

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

ДЛЯ ШИФРУВАННЯ ТА АРХІВАЦІЇ ФАЙЛІВ

SOFTWARE FOR FILE ENCRYPTION AND ARCHIVING

Виконав(ла): студент(ка) 4 курсу, групи КНТ-112

Спеціальності 121 Інженерія програмного

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

КРОТ О.О.

(ПРИЗВИЩЕ та ініціали)

Керівник ШИТКОВА О.В.

(ПРИЗВИЩЕ та ініціали)

Рецензент КОРОТКИЙ О.В.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
“ ” 2026 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

КРОТА Олексія Олександровича

(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення для шифрування та архівації файлів. Software for File Encryption and Archiving

керівник проєкту (роботи) к.т.н., доцент, Шитікова Олена Вікторівна,
(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “ 7 ” квітня 2026 року № 139

2. Строк подання студентом проєкту (роботи) 03 червня 2026 року

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

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

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів) _____

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ШИТІКОВА О.В., доцент		
Нормоконтроль	ДЕЙНЕГА Л.Ю., ст. викладач		

7. Дата видачі завдання “ 7 ” квітня 2026 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка структури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділи 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

(підпис) Олексій КРОТ
(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Олена ШИТІКОВА
(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
86 с., 3 табл., 24 рис., 3 дод., 19 джерел.

C++, VISUAL STUDIO, АРХІВАЦІЯ, ЗАСТОСУНОК, МОВА
ПРОГРАМУВАННЯ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ.

Об'єкт дослідження – процес розробки програмного забезпечення для шифрування та архівації файлів.

Предмет дослідження – програмні засоби для шифрування та архівації файлів.

Мета роботи – розробка програмного забезпечення для шифрування та архівації файлів, що забезпечує зменшення ризику несанкціонованого доступу до інформації та втрати даних.

Матеріали, методи та технічні засоби: мова програмування C++, середовище розробки Visual Studio.

Результати. Розроблено проєктні рішення для створення програмного забезпечення для шифрування та архівації файлів. Створено програмне забезпечення для шифрування та архівації файлів. Проведено тестування розробленої програмної системи для шифрування та архівації файлів.

Висновки. Мету роботи досягнуто. Розроблено програмне забезпечення для шифрування та архівації файлів за допомогою мови програмування C++ та середовища розробки Visual Studio.

Галузь використання – безпека даних.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 86 pages, 3 tables, 24 figures, 3 appendixes, 19 sources.

C++, VISUAL STUDIO, ARCHIVING, APPLICATION, PROGRAMMING LANGUAGE, SOFTWARE.

The object of research is the process of developing software for encrypting and archiving files.

The subject of the research is software for encrypting and archiving files.

The purpose of this work is to develop the software for encrypting and archiving files.

Materials, methods and technical tools: C++ programming language, Visual Studio development environment.

Results. Project solutions for the creation of software for encrypting and archiving files has been designed. The software to support the process of encrypting and archiving files has been developed. The developed software system for encrypting and archiving files has been tested.

Conclusions. The goal of the work has been achieved. The software tools for encrypting and archiving files using the C++ programming language and the Visual Studio development environment have been developed.

Scope of use – data security.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	8
Вступ	9
1 Аналіз предметної області	11
1.1 Програмне забезпечення для шифрування та архівації файлів	11
1.2 Аналіз програмного забезпечення для шифрування та архівації файлів	17
1.3 Висновки за розділом 1	30
2 Матеріали і методи	32
2.1 Вибір мови програмування	32
2.2 Вибір середовища розробки для створення програмного забезпечення для шифрування та архівації файлів	35
2.3 Висновки за розділом 2	38
3 Опис програми	39
3.1 Структура програмного забезпечення для шифрування та архівації файлів	39
3.2 Функціонування програмного забезпечення для шифрування та архівації файлів	41
3.3 Структура внутрішніх даних. Опис механізму збереження і організації файлів	44
3.4 Проєктування інтерфейсу програмного забезпечення для шифрування та архівації файлів	46
3.5 Висновки за розділом 3	48
4 Експлуатація, тестування та експериментальне дослідження програми	49
4.1 Призначення й умови застосування програми	49
4.2 Характеристики програми для шифрування та архівації файлів	51
4.3 Інструкція по експлуатації програми	52
4.3.1 Звернення до програми	52
4.3.2 Вхідні й вихідні дані	52
4.3.3 Повідомлення	53

4.4 Виконання програмного забезпечення для шифрування та архівації файлів	53
4.5 Тестування програмного забезпечення для шифрування та архівації файлів	56
4.6 Висновки за розділом 4	56
Висновки.....	57
Перелік джерел посилання	60
Додаток А Технічне завдання.....	62
Додаток Б Фрагмент тексту програми	67
Додаток В Слайди презентації	79

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

VS – Visual Studio;

ПЗ – програмне забезпечення;

ШД – шифрування даних.

ВСТУП

У сучасному цифровому середовищі спостерігається постійне зростання обсягів інформації, що створюється, передається та зберігається, у зв'язку з чим особливої ваги набуває питання її захисту та ефективного управління. У таких умовах програмне забезпечення, призначене для шифрування та архівації файлів, набуває значної актуальності, оскільки дозволяє одночасно забезпечити як безпеку даних, так і оптимізацію їх зберігання [1]-[3].

Важливість таких програмних рішень посилюється в умовах поширення віддаленої роботи та використання хмарних сервісів. Передача даних через мережу потребує додаткових гарантій безпеки, адже інформація часто проходить через відкриті або недостатньо захищені канали. У цьому контексті поєднання архівації та шифрування дозволяє не лише зменшити навантаження на мережу, але й забезпечити конфіденційність переданих даних [2], [3].

Таким чином, актуальність програмного забезпечення для шифрування та архівації файлів визначається сукупністю факторів, серед яких ключову роль відіграють зростання кіберзагроз, збільшення обсягів даних, розвиток мережових технологій і підвищення вимог до захисту інформації. Усе це формує стійку потребу в ефективних, надійних і зручних інструментах, здатних забезпечити безпечне зберігання та передачу даних у різноманітних умовах використання [3]-[5].

Проте поряд із безперечними перевагами, програмне забезпечення для шифрування та архівації файлів має низку недоліків, пов'язаних із складністю використання, підвищеними вимогами до ресурсів, ризиком втрати доступу до даних, проблемами сумісності та потенційними вразливостями. Попри значну користь і широке застосування, програмне забезпечення для шифрування та архівації файлів не позбавлене певних обмежень, які можуть впливати на ефективність його використання. У процесі експлуатації подібних рішень

виявляється, що підвищення рівня захисту інформації часто супроводжується ускладненням доступу до неї для самого користувача. У разі втрати пароля або ключа дешифрування доступ до зашифрованих даних може бути безповоротно втрачений, що створює суттєві ризики, особливо за відсутності належної системи резервного копіювання. Також спостерігається збільшення навантаження на обчислювальні ресурси, оскільки процеси шифрування і розшифрування потребують додаткового часу та потужностей. Це може бути критичним у випадках обробки великих обсягів інформації або роботи на пристроях із обмеженими ресурсами. У свою чергу, архівація, хоча й зменшує розмір файлів, інколи потребує додаткового часу на стиснення та розпакування, що може знижувати оперативність доступу до даних. Окремої уваги заслуговує складність налаштування та використання таких програм для непідготовлених користувачів. Неправильне застосування параметрів шифрування або архівації може призвести до зниження рівня безпеки або втрати даних. Це підкреслює важливість наявності відповідних знань і навичок для ефективної роботи з подібними інструментами.

Тому актуальною є розробка програмного забезпечення для шифрування та архівації файлів. У дипломній кваліфікаційній роботі бакалавра розв'язується актуальне завдання розробки програмного забезпечення для шифрування та архівації файлів.

Для досягнення поставленої мети у кваліфікаційній роботі бакалавра необхідно розв'язати такі задачі:

- виконати аналіз предметної області та програмних засобів для шифрування та архівації файлів;
- здійснити проєктування програмного забезпечення для шифрування та архівації файлів;
- створити програмне забезпечення для шифрування та архівації файлів;
- виконати тестування розробленого програмного забезпечення для шифрування та архівації файлів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Програмне забезпечення для шифрування та архівації файлів

Програмне забезпечення для шифрування та архівації файлів являє собою клас прикладних систем, призначених для забезпечення захисту інформації та її впорядкованого зберігання. Його сутність полягає у перетворенні даних у такий вигляд, який стає недоступним для несанкціонованого перегляду, а також у об'єднанні окремих файлів у компактні структуровані набори, що спрощують їх передачу, збереження та обробку. У результаті поєднання цих двох підходів досягається одночасно підвищення рівня безпеки та оптимізація роботи з інформаційними ресурсами [1].

Функціонування подібних програмних засобів базується на використанні криптографічних алгоритмів, які забезпечують перетворення відкритих даних у зашифрований вигляд із застосуванням спеціальних параметрів доступу. Одночасно реалізуються механізми архівації, що дозволяють зменшити обсяг інформації, об'єднати її у зручні структури та підвищити ефективність використання дискового простору. Такий підхід є особливо актуальним у середовищах, де обробляються значні обсяги даних або необхідно забезпечити їх захист під час передачі через мережу [1].

Особливістю таких програмних систем є здатність забезпечувати прозору взаємодію з файлами, коли процеси шифрування та архівації виконуються без значного втручання користувача. При цьому інтерфейс зазвичай орієнтований на інтуїтивне використання, що дозволяє виконувати складні операції захисту даних без глибоких технічних знань. Важливим аспектом є також можливість налаштування параметрів обробки, що дозволяє адаптувати програму до різних сценаріїв застосування та рівнів безпеки [1].

Переваги такого програмного забезпечення проявляються у забезпеченні конфіденційності інформації, зниженні ризиків її втрати або несанкціонованого доступу, а також у підвищенні ефективності організації

даних. Додатково досягається зручність передачі файлів, оскільки архівовані та зашифровані дані займають менше місця та можуть бути передані як єдине ціле. У сучасних умовах цифровізації подібні рішення стають невід'ємною складовою інформаційної безпеки як для окремих користувачів, так і для організацій [1].

Програмні засоби, призначені для захисту інформації, можуть реалізовуватися у різних варіантах, що відрізняються за сферою застосування та рівнем охоплення даних. У межах одного підходу передбачається обробка окремих файлів або каталогів, які зберігаються на пристрої, що дозволяє забезпечити конфіденційність конкретних документів чи інших інформаційних об'єктів. Такий спосіб часто застосовується у випадках, коли необхідно обмежити доступ до вибіркового даних перед їх передачею або розміщенням у зовнішніх сховищах. В іншому випадку захист у подібних програмних системах поширюється на весь носій інформації, що означає повне перетворення вмісту накопичувача у зашифрований вигляд. Це унеможливорює доступ до будь-яких даних без відповідних засобів автентифікації навіть при фізичному доступі до пристрою [1], [2].

Окремим напрямом є забезпечення конфіденційності електронного листування, де захист поширюється як на текст повідомлення, так і на вкладення, що передаються разом із ним. Такий механізм гарантує, що інформація може бути прочитана виключно адресатом, для якого вона призначена. Поряд із цим існують рішення, орієнтовані на захист процесів обміну даними у реальному часі, включаючи текстові повідомлення, голосові виклики та відеозв'язок, де криптографічні методи застосовуються безпосередньо під час передачі інформації через мережу. Ще один варіант використання пов'язаний із захистом структурованих сховищ, де обробці підлягають записи, що містять конфіденційні відомості, зокрема персональні або фінансові дані [1], [2].

Принцип функціонування таких програмних систем базується на використанні спеціалізованих математичних перетворень, які змінюють

початковий вигляд інформації таким чином, щоб вона стала непридатною для сприйняття без відповідного ключа. У межах одного підходу застосовується єдиний ключ для виконання як перетворення даних, так і їх відновлення, що забезпечує високу швидкість обробки, але потребує надійного способу передачі цього ключа між сторонами. Інший підхід передбачає використання пари взаємопов'язаних ключів, один із яких використовується для перетворення інформації, а інший для її відновлення, що дозволяє уникнути необхідності передавати секретні дані відкритими каналами [1], [2].

Сучасні програмні засоби для шифрування та архівації файлів автоматизують більшість операцій, пов'язаних із генерацією, зберіганням і застосуванням ключів, що дозволяє використовувати складні криптографічні алгоритми без необхідності глибокого розуміння їх внутрішньої будови. Завдяки цьому забезпечується поєднання високого рівня захисту з доступністю для широкого кола користувачів, що робить такі рішення ефективними як у професійному, так і в повсякденному використанні [1], [2].

Значущість програмних засобів для шифрування та архівації файлів важко переоцінити в умовах сучасної цифрової взаємодії, де обмін даними відбувається практично миттєво та глобально. Шифрування забезпечує конфіденційність персональних і чутливих даних, обмежуючи доступ сторонніх осіб і зменшуючи ризик викрадення інформації чи порушення приватності. Для бізнесу такі системи гарантують збереження корпоративних ресурсів, включно з інтелектуальною власністю та даними клієнтів, від несанкціонованого доступу, кібератак та промислового шпигунства, створюючи додатковий рівень безпеки та стабільності у роботі. Крім того, застосування криптографічних методів допомагає відповідати вимогам законодавчих і галузевих стандартів щодо захисту даних, що стає обов'язковим елементом ділової практики у багатьох сферах. Програмне забезпечення для шифрування формує основу для безпечних транзакцій в Інтернеті, у тому числі для банківських операцій, онлайн покупок та обміну інформацією, забезпечуючи надійний захист фінансових та особистих даних.

Надійне шифрування підвищує рівень довіри клієнтів і партнерів до організацій, що застосовують його для захисту своєї інформації, створюючи впевненість у безпечності цифрової взаємодії [2], [3].

Разом із цим варто враховувати певні обмеження та аспекти використання таких програмних систем. Управління ключами шифрування є критично важливим, оскільки їхня втрата унеможлиблює доступ до захищених даних. Процеси шифрування та дешифрування потребують значних обчислювальних ресурсів, що може впливати на продуктивність пристроїв, особливо тих, що мають застаріле обладнання. Деякі рішення вимагають технічних знань для ефективного налаштування та користування, а у ряді країн існують юридичні обмеження щодо застосування високого рівня криптографічного захисту, що теж слід враховувати при впровадженні таких технологій [2], [3].

Програмні засоби для шифрування виступають ключовим компонентом сучасної кібербезпеки, що дозволяє забезпечувати приватність і цілісність даних, дотримуватися правових стандартів та формувати довіру у цифровому середовищі. В умовах постійного розвитку кіберзагроз розуміння принципів роботи та ефективне застосування таких систем стає необхідністю для кожного користувача, незалежно від рівня професійної підготовки, забезпечуючи надійний захист інформації у цифровому просторі [2], [3].

Під час підбору програмного забезпечення для шифрування та архівації файлів слід ретельно оцінювати його характеристики, щоб упевнитися в тому, що воно здатне забезпечити необхідний рівень безпеки. Особливу увагу приділяють надійності алгоритмів шифрування та довжині ключів, які визначають міцність захисту даних від несанкціонованого доступу. Важливо, щоб програмні засоби були зручними у використанні, інтуїтивними та простими у налаштуванні, що дозволяє без зайвих труднощів виконувати процеси шифрування та дешифрування інформації. Критичною є сумісність із застосовуваними операційними системами та апаратними платформами, що гарантує стабільну роботу без технічних конфліктів. Значну роль відіграють

механізми управління ключами, які забезпечують надійне зберігання та ефективну організацію криптографічних ключів, дозволяючи уникнути втрати доступу до даних. Окрім цього, слід перевіряти відповідність програмного забезпечення вимогам галузевих стандартів і нормативів, що регулюють захист інформації, щоб забезпечити правову безпеку під час обробки конфіденційних даних [2], [3].

Таким чином, програмне забезпечення для шифрування виступає основним інструментом забезпечення конфіденційності та цілісності інформації. Усвідомлення принципів його роботи, типів використовуваних алгоритмів та функціональних можливостей дозволяє користувачам захищати свої дані на високому рівні, зберігаючи контроль над доступом та створюючи надійну основу для безпечної цифрової взаємодії. Ефективне використання таких систем формує впевненість у безпеці інформації та зменшує ризики витоку або компрометації даних [2], [3].

Процес розробки програмного забезпечення для шифрування починається з визначення меж проєкту та чіткого усвідомлення мети, яка полягає у створенні додатку для надійного шифрування файлів, а не у винаході нового алгоритму. Створення власного алгоритму потребує глибоких знань з математики та статистики, а також ретельної перевірки для забезпечення безпеки, тому зазвичай застосовуються вже перевірені стандартні алгоритми шифрування. Наступним етапом є планування підходу до розробки, де основна увага приділяється бізнес-функціоналу додатку, щоб інтеграція шифрування була максимально простою та ефективною для кінцевого користувача [4], [5].

