**МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ**

**ЧЕРКАСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ім. Б. ХМЕЛЬНИЦЬКОГО**

**КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ АВТОМАТИЗОВАНИХ СИСТЕМ**

ЗАТВЕРДЖУЮ

Завідувач кафедри \_\_\_\_\_\_,

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис)

„\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_2019 р.

**ІНДИВІДУАЛЬНЕ ЗАВДАННЯ**

**на курсову роботу**

студента\_\_\_\_\_\_\_\_Сапейко М.О.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ групи \_ КС-181 \_ першого курсу

**ТЕМА** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Шифр Віженера\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Вихідні дані до курсової роботи:** Методичка з курсового проектування\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Зміст **Пояснювальної записки** до курсової роботи:

Індивідуальне завдання

Вступ

1 Огляд інформаційних джерел

2 Проектування програмного продукту

3 Реалізація програмного продукту

Висновки

Список літератури

Додатки

Перелік наочного матеріалу:

Програмний продукт, пояснювальна записка, презентація результатів роботи.

**Календарний план виконання роботи:**

|  |  |  |  |
| --- | --- | --- | --- |
| №  п/п | Назва етапу дипломного проекту (роботи) | Термін виконання етапу: до … | Примітка |
| 1. | Отримання завдання на курсову роботу. | 28.01.2019 |  |
| 2. | Огляд джерел технічної інформації за темою роботи. | 29.01.2019 |  |
| 3. | Виконання аналізу методів реалізації завдання | 10.03.2019 |  |
| 3. | Розробка алгоритму реалізації завдання | 10.03.2019 |  |
| 4. | Програмування алгоритму, створення програмного продукту | 11.03.2019 |  |
| 5. | Тестування розробленого програмного продукту | 11.03.2019 |  |
| 6. | Написання пояснювальної записки | 03.05.2019 |  |
| 7. | Подача роботи керівнику для написання відгуку. | 07.05.2019 |  |
| 8. | Корегування роботи за результатами розгляду керівника. Остаточне оформлення пояснювальної роботи. | 09.05.2019 |  |
| 9. | Написання доповіді, створення слайдів. | 10.05.2019 |  |
| 10. | Захист курсової роботи | 14.05.2019 |  |

Студент \_\_\_\_\_\_\_\_\_\_\_\_ / \_\_\_\_\_\_\_\_\_\_\_/ Науковий керівник \_\_\_\_\_\_\_\_\_\_\_\_ / \_\_\_\_\_\_\_\_\_\_\_/

(підпис) (Прізвище, ініціали) (підпис) (Прізвище, ініціали)

Завдання видане «\_\_\_» \_\_\_\_\_\_\_\_\_ 2019 року

**Зміст**

[ВСТУП 3](#_Toc8658049)

[Розділ 1. Огляд інформаційних джерел 4](#_Toc8658050)

[**1.1** **Початок огляду.** 4](#_Toc8658051)

[**1.2 Метод Віженера.** 4](#_Toc8658052)

[**1.2** **Вибір мови програмування** 9](#_Toc8658053)

[**1.4 Висновок до першого розділу** 9](#_Toc8658054)

[Розділ 2. Проектування програмного продукту 10](#_Toc8658055)

[**2.1 Алгоритм шифру Віженера.** 10](#_Toc8658056)

[**2.2 Графічний інтерфейс** 10](#_Toc8658057)

[**2.3 Висновок до другого розділу** 12](#_Toc8658058)

[Розділ 3. Реалізація програмного продукту 13](#_Toc8658059)

[**3.1** **Реалізація алгоритму Віженера** 13](#_Toc8658060)

[**3.2** **Реалізація інтерфейсу** 15](#_Toc8658061)

[**3.3** **Тестування додатку** 16](#_Toc8658062)

[**3.4** **Висновки до третього розділу** 21](#_Toc8658063)

[Висновок 22](#_Toc8658064)

[Список використаних джерел 23](#_Toc8658065)

[Додаток А. Блок-схема алгоритму Віженера 24](#_Toc8658066)

[Додаток Б. Лістинг алгоритму Віженера 26](#_Toc8658067)

**ВСТУП**

**Мета**: удосконалити практичні навички створення програмних проектів, по криптографічним перетворенням інформації, ознайомитися з основними методами криптографічного захисту інформації.

**Завдання**: розробити програму шифрування за допомогою шифру Віженера для перетворення тексту, на латиниці, в зашифрований текст, а потім дешифрувати його. За потреби мати змогу зберегти отриманий результат.

**Постановка задач курсової роботи:**

1. Дослідити метод шифрування Віженера.
2. Побудова структурної схеми програмного додатку та блок-схем основних алгоритмів.
3. Розробити додаток, який перетворює текст на латиниці в зашифрований текст, а також дешифрує його.
4. Написання пояснювальної записки та підготовка доповіді з описом процесу реалізації поставленого завдання курсової роботи.

**Актуальність теми:** в даний час безпека є критичним завданням у багатьох галузях, а тому зацікавленість інформаційного співтовариства в більш стійких до злому, а також більш швидких в роботі алгоритмів може і буде проявлятися як зараз, так і надалі. В умовах сучасного інформаційного суспільства проблема захисту інформації гостро постає перед людством. Саме технології шифрування, які базуються на математичних методах, є останньою лінією оборони та забезпечують безпеку інформаційних даних.

**Базові поняття:**

*Шифр –* це математична формула, що описує процеси шифрування і розшифрування.

*Шифрування –* це спосіб зміни повідомлення або іншого документа, що забезпечує спотворення його вмісту.

*Дешифрування –* це обернений процес відтворення інформації із зашифрованого тексту.

**Розділ 1. Огляд інформаційних джерел**

1. **Початок огляду.**

Відомо велику кількість статей про різноманітні методи шифрування та їх реалізації. І перед тим як почати написання власного додатку методом Віженера, потрібно ретельно його дослідити.

1. **Метод Віженера.**

Шифр Віженера (фр. Chiffre de Vigenère) - метод поліалфавітного шифрування буквеного тексту з використанням ключового слова.

Цей метод є простою формою поліалфавітної заміни. Шифр Віженер винаходився багаторазово. Вперше цей метод описав Джован Баттіста Беллазо (італ. Giovan Battista Bellaso) у книзі «La cifra del. Sig. Giovan Battista Bellasо» в 1553 році, проте в XIX столітті отримав ім'я Блеза Віженера, французького дипломата. Метод простий для розуміння і реалізації.

