Форма № 17

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет «Запорізька політехніка»

Кафедра програмних засобів

КУРСОВИЙ ПРОЕКТ

з дисципліни: «Об'єктно-орієнтоване програмування»
на тему: «Візуалізація криптографічних алгоритмів для Blockchain технології. RSA (Rivest–Shamir–Adleman)»

Студентів 1 курсу КНТ-219сп групи
спеціальності Комп'ютерні науки
освітня програма (спеціалізація)
Комп'ютерні науки
Ковальов Р.В.,
Резніченко А.С.,
Рогова Є. В.,
Тихоновська Т.В.
Керівник доцент Миронова Н.О.
Національна шкала _____
Кількість балів: _____Оцінка: ECTS _____

Члени комісії _____ Табунщик Г.В.
(підпис) (прізвище та ініціали)
_____ Миронова Н.О.
(підпис) (прізвище та ініціали)
_____ Каплієнко Т.І.
(підпис) (прізвище та ініціали)

2020 рік

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЗАПОРІЗЬКА ПОЛІТЕХНІКА»

Кафедра *програмних засобів*
Дисципліна *Об'єктно-орієнтоване програмування*
Спеціальність *Комп'ютерні Науки*
Курс *1* Група *КНТ-219сп* Семестр *I*

ЗАВДАННЯ
на курсовий проект (роботу) студентам

Ковальову Роману Віталійовичу, Резніченко Андрію Сергійович, Роговій Єлизаветі Віталіївні, Тихоновській Тетяні Володимирівні

1. Тема проекту (роботи): *Візуалізація криптографічних алгоритмів для Blockchain технології. RSA (Rivest–Shamir–Adleman)*

2. Термін здачі студентом закінченого проекту (роботи):
3. Вихідні дані до проекту: *Реалізувати криптографічного алгоритму RSA (Rivest–Shamir–Adleman)*
вхідні дані: Зображення формату bmp, png, jpeg, jpg, txt
вихідні дані: програма повинна забезпечувати відображення шифрованої інформації та її збереження в відповідному форматі до вхідного і відображення дешифрованої інформації – її збереження в відповідному форматі до вхідного
4. ЗМІСТ розрахунково-пояснювальної записки (перелік питань, що їх належить розробити*): 1 Аналіз предметної області*
2 Аналіз програмних засобів
3 Основні рішення з реалізації компонентів системи
4 Посібник програміста
5 Інструкція користувача
Висновки, Додаток А Текст програми,
Додаток Б Інтерфейс програми,
Додаток В Слайди презентації

5. Дата видачі завдання: *вересень 2019р*

Календарний план

№ пор.	Назва етапів курсового проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітка
1.	Аналіз індивідуального завдання	1-2 тиждень	
2.	Аналіз програмних засобів, що будуть використовуватись в роботі	3-4 тиждень	
3.	Аналіз структур даних, що необхідно використати в курсової роботі	4-5 тиждень	
4.	Вивчення можливостей програмної реалізації структур даних та інтерфейсу користувача	5-6 тиждень	
5.	Оформлення відповідних пунктів пояснювальної записки	4 тиждень	розділи 1,2 ПЗ
6.	Проміжний контроль	8 тиждень	
7.	Аналіз вимог до апаратних засобів	9 тиждень	
8.	Розробка програмного забезпечення	10-15 тиждень	
9.	Оформлення, відповідних пунктів пояснювальної записки	9-16 тиждень	розділи 3,4,5 ПЗ
10.	Захист курсової роботи	17 тиждень	

Студент_____ Ковальов Р.В.

Студент _____ Резніченко А.С.

Студентка _____ Рогова Є.В.

Студентка _____ Тихоновська Т.В.

Керівник _____Миронова Н.О.

“_____” _____ 2019р

ЗМІСТ

РЕФЕРАТ

ПЗ: 65 стор., 21 рис., 31 додатки, 10 джерел.

Метою даного курсового проекту є розробка візуалізації криптографічних алгоритмів для Blockchain технології. RSA (Rivest–Shamir–Adleman).

Було виконано аналіз предметної області, розглянуто аналогічні існуючі методи та програмні засоби для вирішення завдання.

Для реалізації програмного продукту використовувалася мова програмування C# та середа розробки JetBrains Rider.

При виконані курсового проекту було розглянуто особливості мови програмування C# та компілятору. Також приділяється увага розробці інтерфейсу користувача, який було розроблено за допомогою Windows Forms. Також приділяється акцент розробці графічного інтерфейсу користувача, файловому вводу-виводу.

RSA, BLOCKCHAIN, LSB, КРИПТОГРАФІЯ, СОРТУВАННЯ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Blockchain	Ланцюжок з формованих блоків транзакцій, який побудований за певними правилами.
RSA	Криптографічний алгоритм з відкритим ключем, що базується на обчислювальній складності задачі факторизації великих цілих чисел.
LSB	Суть цього методу полягає в заміні останніх значущих бітів в контейнері (зображення, аудіо або відеозапису) на біти приховуваного повідомлення.
Криптографія	Наука про математичні методи забезпечення конфіденційності, цілісності і автентичності інформації.
ООП	Об'єктно-орієнтоване програмування - одна з парадигм програмування, яка розглядає програму як множину «об'єктів», що взаємодіють між собою

ВСТУП

Технологія Blockchain часто асоціюється у спільноти з «Інтернетом цінностей». З її допомогою будь-яка людина здатна розмістити в мережі дані, після чого інші користувачі зможуть отримати до неї простий доступ, де б вони не знаходилися. Щоб отримати доступ до розподіленої бази даних, людина повинна володіти закритим ключем, який створений на основі криптографічних алгоритмів. Такий ключ відкриє доступ тільки до тих блоків, які «належать» вам. А якщо ви передаватимете іншій особі закритий ключ, то делегуєте таким чином доступ до даних цих блоків.

Завдяки цьому алгоритму встановлюються довірчі відносини і реалізується принцип автентичності особистості. Нікому не можна змінити ланцюжок блоків без володіння потрібними ключами.

Таким чином, актуальність роботи полягає в практичному підході до роботи із текстом і зображенням, де введена інформація зчитується і додається в масив, що шифрується за відповідним ключем і дешифрується за другим ключе відповідним першому повертаючи початкову інформацію.

Об’єктом дослідження є процес шифрування і дешифрування за створеними ключами.

Метою даного курсового проекту є розробка програмного забезпечення візуалізації для криптографічних алгоритмів Blockchain технології.

Досягнення поставленої мети вимагає розв’язання таких завдань:

- дослідити та розглянути аналоги реалізацій алгоритму RSA;
- обрати метод роботи з зображенням;
- організовувати відображення результатів роботи алгоритму і можливість їх збереження.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд існуючих методів вирішення завдання

Прикладна програма (застосунок) - користувацька комп'ютерна програма, що дає змогу вирішувати конкретні прикладні задачі користувача.

Види прикладних програм:

- застосунки підприємств та організацій. Наприклад, фінансове управління, система взаємовідносин із споживачами, система постачання. До цього типу відносяться також відомче ПЗ підприємств малого бізнесу, а також окремі підрозділи великого підприємства (Керування транспортними витратами, служба IT підтримки);
- застосунки інфраструктури підприємства. Забезпечує загальні потреби у підтримці ПЗ підприємств. Це бази даних, сервери електронної пошти, керування мережею та безпекою;
- застосунки інформаційного робітника. Обслуговує потреби індивідуальних користувачів у створенні та керуванні інформацією. Це, як правило, керування часом, ресурсами, документацією, наприклад, текстові редактори, електронні таблиці, програми-клієнти для електронної пошти та блогів, персональні інформаційні системи та медіа редактори;
- застосунки для доступу до змісту. Використовується для доступу до тих чи інших програм або ресурсів без їх редагування (може включати й функцію редагування). Призначено для груп та індивідуальних користувачів цифрового контенту. Це, наприклад, медіаплеєри, веб-браузери, допоміжні браузери та інше;
- освітні застосунки за змістом близько до ПЗ для медіа та розваг, але на відміну від нього має чіткі вимоги по тестуванню знань користувача та відслідковуванню прогресу у вивченні того чи іншого матеріалу. Багато освітніх програм включають функції спільного користування та багатостороннього співробітництва. Способи використання освітніх програмних засобів у навчальному процесі вивчає технопедагогіка;
- імітаційні програми. Використовуються для симуляції фізичних або абстрактних систем у наукових, освітніх або інших цілях;
- інструментальні застосунки у галузі медіа. Забезпечують потреби користувачів, що виробляють друковані або електронні медіа ресурси для інших

споживачів, на комерційних або освітніх засадах. Це програми поліграфічної обробки, верстання, обробки мультимедіа, редактори HTML, редактори цифрової анімації, цифрового звуку та інше;

- застосунки для проектування та конструювання. Використовуються при розробці апаратного («залізо») і програмного забезпечення. [6]

Для реалізації проекту обрано такий вид застосунку як імітаційна програма. Це обумовлене тим, що програма призначена для демонстрації роботи алгоритму RSA.

1.2 Огляд існуючих програмних засобів

На сьогоднішній день існує багато аналогів, які часткового або повністю вирішують цю проблему. Одним з таких аналогів є програмний застосунок «Кодировка и декодировка RSA» [3]. Інтерфейс програми наведений на рисунку 1.1.

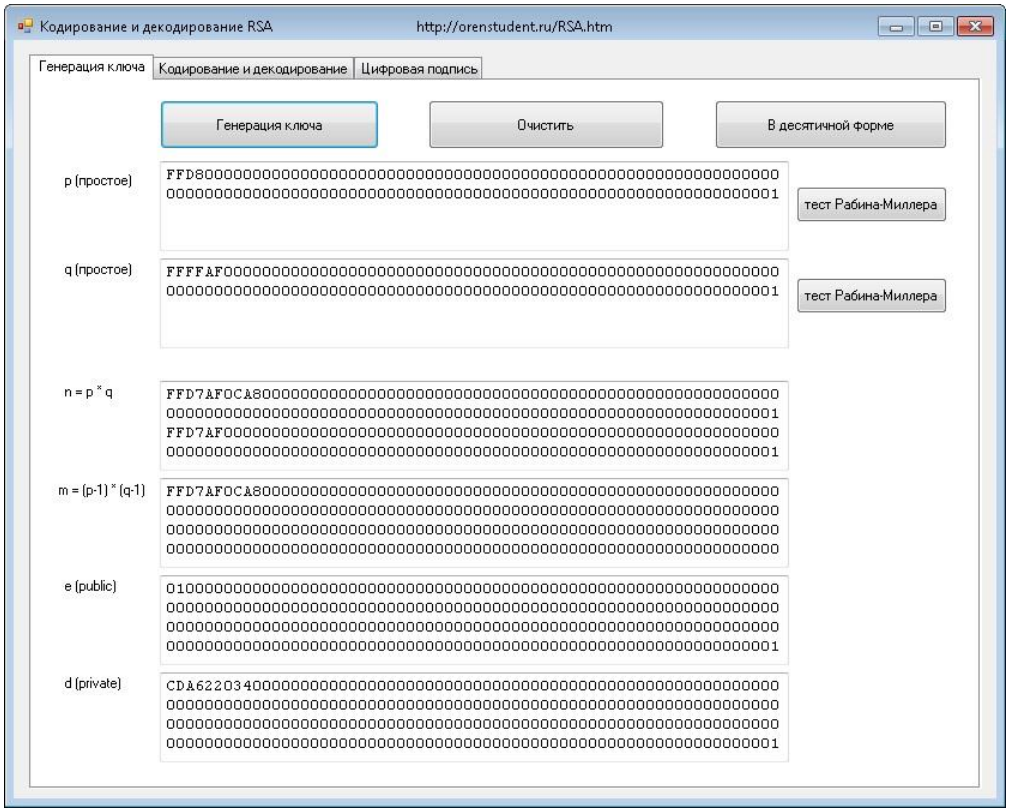

Рисунок 1.1 – Програмний застосунок «Кодировка и декодировка RSA»

Цей програмний застосунок містить зручний, інтуїтивно зрозумілий інтерфейс користувача і окрім можливості шифрувати і дешифрувати інформацію, а й створення цифрового підпису. Користувач отримує на виході громіздкі ключі і їх можливо перенести в десятковий формат, що є ще одною перевагою застосунку. Але користувачу необхідно вводити адресу файлу повністю в відповідне поле, що не зручно при швидкій роботі. Також не має можливості прослідкувати в самій формі за результатом шифрування і дешифрування.

Іншим аналогом є «ImageCryptography»[4]. Інтерфейс програми наведений на рисунку 1.2.

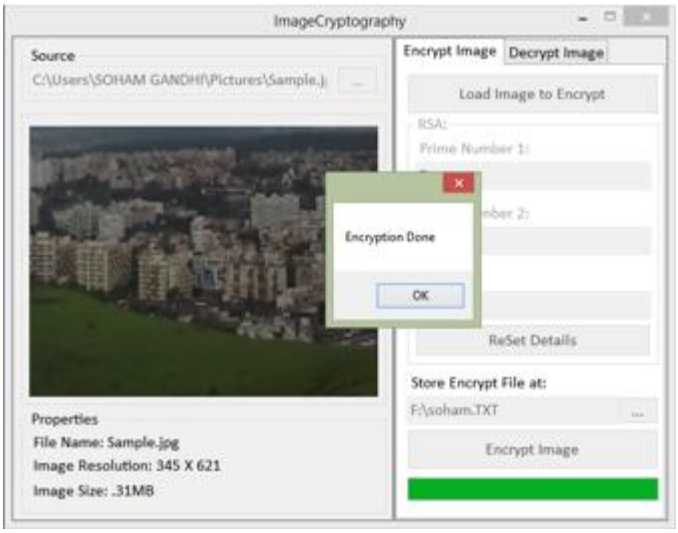

Рисунок 1.2 – Програмний застосунок «ImageCryptography»

В даному прикладі перевагами додатку є можливість побачити початкове зображення та інформацію про нього, також поле прогресу, що дає зрозуміти скільки ще часу триватиме процес шифрування чи дешифрування, і звичайно можливість обрати потрібне зображення за допомогою діалогового вікна. Про те до ряду недоліків додатку належить використання зображень великого розміру, що уповільнює роботу, занесення шифрованого зображення до файлу у вигляді масиву, що не дає змоги візуально перевірити результат роботи алгоритму.

1.2.1 Алгоритм шифрування RSA (Rivest–Shamir–Adleman)

Алгоритм RSA складається з 4 етапів: генерації ключів, шифрування, розшифрування та розповсюдження ключів.

Безпека алгоритму RSA побудована на принципі складності факторизації цілих чисел. Алгоритм використовує два ключі — відкритий (public) і секретний (private), разом відкритий і відповідний йому секретний ключі утворюють пари ключів (keypair). Відкритий ключ не потрібно зберігати в таємниці, він використовується для шифрування даних. Якщо повідомлення було зашифровано відкритим ключем, то розшифрувати його можна тільки відповідним секретним ключем.

Розповсюдження ключів. Для того, щоб Боб міг відправити свої секретні повідомлення, Аліса передає свій відкритий ключ (n, e) Бобу через надійний, але не обов'язково секретний маршрут. Секретний ключ d ніколи не розповсюджується.

