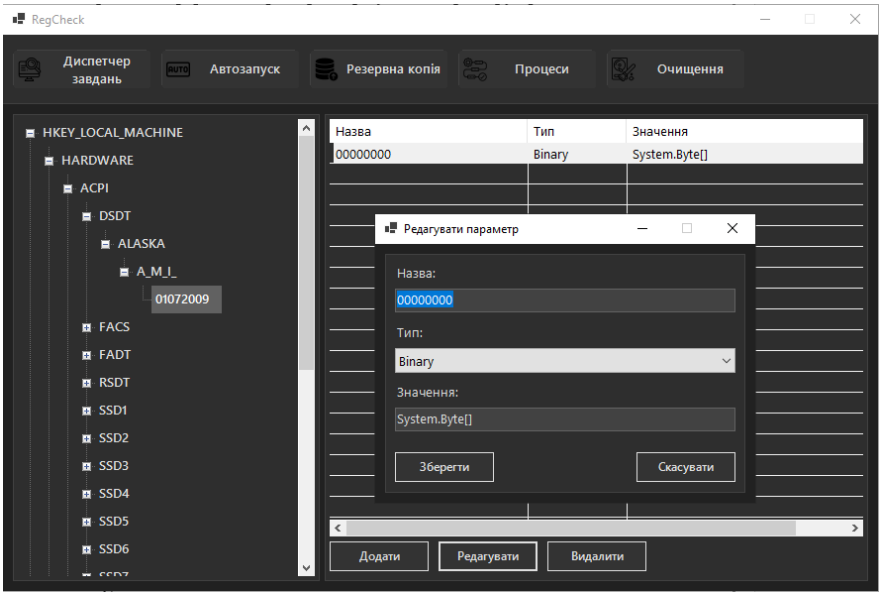


Рисунок 4.5 – Відкритий розділу реєстру та меню редагування параметру

Також, на рисунку 4.6 показано меню додавання нового параметру у реєстрі.

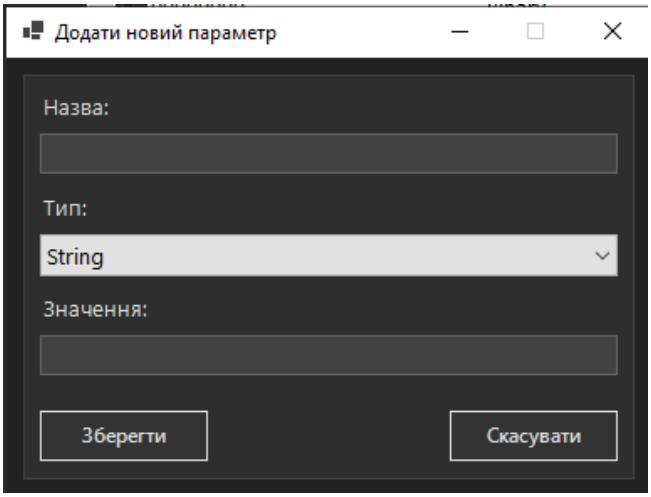


Рисунок 4.6 – Меню додавання нового параметру у реєстрі

Функції створення нового параметру, редагування або видалення вже існуючого потребують надання від користувача програмі прав адміністратора, інакше програмний застосунок видасть помилку, яку показано на рисунку 4.7.

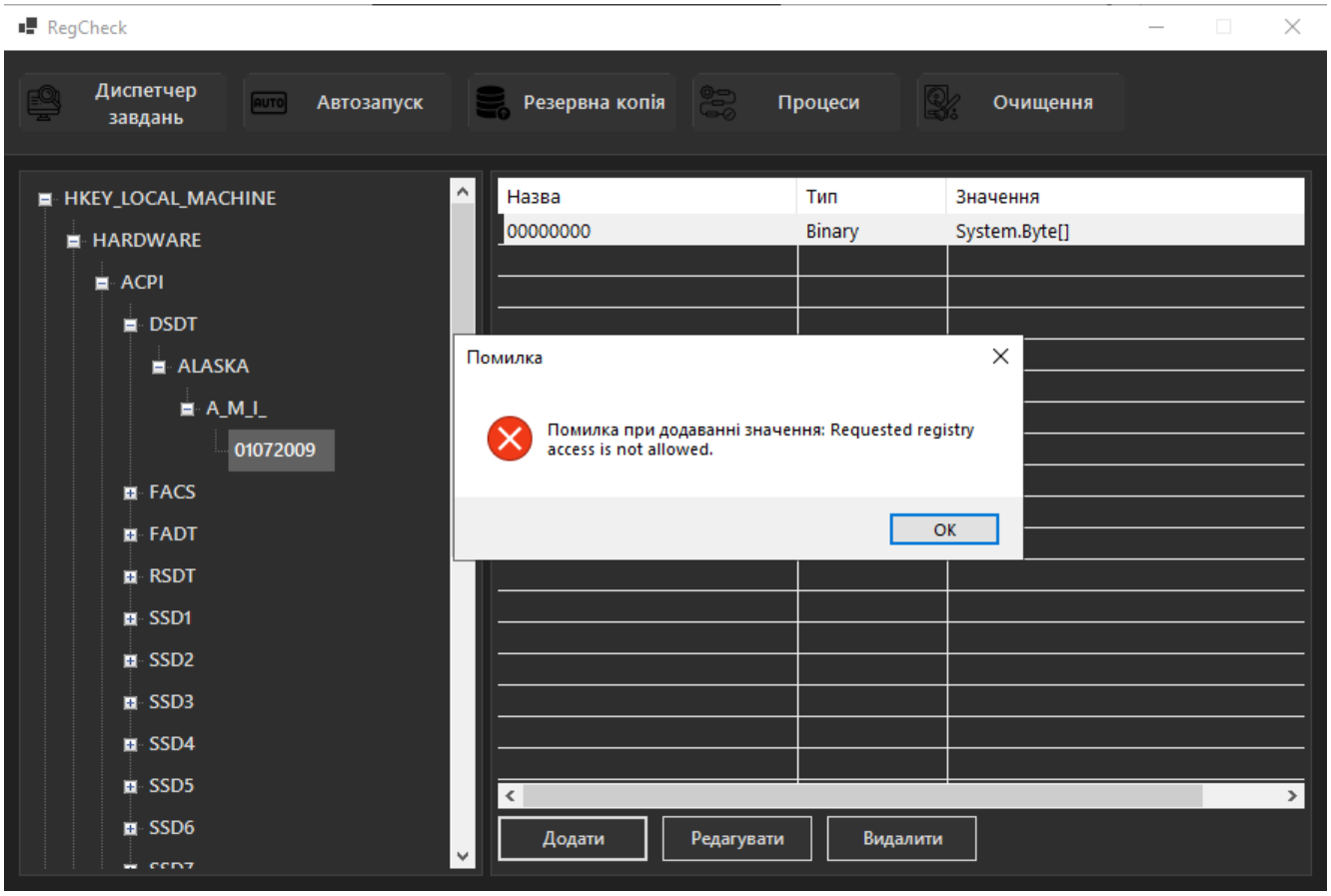


Рисунок 4.7 – Помилка про відсутність необхідних прав

У верхній частині програмного застосунку показано навігаційну панель, на якій розміщено всі потрібні користувачеві кнопки. Навігаційну панель показано на рисунку 4.8.

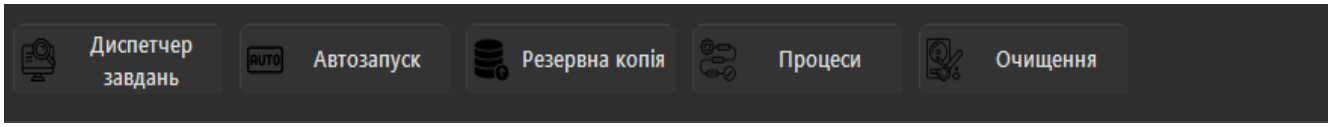


Рисунок 4.8 – Навігаційна панель

При натисненні, на кнопку «Диспетчер завдань», користувач відкриє нове вікно програми, яке, після завантаження, покаже прогрес-бари та числові значення навантаження на процесор та оперативну пам’ять. На рисунку 4.9 показано вікно «Диспетчер завдань».

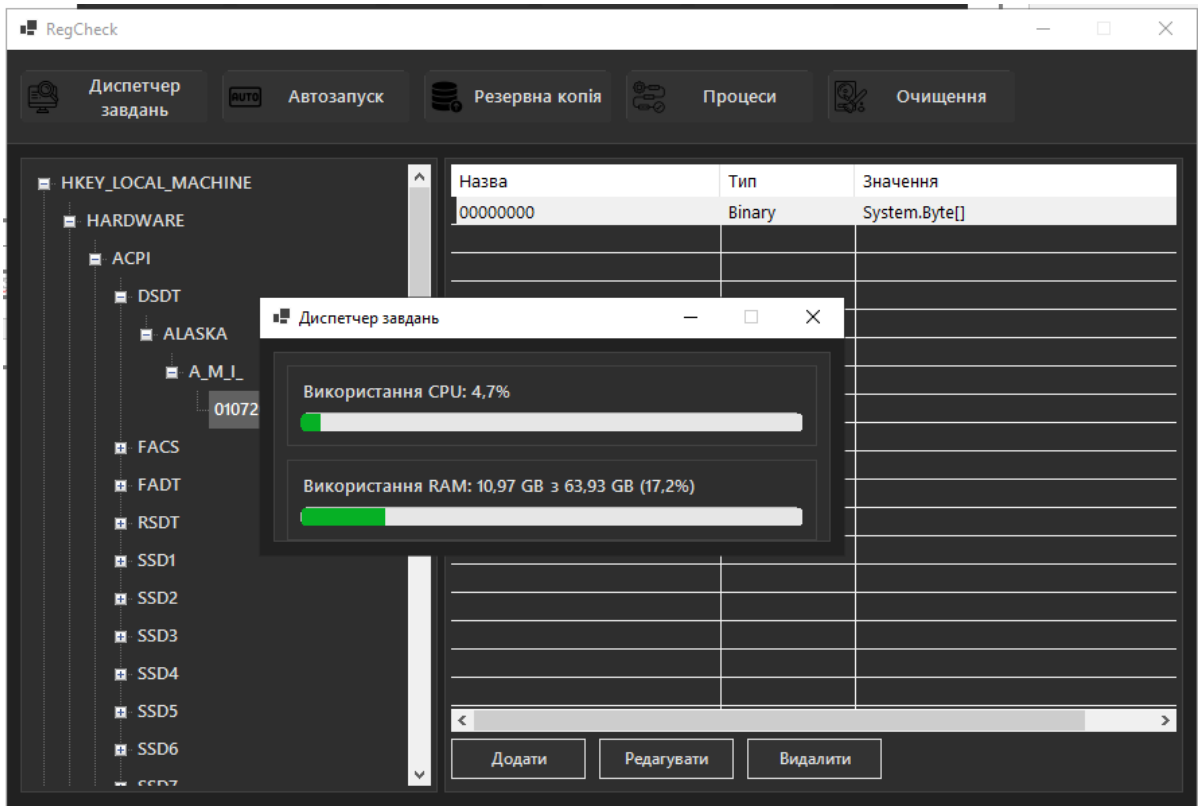


Рисунок 4.9 – Вікно «Диспетчер завдань»

Дане вікно відповідає за моніторинг за навантаженням на процесор та оперативну пам’ять, користувачеві слід перед початком оптимізаційних процесів

обов’язково порівняти наявні показники у цьому вікні із безпечними, і лише потім розпочинати оптимізацію системи.

При натисненні користувачем на кнопку «Автозапуск», буде відкрито нове вікно, в якому можна додавати або видаляти програми із автозапуску у системі. На рисунку 4.10 показано вікно «Автозапуск».

У вікні «Автозапуск» користувач може керувати програмами у автозапуску, залишаючи лише потрібні користувачеві програми, оскільки чим більше програм буде у автозапуску, тим довше буде запускатись комп’ютер та тим більше процесів буде використовуватись, тому відключення непотрібних програм з автозапуску сприяє підвищенню продуктивності роботи комп’ютера. Також, відкриття та змінення параметрів у вікні «Автозапуск» потребує прав адміністратора, тому, якщо програмний застосунок буде відкрито зі звичайними правами, користувач не отримає доступу до повного функціоналу у цьому вікні.

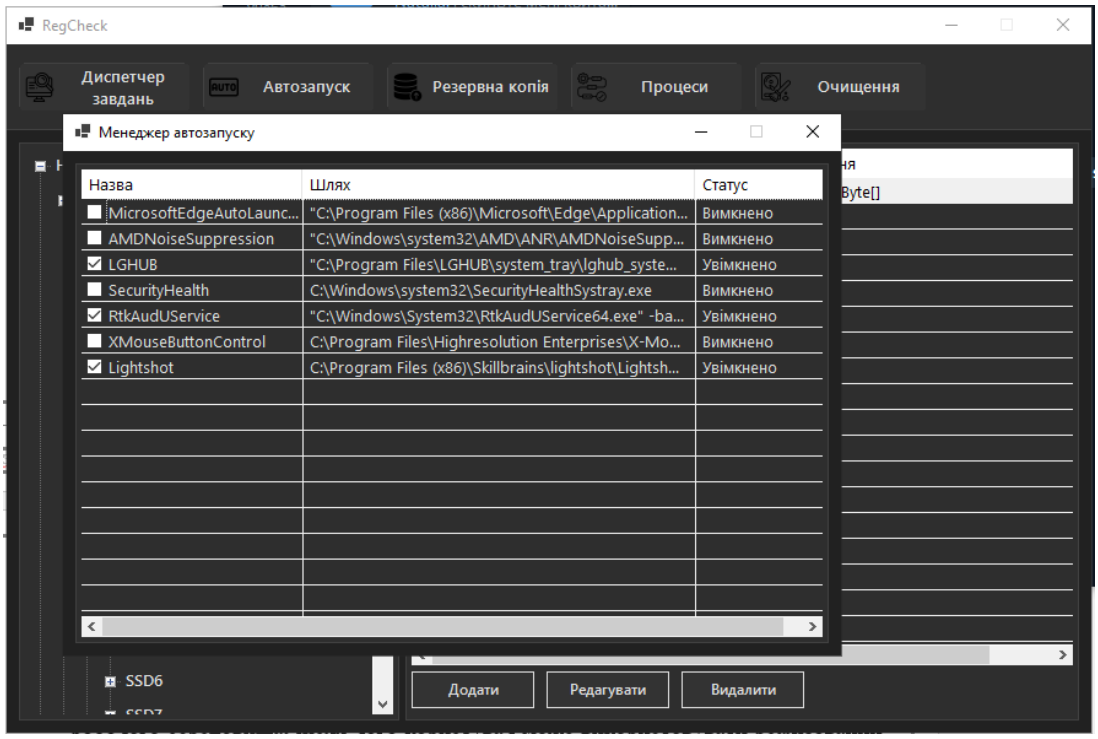


Рисунок 4.10 – Вікно «Автозапуск»

При натисненні на кнопку «Резервна копія», програма відкриє вікно із вибором шляху для папки, у якій буде розміщено резервні копії головних ключів

реєстру, назви яких містять назву ключа а також дату створення, для зручного користування резервними копіями.

На рисунку 4.11 показано відкрите вікно вибору папки для створення резервних копій.

Рекомендовано, перед початком будь-яких оптимізаційних процесів за допомогою програмного застосунку, зробити резервну копію реєстру, щоб у разі виникнення помилок, була можливість повернутися до попередньої конфігурації системи.

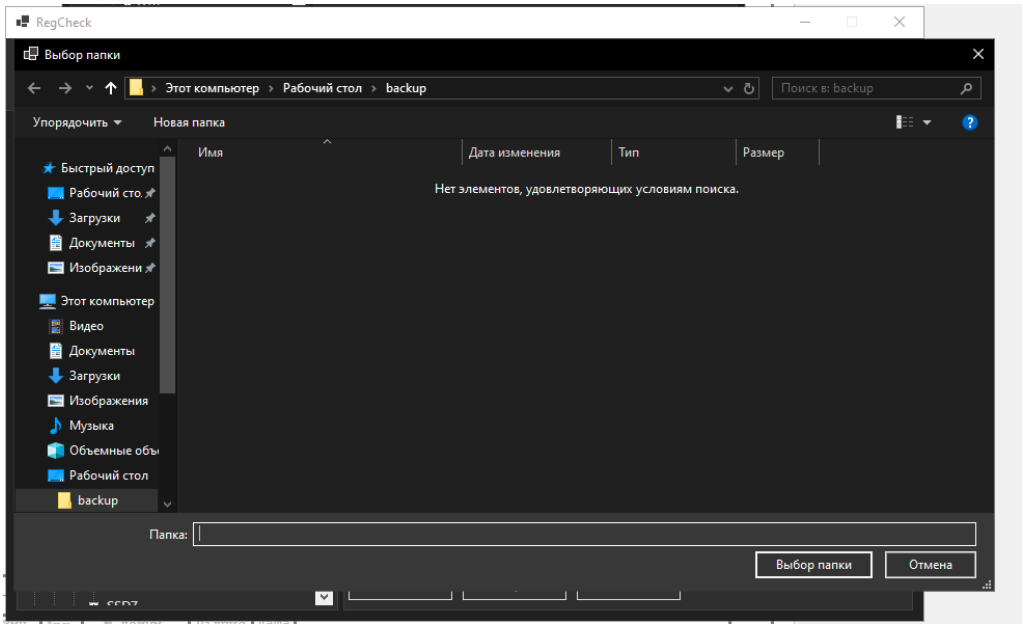


Рисунок 4.11 – Вікно вибору папку для створення резервних копій

На рисунку 4.12 показано зовнішній вигляд створених файли резервної копії ключів реєстру.

Имя	Дата изменения	Тип	Размер
HKEY_CLASSES_ROOT_24_03_2025.reg	24.03.2025 04:56	Файл реестра	62 195 КБ
HKEY_CURRENT_CONFIG_24_03_2025.reg	24.03.2025 04:56	Файл реестра	2 КБ
HKEY_CURRENT_USER_24_03_2025.reg	24.03.2025 04:56	Файл реестра	15 419 КБ
HKEY_LOCAL_MACHINE_24_03_2025.reg	24.03.2025 04:56	Файл реестра	309 010 КБ
HKEY_USERS_24_03_2025.reg	24.03.2025 04:56	Файл реестра	19 077 КБ

Рисунок 4.12 – Створені файли резервної копії

При натисненні на кнопку «Процеси», відкривається вікно менеджера процесів, у якому можна відстежувати всі запущені процеси на комп’ютері, кількість задіяної для певних процесів пам’яті, задавати пріоритет певним процесам, а також завершувати як окремі процеси, так і цілі дерева процесів. На рисунку 4.13 показано вікно «Процеси».

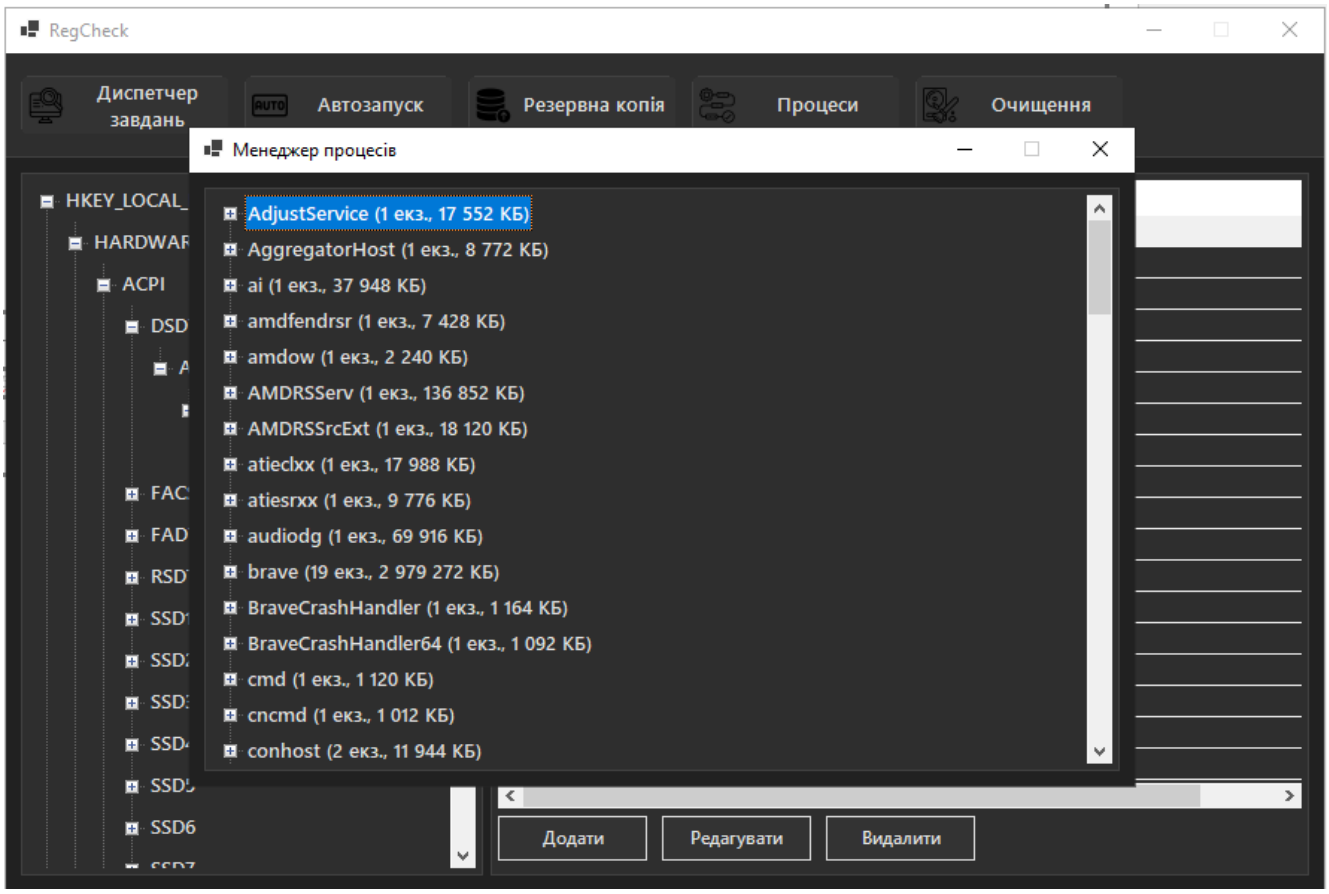


Рисунок 4.13 – Вікно «Процеси»

При натисненні правою кнопкою миші на процес або дерево процесів, відкривається контекстне меню, у якому можна вибрати опції завершення процесу, завершення дерева процесів або задання пріоритету. Додатково, при наведенні на опцію «Встановити пріоритет», відкривається ще одне контексте меню, у якому можна обрати який саме пріоритет потрібно встановити для певного процесу чи дерева процесів.