Суть шифрування Віженера схожа з шифром Цезаря, з тією лише різницею, що якщо шифр Цезаря зіставляє для всіх символів повідомлення (відкритого тексту, приховуваного в кодуванні) одне і те ж значення зсуву, то в шифрі Віженера для кожного символу відкритого тексту задане власне значення зсуву. Це означає, що довжина ключа шифру Віженера повинна бути рівною довжині повідомлення. Однак запам'ятати такий ключ розшифрування, якщо повідомлення буде довгим, непросто. З цього скрутного становища виходять так: за ключ шифру Віженера беруть слово (фразу), зручне для запам'ятовування, слово (кодова фраза) повторюється до тих пір, поки не стане рівним довжині повідомлення. Отриману послідовність символів і використовують для шифрування шифром Віженера за допомогою таблиці алфавітів, так звана tabula recta або квадрат (таблиця) Віженера (Рисунок 1.1).. Якщо за основу взяти латинський алфавіт, то таблиця Віженера складатиметься з рядків по 26 символів, причому кожний наступний рядок зсувається на одну позицію. Таким чином, в таблицю входить 26 різних шифрів Цезаря. [2].

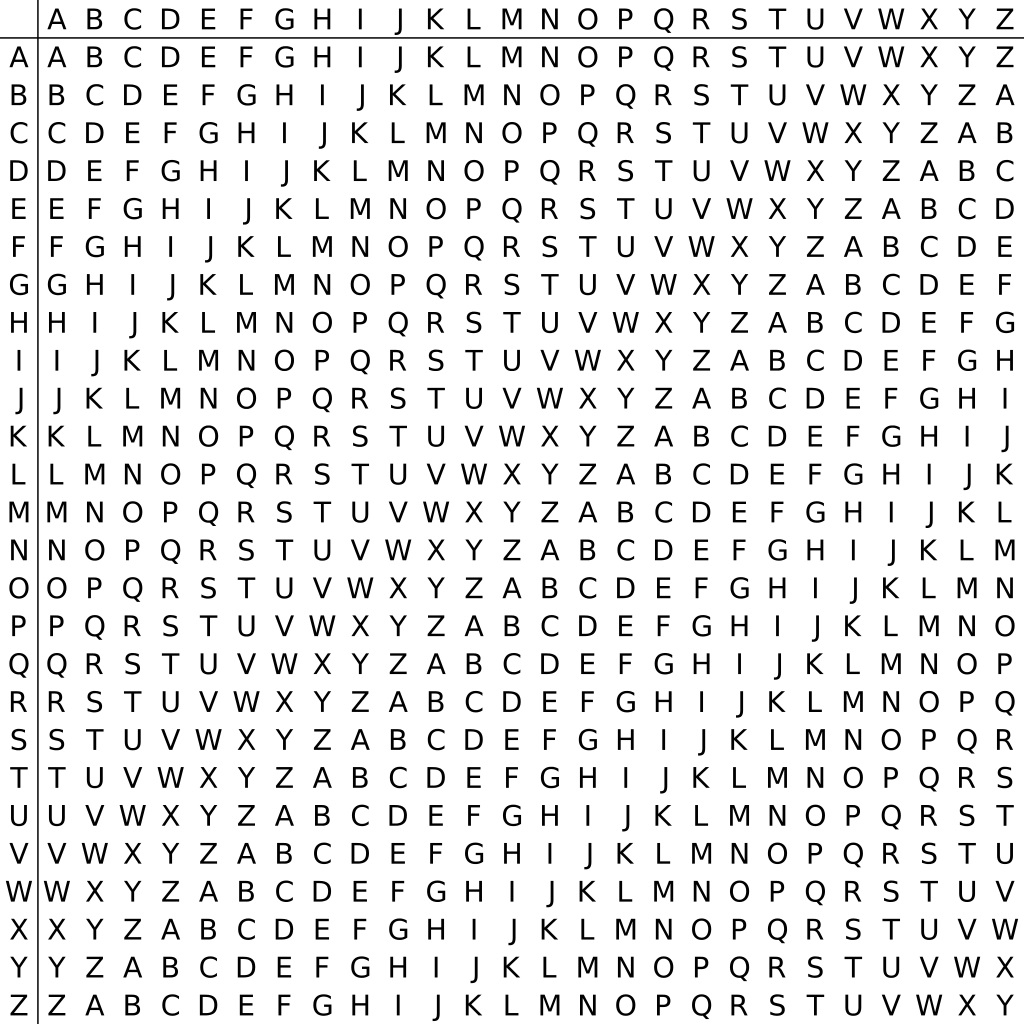


Рисунок 1.1 Таблиця Віженера

При шифруванні вихідного повідомлення його виписують у рядок, а під ним записують ключове слово (або фразу). Якщо ключ виявився коротший за повідомлення, то його циклічно повторюють. У процесі шифрування знаходять у верхньому рядку таблиці чергову літеру вихідного тексту, а у лівому стовпці – чергове значення ключа. Літеру шифротексту знаходять на перетинанні стовпця, обумовленого літерою, що шифрують, та рядка, обумовленого відповідною літерою ключа.

Але на мою думку, цей метод є важкий для розуміння і реалізації, тому я схиляюсь більше до представлення шифру у вигляді формули.

Якщо букви A-Z відповідають числам 0-25, то шифрування Віженера можна записати у вигляді формули:

(1)

Дешифрування:

(2)

де, зашифрована літера,

- літера повідомлення,

- літера ключа.

26 – довжина алфавіту.

Ключ - це кодове слово, яке використовується при процесі шифрування, грає велику роль при процесі дешифрування. При записі ключа можна використовувати тільки символи латинського алфавіту і не використовувати пробіл. Ключ може бути як безглуздим набором символів, так і деяким словом або смисловим словосполученням, але записаним без пробілу. При шифруванні тексту методом Віженера дуже важливу роль відіграє довжина ключа, так як від довжини ключа залежить складність процесу дешифрування або злому, тобто чим довший ключ тим складніше буде зламати даний шифр. [6].

Приклад:

За основу беремо латинський алфавіт, відповідно довжина алфавіту 26 символів. Для полегшення роботи використаємо таблицю (Таблиця 1.1.) відповідності букв латинського алфавіту до їхнього порядкового номеру.

Таблиця 1.1

Таблиця відповідності букв латинського алфавіту до їхнього порядкового номеру.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **D** | **E** | **F** | **G** | **H** | **I** | **J** | **K** | **L** | **M** |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** |
| **N** | **O** | **P** | **Q** | **R** | **S** | **T** | **U** | **V** | **W** | **X** | **Y** | **Z** |
| **13** | **14** | **15** | **16** | **17** | **18** | **19** | **20** | **21** | **22** | **23** | **24** | **25** |

Приклад повідомлення: NETWORKING

Ключем нехай буде слово BOX, записуємо його то тих пір поки воно не буде рівним довжині повідомлення.