Формування команди розробників має відбуватися з урахуванням необхідних компетенцій для створення веб- та мобільного додатку, включаючи фахівців із бізнес-аналізу, дизайну інтерфейсу, Android- та веб-розробки, тестування та керування проєктом. Ітеративна розробка, наприклад методологія Agile, дозволяє ефективно впроваджувати зміни та покращувати функціонал у ході проєкту. Для вебзастосунків доцільно застосовувати

платформу як послугу, яка спрощує налаштування хмарної інфраструктури, управління базою даних, користувачами та інтеграцію сторонніх сервісів. Мобільний додаток отримує користь від мобільного бекенду як послуги, який обробляє серверну частину, сховище, безпеку та сповіщення, дозволяючи розробникам зосередитися на фронтенді, а також легко інтегрувати шифрування за допомогою відповідних бібліотек [4], [5].

Шифрування в додатках зазвичай здійснюється через симетричне або конвертне шифрування, де головний ключ захищає ключі для шифрування даних, часто з використанням засоби з високою довжиною ключа для забезпечення галузевих стандартів безпеки. Для вебзастосунку застосовуються середовища виконання та інструменти DevOps для автоматизації тестування, керування збірками та безперервної інтеграції, що сприяє ітеративній розробці та стабільності продукту. У мобільному середовищі використовуються спеціалізовані інструменти, для забезпечення оптимальної розробки та тестування додатків [4], [5].

Управління проєктом розробки програмного забезпечення для шифрування та архівації файлів доцільно виконувати за принципами Scrum, де міжфункціональна команда працює над поставленими завданнями через ітерації, або «спринти». Кожен спринт включає планування, щоденні короткі зустрічі для оцінки прогресу, огляди результатів із залученням зацікавлених сторін та ретроспективні зустрічі для аналізу виконаної роботи і вдосконалення процесів. Такий підхід забезпечує структуровану та контрольовану розробку, дозволяє швидко впроваджувати зміни та підтримувати високу якість програмного забезпечення для шифрування файлів [4], [5].

Визначено, що програмне забезпечення для шифрування та архівації файлів являє собою клас прикладних систем, призначених для забезпечення захисту інформації та її впорядкованого зберігання. Його сутність полягає у перетворенні даних у такий вигляд, який стає недоступним для несанкціонованого перегляду, а також у об'єднанні окремих файлів у

компактні структуровані набори, що спрощують їх передачу, збереження та обробку.

1.2 Аналіз програмного забезпечення для шифрування та архівації файлів

Проаналізуємо програмне забезпечення для шифрування та архівації файлів [6]-[15].

Програмне забезпечення NordLocker (рис. 1.1) являє собою програмний продукт, орієнтований на забезпечення захисту даних шляхом їх шифрування та організації безпечного зберігання. Основна ідея функціонування цього програмного забезпечення полягає у формуванні спеціальних захищених сховищ, доступ до яких надається лише після автентифікації користувача, що дозволяє ізолювати його конфіденційну інформацію від стороннього втручання [6], [11], [13].

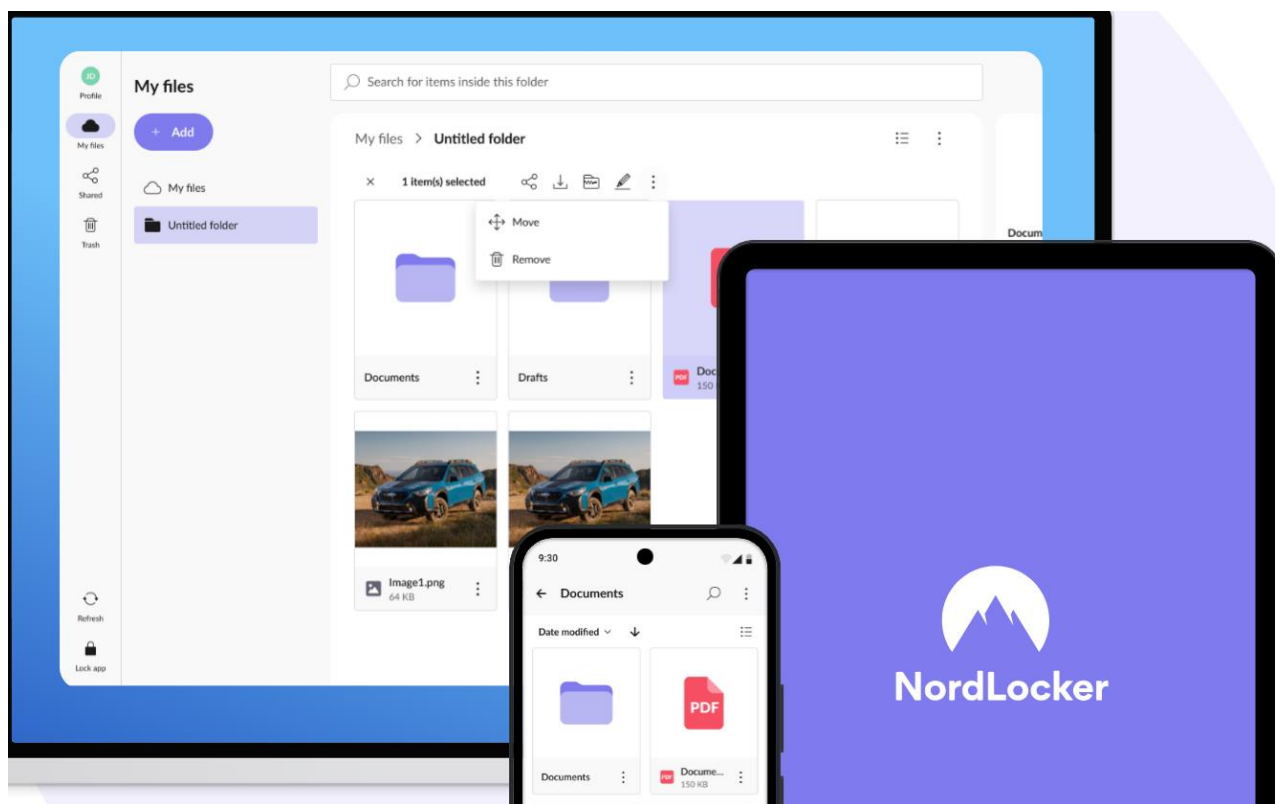


Рисунок 1.1 – Програмне забезпечення NordLocker [13]

У процесі роботи використовується підхід, за якого файли та каталоги поміщаються до так званих захищених контейнерів, що функціонують як замкнені області зберігання. Дані, розміщені всередині таких контейнерів, автоматично піддаються криптографічному перетворенню, що унеможливорює їх перегляд без відповідних прав доступу. Водночас передбачено можливість створення декількох подібних середовищ, що забезпечує гнучку організацію інформації залежно від потреб користувача або організації [6], [11], [13].

Особливістю програмного продукту NordLocker є поєднання функцій локального шифрування з можливостями хмарного зберігання. Завдяки цьому забезпечується доступ до захищених даних з різних пристроїв, що працюють під управлінням популярних операційних систем, а також через вебінтерфейс. Такий підхід сприяє підвищенню мобільності роботи з інформацією та дозволяє організувати безпечний доступ до неї незалежно від місця перебування користувача. Крім того, реалізовано механізми синхронізації, які забезпечують актуальність даних на всіх підключених пристроях [6], [11], [13].

Значна увага у NordLocker приділяється можливості обміну інформацією між користувачами. Передача даних супроводжується шифруванням, що виконується з урахуванням параметрів доступу отримувача, завдяки чому забезпечується конфіденційність навіть під час пересилання файлів. Додатково передбачено функціонал запиту захищених файлів, що розширює можливості взаємодії між користувачами у межах системи. При цьому контроль доступу до інформації базується на використанні спеціального коду безпеки, який необхідний для відкриття отриманих даних [6], [11], [13].

Архітектура програмного забезпечення NordLocker побудована таким чином, що навіть розробник не має доступу до вмісту файлів користувача, що відповідає концепції відсутності розголошення даних. Усі операції шифрування та дешифрування виконуються локально або в захищеному середовищі, що мінімізує ризики витоку інформації. Додатково реалізовано зручні механізми взаємодії з файлами, зокрема можливість додавання даних

до захищеного сховища шляхом простого перетягування, що спрощує використання програмного продукту навіть для користувачів без спеціальних технічних знань [6], [11], [13].

Водночас у процесі використання ПЗ NordLocker можуть виникати певні обмеження, пов'язані з організацією безпеки та особливостями реалізації функціоналу. Зокрема, незашифровані копії файлів, що залишаються поза межами захищених контейнерів, можуть становити потенційну загрозу, якщо не контролюється їхнє зберігання. Крім того, деякі механізми додаткової автентифікації можуть бути відсутні або реалізовані обмежено, що потребує підвищеної уваги з боку користувача при роботі з конфіденційною інформацією [6], [11], [13].

Отже, програмне забезпечення NordLocker являє собою комплексне рішення для шифрування та організації безпечного зберігання файлів, яке поєднує локальні та хмарні підходи до захисту даних, забезпечує зручність використання та підтримує сучасні механізми безпечного обміну інформацією між користувачами [6], [11], [13].

Основними характеристиками програмного забезпечення NordLocker є такі [9], [10], [13]:

- універсальне шифрування даних: програмний продукт NordLocker забезпечує захист файлів шляхом їх перетворення у зашифрований вигляд із можливістю подальшого безпечного зберігання та передачі між користувачами [6], [11], [13];

- контейнерний підхід до зберігання: у програмному забезпеченні NordLocker використовується механізм створення спеціальних захищених сховищ, у межах яких файли ізольовані від зовнішнього доступу та можуть відкриватися лише після авторизації [6], [11], [13];

- інтеграція з хмарним середовищем: у програмному забезпеченні NordLocker реалізовано можливість синхронізації даних між пристроями, що забезпечує доступ до зашифрованих файлів із різних платформ [6], [11], [13];

- безпечний обмін інформацією: у програмному забезпеченні NordLocker передбачено передачу файлів користувача із використанням шифрування, що гарантує захист даних навіть під час їх надсилання іншим користувачам [6], [11], [13];

- архітектура без доступу до даних: у програмному забезпеченні NordLocker застосовується підхід, за якого навіть розробник не має можливості отримати доступ до вмісту зашифрованих файлів [6], [11], [13];

- простота використання: інтерфейс у програмному забезпеченні NordLocker організовано таким чином, що операції додавання файлів до захищеного середовища можуть виконуватися інтуїтивно, зокрема шляхом перетягування [6], [11], [13];

- багатоплатформеність: програмне забезпечення NordLocker може використовуватися на різних операційних системах і пристроях, включаючи настільні та мобільні середовища [6], [11], [13];

- автоматизація процесів захисту: у програмному забезпеченні NordLocker після додавання файлів до захищеного сховища їх шифрування виконується автоматично без додаткових дій користувача [6], [11], [13];

- контроль доступу до даних: доступ до зашифрованої інформації у програмному забезпеченні NordLocker здійснюється лише за наявності відповідних параметрів автентифікації [6], [11], [13];

- підтримка організації файлів: у програмному забезпеченні NordLocker передбачено можливість структурування даних у межах захищених контейнерів для зручного управління інформацією [6], [11], [13].

Серед недоліків програмного забезпечення NordLocker можна відзначити обмеження функціоналу безкоштовної версії, що знижує можливості повноцінного використання системи без придбання розширеного доступу, а також потенційні ризики, пов'язані з наявністю незахищених копій файлів поза межами зашифрованого середовища. Крім того, окремі механізми додаткової автентифікації можуть бути реалізовані недостатньо гнучко, що

потребує більш уважного контролю з боку користувача під час роботи з конфіденційною інформацією [6], [11], [13].

Програмне забезпечення Virtru (рис. 1.2) являє собою програмне рішення, орієнтоване на забезпечення захисту електронної кореспонденції та вкладених файлів шляхом використання сучасних криптографічних механізмів. Функціонування програмного забезпечення Virtru базується на застосуванні інфраструктури відкритих ключів, що дозволяє організувати безпечний обмін інформацією між користувачами та гарантувати, що доступ до переданих даних отримують лише визначені отримувачі. Такий підхід забезпечує високий рівень конфіденційності під час передачі інформації через електронну пошту, що є особливо важливим для організацій, які працюють із чутливими даними [8], [14].

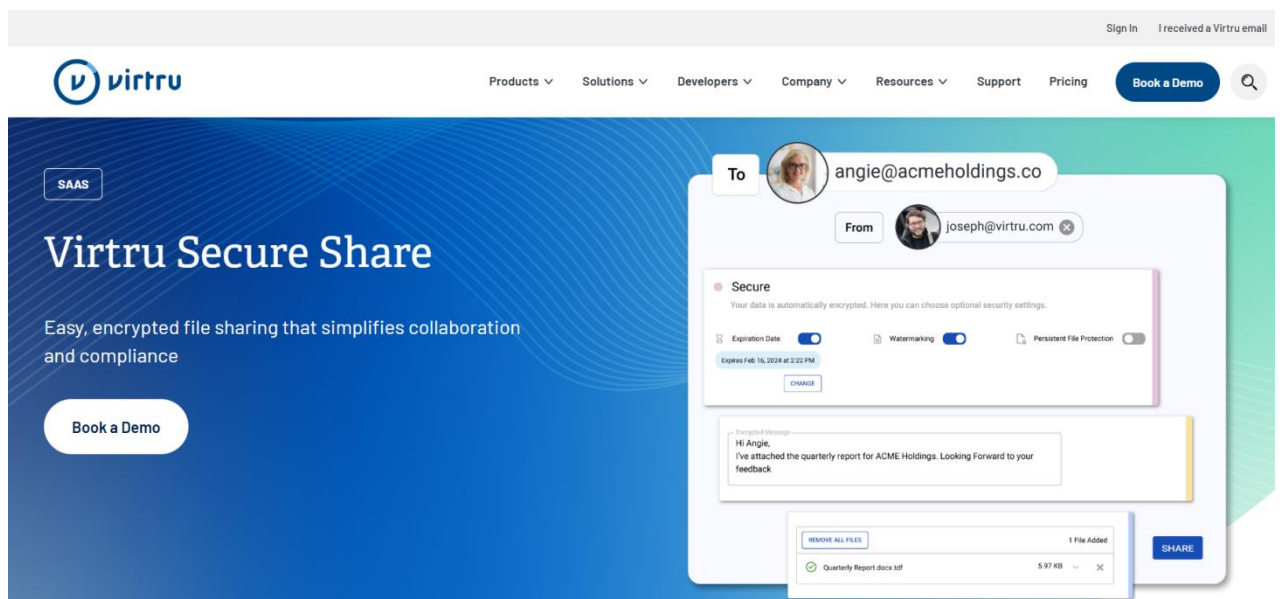


Рисунок 1.2 – Програмне забезпечення Virtru [14]

У процесі використання програмного забезпечення Virtru забезпечується захист як текстового вмісту листів, так і вкладених файлів, що передаються разом із ними. Передача даних супроводжується наскрізним шифруванням, завдяки чому інформація залишається недоступною для сторонніх осіб на всіх етапах її пересилання. Доступ до отриманих матеріалів

можливий лише після виконання відповідних процедур автентифікації, що включають використання ключів та додаткових засобів підтвердження. При цьому користувач отримує можливість самостійно визначати параметри доступу до надісланих матеріалів, що дозволяє контролювати, хто і за яких умов може переглядати вкладення [8], [14].

Особливістю програмного забезпечення Virtru є його здатність інтегруватися з популярними поштовими сервісами, такими як Gmail та Microsoft Outlook, що значно спрощує впровадження у вже існуючі робочі процеси. Після активації відповідних модулів захист застосовується автоматично, не потребуючи складних налаштувань, що підвищує зручність використання. Додатково реалізовано механізм аудиту, який дозволяє відстежувати доступ до надісланих повідомлень і вкладень, забезпечуючи прозорість роботи з даними та можливість контролю їх використання [8], [14].

Важливою характеристикою програмного забезпечення Virtru є можливість керування доступом до інформації навіть після її відправлення. Передбачено інструменти, що дозволяють обмежувати перегляд вкладень або скасовувати доступ до них, що підвищує рівень контролю над переданими матеріалами. Такий функціонал є корисним у випадках, коли необхідно оперативно реагувати на зміну умов доступу або запобігти несанкціонованому використанню інформації [8], [14].

