Зміст

[ВСТУП 3](#_Toc8678349)

[РОЗДІЛ 1. ОГЛЯД АЛГОРИМТІВ ТА МЕТОДІВ РЕАЛІЗАЦІЇ ПОСТАНОВЛЕНОГО ЗАВДАННЯ 6](#_Toc8678350)

[1.1 Алгоритм шифру Цезаря з ключем 6](#_Toc8678351)

[1.2 Алгоритм шифру Цезаря з ключовим словом 7](#_Toc8678352)

[1.3 Вибір мови програмування для реалізації програми 9](#_Toc8678353)

[1.3.1 Мова програмування C# 9](#_Toc8678354)

[1.3.2 Мова програмування Java 10](#_Toc8678355)

[1.3.3 Остаточний вибір мови програмування 10](#_Toc8678356)

[1.4 Висновок 11](#_Toc8678357)

[РОЗДІЛ 2. ПРОЕКТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ 12](#_Toc8678358)

[2.1 Структурна схема 12](#_Toc8678359)

[2.2 Алгоритм вибору файлу 12](#_Toc8678360)

[2.3 Алгоритм задання ключа 12](#_Toc8678361)

[2.4 Алгоритм шифру Цезаря. 13](#_Toc8678362)

[2.5 Алгоритм зберігання зашифрованого/розшифрованого файлу 14](#_Toc8678363)

[2.6 Алгоритм відкриття новоствореного файлу для перевірки 14](#_Toc8678364)

[2.7 Алгоритм закриття програми 14](#_Toc8678365)

[2.8 Висновок до другого розділу 14](#_Toc8678366)

[РОЗДІЛ 3. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ ДЛЯ ПОБУДОВИ ОПУКЛОЇ ОБОЛОНКИ ДВОВИМІРНОЇ ТОЧКОВОЇ МНОЖИНИ 16](#_Toc8678367)

[3.1 Реалізація основних функцій програмного продукту 16](#_Toc8678368)

[3.2 Реалізація користувацького інтерфейсу 18](#_Toc8678369)

[3.3 Тестування програмного продукту 22](#_Toc8678370)

[3.4 Висновок 25](#_Toc8678371)

[ВИСНОВКИ 26](#_Toc8678372)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 28](#_Toc8678373)

[Додаток А. Структурна схема 1А](#_Toc8678375)

[Додаток Б. Алгоритм шифрування Цезаря 1Б](#_Toc8678376)

[Додаток В. Алгоритм дешифрування Цезаря 1В](#_Toc8678377)

[Додаток Г. Відкриття файлу 1Г](#_Toc8678378)

[Додаток Д. Збереження зашифрованого файлу 1Д](#_Toc8678379)

[Додаток Е. Збереження розшифрованого файлу 1Е](#_Toc8678380)

**ВСТУП**

Протягом усієї своєї історії людина відчувала потребу в шифруванні тієї чи іншої інформації. Не дивно, що з цієї потреби виросла ціла наука - криптографія. І якщо раніше криптографія здебільшого служила виключно державним інтересам, то з приходом інтернету її методи стали надбанням приватних осіб і широко використовуються хакерами, борцями за свободу інформації та будь-якими особами, які бажають в тій чи іншій мірі зашифрувати свої дані в мережі[1].

Існує досить багато алгоритмів шифрування інформації. Найбільш відомим з давніх шифрів заміни є шифр Цезаря, названий так на честь римського імператора Гая Юлія Цезаря, який використовував його для секретного листування зі своїми генералами. Шифр Цезаря є одним з найбільш вивчених в криптографії, і він дуже корисний тим, що ілюструє принципи модульної арифметики, однієї з математичних основ кодованого листа. Принцип приховування інформації в даному шифрі досить не складний. Алгоритм шифру Цезаря полягає в тому, щоб зрушити алфавіт, а ключ — число букв, на яке зроблений зсув.

Щоб розшифрувати дане повідомлення, потрібно всього лише знати на скільки позицій необхідно змістити літери. Фахівці розходяться в думках з приводу практичного застосування шифру Цезаря, хтось, стверджує, що сам Гай Юлій Цезар заміняв букву тексту, зміщуючи на чотири позиції в алфавіті, а хтось каже, що він звільняв на 3 позиції щодо початкової літери. Також немає точної відповіді, в яку саме сторону необхідно було рухатися за алфавітом, вправо або вліво[4].