BOXBOXBOXB

Беремо перший символ повідомлення і переведемо його в цифрову форму N=13;

Таким же чином переводимо в цифрову форму перший символ ключа B=1;

Шифруємо перші два символи за алгоритмом

(3)

Отже отримана літера O. Повторимо операцію для 2-го символу. E=4; O=14

(4)

Результат шифрування всього повідомлення:

Оригінальний текст: NETWORKING

Ключ: BOXBOXBOXB

Зашифрований текст: OSQXCOLWKH

Для прикладу розшифруємо першу букву повідомлення O=14; B=1;

(5)

Виконувати поставлене завдання курсової роботи можна різними способами, тобто існують різні методи подання шифру Віженера. Більшість розглянутих мною методів призводили до того, що для зчитування вхідного повідомлення виконувалася операція по переведенню тексту в регістр малих або ж великих літер, тобто в результаті завжди був текст який складався з малих чи великих літер. Мені такий метод не подобається, тому що якщо ми при шифруванні вказували слова, яким важливо містити велику літеру, то при дешифровці ці літери втрачали свій сенс. Наведем приклад:

Вхідний текст: NetWorking

Ключ: bOx

Зашифрований текст: osqxcolwkh

І коли ми будемо намагатися розшифрувати отриманий текст, то в результаті отримаємо:

networking

Як бачимо за таким методом не відбувається повне збереження оригінального тексту. Для вирішення цієї проблеми можна створити алфавіт, масив символів char, в якому міститимуться малі та великі літери латинського алфавіту. Особливістю такого методу є те, що результат шифрування в точності не відповідає розміру відповідних символів оригінального тексту. Наведемо приклад:

Вхідний текст: NetWorking

Ключ: bOx

Зашифрований текст: OSQXcOlWKh

Дешифрований текст: NetWorking

На мою думку такий метод більш якісний, також збільшується складність зламати такий шифр.

1. **Вибір мови програмування**

Перед початком виконання поставленого завдання, необхідно визначитися з мовою програмування. На мою думку найоптимальнішим варіантом стане мова C#, оскільки саме вона вивчалась протягом року, а також, тому що робота з графікою не вимагає чогось більшого ніж те, на що здатна C#.

На сьогоднішній момент мова програмування C# одна з найпотужніших, що швидко розвивається та затребуваних мов в ІТ-галузі. На даний момент на ній пишуться найрізноманітніші програми: від невеликих десктопних програмок до великих веб-порталів і веб-сервісів, які обслуговують щодня мільйони користувачів.

У порівнянні з іншими мовами C# досить молода, але в той же час вона вже пройшла великий шлях. [4].

1. **Висновок до першого розділу**

Було детально розібрано алгоритм шифрування Віженера, а також обрана мова програмування. Також виявлено певні тонкощі даного шифру, а саме формування ключового слова. Розглянуто різні способи подання методу шифрування Віженера, та спроектовано свій варіант, який передає даний метод.

**Розділ 2. Проектування програмного продукту**

1. **Алгоритм шифру Віженера.**

Блок-схему зображено в додатку А. Сам алгоритм для шифрування та дешифровки тексту більшою мірою схожий один на одного, відрізняється лише частина в якій розміщенні формули (1) та (2). Тому, виходячи з цього, щоб не робити, окремо, два алгоритми їх було поєднано в один.

Ретельно дослідивши алгоритм, було розроблено алгоритм, який міститиме в собі декілька частини, а саме: переведення отриманого вхідного повідомлення та ключа в два масиви символів char, потім розміщуємо цикл який буде перебирати кожен символ вхідного повідомлення, в цьому ж циклі розміщуємо ще один цикл котрий буде шукати індекс букви, отриманого символу, в алфавіті, якщо такої букви не знайдено в алфавіті, то це значить що символ не з алфавіту, тому пропускаємо його, якщо ж індекс букви було знайдено, то виходимо з циклу пошуку індексу букви та переходимо до наступного кроку, а саме задаємо умову щоб наше ключове слово повторювалось та заповнювала вхідний текст, далі створюється цикл в якому відбувається пошук індексу букви ключа, після цього відбувається перевірка того, який самий метод обрав користувач, зашифрувати чи дешифрувати текст, визначившись з методом підставляємо значення у відповідну формулу та отримуємо результат для першого символу вхідного повідомлення, який записується в рядок, після чого переходимо до наступного символу та виконуємо аналогічні дії. Якщо алгоритм проходить по всім символам вхідного повідомлення, то виходимо з циклу яким ми проходимо по кожному символу та встановлюємо нульову позицію заповнення ключа, для того щоб, в разі повторного виклику алгоритму, ключ почав заповнювати текст із першого свого символу. В кінці складаємо отримані дані в рядок.

На цьому робота алгоритму закінчена.

1. **Графічний інтерфейс**

Для створення графічного інтерфейсу була обрана технологія WF (Windows Forms), яка є частиною екосистеми платформи .NET і являє собою підсистему для побудови графічних інтерфейсів.

Так як форми є основною частиною програми, необхідно приділити особливу увагу їх зовнішньому вигляду і функціям. В кінцевому рахунку форма являє собою чистий аркуш, який розробник оснащує елементами управління, формуючи призначений для користувача інтерфейс, і кодом для роботи з даними. З цією метою Visual Studio дозволяє середовище розробки (IDE), яке спрощує написання коду, а також розширений набір елементів управління .NET Framework. Доповнюючи функціональні можливості цих елементів управління своїм кодом, ви можете легко і швидко розробити необхідну програму.

При створенні традиційних додатків на основі WinForms за відображення елементів управління і графіки відповідають такі частини OC Windows, як User32 і GDI+. Даний інтерфейс спрощує доступ до елементів інтерфейсу Microsoft Windows за допомогою створення обгортки для Win32 API в керованому коді. Всередині .NET Framework, Windows Forms реалізується в межах простору імен System.Windows.Forms. [5].

1. **Висновок до другого розділу**

В другому розділі було детально розібрано шифр Віженера. Показано особливість шифрування, яка зроблена в саме цьому додатку, що відрізняє його від інших. Було спроектовано сам алгоритм за допомогою блок-схеми.

**Розділ 3. Реалізація програмного продукту**

1. **Реалізація алгоритму Віженера**

Роботу програми можна поділити на три частини:

1. Введення даних.
2. Обробка даних.
3. Виведення результату.

Отже, спочатку необхідно ввести (або відкрити) вхідне повідомлення та записати ключ. Введення даних відбувається за допомогою елементу “TextBox”, при чому на введення ключ накладаються додаткові обмеження, оскільки ключ повинен складатися з одного слова, тому накладаємо обмеження на введення пробілу.

Наведемо фрагмент коду в якому описано реакцію програми на введення пробілу в ключове слово (Лістинг 3.1.)