На рисунку 4.14 показано контекстні меню у вікні «Процеси», при натисненні правої кнопки миші.

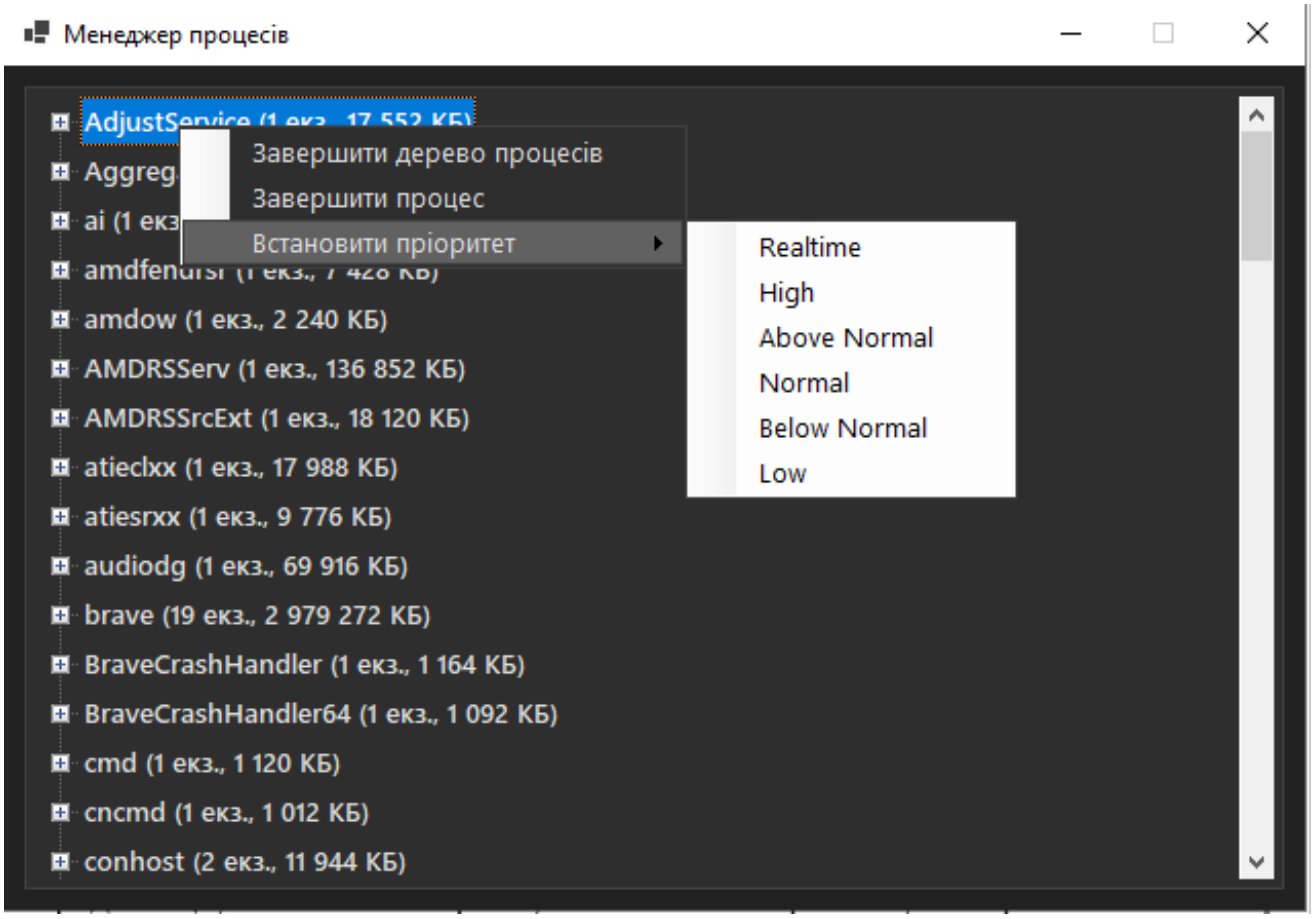


Рисунок 4.14 – Контекстні меню

При натисненні на кнопку «Очищення», відкривається вікно, у якому користувач може очистити тимчасові файли з обраних папок. На рисунку 4.15 показано вікно «Очищення».

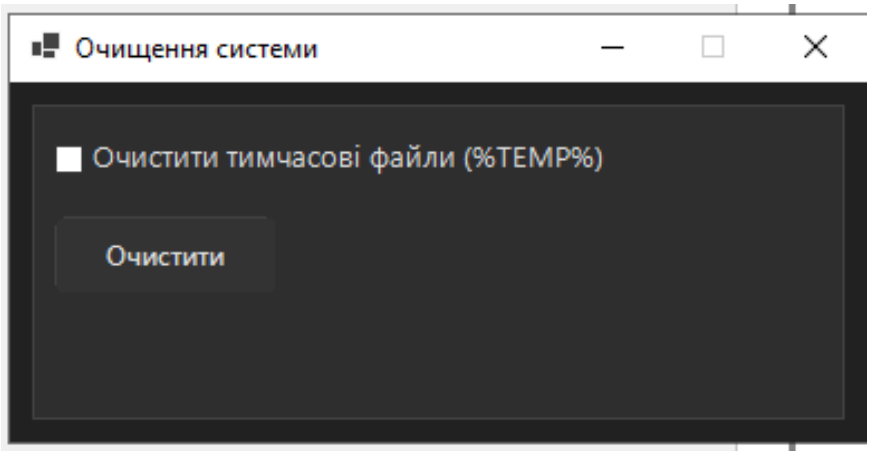


Рисунок 4.15 – Вікно «Очищення»

При натисненні на копку «Очистити», програма розпочне очистку тимчасових файлів, після чого користувач отримає повідомлення звіт про очищенні файли та звільнене місце.

На рисунку 4.16 показано результат очистки.

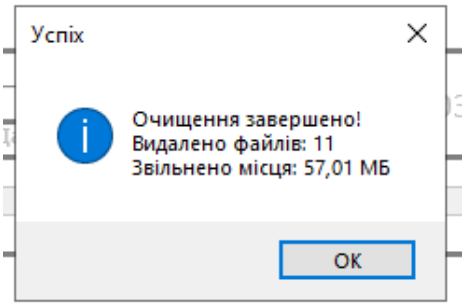


Рисунок 4.16 – Результат очистки

Також, на рисунку 4.17 можна побачити результат очистки у папці %TEMP%, та при можливості видалити залишкові файли.

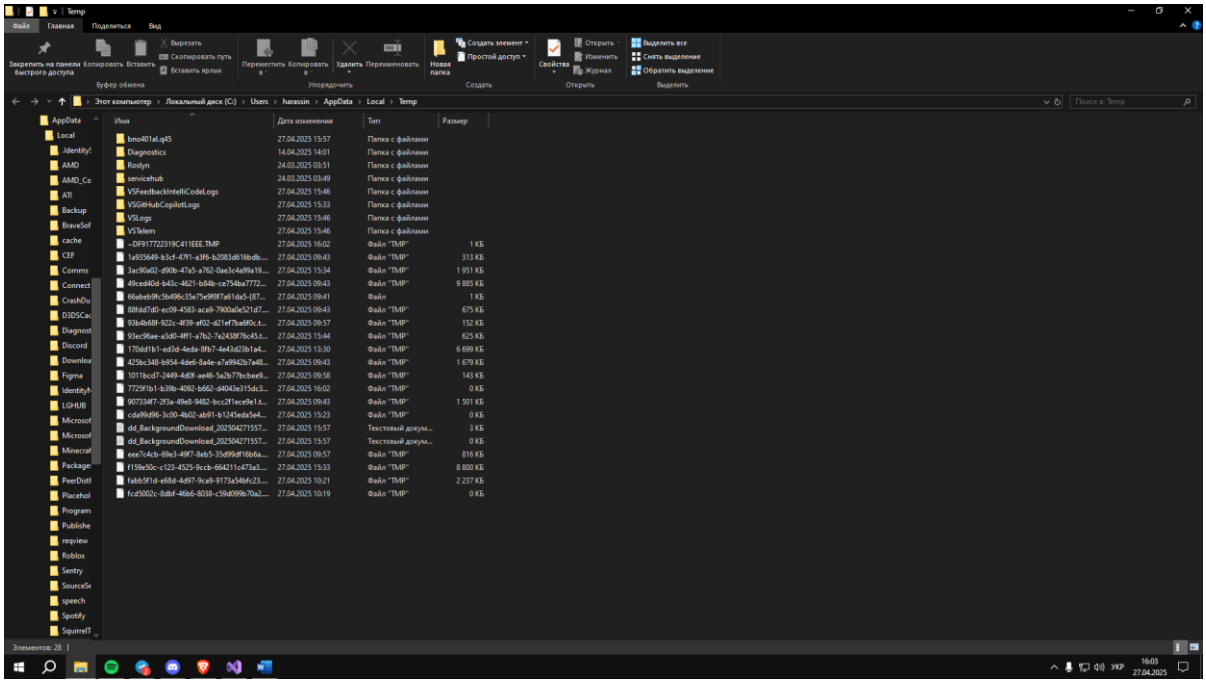


Рисунок 4.17 – Результат очистки папки %TEMP%

Очищення тимчасових ресурсів сприяє звільненню дискового простору від сміття і підвищує продуктивність роботи комп’ютера.

4.3 Практичне застосування програмного застосунку

Програмний застосунок є комплексним інструментом для адміністрування Windows-систем, який поєднує функції редагування системного реєстру, управління процесами, автозавантаженням, моніторингу системних ресурсів, створення резервних копій і очищення тимчасових файлів. Завдяки модульній структурі, він підходить для широкого кола користувачів – від системних адміністраторів і IT-фахівців до досвідчених користувачів, які прагнуть оптимізувати роботу своїх комп’ютерів.

Програмний застосунок допомагає користувачам ефективно керувати ключовими аспектами роботи Windows, надаючи зручний інтерфейс для виконання складних системних завдань. Програма дозволяє тонко налаштовувати систему, діагностувати проблеми, підвищувати продуктивніст. Вона доступна навіть для тих, хто не є професійним адміністратором, але хоче покращити роботу свого комп’ютера.

Основні сценарії використання:

1. Налаштування та оптимізація. Користувачі можуть змінювати параметри реєстру для прискорення роботи системи, вимкнення непотрібних функцій чи персоналізації інтерфейсу, а також вимикати програми з автозавантаження, щоб скоротити час завантаження.

2. Діагностика та усунення проблем. Програма допомагає виявляти проблемні процеси, які перевантажують центральний процесор чи пам’ять, виправляти помилки в реєстрі і видаляти підозрілі записи автозавантаження, що можуть бути пов’язані зі шкідливим програмним забезпеченням.

3. Звільнення ресурсів. Очищення тимчасових файлів із директорії %TEMP% звільняє місце на диску, що особливо корисно для комп’ютерів із обмеженим обсягом пам’яті, а завершення ресурсомістких процесів підвищує швидкодію.

4. Захист і відновлення. Створення резервних копій реєстру дозволяє безпечно експериментувати з налаштуваннями або відновлювати систему після збоїв, що важливо для корпоративних чи критичних середовищ.

Для більш наочного огляду практичного застосування програмного застосунку, було протестовано його роботу, шляхом заміру основних показників швидкодії комп’ютера до та після використання програмного застосунку.

Для тестів, було використано досить популярне програмне забезпечення AIDA64, що являє собою універсальну утиліту для тестування та ідентифікації компонентів персонального комп'ютера на операційних системах Windows, що надає детальні відомості про апаратне та програмне забезпечення.

Програма аналізує конфігурацію комп'ютера та видає докладну інформацію про встановлені у системі пристрої, їх характеристики, конфігурацію програмного забезпечення та багато чого іншого.

Для тестування програмного застосунку було проведено ряд тестів:

1. Перевірено затримки пам’яті. Цей тест показує середній час зчитування процесором даних із оперативної пам’яті.

2. Перевірено копіювання в пам’яті. Цей тест показує швидкість пересилання даних з одних осередків пам’яті до інших через кеш процесора.

3. Перевірено читання з пам’яті. Цей тест показує швидкість пересилання даних з оперативної пам’яті до процесора.

4. Порівняно температурні режими. Цей тест показує поточні температури окремих комплектуючих персонального комп’ютера.

5. Порівняно вільний дисковий простір. Цей тест показує поточні значення заповнення дискового простору.

Для початку тестування, потрібно завантажити та відкрити програмне забезпечення AIDA64, після цього потрібно перейти у вкладку «Тести» та обрати потрібний вид тесту.

Для початку, було перевірено затримки пам’яті. Модуль розроблюваного програмного застосунку, який впливає на цей тест – менеджер процесів, менеджер автозапуску, а також вміле налаштування конфігурації цих модулів.

На рисунку 4.18 показано результат тесту затримки пам’яті до запуску модулів.

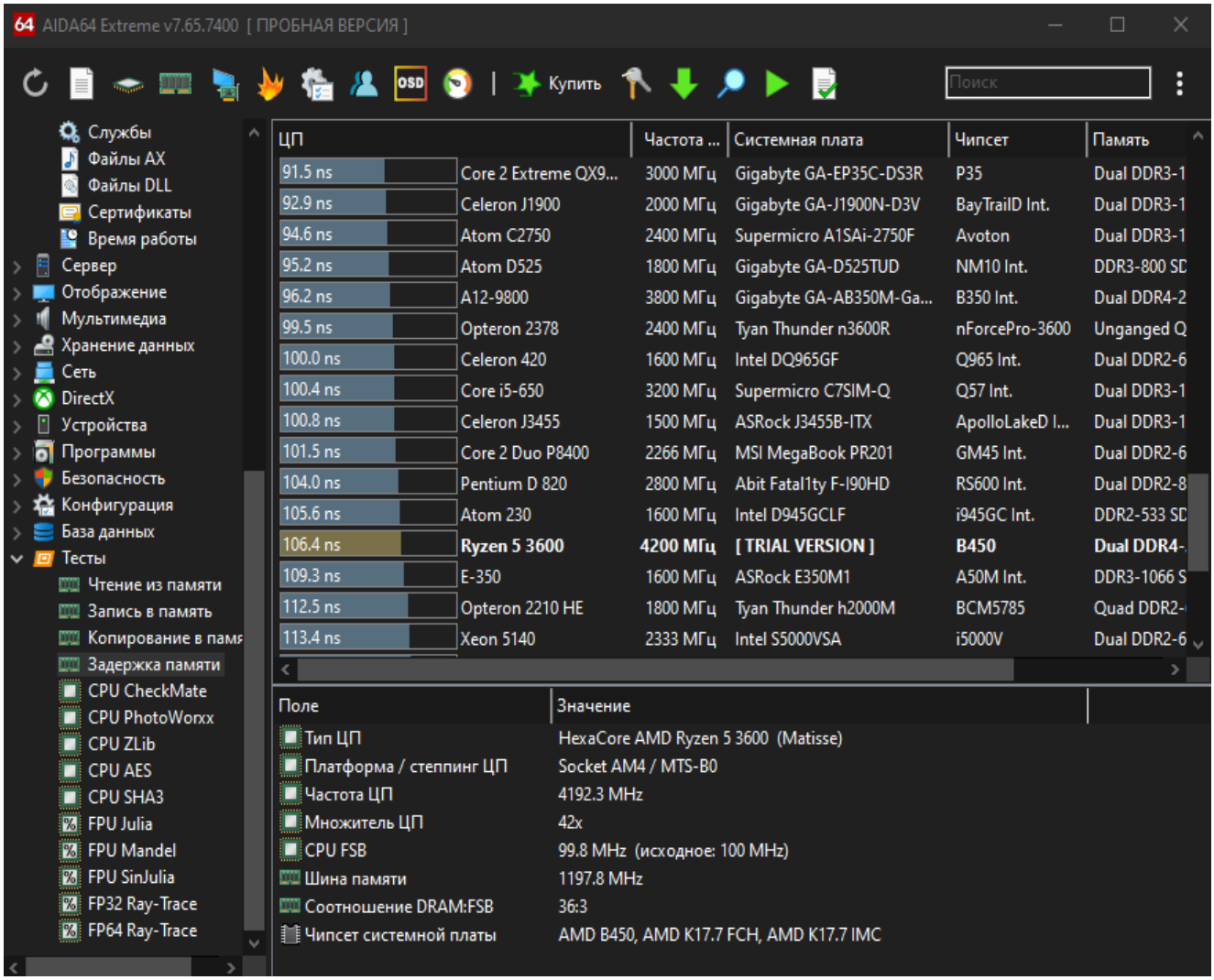


Рисунок 4.18 – Результат тестування затримки пам’яті до запуску модулів

Результат тестування показано у наносекундах, тобто середній час зчитування процесором даних із оперативної пам’яті до запуску потрібних модулів у розроблюваного програмного застосунку складав 106,4 наносекунд.

Після запуску модулів менеджера процесів, автозапуску та налаштування параметрів у них та проведення повторного тестування затримок пам’яті, слід порівняти два значення до та після та знайти відсоток, на скільки затримка зменшилась чи збільшилась.

На рисунку 4.19 показано середній час зчитування процесором даних із оперативної пам’яті після запуску модулів менеджера процесів, автозапуску та налаштування параметрів у них.

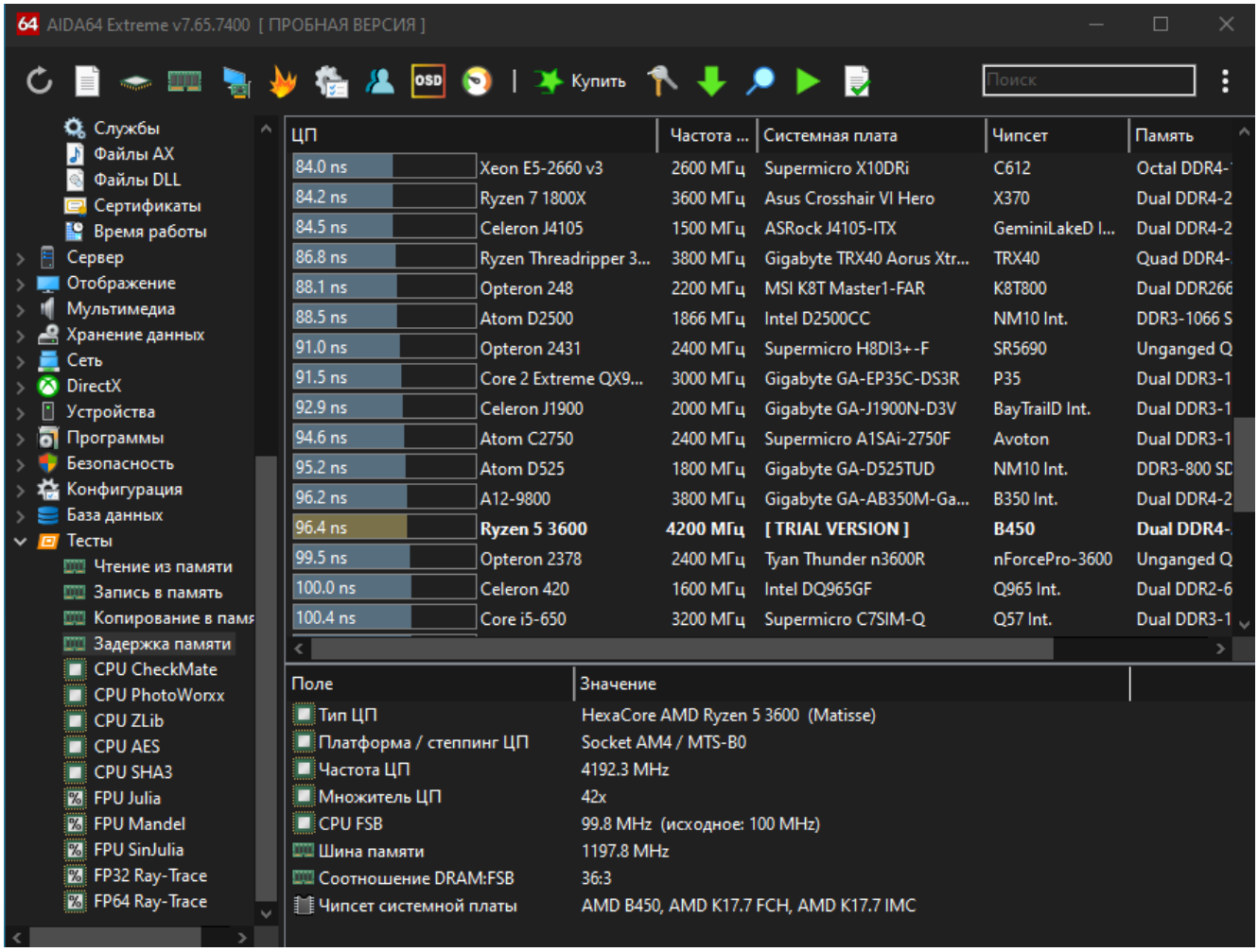


Рисунок 4.19 – Результат тестування затримки пам’яті після запуску модулів

Після налаштування у потрібних модулях програми, результат складає 96,4 наносекунд. Отже, порівнявши ці два значення можна зробити висновок, що середній час зчитування процесором даних із оперативної пам’яті зменшився на 9,4%, що є дуже хорошим результатом.

Другим було перевірено копіювання в пам’яті. Цей тест показує швидкість пересилання даних з одних осередків пам’яті до інших через кеш процесора. Модуль розроблюваного програмного застосунку, який повинен покращити результати цих показників – очищення дискового простору, менеджер процесів, а також редактор реєстру.

На рисунку 4.20 показано результат тестування швидкості пересилання даних з одних осередків пам’яті до інших через кеш процесора до запуску відповідних модулів у розроблюваному програмному забезпеченні.