**Мета:** Основною метою даної курсової роботи є створення програми, яка буде шифрувати і дешифрувати текст у файлі записаний англійською мовою. При цьому будуть створюватися окремі файли з зашифрованим і розшифрованим текстом.

**Завдання цієї курсової роботи:**

1. Виконати аналіз можливих алгоритмів для шифрування тексту шифром Цезаря
2. Побудова структурної схеми програмного додатку та блок-схем основних та допоміжних алгоритмів.
3. Реалізувати шифр Цезаря у вигляді програмного продукту.
4. Зробити висновки щодо можливостей застосування розробленого програмного модуля для шифрування тексту.

**Актуальність теми:**

Проблема шифрування і дешифрування текстових повідомлень зараз особливо актуальна. Протягом багатьох століть шифрування застосовується для захисту інформації від попадання в коло осіб, яким вона не адресована. На сьогоднішній день існує величезна безліч зашифрованих документів, файлів та іншої інформації, що належать різним країнам світу, різних епох і написаних з різних причин.

**Базові поняття:**

Відкритий (вихідний) текст – дані (не обов'язково текстові), що передаються без використання криптографії.

Зашифрований (прихований) текст - дані, отримані після застосування криптосистеми з вказаним ключем.

Шифрування - процес нормального застосування криптографічного перетворення відкритого тексту на основі алгоритму і ключа, в результаті якого виникає шифрований текст.

Розшифрування - процес нормального застосування криптографічного перетворення шифрованого тексту у відкритий.

Дешифрування - процес вилучення відкритого тексту без знання криптографічного ключа на основі відомого шифрованого. Термін дешифрування зазвичай застосовують по відношенню до процесу криптоанализа шифротекста (криптоаналіз сам по собі, взагалі кажучи, може полягати і в аналізі шифросистемамі, а не тільки зашифрованого нею відкритого повідомлення).

Якщо зіставити кожному символу алфавіту його порядковий номер (нумеруючи з 0), то шифрування і дешифрування можна виразити формулами:

де *x* – порядковий номер символу відкритого тексту, *y* – порядковий номер символу шифрованого тексту, *n* – потужність алфавіту, а *k* – ключ.

Можна помітити, що суперпозиція двох шифрувань і є просто шифруванням на ключі . Більш загально, множина шифруючих перетворень шифру Цезаря утворює [групу](https://uk.wikipedia.org/wiki/%D0%93%D1%80%D1%83%D0%BF%D0%B0_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) [3]{\displaystyle ~\mathbb {Z} \_{n}}.

# РОЗДІЛ 1. ОГЛЯД АЛГОРИМТІВ ТА МЕТОДІВ РЕАЛІЗАЦІЇ ПОСТАНОВЛЕНОГО ЗАВДАННЯ

* 1. **Алгоритм шифру Цезаря з ключем**

Шифр Цезаря є окремим випадком шифру простої заміни (одноалфавітной підстановки). Свою назву цей шифр отримав по імені римського імператора Гая Юлія Цезаря, який використовував цей шифр при листуванні.

При шифруванні вихідного тексту кожна буква замінюється іншою літерою того ж алфавіту за таким правилом. Замінює буква визначається шляхом зміщення за алфавітом до кінця від вихідної букви на k букв. При досягненні кінця алфавіту виконується циклічний перехід до його початку.

Наприклад: нехай A - використовуваний алфавіт:

A = {a1, a2, ..., am, ..., aN},

де a1, a2, ..., am, ..., aN - символи алфавіту; N ширина алфавіту.

Нехай k - число позицій зсуву символів алфавіту при шифруванні, 0 < k <N. При шифруванні кожен символ алфавіту з номером m з кодованого тексту замінюється на символ цього ж алфавіту з номером m + k. Якщо m + k > N, номер символу в алфавіті A визначається як m + k - N.

Для дешифрування текстової інформації номер позиції символу відновлюваного тексту визначається як m-k. Якщо m-k < 0, то обчислення цього номера проводиться як m-k + N.

Переваги алгоритму:

* простота шифрування і дешифрування;
* легко реалізувати мовою програмування.

До недоліків системи Цезаря слід віднести:

* підстановки, що виконуються відповідно до системи Цезаря, не маскують частот появи різних букв вихідного і відритого тексту;
* зберігається алфавітний порядок в послідовності змінюючих букв; при зміні значення k змінюються тільки початкові позиції такої послідовності;
* число можливих ключів   досить мале і обмежене кількістю букв алфавіту;
* шифр Цезаря легко розкривається на основі аналізу частот появи літер в шифрі[6].