Лістинг 3.1.

private void textBoxKey\_TextChanged(object sender, EventArgs e)

{

// Перевіряємо ключ на наявність пробілів

for (int i = 0; i < textBoxKey.TextLength; i++)

{

char ch = Convert.ToChar(textBoxKey.Text.Substring(i, 1));

if (ch == ' ')

{

MessageBox.Show("Введений пробiл");

}

}

}

Файли відкриваються за допомогою елементу “openFileDialog”.

Наведемо фрагмент коду, який відповідає за відкриття файлу (Лістинг 3.2.)

Лістинг 3.2.

private void button3\_Click(object sender, EventArgs e)

{

// Відкриття файлу

if (openFileDialog1.ShowDialog() == DialogResult.Cancel)

return;

// Отримуємо обраний файл

string filename = openFileDialog1.FileName;

// Читаємо файл в рядок

string fileText = System.IO.File.ReadAllText(filename);

// Передаємо текстбоксу дані з файлу

textBoxMassage.Text = fileText;

}

Наступний крок – в разі натиску на кнопку “Зашифрувати,” почне працювати алгоритм шифрування Віженера. Далі виводиться на TextBox зашифрований текст, який не можна редагувати, для того щоб мати змогу його коректно дешифрувати, для цього у властивостях TextBox`а знаходимо ReadOnly та встановлюємо значення true, також зашифрований текст можемо зберегти за допомогою елементу “saveFileDialog”.

Наведемо фрагмент коду, який відповідає за збереження отриманого результату (Лістинг 3.3.)

Лістинг 3.3.

private void button4\_Click(object sender, EventArgs e)

{

// Збереження в (\*.txt)

saveFileDialog1.Filter = "txt files (\*.txt)|\*.txt";

saveFileDialog1.DefaultExt = "txt";

saveFileDialog1.AddExtension = true;

// Зберігаємо файл

if (saveFileDialog1.ShowDialog() == DialogResult.OK)

{

// Отримуємо шлях до файлу

string filename = saveFileDialog1.FileName;

// Зберігаємо текст в файл

System.IO.File.WriteAllText(filename, textBoxResult.Text);

MessageBox.Show("Файл збережено");

}

}

В разі натиску на кнопку “Розшифрувати,” почне працювати алгоритм дешифрування Віженера. Далі виводиться на TextBox дешифрований текст, який також можемо зберегти.

Тобто, дивлячись на код алгоритму (Додаток Б), можна побачити, що для шифрування, що для дешифрування виконуються однакові дії.

1. **Реалізація інтерфейсу**

В цій програмі графічний інтерфейс є дуже простим. Він складається всього з декількох частин, а саме: два поля для вводу вхідних даних та ключового слова, поле, на яке буде виводитися результат обраного алгоритму, шість кнопок, які відповідають за запуск алгоритму шифрування або дешифрування, очищення вмісту TextBox`ів, а також відкриття та збереження тексту, та відкриття інструкції до програми. Все це було дуже легко реалізувати за допомогою Windows Forms. (Рисунок 3.1.)

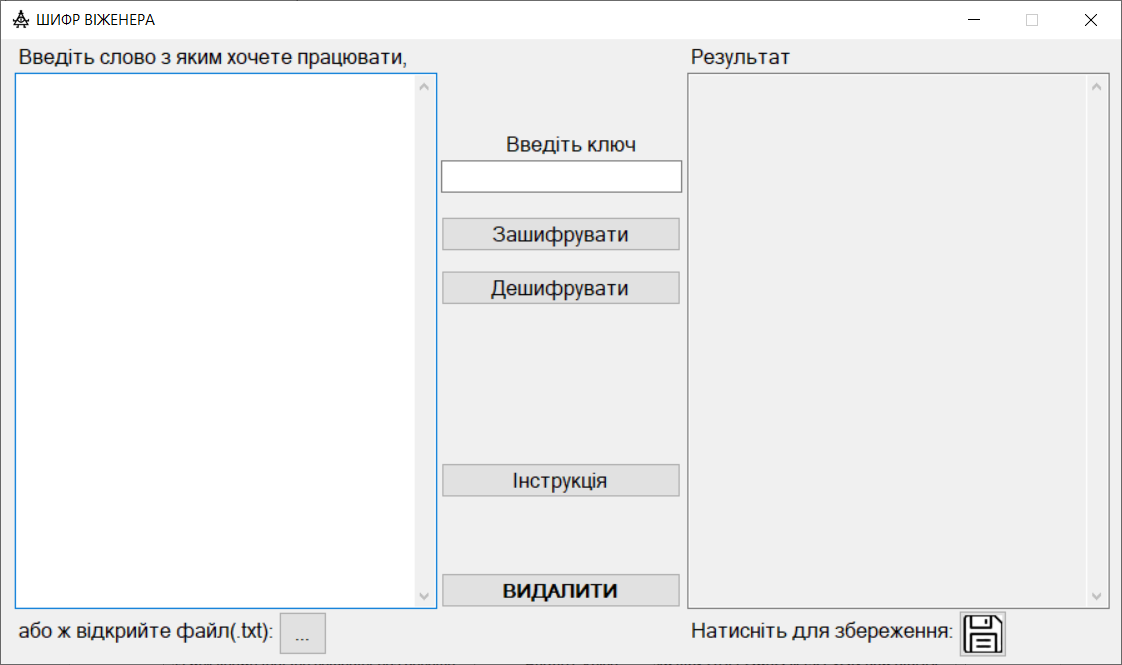


Рисунок 3.1. Інтерфейс програми

Незважаючи на те, що інтерфейс виглядає дуже примітивним, він є дуже звучним у використанні, що є великим плюсом для звичайного користувача, а сама програма чудово виконує поставлене їй завдання. Для того щоб це перевірити було проведено ряд тестів.

1. **Тестування додатку**

Таблиця 3.1.

|  |  |  |  |
| --- | --- | --- | --- |
| № | Поставлена задача | Очікувані результати | Отримані результати |
| 1 | Реакція програми на нульову довжину вхідних даних | Відкривається вікно, яке повідомляє про відсутність даних | Отримали відповідне повідомлення (Рисунок 3.2.) |
| 2 | Реакція програми на нульову довжину ключового слова | Відкривається вікно, яке повідомляє про відсутність ключа | Отримали відповідне повідомлення (Рисунок 3.3.) |
| 3 | Реакція програми на введення пробілу в ключове слово | Відкривається вікно, яке повідомляє про введення пробілу | Отримали відповідне повідомлення (Рисунок 3.4.) |
| 4 | Реакція програми на введення символів, що не містяться в латинському алфавіті | Програма повинна лишити ці символи без зміни | Отримали відповідний результат (Рисунок 3.5. - Рисунок 3.6.) |
| 5 | Перевірка правильності шифрування за допомогою таблички Віженера | Результат роботи програми повинен відповідати табличному методу | Отриманий результат виконаний програмою відповідає результату виконаному табличним методом (Рисунок 3.7. - Рисунок 3.8.) |