Разом із тим під час експлуатації програмного забезпечення Virtru можуть виникати певні труднощі, пов'язані з роботою на мобільних пристроях, де відображення та відкриття захищених повідомлень іноді відбувається менш стабільно. Також можуть спостерігатися обмеження щодо розміру файлів, що передаються, що ускладнює обмін великими обсягами інформації. Додатково інтерфейс системи може сприйматися як складний, а процес обробки зашифрованих повідомлень іноді супроводжується затримками, що впливає на загальну зручність використання [8], [14].

Отже, Virtru можна розглядати як потужний інструмент для забезпечення безпеки електронної пошти та вкладених файлів, який поєднує

криптографічний захист, гнучке управління доступом і інтеграцію з поштовими сервісами, хоча окремі аспекти реалізації можуть потребувати вдосконалення для підвищення ефективності роботи в різних умовах використання [8], [14].

Програмне забезпечення Virtru має такі особливості [8], [14]:

- захист електронної пошти: програмне забезпечення Virtru забезпечує шифрування вмісту листів і вкладених файлів під час їх передавання між користувачами [8], [14];

- використання відкритих ключів: у програмному забезпеченні Virtru реалізовано криптографічний механізм на основі інфраструктури відкритих ключів для забезпечення контрольованого доступу до інформації [8], [14];

- наскрізне шифрування: у програмному забезпеченні Virtru гарантується захист даних на всіх етапах передачі від відправника до отримувача без можливості їх перехоплення [8], [14];

- інтеграція з поштовими сервісами: у програмному забезпеченні Virtru підтримується взаємодія з такими системами, як Gmail та Microsoft Outlook, що спрощує використання у звичних середовищах [8], [14];

- контроль доступу до файлів: у програмному забезпеченні Virtru надається можливість визначати права доступу до вкладень та змінювати їх навіть після відправлення повідомлення [8], [14];

- аудит дій користувачів: у програмному забезпеченні Virtru забезпечується відстеження доступу до повідомлень і вкладень з метою підвищення прозорості та безпеки [8], [14];

- підтримка безпечного обміну файлами: у програмному забезпеченні Virtru передбачено механізми передачі конфіденційної інформації із збереженням її цілісності та захищеності [8], [14];

- гнучкість налаштувань безпеки: користувач програмного забезпечення Virtru може самостійно визначати параметри доступу та політики використання захищених даних [8], [14];

- автоматизація захисту: у програмному забезпеченні Virtru після активації відповідних функцій шифрування застосовується без необхідності складних додаткових дій [8], [14];

- підтримка корпоративного використання: програмне забезпечення Virtru орієнтовано на застосування в організаціях, що працюють із конфіденційною інформацією [8], [14].

Серед недоліків програмного забезпечення Virtru можна відзначити складність інтерфейсу, яка може ускладнювати швидке освоєння функціоналу, а також періодичні затримки під час роботи із зашифрованими повідомленнями. Додатково можуть виникати труднощі при використанні на мобільних пристроях, де стабільність роботи не завжди є достатньою. Окремі обмеження, пов'язані з передачею файлів великого обсягу, також можуть знижувати ефективність застосування системи у випадках інтенсивного обміну даними [8], [14].

Програмне забезпечення Cryptomator (рис. 1.3) призначено для захисту інформації, що зберігається у хмарних середовищах, шляхом попереднього шифрування даних безпосередньо на стороні користувача. Його робота організована таким чином, що файли проходять криптографічне перетворення ще до моменту їх передачі у віддалене сховище, завдяки чому навіть у випадку компрометації сервісу зберігання сторонні особи не можуть отримати доступ до змісту інформації. Програмний продукт взаємодіє з популярними платформами, такими як Dropbox, Google Drive та OneDrive, а також із багатьма іншими подібними сервісами, що забезпечує його універсальність у використанні. У межах функціонування програмного забезпечення Cryptomator створюються спеціальні захищені області зберігання, які зовні виглядають як звичайні каталоги файлової системи. Такий підхід дозволяє працювати з інформацією у звичному форматі без необхідності змінювати робочі процеси, водночас гарантуючи високий рівень безпеки. Кожен файл обробляється окремо, що підвищує стійкість до можливих загроз, оскільки порушення цілісності одного елемента не впливає на інші. Додатково

захищаються не лише самі дані, але й їхні імена, що ускладнює аналіз структури інформації сторонніми особами [10], [15].

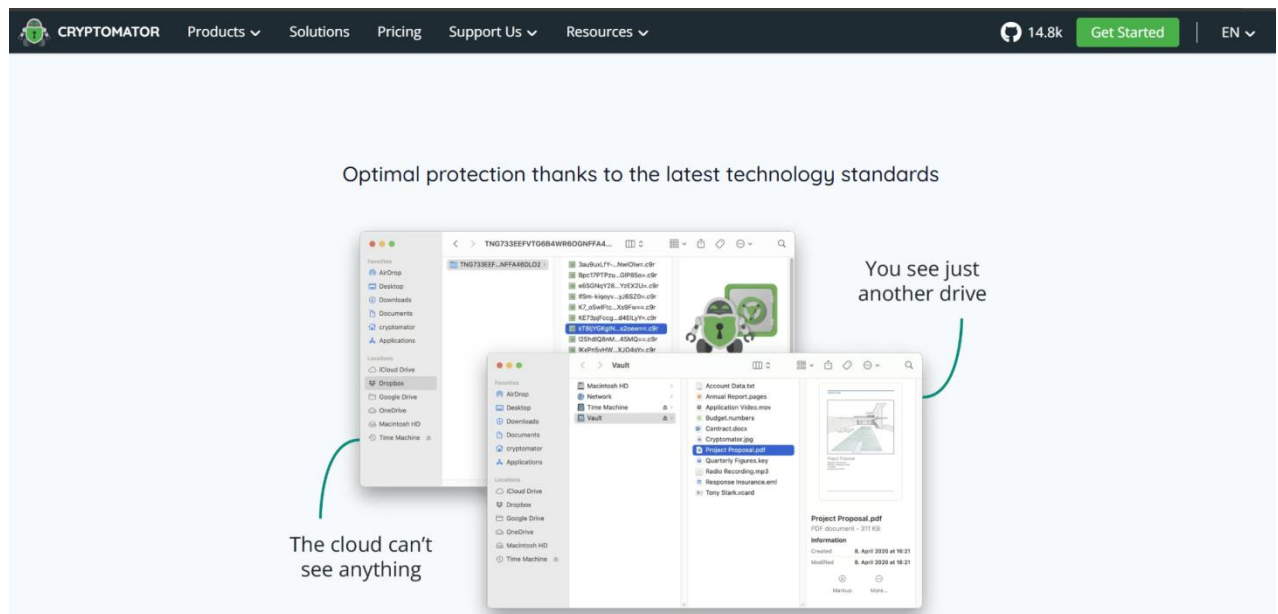


Рисунок 1.3 – Програмне забезпечення Cryptomator [15]

Особливістю цього програмного забезпечення є прозорість процесу шифрування, що означає можливість взаємодії з файлами безпосередньо через звичний інтерфейс файлової системи. Для цього створюється віртуальне середовище, яке імітує роботу стандартного диска, дозволяючи відкривати, змінювати та зберігати файли без додаткових складних дій. При цьому всі зміни автоматично відображаються у зашифрованому вигляді в каталозі синхронізації хмарного сервісу [10], [15].

Важливим аспектом програмного забезпечення Cryptomator є гнучкість у використанні, оскільки програмний продукт може застосовуватися як для локальних носіїв, так і для різних хмарних платформ. Передбачено можливість створення декількох незалежних сховищ для розмежування даних за призначенням або рівнем конфіденційності. Завдяки пофайловій організації зберігання зміни торкаються лише тих елементів, які були модифіковані, що сприяє ефективній синхронізації та зменшенню обсягу переданих користувачем даних [10], [15].

Інтерфейс користувача програмного забезпечення Cryptomator орієнтований на простоту та зрозумілість, що дозволяє швидко освоїти основні принципи роботи навіть без глибоких технічних знань. Водночас акцент зроблено на базових функціях захисту, без значного розширення додатковими інструментами контролю чи аналітики. Це визначає його як спеціалізоване рішення, основне призначення якого полягає саме у забезпеченні конфіденційності даних під час їх зберігання у хмарному середовищі [10], [15].

Разом із тим функціональність програмного забезпечення Cryptomator обмежується переважно процесами шифрування, що означає відсутність вбудованих засобів обміну файлами або розширеного управління доступом, окрім тих можливостей, які надає сам сервіс зберігання. Для повноцінного використання необхідною умовою є наявність відповідного облікового запису у хмарному середовищі. Деякі аспекти налаштування програмного забезпечення Cryptomator можуть вимагати певного рівня підготовки, особливо при інтеграції з різними платформами [10], [15].

Отже, Cryptomator являє собою ефективний інструмент для організації безпечного зберігання даних у хмарі, який поєднує простоту використання, прозорість роботи та надійний криптографічний захист, орієнтуючись передусім на забезпечення конфіденційності інформації без ускладнення взаємодії з нею [10], [15].

Програмне забезпечення Cryptomator має такі особливості [10], [15]:

- клієнтське шифрування даних: програмне забезпечення Cryptomator виконує перетворення файлів у захищений вигляд безпосередньо на стороні користувача перед передачею у віддалене сховище [10], [15];

- прозорий режим роботи: у програмному забезпеченні Cryptomator забезпечується можливість взаємодії з файлами у звичному форматі без необхідності додаткових складних операцій [10], [15];

- контейнерна організація зберігання: у програмному забезпеченні Cryptomator використовується створення окремих захищених сховищ, що функціонують як звичайні папки файлової системи [10], [15];
- пофайлове шифрування: у програмному забезпеченні Cryptomator кожен файл обробляється окремо, що підвищує рівень безпеки та зменшує ризик компрометації всієї структури даних [10], [15];
- захист метаданих: у програмному забезпеченні Cryptomator додатково шифруються імена файлів і структура каталогів для унеможливлення аналізу інформації сторонніми особами [10], [15];
- інтеграція з хмарними сервісами: у програмному забезпеченні Cryptomator забезпечується сумісність із такими платформами, як Dropbox, Google Drive та OneDrive, що дозволяє використовувати існуючу інфраструктуру [10], [15];
- віртуальний диск доступу: у програмному забезпеченні Cryptomator створюється логічне середовище, яке імітує роботу звичайного накопичувача для зручного доступу до зашифрованих даних [10], [15];
- ефективна синхронізація: передача змін до хмарного сховища відбувається лише для модифікованих файлів, що оптимізує використання ресурсів [10], [15];
- кросплатформена підтримка: у програмному забезпеченні Cryptomator забезпечується можливість використання програмного забезпечення на різних операційних системах та пристроях [10], [15];
- відкритий вихідний код: програмний продукт Cryptomator є доступним для аналізу та модифікації, що підвищує рівень довіри до механізмів захисту [10], [15].

Серед недоліків програмного забезпечення Cryptomator можна відзначити обмеженість функціоналу, оскільки основна увага зосереджена виключно на шифруванні даних без розширених можливостей обміну файлами або контролю доступу. Також для повноцінного використання необхідною є наявність стороннього хмарного сервісу, що створює залежність

від зовнішніх платформ. Окремі аспекти налаштування можуть вимагати певного рівня технічної підготовки, що може ускладнювати використання програмного забезпечення для недосвідчених користувачів [10], [15].

Порівняльну характеристику програмного забезпечення для шифрування та архівації файлів наведено у таблиці 1.1.

Таблиця 1.1 – Порівняння програмного забезпечення для шифрування та архівації файлів

Критерій порівняння	NordLocker [13]	Virtru [14]	Cryptomator [15]
інтеграція з хмарними сервісами	+	+—	+
робота з файлами поза поштою	+	—	+
відкритий вихідний код	—	—	+
простота використання	+	+—	+—
підтримка спільного доступу	+—	+	—
робота з великими файлами	+—	—	+

За результатами проведеного аналізу можна зробити висновок, що у наш час існує досить багато програмних засобів для шифрування та архівації файлів. Проте деякі програмні засоби для шифрування та архівації файлів мають низку недоліків, пов'язаних із складністю використання, підвищеними вимогами до ресурсів, ризиком втрати доступу до даних, проблемами сумісності та потенційними вразливостями. Попри значну користь і широке застосування, програмне забезпечення для шифрування та архівації файлів не позбавлене певних обмежень, які можуть впливати на ефективність його використання. У процесі експлуатації подібних рішень виявляється, що підвищення рівня захисту інформації часто супроводжується ускладненням доступу до неї для самого користувача. У разі втрати пароля або ключа дешифрування доступ до зашифрованих даних може бути безповоротно

втрачений, що створює суттєві ризики, особливо за відсутності належної системи резервного копіювання. Також спостерігається збільшення навантаження на обчислювальні ресурси, оскільки процеси шифрування і розшифрування потребують додаткового часу та потужностей. Це може бути критичним у випадках обробки великих обсягів інформації або роботи на пристроях із обмеженими ресурсами. У свою чергу, архівація, хоча й зменшує розмір файлів, інколи потребує додаткового часу на стиснення та розпакування, що може знижувати оперативність доступу до даних. Окремої уваги заслуговує складність налаштування та використання таких програм для непідготовлених користувачів. Неправильне застосування параметрів шифрування або архівації може призвести до зниження рівня безпеки або втрати даних. Це підкреслює важливість наявності відповідних знань і навичок для ефективної роботи з подібними інструментами. Тому актуальною є розробка програмного забезпечення для шифрування та архівації файлів.

При розробці програмного забезпечення для шифрування та архівації файлів необхідно забезпечити такі функціональні вимоги:

- підтримка можливості вибору на комп'ютері для подальшого шифрування;
- забезпечення можливості введення і збереження ключа шифрування разом з параметрами блоку для захисту файлів;
- підтримка можливості створення зашифрованого файлу з вказанням користувачем імені та шляху збереження;
- забезпечення можливості відновлення оригінального файлу за допомогою правильного ключа та параметрів блоку;
- підтримка захисту даних від несанкціонованого доступу шляхом перевірки правильності ключа і параметрів блоку перед дешифруванням;
- інтеграція графічного інтерфейсу користувача для зручного управління процесами шифрування, дешифрування та вибору файлів.

1.3 Висновки за розділом 1

Визначено, що програмне забезпечення для шифрування та архівації файлів являє собою клас прикладних систем, призначених для забезпечення захисту інформації та її впорядкованого зберігання. Його сутність полягає у перетворенні даних у такий вигляд, який стає недоступним для несанкціонованого перегляду, а також у об'єднанні окремих файлів у компактні структуровані набори, що спрощують їх передачу, збереження та обробку.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато програмних засобів для шифрування та архівації файлів. Проте деякі програмні засоби для шифрування та архівації файлів мають низку недоліків, пов'язаних із складністю використання, підвищеними вимогами до ресурсів, ризиком втрати доступу до даних, проблемами сумісності та потенційними вразливостями. Попри значну користь і широке застосування, програмне забезпечення для шифрування та архівації файлів не позбавлене певних обмежень, які можуть впливати на ефективність його використання. У процесі експлуатації подібних рішень виявляється, що підвищення рівня захисту інформації часто супроводжується ускладненням доступу до неї для самого користувача. У разі втрати пароля або ключа дешифрування доступ до зашифрованих даних може бути безповоротно втрачений, що створює суттєві ризики, особливо за відсутності належної системи резервного копіювання. Також спостерігається збільшення навантаження на обчислювальні ресурси, оскільки процеси шифрування і розшифрування потребують додаткового часу та потужностей. Це може бути критичним у випадках обробки великих обсягів інформації або роботи на пристроях із обмеженими ресурсами. Особливої уваги заслуговує складність налаштування та використання таких програм для непідготовлених користувачів. Неправильне застосування параметрів шифрування або архівації може призвести до зниження рівня безпеки або втрати даних. Це підкреслює

важливість наявності відповідних знань і навичок для ефективної роботи з подібними інструментами. Тому актуальною є розробка програмного забезпечення для шифрування та архівації файлів.

Сформульовано функціональні вимоги до програмного забезпечення для шифрування та архівації файлів.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Вибір мови програмування

При розробці програмного забезпечення для шифрування та архівації файлів було використано мову програмування C++ [16], [17].

Мова програмування C++ відзначається широкими можливостями для створення складних програмних систем завдяки поєднанню об'єктно-орієнтованого підходу та низькорівневого контролю над комп'ютерними ресурсами. Її структура дозволяє ефективно організовувати великі проєкти, розділяючи код на класи та модулі, що полегшує його підтримку, масштабування та повторне використання. Одночасно надається прямий доступ до пам'яті та апаратних ресурсів, що забезпечує високу продуктивність програм і точне управління обчислювальними процесами [16], [17].