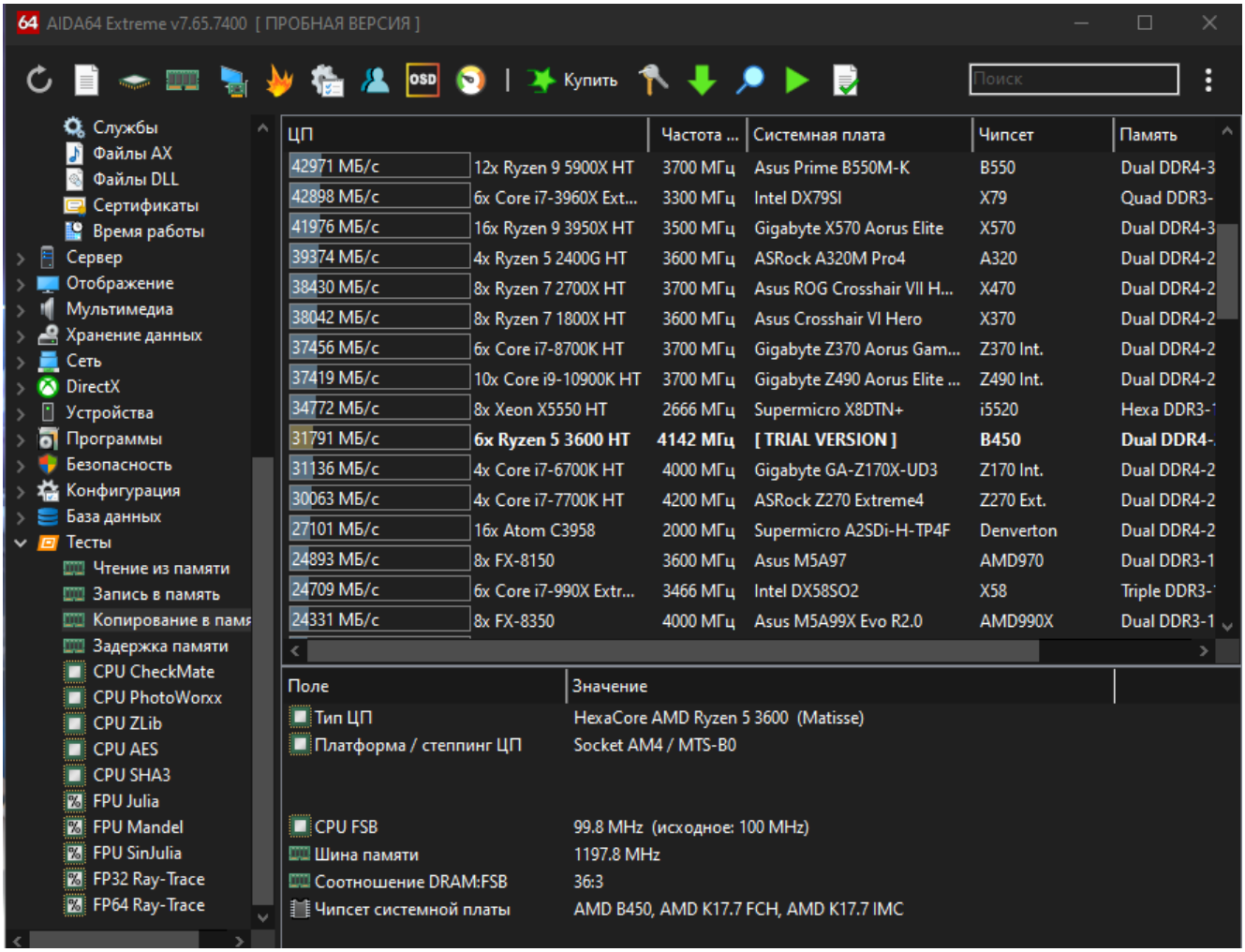


Рисунок 4.20 – Результат тестування копіювання в пам’яті до запуску модулів

Результат тестування показано у мегабайтах за секунду, тобто швидкість пересилання даних з одних осередків пам’яті до інших через кеш процесора до запуску відповідних модулів у розроблюваному програмному забезпеченні становить 31791 мегабайти за секунду.

Після запуску модулів очищення дискового простору, менеджера процесів, редактору реєстру та налаштування параметрів у них та проведення повторного тестування копіювання в пам’яті, слід порівняти два значення до та після та знайти відсоток, на скільки швидкість зменшилась чи збільшилась.

На рисунку 4.21 показано результат тестування швидкості пересилання даних з одних осередків пам’яті до інших через кеш процесора після запуску відповідних модулів у розроблюваному програмному забезпеченні.

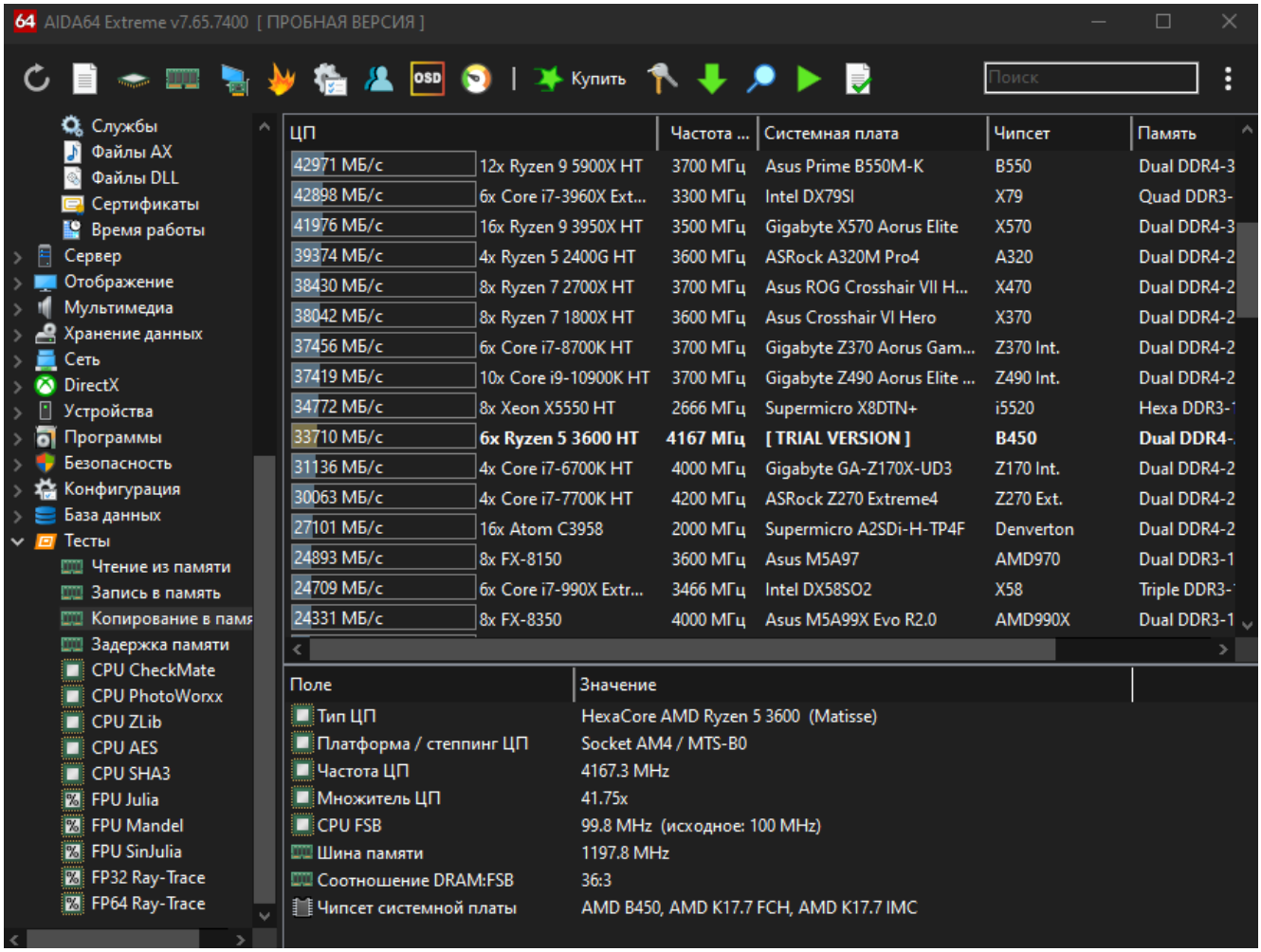


Рисунок 4.21 – Результат тестування копіювання в пам’яті після запуску модулів

Після налаштування у потрібних модулях програми, результат складає 33710 мегабайт за секунду. Отже, порівнявши ці два значення можна зробити висновок, швидкість пересилання даних з одних осередків пам’яті до інших через кеш процесора збільшилась на 6,04%.

Третім було проведено тестування читання з пам’яті. Цей тест показує швидкість пересилання даних з оперативної пам’яті до процесора. Модуль розроблюваного програмного застосунку, який повинен покращити результати цих показників – очищення дискового простору, редактор реєстру та менеджер автозапуску.

На рисунку 4.22 показано результат тестування швидкості пересилання даних з оперативної пам’яті до процесора до запуску відповідних модулів у розроблюваному програмному забезпеченні.

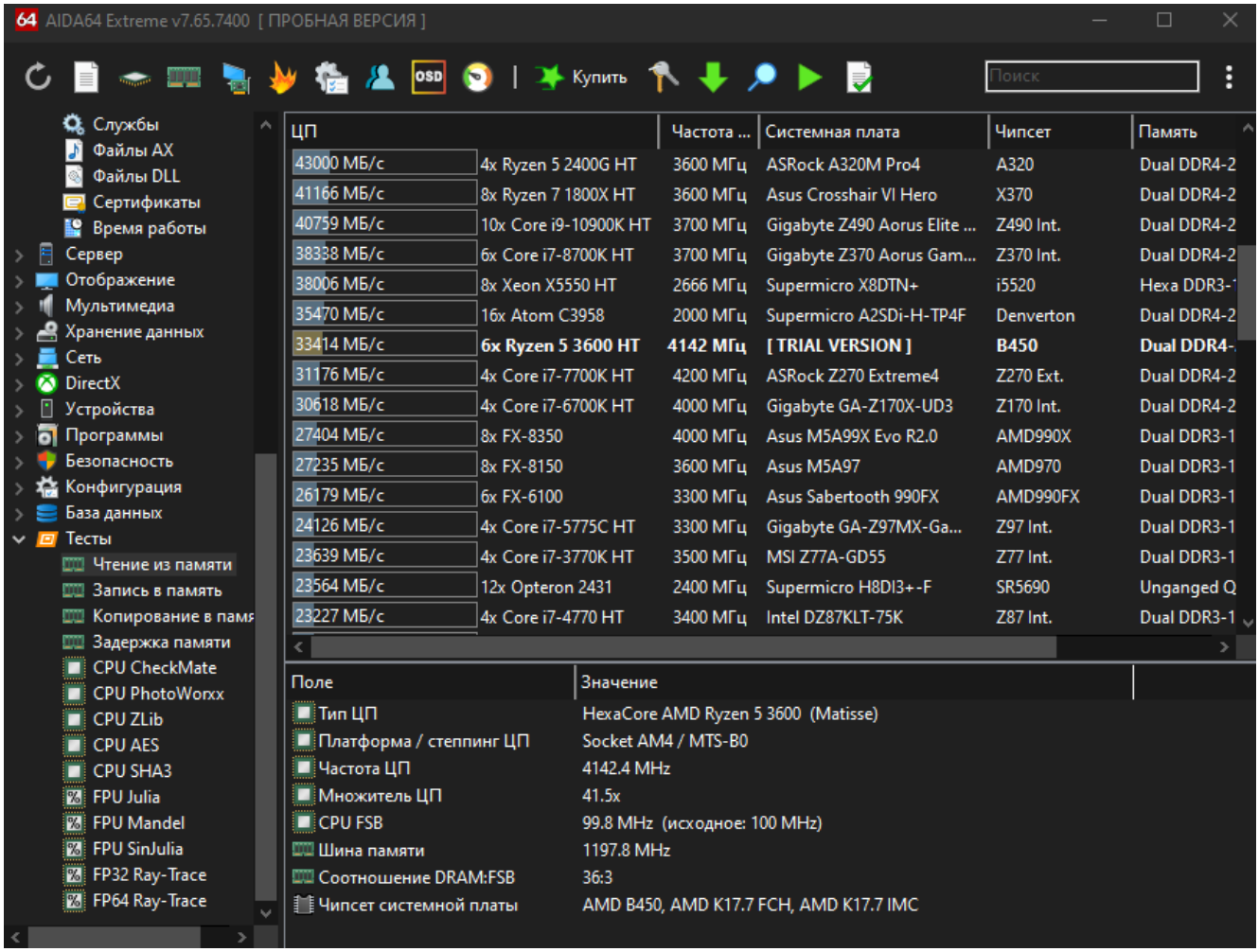


Рисунок 4.21 – Результат тестування читання з пам’яті до запуску модулів

Результат тестування показано у мегабайтах за секунду, тобто швидкість пересилання даних з оперативної пам’яті до процесора до запуску відповідних модулів у розроблюваному програмному забезпеченні становить 33414 мегабайти за секунду.

Після запуску модулів очищення дискового простору, редактор реєстру та менеджер автозапуску та налаштування параметрів у них та проведення повторного тестування копіювання в пам’яті, слід порівняти два значення до та після та знайти відсоток, на скільки швидкість зменшилась чи збільшилась.

На рисунку 4.23 показано результат тестування швидкості пересилання даних з оперативної пам’яті до процесора після запуску відповідних модулів у розроблюваному програмному забезпеченні.

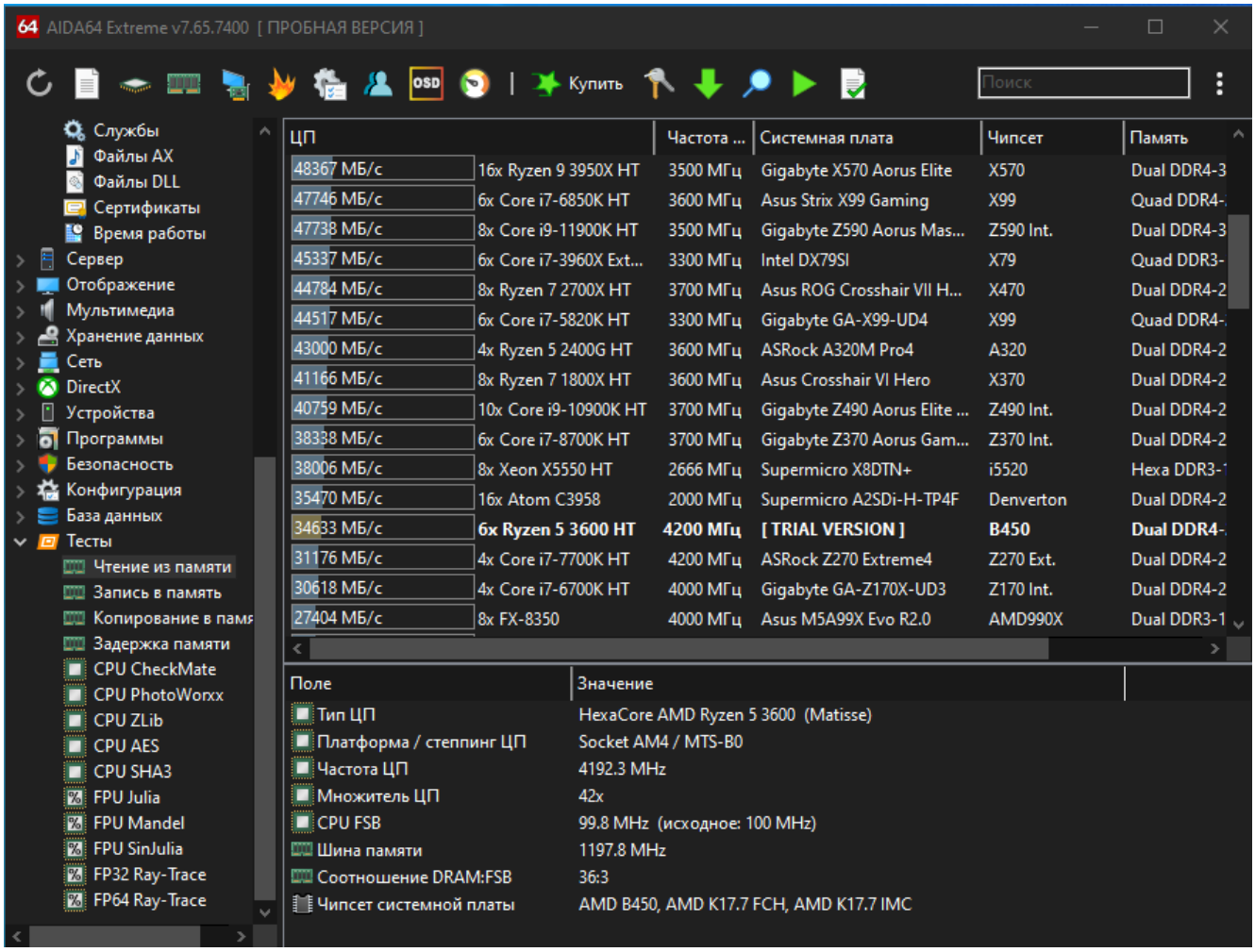


Рисунок 4.23 - Результат тестування читання з пам’яті після запуску модулів

Після налаштування у потрібних модулях програми, результат складає 34633 мегабайт за секунду. Отже, порівнявши ці два значення можна зробити висновок, що швидкість пересилання даних з оперативної пам’яті до процесора збільшилась на 3,65%.

Далі, було проведено порівняння температурних режимів. Цей тест показує поточні температури окремих комплектуючих персонального комп’ютера. В цілому, кожен модуль який є у програмному застосунку комплексно впливає на цей показник, тому слід проводити тестування після запуску всіх модулів розроблюваної програми.

На рисунку 4.24 показано результат тестування температурних режимів до запуску відповідних модулів у розроблюваному програмному забезпеченні.

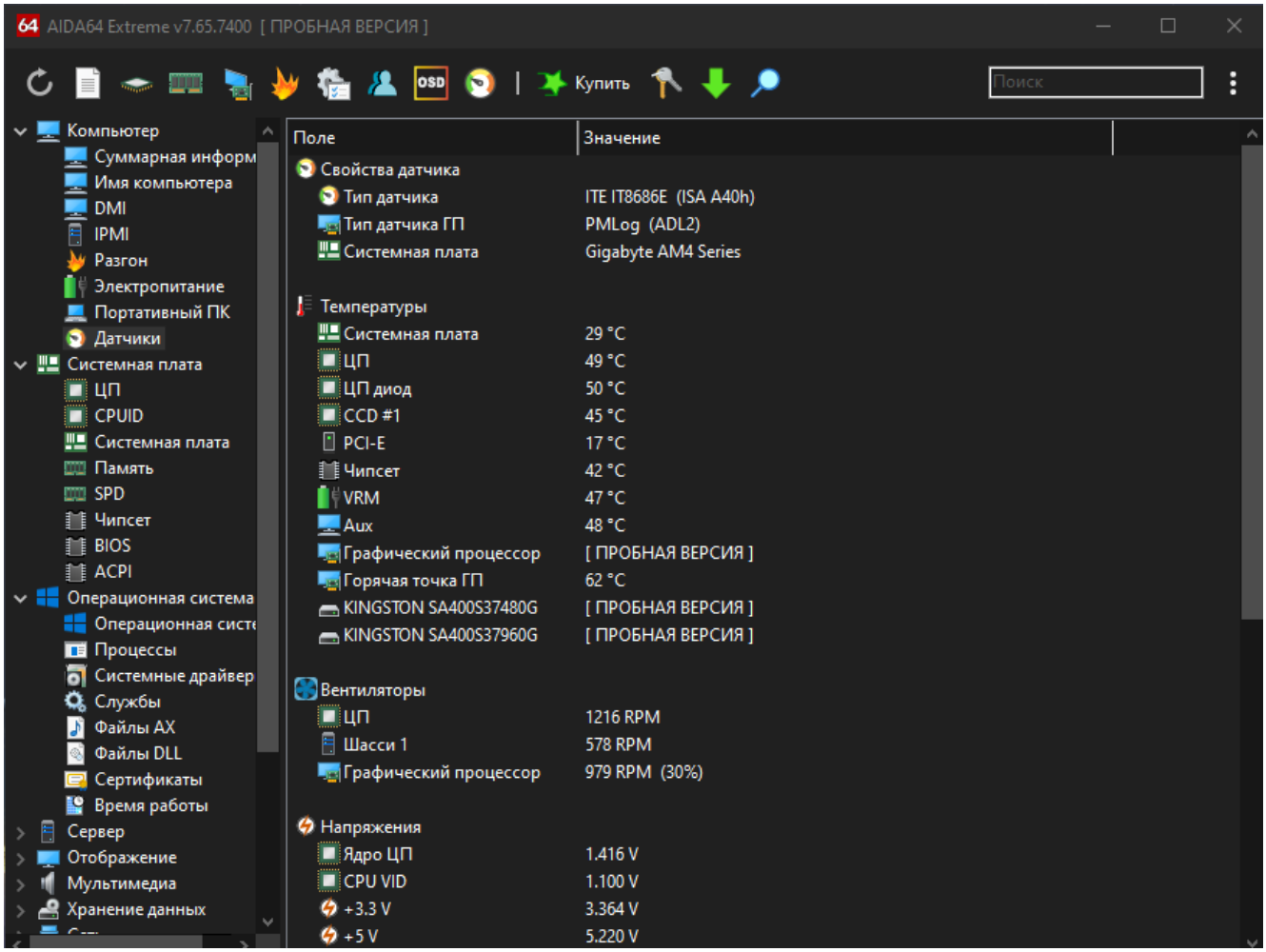


Рисунок 4.24 – Результат тестування температурних режимів до запуску модулів

Результат тестування показано у градусах Цельсію. Слід звернути увагу на показники температури центрального процесора, а також графічного процесора, які становлять 43 та 60 градусів відповідно. Також, варто звернути увагу на кількість обертів вентилятора за хвилину, тобто RPM у вентилятора, який автоматично підлаштовує швидкість обертання вентилятора під поточні температури.

Після запуску модулів програмного застосунку та проведення повторного тестування температурного режиму, слід порівняти два значення до та після та знайти відсоток, на скільки температура, а також швидкість обертів вентилятора змінилась.

На рисунку 4.25 показано результат тестування температурного режиму після запуску відповідних модулів у розроблюваному програмному забезпеченні.