Генерація ключів. Для того, щоб згенерувати пари ключів виконуються такі дії:

1. Вибираються два великі прості p і q приблизно 512 біт завдовжки кожне
2. Обчислюється їх добуток n=pq
3. Обчислюється функція Ейлера $\varphi(n)=(p-1)(q-1)$
4. Вибирається ціле число $1<e<\varphi(n)$ та e взаємно просте з $\varphi(n)$
5. За допомогою розширеного алгоритму Евкліда знаходиться d, таке, що $ed=1 \pmod{\varphi(n)}$

Число n називається модулем, а e і d — відкритою й секретною експонентами (англ. encryption and decryption exponents), відповідно. Пари чисел (n,e) є відкритою частиною ключа, а (n,d)— секретною. Числа p і q після генерації пари ключів можуть бути знищені, але в жодному разі не повинні бути розкриті.

Шифрування. Припустимо, що Боб хотів би відправити повідомлення M Алісі. Спочатку він перетворює M в ціле число m так, щоб $0 \leq m < n$ за допомогою узгодженого оборотного протоколу, відомого як схеми доповнення. Потім він обчислює зашифрований текст c, використовуючи відкритий ключ Аліси e, за допомогою рівняння:

$$c=m^e \bmod n$$

Це може бути зроблено досить швидко, навіть для 500-бітних чисел, з використанням модульного зведення в ступінь. Потім Боб передає c Алісі.

Розшифрування. Для розшифрування повідомлення Боба m Алісі потрібно обчислити таку рівність:

$$m=c^d \bmod n$$

Неважко переконатися, що при розшифруванні відновиться вихідне повідомлення:

$$C^d=(m^e)^d=m^{ed}\ (\text{mod } n)$$

З умови $ed=1(\text{mod } \varphi(n))$ випливає, що $ed=k\ \varphi\ (n)+1$ для деякого цілого k, отже $m^{ed}=m^{k\varphi(n)+1}\text{mod } n$.

Згідно з теоремою Ейлера: $m^{\varphi(n)}=1(\text{mod } n)$ тому:

$$m^{k\varphi(n)+1}=m(\text{mod } n)$$

$$c^d=m(\text{mod } n)$$

1.3 Постановка завдання роботи

Створити програмний продукт, який шляхом взаємодії з користувачем шифрувати і дешифрувати зображення за створеними ключами.

Система повинна забезпечувати:

- завантаження зображення або тексту;
- зчитати зображення або текст в масив;
- шифрувати елементи масиву;
- вивести шифровану інформацію;
- завантажити шифроване зображення або текст;
- зчитати шифроване зображення або текст в масив;
- дешифрувати елементи масиву;
- вивести дешифровану інформацію.

2 АНАЛІЗ ПРОГРАМНИХ ЗАСОБІВ

2.1 Огляд особливостей мови програмування

C# - об'єктно-орієнтована мова програмування. Синтаксис C# близький до C++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки. Ось лише кілька функцій мови C #, що забезпечують надійність і стійкість додатків: прибирання сміття автоматично звільняє пам'ять, зайняту знищеними і невикористовуваними об'єктами; обробка виключень надає структурований і розширюваний спосіб виявляти і обробляти помилки; сувора типізація мови не дозволяє звертатися до неініціалізованих змінним, виходити за межі індексованих масивів або виконувати неконтрольоване приведення типів.

Основними перевагами мови є:

а) багато синтаксичного цукру. Синтаксичний цукор - це такі конструкції, які створені для полегшення написання і розуміння коду (особливо якщо це код іншого програміста) і не грають ролі при компіляції;

б) гарний розвиток завдяки підтримки Microsoft;

в) повністю ООП. Об'єктно-орієнтоване програмування - одна з парадигм програмування, яка розглядає програму як множину «об'єктів», що взаємодіють між собою;

г) величезна кількість вже готових класів на всі випадки життя;

ґ) є справжні (а не тільки вкладені) багатовимірні масиви і опціонально - перевірка переповнення.[5]

2.2 Огляд особливостей обраного середовища програмування

JetBrains Rider – кросплатформене інтегроване середовище розробки програмного забезпечення. Функціональність Rider полягає у:

а) аналіз коду. Rider надає більше 2200 інспекцій коду і автоматизованих виправлень для усунення виявлених проблем як в індивідуальному, так і масовому

порядку. Механізм аналізу помилок по всьому рішенням буде шукати помилки в кодової базі і повідомляти про них, навіть якщо проблемний файл не відкритий в редакторі;

б) редагування коду. Розумний редактор Rider надає різні види автодоповнення і шаблонів, автоматично вставляє парні дужки і імпортує відсутні простору імен. Підказки та іконки на полях допомагають легко переміщатися по ієрархії успадкування, контекстні дії роблять розробку зручною та ефективною;

в) налагодження та інші інструменти. Вбудований відладчик для додатків на .NET Framework, Mono і .NET Core підтримує покрокове виконання, дозволяє обчислювати вирази на льоту, запускати програму від поточної виконуваної рядки до рядка з курсором, відстежувати і змінювати значення змінних. Крім того, Rider включає в себе браузер NuGet, дозволяє переглядати трасування стека, підтримує різні системи контролю версій і бази даних;

г) навігація та пошук. Ви можете моментально переходити до будь-якого файлу, типу або члену у вашій кодової базі, а також швидко знаходити потрібні налаштування і дії IDE. Від будь-якого символу в коді ви можете миттєво перейти до базових і похідних символам, реалізацій та перевизначення, а також до місць використання.[1]

2.3 Основні компоненти Windows Forms

Для створення програмного застосунку було використано такі компоненти Windows Forms:

- Form – основний клас для роботи з формою.
- PictureBox – елемент, що дозволяє виводити зображення на форму. Він дозволяє відобразити файли у форматі bmp, jpeg, jpg, png, а також метафайли зображень та іконки.
- MessageBox – елемент, що дозволяє показати користувачу повідомлення.
- Label – елемент для відображення тексту на формі, доступного тільки для читання.
- TextBox – елемент для вводу та редагування тексту.
- OpenFileDialog и SaveFileDialog – класи для відкриття та збереження файлів

2.4 Використані класи мови C#

Клас MemoryStream безпосередньо розширено з класу Stream, це потік (stream), дані якого зберігаються (store) в пам'яті. Stream - клас, який імітує потік byte (байт), збудовані в ряд. Клас Stream надає базові методи роботи з потоками даних, а саме метод читання та запису байт або масиву байт.[2]

Клас використано для розкладу зображення на масив байт і роботи з ним для безпосередньо шифрування зображення. Також він використовується для зворотного процесу – збору зашифрованого масиву в зображення. Відповідна частина коду наведена на рисунку 2.1 та 2.2.

```
var ms = new MemoryStream();
if (image.RawFormat.Equals(System.Drawing.Imaging.ImageFormat.Jpeg))
{
    image.Save(ms, System.Drawing.Imaging.ImageFormat.Jpeg);
}
if (image.RawFormat.Equals(System.Drawing.Imaging.ImageFormat.Png))
{
    image.Save(ms, System.Drawing.Imaging.ImageFormat.Png);
}
if (image.RawFormat.Equals(System.Drawing.Imaging.ImageFormat.Bmp))
{
    image.Save(ms, System.Drawing.Imaging.ImageFormat.Bmp);
}
bytes = ms.ToArray();
```

Рисунок 2.1 – Розклад обраного зображення на масив байт

```
byte[] b = mas.Select(i => (byte)i).ToArray();
var imageMemoryStream = new MemoryStream(b);
Image imgFromStream = Image.FromStream(imageMemoryStream);
```

Рисунок 2.2 – Створення зображення з масиву байт

BigInteger структура надає тип BigInteger - це незмінний тип, який представляє довільно велике ціле число. Цей тип дозволяє зберігати цілі числа довільної (будь-який)

довжини і вирішувати математичні операції з ними. При зведенні числа в ступінь в даному випадку виходять дуже великі числа, які не поміщаються ні в один зі стандартних типів. Тому для їх зберігання використовується екземпляр класу BigInteger.[2]

Клас List <T> є універсальним еквівалентом класу ArrayList. Він реалізує IList <T> універсальний інтерфейс за допомогою масиву, розмір якого динамічно збільшується в міру необхідності. Елементи можна додавати в List <T> за допомогою методів Add або AddRange. Являє строго типізований список об'єктів, доступних за індексом. Підтримує методи для пошуку за списком, виконання сортування і інших операцій зі списками.

2.5 Висновки з розділу

У даному розділі було описано особливості мови програмування, середи розробки та обраного компілятору. Також розглянули аналоги існуючих програм для алгоритму RSA. Було розглянуто основні компоненти Windows Forms, які були використані для розробки програмного застосунку.

3 ОСНОВНІ РІШЕННЯ З РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

3.1 Основні рішення щодо уявлення даних системи

В ході роботи над програмою курсового проекту було розроблено клас Form1.

Клас Form1 є основним та містить в собі дані головної форми програми, забезпечує взаємодію з користувачем. Дані та методи класу наведені в табл. 3.1.

Таблиця 3.1 – Дані та методи класу RipperForm.

Поля та методи класу	Опис
1	2
public:	
public Form1()	Завантаження вмісту призначеного для користувача інтерфейсу
private:	
private bool IsTheNumberSimple(long n)	Функція перевірка чи є число простим
private List<string> RSA_Endoce(string s, long e, long n)	Функція шифрування алгоритмом RSA
private string RSA_Dedoce(List<string> input, long d, long n)	Функція дешифрування алгоритмом RSA
private long Calculate_e()	Функція визначення параметру e
private long Calculate_d(long d, long m)	Функція визначення параметру d
private void домопогаToolStripMenuItem_Click(object sender, EventArgs e)	Відображення інструкції для користувача
private void проАвторівToolStripMenuItem1_Click(object sender, EventArgs e)	Відображення інформації стосовно авторів проекту

Продовження табл. 3.1

1	2
private void Інструкція_Click(object sender, EventArgs e)	Відображення інформації стосовно призначення проекту
private void Ключі_Click(object sender, EventArgs e)	Кнопка меню «Ключі»
private void Шифрувати_Click(object sender, EventArgs e)	Кнопка меню «Шифрувати»
private void Дешифрувати_Click(object sender, EventArgs e)	Кнопка меню «Дешифрувати»
private void ЗавантажитиЗображення_Click(object sender, EventArgs e)	Кнопка для завантаження зображення і створення масиву байт з нього
private void ШифрФайл_Click(object sender, EventArgs e)	Кнопка для завантаження файлу, що містить шифрований текст
private void ШифрЗоб_Click(object sender, EventArgs e)	Кнопка для завантаження зашифрованого масиву та додаткової інформації до нього
private void ГенеруванняКлючів_Click(object sender, EventArgs e)	Кнопка генерування ключів
private void ШифруватиІнф_Click(object sender, EventArgs e)	Кнопка шифрування інформації
private void ЗавантажитиФайл_Click(object sender, EventArgs e)	Кнопка завантаження файлу
private void Зберигти_Click(object sender, EventArgs e)	Кнопка збереження шифрованої інформації
private void Дешифрування_Click(object sender, EventArgs e)	Кнопка дешифрування інформації
private void ЗберигтиДешифр_Click(object sender, EventArgs e)	Кнопка збереження дешифрованої інформації

3.2 Основні розроблені алгоритми

Під час розробки було реалізовано наступні алгоритми:

- файлове зчитування та запис у файл;
- відображення візуальних та текстових елементів;
- завантаження зображення або тексту;
- розбиття зображення на байтовий масив та переведення цього масиву в інший формат;
- застосування алгоритму криптографічного шифрування RSA (Rivest–Shamir–Adleman).

Для розбиття файлу на байтовий масив використовується клас C# - MemoryStream, що перетворює зображення потік байт і тим самим допомагає записати його то байтового масиву. Далі байтовий масив для зручності конвертується в масив цілих чисел. З нього обирається двадцять довільних значень. Це зроблено для прискорення роботи алгоритму, та можливості зберегти зашифроване зображення. Після чого, випадково обрані значення передаються у викликану функцію шифрування у вигляді рядку.

Алгоритм RSA розбито на 5 функцій, що виконують певний етап його роботи. Для створення ключів вводяться два простих числа, на основі яких генеруватимуться ключі. Одна з функцій перевіряє введені числа на простоту, тобто число має ділитися на себе та одиницю. Наступним необхідно вирахувати функцію Ейлера, що знаходиться шляхом перемноження введених значень менших на одиницю. Далі обирається відкрита експонента, що також є простим числом. Для цього створено функцію в якій на основі запропонованого масиву випадково обирається таке число. Інша функція вираховує секретну експоненту за формулою: відкрита експоненти в ступені мінус одиниці перемножується з модулем результату функції Ейлера. Ключі створено. Відкритий ключ матиме значення обраної відкритої експоненти та добутку введених простих чисел. Закритий ключ матиме значення секретної експоненти та добутку введених простих чисел. Шифрування відбувається за наступною формулою: кожне значення введеної інформації зводиться в ступінь введеної експоненти та перемножується з модулем добутку простих чисел. Дешифрування відбувається за тією самою формулою, проте необхідно вказувати значення другого ключа.

Додавання нових зображень у програму або тексту. При роботі алгоритму шифрування враховано можливість не заповнених полів, що є обов'язковими, тому тоді коли вони є вільними з'являється відповідне діалогове вікно, що повідомляє які поля мають бути заповненими для подальшої роботи.

3.4 Особливості реалізації системи

Особливістю реалізації системи є зрозумілий для користувача інтерфейс. Будь-яка людина зможе просто користуватися розробленим додатком на інтуїтивному рівні, завантажувати потрібне йому зображення або текст, перетворювати зображення в масив байт та застосовувати алгоритм шифрування і дешифрування. Також особливістю є шифрування не всього масиву байт створеного з зображення, а обраних навмання байт, що є додатковою інформацією для дешифрування. Відповідну частину коду з довільним обранням позицій масиву байт для шифрування наведено на рис. 3.1.

```
var ran = new Random();
int[] mos = new int[20];
int[] g = new int[20];

for (int i = 0; i < 20; i++)
{
    g[i] = ran.Next(100, mas.Length);
    mos[i] = mas[g[i]];
}

strImg = string.Join(" ", mos);

result = RSA_Endoce(strImg, e_, n);
```

Рисунок 3.1 – Обрання довільного масиву байт для шифрування

3.5 Результати тестування системи

У ході тестування програми усі помилки було виправлено. На даному етапі програма функціонує правильно та злагоджено. На рис. 3.2 зображено повідомлення, яке виникає при не заповнених полях для генерації ключів.

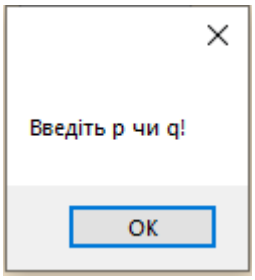

Рисунок 3.2 – Повідомлення не заповнених полів простих чисел

На рис. 3.3 зображено повідомлення, яке виникає при не заповнених полях ключів для шифрування чи дешифрування.

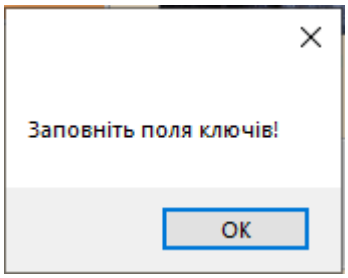

Рисунок 3.3 – Повідомлення не заповнених полів ключів

3.6 Висновки з розділу

В ході роботи над програмою курсового проекту було розроблено клас Form1, та використано клас MemoryStream та структуру BigInteger.

При виконанні курсової роботи було розроблено декілька алгоритмів (завантаження зображення, завантаження файлу, відображення візуальних та текстових елементів) які відповідають певним функціональним частинам програми, що допомагає створенню зрозумілого користувачеві інтерфейсу. При розробці інтерфейсу програми були використані такі візуальні компоненти Windows Forms: TextBox, Label, PictureBox, panel, ToolStripMenuItem.

У ході тестування системи усі помилки було виправлено.