Особливістю мови є здатність працювати з різними парадигмами програмування, включаючи процедурний, об'єктно-орієнтований та шаблонний підхід. Це дозволяє розробникам обирати оптимальний стиль для конкретних завдань і створювати гнучкі та універсальні рішення. Крім того, C++ має потужні механізми абстракції, які дозволяють приховувати складні деталі реалізації і водночас забезпечувати ефективну взаємодію компонентів системи [16], [17].

Програмне забезпечення на C++ часто вирізняється стабільністю та надійністю, оскільки мова дозволяє максимально контролювати роботу з пам'яттю та ресурсами, запобігаючи небажаним витокам і збоїв у роботі програми. Крім того, наявність великої кількості бібліотек і розширень сприяє швидкій розробці навіть складних алгоритмів, включаючи ті, що потребують інтенсивних обчислень або складних структур даних [16], [17].

C++ також забезпечує високу портативність, що дозволяє переносити програмні рішення між різними операційними системами та апаратними платформами без значної переробки коду. Завдяки цьому мова залишається актуальною для створення як системного програмного забезпечення, так і

спеціалізованих прикладних рішень, де важливі продуктивність, надійність і контроль над ресурсами [16], [17].

Для розробки програмного забезпечення, призначеного для шифрування та архівації файлів, мова програмування C++ виявляється особливо доцільною через поєднання високої продуктивності, гнучкості та контролю над ресурсами. Однією з ключових переваг є те, що C++ дозволяє безпосередньо працювати з пам'яттю та низькорівневими структурами даних, що є критично важливим для алгоритмів шифрування, які потребують точного управління обсягами оброблюваних даних та оптимізації швидкодії. Такий контроль забезпечує швидке виконання складних криптографічних обчислень і стискування великих обсягів інформації без зайвих затримок [16], [17].

C++ надає широкі можливості для роботи з багатопоточністю та паралельними обчисленнями, що дозволяє значно прискорювати процеси архівації та шифрування на сучасних багатоядерних процесорах. Це особливо важливо при обробці великих файлів або пакетів даних, де одночасне виконання операцій може зменшити час очікування і підвищити ефективність програмного забезпечення [16], [17].

Також важливим чинником є наявність потужних бібліотек та фреймворків для C++, які спрощують реалізацію складних криптографічних алгоритмів, підтримку сучасних форматів архівів і шифрів, а також забезпечують сумісність із різними операційними системами. Завдяки цьому можна створювати кросплатформні рішення, які зберігають високу швидкодію і надійність [16], [17].

Не менш суттєвим є те, що C++ дозволяє реалізовувати програмне забезпечення з високим рівнем оптимізації під конкретне обладнання. Це дає змогу мінімізувати витрати оперативної пам'яті, скоротити розмір виконуваних файлів та знизити енергоспоживання, що особливо важливо для мобільних пристроїв або серверних систем, де продуктивність і ефективність ресурсів мають значення [16], [17].

Таким чином, C++ забезпечує баланс між продуктивністю, контролем над ресурсами та гнучкістю, що робить її ідеальною для створення надійного та ефективного програмного забезпечення для шифрування та архівації файлів, здатного обробляти великі обсяги даних і витримувати високі вимоги до безпеки та швидкодії [16], [17].

Обґрунтування вибору мови програмування для розробки програмного забезпечення для шифрування та архівації файлів наведено у таблиці 2.1.

Таблиця 2.1 – Обґрунтування вибору мови програмування для розробки програмного забезпечення для шифрування та архівації файлів

Критерій порівняння мов програмування	Мова програмування		
	C++	Python	Go
Продуктивність і швидкодія	+	+-	+-
Контроль над пам'яттю	+	-	+-
Можливості низькорівневого програмування	+	-	+-
Підтримка багатопоточності та паралелізму	+	+-	+
Оптимізація під апаратні ресурси	+	-	+-

Отже, для реалізації програмного забезпечення для шифрування та архівації файлів обрано мову програмування C++ через поєднання високої продуктивності, гнучкості та контролю над ресурсами. Також важливим чинником є наявність потужних бібліотек та фреймворків для C++, які спрощують реалізацію складних криптографічних алгоритмів, підтримку сучасних форматів архівів і шифрів, а також забезпечують сумісність із різними операційними системами, що дозволяє створювати кросплатформні рішення, які зберігають високу швидкодію і надійність.

2.2 Вибір середовища розробки для створення програмного забезпечення для шифрування та архівації файлів

В якості середовища розробки для створення програмного забезпечення для шифрування та архівації файлів було обрано середовище Visual Studio [18], [19].

Середовище розробки Visual Studio вирізняється комплексністю та гнучкістю, що робить його придатним для створення як невеликих програм, так і великих корпоративних проєктів. Воно інтегрує в собі велику кількість інструментів, які значно спрощують написання, редагування та підтримку коду. Інтерфейс середовища розробки Visual Studio забезпечує швидкий доступ до всіх необхідних функцій, включаючи підключення бібліотек, роботу з базами даних, управління ресурсами і автоматичне форматування коду. Це сприяє підвищенню продуктивності розробки та зменшенню часу на виправлення помилок [18], [19].

Середовище розробки Visual Studio надає розвинуту систему налагодження, що дозволяє виявляти помилки на будь-якому етапі виконання програми, перевіряти значення змінних у реальному часі, аналізувати потоки виконання та здійснювати профайлінг продуктивності. Така функціональність особливо цінна при розробці програмного забезпечення для шифрування та архівації файлів, де критичною є точність обчислень і ефективне використання ресурсів [18], [19].

Середовище підтримує інтеграцію з системами контролю версій, що забезпечує безпечну командну роботу та можливість відслідковувати всі зміни у коді. Крім того, Visual Studio надає інструменти для автоматизації збірки та тестування, що підвищує стабільність і надійність кінцевих програмних продуктів [18], [19].

Важливою перевагою є підтримка різних мов програмування, а також можливість створення кросплатформних рішень, що дозволяє запускати програмне забезпечення на різних операційних системах без суттєвих змін у

кодi. Середовище також пропонує iнтеграцію з сучасними фреймворками, бiблiотеками та сервісами, що полегшує реалізацію складних алгоритмiв i використання готових рiшень для шифрування, архiвації та роботи з великими обсягами даних [18], [19].

Отже, Visual Studio забезпечує не лише комфортне написання та організацію коду, але й потужні інструменти для налагодження, оптимізації, тестування та командної роботи, роблячи процес розробки ефективним i надійним [18], [19].

Використання середовища розробки Visual Studio для створення програмного забезпечення для шифрування та архiвації файлiв виявляється доцільним через комплекс переваг, що підвищують ефективність розробки та якість кінцевого продукту. По-перше, Visual Studio забезпечує iнтегроване середовище, яке об'єднує редактор коду, компілятор, налагоджувач i профайлери продуктивності, що дозволяє розробляти складні програми на C++ iз високою швидкістю та точністю. Це особливо важливо для програм, де критичною є продуктивність обчислень, як у випадку шифрування великих обсягiв даних або архiвації великих файлiв [18], [19].

По-друге, Visual Studio надає зручні інструменти для управління проєктом i структурою коду. Автоматичне підключення бiблiотек, система IntelliSense, підказки щодо синтаксису та помилок дозволяють зменшити кількість програмних помилок i прискорити написання коду. Така організація роботи особливо важлива при реалізації складних алгоритмiв шифрування, де навіть незначна помилка може призвести до втрати доступу до даних або зниження безпеки [18], [19].

Середовище розробки Visual Studio також підтримує потужні засоби налагодження та тестування, що дозволяє відслідковувати роботу програми на різних етапах її виконання. Можливість використання точок зупинки, аналізу стеку викликiв i профайлінгу ресурсiв дає змогу оптимізувати алгоритми шифрування i архiвації, підвищуючи швидкодію i надійність програмного забезпечення [18], [19].

Важливим аспектом є підтримка кросплатформної розробки і інтеграція з сучасними бібліотеками та фреймворками, що дозволяє створювати універсальні рішення, здатні працювати на різних операційних системах і використовувати сучасні криптографічні стандарти. Крім того, Visual Studio забезпечує просту інтеграцію з системами контролю версій і автоматизації збірки, що робить роботу над проєктом більш структурованою і прозорою, особливо для командної розробки [18], [19].

Таким чином, застосування Visual Studio доцільне завдяки її здатності підвищувати продуктивність розробника, забезпечувати точний контроль над кодом, сприяти оптимізації обчислень і полегшувати створення надійного, безпечного і високоефективного програмного забезпечення для шифрування та архівації файлів [18], [19].

Обґрунтування вибору середовища розробки для створення програмного забезпечення для шифрування та архівації файлів наведено у таблиці 2.2.

Таблиця 2.2 – Обґрунтування вибору середовища розробки для створення програмного забезпечення для шифрування та архівації файлів

Критерій порівняння середовищ розробки	Середовища розробки		
	Visual Studio	CLion	Dev-C++
Інтегрованість інструментів	+	+-	-
Зручність налагодження	+	+-	+-
Підтримка бібліотек та фреймворків	+	+	+-
Автоматизація збірки та управління проєктом	+	+-	-
Інтелектуальні підказки та автодоповнення	+	+	+-

Таким чином, для створення програмного забезпечення для шифрування та архівації файлів обрано середовище розробки Visual Studio, яке об'єднує редактор коду, компілятор, налагоджувач і профайлери продуктивності, що дозволяє розробляти складні програми на C++ із високою швидкістю та точністю. Крім того, Visual Studio надає зручні інструменти для управління проєктом і структурою коду, зокрема, автоматичне підключення бібліотек, та підказки щодо синтаксису та помилок, що дозволяють зменшити кількість програмних помилок і прискорити написання коду.

2.3 Висновки за розділом 2

Для реалізації програмного забезпечення для шифрування та архівації файлів обрано мову програмування C++ через поєднання високої продуктивності, гнучкості та контролю над ресурсами. Також важливим чинником є наявність потужних бібліотек та фреймворків для C++, які спрощують реалізацію складних криптографічних алгоритмів, підтримку сучасних форматів архівів і шифрів, а також забезпечують сумісність із різними операційними системами, що дозволяє створювати кросплатформні рішення, які зберігають високу швидкодію і надійність.

Для створення програмного забезпечення для шифрування та архівації файлів обрано середовище розробки Visual Studio, яке об'єднує редактор коду, компілятор, налагоджувач і профайлери продуктивності, що дозволяє розробляти складні програми на C++ із високою швидкістю та точністю. Крім того, Visual Studio надає зручні інструменти для управління проєктом і структурою коду, зокрема, автоматичне підключення бібліотек, та підказки щодо синтаксису та помилок, що дозволяють зменшити кількість програмних помилок і прискорити написання коду.

3 ОПИС ПРОГРАМИ

3.1 Структура програмного забезпечення для шифрування та архівації файлів

Структуру програмного забезпечення для шифрування та архівації файлів наведено на рис. 3.1.

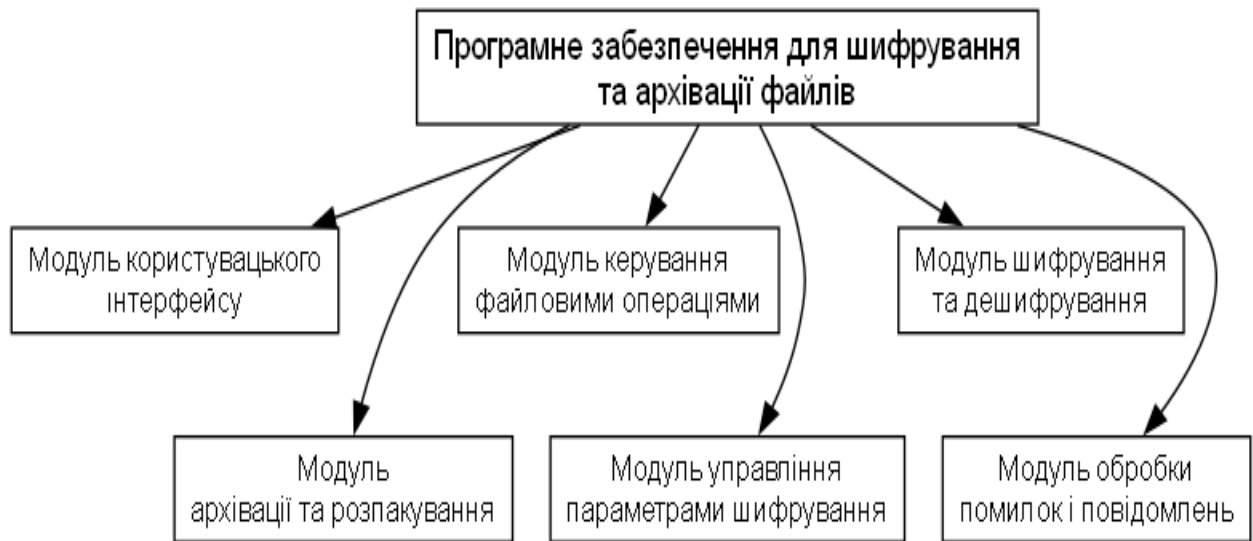


Рисунок 3.1 – Структура програмного забезпечення для шифрування та архівації файлів

Структура програмного забезпечення для шифрування та архівації файлів створена як сукупність взаємопов'язаних функціональних модулів: модуль користувацького інтерфейсу, модуль керування файловими операціями, модуль шифрування та дешифрування, модуль архівації та розпакування, модуль управління параметрами шифрування, модуль обробки помилок і повідомлень. Кожен з модулів відповідає за окремий аспект обробки даних, забезпечуючи при цьому цілісність, безпеку та зручність взаємодії з користувачем.

У центрі такої системи знаходиться модуль користувацького інтерфейсу, реалізований у вигляді головної форми, через яку здійснюється взаємодія з усіма іншими компонентами. Саме в цьому модулі відображаються

поля для введення шляхів до файлів, параметрів шифрування, а також кнопки для запуску процесів шифрування і дешифрування. Він забезпечує введення даних, їх первинну перевірку, маскуванню ключа у вигляді символів та відображення повідомлень про результати виконання операцій.

Логіка обробки файлів зосереджується у модулі керування файловими операціями, який відповідає за відкриття, читання, запис і збереження як початкових, так і оброблених файлів. У цьому компоненті реалізується робота з файловою системою, перевірка існування файлів, їх перезапис у разі необхідності, а також забезпечення коректної передачі даних між іншими модулями. Завдяки цьому досягається узгодженість роботи програми при взаємодії з різними типами файлів незалежно від їхнього розміру та формату.

Ключову роль у забезпеченні захисту інформації відіграє модуль шифрування та дешифрування, який реалізує алгоритми перетворення даних із використанням заданого ключа та розміру блоку. У цьому модулі здійснюється безпосередня обробка інформації, розбиття її на блоки відповідного розміру, застосування криптографічних операцій і формування шифротексту. У зворотному процесі цей же компонент відповідає за відновлення початкових даних за умови коректного введення всіх параметрів. Саме тут реалізується основний механізм захисту, оскільки навіть незначне відхилення у значеннях ключа або розміру блоку унеможлиблює відновлення інформації.

Паралельно функціонує модуль архівації та розпакування, який відповідає за стиснення даних перед їх шифруванням або після нього, залежно від обраної логіки роботи програми. Цей компонент забезпечує зменшення обсягу файлів, оптимізує їх зберігання і передачу, а також дозволяє об'єднувати декілька файлів в єдину структуру. У процесі розпакування він виконує зворотні дії, відновлюючи первинний стан даних після їх дешифрування.

Окреме місце займає модуль управління параметрами шифрування, який відповідає за обробку введених користувачем значень розміру блоку та

ключа, а також за їх перевірку на коректність. У цьому компоненті забезпечується узгодженість параметрів, їх передача до криптографічного ядра, а також контроль за тим, щоб введені дані відповідали вимогам алгоритму. Саме цей модуль відіграє важливу роль у забезпеченні правильності виконання операцій і попередженні помилок, що можуть призвести до втрати доступу до інформації.

Завершує структуру модуль обробки помилок і повідомлень, який забезпечує контроль за всіма етапами виконання програми та інформує користувача про результати операцій. У разі некоректного введення параметрів або виникнення збоїв він формує відповідні повідомлення, що дозволяє своєчасно реагувати на помилки та запобігати неправильному використанню програмного забезпечення.

Таким чином, описана структура програмного забезпечення для шифрування та архівації файлів характеризується чітким розподілом функцій між модулями, що забезпечує ефективну обробку даних, високий рівень безпеки та зручність використання. Взаємодія всіх компонентів дозволяє реалізувати надійний механізм захисту інформації та оптимізувати процеси її зберігання і передачі.