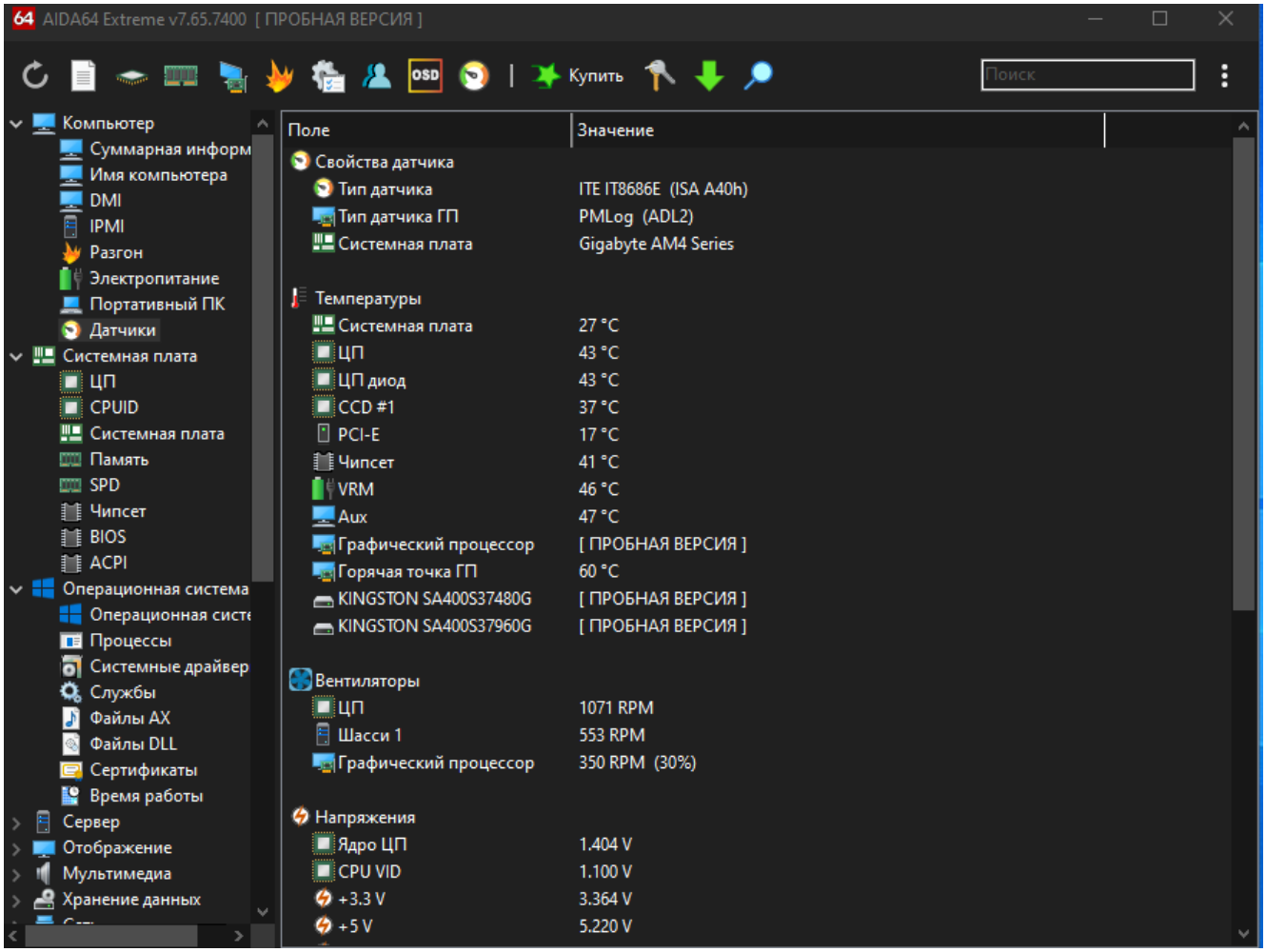


Рисунок 4.25 - Результат тестування температурного режиму після запуску модулів

Після налаштування у потрібних модулях програми, результат складає 49 градусів для центрального процесора, а також 62 градуси для графічного процесора. Отже, порівнявши ці значення можна зробити висновок, що температура центрального процесора зменшилась на 12,24%, а температура графічного процесора зменшилась на 3,23%. Ці покращення дозволять значно подовжити строк експлуатації комплектуючих персонального комп’ютера.

Також, варто звернути увагу на показники RPM. До запуску оптимізаційних процесів програми, швидкість обертання вентилятора складала 1216 обертів на хвилину, а після оптимізації системи за допомогою розроблюваного програмного забезпечення складає 1071 оберт на хвилину. Це істотно зменшує кількість шуму, які викликає вентилятор, а також зменшує фактор зносу.

Останнім тестом було порівняно вільний дисковий простір. Цей тест показує поточні значення заповнення дискового простору.

На рисунку 4.26 показано результат перевірки дискового простору до очищення у програмному застосунку.

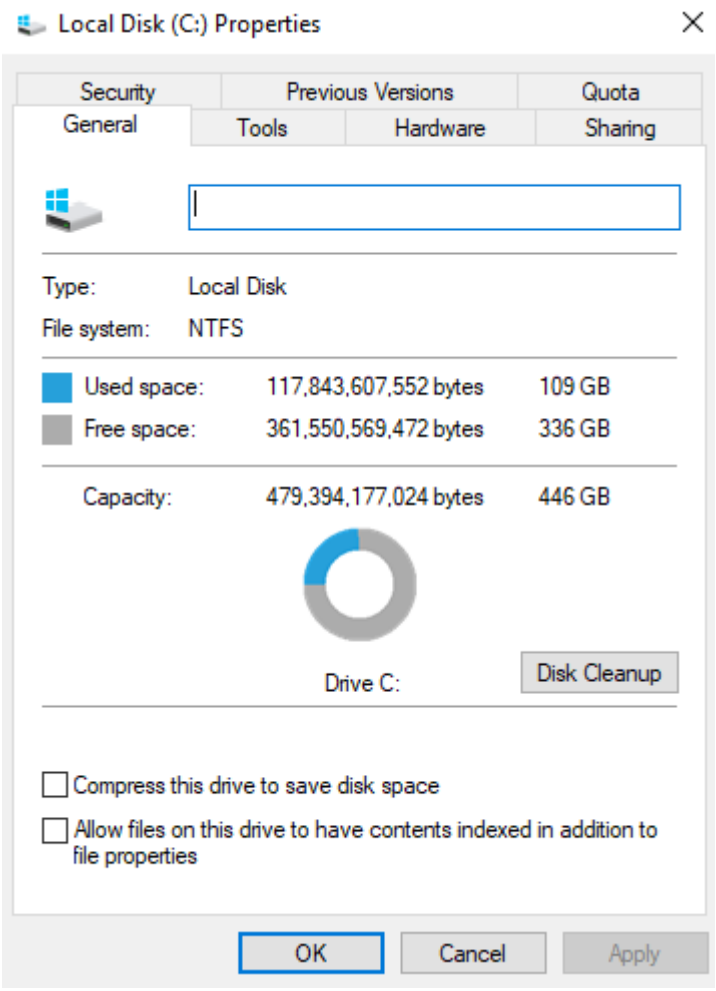


Рисунок 4.26 – Результат перевірки дискового простору до очищення у програмному застосунку

Результат показано у байтах та гігабайтах. До очищення дискового простору у системі, ємність вільного місця на диску складає 336 гігабайт.

Після проведення очистки та повторного тестування, слід порівняти значення вільного дискового простору до та після використання програми, та зробити висновок, який відсоток вільного дискового простору з’явився, завдяки очищенню тимчасових файлів за допомогою програмного застосунку.

На рисунку 4.27 показано результат перевірки дискового простору після очищення у програмному застосунку.

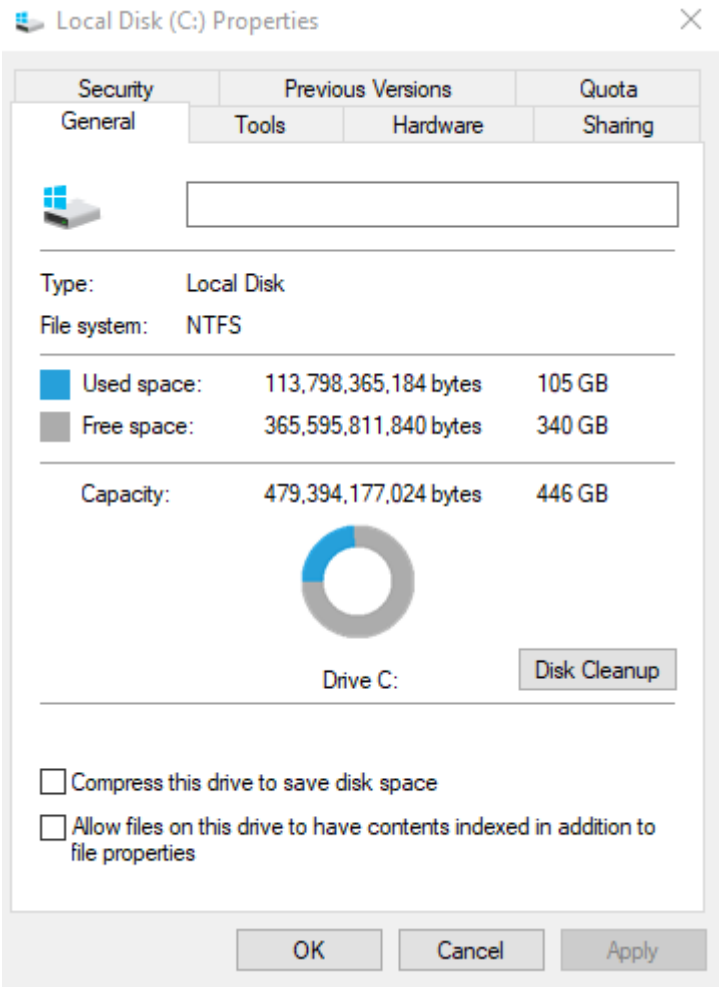


Рисунок 4.27 – Результат перевірки дискового простору після очищення у програмному застосунку

Після проведення очистки тимчасових файлів у програмному забезпеченні, ємність вільного місця на диску становить 340 гігабайт, що порівняно з показниками до очистки, складає близько 0,89%.

Підсумок зміни числових показників:

1. Зменшення середнього часу зчитування процесором даних із оперативної пам’яті на 9,4%.
2. Підвищення швидкості пересилання даних з одних осередків пам’яті до інших через кеш процесора на 6,04%.

3. Підвищення швидкості пересилання даних з оперативної пам’яті до процесора на 3,65%.
4. Покращення температурного режиму для центрального та графічного процесорів на 12,24% та 3,23% відповідно.
5. Збільшення вільного дискового простору на 0,89%.

4.4 Висновки

У четвертому розділі було проведено тестування програмних модулів застосунку для аналізу та підвищення продуктивності роботи комп’ютера. Було перевірено реакцію програмних засобів на помилкові ситуації, реакцію програми на запуск функцій від імені адміністратора та на звичайний запуск. Також, було розроблено інструкцію користувача по експлуатації програмного продукту. Також, було проведено експериментальну перевірку програмного застосунку. Проведена експериментальна перевірка показала, що застосування розроблених програмних засобів дозволяє підвищити низку важливих показників продуктивності роботи комп’ютера на 1 – 12%.

ВИСНОВКИ

У рамках бакалаврської кваліфікаційної роботи були розроблені модулі програмного забезпечення для аналізу та підвищення продуктивності роботи комп’ютера. Для розробки використовувалося середовище програмування Visual Studio. Реалізація бакалаврської кваліфікаційної роботи відповідає методичним вказівкам [26].

Під час виконання роботи було проведено аналіз сучасного стану проблеми. Були розглянуті основні аналоги програмного продукту, виявлені їхні недоліки та порівняно з власним розробленим продуктом. На основі цього порівняння були сформульовані основні завдання бакалаврської кваліфікаційної роботи. Під час аналізу технологій розробки було обґрунтовано вибір мов програмування C#, а також бібліотек, які сприяють розробці програмного застосунку.

У бакалаврській кваліфікаційній роботі було зроблено наступне:

- розроблено алгоритм аналізу продуктивності, який забезпечать оцінку навантаження на систему та ефективність оптимізації використання ресурсів;
- розроблено модуль моніторингу системних ресурсів, який буде відстежувати, аналізувати та відображати основні параметри продуктивності комп’ютера;
- розроблено модуль графічного інтерфейсу, який буде зручним у використанні для користувачів, забезпечуючи наочне відображення стану системи;
- інтегровано різні модулі в єдиний програмний засіб, який дозволить проводити комплексний аналіз продуктивності комп’ютера.

Розроблено модель системи з використанням UML діаграм, схеми загального алгоритму роботи програмного застосунку. Тестування програми довело повну працездатність даного програмного продукту та відповідність поставленому технічному завданню. Проведена експериментальна перевірка показала, що застосування розроблених програмних засобів дозволяє підвищити низку важливих показників продуктивності роботи комп’ютера на 3 – 12%. Отже мету бакалаврської кваліфікаційної роботи досягнуто.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Поради щодо підвищення продуктивності ПК у Windows. [Електронний ресурс] – Режим доступу до ресурсу: https://goo.su/z1Nwigm (дата звернення 02.03.2025)

2. Як оптимізувати роботу ПК. [Електронний ресурс] – Режим доступу до ресурсу: https://www.kingston.com/ua/blog/pc-performance/how-to-optimize-pc (дата звернення 02.03.2025)

3. Culbertson R. B., Chris B.J., Rapid Testing. Williams: Prentice Hall, 2002. 374c.

4. Сініцин С. В., Налютін Н. Ю. Верифікація програмного забезпечення. Москва: ИНТУИТ, 2008. 368с.

5. Куліков С. С. Тестування програмного забезпечення. Базовий курс. Мінськ: Чотири чверті, 2017. 312 с.

6. Ведельський В.Ю. Програмний застосунок для аналізу та підвищення продуктивності роботи комп’ютера. LIV Всеукраїнська науково-технічна конференція факультету інформаційних технологій та комп'ютерної інженерії Вінниця: ВНТУ, 2025. 3с. [Електронний ресурс] – Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2025/paper/view/24372 (дата звернення 02.03.2025)

7. Ведельський В.Ю. Розробка методу резервного копіювання. XIII Міжнародна науково-практична конференція з інформаційних систем та технологій Infocom Advanced Solutions 2025. Київ: КПІ, 2025. 3с. [Електронний ресурс] – Режим доступу до ресурсу: https://ist.kpi.ua/uk/konferencziya-infocom/ (дата звернення 02.03.2025)

8. Сучасний розвиток інформаційних технологій. [Електронний ресурс] – Режим доступу до ресурсу: https://profosvita.od.ua/a193741-suchasnij-navchalnij-protses.html (дата звернення 15.03.2025)

9. Важливість моніторингу систем. [Електронний ресурс] – Режим доступу до ресурсу: https://www.globalyo.com/uk/blog/the-importance-of-network-monitoring-ensuring-optimal-performance-and-security/ (дата звернення 15.03.2025)

10. Дослідження різноманітності програмного забезпечення для комп'ютерного моніторингу. [Електронний ресурс] – Режим доступу до ресурсу: https://clevercontrol.com/uk/exploring-the-diversity/ (дата звернення 15.03.2025)

11. Що таке Windows Scanreg. [Електронний ресурс] – Режим доступу до ресурсу: https://www.oocities.org/~budallen/scanreg.html (дата звернення 15.03.2025)

12. Що представляє собою CCleaner. [Електронний ресурс] – Режим доступу до ресурсу: https://support.ccleaner.com/s/article/what-is-ccleaner-for-windows?language=ru (дата звернення 15.03.2025)

13. Огляд Auslogics Registry Cleaner. [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Auslogics_Registry_Cleaner (дата звернення 15.03.2025)

14. Огляд RegOrganizer. [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Reg_Organizer (дата звернення 15.03.2025)

15. C# як мова програмування. [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/hubs/csharp/articles/ (дата звернення 18.03.2025)

16. Що нового у Windows Forms в .NET 6.0. [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/companies/microsoft/articles/590057/ (дата звернення 18.03.2025)

17. Understanding and Using Windows Performance Counters. [Електронний ресурс] – Режим доступу до ресурсу: https://openobserve.ai/articles/windows-perf-counters-receiver/ (дата звернення 18.03.2025)

18. Що таке .NET Framework. [Електронний ресурс] – Режим доступу до ресурсу: https://dotnet.microsoft.com/ru-ru/learn/dotnet/what-is-dotnet-framework (дата звернення 18.03.2025)

19. Visual Studio як основний інструмент розробника програмного забезпечення. [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/hubs/vs/articles/ (дата звернення 25.03.2025)

20. Елементи Windows Forms. [Електронний ресурс] – Режим доступу до ресурсу: https://goo.su/ksU6XV (дата звернення 25.03.2025)

21. ComputerInfo Клас. [Електронний ресурс] – Режим доступу до ресурсу: https://goo.su/mTXZk (дата звернення 25.03.2025)

22. Папка %TEMP% і що вона собою представляє. [Електронний ресурс] – Режим доступу до ресурсу: https://goo.su/WBVn (дата звернення 25.03.2025)

23. Різниця між функціональним та нефункціональним тестуванням. [Електронний ресурс] – Режим доступу до ресурсу: https://goo.su/NTIBK (дата звернення 14.04.2025)

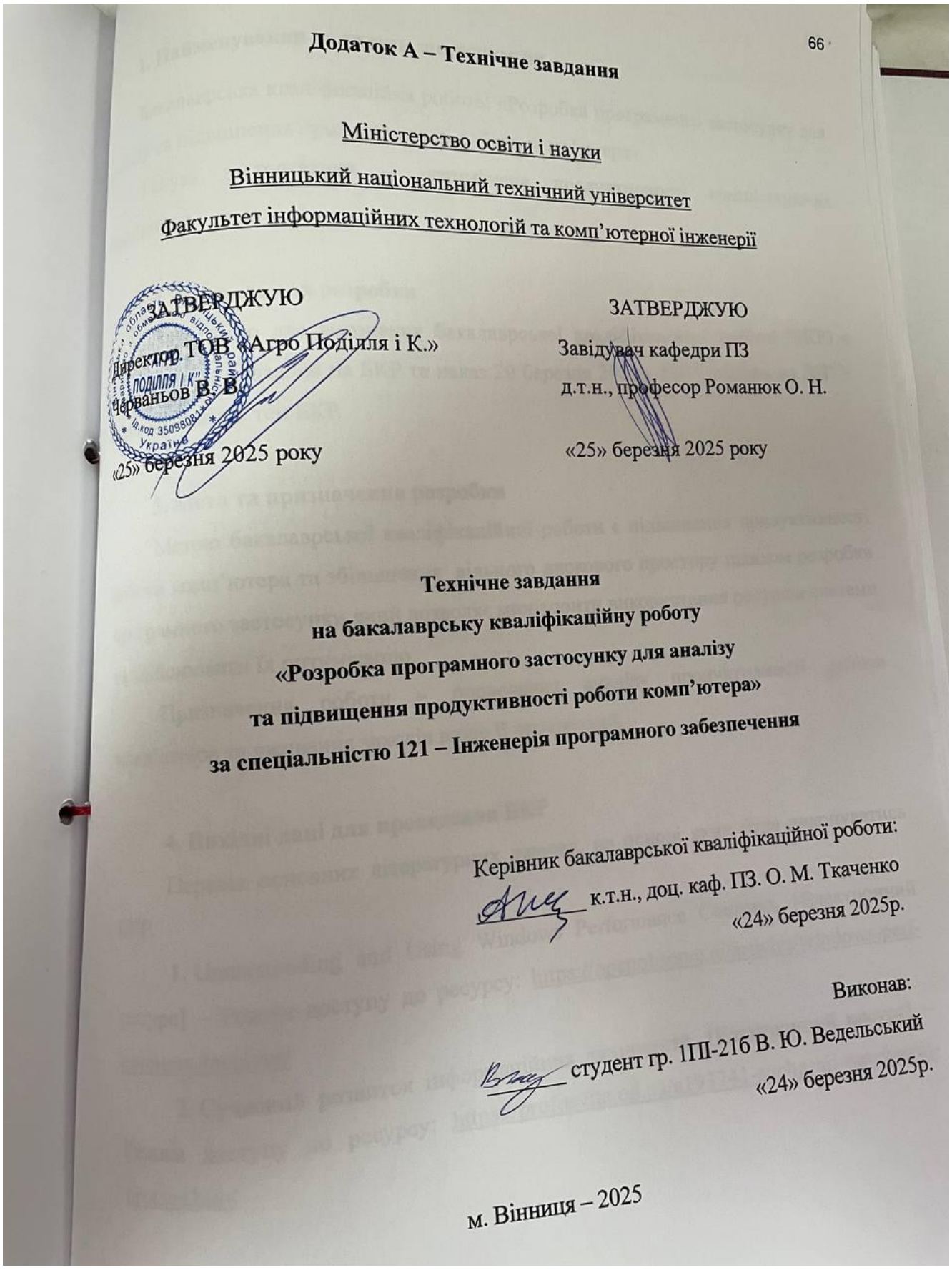
24. Функціональне тестування. [Електронний ресурс] – Режим доступу до ресурсу: https://qalight.ua/baza-znaniy/funktsionalne-testuvannya/ (дата звернення 14.04.2025)

25. Нефункціональне тестування. [Електронний ресурс] – Режим доступу до ресурсу: https://qalight.ua/baza-znaniy/nefunktsionalne-testuvannya/ (дата звернення 14.04.2025)

26. Романюк О. Н., Чорноволик Г. О., Методичні вказівки до виконання бакалаврських дипломних робіт для студентів спеціальності 121 «Інженерія програмного забезпечення»: методичні – вказівки Вінниця : ВНТУ, 2022. 51 с.

ДОДАТКИ

Додаток А – Технічне завдання



Додаток А – Технічне завдання 66

Міністерство освіти і науки

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ

Директор ТОВ «Агро Поділля і К.»

Черваньов В. В.

«25» березня 2025 року

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ

д.т.н., професор Романюк О. Н.

«25» березня 2025 року

Технічне завдання

на бакалаврську кваліфікаційну роботу

«Розробка програмного застосунку для аналізу

та підвищення продуктивності роботи комп'ютера»

за спеціальністю 121 – Інженерія програмного забезпечення

Керівник бакалаврської кваліфікаційної роботи:

к.т.н., доц. каф. ПЗ. О. М. Ткаченко

«24» березня 2025р.

Виконав:

студент гр. 1ПІ-21б В. Ю. Ведельський

«24» березня 2025р.

м. Вінниця – 2025

1. Найменування та галузь застосування

Бакалаврська кваліфікаційна робота: «Розробка програмного застосунку для аналізу та підвищення продуктивності роботи комп’ютера»

Галузь застосування – оптимізація продуктивності комплектуючих комп’ютера.

2. Підстава для розробки

Підставою для виконання бакалаврської кваліфікаційної роботи (БКР) є індивідуальне завдання на БКР та наказ 20 березня 2025р. №97 ректора по ВНТУ про закріплення тем БКР.

3. Мета та призначення розробки

Метою бакалаврської кваліфікаційної роботи є підвищення продуктивності роботи комп’ютера та збільшення вільного дискового простору шляхом розробки програмного застосунку, який дозволяє моніторити використання ресурсів системи та здійснювати їх оптимізацію.

Призначення роботи – проведення аналізу продуктивності роботи комп’ютера та вживання заходів щодо її оптимізації.

4. Вихідні дані для проведення БКР

Перелік основних літературних джерел, на основі яких буде виконуватись БКР.

1. Understanding and Using Windows Performance Counters. [Електронний ресурс] – Режим доступу до ресурсу: https://openobserve.ai/articles/windows-perf-counters-receiver/

2. Сучасний розвиток інформаційних технологій. [Електронний ресурс] – Режим доступу до ресурсу: https://profosvita.od.ua/a193741-suchasnij-navchalnij-protses.html

3. Як оптимізувати роботу ПК. [Електронний ресурс] – Режим доступу до ресурсу: https://www.kingston.com/ua/blog/pc-performance/how-to-optimize-pc

4. ComputerInfo Клас. [Електронний ресурс] – Режим доступу до ресурсу: https://goo.su/mTXZk

5. Технічні вимоги

Методи перегляду дерева реєстру, методи редагування дерева реєстру, методи моніторингу за навантаженням центрального процесора та оперативної пам’яті, методи менеджера процесів, методи менеджера автозапуску, методи очищення дискового простору, методи резервного копіювання реєстру.