4 ПОСІБНИК ПРОГРАМІСТА

4.1 Призначення та умови застосування програми

Призначення програми – забезпечення візуалізації шифрування зображень та тексту, які завантажуються користувачем.

Умовою застосування для коректної роботи є використання останніх версій ПЗ, що необхідно для роботи програми. Для отримання повного функціоналу цілком достатньо обирати зображення розміром не більше 400 КБ, текст може бути довільного розміру, але дія алгоритму буде повільнішою.

4.2 Характеристики програми

Програма виконана за допомогою мови програмування високого рівня C# в середовищі розробки JetBrains Rider. Проект (рис. 4.1) файли класу форми, ресурсів та проекту.

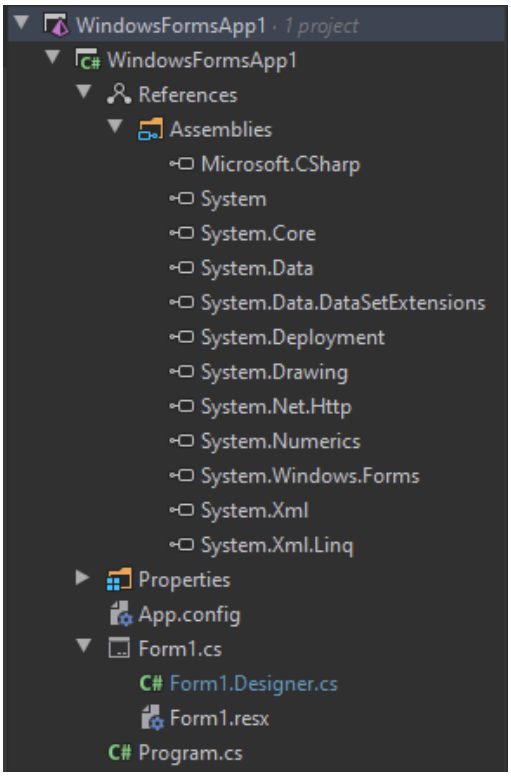

Рисунок 4.1 – Структура проекту

4.3 Звертання до програми

Для звернення необхідно розархівувати папку з програмою та розташувати її у каталог, до якого має доступ оператор. Далі для запуску програми у папці WindowsFormsApp1 (рис.4.2) треба запустити файл WindowsFormsApp1.csproj, дочекатися запуску проекту. Потім натиснути кнопку «Local Windows Debugger» і програма готова до роботи.

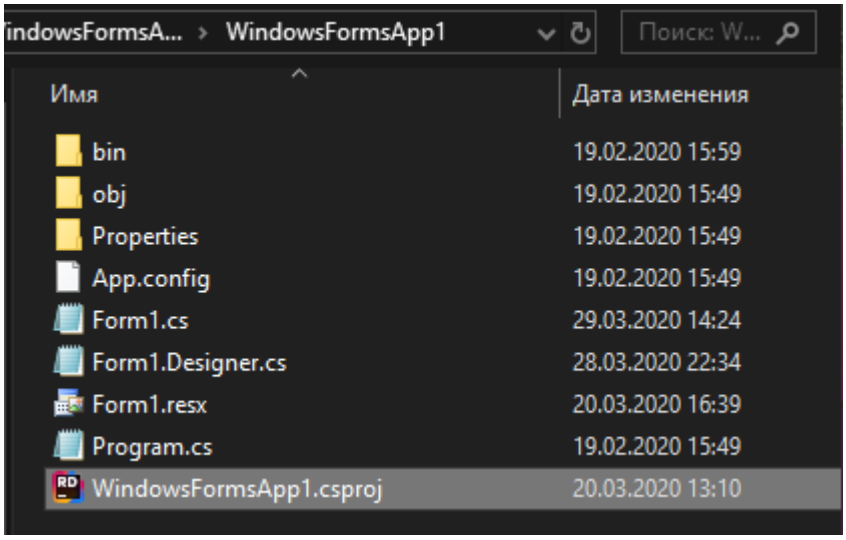

Рисунок 4.2 – Папка з файлами програми

4.4 Вхідні та вихідні дані

Вхідними даними виступає зображення або файл з текстом, яке користувач бажає шифрувати, чи вже шифроване зображення або файл з текстом, за наступними форматами: bmp, jpeg, jpg, png, txt. Вихідними даними виступають вже шифроване чи дешифроване зображення або файл з текстом, за наступними форматами: bmp, jpeg, jpg, png, txt. Також в програмі реалізовано обробка ситуацій коли вхідна чи вихідна інформація є пошкодженою.

4.5 Повідомлення

В процесі роботи програми можуть виникнути помилки в роботі програми. В такому разі оператор отримає інформаційні повідомлення щодо деталей помилки. При роботі може бути отримане таке повідомлення – помилка відкриття файлу, така помилка

може виникнути у випадку, якщо обраний користувачем файл пошкоджено. Або може з’явитися повідомлення пов’язане з незаповненими полями, що є до цього обов’язковими (рис. 4.3 та 4.4).

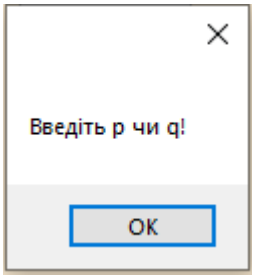

Рисунок 4.3 – Повідомлення не заповнених полів простих чисел

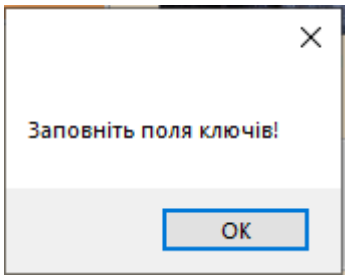

Рисунок 4.4 – Повідомлення не заповнених полів ключів

5 ІНСТРУКЦІЯ КОРИСТУВАЧА

5.1 Призначення програми

Програма призначена для візуалізації криптографічного алгоритму шифрування RSA (Rivest–Shamir–Adleman). Програма працює із зображеннями та текстовими файлами, перетворює зображення в масив байт, шифрує масив байт або текст, після виводить шифровану інформацію. Також дешифрує зображення або текст, виводить результат. Надаємо можливість збереження інформації.

5.2 Умови виконання програми

Програма має коректно працювати за умов:

- запуску останньої версії програми;
- збереження наборів інформації у директорії, в яких є права на читання та запис файлів, що не є заблокованими;
- правильного вводу даних в обов’язкові поля;
- вхідне зображення або текстовий файл не пошкоджені.

5.3 Як запустити програму

Для роботи у програмі перш за все необхідно розархівувати папку з програмою та розташувати її у каталог, до якого ви маєте доступ. Для запуску застосунку у папці WindowsFormsApp1 (рис.5.1) треба запустити файл WindowsFormsApp1.csproj, дочекатися запуску проекту та натиснути кнопку «Local Windows Debugger». Програма готова до роботи.

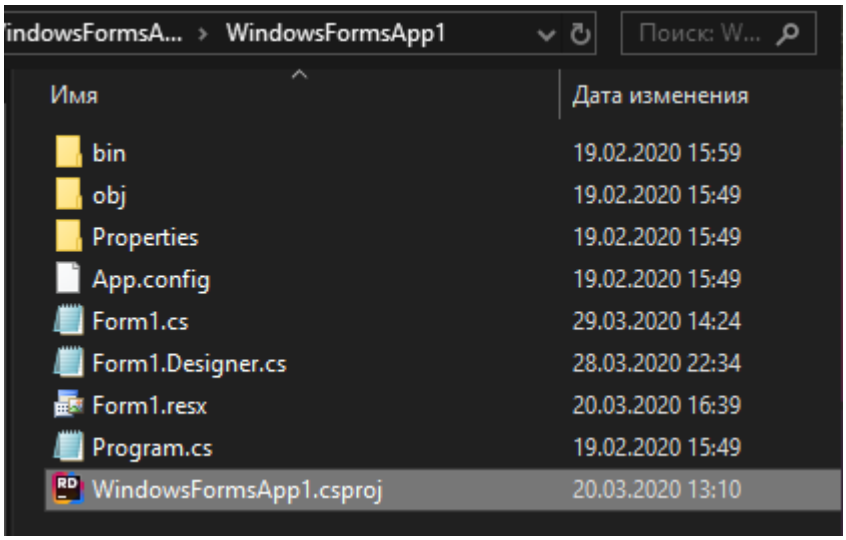

Рисунок 5.1 – Каталог з файлами програми

5.4 Виконання програми

Після налагодження програми перед користувачем з'являється головне вікно. Форма головного вікна є стартовою. Містить у собі кнопки переходу на інші панелі. Головна форма зображена на рис. 5.2.

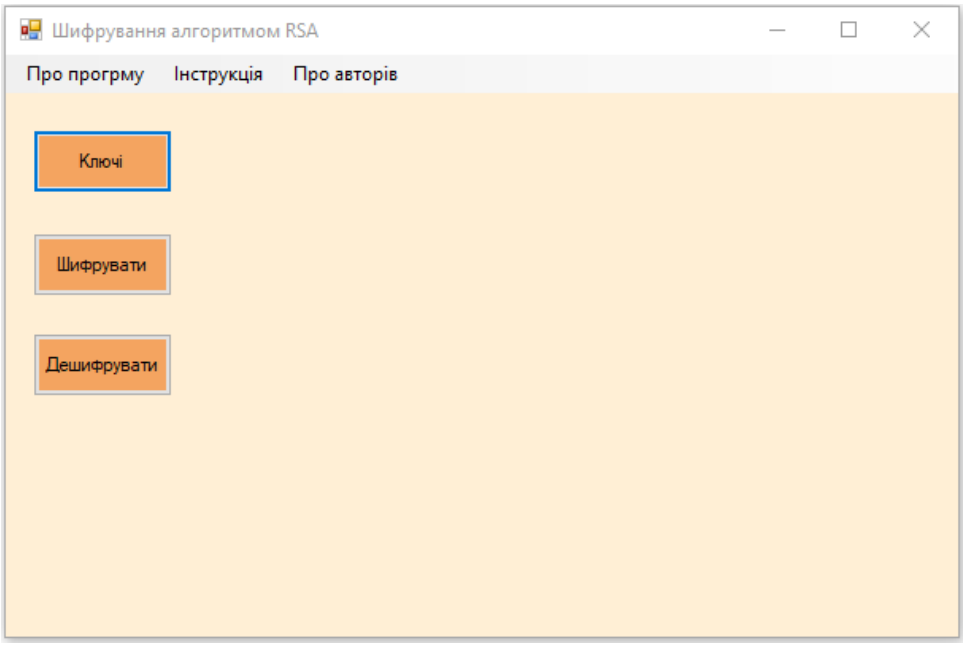

Рисунок 5.2 – Головна форма програми

Після того, як програму було відкрито слід натиснути кнопку «Ключі» і заповнити відповідні поля та натиснути кнопку «Створити пару ключів» після чого програма згенерує ключі відповідно до заповнених полів і збереже їх до файлу, що знаходиться в директорії проекту (рис. 5.3).

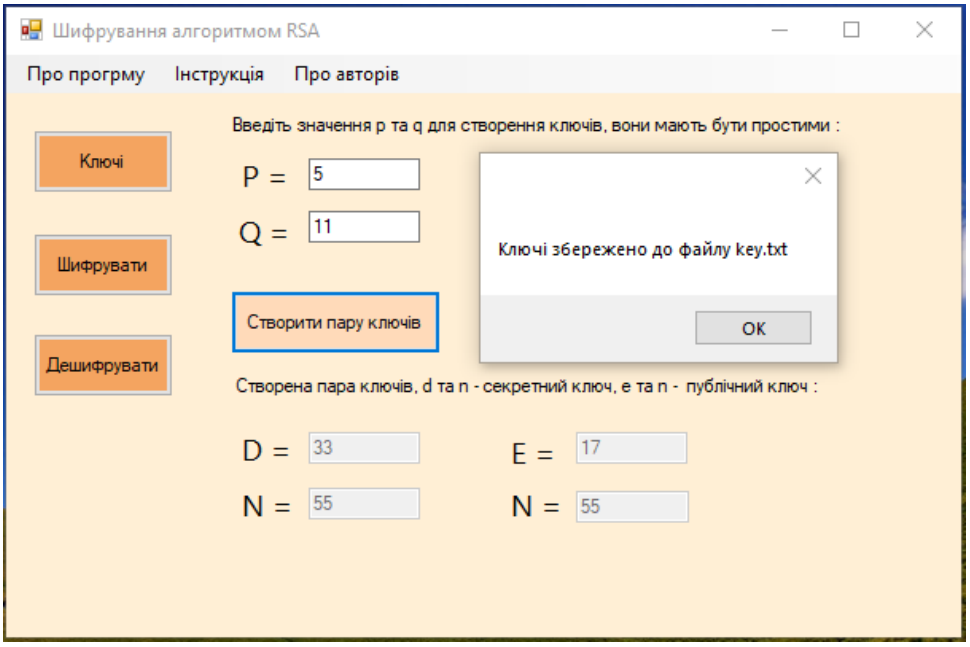

Рисунок 5.3 – Вікно панелі створення ключів

Після того, як буде згенеровано ключі слід натиснути кнопку «Шифрувати» і там обрати файл чи зображення для шифрування, для цього потрібно натиснути на кнопку «Завантажити зображення» чи «Завантажити документ». З’явиться відповідне діалогове вікно (рис. 5.4). Ці кнопки відкривають стандартне вікно відкриття файлів операційної системи Windows, у якому треба обрати зображення або файл відповідного формату та натиснути кнопку «Відкрити». Доступними форматами для завантаження є: bmp, jpeg, jpg, png, txt.

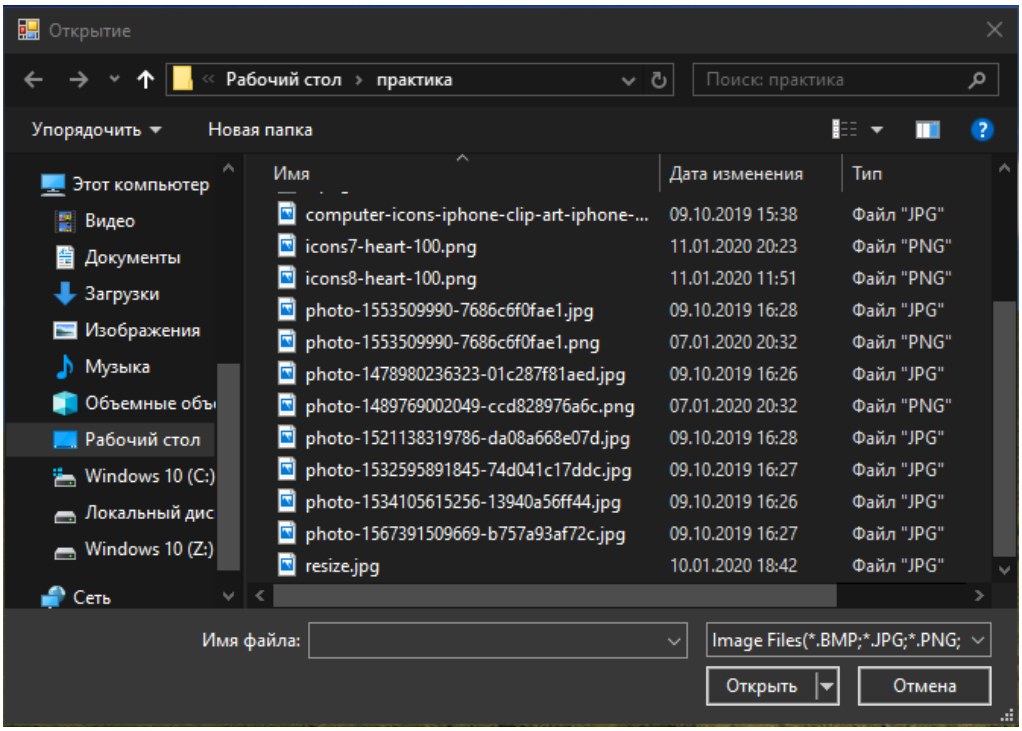

Рисунок 5.4 – Вікно для завантаження інформації

Далі слід заповнити поля ключів і натиснути кнопку «Шифрувати» в результаті чого введену інформацію буде змінено в відповідному її формату полі (рис. 5.5).