3.2 Функціонування програмного забезпечення для шифрування та архівації файлів

Функціонування програмного забезпечення для шифрування та архівації файлів базується на взаємодії користувача з головною формою, яка виступає центральним елементом керування всіма процесами обробки даних. Саме через цей інтерфейс здійснюється вибір вихідного файлу, налаштування параметрів безпеки та запуск відповідних операцій. Програма орієнтована на роботу з довільними файлами, розташованими на комп'ютері користувача, причому шлях до них може задаватися як шляхом введення, так і за допомогою спеціального елемента вибору. Такий підхід забезпечує

універсальність і зручність у використанні незалежно від типу чи розміру даних.

Функціонування програмного забезпечення для шифрування та архівації файлів подамо за допомогою схеми, зображеної на рис. 3.2.

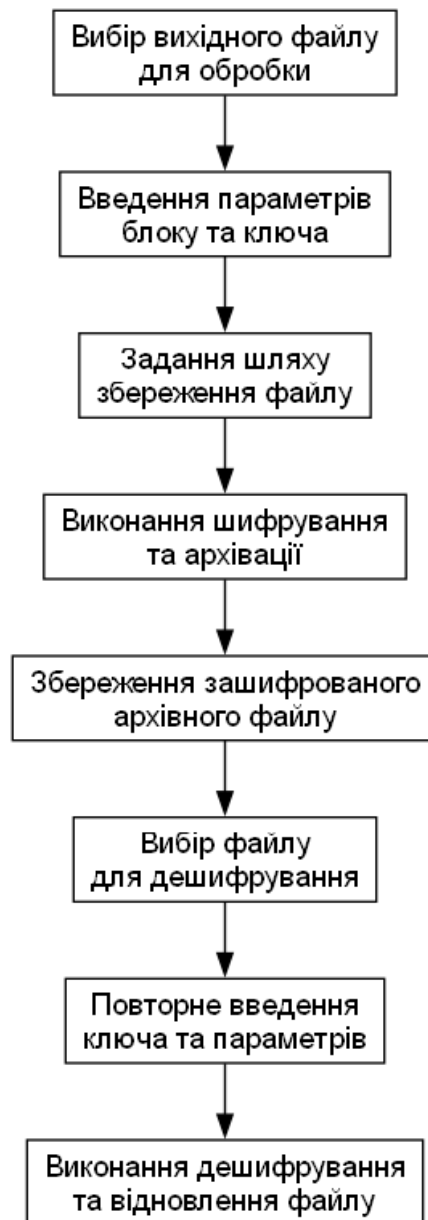


Рисунок 3.2 – Схема функціонування програмного забезпечення для шифрування та архівації файлів

У процесі підготовки до шифрування особлива увага приділяється встановленню параметрів, які визначають подальшу обробку інформації. До них належать розмір блоку та характеристика ключа, а також саме значення

ключа шифрування. Введення ключа супроводжується його приховуванням у вигляді символів, що підвищує рівень захисту під час взаємодії з програмою. Після зазначення всіх необхідних параметрів і вибору файлу призначення, у який буде записано результат, запускається процедура перетворення даних. У ході виконання цієї операції інформація обробляється відповідно до встановлених налаштувань, у результаті чого формується зашифрований файл, який зберігається у визначеному користувачем місці. У випадку, якщо файл із таким іменем уже існує, відбувається його заміна, що забезпечує оновлення даних без створення дублюючих копій.

Процес відновлення початкової інформації організовано аналогічним чином, проте вимагає точного відтворення параметрів, які використовувалися під час шифрування. Для цього необхідно обрати зашифрований файл, задати нове місце для збереження результату та повторно вказати значення розміру блоку і ключа. Важливим аспектом є те, що навіть незначна невідповідність цих параметрів унеможливорює коректне виконання процедури дешифрування. Таким чином, механізм перевірки правильності введених даних виступає своєрідним бар'єром, який запобігає доступу сторонніх осіб до захищеної інформації. Після запуску відповідної операції здійснюється зворотне перетворення даних, і у разі успішного завершення користувач отримує відновлений файл у зазначеному місці.

Робота програми супроводжується інформаційними повідомленнями, які сигналізують про завершення процесів або можливі помилки. Це дозволяє контролювати стан виконання операцій та своєчасно реагувати на некоректні дії. Завдяки поєднанню зручного інтерфейсу, чітко визначених параметрів обробки та механізмів контролю доступу забезпечується надійне функціонування програмного засобу.

Отже, функціонування програмного забезпечення для шифрування та архівації файлів характеризується логічною послідовністю дій, що охоплює вибір даних, налаштування параметрів, їх обробку та отримання результату. Вбудований механізм перевірки коректності введених параметрів гарантує

захист інформації, а організація взаємодії через єдину форму забезпечує зручність і зрозумілість використання програми.

3.3 Структура внутрішніх даних. Опис механізму збереження і організації файлів

У центрі структури внутрішніх даних програмного забезпечення для шифрування та архівації файлів знаходяться об'єкти, що зберігають інформацію про вибрані користувачем файли, включно з їхнім шляхом на диску, розміром та іншими метаданими, необхідними для коректної обробки. Кожен файл представлено як окремий запис у пам'яті програми, який взаємодіє з компонентами, відповідальними за шифрування та архівацію, забезпечуючи контроль над станом даних у режимі реального часу. Ці об'єкти містять в собі параметри обробки, такі як розмір блоку та довжина ключа, а також сам ключ у зашифрованому або тимчасово прихованому вигляді, що дозволяє гарантувати безпеку інформації навіть під час активної взаємодії з інтерфейсом користувача.

Для управління операціями з файлами та забезпечення послідовності обробки застосовується додатковий рівень внутрішніх структур, який відповідає за логіку черги операцій і взаємозв'язок між модулями. Це дозволяє підтримувати узгодженість виконання дій, наприклад, від вибору файлу до завершення шифрування або дешифрування, і уникати конфліктів між одночасними запитами на обробку різних файлів. Крім того, в пам'яті утримуються тимчасові дані, що використовуються для відстеження стану кожної операції, включно з інформацією про успішність виконання та наявність помилок, що забезпечує можливість реалізації функціоналу повідомлень і підказок для користувача.

Схему структури внутрішніх даних програмного забезпечення для шифрування та архівації файлів наведено на рис. 3.3.

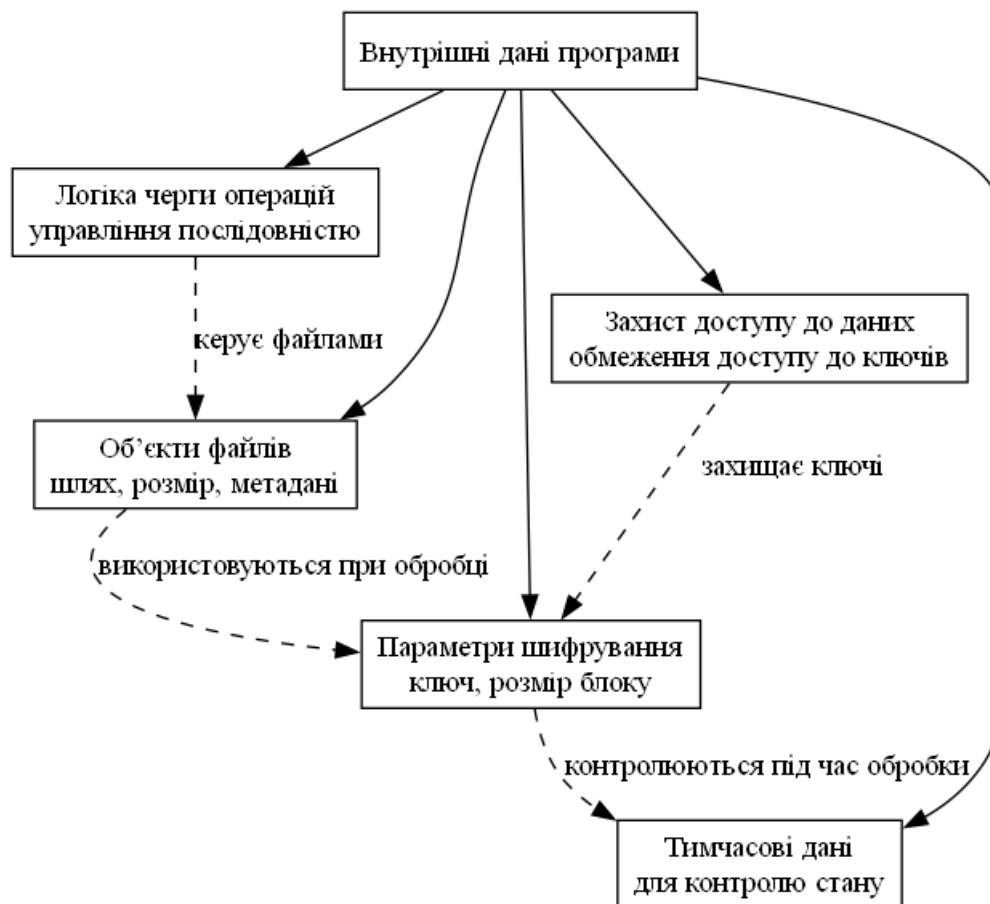


Рисунок 3.3 – Схема структури внутрішніх даних програмного забезпечення для шифрування та архівації файлів

Структура внутрішніх даних також передбачає наявність механізмів для контролю безпеки. Ключі шифрування не зберігаються у відкритому вигляді довше необхідного часу, а доступ до метаданих і тимчасових структур обмежено лише логікою програмних модулів, що забезпечує захист від несанкціонованого доступу та випадкових змін. Усі ці елементи взаємопов'язані таким чином, щоб забезпечити ефективну роботу системи, мінімізувати ризик втрати даних і підтримувати стабільність під час виконання операцій з великими файлами.

Отже, структура внутрішніх даних у програмному забезпеченні для шифрування та архівації файлів є багаторівневою, оскільки вона не лише зберігає параметри файлів і ключів, а й забезпечує управління чергою операцій, контроль стану процесів та захист інформації. Така організація даних створює надійну основу для стабільного та безпечного функціонування

програми, забезпечуючи коректність обробки та збереження файлів користувача.

3.4 Проєктування інтерфейсу програмного забезпечення для шифрування та архівації файлів

Проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для шифрування та архівації файлів виконано з урахуванням вимог технічного завдання. При розробленні програмного забезпечення для шифрування та архівації файлів враховані функціональні вимоги до програми:

- підтримка можливості вибору на комп'ютері для подальшого шифрування;
- забезпечення можливості введення і збереження ключа шифрування разом з параметрами блоку для захисту файлів;
- підтримка можливості створення зашифрованого файлу з вказанням користувачем імені та шляху збереження;
- забезпечення можливості відновлення оригінального файлу за допомогою правильного ключа та параметрів блоку;
- підтримка захисту даних від несанкціонованого доступу шляхом перевірки правильності ключа і параметрів блоку перед дешифруванням;
- інтеграція графічного інтерфейсу користувача для зручного управління процесами шифрування, дешифрування та вибору файлів.

Інтерфейс форми шифрування файлів наведено на рис. 3.4.

Інтерфейс форми дешифрування файлів наведено на рис. 3.5.

The dialog box titled "Шифрування даних" (Data Encryption) contains the following elements:

- Файл для шифрування** (File for encryption): A text input field containing "File1.txt".
- Відкрити** (Open): A button next to the file input field.
- Назва файлу після зашифровки** (File name after encryption): A text input field containing "File1_crypt.txt".
- Зашифрувати** (Encrypt): A button next to the output file name field.
- Розмір блоку** (Block size): A group box containing three radio buttons: 128, 256 (selected), and 512.
- Розмір ключа** (Key size): A group box containing three radio buttons: 256 (selected), 512, and 128.
- Ключ** (Key): A label next to a password input field filled with asterisks.

Рисунок 3.4 – Інтерфейс форми шифрування файлів

The dialog box titled "Дешифрування даних" (Data Decryption) contains the following elements:

- Файл для дешифрування** (File for decryption): An empty text input field.
- Відкрити** (Open): A button next to the file input field.
- Назва файлу після розифровки** (File name after decryption): An empty text input field.
- Дешифрувати** (Decrypt): A button next to the output file name field.
- Розмір блоку** (Block size): A group box containing three radio buttons: 128, 256 (selected), and 512.
- Розмір ключа** (Key size): A group box containing three radio buttons: 256 (selected), 512, and 128.
- Ключ** (Key): A label next to a password input field filled with asterisks.

Рисунок 3.5 – Інтерфейс форми шифрування дефайлів

3.5 Висновки за розділом 3

Розроблено програмне забезпечення для шифрування та архівації файлів.

Запропоновано структуру програмного забезпечення для шифрування та архівації файлів. Структура програмного забезпечення для шифрування та архівації файлів створена як сукупність взаємопов'язаних функціональних модулів: модуль користувацького інтерфейсу, модуль керування файловими операціями, модуль шифрування та дешифрування, модуль архівації та розпакування, модуль управління параметрами шифрування, модуль обробки помилок і повідомлень. Кожен з модулів відповідає за окремий аспект обробки даних, забезпечуючи при цьому цілісність, безпеку та зручність взаємодії з користувачем.

Описано функціонування програмного забезпечення для шифрування та архівації файлів, що характеризується логічною послідовністю дій, що охоплює вибір даних, налаштування параметрів, їх обробку та отримання результату. Вбудований механізм перевірки коректності введених параметрів гарантує захист інформації, а організація взаємодії через єдину форму забезпечує зручність і зрозумілість використання програми.

Описано структуру внутрішніх даних у програмному забезпеченні для шифрування та архівації файлів. Визначено, що структура внутрішніх даних програми є багаторівневою, оскільки вона не лише зберігає параметри файлів і ключів, а й забезпечує управління чергою операцій, контроль стану процесів та захист інформації. Така організація даних створює надійну основу для стабільного та безпечного функціонування програми, забезпечуючи коректність обробки та збереження файлів користувача.

Виконано проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для шифрування та архівації файлів.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення й умови застосування програми

Програма призначена для шифрування та архівації файлів. Програма написана на мові C++ через поєднання високої продуктивності, гнучкості та контролю над ресурсами. В якості середовища розробки для створення програмного забезпечення для шифрування та архівації файлів обрано Visual Studio, яке об'єднує редактор коду, компілятор, налагоджувач і профайлери продуктивності, що дозволяє розробляти складні програми на C++ із високою швидкістю та точністю.

Програмне забезпечення для шифрування та архівації файлів забезпечує виконання таких функцій:

- підтримка можливості вибору на комп'ютері для подальшого шифрування;
- забезпечення можливості введення і збереження ключа шифрування разом з параметрами блоку для захисту файлів;
- підтримка можливості створення зашифрованого файлу з вказанням користувачем імені та шляху збереження;
- забезпечення можливості відновлення оригінального файлу за допомогою правильного ключа та параметрів блоку;
- підтримка захисту даних від несанкціонованого доступу шляхом перевірки правильності ключа і параметрів блоку перед дешифруванням;
- інтеграція графічного інтерфейсу користувача для зручного управління процесами шифрування, дешифрування та вибору файлів.

Для роботи програмного забезпечення для шифрування та архівації файлів обчислювальна система повинна забезпечувати достатній рівень продуктивності для виконання операцій обробки даних, що включають інтенсивні обчислення, пов'язані з криптографічними перетвореннями та стисненням інформації. Центральний процесор повинен бути із тактовою

частотою не нижче двох гігагерців, що дозволяє забезпечити прийнятний час виконання операцій навіть при роботі з файлами значного обсягу. Багатоядерна архітектура процесора сприяє підвищенню ефективності обробки за рахунок можливості паралельного виконання окремих етапів шифрування та архівації.

Для стабільного функціонування програми для шифрування та архівації файлів доцільно мати не менше восьми гігабайтів оперативної пам'яті, що дозволяє уникнути перевантаження системи та забезпечує безперервність обробки навіть великих файлів. Додатково необхідно передбачити достатній обсяг вільного дискового простору (хоча б сто гігабайтів), який використовується для збереження зашифрованих або розшифрованих файлів, а також для тимчасових даних, що створюються у процесі роботи програми.

Функціонування програми для шифрування та архівації файлів передбачає наявність операційної системи, яка підтримує роботу з файловою системою, управління пам'яттю та виконання прикладних програм. Важливою умовою є також наявність необхідних системних бібліотек і середовища виконання, що забезпечують коректну роботу реалізованих алгоритмів шифрування та архівації. У випадку використання зовнішніх бібліотек для реалізації криптографічних функцій або стиснення даних необхідно забезпечити їх встановлення та сумісність із середовищем виконання програми.