6. Конструктивні вимоги

Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації, що пред’являється по закінченню робіт

1. Пояснювальна записка до БКР.
2. Технічне завдання.
3. Лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ

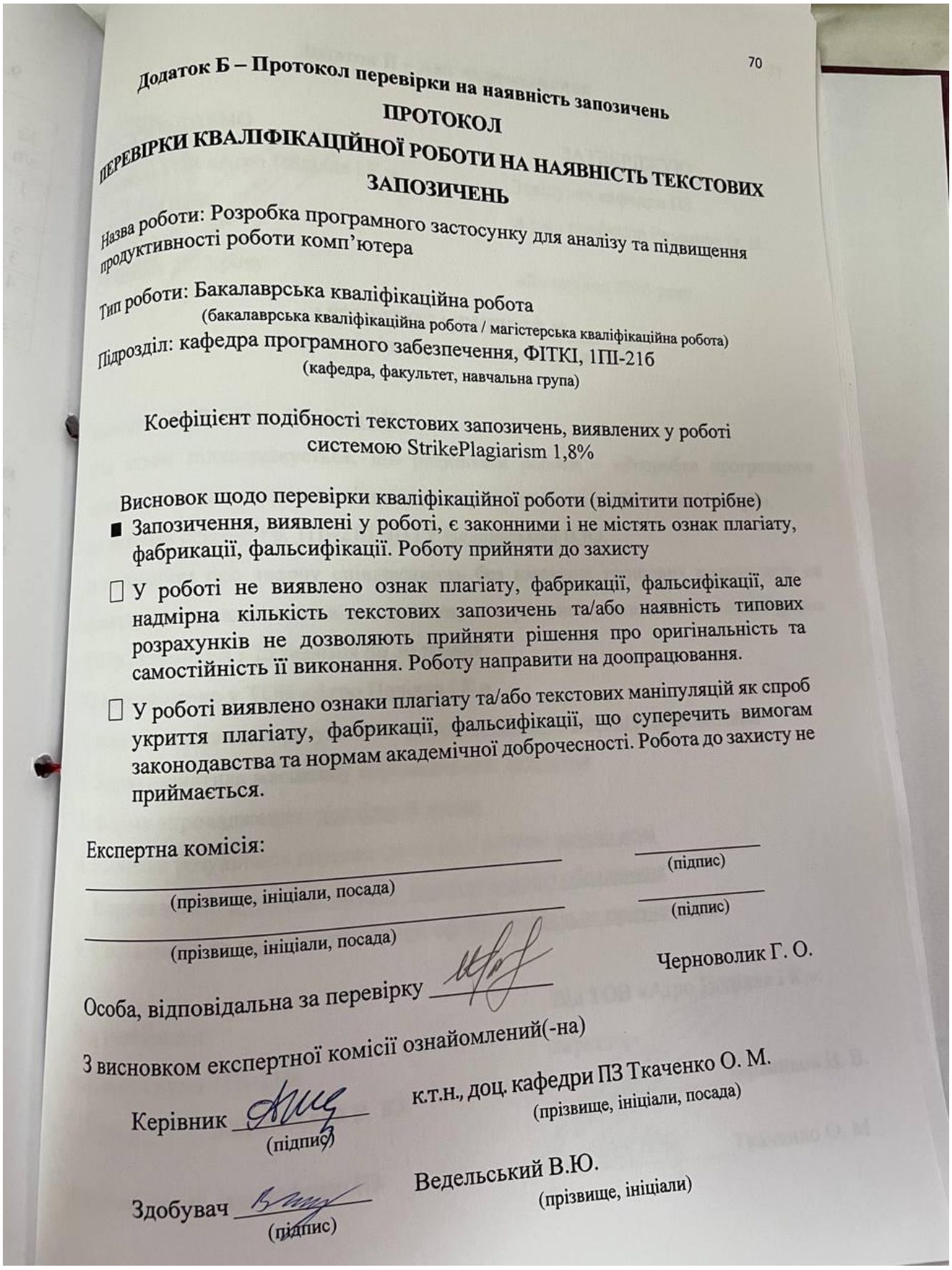
9. Стадії та етапи розробки

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи
1.	Аналіз розвитку систем моніторингу та процесів підвищення продуктивності роботи комп’ютера	25.03.25-05.04.25
2.	Розробка моделі та методів програмного забезпечення	06.04.25-13.04.25
3.	Розробка програмних модулів	14.04.25-07.05.25
4.	Тестування та практичне застосування програмного забезпечення	08.05.25-20.05.25
5.	Оформлення матеріалів до захисту БКР	21.05.25-30.05.25

10. Порядок контролю та прийняття

Виконання етапів бакалаврської кваліфікаційної роботи контролюється керівником згідно з графіком виконання роботи. Прийняття бакалаврської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком.

Додаток Б – Протокол перевірки на наявність запозичень



Додаток Б – Протокол перевірки на наявність запозичень 70

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Розробка програмного застосунку для аналізу та підвищення продуктивності роботи комп'ютера

Тип роботи: Бакалаврська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ: кафедра програмного забезпечення, ФІТКІ, 1ПІ-21б
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 1,8%

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

■ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

_____ _____
(прізвище, ініціали, посада) (підпис)

_____ _____
(прізвище, ініціали, посада) (підпис)

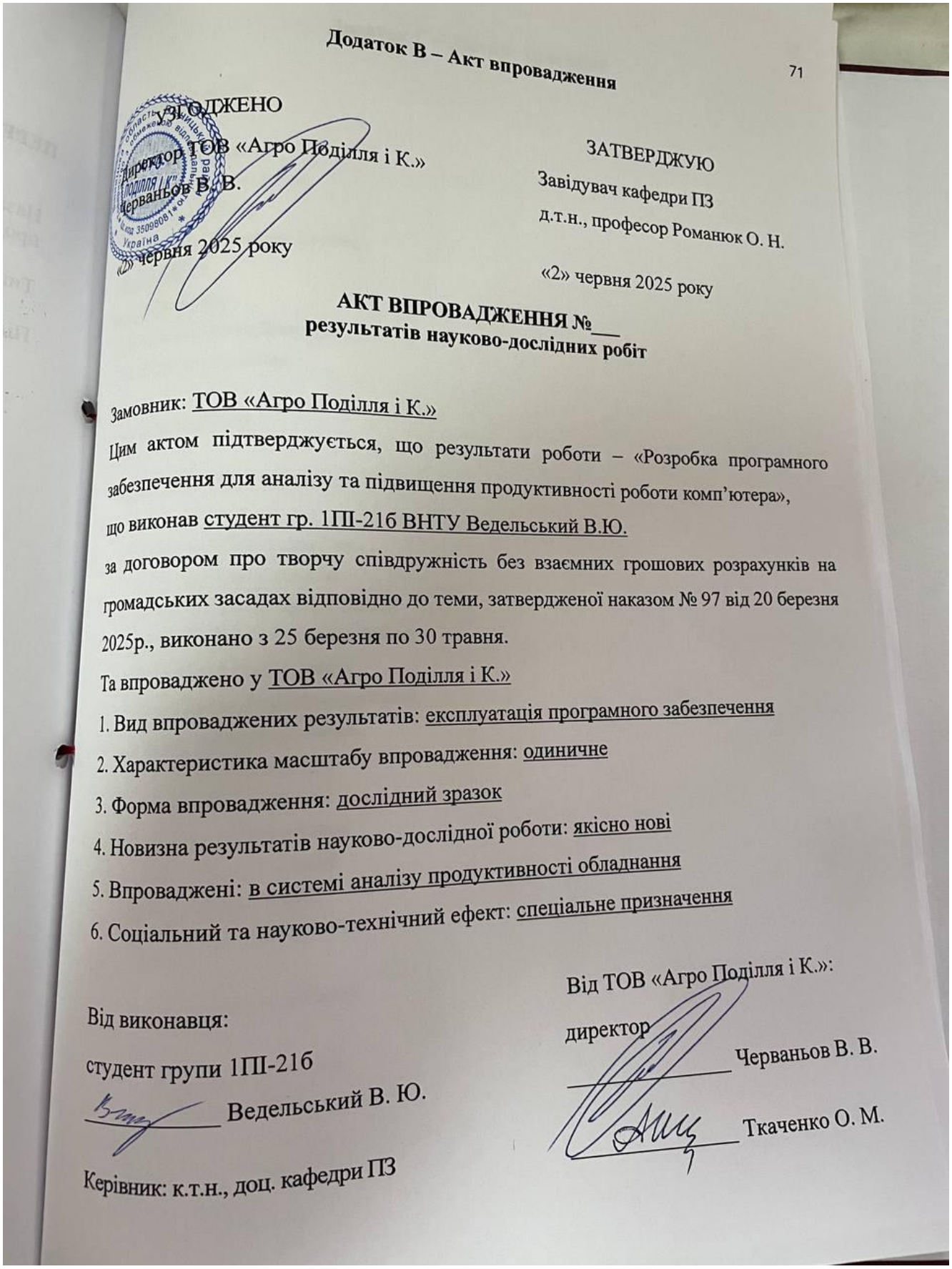
Особа, відповідальна за перевірку _____ Черноволик Г. О.

З висновком експертної комісії ознайомлений(-на)

Керівник _____ к.т.н., доц. кафедри ПЗ Ткаченко О. М.
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Ведельський В.Ю.
(підпис) (прізвище, ініціали)

Додаток В – Акт впровадження



Додаток В – Акт впровадження 71

УЗГОДЖЕНО
Директор ТОВ «Агро Поділля і К.»
Черваньов В. В.
«2» червня 2025 року

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
д.т.н., професор Романюк О. Н.
«2» червня 2025 року

АКТ ВПРОВАДЖЕННЯ №___
результатів науково-дослідних робіт

Замовник: ТОВ «Агро Поділля і К.»

Цим актом підтверджується, що результати роботи – «Розробка програмного забезпечення для аналізу та підвищення продуктивності роботи комп'ютера», що виконав студент гр. 1ПІ-21б ВНТУ Ведельський В.Ю.

за договором про творчу співдружність без взаємних грошових розрахунків на громадських засадах відповідно до теми, затвердженої наказом № 97 від 20 березня 2025р., виконано з 25 березня по 30 травня.

Та впроваджено у ТОВ «Агро Поділля і К.»

1. Вид впроваджених результатів: експлуатація програмного забезпечення
2. Характеристика масштабу впровадження: одиничне
3. Форма впровадження: дослідний зразок
4. Новизна результатів науково-дослідної роботи: якісно нові
5. Впроваджені: в системі аналізу продуктивності обладнання
6. Соціальний та науково-технічний ефект: спеціальне призначення

Від виконавця:
студент групи 1ПІ-21б
Ведельський В. Ю.

Керівник: к.т.н., доц. кафедри ПЗ

Від ТОВ «Агро Поділля і К.»:
директор
Черваньов В. В.
Ткаченко О. М.