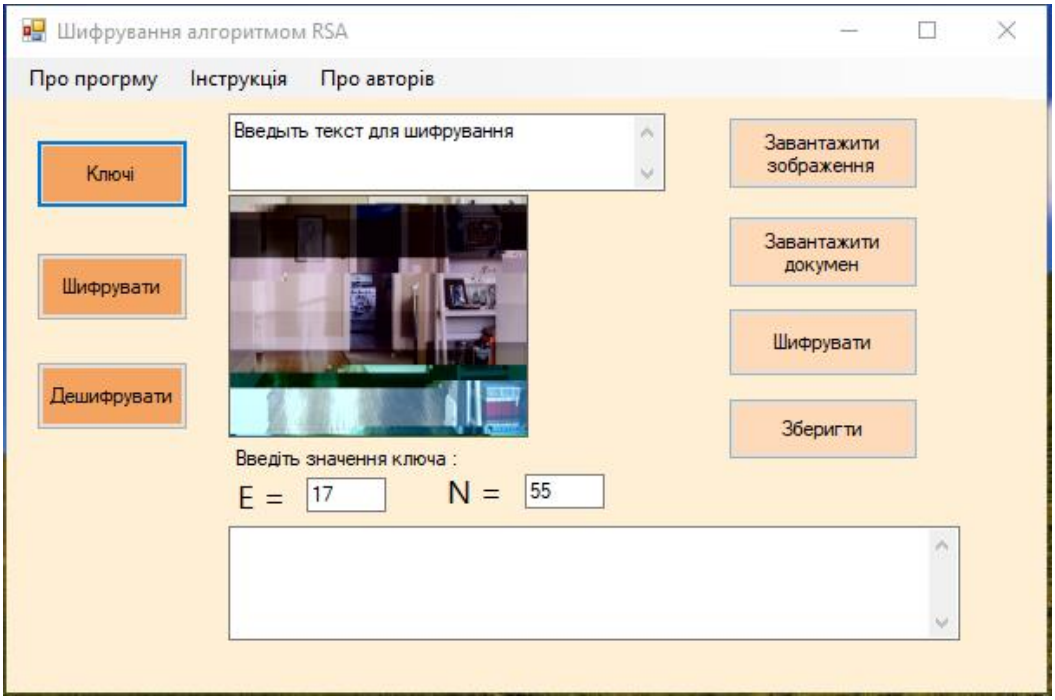

Рисунок 5.5 – Панель шифрування

Коли інформацію було зашифровано її можливо зберегти потрібно натиснути кнопку «Зберегти». З’явиться відповідне діалогове вікно (рис. 5.6). Ця кнопка відкриває стандартне вікно збереження файлу операційної системи Windows, у якому треба обрати місце розташування і ввести нову назву файлу, далі натиснути кнопку «Зберегти». Доступними форматами для збереження є: jpg, txt.

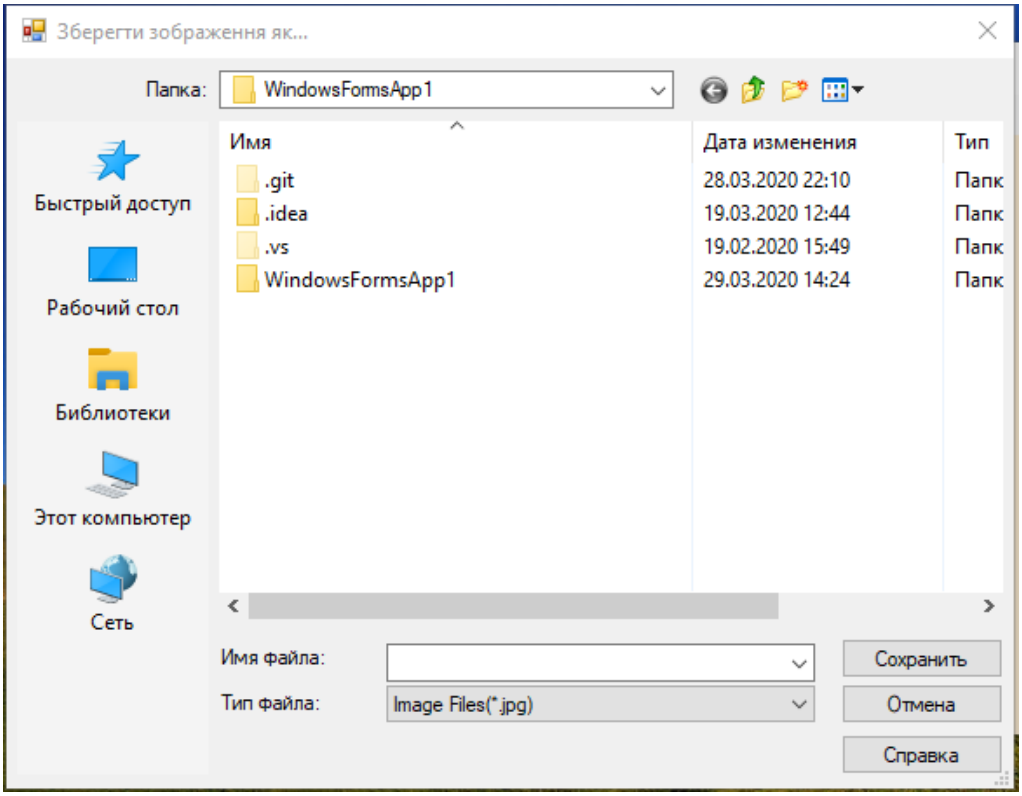

Рисунок 5.6 – Вікно для збереження інформації

Тоді можна дешифрувати нову збережену інформацію натиснувши кнопку «Дешифрувати» і там обрати файл чи зображення для дешифрування, для цього потрібно натиснути на кнопку «Завантажити зображення» чи «Завантажити документ». З’явиться відповідне діалогове вікно (рис. 5.4). Ці кнопки відкривають стандартне вікно відкриття файлів операційної системи Windows, у якому треба обрати зображення або файл відповідного формату та натиснути кнопку «Відкрити». Доступними форматами для завантаження є: bmp, jpeg, jpg, png, txt. Далі слід заповнити поля ключів і натиснути кнопку «Дешифрувати» в результаті чого введену інформацію буде змінено в відповідному її формату полі. Коли інформацію було зашифровано її можливо зберегти

потрібно натиснути кнопку «Зберегти». З’явиться відповідне діалогове вікно. Ця кнопка відкриває стандартне вікно збереження файлу операційної системи Windows, у якому треба обрати місце розташування і ввести нову назву файлу, далі натиснути кнопку «Зберегти». Доступними форматами для збереження є: jpg, txt.

При натисканні на кнопку «Про програму» відкривається форма, що містить інформацію стосовно призначення програми (рис. 5.7).

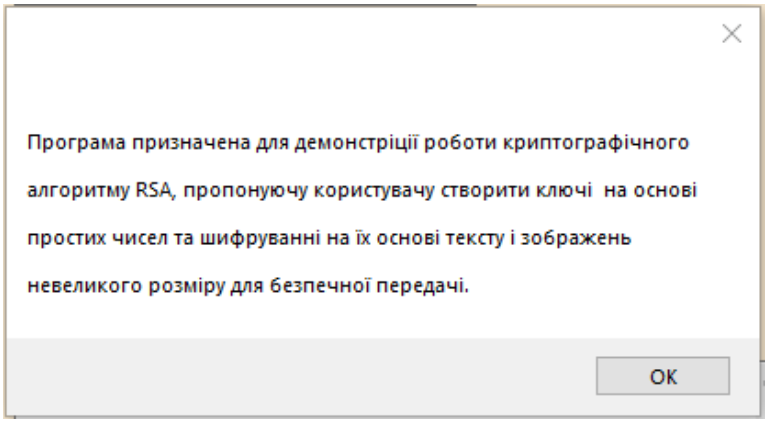

Рисунок 5.7 – Форма «Про програму»

При натисканні на кнопку «Автори» відкривається форма, що містить список розробників програми (рис. 5.8).

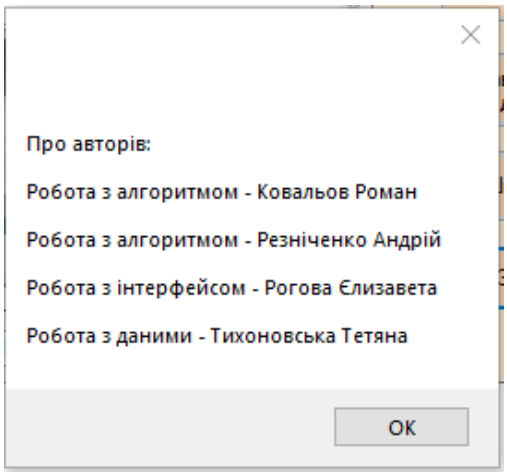

Рисунок 5.8 – Форма «Автори»

При натисканні на кнопку «Інструкція» відкривається форма з поясненням, як користуватися програмою (рис. 5.9).

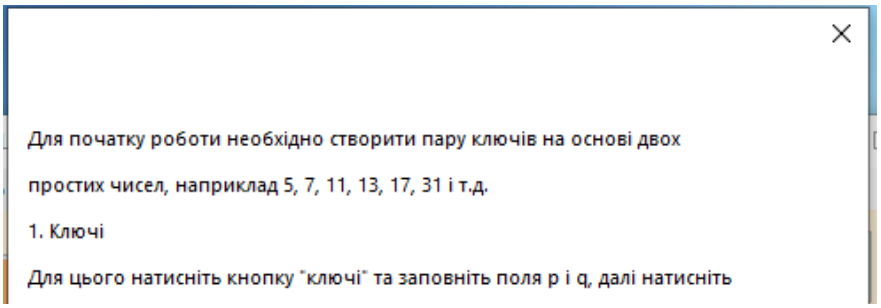

Рисунок 5.9 – Форма «Інструкція»

5.5 Повідомлення користувачу

В процесі роботи програми можуть виникнути помилки в роботі програми. В такому разі користувач отримає інформаційні повідомлення щодо деталей помилки. Одне з можливих повідомлень, що може бути отримане при роботі – така помилка може виникнути у випадку, якщо обраний користувачем файл пошкоджено. Або може з’явитися повідомлення пов’язане з незаповненими полями, що є до цього обов’язковими. В такому випадку користувачу лише необхідно заповнити відповідні поля. Ілюстрація повідомлення приведена на рис. 5.10.

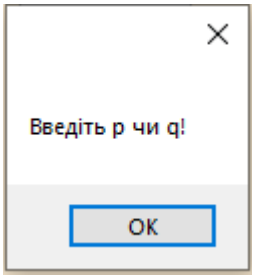

Рисунок 5.10 – Повідомлення не заповнених полів простих чисел

ВИСНОВКИ

Під час виконання курсового проекту було розроблено імітаційну програму на тему «Візуалізація криптографічних алгоритмів для Blockchain технології. RSA (Rivest–Shamir–Adleman)» для роботи з даними для забезпечення відображення алгоритму шифрування і дешифрування.

Створена імітаційна програма організовує систему для забезпечення вибору даних: зображення, текстовий файл. Для шифрування та дешифрування було використано криптографічний алгоритм RSA (Rivest–Shamir–Adleman). Таким чином, розробивши імітаційну програму ми можемо зробити висновок, що алгоритм є ефективним для шифрування даних, проте його час роботи залежить від розміру цих даних. Відповідно, якщо файл є великим тоді і час виконання є довшим.

Також було виконано огляд та аналіз існуючих методів вирішення завдання та існуючих програмних засобів, таких як «Кодировка и декодировка RSA» та «ImageCryptography».

Вирішені наступні завдання курсового проекту:

- виконано огляд сучасних програмних засобів реалізації візуалізації;
- розглянуті можливості та переваги мови програмування C#,
- розглянуті можливості та переваги середовища розробки JetBrains Rider;
- розроблено систему для відображення результатів роботи алгоритму.

ПЕРЕЛІК ПОСИЛАНЬ

1. Rider documentation [Електронний ресурс] – Режим доступу до ресурсу: https://www.jetbrains.com/rider/

2. .Net documentation [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/ru-ru/dotnet/api/system.numerics?view=netframework-4.8

3. «Алгоритм шифрования RSA» Учебная программа на C++ [Електронний ресурс] - Режим доступу до ресурсу: https://orenstudent.ru/RSA.htm

4. Image Cryptography using RSA Algorithm in C# [Електронний ресурс] - Режим доступу до ресурсу: https://www.codeproject.com/Articles/723175/Image-Cryptography-using-RSA-Algorithm-in-Csharp

5. Костриба О.В. Основи програмування. Частина 1. Visual C# Express Edition [Текст] / О.В. Костриба. – Білогір'я, 2008. – 74 с.

6. Плескач В.Л., Затонацька Т.Г. Інформаційні системи і технології на підприємствах [Текст] / В.Л. Плескач, Т.Г. Затонацька. – Київ, 2011. – 718 с.

7. Шнайер Б. Прикладна криптографія. Протоколи та алгоритми [Текст] / Б. Шнайер - М.: Триумф, 2003. - 806 с.

8. Ященко В.В. Вступ у криптографію [Текст] / В.В. Ященко - М.: МЦНМО, 1999. - 272 с.