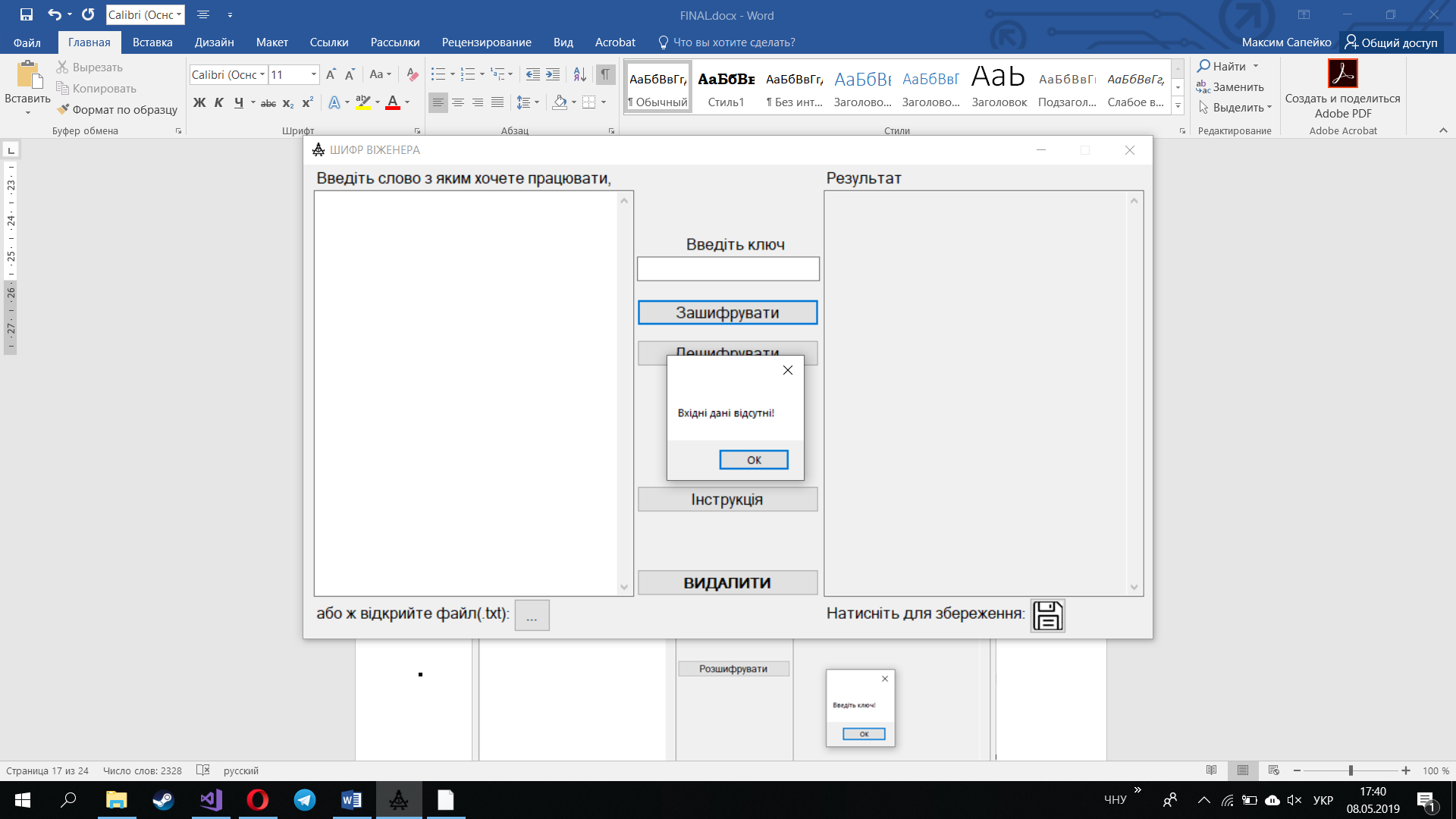


Рисунок 3.2. Реакція програми на відсутні вхідні дані

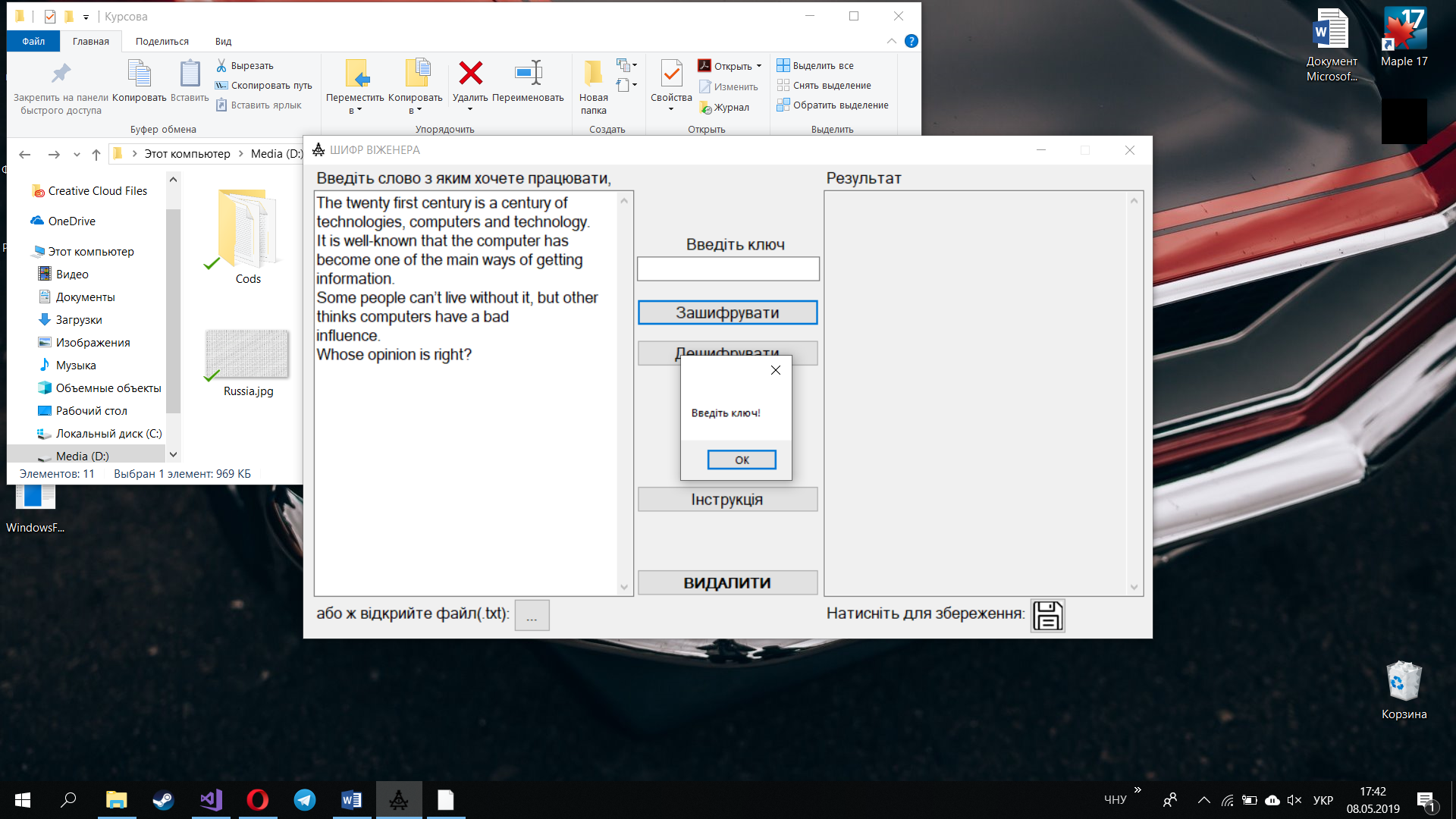


Рисунок 3.3. Реакція програми на порожній ключ

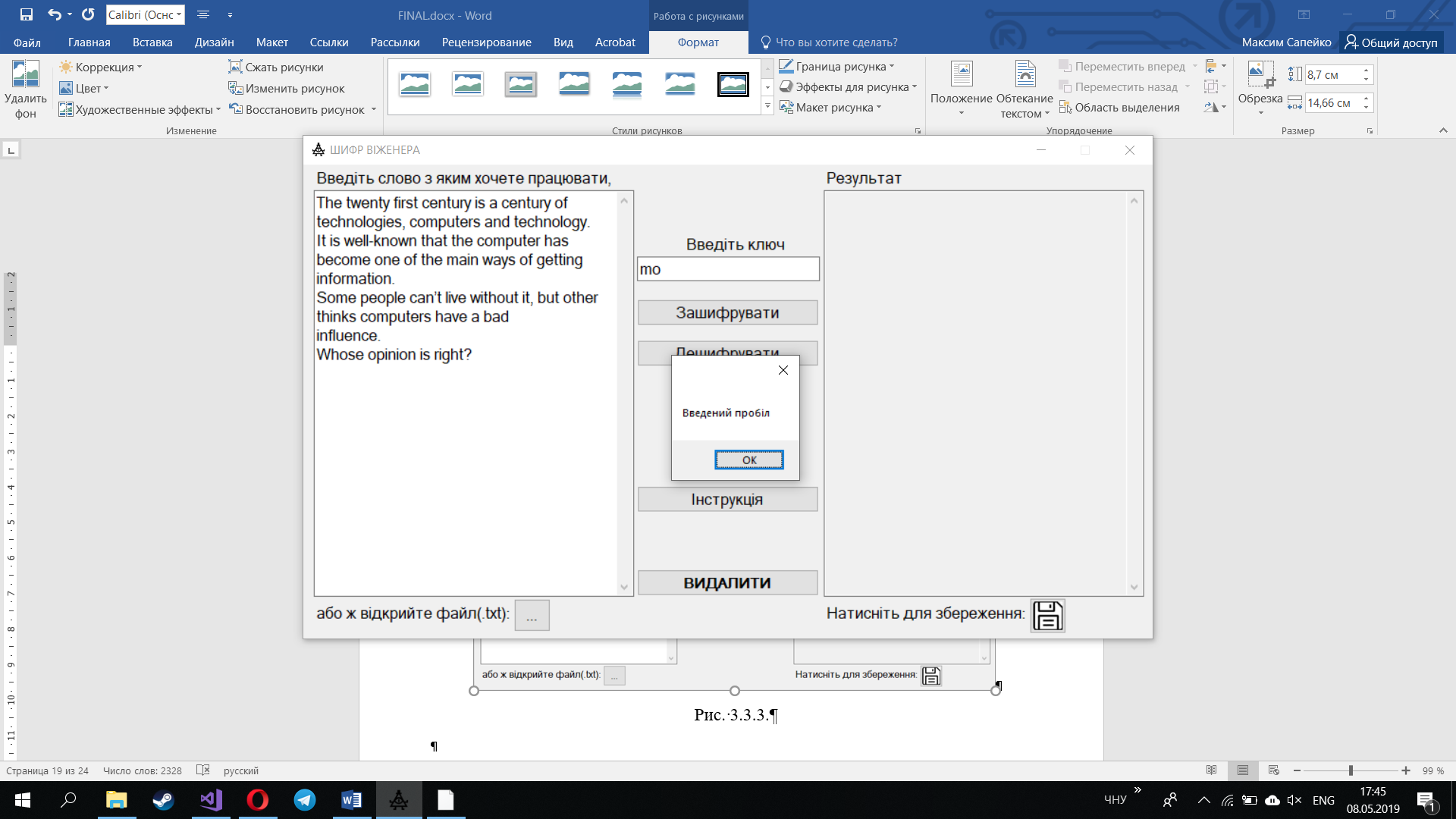


Рисунок 3.4. Реакція програми на введення пробілу у ключ

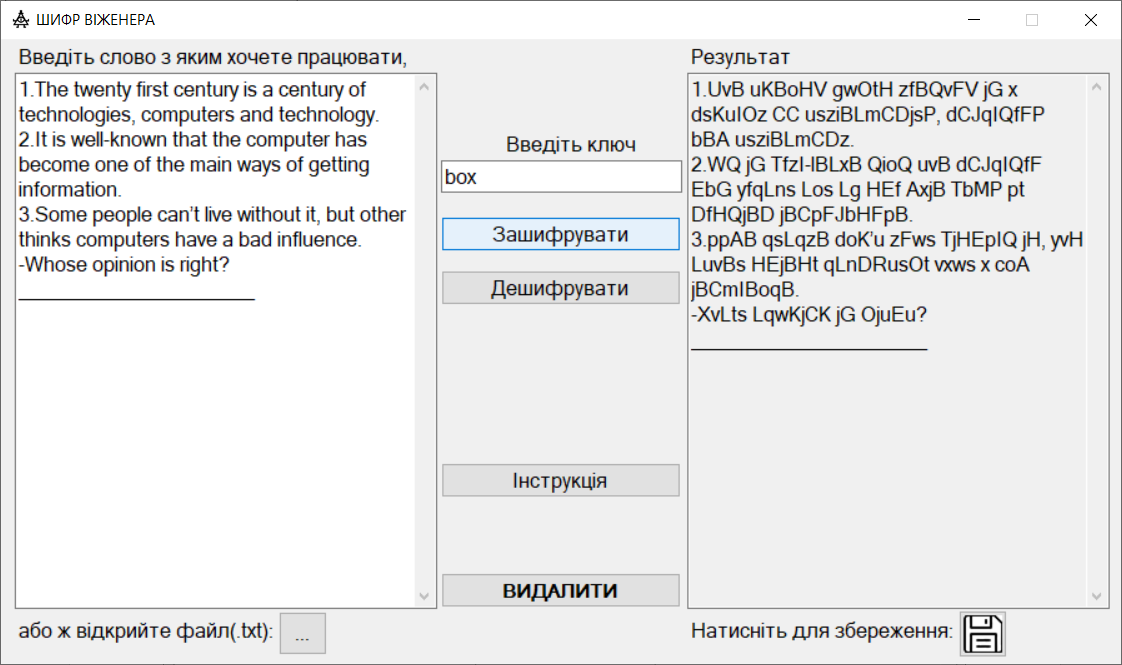