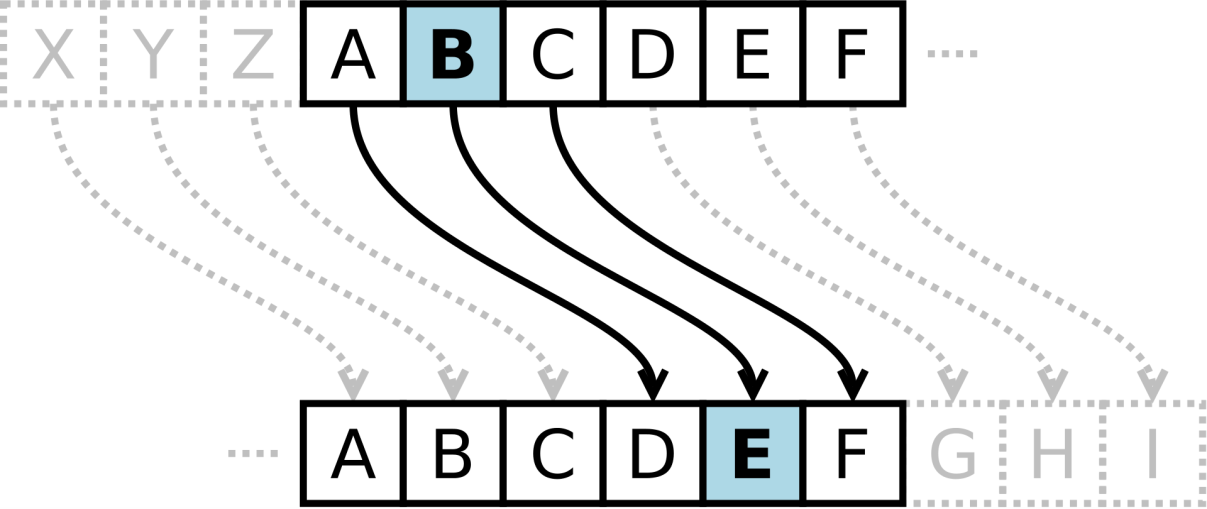


Рисунок 1.1. Приклад роботи алгоритму

шифрування Цезаря з ключем k рівним 3[3].

* 1. **Алгоритм шифру Цезаря з ключовим словом**

У даного різновиду шифру Цезаря ключ задається числом k (0 <= k <= n-1) і коротким ключовим словом або пропозицією. Виписується алфавіт, а під ним, починаючи з k-й позиції, ключове слово. Решта букви записуються в алфавітному порядку після ключового слова. У підсумку ми отримуємо підстановку для кожної літери. Вимога, щоб всі букви ключового слова були різними не обов'язково - можна записувати ключове слово без повторення однакових букв[7].

Переваги алгоритму:

* цей алгоритм на відміну від попереднього більш безпечний і його складніше розшифрувати;
* більша кількість ключів, що ускладнює розшифровку постороннім особам;
* змінюється алфавітний порядок в послідовності змінюючиз букв; при введенні ключогового слова змінюється алфавітний порядок букв залежно від введеного слова.

Недоліки алгоритму:

* більш складний алгоритм шифрування і дешифрування;
* можливість злому шифртекста на основі аналізу частот появи літер;
* більш-менш складний у реалізації мовою програмування.



Таблиця 1.1. Таблиця замін символів для системи шифрування Цезаря при K = 3, M = 32 і ключовому слові «ШИФРОВКА» (приклад роботи)[7].

* 1. **Вибір мови програмування для реалізації програми**
     1. **Мова програмування C#**

**C#** (вимовляється *Сі-шарп*) — об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET. Розроблена Андерсом Гейлсбергом, Скотом Вілтамутом та Пітером Гольде під егідою Microsoft Research (при фірмі Microsoft).

Синтаксис C# близький до С++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML. Перейнявши багато що від своїх попередників — мов С++, Delphi, Модула і Smalltalk — С#, спираючись на практику їхнього використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад множинне спадкування класів (на відміну від C++).