9. Palzl J. Understanding Cryptography: A Textbook for Students and Practitioners [Текст] / J. Palzl – 2009. – 372 с.

10. Song Y. Cryptanalytic Attacks on RSA [Текст] / Y. Song - New York: Springer, 2008. - 270 с.

ДОДАТОК А
Текст програми

А.1 Текст файлу Form1.cpp

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


namespace WindowsFormsApp1
{
    public partial class Form1 : Form
    {
        //словник для шифрування і дешифрування
       char[] characters = new char[] {'а',' ', 'б', 'в', 'г', 'ґ', 'д', 'е', 'є', 'ж', 'з', 'и',
'і', 'ї', 'й', 'к',
           'л', 'м', 'н', 'о', 'п', 'р', 'с', 'т', 'у', 'ф', 'х', 'ц', 'ч', 'ш', 'щ', 'ь', 'ю',
'я', '-', ',',
           '.',  'А', 'Б', 'В', 'Г', 'Ґ', 'Д', 'Е', 'Є', 'Ж', 'З', 'И', 'І', 'Ї', 'Й', 'К', 'Л',
'М', 'Н', 'О', 'П',
           'Р', 'С', 'Т', 'У', 'Ф', 'Х', 'Ц', 'Ч', 'Ш', 'Щ', 'Ь', 'Ю', 'Я', '1', '2', '3', '4',
'5', '6', '7', '8', '9', '0', '!', '?' };
        //змінні, що є глобальними
        public static Image image;
        public static int[] mas;
        public byte[] bytes;
        public string s = "";
        public List<string> result = new List<string>();
        public List<string> input = new List<string>();
        public string resalt = "";
        public string strImg = "";
        public string stIm = "";
        public List<string> sI = new List<string>();
```

```
public Image imgFromStream;
// Завантаження вмісту призначеного для користувача інтерфейсу
public Form1()
{
    InitializeComponent();
}
// Функція перевірка чи є число простим
private bool IsTheNumberSimple(long n)
{
    if (n < 2)
        return false;

    if (n == 2)
        return true;

    for (long i = 2; i < n; i++)
        if (n % i == 0)
            return false;

    return true;
}
// Функція шифрування алгоритмом RSA
private List<string> RSA_Endoce(string s, long e, long n)
{
    List<string> result = new List<string>();

    BigInteger bi;

    for (int i = 0; i < s.Length; i++)
    {
        int index = Array.IndexOf(characters, s[i]);

        bi = new BigInteger(index);
        bi = BigInteger.Pow(bi, (int)e);

        BigInteger n_ = new BigInteger((int)n);

        bi = bi % n_;

        result.Add(bi.ToString());
    }
```

```
        return result;
    }
    // Функція дешифрування алгоритмом RSA
    private string RSA_Dedoce(List<string> input, long d, long n)
    {
        string result = "";

        BigInteger bi;

        foreach (string item in input)
        {
            bi = new BigInteger(Convert.ToDouble(item));
            bi = BigInteger.Pow(bi, (int)d);

            BigInteger n_ = new BigInteger((int)n);

            bi = bi % n_;

            int index = Convert.ToInt32(bi.ToString());

            result += characters[index].ToString();
        }

        return result;
    }
    // Функція визначення параметру e
    private long Calculate_e()
    {
        long d = 0;
        long[] mas = {3, 5, 7, 11, 17, 31};

        var ran = new Random();
        d = mas[ran.Next(0, 5)];


        return d;

    }

    // Функція визначення параметру d
    private long Calculate_d(long d, long m)
    {
```

```
            long e = 10;

            while (true)
            {
                if ((e * d) % m == 1)
                    break;
                else
                    e++;
            }

            return e;
        }
        // Відображення інструкції для користувача
        private void домопогаToolStripMenuItem_Click(object sender, EventArgs e)
        {
            string t = "";

            StreamReader sr = new
StreamReader(@"C:\Users\User\Desktop\WindowsFormsApp1\про_програму.txt");

            while (!sr.EndOfStream)
            {
                t += "\n";
                t += sr.ReadLine();
                t += "\n";
            }

            sr.Close();

            MessageBox.Show(t);
        }
       // Кнопка меню «Ключі»
        private void Ключі_Click(object sender, EventArgs e)
        {
            panel1.Show();
            panel2.Visible = true;
            panel1.Visible = false;
            panel3.Visible = false;
            textBox7.Enabled = false;
            textBox8.Enabled = false;
            textBox9.Enabled = false;
            textBox10.Enabled = false;
```

```
}
// Кнопка меню «Шифрувати»
private void Шифрувати_Click(object sender, EventArgs e)
{
    panel2.Show();
    panel1.Visible = true;
    panel2.Visible = false;
    panel3.Visible = false;
    textBox14.Enabled = false;
    textBox13.Enabled = false;
    textBox2.Enabled = false;
    button5.Enabled = false;
    button6.Enabled = false;
}
// Кнопка меню «Дешифрувати»
private void Дешифрувати_Click(object sender, EventArgs e)
{
    panel3.Show();
    panel3.Visible = true;
    panel2.Visible = false;
    panel1.Visible = false;
    textBox11.Enabled = false;
    textBox12.Enabled = false;
    textBox3.Enabled = false;
    button8.Enabled = false;
    button7.Enabled = false;
}
private void textBox4_Enter(object sender, EventArgs e)
{
    textBox4.Text = null;
    textBox4.ForeColor = Color.Black;
}
private void Form1_Load(object sender, EventArgs e)
{


    textBox4.Text = "Введыть текст для шифрування";//підказка
    textBox4.ForeColor = Color.Black;
    textBox1.Text = "Введыть текст для розшифрування";//підказка
    textBox1.ForeColor = Color.Black;
}
private void textBox1_Enter(object sender, EventArgs e)
```

```
        {
            textBox1.Text = null;
            textBox1.ForeColor = Color.Black;
        }
        // Кнопка для завантаження зображення і створення масиву байт з нього
        private void ЗавантажитиЗображення_Click(object sender, EventArgs e)
        {
            OpenFileDialog open_dialog = new OpenFileDialog();
            open_dialog.Filter = "Image Files(*.BMP;*.JPG;*.PNG;)|*.BMP;*.JPG;*.PNG|All files
(*.*)|*.*";
            if (open_dialog.ShowDialog() == DialogResult.OK)
            {
                try
                {
                    image = new Bitmap(open_dialog.FileName);
                    pictureBox1.SizeMode = PictureBoxSizeMode.StretchImage;
                    pictureBox1.Image = image;
                    pictureBox1.Invalidate();
                }
                catch
                {
                    DialogResult rezult = MessageBox.Show("Неможливо відкрити зображення",
                        "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }
            var ms = new MemoryStream();
            if (image.RawFormat.Equals(System.Drawing.Imaging.ImageFormat.Jpeg))
            {
                image.Save(ms, System.Drawing.Imaging.ImageFormat.Jpeg);
            }
            if (image.RawFormat.Equals(System.Drawing.Imaging.ImageFormat.Png))
            {
                image.Save(ms, System.Drawing.Imaging.ImageFormat.Png);
            }
            if (image.RawFormat.Equals(System.Drawing.Imaging.ImageFormat.Bmp))
            {
                image.Save(ms, System.Drawing.Imaging.ImageFormat.Bmp);
            }
            bytes = ms.ToArray();
            mas = bytes.Select(i =>(int) i).ToArray();
            textBox11.Enabled = true;
            textBox12.Enabled = true;
```

```
            button8.Enabled = true;
            button12.Enabled = false;
            button9.Enabled = false;
        }
        // Кнопка для завантаження файлу, що містить шифрований текст
        private void ШифрФайл_Click(object sender, EventArgs e)
        {
            OpenFileDialog open_dialog = new OpenFileDialog();
            open_dialog.Filter = "Image Files(*.txt)|*.txt|All files (*.*)|*.*";
            if (open_dialog.ShowDialog() == DialogResult.OK)
            {
                try
                {
                    StreamReader sr = new StreamReader(open_dialog.FileName);
                    while (!sr.EndOfStream)
                    {
                        input.Add(sr.ReadLine());
                    }
                    sr.Close();
                }
                catch
                {
                    DialogResult rezult = MessageBox.Show("Неможливо відкрити документ",
                        "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }
            textBox1.Clear();
            foreach (string item in input)
            {
                textBox1.Text += item + " ";
            }
            textBox13.Enabled = true;
            textBox14.Enabled = true;
            button5.Enabled = true;
            button6.Enabled = false;
        }
        // Кнопка для завантаження зашифрованого масиву та додаткової інформації до нього
        private void ШифрЗоб_Click(object sender, EventArgs e)
        {
            OpenFileDialog open_dialog = new OpenFileDialog();
            open_dialog.Filter = "Image Files(*.BMP;*.JPG;*.PNG;)|*.BMP;*.JPG;*.PNG|All files
(*.*)|*.*";
```

```
if (open_dialog.ShowDialog() == DialogResult.OK)
{
    try
    {
        image = new Bitmap(open_dialog.FileName);
        pictureBox2.SizeMode = PictureBoxSizeMode.StretchImage;
        pictureBox2.Image = image;
        pictureBox2.Invalidate();
    }
    catch
    {
        DialogResult rezult = MessageBox.Show("Неможливо відкрити зображення",
            "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
var ms = new MemoryStream();
if (image.RawFormat.Equals(System.Drawing.Imaging.ImageFormat.Jpeg))
{
    image.Save(ms, System.Drawing.Imaging.ImageFormat.Jpeg);
}
if (image.RawFormat.Equals(System.Drawing.Imaging.ImageFormat.Png))
{
    image.Save(ms, System.Drawing.Imaging.ImageFormat.Png);
}
if (image.RawFormat.Equals(System.Drawing.Imaging.ImageFormat.Bmp))
{
    image.Save(ms, System.Drawing.Imaging.ImageFormat.Bmp);
}
bytes = ms.ToArray();
mas = bytes.Select(i =>(int) i).ToArray();
OpenFileDialog open_dialog1 = new OpenFileDialog();
open_dialog1.Filter = "Image Files(*.txt)|*.txt|All files (*.*)|*.*";
if (open_dialog1.ShowDialog() == DialogResult.OK)
{
    try
    {
        StreamReader sr = new StreamReader(open_dialog1.FileName);
        while (!sr.EndOfStream)
        {
            sI.Add(sr.ReadLine());
        }
```

```
            sr.Close();
        }
        catch
        {
            DialogResult rezult = MessageBox.Show("Неможливо відкрити документ",
                "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }

    textBox13.Enabled = true;
    textBox14.Enabled = true;
    button5.Enabled = true;
    button6.Enabled = false;
}
// Кнопка генерування ключів
private void ГенеруванняКлючів_Click(object sender, EventArgs e)
{
    if ((textBox5.Text.Length > 0) && (textBox6.Text.Length > 0))
    {
        long p = Convert.ToInt64(textBox5.Text);
        long q = Convert.ToInt64(textBox6.Text);
        if (IsTheNumberSimple(p) && IsTheNumberSimple(q))
        {
            button1.Enabled = false;
            button2.Enabled = false;
            button3.Enabled = false;
            textBox5.Enabled = false;
            textBox6.Enabled = false;
            long n = p * q;
            long m = (p - 1) * (q - 1);
            long e_ = Calculate_e();
            long d = Calculate_d(e_, m);
            textBox7.Text = d.ToString();
            textBox8.Text = n.ToString();
            textBox9.Text = e_.ToString();
            textBox10.Text = n.ToString();
            textBox5.Enabled = true;
            textBox6.Enabled = true;
            textBox7.Enabled = false;
            textBox8.Enabled = false;
            textBox9.Enabled = false;
            textBox10.Enabled = false;
```

```
                    button1.Enabled = true;
                    button2.Enabled = true;
                    button3.Enabled = true;
                }
                else
                    MessageBox.Show("p чи q - не росте число!");
            }
            else
                MessageBox.Show("Введіть p чи q!");

            List<string> result = new List<string>();

            result.Add("Секретний ключ: ");
            result.Add(textBox7.Text);
            result.Add(textBox8.Text);
            result.Add("Публічний ключ: ");
            result.Add(textBox9.Text);
            result.Add(textBox10.Text);

            StreamWriter sw = new
StreamWriter(@"C:\Users\User\Desktop\WindowsFormsApp1\key.txt");
            foreach (string item in result)
                sw.WriteLine(item);
            sw.Close();
            MessageBox.Show("Ключі збережено до файлу key.txt");
        }
       //Кнопка шифрування інформації

        private void ШифруватиІнф_Click(object sender, EventArgs e)
        {
            if (textBox4.Text.Length < 0 && pictureBox1.Image == null)
            {
                MessageBox.Show("Заповніть текстове поле, чи завантажте зображення!");
            }
            else
            {
                if (textBox4.Text.Length > 0)
                {
                    if ((textBox11.Text.Length > 0) && (textBox12.Text.Length > 0))
                    {
                        textBox11.Enabled = false;
                        textBox12.Enabled = false;
                        textBox4.Enabled = false;
```

```
            textBox3.Enabled = false;
            button8.Enabled = false;
            button9.Enabled = false;
            button12.Enabled = false;
            button7.Enabled = false;

            long e_ = Convert.ToInt64(textBox11.Text);
            long n = Convert.ToInt64(textBox12.Text);
            result = RSA_Endoce(s, e_, n);

            for (int i = 0; i < result.Count; i++)
                textBox3.Text += result[i] + " ";

            textBox11.Enabled = true;
            textBox12.Enabled = true;
            textBox4.Enabled = true;
            textBox3.Enabled = true;
            button8.Enabled = true;
            button9.Enabled = true;
            button12.Enabled = true;
            button7.Enabled = true;
        }
        else
            MessageBox.Show("Заповніть поля ключів!");
    }
    if (pictureBox1.Image != null)
    {
        if ((textBox11.Text.Length > 0) && (textBox12.Text.Length > 0))
        {
            textBox11.Enabled = false;
            textBox12.Enabled = false;
            textBox4.Enabled = false;
            textBox3.Enabled = false;
            button8.Enabled = false;
            button9.Enabled = false;
            button12.Enabled = false;
            button7.Enabled = false;

            long e_ = Convert.ToInt64(textBox11.Text);
            long n = Convert.ToInt64(textBox12.Text);

            var ran = new Random();
```

```
            int[] mos = new int[20];
            int[] g = new int[20];

            for (int i = 0; i < 20; i++)
            {
                g[i] = ran.Next(100, mas.Length);
                mos[i] = mas[g[i]];
            }

            strImg = string.Join(" ", mos);

            result = RSA_Endoce(strImg, e_, n);
            string som = string.Join(" ", result);
            int[] ia = som.Split(' ').Select(w => Convert.ToInt32(w)).ToArray();
            for (int i = 0; i < 20; i++)
            {
                mas[g[i]] = ia[i];
            }

            strImg = string.Join(" ", g);
            stIm = string.Join(" ", ia);

            byte[] b = mas.Select(i => (byte)i).ToArray();
            var imageMemoryStream = new MemoryStream(b);
            imgFromStream = Image.FromStream(imageMemoryStream);
            pictureBox1.SizeMode = PictureBoxSizeMode.StretchImage;
            pictureBox1.Image = imgFromStream;

            textBox11.Enabled = true;
            textBox12.Enabled = true;
            textBox4.Enabled = true;
            textBox3.Enabled = true;
            button8.Enabled = true;
            button9.Enabled = true;
            button12.Enabled = true;
            button7.Enabled = true;
        }
        else
            MessageBox.Show("Заповніть поля ключів!");

    }
}
```

```
}
// Кнопка завантаження файлу
private void ЗавантажитиФайл_Click(object sender, EventArgs e)
{
    OpenFileDialog open_dialog = new OpenFileDialog();
    open_dialog.Filter = "Image Files(*.txt)|*.txt|All files (*.*)|*.*";
    if (open_dialog.ShowDialog() == DialogResult.OK)
    {
        try
        {
            StreamReader sr = new StreamReader(open_dialog.FileName);
            while (!sr.EndOfStream)
            {
                s += sr.ReadLine();
            }
            sr.Close();
        }
        catch
        {
            DialogResult rezult = MessageBox.Show("Неможливо відкрити документ",
                "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
    textBox4.Text = s;

    textBox11.Enabled = true;
    textBox12.Enabled = true;
    button8.Enabled = true;
    button12.Enabled = false;
    button9.Enabled = false;
}
//Кнопка збереження шифрованої інформації
private void Зберигти_Click(object sender, EventArgs e)
{
    if (textBox3.Text.Length < 0 && pictureBox1.Image == null)
    {
        MessageBox.Show("Заповніть текстове поле, чи завантажте зображення!");
    }
    else
    {
        if (textBox3.Text.Length > 0)
```