Рисунок 3.5. Результат шифрування, з введеними символами, що не містяться в алфавіті

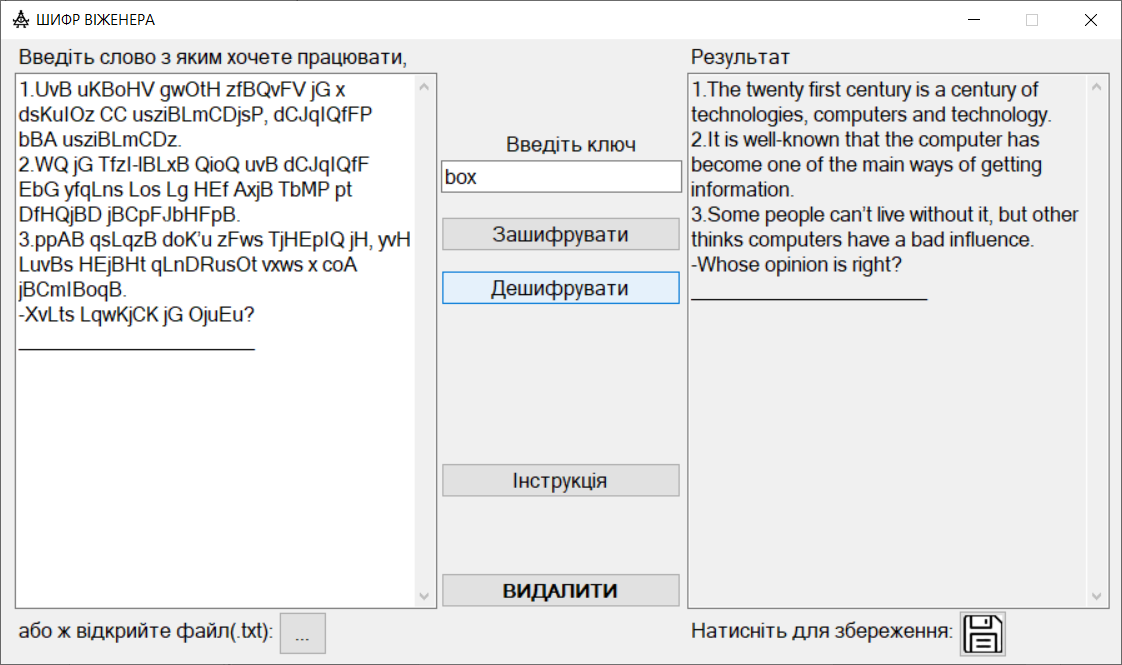


Рисунок 3.6. Результат дешифрування, з введеними символами, що не містяться в алфавіті

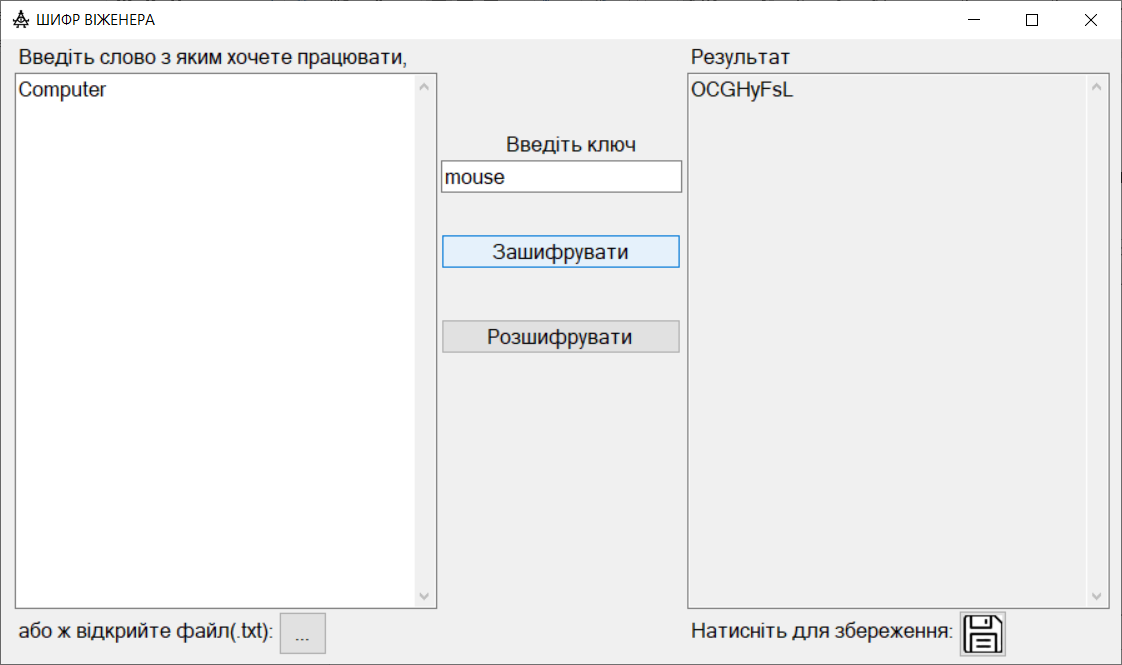


Рисунок 3.7. Результат шифрування

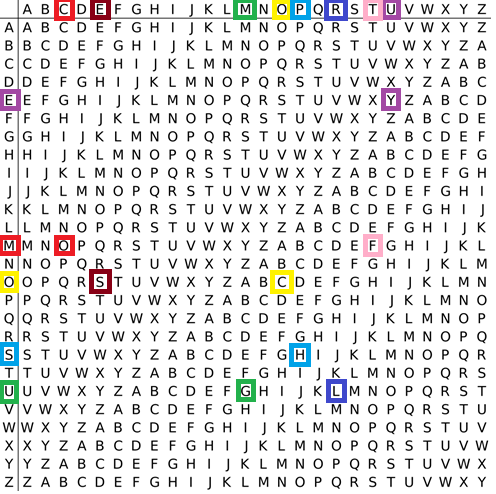


Рисунок 3.8. Перевірка роботи програми за допомогою таблички Віженера (Перший рядок – символи повідомлення, перший стовпчик – символи ключа)

1. **Висновки до третього розділу**

В даному розділі було описано реалізацію інтерфейсу програми. Всі тестування програми були пройдені успішно, а також був реалізований весь функціонал програми.