Додаток Г – Лістинг програми

```
using Microsoft.Win32;
using System;
using System.Diagnostics;
using System.Windows.Forms;
using System.Drawing;
using Microsoft.VisualBasic.Devices;
using System.IO;
using System.Linq;
using System.Drawing.Drawing2D;
using RegCheck.Properties;

namespace RegCheck
{
   public class RegistryEditorForm : Form
   {
      private TreeView registryTree;
      private ListView valuesList;
      private Button taskManagerButton;
      private Button startupManagerButton;
      private Button backupButton;
      private Button processManagerButton;
      private Button cleanupButton;
      private Button addButton;
      private Button editButton;
      private Button deleteButton;
      private Panel toolbarPanel;
      private Panel treePanel;
      private Panel listPanel;

      public RegistryEditorForm()
      {
         InitializeComponents();
         LoadRegistryTree();
      }
```

```
private void InitializeComponents()
{
    this.Text = "RegCheck";
    this.Width = 900;
    this.Height = 600;
    this.FormBorderStyle = FormBorderStyle.FixedSingle;
    this.MaximizeBox = false;
    this.BackColor = Color.FromArgb(33, 33, 33);
    this.DoubleBuffered = true;

    toolbarPanel = new Panel
    {
        Location = new Point(0, 0),
        Size = new Size(this.ClientSize.Width, 70),
        BorderStyle = BorderStyle.None,
        BackColor = Color.FromArgb(46, 46, 46)
    };
    toolbarPanel.Paint += (s, e) =>
    {
        using (var pen = new Pen(Color.FromArgb(66, 66, 66), 1))
        {
            e.Graphics.DrawLine(pen, 0, toolbarPanel.Height - 1, toolbarPanel.Width, toolbarPanel.Height - 1);
        }
    };

    Image taskManagerIcon;
    try
    {
        byte[] monitoringBytes = Properties.Resources.monitoring;
        using (var ms = new MemoryStream(monitoringBytes))
        {
            taskManagerIcon = new Bitmap(Image.FromStream(ms), new Size(24, 24));
        }
    }
    catch
    {
        taskManagerIcon = new Bitmap(SystemIcons.Information.ToBitmap(), new Size(24, 24));
    }
```

```
Image startupIcon;
try
{
   byte[] autoBytes = Properties.Resources.auto;
   using (var ms = new MemoryStream(autoBytes))
   {
      startupIcon = new Bitmap(Image.FromStream(ms), new Size(24, 24));
   }
}
catch
{
   startupIcon = new Bitmap(SystemIcons.Shield.ToBitmap(), new Size(24, 24));
}

Image backupIcon;
try
{
   byte[] backupBytes = Properties.Resources.backup;
   using (var ms = new MemoryStream(backupBytes))
   {
      backupIcon = new Bitmap(Image.FromStream(ms), new Size(24, 24));
   }
}
catch
{
   backupIcon = new Bitmap(SystemIcons.WinLogo.ToBitmap(), new Size(24, 24));
}

Image processIcon;
try
{
   byte[] processBytes = Properties.Resources.process;
   using (var ms = new MemoryStream(processBytes))
   {
      processIcon = new Bitmap(Image.FromStream(ms), new Size(24, 24));
   }
}
```

```
        catch
        {
            processIcon = new Bitmap(SystemIcons.Application.ToBitmap(), new Size(24, 24));
        }

        Image cleanupIcon;
        try
        {
            byte[] cleanupBytes = Properties.Resources.cleanup;
            using (var ms = new MemoryStream(cleanupBytes))
            {
                cleanupIcon = new Bitmap(Image.FromStream(ms), new Size(24, 24));
            }
        }
        catch
        {
            cleanupIcon = new Bitmap(SystemIcons.Warning.ToBitmap(), new Size(24, 24));
        }

        taskManagerButton = CreateModernButton("Диспетчер завдань", new Point(10, 15), taskManagerIcon,
TaskManagerButton_Click);
        startupManagerButton = CreateModernButton("Автозапуск", new Point(160, 15), startupIcon,
StartupManagerButton_Click);
        backupButton = CreateModernButton("Резервна копія", new Point(310, 15), backupIcon,
BackupButton_Click);
        processManagerButton = CreateModernButton("Процеси", new Point(460, 15), processIcon,
ProcessManagerButton_Click);
        cleanupButton = CreateModernButton("Очищення", new Point(610, 15), cleanupIcon,
CleanupButton_Click);

        toolbarPanel.Controls.Add(taskManagerButton);
        toolbarPanel.Controls.Add(startupManagerButton);
        toolbarPanel.Controls.Add(backupButton);
        toolbarPanel.Controls.Add(processManagerButton);
        toolbarPanel.Controls.Add(cleanupButton);

        treePanel = new Panel
        {
```

```
            Location = new Point(10, 80),
            Size = new Size(310, this.ClientSize.Height - 90),
            BorderStyle = BorderStyle.None,
            BackColor = Color.FromArgb(46, 46, 46)
        };
        treePanel.Paint += (s, e) =>
        {
            ControlPaint.DrawBorder(e.Graphics, treePanel.ClientRectangle, Color.FromArgb(66, 66, 66), ButtonBorderStyle.Solid);
        };

        registryTree = new TreeView
        {
            Location = new Point(5, 5),
            Size = new Size(300, treePanel.Height - 10),
            BackColor = Color.FromArgb(46, 46, 46),
            ForeColor = Color.FromArgb(224, 224, 224),
            BorderStyle = BorderStyle.None,
            Font = new Font("Segoe UI Semibold", 10F),
            ItemHeight = 28,
            DrawMode = TreeViewDrawMode.OwnerDrawText
        };
        registryTree.DrawNode += (s, e) =>
        {
            if (e.Node.IsSelected)
            {
                e.Graphics.FillRectangle(new SolidBrush(Color.FromArgb(97, 97, 97)), e.Bounds);
                e.Graphics.DrawString(e.Node.Text, registryTree.Font, new SolidBrush(Color.White), e.Bounds.Left + 2, e.Bounds.Top + 4);
            }
            else
            {
                e.Graphics.DrawString(e.Node.Text, registryTree.Font, new SolidBrush(registryTree.ForeColor), e.Bounds.Left + 2, e.Bounds.Top + 4);
            }
        };
        registryTree.AfterSelect += RegistryTree_AfterSelect;
        registryTree.BeforeExpand += RegistryTree_BeforeExpand;
```

```
        treePanel.Controls.Add(registryTree);

        listPanel = new Panel
        {
            Location = new Point(325, 80),
            Size = new Size(this.ClientSize.Width - 335, this.ClientSize.Height - 90),
            BorderStyle = BorderStyle.None,
            BackColor = Color.FromArgb(46, 46, 46)
        };
        listPanel.Paint += (s, e) =>
        {
            ControlPaint.DrawBorder(e.Graphics, listPanel.ClientRectangle, Color.FromArgb(66, 66, 66),
ButtonBorderStyle.Solid);
        };

        valuesList = new ListView
        {
            Location = new Point(5, 5),
            Size = new Size(listPanel.Width - 10, listPanel.Height - 50),
            BackColor = Color.FromArgb(46, 46, 46),
            ForeColor = Color.FromArgb(224, 224, 224),
            BorderStyle = BorderStyle.None,
            View = View.Details,
            FullRowSelect = true,
            GridLines = true,
            Font = new Font("Segoe UI", 10F)
        };
        valuesList.Columns.Add("Назва", 200);
        valuesList.Columns.Add("Тип", 100);
        valuesList.Columns.Add("Значення", 300);
        valuesList.SelectedIndexChanged += ValuesList_SelectedIndexChanged;

        addButton = new Button
        {
            Text = "Додати",
            Location = new Point(5, listPanel.Height - 40),
            Size = new Size(100, 30),
            FlatStyle = FlatStyle.Flat,
```

```
    ForeColor = Color.FromArgb(224, 224, 224),
    Font = new Font("Segoe UI Semibold", 9F),
    BackColor = Color.FromArgb(51, 51, 51)
};
addButton.MouseEnter += (s, e) =>
{
    addButton.BackColor = Color.FromArgb(97, 97, 97);
    addButton.Invalidate();
};
addButton.MouseLeave += (s, e) =>
{
    addButton.BackColor = Color.FromArgb(51, 51, 51);
    addButton.Invalidate();
};
addButton.Click += (s, e) => AddNewRegistryValue();

editButton = new Button
{
    Text = "Редагувати",
    Location = new Point(115, listPanel.Height - 40),
    Size = new Size(100, 30),
    FlatStyle = FlatStyle.Flat,
    ForeColor = Color.FromArgb(224, 224, 224),
    Font = new Font("Segoe UI Semibold", 9F),
    BackColor = Color.FromArgb(51, 51, 51),
    Enabled = false
};
editButton.MouseEnter += (s, e) =>
{
    if (editButton.Enabled)
    {
        editButton.BackColor = Color.FromArgb(97, 97, 97);
        editButton.Invalidate();
    }
};
editButton.MouseLeave += (s, e) =>
{
    editButton.BackColor = Color.FromArgb(51, 51, 51);
```

```
    editButton.Invalidate();
};
editButton.Click += (s, e) =>
{
    if (valuesList.SelectedItems.Count > 0)
    {
        EditRegistryValue(valuesList.SelectedItems[0]);
    }
};

deleteButton = new Button
{
    Text = "Видалити",
    Location = new Point(225, listPanel.Height - 40),
    Size = new Size(100, 30),
    FlatStyle = FlatStyle.Flat,
    ForeColor = Color.FromArgb(224, 224, 224),
    Font = new Font("Segoe UI Semibold", 9F),
    BackColor = Color.FromArgb(51, 51, 51),
    Enabled = false
};
deleteButton.MouseEnter += (s, e) =>
{
    if (deleteButton.Enabled)
    {
        deleteButton.BackColor = Color.FromArgb(97, 97, 97);
        deleteButton.Invalidate();
    }
};
deleteButton.MouseLeave += (s, e) =>
{
    deleteButton.BackColor = Color.FromArgb(51, 51, 51);
    deleteButton.Invalidate();
};
deleteButton.Click += (s, e) =>
{
    if (valuesList.SelectedItems.Count > 0)
    {
```

```
            DeleteRegistryValue(valuesList.SelectedItems[0]);
        }
    };

    listPanel.Controls.Add(valuesList);
    listPanel.Controls.Add(addButton);
    listPanel.Controls.Add(editButton);
    listPanel.Controls.Add(deleteButton);

    this.Controls.Add(toolbarPanel);
    this.Controls.Add(treePanel);
    this.Controls.Add(listPanel);
}

private void ValuesList_SelectedIndexChanged(object sender, EventArgs e)
{
    editButton.Enabled = valuesList.SelectedItems.Count > 0;
    deleteButton.Enabled = valuesList.SelectedItems.Count > 0;
}

private Button CreateModernButton(string text, Point location, Image icon, EventHandler clickHandler)
{
    var button = new Button
    {
        Text = text,
        Location = location,
        Size = new Size(140, 40),
        FlatStyle = FlatStyle.Flat,
        ForeColor = Color.FromArgb(224, 224, 224),
        Font = new Font("Segoe UI Semibold", 10F),
        TextImageRelation = TextImageRelation.ImageBeforeText,
        Image = icon,
        ImageAlign = ContentAlignment.MiddleLeft,
        TextAlign = ContentAlignment.MiddleCenter,
        Padding = new Padding(30, 0, 0, 0)
    };

    button.BackColor = Color.FromArgb(51, 51, 51);
```

```
button.Paint += (s, e) =>
{
    using (var brush = new SolidBrush(button.BackColor))
    {
        e.Graphics.FillRectangle(brush, button.ClientRectangle);
    }
    ControlPaint.DrawBorder(e.Graphics, button.ClientRectangle, Color.FromArgb(66, 66, 66), ButtonBorderStyle.Solid);
    e.Graphics.DrawString(button.Text, button.Font, new SolidBrush(button.ForeColor), new Rectangle(30, 0, button.Width - 30, button.Height), new StringFormat { Alignment = StringAlignment.Center, LineAlignment = StringAlignment.Center });
    e.Graphics.DrawImage(button.Image, new Rectangle(5, (button.Height - button.Image.Height) / 2, button.Image.Width, button.Image.Height));
};

using (GraphicsPath path = new GraphicsPath())
{
    int radius = 8;
    Rectangle rect = new Rectangle(0, 0, button.Width - 1, button.Height - 1);
    path.AddArc(rect.X, rect.Y, radius, radius, 180, 90);
    path.AddArc(rect.Width - radius, rect.Y, radius, radius, 270, 90);
    path.AddArc(rect.Width - radius, rect.Height - radius, radius, radius, 0, 90);
    path.AddArc(rect.X, rect.Height - radius, radius, radius, 90, 90);
    path.CloseFigure();
    button.Region = new Region(path);
}

button.MouseEnter += (s, e) =>
{
    button.BackColor = Color.FromArgb(97, 97, 97);
    button.Invalidate();
};

button.MouseLeave += (s, e) =>
{
    button.BackColor = Color.FromArgb(51, 51, 51);
    button.Invalidate();
```

```
    };

    button.Click += clickHandler;
    return button;
}

private void TaskManagerButton_Click(object sender, EventArgs e)
{
    TaskManagerForm taskManager = new TaskManagerForm();
    taskManager.Show();
}

private void StartupManagerButton_Click(object sender, EventArgs e)
{
    StartupManagerForm startupManager = new StartupManagerForm();
    startupManager.Show();
}

private void BackupButton_Click(object sender, EventArgs e)
{
    using (var selectionForm = new RegistryBackupSelectionForm())
    {
        if (selectionForm.ShowDialog() == DialogResult.OK)
        {
            using (FolderBrowserDialog folderDialog = new FolderBrowserDialog())
            {
                if (folderDialog.ShowDialog() == DialogResult.OK)
                {
                    string backupPath = folderDialog.SelectedPath;
                    CreateRegistryBackups(backupPath, selectionForm.SelectedHives);
                }
            }
        }
    }
}

private void ProcessManagerButton_Click(object sender, EventArgs e)
{
```

```
        ProcessManagerForm processManager = new ProcessManagerForm();
        processManager.Show();
    }

    private void CleanupButton_Click(object sender, EventArgs e)
    {
        CleanupForm cleanupForm = new CleanupForm();
        cleanupForm.Show();
    }

    private void CreateRegistryBackups(string backupPath, string[] hives)
    {
        string dateStamp = DateTime.Now.ToString("dd_MM_yyyy");

        foreach (string hive in hives)
        {
            string fileName = Path.Combine(backupPath, $"{hive}_{dateStamp}.reg");
            try
            {
                ProcessStartInfo processInfo = new ProcessStartInfo
                {
                    FileName = "reg.exe",
                    Arguments = $"export \"{hive}\" \"{fileName}\" /y",
                    UseShellExecute = false,
                    RedirectStandardOutput = true,
                    RedirectStandardError = true,
                    CreateNoWindow = true
                };

                using (Process process = Process.Start(processInfo))
                {
                    process.WaitForExit();
                    if (process.ExitCode == 0)
                    {
                        MessageBox.Show($"Резервна копія для {hive} успішно створена: {fileName}", "Успіх",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                    }
                    else
```

```
                {
                    string error = process.StandardError.ReadToEnd();
                    MessageBox.Show($"Помилка при створенні резервної копії для {hive}: {error}", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Помилка при створенні резервної копії для {hive}: {ex.Message}", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
}

private void LoadRegistryTree()
{
    AddRootNode("HKEY_LOCAL_MACHINE", Registry.LocalMachine);
    AddRootNode("HKEY_CURRENT_USER", Registry.CurrentUser);
    AddRootNode("HKEY_CLASSES_ROOT", Registry.ClassesRoot);
    AddRootNode("HKEY_USERS", Registry.Users);
    AddRootNode("HKEY_CURRENT_CONFIG", Registry.CurrentConfig);
}

private void AddRootNode(string name, RegistryKey key)
{
    TreeNode node = new TreeNode(name) { ForeColor = Color.FromArgb(224, 224, 224) };
    node.Tag = key;
    node.Nodes.Add(new TreeNode("Loading..."));
    registryTree.Nodes.Add(node);
}

private void RegistryTree_BeforeExpand(object sender, TreeViewCancelEventArgs e)
{
    TreeNode node = e.Node;
    if (node.Nodes.Count == 1 && node.Nodes[0].Text == "Loading...")
    {
        node.Nodes.Clear();
```

```
            LoadSubKeys(node);
        }
    }

    private void LoadSubKeys(TreeNode node)
    {
        try
        {
            RegistryKey key = (RegistryKey)node.Tag;
            string[] subKeyNames = key.GetSubKeyNames();

            foreach (string subKeyName in subKeyNames)
            {
                try
                {
                    TreeNode subNode = new TreeNode(subKeyName) { ForeColor = Color.FromArgb(224, 224,
224) };
                    RegistryKey subKey = key.OpenSubKey(subKeyName);
                    subNode.Tag = subKey;
                    subNode.Nodes.Add(new TreeNode("Loading..."));
                    node.Nodes.Add(subNode);
                }
                catch (Exception)
                {
                    continue;
                }
            }
        }
        catch (Exception)
        {
        }
    }

    private void RegistryTree_AfterSelect(object sender, TreeViewEventArgs e)
    {
        valuesList.Items.Clear();
        RegistryKey key = (RegistryKey)e.Node.Tag;
```

```
        try
        {
            string[] valueNames = key.GetValueNames();

            foreach (string valueName in valueNames)
            {
                try
                {
                    object value = key.GetValue(valueName);
                    RegistryValueKind valueKind = key.GetValueKind(valueName);

                    ListViewItem item = new ListViewItem(valueName.Length > 0 ? valueName : "(За замовчуванням)");
                    item.SubItems.Add(valueKind.ToString());
                    item.SubItems.Add(value?.ToString() ?? "(порожньо)");
                    valuesList.Items.Add(item);
                }
                catch (Exception)
                {
                    continue;
                }
            }
        }
        catch (Exception)
        {
        }
    }

    private void EditRegistryValue(ListViewItem item)
    {
        if (registryTree.SelectedNode == null) return;

        RegistryKey key = (RegistryKey)registryTree.SelectedNode.Tag;
        string valueName = item.Text == "(За замовчуванням)" ? "" : item.Text;
        RegistryValueKind valueKind = (RegistryValueKind)Enum.Parse(typeof(RegistryValueKind), item.SubItems[1].Text);
        string valueData = item.SubItems[2].Text;
```

```
        using (var form = new EditRegistryValueForm(valueName, valueKind, valueData))
        {
            if (form.ShowDialog() == DialogResult.OK)
            {
                try
                {
                    using (RegistryKey writableKey = key.OpenSubKey("", true))
                    {
                        if (writableKey != null)
                        {
                            if (form.ValueName != valueName)
                            {
                                writableKey.DeleteValue(valueName, false);
                            }

                            writableKey.SetValue(form.ValueName, form.ValueData, form.ValueKind);
                            RegistryTree_AfterSelect(null, new TreeViewEventArgs(registryTree.SelectedNode));
                        }
                        else
                        {
                            MessageBox.Show("Недостатньо прав для редагування значення. Запустіть програму від імені адміністратора.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                        }
                    }
                }
                catch (Exception ex)
                {
                    MessageBox.Show($"Помилка при редагуванні значення: {ex.Message}", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }
        }

        private void DeleteRegistryValue(ListViewItem item)
        {
            if (registryTree.SelectedNode == null) return;
```

```
        string valueName = item.Text == "(За замовчуванням)" ? "" : item.Text;
        if (MessageBox.Show($"Ви впевнені, що хочете видалити параметр '{item.Text}'?", "Підтвердження", MessageBoxButtons.YesNo, MessageBoxIcon.Warning) == DialogResult.Yes)
        {
            try
            {
                RegistryKey key = (RegistryKey)registryTree.SelectedNode.Tag;
                using (RegistryKey writableKey = key.OpenSubKey("", true))
                {
                    if (writableKey != null)
                    {
                        writableKey.DeleteValue(valueName, false);
                        RegistryTree_AfterSelect(null, new TreeViewEventArgs(registryTree.SelectedNode));
                        MessageBox.Show($"Параметр '{item.Text}' успішно видалено.", "Успіх", MessageBoxButtons.OK, MessageBoxIcon.Information);
                    }
                    else
                    {
                        MessageBox.Show("Недостатньо прав для видалення значення. Запустіть програму від імені адміністратора.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                    }
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Помилка при видаленні значення: {ex.Message}", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
    }

    private void AddNewRegistryValue()
    {
        if (registryTree.SelectedNode == null)
        {
            MessageBox.Show("Спочатку виберіть ключ у дереві реєстру.", "Попередження", MessageBoxButtons.OK, MessageBoxIcon.Warning);
            return;
```

```
        }

        using (var form = new EditRegistryValueForm())
        {
            if (form.ShowDialog() == DialogResult.OK)
            {
                try
                {
                    RegistryKey key = (RegistryKey)registryTree.SelectedNode.Tag;
                    using (RegistryKey writableKey = key.OpenSubKey("", true))
                    {
                        if (writableKey != null)
                        {
                            writableKey.SetValue(form.ValueName, form.ValueData, form.ValueKind);
                            RegistryTree_AfterSelect(null, new TreeViewEventArgs(registryTree.SelectedNode));
                            MessageBox.Show($"Параметр '{form.ValueName}' успішно додано.", "Успіх",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                        }
                        else
                        {
                            MessageBox.Show("Недостатньо прав для створення значення. Запустіть програму від
імені адміністратора.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                        }
                    }
                }
                catch (Exception ex)
                {
                    MessageBox.Show($"Помилка при додаванні значення: {ex.Message}", "Помилка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }
        }
    }

    public class RegistryBackupSelectionForm : Form
    {
        private CheckBox[] hiveCheckBoxes;
```

```
    private Button saveButton;
    private Button cancelButton;
    private Panel contentPanel;

    public string[] SelectedHives { get; private set; }

    public RegistryBackupSelectionForm()
    {
        InitializeComponents();
    }

    private void InitializeComponents()
    {
        this.Text = "Вибір ключів для резервної копії";
        this.Width = 400;
        this.Height = 300;
        this.FormBorderStyle = FormBorderStyle.FixedSingle;
        this.MaximizeBox = false;
        this.BackColor = Color.FromArgb(33, 33, 33);
        this.DoubleBuffered = true;
        this.StartPosition = FormStartPosition.CenterParent;

        contentPanel = new Panel
        {
            Location = new Point(10, 10),
            Size = new Size(this.ClientSize.Width - 20, this.ClientSize.Height - 20),
            BorderStyle = BorderStyle.None,
            BackColor = Color.FromArgb(46, 46, 46)
        };
        contentPanel.Paint += (s, e) =>
        {
            ControlPaint.DrawBorder(e.Graphics, contentPanel.ClientRectangle, Color.FromArgb(66, 66, 66),
ButtonBorderStyle.Solid);
        };

        string[] hiveNames = { "HKEY_CLASSES_ROOT", "HKEY_CURRENT_USER",
"HKEY_LOCAL_MACHINE", "HKEY_USERS", "HKEY_CURRENT_CONFIG" };
        hiveCheckBoxes = new CheckBox[hiveNames.Length];
```

```
for (int i = 0; i < hiveNames.Length; i++)
{
    hiveCheckBoxes[i] = new CheckBox
    {
        Text = hiveNames[i],
        Location = new Point(10, 10 + (i * 30)),
        Size = new Size(300, 30),
        ForeColor = Color.FromArgb(224, 224, 224),
        Font = new Font("Segoe UI", 9.5F),
        BackColor = Color.FromArgb(46, 46, 46),
        Checked = true
    };
    contentPanel.Controls.Add(hiveCheckBoxes[i]);
}

saveButton = new Button
{
    Text = "Зберегти",
    Location = new Point(10, 170),
    Size = new Size(100, 30),
    FlatStyle = FlatStyle.Flat,
    ForeColor = Color.FromArgb(224, 224, 224),
    Font = new Font("Segoe UI Semibold", 9F),
    BackColor = Color.FromArgb(51, 51, 51)
};
saveButton.MouseEnter += (s, e) =>
{
    saveButton.BackColor = Color.FromArgb(97, 97, 97);
    saveButton.Invalidate();
};
saveButton.MouseLeave += (s, e) =>
{
    saveButton.BackColor = Color.FromArgb(51, 51, 51);
    saveButton.Invalidate();
};
saveButton.Click += SaveButton_Click;
```

```
        cancelButton = new Button
        {
            Text = "Скасувати",
            Location = new Point(contentPanel.Width - 110, 170),
            Size = new Size(100, 30),
            FlatStyle = FlatStyle.Flat,
            ForeColor = Color.FromArgb(224, 224, 224),
            Font = new Font("Segoe UI Semibold", 9F),
            BackColor = Color.FromArgb(51, 51, 51)
        };
        cancelButton.MouseEnter += (s, e) =>
        {
            cancelButton.BackColor = Color.FromArgb(97, 97, 97);
            cancelButton.Invalidate();
        };
        cancelButton.MouseLeave += (s, e) =>
        {
            cancelButton.BackColor = Color.FromArgb(51, 51, 51);
            cancelButton.Invalidate();
        };
        cancelButton.Click += (s, e) => this.DialogResult = DialogResult.Cancel;

        contentPanel.Controls.Add(saveButton);
        contentPanel.Controls.Add(cancelButton);
        this.Controls.Add(contentPanel);
    }

    private void SaveButton_Click(object sender, EventArgs e)
    {
        SelectedHives = hiveCheckBoxes
            .Where(cb => cb.Checked)
            .Select(cb => cb.Text)
            .ToArray();

        if (SelectedHives.Length == 0)
        {
            MessageBox.Show("Виберіть хоча б один ключ реєстру.", "Попередження",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
```

```
            return;
        }

        this.DialogResult = DialogResult.OK;
    }
}

public class EditRegistryValueForm : Form
{
    private TextBox nameTextBox;
    private ComboBox typeComboBox;
    private TextBox valueTextBox;
    private Button saveButton;
    private Button cancelButton;
    private string initialName;

    public string ValueName => nameTextBox.Text;
    public RegistryValueKind ValueKind => (RegistryValueKind)Enum.Parse(typeof(RegistryValueKind), typeComboBox.SelectedItem.ToString());
    public object ValueData => ParseValueData(valueTextBox.Text, ValueKind);

    public EditRegistryValueForm(string name = "", RegistryValueKind kind = RegistryValueKind.String, string value = "")
    {
        initialName = name;
        InitializeComponents(name, kind, value);
    }

    private void InitializeComponents(string name, RegistryValueKind kind, string value)
    {
        this.Text = string.IsNullOrEmpty(name) ? "Додати новий параметр" : "Редагувати параметр";
        this.Width = 400;
        this.Height = 300;
        this.FormBorderStyle = FormBorderStyle.FixedSingle;
        this.MaximizeBox = false;
        this.BackColor = Color.FromArgb(33, 33, 33);
        this.DoubleBuffered = true;
        this.StartPosition = FormStartPosition.CenterParent;
```

```
        var panel = new Panel
        {
            Location = new Point(10, 10),
            Size = new Size(this.ClientSize.Width - 20, this.ClientSize.Height - 20),
            BorderStyle = BorderStyle.None,
            BackColor = Color.FromArgb(46, 46, 46)
        };
        panel.Paint += (s, e) =>
        {
            ControlPaint.DrawBorder(e.Graphics, panel.ClientRectangle, Color.FromArgb(66, 66, 66),
ButtonBorderStyle.Solid);
        };

        var nameLabel = new Label
        {
            Text = "Назва:",
            Location = new Point(10, 10),
            Size = new Size(100, 20),
            ForeColor = Color.FromArgb(224, 224, 224),
            Font = new Font("Segoe UI", 9.5F)
        };

        nameTextBox = new TextBox
        {
            Text = name,
            Location = new Point(10, 35),
            Size = new Size(panel.Width - 20, 25),
            BackColor = Color.FromArgb(66, 66, 66),
            ForeColor = Color.FromArgb(224, 224, 224),
            BorderStyle = BorderStyle.FixedSingle,
            Font = new Font("Segoe UI", 9.5F)
        };

        var typeLabel = new Label
        {
            Text = "Тип:",
            Location = new Point(10, 70),
```

```
    Size = new Size(100, 20),
    ForeColor = Color.FromArgb(224, 224, 224),
    Font = new Font("Segoe UI", 9.5F)
};

typeComboBox = new ComboBox
{
    Location = new Point(10, 95),
    Size = new Size(panel.Width - 20, 25),
    BackColor = Color.FromArgb(66, 66, 66),
    ForeColor = Color.FromArgb(224, 224, 224),
    Font = new Font("Segoe UI", 9.5F),
    DropDownStyle = ComboBoxStyle.DropDownList
};
typeComboBox.Items.AddRange(Enum.GetNames(typeof(RegistryValueKind)));
typeComboBox.SelectedItem = kind.ToString();

var valueLabel = new Label
{
    Text = "Значення:",
    Location = new Point(10, 130),
    Size = new Size(100, 20),
    ForeColor = Color.FromArgb(224, 224, 224),
    Font = new Font("Segoe UI", 9.5F)
};

valueTextBox = new TextBox
{
    Text = value,
    Location = new Point(10, 155),
    Size = new Size(panel.Width - 20, 25),
    BackColor = Color.FromArgb(66, 66, 66),
    ForeColor = Color.FromArgb(224, 224, 224),
    BorderStyle = BorderStyle.FixedSingle,
    Font = new Font("Segoe UI", 9.5F)
};

saveButton = new Button
```

```
{
    Text = "Зберегти",
    Location = new Point(10, 200),
    Size = new Size(100, 30),
    FlatStyle = FlatStyle.Flat,
    ForeColor = Color.FromArgb(224, 224, 224),
    Font = new Font("Segoe UI Semibold", 9F),
    BackColor = Color.FromArgb(51, 51, 51)
};
saveButton.MouseEnter += (s, e) =>
{
    saveButton.BackColor = Color.FromArgb(97, 97, 97);
    saveButton.Invalidate();
};
saveButton.MouseLeave += (s, e) =>
{
    saveButton.BackColor = Color.FromArgb(51, 51, 51);
    saveButton.Invalidate();
};
saveButton.Click += (s, e) => this.DialogResult = DialogResult.OK;

cancelButton = new Button
{
    Text = "Скасувати",
    Location = new Point(panel.Width - 110, 200),
    Size = new Size(100, 30),
    FlatStyle = FlatStyle.Flat,
    ForeColor = Color.FromArgb(224, 224, 224),
    Font = new Font("Segoe UI Semibold", 9F),
    BackColor = Color.FromArgb(51, 51, 51)
};
cancelButton.MouseEnter += (s, e) =>
{
    cancelButton.BackColor = Color.FromArgb(97, 97, 97);
    cancelButton.Invalidate();
};
cancelButton.MouseLeave += (s, e) =>
{
```

```
        cancelButton.BackColor = Color.FromArgb(51, 51, 51);
        cancelButton.Invalidate();
    };
    cancelButton.Click += (s, e) => this.DialogResult = DialogResult.Cancel;

    panel.Controls.Add(nameLabel);
    panel.Controls.Add(nameTextBox);
    panel.Controls.Add(typeLabel);
    panel.Controls.Add(typeComboBox);
    panel.Controls.Add(valueLabel);
    panel.Controls.Add(valueTextBox);
    panel.Controls.Add(saveButton);
    panel.Controls.Add(cancelButton);
    this.Controls.Add(panel);
}

private object ParseValueData(string value, RegistryValueKind kind)
{
    try
    {
        switch (kind)
        {
            case RegistryValueKind.String:
            case RegistryValueKind.ExpandString:
                return value;
            case RegistryValueKind.DWord:
                return int.Parse(value);
            case RegistryValueKind.QWord:
                return long.Parse(value);
            case RegistryValueKind.MultiString:
                return value.Split(new[] { Environment.NewLine }, StringSplitOptions.None);
            case RegistryValueKind.Binary:
                return value.Split(',').Select(b => byte.Parse(b.Trim())).ToArray();
            default:
                return value;
        }
    }
    catch (Exception ex)
```

```
        {
            throw new Exception($"Помилка при парсингу значення: {ex.Message}");
        }
    }
}

public class TaskManagerForm : Form
{
    private Label cpuLabel;
    private Label ramLabel;
    private ProgressBar cpuProgressBar;
    private ProgressBar ramProgressBar;
    private System.Windows.Forms.Timer updateTimer;
    private PerformanceCounter cpuCounter;
    private Panel contentPanel;

    public TaskManagerForm()
    {
        InitializeComponents();
        InitializePerformanceCounters();
    }

    private void InitializeComponents()
    {
        this.Text = "Диспетчер завдань";
        this.Width = 450;
        this.Height = 200;
        this.FormBorderStyle = FormBorderStyle.FixedSingle;
        this.MaximizeBox = false;
        this.BackColor = Color.FromArgb(33, 33, 33);
        this.DoubleBuffered = true;

        contentPanel = new Panel
        {
            Location = new Point(10, 10),
            Size = new Size(this.ClientSize.Width - 20, this.ClientSize.Height - 20),
            BorderStyle = BorderStyle.None,
            BackColor = Color.FromArgb(46, 46, 46)
```

```
        };
        contentPanel.Paint += (s, e) =>
        {
            ControlPaint.DrawBorder(e.Graphics, contentPanel.ClientRectangle, Color.FromArgb(66, 66, 66),
ButtonBorderStyle.Solid);
        };

        var cpuPanel = new Panel
        {
            Location = new Point(10, 10),
            Size = new Size(contentPanel.Width - 20, 60),
            BorderStyle = BorderStyle.None,
            BackColor = Color.FromArgb(46, 46, 46)
        };
        cpuPanel.Paint += (s, e) =>
        {
            ControlPaint.DrawBorder(e.Graphics, cpuPanel.ClientRectangle, Color.FromArgb(66, 66, 66),
ButtonBorderStyle.Solid);
        };

        cpuProgressBar.Region = new Region(new GraphicsPath().AddRoundRect(new Rectangle(0, 0,
cpuProgressBar.Width - 1, cpuProgressBar.Height - 1), 8));
        cpuProgressBar.Paint += (s, e) =>
        {
            var rect = cpuProgressBar.ClientRectangle;
            rect.Inflate(-2, -2);
            float percentage = cpuProgressBar.Value / (float)cpuProgressBar.Maximum;
            int width = (int)(rect.Width * percentage);
            using (var brush = new SolidBrush(Color.FromArgb(176, 176, 176)))
            {
                e.Graphics.FillRectangle(brush, 2, 2, width, rect.Height);
            }
            using (var brush = new SolidBrush(Color.FromArgb(66, 66, 66)))
            {
                e.Graphics.FillRectangle(brush, width + 2, 2, rect.Width - width, rect.Height);
            }
        };
```

```
        cpuPanel.Controls.Add(cpuLabel);
        cpuPanel.Controls.Add(cpuProgressBar);

        var ramPanel = new Panel
        {
            Location = new Point(10, 80),
            Size = new Size(contentPanel.Width - 20, 60),
            BorderStyle = BorderStyle.None,
            BackColor = Color.FromArgb(46, 46, 46)
        };
        ramPanel.Paint += (s, e) =>
        {
            ControlPaint.DrawBorder(e.Graphics, ramPanel.ClientRectangle, Color.FromArgb(66, 66, 66),
ButtonBorderStyle.Solid);
        };

        ramLabel = new Label
        {
            Location = new Point(10, 10),
            Size = new Size(ramPanel.Width - 20, 20),
            Font = new Font("Segoe UI Semibold", 10F),
            ForeColor = Color.FromArgb(224, 224, 224)
        };

        ramProgressBar = new ProgressBar
        {
            Location = new Point(10, 35),
            Size = new Size(ramPanel.Width - 20, 15),
            Maximum = 100,
            Style = ProgressBarStyle.Continuous
        };
        ramProgressBar.Region = new Region(new GraphicsPath().AddRoundRect(new Rectangle(0, 0,
ramProgressBar.Width - 1, ramProgressBar.Height - 1), 8));
        ramProgressBar.Paint += (s, e) =>
        {
            var rect = ramProgressBar.ClientRectangle;
            rect.Inflate(-2, -2);
            float percentage = ramProgressBar.Value / (float)ramProgressBar.Maximum;
```

```
            int width = (int)(rect.Width * percentage);
            using (var brush = new SolidBrush(Color.FromArgb(176, 176, 176)))
            {
                e.Graphics.FillRectangle(brush, 2, 2, width, rect.Height);
            }
            using (var brush = new SolidBrush(Color.FromArgb(66, 66, 66)))
            {
                e.Graphics.FillRectangle(brush, width + 2, 2, rect.Width - width, rect.Height);
            }
        };

        ramPanel.Controls.Add(ramLabel);
        ramPanel.Controls.Add(ramProgressBar);

        updateTimer = new System.Windows.Forms.Timer();
        updateTimer.Interval = 1000;
        updateTimer.Tick += UpdateTimer_Tick;
        updateTimer.Start();

        contentPanel.Controls.Add(cpuPanel);
        contentPanel.Controls.Add(ramPanel);
        this.Controls.Add(contentPanel);
    }

    private void InitializePerformanceCounters()
    {
        cpuCounter = new PerformanceCounter("Processor", "% Processor Time", "_Total");
    }

    private void UpdateTimer_Tick(object sender, EventArgs e)
    {
        float cpuUsage = cpuCounter.NextValue();
        cpuLabel.Text = $"Використання CPU: {cpuUsage:F1}%";
        cpuProgressBar.Value = (int)cpuUsage;

        var computerInfo = new ComputerInfo();
        float totalRam = computerInfo.TotalPhysicalMemory / (1024f * 1024f * 1024f);
        float availableRam = computerInfo.AvailablePhysicalMemory / (1024f * 1024f * 1024f);
```

```
        float usedRam = totalRam - availableRam;
        float ramUsagePercent = (usedRam / totalRam) * 100;
        ramLabel.Text = $"Використання RAM: {usedRam:F2} GB з {totalRam:F2} GB ({ramUsagePercent:F1}%)";
        ramProgressBar.Value = (int)ramUsagePercent;
    }

    protected override void OnClosed(EventArgs e)
    {
        updateTimer.Stop();
        cpuCounter.Dispose();
        base.OnClosed(e);
    }
}

    private void InitializeComponents()
    {
        this.Text = "Менеджер автозапуску";
        this.Width = 650;
        this.Height = 450;
        this.FormBorderStyle = FormBorderStyle.FixedSingle;
        this.MaximizeBox = false;
        this.BackColor = Color.FromArgb(33, 33, 33);
        this.DoubleBuffered = true;

        contentPanel = new Panel
        {
            Location = new Point(10, 10),
            Size = new Size(this.ClientSize.Width - 20, this.ClientSize.Height - 20),
            BorderStyle = BorderStyle.None,
            BackColor = Color.FromArgb(46, 46, 46)
        };
        contentPanel.Paint += (s, e) =>
        {
            ControlPaint.DrawBorder(e.Graphics, contentPanel.ClientRectangle, Color.FromArgb(66, 66, 66), ButtonBorderStyle.Solid);
        };
```

```
            startupList = new ListView
            {
                Location = new Point(5, 5),
                Size = new Size(contentPanel.Width - 10, contentPanel.Height - 10),
                BackColor = Color.FromArgb(46, 46, 46),
                ForeColor = Color.FromArgb(224, 224, 224),
                BorderStyle = BorderStyle.None,
                View = View.Details,
                FullRowSelect = true,
                GridLines = true,
                Font = new Font("Segoe UI", 9.5F),
                CheckBoxes = true
            };
            startupList.Columns.Add("Назва", 180);
            startupList.Columns.Add("Шлях", 320);
            startupList.Columns.Add("Статус", 120);
            startupList.ItemCheck += StartupList_ItemCheck;

            contentPanel.Controls.Add(startupList);
            this.Controls.Add(contentPanel);
        }

    class Program
    {
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.Run(new RegistryEditorForm());
        }
    }
}
```