```
                {
                    SaveFileDialog savedialog = new SaveFileDialog();
                    savedialog.Title = "Зберегти як...";
                    savedialog.OverwritePrompt = true;
                    savedialog.CheckPathExists = true;
                    savedialog.Filter = "Image Files(*.txt)|*.txt|All files (*.*)|*.*";
                    savedialog.ShowHelp = true;
                    if (savedialog.ShowDialog() == DialogResult.OK)
                    {
                        try
                        {
                            StreamWriter sw = new StreamWriter(savedialog.FileName);
                            foreach (string item in result)
                                sw.WriteLine(item);
                            sw.Close();
                            MessageBox.Show("Зашифрований текс збережено!");
                        }
                        catch
                        {
                            MessageBox.Show("Неможливо зберегти", "Помилка",
                                MessageBoxButtons.OK, MessageBoxIcon.Error);
                        }
                    }
                }
                if (pictureBox1.Image != null)
                {
                    SaveFileDialog savedialog = new SaveFileDialog();
                    savedialog.Title = "Зберегти зображення як...";
                    savedialog.OverwritePrompt = true;
                    savedialog.CheckPathExists = true;
                    savedialog.Filter = "Image Files(*.jpg)|*.jpg | All files (*.*)|*.*";
                    savedialog.ShowHelp = true;
                    if (savedialog.ShowDialog() == DialogResult.OK)
                    {
                        try
                        {
                            imgFromStream.Save(savedialog.FileName,
System.Drawing.Imaging.ImageFormat.Jpeg);
                            MessageBox.Show("Зашифроване зображення збережено!");
                        }
                        catch
                        {
```

```
                    MessageBox.Show("Неможливо зберегти зображення", "Помилка",
                        MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }

            SaveFileDialog savedialog1 = new SaveFileDialog();
            savedialog1.Title = "Зберегти як...";
            savedialog1.OverwritePrompt = true;
            savedialog1.CheckPathExists = true;
            savedialog1.Filter = "Image Files(*.txt)|*.txt|All files (*.*)|*.*";
            savedialog1.ShowHelp = true;
            if (savedialog1.ShowDialog() == DialogResult.OK)
            {
                try
                {
                    StreamWriter sw = new StreamWriter(savedialog1.FileName);
                    sw.WriteLine(strImg);
                    sw.WriteLine(stIm);
                    sw.Close();
                    MessageBox.Show("Допоміжну інформацію збережено!");
                }
                catch
                {
                    MessageBox.Show("Неможливо зберегти", "Помилка",
                        MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }
        }
    }
}
// Кнопка дешифрування інформації

private void Дешифрування_Click(object sender, EventArgs e)
{
    if (textBox4.Text.Length < 0 && pictureBox2.Image == null)
    {
        MessageBox.Show("Заповніть текстове поле, чи завантажте зображення!");
    }
    else
    {
        if (textBox4.Text.Length > 0)
        {
            if ((textBox13.Text.Length > 0) && (textBox14.Text.Length > 0))
```

```
    {
        textBox13.Enabled = false;
        textBox14.Enabled = false;
        textBox1.Enabled = false;
        textBox2.Enabled = false;
        button10.Enabled = false;
        button6.Enabled = false;
        button5.Enabled = false;
        button11.Enabled = false;

        long d = Convert.ToInt64(textBox13.Text);
        long n = Convert.ToInt64(textBox14.Text);
        resalt = RSA_Dedoce(input, d, n);
        textBox2.Text = resalt;

        textBox13.Enabled = true;
        textBox14.Enabled = true;
        textBox1.Enabled = true;
        textBox2.Enabled = true;
        button10.Enabled = true;
        button6.Enabled = true;
        button5.Enabled = true;
        button11.Enabled = true;
    }
    else
        MessageBox.Show("Заповніть поля ключів!");
}
if (pictureBox2.Image != null)
{
    if ((textBox13.Text.Length > 0) && (textBox14.Text.Length > 0))
    {
        textBox13.Enabled = false;
        textBox14.Enabled = false;
        textBox1.Enabled = false;
        textBox2.Enabled = false;
        button10.Enabled = false;
        button6.Enabled = false;
        button5.Enabled = false;
        button11.Enabled = false;

        long d = Convert.ToInt64(textBox13.Text);
        long n = Convert.ToInt64(textBox14.Text);
```

```
            long[] ia = new long[20];
            int[] ar = new int[20];

            ia = sI[0].Split(' ').Select(v => Convert.ToInt64(v)).ToArray();
            int[] kj = sI[1].Split(' ').Select(v => Convert.ToInt32(v)).ToArray();
            ar = ia.Select(i => (int)i).ToArray();
            int[] ms = new int[20];

            for (int i = 0; i < 20; i++)
            {
                ms[i] = mas[ar[i]];
            }

            string rlt = string.Join(" ", kj);
            List<string> res = rlt.Split(' ').ToList();
            resalt = RSA_Dedoce(res, d, n);
            int[] ai = resalt.Split(' ').Select(k => Convert.ToInt32(k)).ToArray();

            for (int i = 0; i < 20; i++)
            {
                mas[ar[i]] = ai[i];
            }

            byte[] b = mas.Select(i => (byte)i).ToArray();
            var imageMemoryStream = new MemoryStream(b);
            imgFromStream = Image.FromStream(imageMemoryStream);
            pictureBox2.SizeMode = PictureBoxSizeMode.StretchImage;
            pictureBox2.Image = imgFromStream;

            textBox13.Enabled = true;
            textBox14.Enabled = true;
            textBox1.Enabled = true;
            textBox2.Enabled = true;
            button10.Enabled = true;
            button6.Enabled = true;
            button5.Enabled = true;
            button11.Enabled = true;
        }
        else
            MessageBox.Show("Заповніть поля ключів!");
    }
}
```

```
}
// Кнопка збереження дешифрованої інформації
private void ЗберигтиДешифр_Click(object sender, EventArgs e)
{
    if (textBox2.Text.Length < 0 && pictureBox2.Image == null)
    {
        MessageBox.Show("Заповніть текстове поле, чи завантажте зображення!");
    }
    else
    {
        if (textBox2.Text.Length > 0)
        {
            SaveFileDialog savedialog = new SaveFileDialog();
            savedialog.Title = "Зберегти як...";
            savedialog.OverwritePrompt = true;
            savedialog.CheckPathExists = true;
            savedialog.Filter = "Image Files(*.txt)|*.txt|All files (*.*)|*.*";
            savedialog.ShowHelp = true;
            if (savedialog.ShowDialog() == DialogResult.OK)
            {
                try
                {
                    StreamWriter sw = new StreamWriter(savedialog.FileName);
                    sw.WriteLine(resalt);
                    sw.Close();
                }
                catch
                {
                    MessageBox.Show("Неможливо зберегти", "Помилка",
                        MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }
            MessageBox.Show("Дешифрований текс збережено!");
        }
        if (pictureBox2.Image != null)
        {
            SaveFileDialog savedialog = new SaveFileDialog();
            savedialog.Title = "Зберегти зображення як...";
            savedialog.OverwritePrompt = true;
            savedialog.CheckPathExists = true;
            savedialog.Filter = "Image Files(*.jpg)|*.jpg | All files (*.*)|*.*";
            savedialog.ShowHelp = true;
```

```
                    if (savedialog.ShowDialog() == DialogResult.OK)
                    {
                        try
                        {
                            imgFromStream.Save(savedialog.FileName,
System.Drawing.Imaging.ImageFormat.Jpeg);
                            MessageBox.Show("Зображення збережено!");
                        }
                        catch
                        {
                            MessageBox.Show("Неможливо зберегти зображення", "Помилка",
                                MessageBoxButtons.OK, MessageBoxIcon.Error);
                        }
                    }
                }
            }
        }
        // Відображення інформації стосовно авторів проекту
        private void проАвторівToolStripMenuItem1_Click(object sender, EventArgs e)
        {
            string t = "";

            StreamReader sr = new
StreamReader(@"C:\Users\User\Desktop\WindowsFormsApp1\про_авторів.txt");

            while (!sr.EndOfStream)
            {
                t += "\n";
                t += sr.ReadLine();
                t += "\n";
            }

            sr.Close();

            MessageBox.Show(t);
        }
        // Відображення інформації стосовно призначення проекту
        private void Інструкція_Click(object sender, EventArgs e)
        {
            string t = "";

            StreamReader sr = new
```

```
StreamReader(@"C:\Users\User\Desktop\WindowsFormsApp1\Інструкція.txt");

            while (!sr.EndOfStream)
            {
                t += "\n";
                t += sr.ReadLine();
                t += "\n";
            }

            sr.Close();