Крім того, для повноцінного використання програмного забезпечення для шифрування та архівації файлів потрібні стандартні засоби взаємодії з користувачем, такі як клавіатура та пристрій вказування, що дозволяють вводити параметри шифрування, обирати файли та керувати процесом обробки. Графічне середовище операційної системи забезпечує відображення інтерфейсу програми та повідомлень про стан виконання операцій.

Функціональні характеристики розробленого програмного забезпечення для шифрування та архівації файлів наведено у технічному

завданні (додаток А). Фрагмент тексту програмного забезпечення для шифрування та архівації файлів наведено у додатку Б.

4.2 Характеристики програми для шифрування та архівації файлів

Програмне забезпечення для шифрування та архівації файлів характеризується поєднанням функціональних і технічних властивостей, що забезпечують ефективну обробку, збереження та захист інформації. Основною його особливістю є здатність перетворювати вміст файлів у недоступний для стороннього перегляду вигляд із подальшою можливістю відновлення за умови правильного введення параметрів. Такий підхід дозволяє гарантувати конфіденційність даних незалежно від середовища їх зберігання або передачі.

Важливою характеристикою програми для шифрування та архівації файлів є гнучкість у налаштуванні параметрів обробки, що дає змогу адаптувати роботу програми до різних вимог користувача. Передбачено введення ключа та вибір параметрів, які визначають спосіб перетворення даних, при цьому сам ключ захищається під час введення шляхом прихованого відображення. Такий механізм підвищує рівень безпеки і зменшує ризик несанкціонованого доступу до критичних даних.

Програмний засіб відзначається здатністю працювати з файлами різних типів без обмежень щодо їхнього формату, що забезпечує універсальність застосування. Організація обробки даних передбачає послідовне виконання операцій, у результаті чого формується новий файл, який містить зашифровану або стиснену інформацію. При цьому зберігається можливість керування місцем розташування та іменем результату, що забезпечує зручність використання і контроль над збереженими даними.

Окремою характеристикою програми для шифрування та архівації файлів є надійність функціонування, яка досягається завдяки перевірці введених параметрів і контролю правильності виконання операцій. У випадку

невідповідності заданих значень відновлення інформації стає неможливим, що виступає додатковим елементом захисту. Водночас програма для шифрування та архівації файлів забезпечує інформування користувача про стан виконання операцій, що дозволяє своєчасно реагувати на помилки та контролювати процес обробки.

Крім того, програмний продукт для шифрування та архівації файлів характеризується ефективністю використання системних ресурсів, що дозволяє виконувати операції шифрування та архівації без суттєвого навантаження на обчислювальну систему. Це досягається завдяки оптимізації алгоритмів обробки даних і раціональному використанню пам'яті та обчислювальних можливостей.

Отже, програмне забезпечення для шифрування та архівації файлів поєднує у собі властивості безпеки, універсальності, гнучкості та надійності, що забезпечує ефективне управління інформацією та її захист у різних умовах використання.

4.3 Інструкція по експлуатації програми

4.3.1 Звернення до програми

Для запуску програмного забезпечення для шифрування та архівації файлів необхідно запустити відповідний виконуваний файл.

4.3.2 Вхідні й вихідні дані

Вхідними даними до програмного забезпечення для шифрування та архівації файлів є файл, обраний користувачем для подальшої обробки, параметри, що визначають особливості перетворення інформації, зокрема розмір ключа шифрування, що використовується для забезпечення конфіденційності, а також характеристики, пов'язані з організацією обробки даних. У процесі дешифрування вхідною інформацією виступає вже

зашифрований файл разом із тими самими параметрами, що були застосовані під час його створення, оскільки без їх точного відтворення коректне відновлення початкового вмісту стає неможливим.

Вихідними даними програмного забезпечення для шифрування та архівації файлів є сформований у результаті обробки файл, який містить зашифровану або стиснену інформацію та зберігається у визначеному користувачем місці. У разі виконання зворотної операції результатом є відновлений файл, що відповідає початковому вмісту за умови правильного введення параметрів.

4.3.3 Повідомлення

Користувач програмного забезпечення для шифрування та архівації файлів може отримати такі повідомлення: повідомлення про успішне завершення процесу шифрування або дешифрування; у разі спроби перезапису існуючого файлу система може інформувати про це користувача, дозволяючи усвідомлено прийняти рішення щодо подальших дій.

4.4 Виконання програмного забезпечення для шифрування та архівації файлів

Після запуску програмного забезпечення для шифрування та архівації файлів користувачу відображається головна форма (рис. 4.1), де необхідно обрати один з двох режимів роботи програми – шифрування або дешифрування.

Форма шифрування (рис. 4.2) забезпечує доступ до функціоналу програми, пов'язаного з шифруванням та архівацією файлів, включаючи вибір файлу для обробки, його перетворення у захищений формат. Інтерфейс дозволяє користувачу керувати процесом шифрування, задаючи необхідні налаштування перед початком операцій.

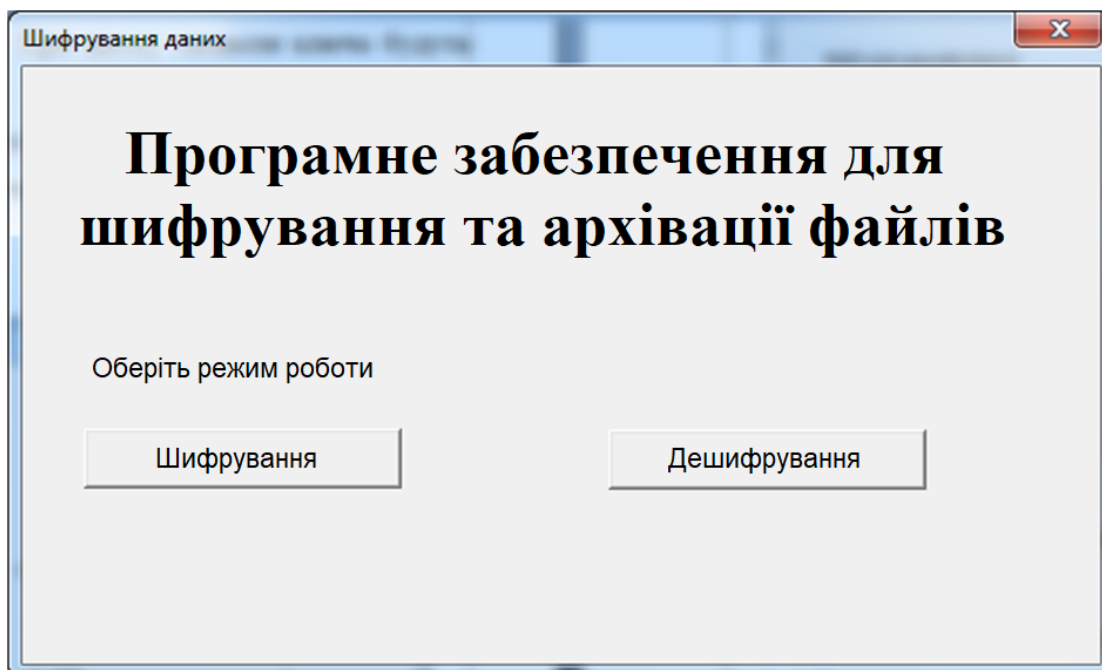


Рисунок 4.1 – Інтерфейс головної форми програмного забезпечення для шифрування та архівації файлів

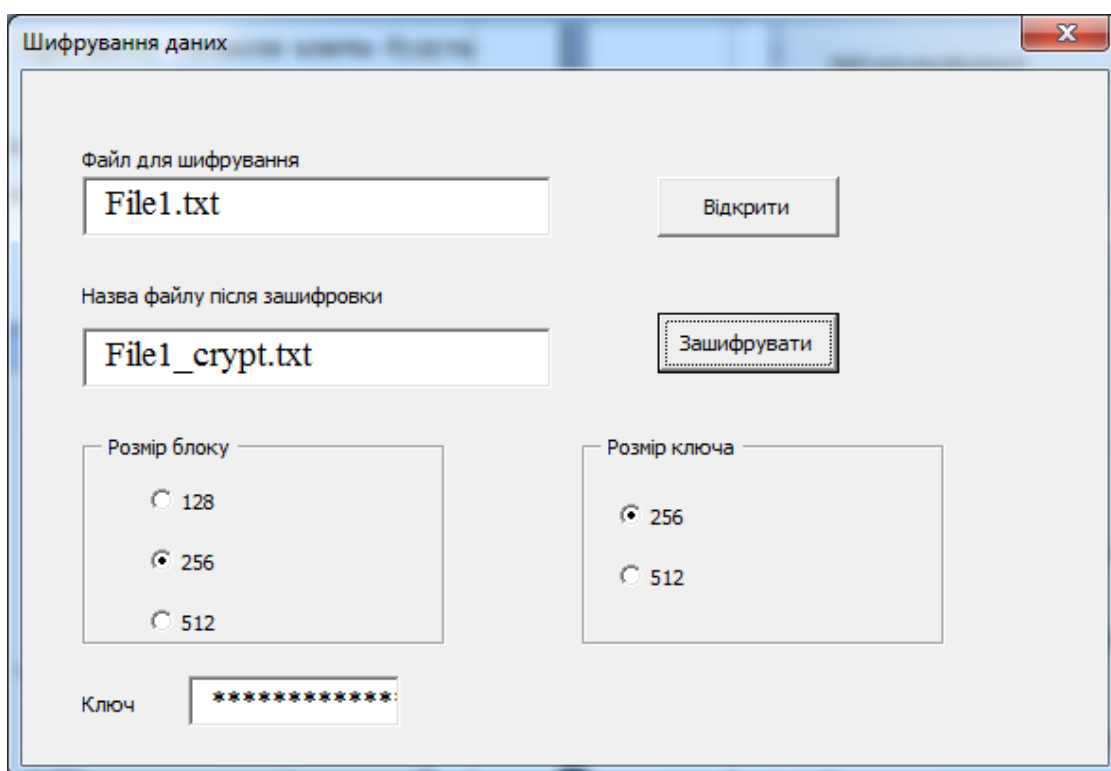


Рисунок 4.2 – Форма шифрування та архівації файлів

Процедура шифрування файлу передбачає попереднє відкриття вихідного документа шляхом використання відповідної кнопки або введення

його повного шляху безпосередньо у поле введення. Після цього задається ім'я файлу для збереження зашифрованих даних. Далі встановлюються параметри обробки, включно з розміром блоку і довжиною ключа шифрування, та вводиться сам ключ у спеціальне поле, де його символи приховуються за допомогою зірочок. Після завершення налаштувань користувач запускає процес шифрування і чекає на повідомлення про успішне створення зашифрованого файлу.

Дешифрування зашифрованого файлу (рис. 4.3) відбувається аналогічним чином, з тією різницею, що обробляється вже захищений файл. Відкриття здійснюється через відповідну кнопку, після чого задається шлях для збереження розшифрованого документа, який при збігу імені перезаписує попередній. Користувач встановлює розмір блоку та ключ, вводить сам ключ, і після цього запускає процедуру відновлення даних, чекаючи на повідомлення про завершення дешифрування. У випадку введення некоректного значення будь-якого параметра процес відновлення не відбувається, що гарантує захист інформації від несанкціонованого доступу.

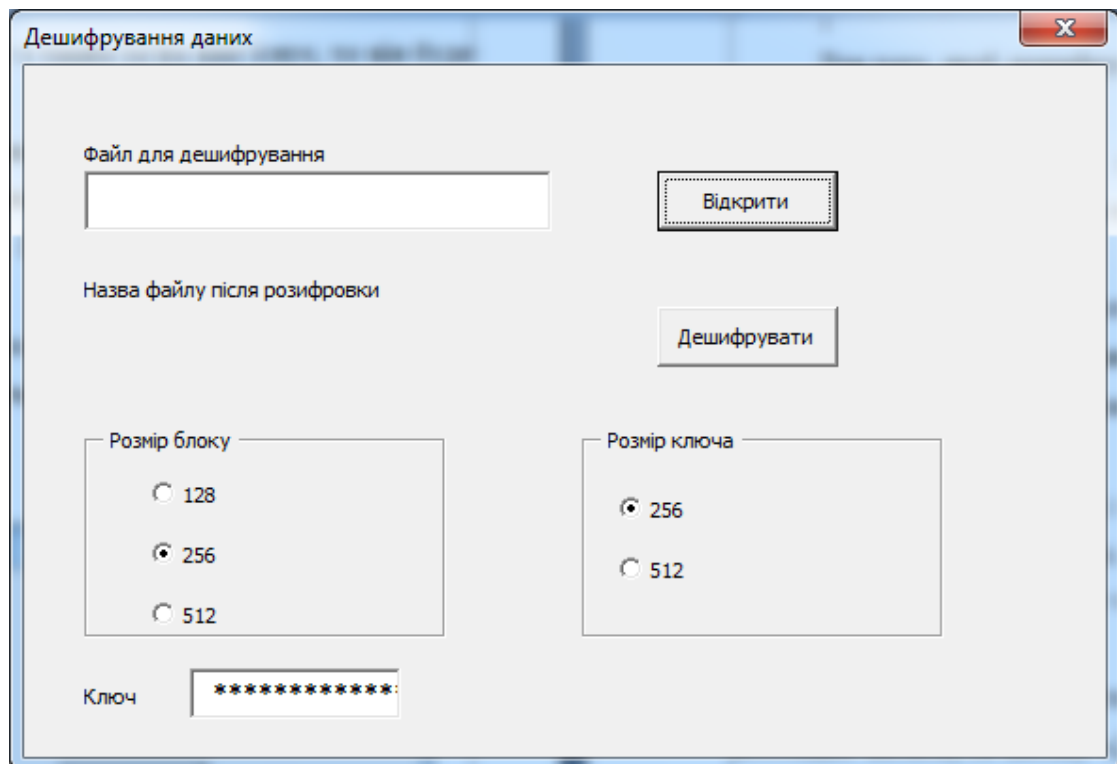


Рисунок 4.3 – Форма дешифрування файлів

4.5 Тестування програмного забезпечення для шифрування та архівації файлів

Виконано тестування програмного забезпечення для шифрування та архівації файлів.

Виявлено, що програма для шифрування та архівації файлів функціонує правильно та злагоджено. Вона реалізує всі функціональні вимоги та успішно виконує свою основну задачу шифрування та архівації файлів.

Розроблене програмне забезпечення дозволяє забезпечувати автоматизацію процесів, пов'язаних з підтримкою процесу шифрування та архівації файлів.

4.6 Висновки за розділом 4

Описано програмне забезпечення для шифрування та архівації файлів. Виконано тестування розробленого програмного забезпечення для шифрування та архівації файлів. Результати тестування програмного забезпечення показали, що розроблена програма дозволяє забезпечити автоматизацію процесів, пов'язаних з підтримкою процесу шифрування та архівації файлів.

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процес розробки програмного забезпечення для шифрування та архівації файлів.

Визначено, що програмне забезпечення для шифрування та архівації файлів являє собою клас прикладних систем, призначених для забезпечення захисту інформації та її впорядкованого зберігання. Його сутність полягає у перетворенні даних у такий вигляд, який стає недоступним для несанкціонованого перегляду, а також у об'єднанні окремих файлів у компактні структуровані набори, що спрощують їх передачу, збереження та обробку.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато програмних засобів для шифрування та архівації файлів. Проте деякі програмні засоби для шифрування та архівації файлів мають низку недоліків, пов'язаних із складністю використання, підвищеними вимогами до ресурсів, ризиком втрати доступу до даних, проблемами сумісності та потенційними вразливостями. Попри значну користь і широке застосування, програмне забезпечення для шифрування та архівації файлів не позбавлене певних обмежень, які можуть впливати на ефективність його використання. У процесі експлуатації подібних рішень виявляється, що підвищення рівня захисту інформації часто супроводжується ускладненням доступу до неї для самого користувача. У разі втрати пароля або ключа дешифрування доступ до зашифрованих даних може бути безповоротно втрачений, що створює суттєві ризики, особливо за відсутності належної системи резервного копіювання. Також спостерігається збільшення навантаження на обчислювальні ресурси, оскільки процеси шифрування і розшифрування потребують додаткового часу та потужностей. Це може бути критичним у випадках обробки великих обсягів інформації або роботи на пристроях із обмеженими ресурсами. У свою чергу, архівація, хоча й зменшує

розмір файлів, інколи потребує додаткового часу на стиснення та розпакування, що може знижувати оперативність доступу до даних. Окремої уваги заслуговує складність налаштування та використання таких програм для непідготовлених користувачів. Неправильне застосування параметрів шифрування або архівації може призвести до зниження рівня безпеки або втрати даних. Тому актуальною є розробка програмного забезпечення для шифрування та архівації файлів.