Додаток Д – Ілюстрований матеріал

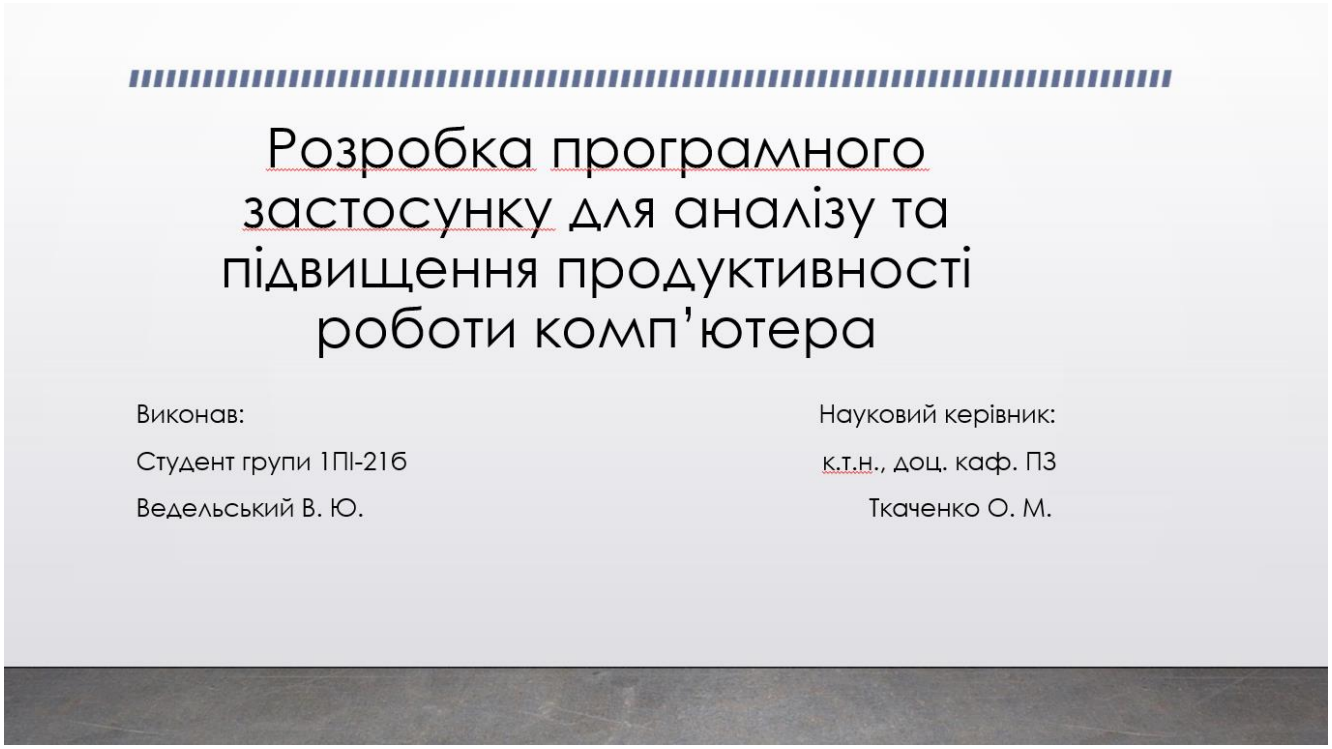


Рисунок Д.1 – Титульний слайд

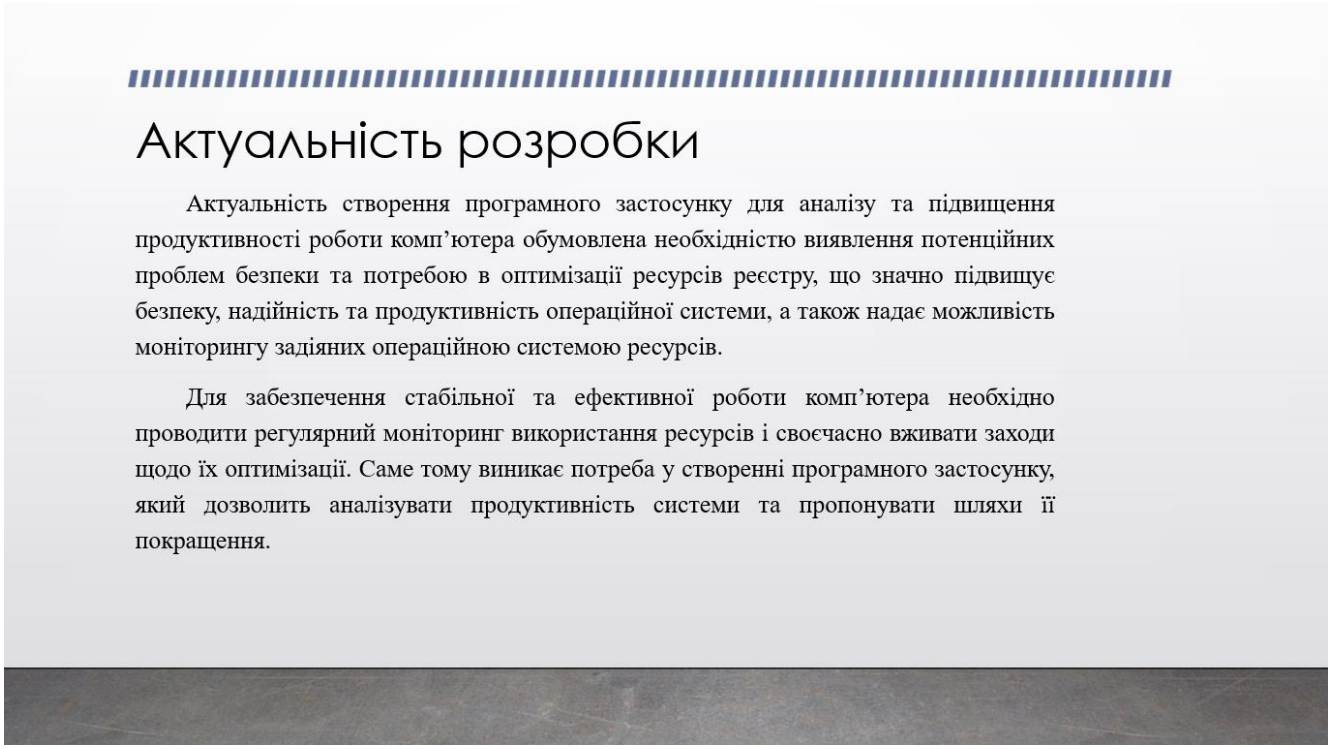


Рисунок Д.2 – Актуальність розробки

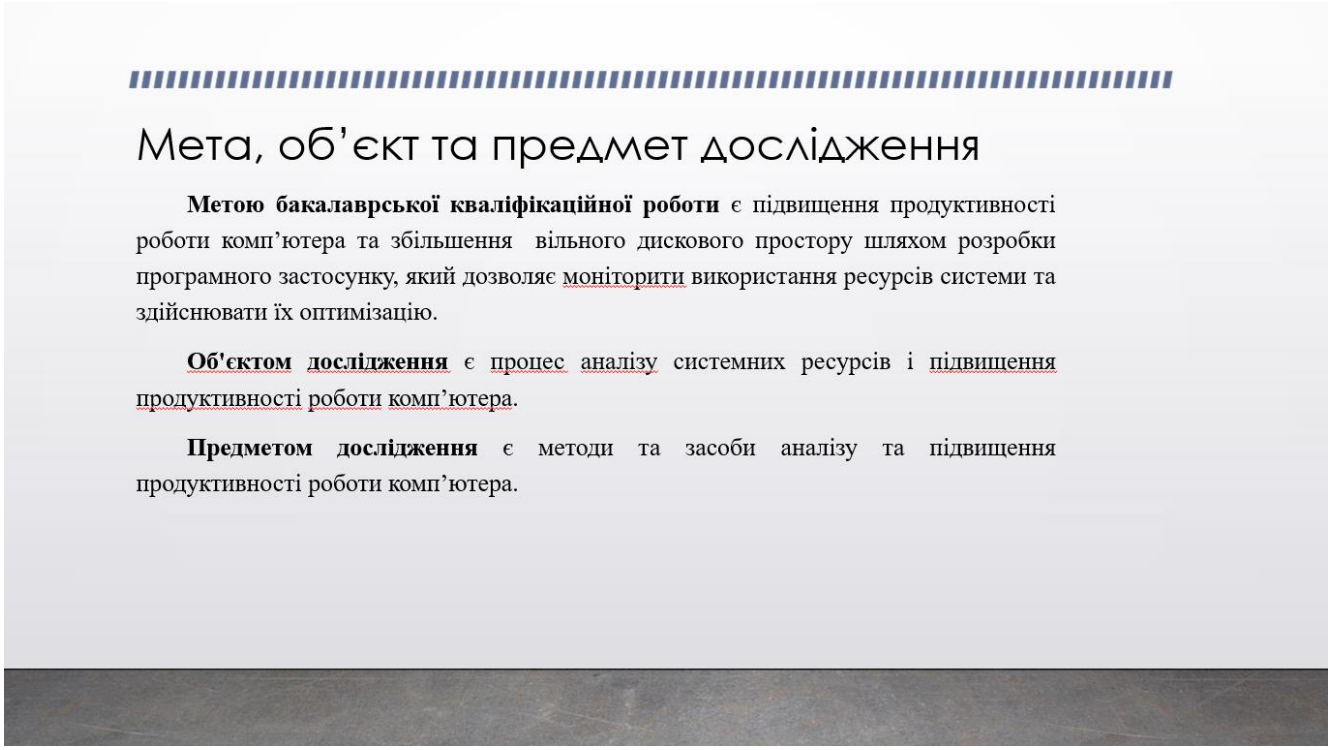


Рисунок Д.3 – Мета, об’єкт та предмет дослідження

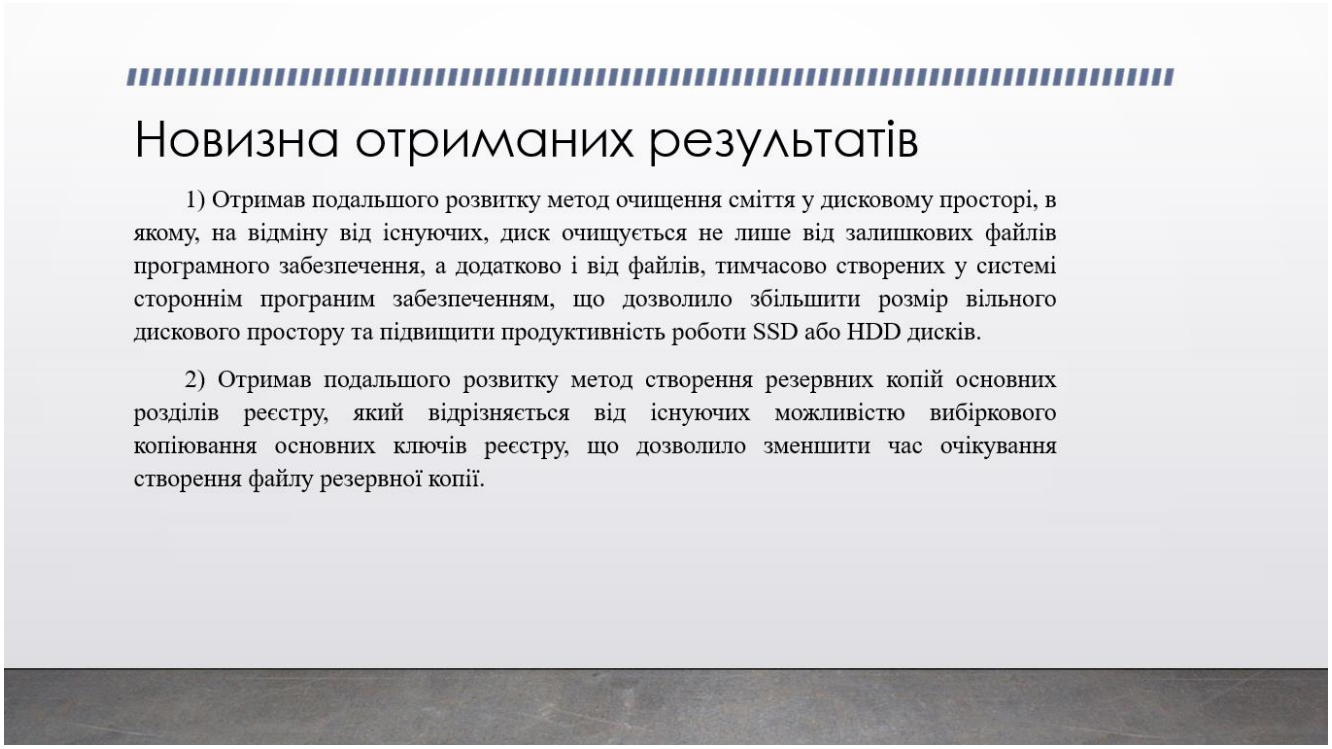


Рисунок Д.4 – Новизна отриманих результатів

Практичне значення отриманих результатів

Практичне значення роботи полягає у створенні програмного забезпечення, яке дозволить моніторити завантаженість операційної системи, а також вживати заходів щодо підвищення її продуктивності. Розроблені програмні модулі можуть бути використані як у вигляді окремої утиліти, так і у складі інших системних застосунків

Рисунок Д.5 – Практичне значення отриманих результатів

Аналіз аналогів

- Розроблюване програмне забезпечення є комплексним рішенням, яке поєднує в собі переваги аналогів, позбавлене деяких їх недоліків, а також має розширений функціонал.

Критерій	CCleaner	Auslogics Registry Cleaner	Windows Scanreg	Reg Organizer	Власна розробка
Очистка сміття	+	-	-	+	+
Менеджер автозапуску	+	+	-	-	+
Менеджер процесів	-	-	-	+	+
Моніторинг задіяних ресурсів	-	+	+	-	+
Резервне копіювання	+	-	+	+	+
Сумарний коефіцієнт	60%	40%	40%	60%	100%

Рисунок Д.6 – Аналіз аналогів

UML-діаграма діяльності програмного застосунку

Дана UML-діаграма показує найбільш ефективний алгоритм дій у програмному застосунку, виконання якого гарантує безпеку користування програмним забезпеченням та надає доступ до усього функціоналу програми.

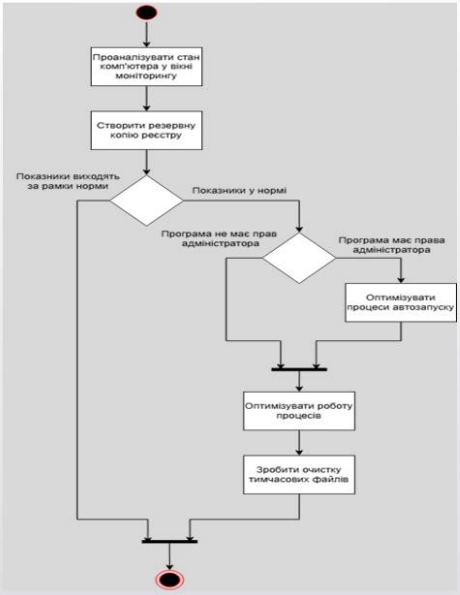


Рисунок Д.7 – UML-діаграма діяльності програмного застосунку

Блок-схема алгоритму роботи застосунку

Дана блок-схема показує загальний принцип роботи програмного застосунку.