            MessageBox.Show(t);
        }
    }
}
```

ДОДАТОК Б
Інтерфейс програми

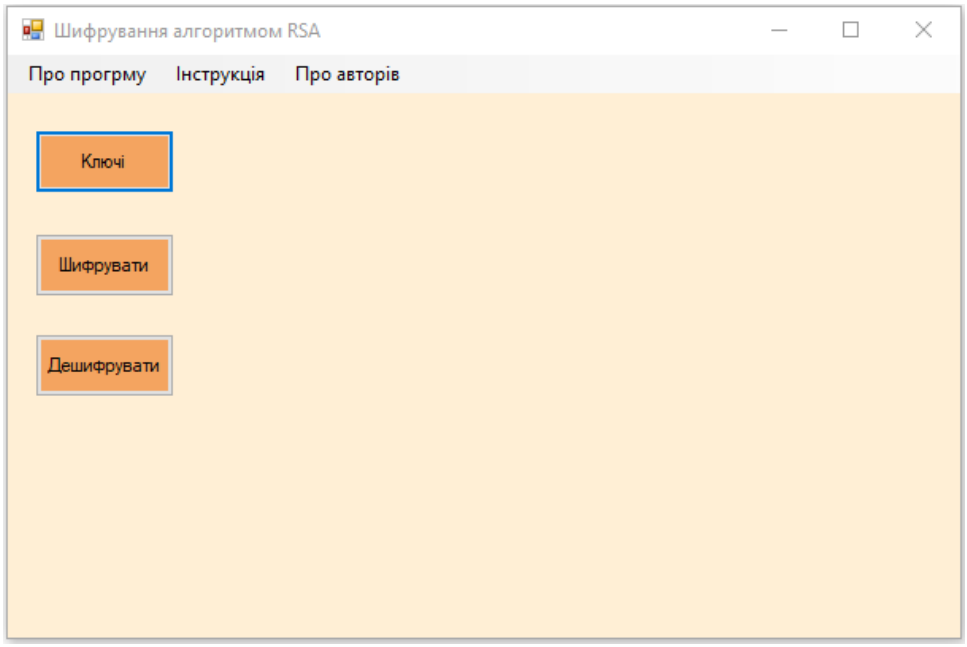

Рисунок Б.1 – Головне вікно програми

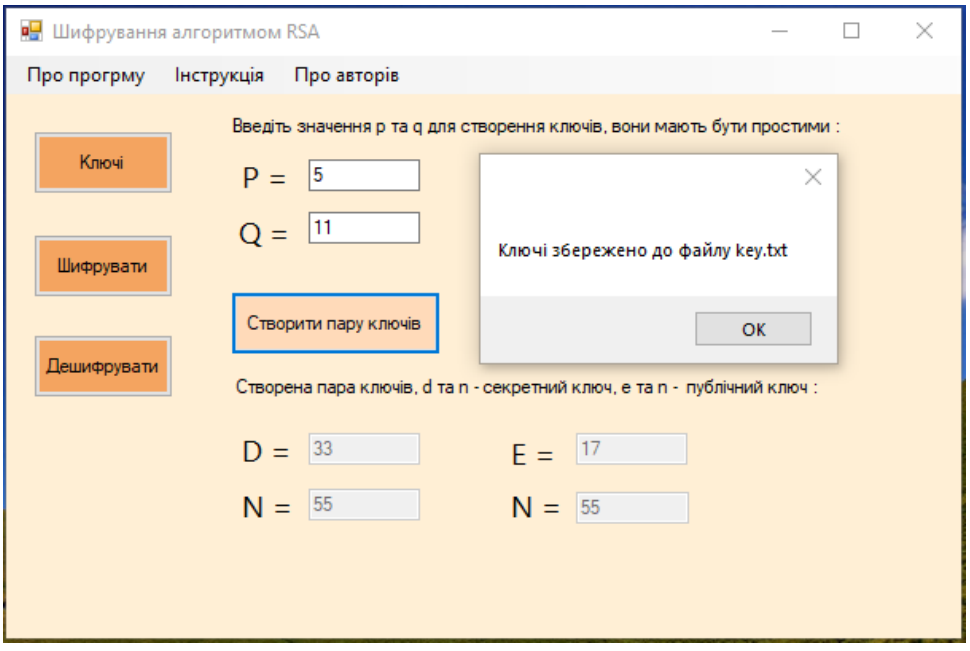

Рисунок Б.2 – Вікно панелі створення ключів

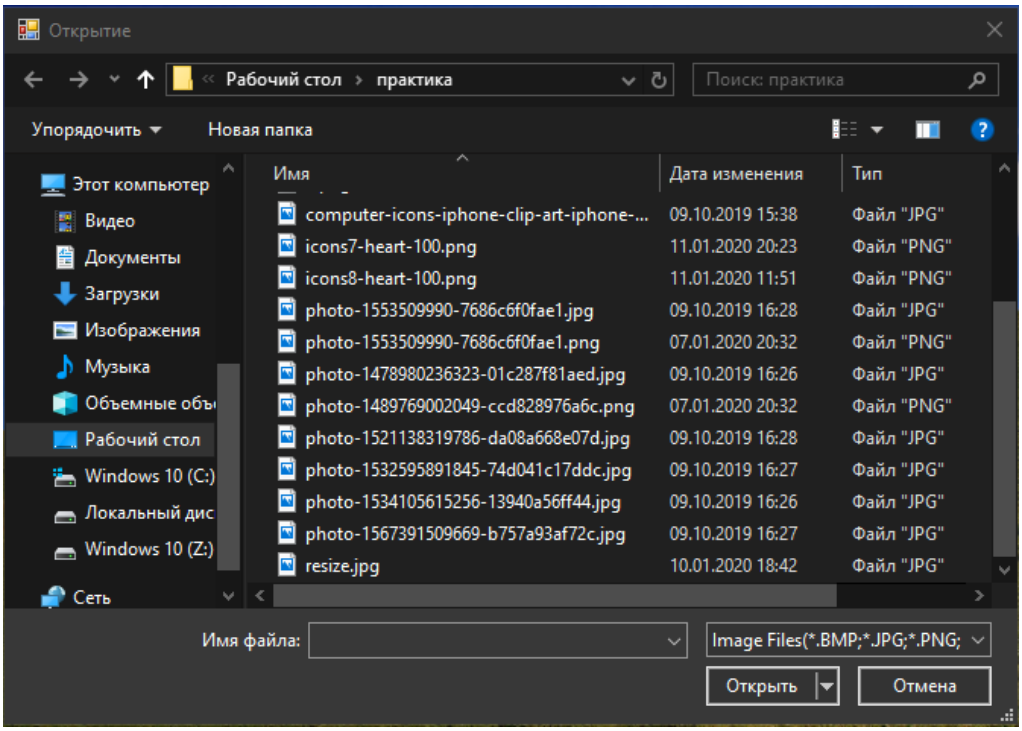

Рисунок Б.3 – Вікно для завантаження інформації

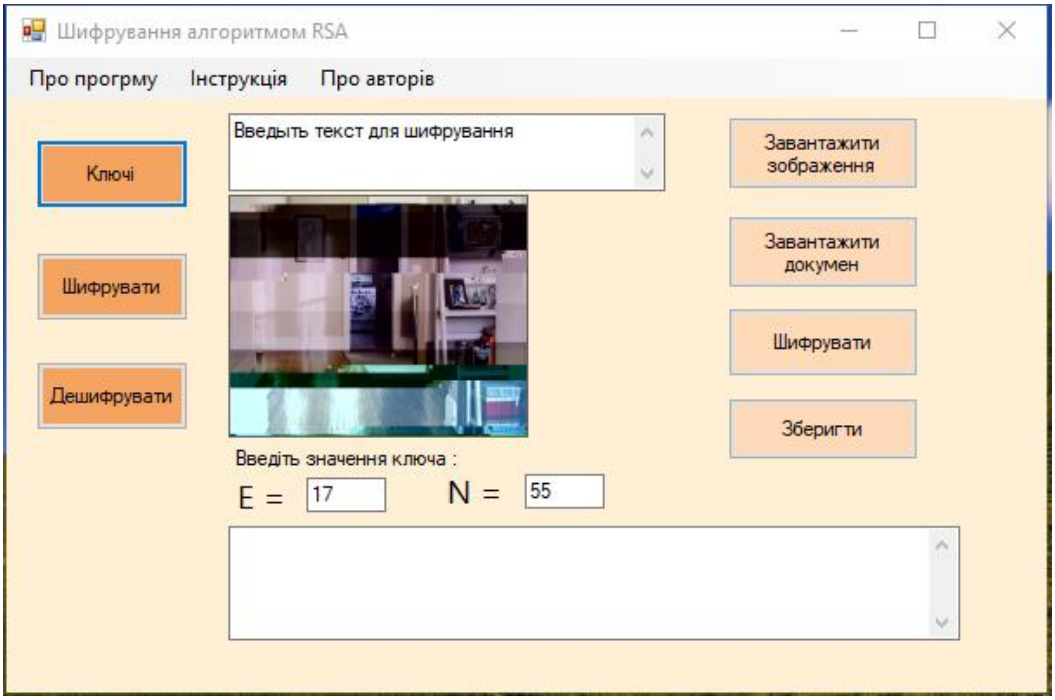

Рисунок Б.4 – Панель шифрування

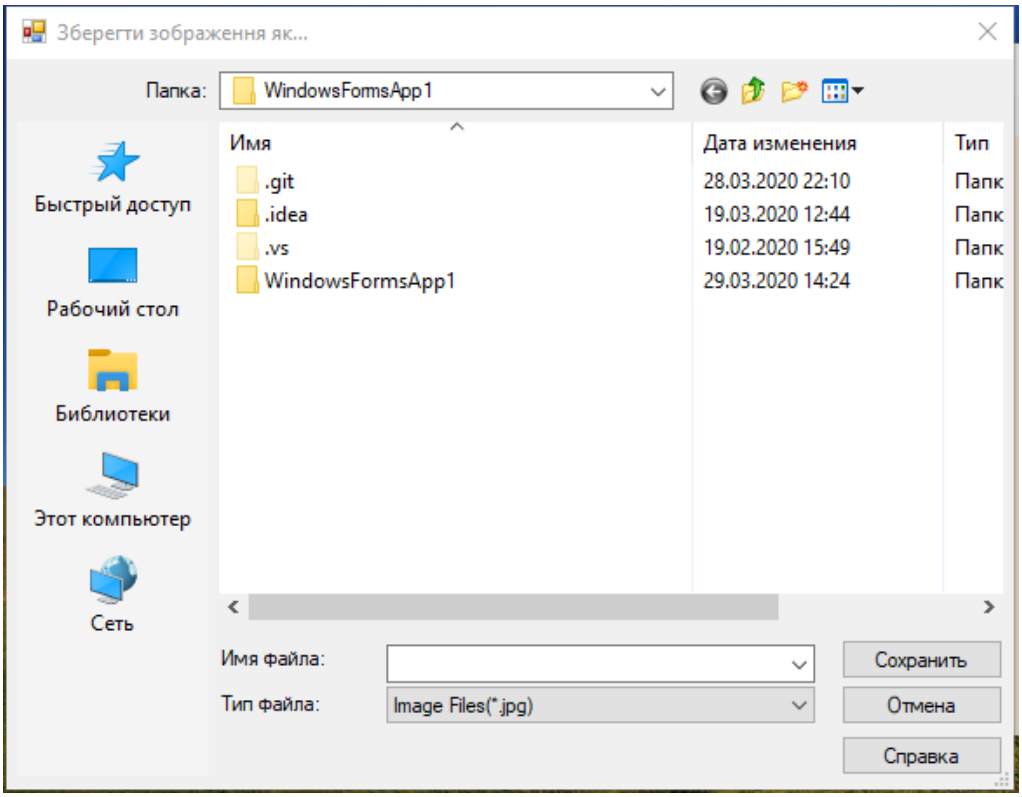

Рисунок Б.5 – Вікно для збереження інформації

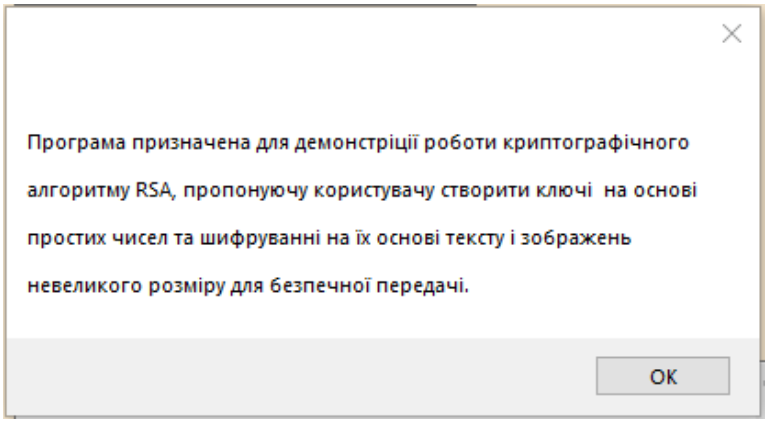

Рисунок Б.6 – Форма «Про програму»

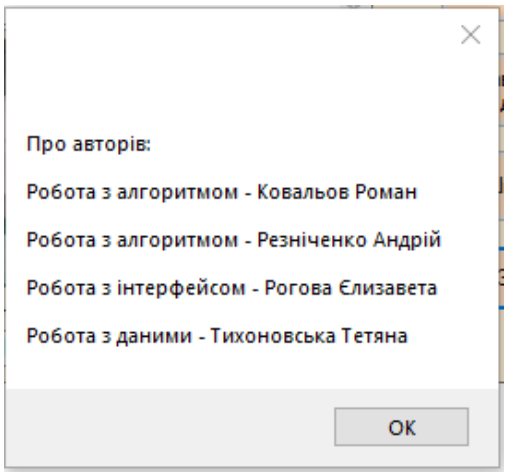

Рисунок Б.7 – Форма «Автори»

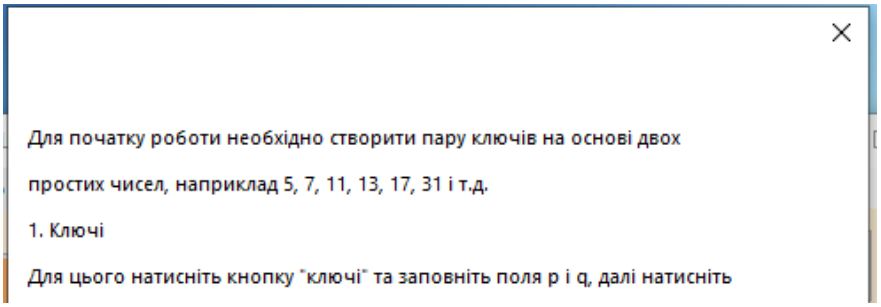

Рисунок Б.8 – Форма «Інструкція»

ДОДАТОК В
Слайди презентації

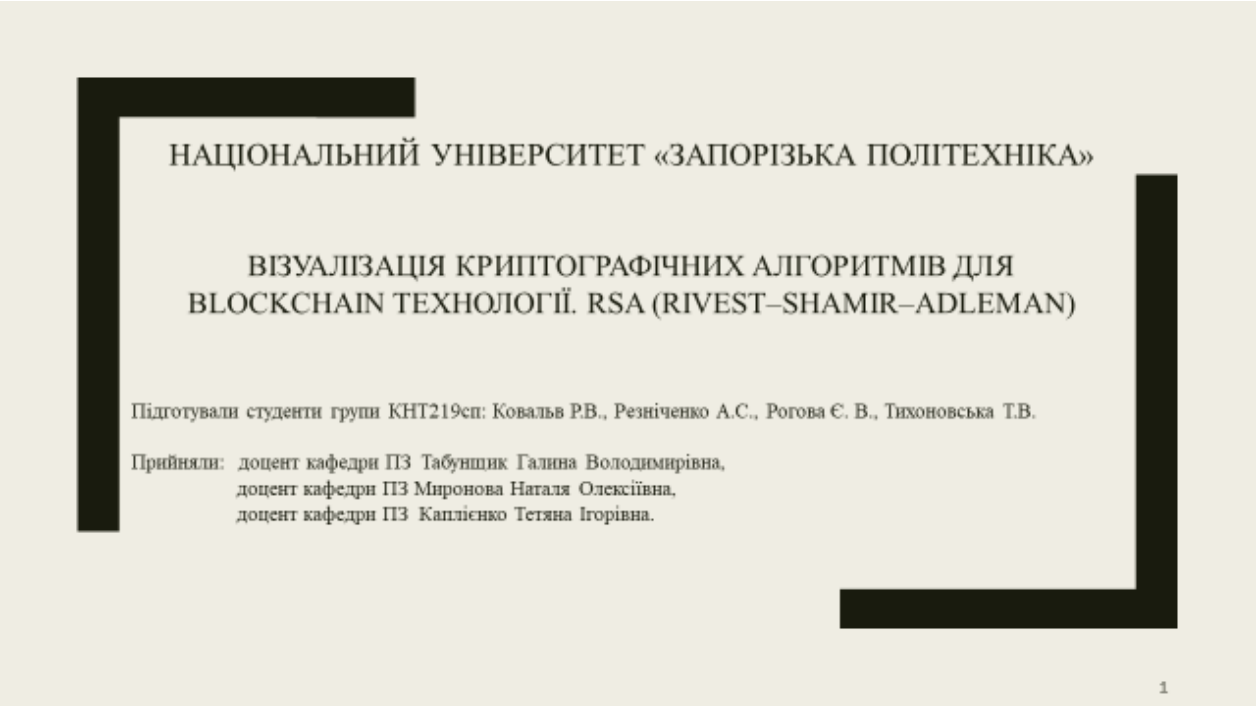