Сформульовано функціональні вимоги до програмного забезпечення для шифрування та архівації файлів.

Для реалізації програмного забезпечення для шифрування та архівації файлів обрано мову програмування C++ через поєднання високої продуктивності, гнучкості та контролю над ресурсами. Також важливим чинником є наявність потужних бібліотек та фреймворків для C++, які спрощують реалізацію складних криптографічних алгоритмів, підтримку сучасних форматів архівів і шифрів, а також забезпечують сумісність із різними операційними системами, що дозволяє створювати кросплатформні рішення, які зберігають високу швидкодію і надійність.

Для створення програмного забезпечення для шифрування та архівації файлів обрано середовище розробки Visual Studio, яке об'єднує редактор коду, компілятор, налагоджувач і профайлери продуктивності, що дозволяє розробляти складні програми на C++ із високою швидкістю та точністю. Крім того, Visual Studio надає зручні інструменти для управління проєктом і структурою коду, зокрема, автоматичне підключення бібліотек, та підказки щодо синтаксису та помилок, що дозволяють зменшити кількість програмних помилок і прискорити написання коду.

Розроблено програмне забезпечення для шифрування та архівації файлів.

Запропоновано структуру програмного забезпечення для шифрування та архівації файлів. Структура програмного забезпечення для шифрування та архівації файлів створена як сукупність взаємопов'язаних функціональних

модулів: модуль користувацького інтерфейсу, модуль керування файловими операціями, модуль шифрування та дешифрування, модуль архівації та розпакування, модуль управління параметрами шифрування, модуль обробки помилок і повідомлень. Кожен з модулів відповідає за окремий аспект обробки даних, забезпечуючи при цьому цілісність, безпеку та зручність взаємодії з користувачем.

Описано функціонування програмного забезпечення для шифрування та архівації файлів, що характеризується логічною послідовністю дій, що охоплює вибір даних, налаштування параметрів, їх обробку та отримання результату. Вбудований механізм перевірки коректності введених параметрів гарантує захист інформації, а організація взаємодії через єдину форму забезпечує зручність і зрозумілість використання програми.

Описано структуру внутрішніх даних у програмному забезпеченні для шифрування та архівації файлів. Визначено, що структура внутрішніх даних програми є багаторівневою, оскільки вона не лише зберігає параметри файлів і ключів, а й забезпечує управління чергою операцій, контроль стану процесів та захист інформації. Така організація даних створює надійну основу для стабільного та безпечного функціонування програми, забезпечуючи коректність обробки та збереження файлів користувача.

Виконано проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для шифрування та архівації файлів.

Виконано тестування розробленого програмного забезпечення для шифрування та архівації файлів. Результати тестування програмного забезпечення показали, що розроблена програма дозволяє забезпечити автоматизацію процесів, пов'язаних з підтримкою процесу шифрування та архівації файлів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. What Is Encryption Software. URL: <https://www.cyberly.org/en/what-is-encryption-software/index.html> (date of access: 07.05.2026).
2. Understanding Encryption Software: Securing Digital Data. URL: <https://www.eunetic.com/en/kb/protection-measures-and-security-tools/encryption-software> (date of access: 07.05.2026).
3. Encryption Software Definition & Detailed Explanation. URL: <https://pcpartsgeek.com/encryption-software/> (date of access: 07.05.2026).
4. How to Make an Encryption Program: 10 Steps Development Guide. URL: <https://www.devteam.space/blog/how-to-build-your-own-file-encryption-software/> (date of access: 07.05.2026).
5. How to Make Your Own Encryption Software. URL: <https://enjoymachinelearning.com/blog/how-do-i-make-my-own-encryption-software/> (date of access: 07.05.2026).
6. Top 10 Encryption Software for Maximum Security. URL: <https://www.tutorialspoint.com/article/top-10-encryption-software-for-maximum-security> (date of access: 07.05.2026).
7. The Most Popular Free Encryption Software Tools. URL: <https://heimdalsecurity.com/blog/free-encryption-software-tools/> (date of access: 07.05.2026).
8. Best Encryption Software I Tried and Tested. URL: <https://learn.g2.com/best-encryption-software> (date of access: 07.05.2026).
9. The Best Encryption Software Choices. URL: <https://www.techrepublic.com/article/encryption-software/> (date of access: 07.05.2026).
10. Best File Encryption Software. URL: <https://www.papermark.com/blog/best-file-encryption-software> (date of access: 07.05.2026).

11. The Best Encryption Software for Sensitive Data: Encrypted Data on The Cloud. URL: <https://www.cloudwards.net/best-encryption-software/> (date of access: 07.05.2026).
12. Best Free Encryption Software. URL: <https://privacy.net/best-free-encryption-software-tools/> (date of access: 07.05.2026).
13. NordLocker. URL: <https://nordlocker.com/> (date of access: 07.05.2026).
14. Virtru. URL: <https://www.virtu.com/data-security-platform/virtu-secure-share> (date of access: 07.05.2026).
15. Cryptomator. URL: <https://cryptomator.org/> (date of access: 07.05.2026).
16. Learn C++. URL: <https://www.learncpp.com/> (date of access: 07.05.2026).
17. C++ Tutorials. URL: <https://cplusplus.com/doc/> (date of access: 07.05.2026).
18. What is Visual Studio? Microsoft's Full IDE for .NET, C++, & Cloud Apps. URL: <https://www.netcomlearning.com/blog/what-is-visual-studio> (date of access: 07.05.2026).
19. Visual Studio documentation. URL: <https://learn.microsoft.com/en-us/visualstudio/windows/?view=visualstudio> (date of access: 07.05.2026).

ДОДАТОК А
Технічне завдання

Вступ

Програмне забезпечення може використовуватися для попередження несанкціонованого доступу до файлів.

А.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Програмне забезпечення для шифрування та архівації файлів», затверджене наказом Національного університету «Запорізька політехніка» № 139 від 7 квітня 2026 р.

А.2 Призначення розробки

Програмний продукт призначений для шифрування та архівації файлів.

А.3 Основні вимоги до програми, що розробляється

А.3.1 Вимоги до функціональних характеристик

Програмне забезпечення для шифрування та архівації файлів забезпечує виконання таких функцій:

- підтримка можливості вибору на комп'ютері для подальшого шифрування;
- забезпечення можливості введення і збереження ключа шифрування разом з параметрами блоку для захисту файлів;
- підтримка можливості створення зашифрованого файлу з вказанням користувачем імені та шляху збереження;
- забезпечення можливості відновлення оригінального файлу за допомогою правильного ключа та параметрів блоку;

- підтримка захисту даних від несанкціонованого доступу шляхом перевірки правильності ключа і параметрів блоку перед дешифруванням;
- інтеграція графічного інтерфейсу користувача для зручного управління процесами шифрування, дешифрування та вибору файлів.

А.3.2 Вимоги до інтерфейсу програми

Інтерфейс програмного забезпечення для шифрування та архівації файлів повинен бути зручним для користувачів. Повинна бути забезпечена можливість роботи в візуальному режимі.

А.3.3 Вимоги до надійності

Програмне забезпечення для шифрування та архівації файлів повинно забезпечити надійне функціонування.

А.3.4 Умови експлуатації

Для експлуатації програмного забезпечення для шифрування та архівації файлів необхідна наявність персонального комп'ютера.

А.3.5 Вимоги до складу та параметрів технічний засобів

Для роботи програмного забезпечення для шифрування та архівації файлів обчислювальна система повинна забезпечувати достатній рівень продуктивності для виконання операцій обробки даних, що включають інтенсивні обчислення, пов'язані з криптографічними перетвореннями та стисненням інформації. Центральний процесор повинен бути із тактовою

частотою не нижче двох гігагерців, що дозволяє забезпечити прийнятний час виконання операцій навіть при роботі з файлами значного обсягу. Багатоядерна архітектура процесора сприяє підвищенню ефективності обробки за рахунок можливості паралельного виконання окремих етапів шифрування та архівації.

Для стабільного функціонування програми для шифрування та архівації файлів доцільно мати не менше восьми гігабайтів оперативної пам'яті, що дозволяє уникнути перевантаження системи та забезпечує безперервність обробки навіть великих файлів. Додатково необхідно передбачити достатній обсяг вільного дискового простору (хоча б сто гігабайтів), який використовується для збереження зашифрованих або розшифрованих файлів, а також для тимчасових даних, що створюються у процесі роботи програми.

Функціонування програми для шифрування та архівації файлів передбачає наявність операційної системи, яка підтримує роботу з файловою системою, управління пам'яттю та виконання прикладних програм. Важливою умовою є також наявність необхідних системних бібліотек і середовища виконання, що забезпечують коректну роботу реалізованих алгоритмів шифрування та архівації. У випадку використання зовнішніх бібліотек для реалізації криптографічних функцій або стиснення даних необхідно забезпечити їх встановлення та сумісність із середовищем виконання програми.

А.3.6 Вимоги до маркування і пакування

Програма для шифрування та архівації файлів може бути записана на будь-якому носії інформації.

На пакуванні повинна бути назва програми – «Програмне забезпечення для шифрування та архівації файлів».

A.4 Вхідні дані до роботи

Вхідними даними до програмного забезпечення для шифрування та архівації файлів є файл, обраний користувачем для подальшої обробки, параметри, що визначають особливості перетворення інформації, зокрема розмір ключа шифрування, що використовується для забезпечення конфіденційності, а також характеристики, пов'язані з організацією обробки даних. У процесі дешифрування вхідною інформацією виступає вже зашифрований файл разом із тими самими параметрами, що були застосовані під час його створення, оскільки без їх точного відтворення коректне відновлення початкового вмісту стає неможливим.

Вихідними даними програмного забезпечення для шифрування та архівації файлів є сформований у результаті обробки файл, який містить зашифровану або стиснену інформацію та зберігається у визначеному користувачем місці. У разі виконання зворотної операції результатом є відновлений файл, що відповідає початковому вмісту за умови правильного введення параметрів.

ДОДАТОК Б
Фрагмент тексту програми

```

void slctsrc()
{
    QString flpth = QFileDialog::getOpenFileName(this);
    if(!flpth.isEmpty())
    {
        txtsrc->setText(flpth);
    }
}

void slctdst()
{
    QString flpth = QFileDialog::getSaveFileName(this);
    if(!flpth.isEmpty())
    {
        txtdst->setText(flpth);
    }
}

QByteArray fncprc(QByteArray dta, QByteArray ky, int blksz)
{
    QByteArray rslt;
    int kysz = ky.size();
    int idx = 0;

    for(int i = 0; i < dta.size(); i += blksz)
    {
        QByteArray blk = dta.mid(i, blksz);
        QByteArray nblk;

        for(int j = 0; j < blk.size(); ++j)
        {
            char ch = blk[j];
            char kch = ky[(idx + j) % kysz];
            char rch = ch ^ kch;
            rch = (rch << 1) | ((rch & 0x80) >> 7);
            nblk.append(rch);
        }

        rslt.append(nblk);
        idx += blksz;
    }

    return rslt;
}

QByteArray fncdpr(QByteArray dta, QByteArray ky, int blksz)
{

```

```

    QByteArray rslt;
    int kysz = ky.size();
    int idx = 0;

    for(int i = 0; i < dta.size(); i += blksz)
    {
        QByteArray blk = dta.mid(i, blksz);
        QByteArray nblk;

        for(int j = 0; j < blk.size(); ++j)
        {
            char ch = blk[j];
            char rch = (ch >> 1) | ((ch & 1) << 7);
            char kch = ky[(idx + j) % kysz];
            char fch = rch ^ kch;
            nblk.append(fch);
        }

        rslt.append(nblk);
        idx += blksz;
    }

    return rslt;
}

QByteArray fncrch(QByteArray dta)
{
    QByteArray rslt;
    int cnt = 1;

    for(int i = 0; i < dta.size(); ++i)
    {
        if(i < dta.size() - 1 && dta[i] == dta[i + 1])
        {
            cnt++;
        }
        else
        {
            rslt.append(dta[i]);
            rslt.append((char)cnt);
            cnt = 1;
        }
    }

    return rslt;
}

QByteArray fncdch(QByteArray dta)

```

```

{
    QByteArray rslt;

    for(int i = 0; i < dta.size(); i += 2)
    {
        char ch = dta[i];
        int cnt = (unsigned char)dta[i + 1];

        for(int j = 0; j < cnt; ++j)
        {
            rslt.append(ch);
        }
    }

    return rslt;
}

void prcenc()
{
    QString src = txtsrc->text();
    QString dst = txtdst->text();
    QByteArray ky = txtkey->text().toUtf8();
    int blkksz = txtblk->text().toInt();

    if(src.isEmpty() || dst.isEmpty() || ky.isEmpty() ||
blkksz <= 0)
    {
        QMessageBox::warning(this, "Error", "Invalid input");
        return;
    }

    QFile fl(src);
    if(!fl.open(QIODevice::ReadOnly))
    {
        QMessageBox::warning(this, "Error", "Cannot open
file");
        return;
    }

    QByteArray dta = fl.readAll();
    fl.close();

    QByteArray enc = fncprc(dta, ky, blkksz);
    QByteArray cmp = fncrch(enc);

    QFile of(dst);
    if(!of.open(QIODevice::WriteOnly))
    {

```

```

        QMessageBox::warning(this, "Error", "Cannot save
file");
        return;
    }

    of.write(cmp);
    of.close();

    QMessageBox::information(this, "Done", "Encryption
complete");
}

void prcdec()
{
    QString src = txtsrc->text();
    QString dst = txtdst->text();
    QByteArray ky = txtkey->text().toUtf8();
    int blksize = txtblk->text().toInt();

    if(src.isEmpty() || dst.isEmpty() || ky.isEmpty() ||
blksize <= 0)
    {
        QMessageBox::warning(this, "Error", "Invalid input");
        return;
    }

    QFile fl(src);
    if(!fl.open(QIODevice::ReadOnly))
    {
        QMessageBox::warning(this, "Error", "Cannot open
file");
        return;
    }

    QByteArray dta = fl.readAll();
    fl.close();

    QByteArray dcmp = fncdch(dta);
    QByteArray dec = fncdpr(dcmp, ky, blksize);

    QFile of(dst);
    if(!of.open(QIODevice::WriteOnly))
    {
        QMessageBox::warning(this, "Error", "Cannot save
file");
        return;
    }
}

```

```

        of.write(dec);
        of.close();

        QMessageBox::information(this, "Done", "Decryption
complete");
    }

    QByteArray fncnrm(QByteArray dta)
    {
        QByteArray rslt;
        for(int i = 0; i < dta.size(); ++i)
        {
            char ch = dta[i];
            if(ch < 32)
            {
                ch = ch + 32;
            }
            rslt.append(ch);
        }
        return rslt;
    }

    QByteArray fncrvs(QByteArray dta)
    {
        QByteArray rslt;
        for(int i = dta.size() - 1; i >= 0; --i)
        {
            rslt.append(dta[i]);
        }
        return rslt;
    }

    QByteArray fncmrg(QByteArray dta1, QByteArray dta2)
    {
        QByteArray rslt;
        int mx = dta1.size() > dta2.size() ? dta1.size() :
dta2.size();

        for(int i = 0; i < mx; ++i)
        {
            if(i < dta1.size())
            {
                rslt.append(dta1[i]);
            }
            if(i < dta2.size())
            {
                rslt.append(dta2[i]);
            }
        }
    }

```

```

        return rslt;
    }

QByteArray fncspl(QByteArray dta)
{
    QByteArray rslt;
    for(int i = 0; i < dta.size(); ++i)
    {
        char ch = dta[i];
        rslt.append(ch);
        rslt.append(ch ^ 0x0F);
    }
    return rslt;
}

QByteArray fncuns(QByteArray dta)
{
    QByteArray rslt;
    for(int i = 0; i < dta.size(); i += 2)
    {
        rslt.append(dta[i]);
    }
    return rslt;
}

QByteArray fncchk(QByteArray dta)
{
    QByteArray rslt;
    int sm = 0;

    for(int i = 0; i < dta.size(); ++i)
    {
        sm += (unsigned char)dta[i];
        sm = sm % 256;
    }

    rslt.append((char)sm);
    return rslt;
}

bool fncvld(QByteArray dta)
{
    if(dta.isEmpty())
    {
        return false;
    }

    int sm = 0;
    for(int i = 0; i < dta.size() - 1; ++i)

```

```

    {
        sm += (unsigned char)dta[i];
        sm = sm % 256;
    }

    return sm == (unsigned char)dta.last();
}

```

```

QByteArray fncadd(QByteArray dta)
{
    QByteArray rslt = dta;
    QByteArray chk = fncchk(dta);
    rslt.append(chk);
    return rslt;
}