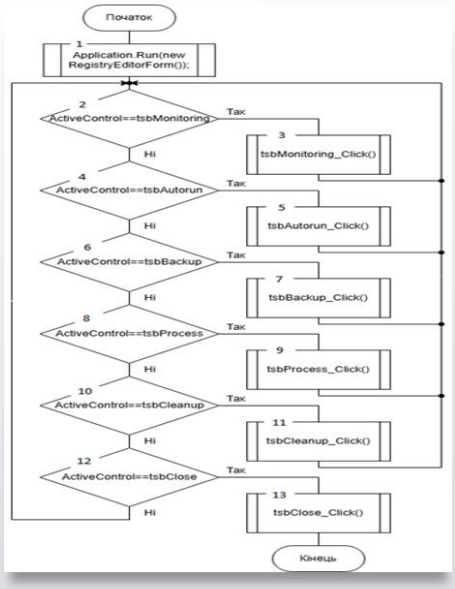


Рисунок Д.8 – Блок-схема алгоритму роботи застосунку

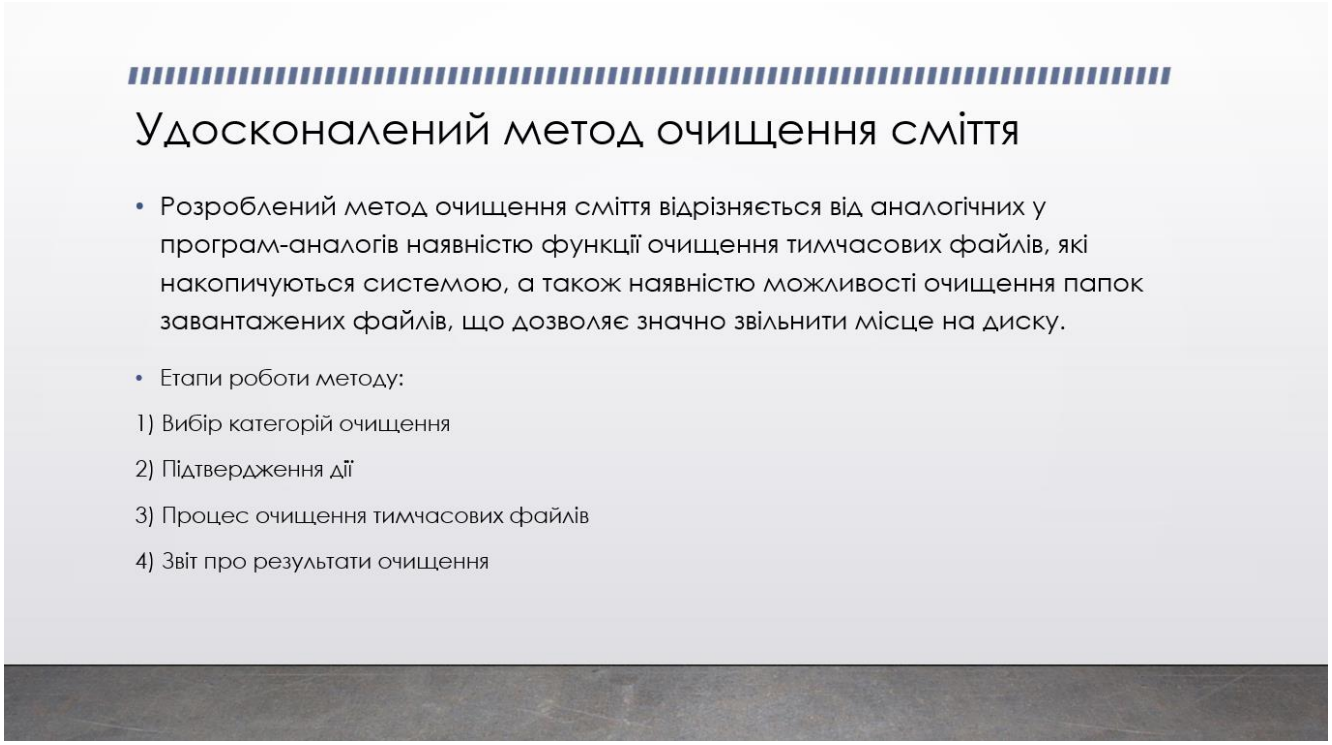


Рисунок Д.9 – Удосконалений метод очищення сміття

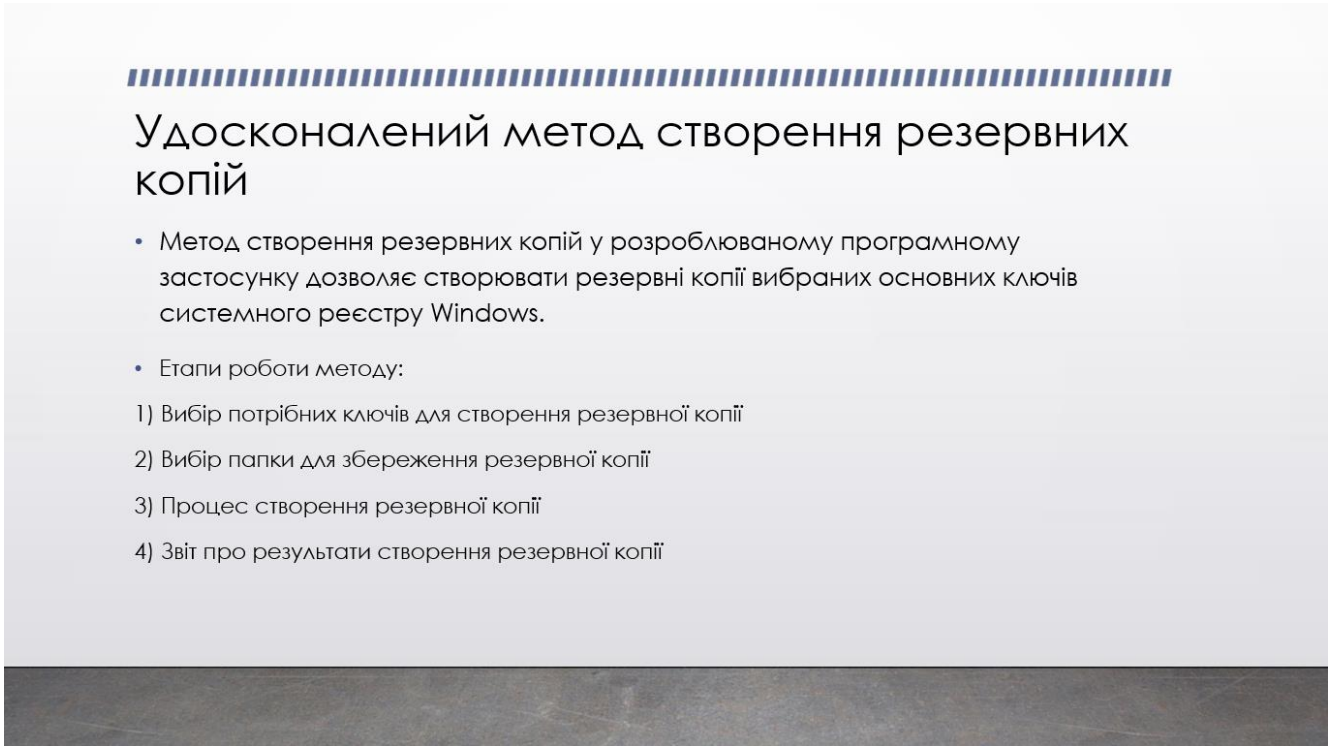


Рисунок Д.10 – Удосконалений метод створення резервних копій

Інтерфейс застосунку

На головному вікні програми розміщено навігаційну панель з кнопками, які потрібні для переходу між вікнами програмного застосунку, також показано відображення ключів реєстру у вигляді дерева, а також список, який буде відображати інформацію про параметри у ключах реєстру при натисненні на них, а також можливість редагування, видалення та додавання параметрів у обраних ключах в реєстрі.

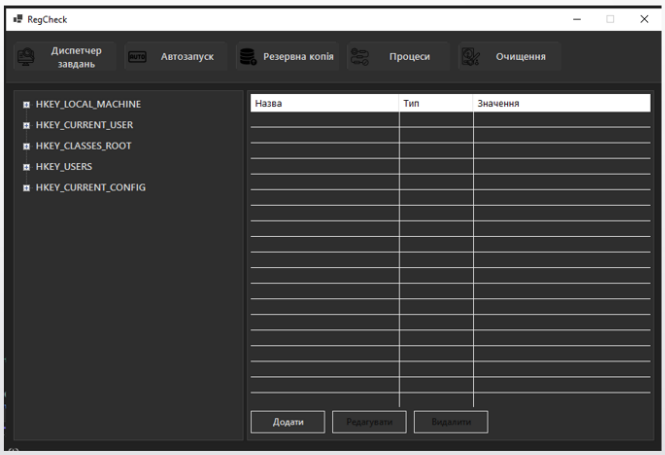


Рисунок Д.11 – Інтерфейс застосунку

Підсумок зростання продуктивності

На цій діаграмі показано підсумок зростання продуктивності низки важливих показників продуктивності роботи комп'ютера.

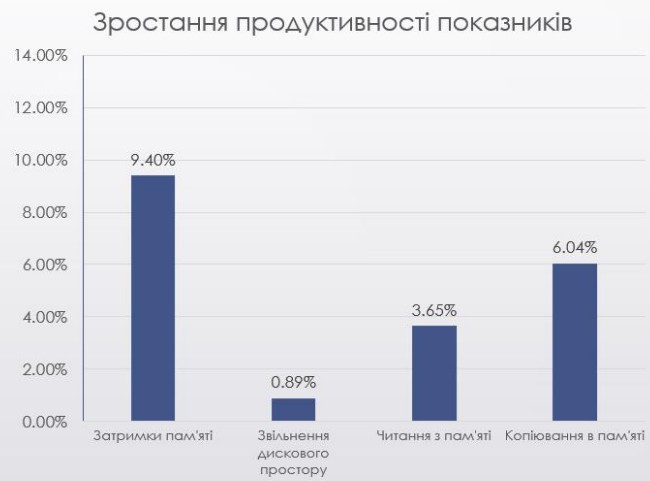


Рисунок Д.12 – Підсумок зростання продуктивності

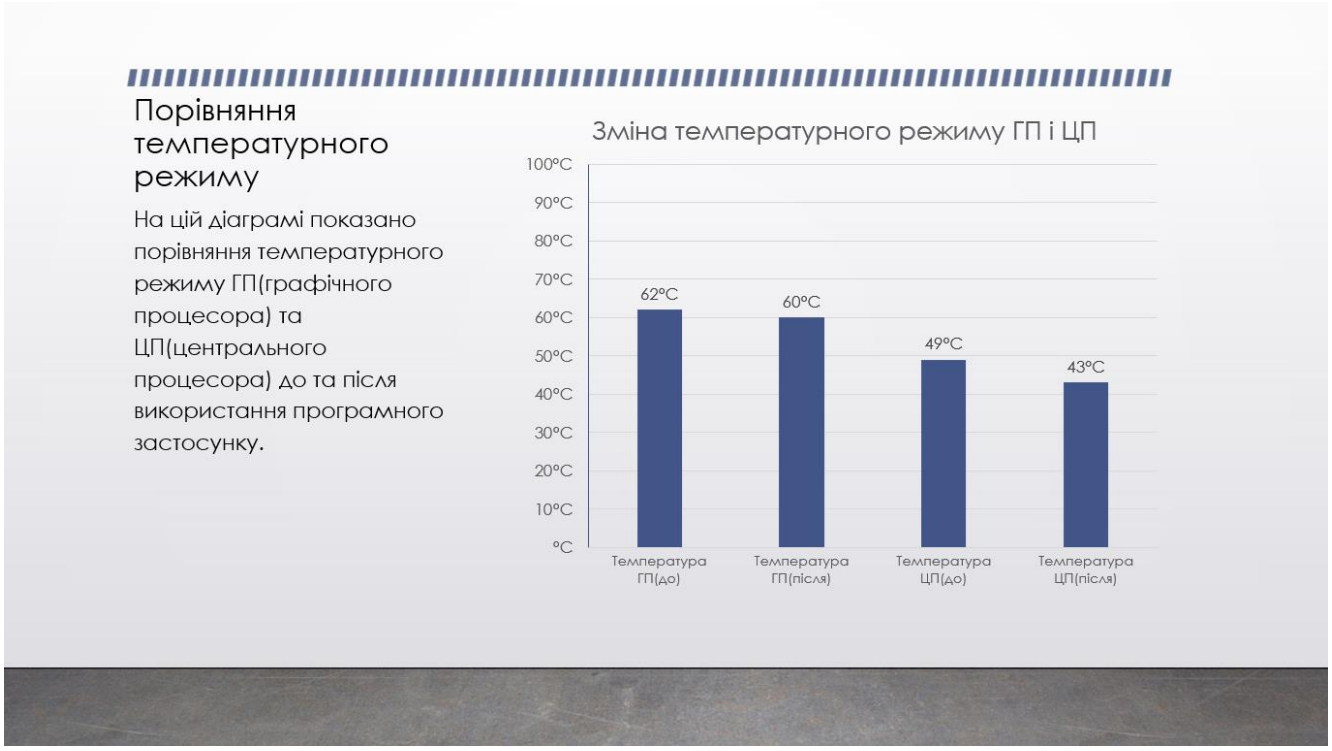


Рисунок Д.13 – Порівняння температурного режиму

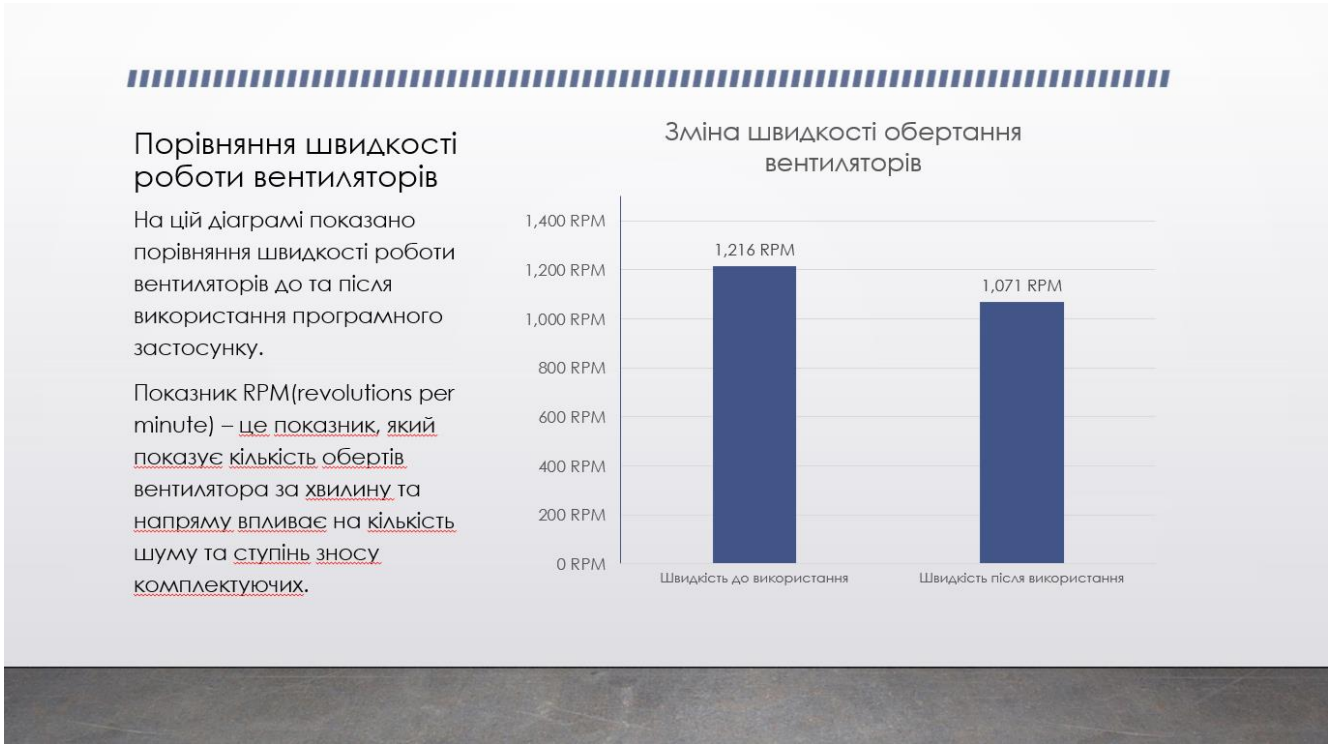


Рисунок Д.14 – Порівняння швидкості роботи вентиляторів

Апробація та публікації результатів роботи

Результати роботи було представлено на LIV Всеукраїнській науково-технічній конференції підрозділів Вінницького національного технічного університету (Вінниця, 2025 р.) і XIII Міжнародній науково-практичній конференції з інформаційних систем та технологій Infocom Advanced Solutions 2025 (Київ, 2025 р.).

Публікації.

1. Ведельський В.Ю. Програмний застосунок для аналізу та підвищення продуктивності роботи комп'ютера. LIV Всеукраїнська науково-технічна конференція факультету інформаційних технологій та комп'ютерної інженерії Вінниця: ВНТУ, 2025.
2. Ведельський В.Ю. Розробка методу резервного копіювання. XIII Міжнародна науково-практична конференція з інформаційних систем та технологій Infocom Advanced Solutions 2025. Київ: КПІ, 2025.

Рисунок Д.15 – Апробація та публікація результатів роботи

Акт впровадження

Програмне забезпечення було використано на підприємстві, що підтверджено актом впровадження.

Акт впровадження

УЗГОДЖЕНО
Директор ТОВ «Агро Поділля і К.»
Черваньов В.В.
«____» _____ 2025 року

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
д.т.н., професор Романюк О.Н.
«__» _____ 2025 року

АКТ ВПРОВАДЖЕННЯ №___
результатів науково-дослідних робіт

Замовник: ТОВ «Агро Поділля і К.»
Цим актом підтверджується, що результати роботи – «Розробка програмного забезпечення для аналізу та підвищення продуктивності роботи комп'ютера», що виконав студент гр. 1ПІ-21б ВНТУ Ведельський В.Ю. за договором про творчу співдружність без взаємних грошових розрахунків на громадських засадах відповідно до теми, затвердженої наказом № __ від __ березня 2025р., виконано з _____ по _____.
Та впроваджено у ТОВ «Агро Поділля і К.»
1. Вид впроваджених результатів: експлуатація програмного забезпечення
2. Характеристика масштабу впровадження: одиничне
3. Форма впровадження: дослідний зразок
4. Новизна результатів науково-дослідної роботи: якісно нові
5. Впроваджені: в системі аналізу продуктивності обладнання
6. Соціальний та науково-технічний ефект: спеціальне призначення

Від виконавця:
студент групи 1ПІ-216
_____ Ведельський В.Ю.

Від ТОВ «Агро Поділля і К.»:
директор
_____ Черваньов В.В.

Керівник: к.т.н., доц. кафедри ПЗ
_____ Ткаченко О. М.

Рисунок Д.16 – Акт впровадження

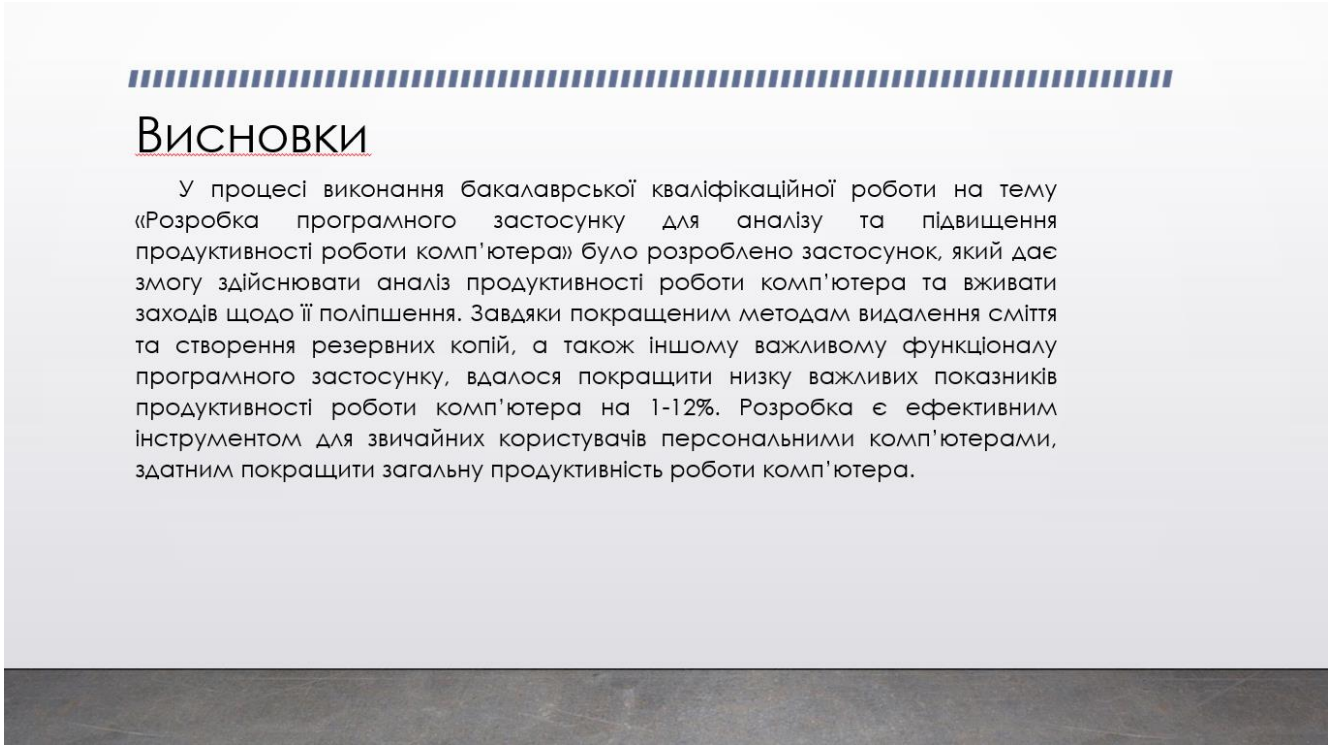


Рисунок Д.17 – Висновки

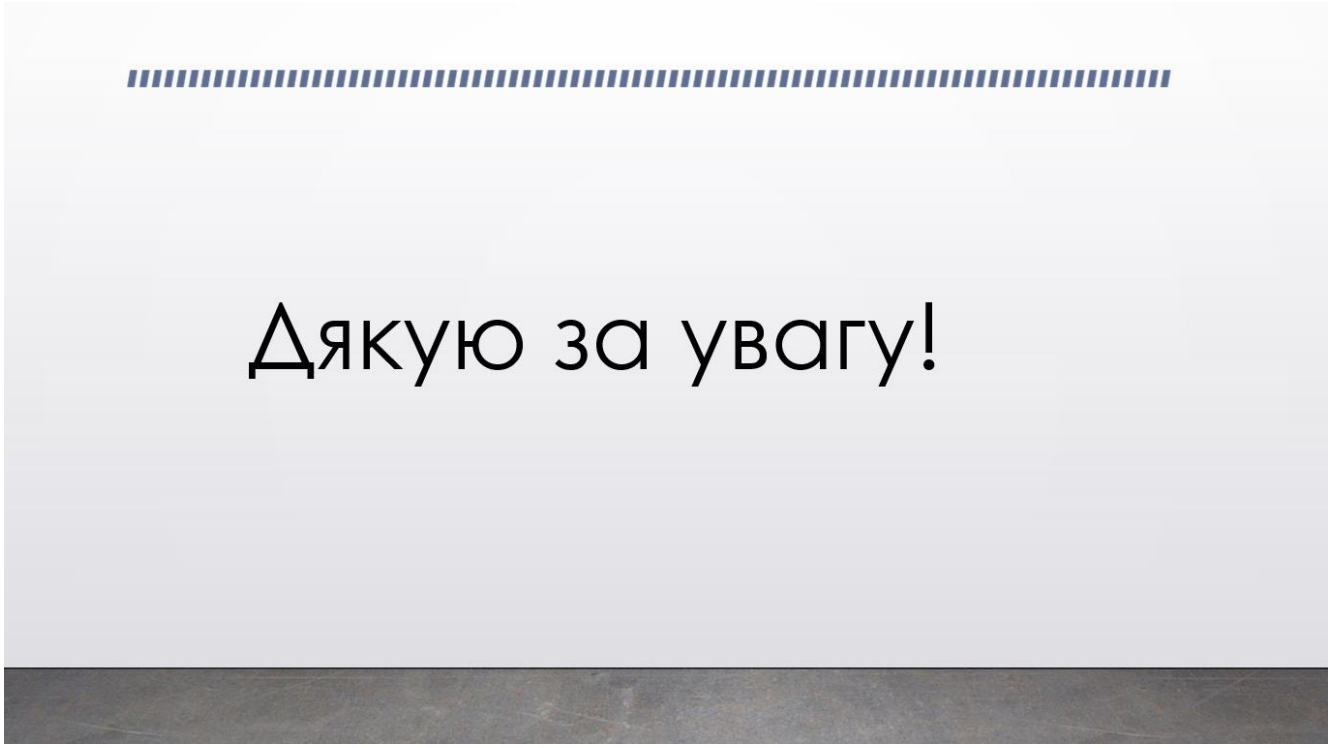


Рисунок Д.18 – Фінальний слайд