Було проведено тестування додатку, за допомогою ряду тестів. Перші три тести пов’язані з неправильним формуванням вхідних даних. За допомогою першого тесту виявляється реакція програми на нульову довжину вхідного повідомлення. Другий тест показує результат роботи програми в разі нульової довжини ключа. Як вже зазначалося що ключ не повинен містити пробілів, тому в третьому тесті перевіряємо реакцію програми на введення пробілу в ключ. Четвертий тест дає змогу перевірити роботу додатку в разі введення символів які не належать, в нашому випадку, латинському алфавіту. В п’ятому тесті перевіряємо роботу програми табличним методом.

Як бачимо всі тести програма пройшла успішно.

**Висновок**

В курсовій роботі було реалізовано алгоритм шифрування методом Віженера, мовою програмування C#, за допомогою Windows Forms. Також була створена блок-схема алгоритму шифрування. В ході розробки програми були вивчені особливості шифрування за допомогою даного методу, а саме алгоритм шифрування Віженера, а також проведено ознайомлення з іншими методами шифрування, та відмінностями між ними. Сама програма є дуже зручною у використанні. Як окремий продукт, дану програму буде важко розвивати, тому що існує дуже багато надійніших аналогів. Але цей алгоритм є дуже зручним для тренування практичних навичок в шифруванні.

Як і будь-який інший програмний продукт, отриманий додаток може бути покращеним за рахунок доопрацювання. Головною перспективою розробки програмного продукту може бути додавання можливості вибору мови шифрування.

Написання курсової роботи дало змогу удосконалити практичні навички в розробці програмного забезпечення, на мові програмування C#.

**Список використаних джерел**

1. Гребенович Ю.Є., Онищенко Б.О.,Супруненко О.О. методичні вказівки до виконання та оформлення курсової роботи з дисциплін “Основи програмування” , “Програмування та алгоритмічні мови”, “Алгоритмізація та програмування” для студентів, які навчаються за напрямами підготовки 050101 – „Комп’ютерні науки”, 050103 – „Програмна інженерія”, 040303 – „Системний аналіз” усіх форм навчання Черкаси: Вид. ЧНУ імені Богдана Хмельницького, 2016. – 32с.
2. Шифр Віженера. [Електронний ресурс]. Режим доступу: <https://dehtyarov09.wordpress.com/2014/03/24/поліалфавітні-шифри/> . Перевірено: 07.05.2019.
3. Шифр Віженера. [Електронний ресурс]. Режим доступу: <https://ru.wikipedia.org/wiki/Шифр_Виженера> . Перевірено: 07.05.2019.
4. Руководство по С#. [Електронний ресурс]. Режим доступу: <https://metanit.com/sharp/> . Перевірено: 07.05.2019.
5. Windows Forms. [Електронний ресурс]. Режим доступу: <https://uk.wikipedia.org/wiki/Windows_Forms> . Перевірено: 07.05.2019.
6. Шифр Віженера. [Електронний ресурс]. Режим доступу: <https://sites.google.com/site/simplecryptography/pc/simplevige> . Перевірено: 07.05.2019.

**Додаток А**

початок

int j, f, i;

int t = 0;

char[] massage = m.ToCharArray();

char[] key = k.ToCharArray();

i = 0; i < massage.Length; i++

j = 0; j < alfavit.Length; j++

massage[i] = = alfavit[j]

break;

j != 52

t = 0;

t > key.Length - 1

Приймаємо параметри alfavit, m, k, btnEnFlag, btnDeFlag

**+**

**-**

**+**

**+**

**+**

**-**

**-**

**+**

**-**

**-**

кінець

f = 0; f < alfavit.Length; f++

key[t] == alfavit[f]

break;

key[t] == ' '

t = 0;

t++;

f != 52

int offset = (j + f) % 52;

massage[i] = alfavit[offset];

string result = new string(massage);

**+**

**-**

**-**

**+**

**+**

**-**

**-**

**+**

btnEnFlag == true

btnDeFlag == true

int nomer = (j - f + 52) % 52;

massage[i] = alfavit[nomer];

**+**

**+**

**-**

**-**

Повертаємо рядок result

**Додаток Б**

public static string Cipher(char[] alfavit, string m, string k, bool btnEnFlag, bool btnDeFlag)

{

int letter\_index, key\_index, massage\_index; // Змінні для циклів

int t = 0; // Змінна для нумерації символів ключа

char[] massage = m.ToCharArray(); // Перетворюємо рядок в масив символів.

char[] key = k.ToCharArray(); // Перетворюємо ключ в масив символів.

// Цикл яким проходимо по кожному символу

for (massage\_index = 0; massage\_index < massage.Length; massage\_index++)

{

// Шукаємо індекс букви

for (letter\_index = 0; letter\_index < alfavit.Length; letter\_index++)

{

if (massage[massage\_index] == alfavit[letter\_index])

{

break;

}

}

//Якщо j не дорівнює довжині алфавіту, значить символ з алфавіту

if (letter\_index != alfavit.Length)

{

// Задаємо умову щоб ключ повторювався при заповненні

if (t > key.Length - 1)

{

t = 0;

}

// Шукаємо індекс букви ключа

for (key\_index = 0; key\_index < alfavit.Length; key\_index++)

{

if (key[t] == alfavit[key\_index])

{

break;

}

if (key[t] == ' ')

{

t = 0;

}

}

t++;

// Якщо f не дорівнює довжині алфавіту, значить символ ключа із алфавіту

if (key\_index != alfavit.Length)

{

if (btnEnFlag == true)

{

// Шукаємо індекс дешифрованої букви

int offset = (letter\_index + key\_index) % alfavit.Length;

massage[massage\_index] = alfavit[offset]; // Змінюємо букву

}

else if (btnDeFlag == true)

{

// Шукаємо індекс зашифрованої букви

int nomer = (letter\_index - key\_index + alfavit.Length) % alfavit.Length;

// Змінюємо букву

massage[massage\_index] = alfavit[nomer];

}

}

}

}

t = 0; // Скидаємо індекс ключа

string result = new string(massage); // Складаємо символи назад в рядок.

return result;

}