Рисунок В.1 – Титульний лист

Рисунок В.2 – Постановка задачі

Рисунок В.3 – Мова програмування та середовище розробки

Рисунок В.4 – Клас MemoryStream

Рисунок В.5 – Структура BigInteger

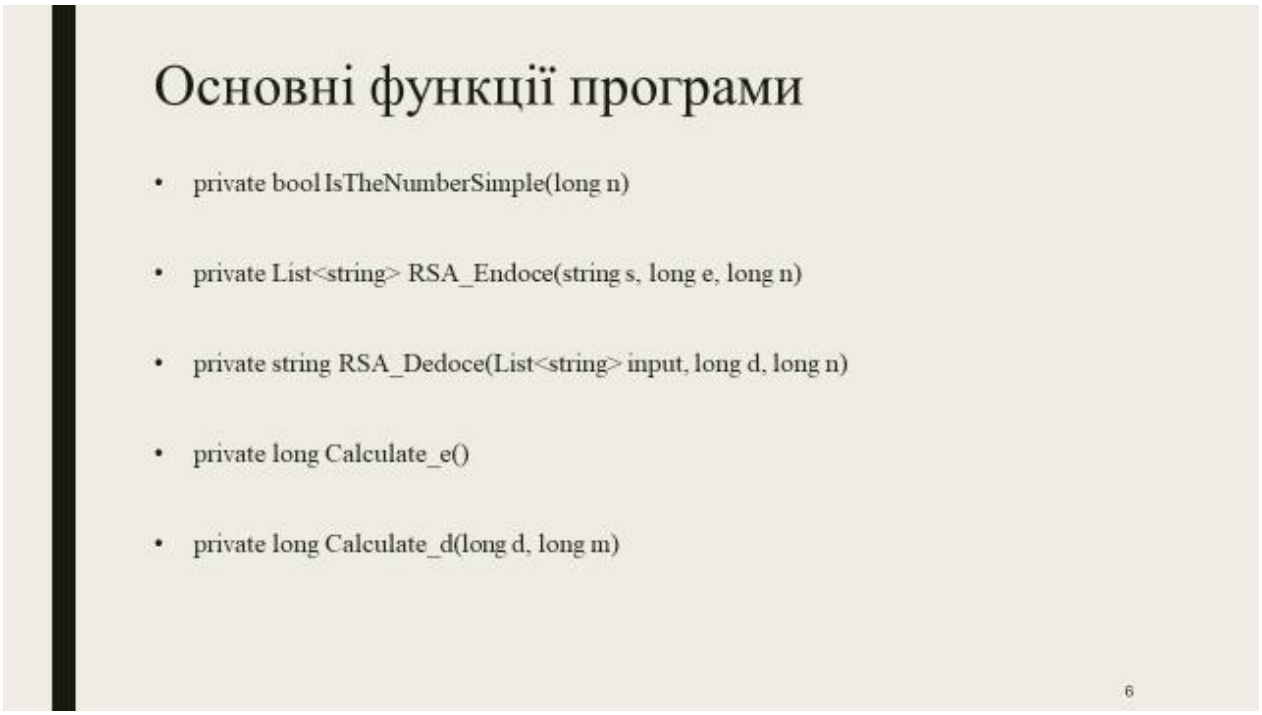

Рисунок В.6 – Основні функції програми

Рисунок В.7 – Алгоритм RSA

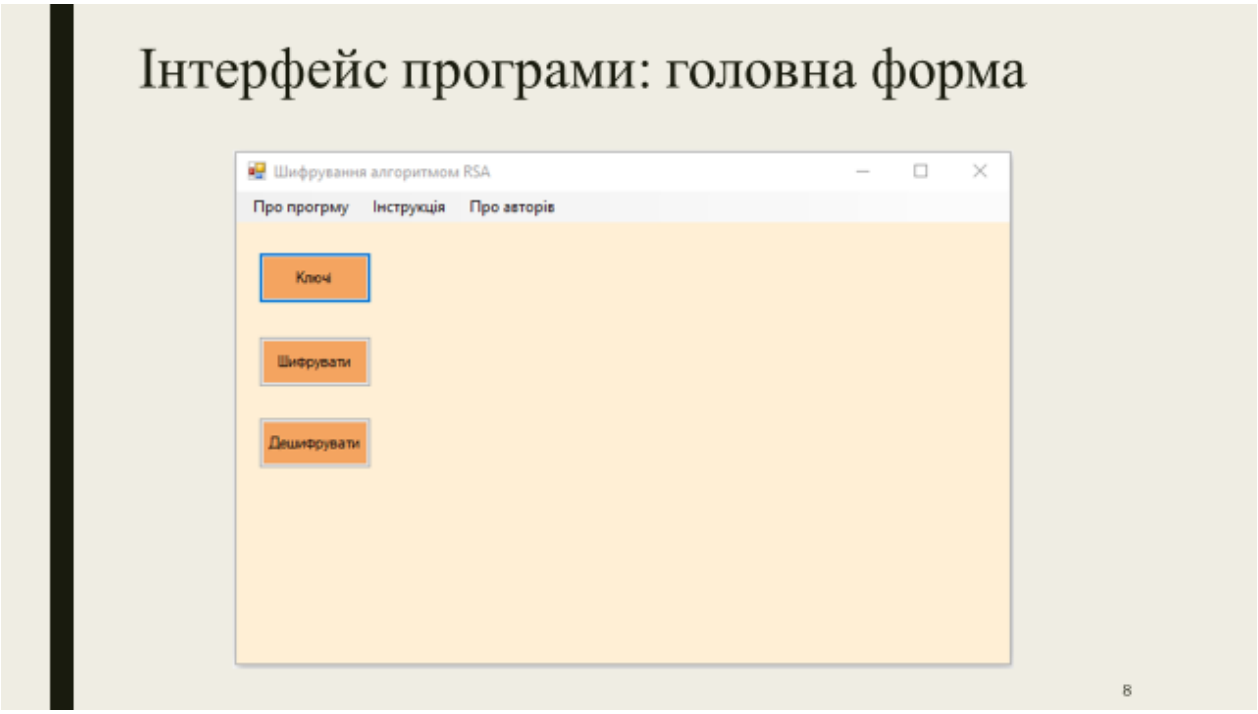

Рисунок В.8 – Інтерфейс програми: головна форма

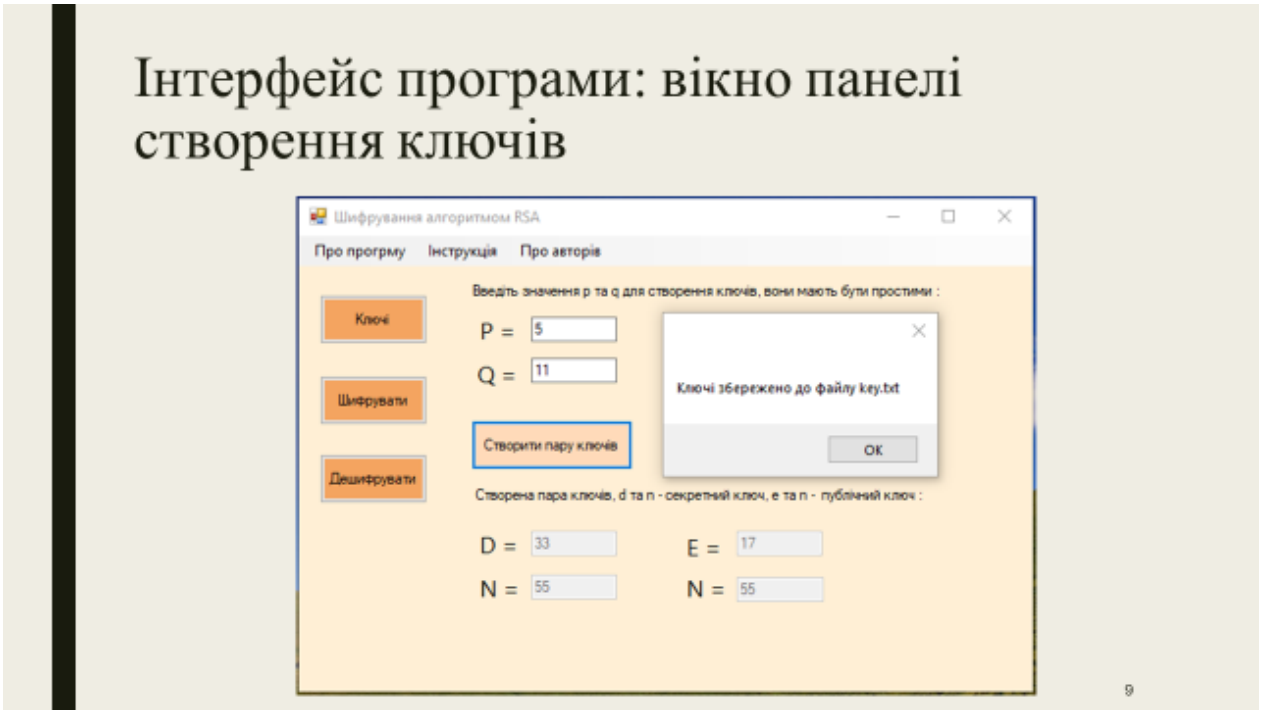

Рисунок В.9 – Інтерфейс програми: вікно панелі створення ключів

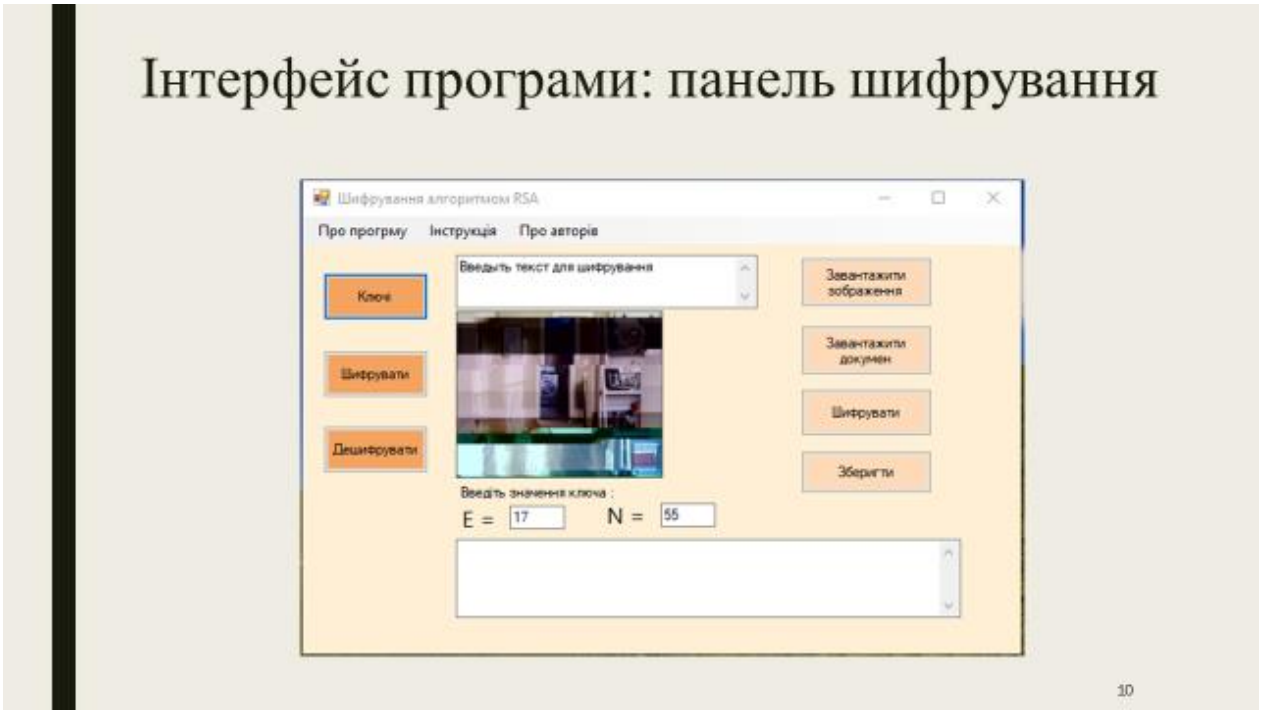

Рисунок В.10 – Інтерфейс програми: панель шифрування

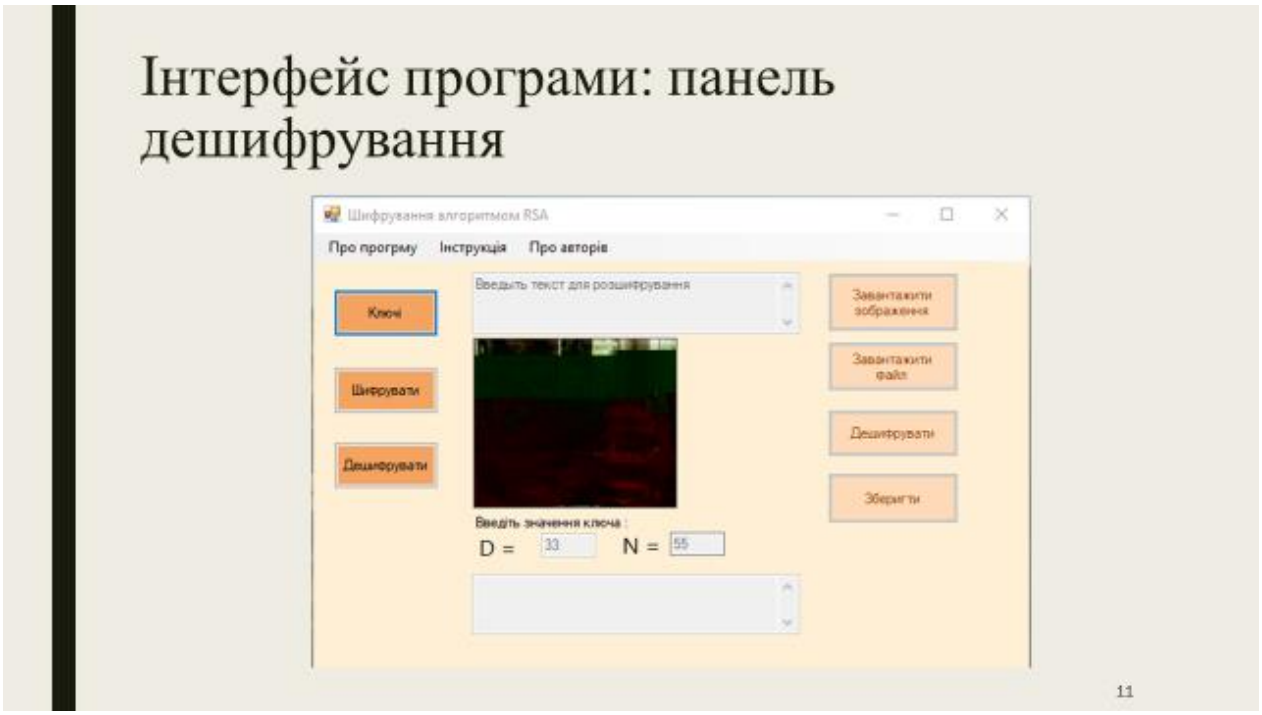

Рисунок В.11 – Інтерфейс програми: панель дешифрування

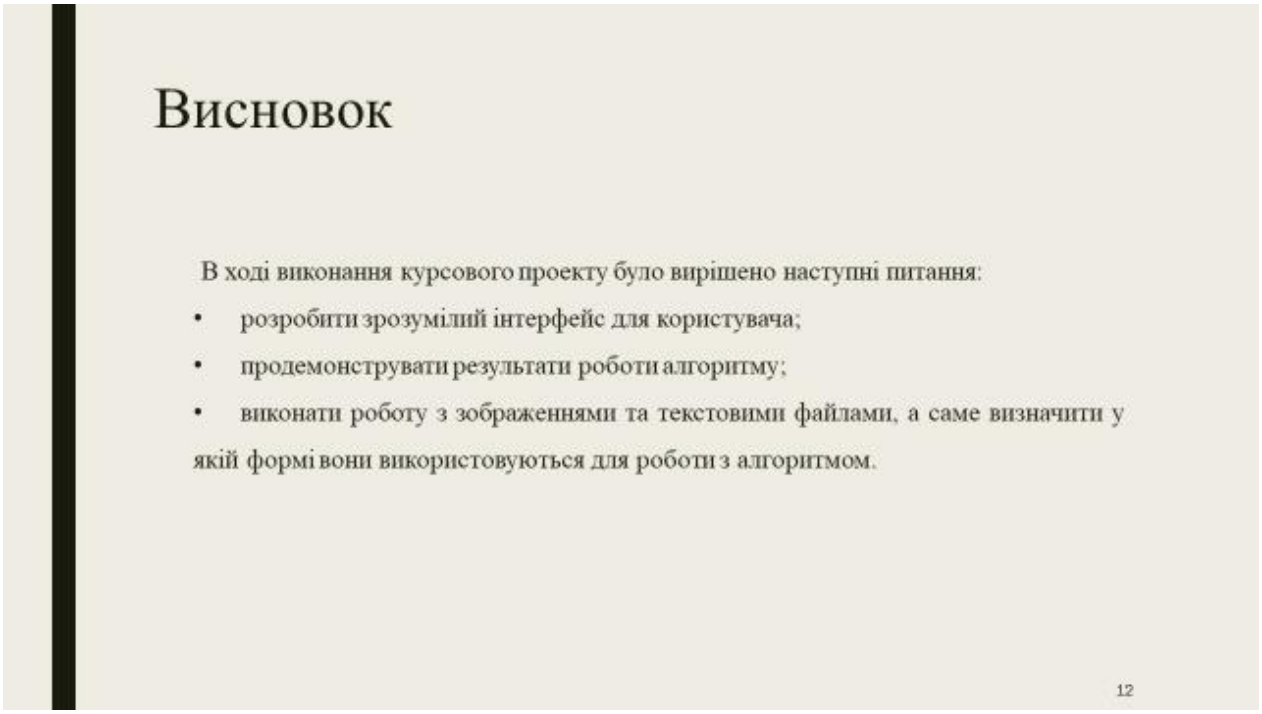

Рисунок В.12 – Висновок

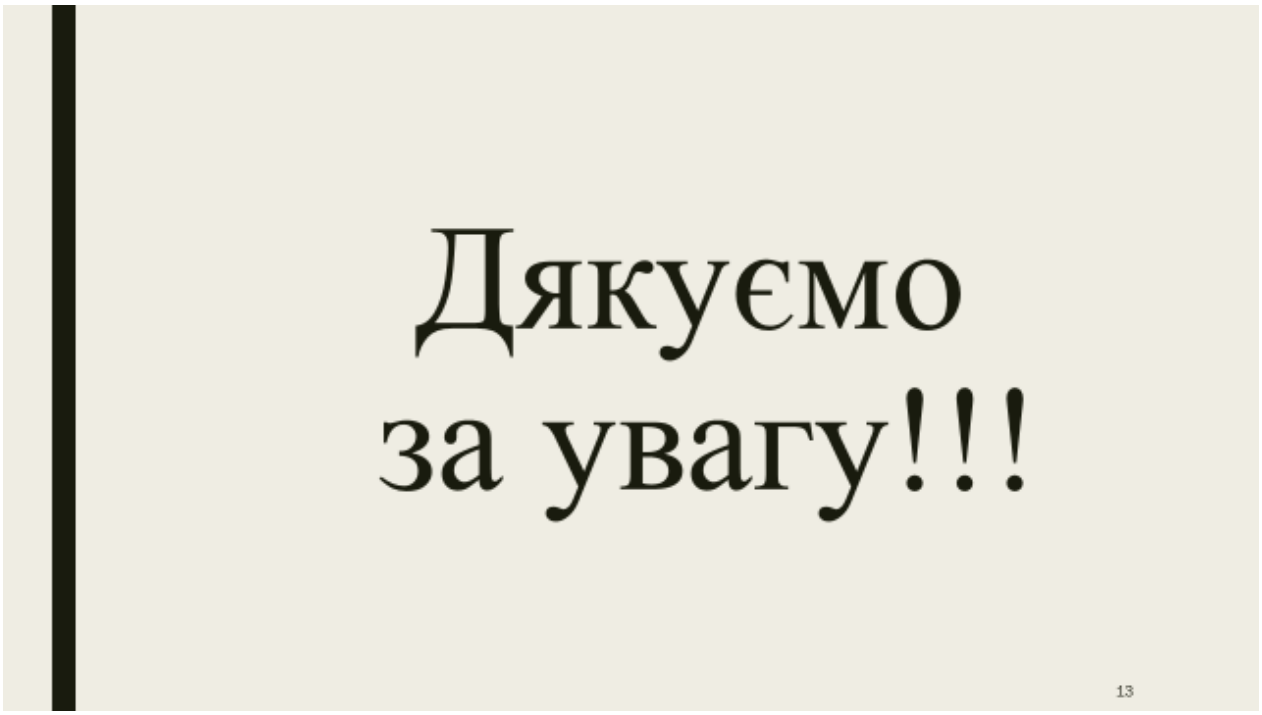

Рисунок В.13 – Заключний слайд