```

```

QByteArray fncrmchk(QByteArray dta)
{
    if(dta.size() < 1)
    {
        return QByteArray();
    }
    return dta.left(dta.size() - 1);
}

```

```

QByteArray fncexp(QByteArray dta)
{
    QByteArray rslt;
    for(int i = 0; i < dta.size(); ++i)
    {
        char ch = dta[i];
        rslt.append(ch);
        rslt.append((char)(ch + 1));
        rslt.append((char)(ch - 1));
    }
    return rslt;
}

```

```

QByteArray fnccmpx(QByteArray dta)
{
    QByteArray rslt;
    int cnt = 0;

    for(int i = 0; i < dta.size(); ++i)
    {
        cnt += (unsigned char)dta[i];
        char ch = (char)(cnt % 256);
        rslt.append(ch);
    }
}

```

```

        return rslt;
    }

QByteArray fncdcmpx(QByteArray dta)
{
    QByteArray rslt;
    int prev = 0;

    for(int i = 0; i < dta.size(); ++i)
    {
        int cur = (unsigned char)dta[i];
        char ch = (char)((cur - prev + 256) % 256);
        rslt.append(ch);
        prev = cur;
    }
    return rslt;
}

QByteArray fncfltr(QByteArray dta)
{
    QByteArray rslt;
    for(int i = 0; i < dta.size(); ++i)
    {
        if((unsigned char)dta[i] % 2 == 0)
        {
            rslt.append(dta[i]);
        }
    }
    return rslt;
}

QByteArray fncfltrr(QByteArray dta)
{
    QByteArray rslt;
    for(int i = 0; i < dta.size(); ++i)
    {
        if((unsigned char)dta[i] % 2 != 0)
        {
            rslt.append(dta[i]);
        }
    }
    return rslt;
}

QByteArray fncshft(QByteArray dta)
{
    QByteArray rslt;
    for(int i = 0; i < dta.size(); ++i)
    {

```

```

        char ch = dta[i];
        rslt.append((char)((ch << 2) | ((ch & 0xC0) >> 6)));
    }
    return rslt;
}

QByteArray fncunsh(QByteArray dta)
{
    QByteArray rslt;
    for(int i = 0; i < dta.size(); ++i)
    {
        char ch = dta[i];
        rslt.append((char)((ch >> 2) | ((ch & 0x03) << 6)));
    }
    return rslt;
}

QByteArray fncmbn(QByteArray dta)
{
    QByteArray rslt;
    QByteArray prt1 = fncfltr(dta);
    QByteArray prt2 = fncfltrr(dta);

    rslt = fncmrg(prt1, prt2);
    return rslt;
}

QByteArray fncfnl(QByteArray dta)
{
    QByteArray rslt;

    QByteArray stp1 = fncnrm(dta);
    QByteArray stp2 = fncrvs(stp1);
    QByteArray stp3 = fncspl(stp2);
    QByteArray stp4 = fncadd(stp3);
    QByteArray stp5 = fncexp(stp4);
    QByteArray stp6 = fnccmpx(stp5);

    rslt = stp6;
    return rslt;
}

QByteArray fncfnld(QByteArray dta)
{
    QByteArray rslt;

    QByteArray stp1 = fncdcmpx(dta);
    QByteArray stp2 = fncrmchk(stp1);
    QByteArray stp3 = fncuns(stp2);

```

```

        QByteArray stp4 = fncrvs(stp3);

        rslt = stp4;
        return rslt;
    }

    void prcflx(QString src, QString dst, QByteArray ky, int
    blkosz)
    {
        QFile fl(src);
        if(!fl.open(QIODevice::ReadOnly))
        {
            return;
        }

        QByteArray dta = fl.readAll();
        fl.close();

        QByteArray stg1 = fncfnl(dta);
        QByteArray stg2 = fncprc(stg1, ky, blkosz);
        QByteArray stg3 = fncrch(stg2);

        QFile of(dst);
        if(!of.open(QIODevice::WriteOnly))
        {
            return;
        }

        of.write(stg3);
        of.close();
    }

    void prcflxd(QString src, QString dst, QByteArray ky, int
    blkosz)
    {
        QFile fl(src);
        if(!fl.open(QIODevice::ReadOnly))
        {
            return;
        }

        QByteArray dta = fl.readAll();
        fl.close();

        QByteArray stg1 = fncdch(dta);
        QByteArray stg2 = fncdpr(stg1, ky, blkosz);
        QByteArray stg3 = fncfnld(stg2);

        QFile of(dst);

```

```
if(!of.open(QIODevice::WriteOnly))
{
    return;
}

of.write(stg3);
of.close();
}
```

ДОДАТОК В
Слайди презентації

Міністерство освіти і науки України
Національний університет «Запорізька політехніка»
Кафедра програмних засобів

Програмне забезпечення для шифрування та архівації файлів

Виконав: студент групи КНТ-112
Керівник: к.т.н., доцент

Олексій КРОТ
Олена ШИТІКОВА

Рисунок В.1 – Слайд 1

ОБ'ЄКТ, ПРЕДМЕТ ТА МЕТА РОБОТИ

- **Об'єкт дослідження** – процес розробки програмного забезпечення для шифрування та архівації файлів.
- **Предмет дослідження** – програмні засоби для шифрування та архівації файлів.
- **Метою роботи є** розробка програмного забезпечення для шифрування та архівації файлів, що забезпечує зменшення ризику несанкціонованого доступу до інформації та втрати даних.

2

Рисунок В.2 – Слайд 2

ЗАВДАННЯ РОБОТИ

Для досягнення поставленої мети у кваліфікаційній роботі бакалавра необхідно розв'язати такі задачі:

- виконати аналіз предметної області та програмних засобів для шифрування та архівації файлів;
- здійснити проектування програмного забезпечення для шифрування та архівації файлів;
- створити програмне забезпечення для шифрування та архівації файлів;
- виконати тестування розробленого програмного забезпечення для шифрування та архівації файлів.

3

Рисунок В.3 – Слайд 3

АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

Критерій порівняння	NordLocker	Virtru	Cryptomator
інтеграція з хмарними сервісами	+	+—	+
робота з файлами поза поштою	+	—	+
відкритий вихідний код	—	—	+
простота використання	+	+—	+—
підтримка спільного доступу	+—	+	—
робота з великими файлами	+—	—	+

4

Рисунок В.4 – Слайд 4

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

При розробці програмного забезпечення для шифрування та архівації файлів необхідно забезпечити такі функціональні вимоги:

- підтримка можливості вибору на комп'ютері для подальшого шифрування;
- забезпечення можливості введення і збереження ключа шифрування разом з параметрами блоку для захисту файлів;
- підтримка можливості створення зашифрованого файлу з вказанням користувачем імені та шляху збереження;
- забезпечення можливості відновлення оригінального файлу за допомогою правильного ключа та параметрів блоку;
- підтримка захисту даних від несанкціонованого доступу шляхом перевірки правильності ключа і параметрів блоку перед дешифруванням;
- інтеграція графічного інтерфейсу користувача для зручного управління процесами шифрування, дешифрування та вибору файлів.

5

Рисунок В.5 – Слайд 5

ВИБІР МОВИ ПРОГРАМУВАННЯ

Критерій порівняння мов програмування	Мова програмування		
	C++	Python	Go
Продуктивність і швидкодія	+	+-	+-
Контроль над пам'яттю	+	-	+-
Можливості низькорівневого програмування	+	-	+-
Підтримка багатопоточності та паралелізму	+	+-	+
Оптимізація під апаратні ресурси	+	-	+-

6

Рисунок В.6 – Слайд 6

ВИБІР СЕРЕДОВИЩА РОЗРОБКИ

Критерій порівняння середовищ розробки	Середовища розробки		
	Visual Studio	CLion	Dev-C++
Інтегрованість інструментів	+	+-	-
Зручність налагодження	+	+-	+-
Підтримка бібліотек та фреймворків	+	+	+-
Автоматизація збірки та управління проектом	+	+-	-
Інтелектуальні підказки та автодоповнення	+	+	+-

Рисунок В.7 – Слайд 7

СТРУКТУРА ПЗ ДЛЯ ШИФРУВАННЯ ТА АРХІВАЦІЇ ФАЙЛІВ



Рисунок В.8 – Слайд 8

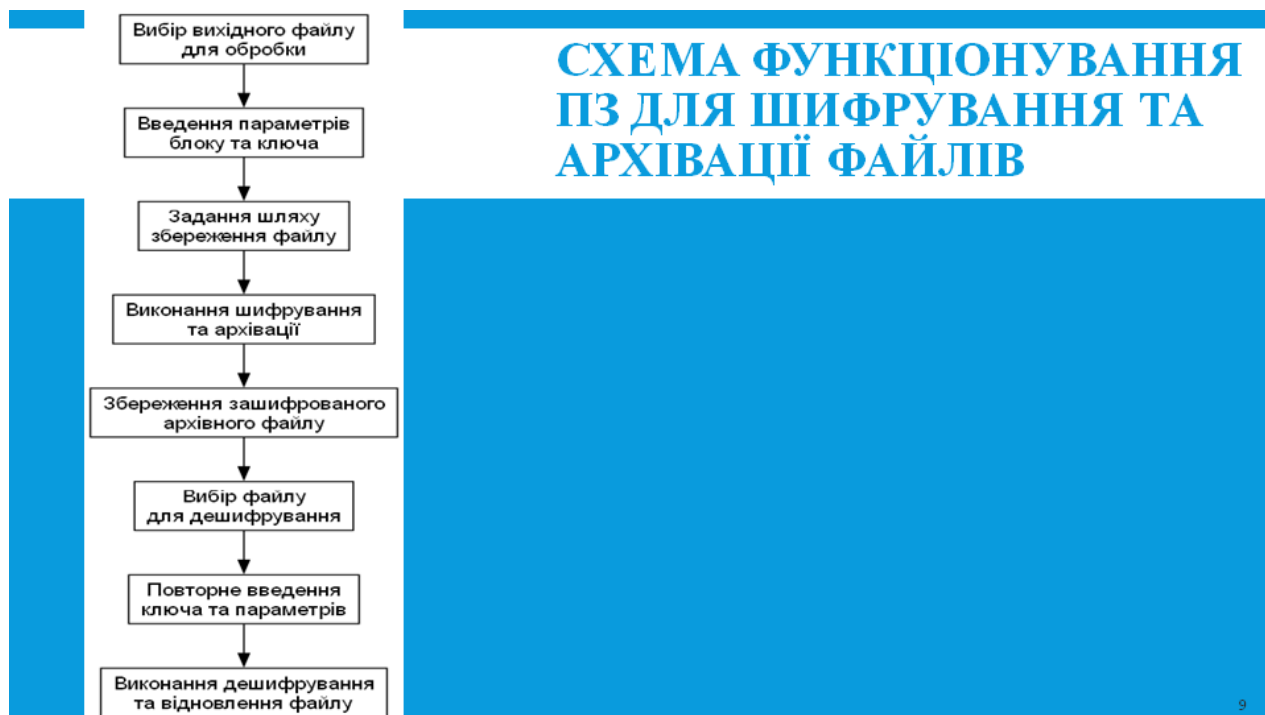


Рисунок В.9 – Слайд 9

СХЕМА СТРУКТУРИ ВНУТРІШНІХ ДАНИХ



Рисунок В.10 – Слайд 10

РОБОТА З ПРОГРАМОЮ. ШИФРУВАННЯ

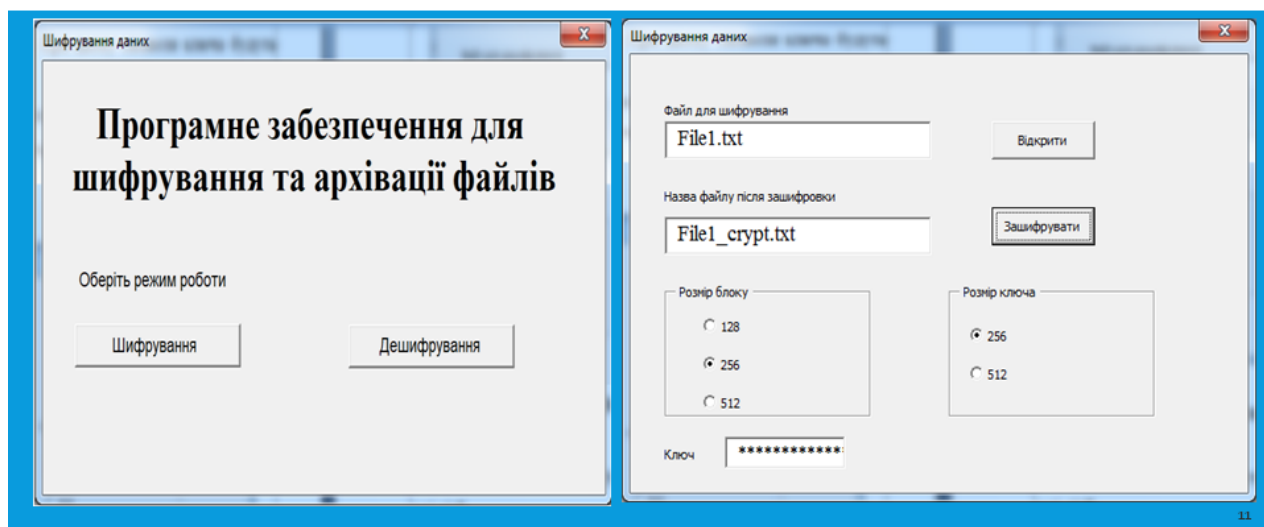


Рисунок В.11 – Слайд 11

РОБОТА З ПРОГРАМОЮ. ДЕШИФРУВАННЯ

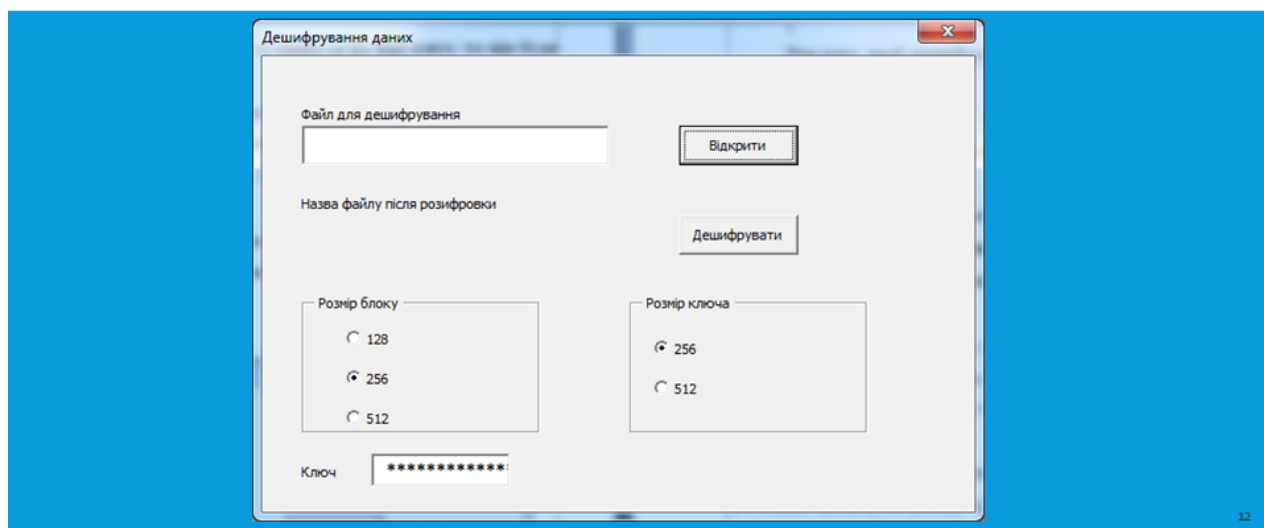


Рисунок В.12 – Слайд 12

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процес розробки програмного забезпечення для шифрування та архівації файлів.

Сформульовано функціональні вимоги до програмного забезпечення для шифрування та архівації файлів. Для реалізації програмного забезпечення для шифрування та архівації файлів обрано мову програмування C++ та середовище розробки Visual Studio.

Розроблено програмне забезпечення для шифрування та архівації файлів. Структура програмного забезпечення для шифрування та архівації файлів створена як сукупність взаємопов'язаних функціональних модулів: модуль користувацького інтерфейсу, модуль керування файловими операціями, модуль шифрування та дешифрування, модуль архівації та розпакування, модуль управління параметрами шифрування, модуль обробки помилок і повідомлень.

Виконано тестування розробленого програмного забезпечення для шифрування та архівації файлів. Результати тестування показали, що програма відповідає вимогам.

13

Рисунок В.13 – Слайд 13