C# розроблялась як мова програмування прикладного рівня для [CLR](https://uk.wikipedia.org/wiki/CLR) і тому вона залежить, перш за все, від можливостей самої CLR. Це стосується, перш за все, системи типів C#. Присутність або відсутність тих або інших виразних особливостей мови диктується тим, чи може конкретна мовна особливість бути трансльована у відповідні конструкції CLR. Так, з розвитком CLR від версії 1.1 до 2.0 значно збагатився і сам C#; подібної взаємодії слід чекати і надалі. (Проте ця закономірність буде порушена з виходом C# 3.0, що є розширеннями мови, що не спираються на розширення платформи .NET.) CLR надає C#, як і всім іншим .NET-орієнтованим мовам, багато можливостей, яких позбавлені «класичні» мови програмування. Наприклад, збірка сміття не реалізована в самому C#, а проводиться CLR для програм, написаних на C# точно так, як і це робиться для програм на VB.NET, J# тощо[9].

* + 1. **Мова програмування Java**

**Java** (вимовляється *Джава*) — об'єктно-орієнтована мова програмування, випущена 1995 року компанією «Sun Microsystems» як основний компонент платформи Java. З 2009 року мовою займається компанія «Oracle», яка того року придбала «Sun Microsystems». В офіційній реалізації Java-програми компілюються у байт-код, який при виконанні інтерпретується віртуальною машиною для конкретної платформи.

«Oracle» надає компілятор Java та віртуальну машину Java, які задовольняють специфікації Java Community Process, під ліцензією GNU General Public License.

Мова значно запозичила синтаксис із C і C++. Зокрема, взято за основу об'єктну модель С++, проте її модифіковано. Усунуто можливість появи деяких конфліктних ситуацій, що могли виникнути через помилки програміста та полегшено сам процес розробки об'єктно-орієнтованих програм. Ряд дій, які в С/C++ повинні здійснювати програмісти, доручено віртуальній машині. Передусім Java розроблялась як платформо-незалежна мова, тому вона має менше низькорівневих можливостей для роботи з апаратним забезпеченням, що в порівнянні, наприклад, з C++ зменшує швидкість роботи програм. За необхідності таких дій Java дозволяє викликати підпрограми, написані іншими мовами програмування[10].

* + 1. **Остаточний вибір мови програмування**

Для реалізації даного програмного продукту було обрано мову програмування С#. З точки зору розробника мови Java і C # дуже схожі. Обидві мови є строго типізований, об'єктними. Обидва увібрали в себе багато чого з синтаксису C ++, але на відміну від C ++, простіше в освоєнні для початківців. Обидва запозичили з C набір основних ключових слів і службових символів, в тому числі фігурні дужки для виділення блоків. Обидві мови спираються на збірку сміття. Обидві мови супроводжуються багатими колекціями бібліотек. Головною причиною вибору стало те, що це об’єктно-орієнтована, компільована мова із статичною типізацією і програмувати інтерфейс користувача на ній досить легко. В MS Visual Studio С# присутні багато різних компонентів, які надають широкий вибір при проектуванні, що дозволяє найкращим чином налагодити програму під поставлену задачу[11].

## 1.4 Висновок

Серед алгоритмів, наведених в даному розділі, обрано найперший алгоритм шифрування Цезаря, оскільки цей алгоритм має ряд переваг над іншими алгоритмами: простий для розуміння, легкий для реалізації мовою програмування.

# РОЗДІЛ 2. ПРОЕКТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ

* 1. **Структурна схема**

Структурна схема наведена в додатку «А». Програма буде поділена на модулі, кожний з яких відповідає за свою частину і викликається якоюсь подією. Спочатку користувач повинен обрати файл який потрібно буде зашифрувати/розшифрувати. Потім потрібно задати ключ за яким буде проводитися шифрування або дешифрування тексту у файлі. Далі потрібно вибрати одну з потрібних нам опцій “зашифрувати/розшифрувати”, натиснувши одну із цих кнопок вилізе меню для зберігання файла у потрібному вам місці. А після цих всіх дій по бажанню можна натиснути кнопки для перегляду вмісту зашифрованого/розшифрованого файлу.

* 1. **Алгоритм вибору файлу**

Є кнопка «Виберіть файл». Алгоритм вибору файлу відбувається досить просто. Буде створено об’єкт openFileDialog1 за допомогою якого ми напишемо власну логіку для відкриття файлу. Для цього нам потрібно буде використати таку властивість openFileDialog1, як **Filter** (задає фільтр файлів, завдяки чому в діалоговому вікні можна відфільтрувати файли з розширення. Фільтр задається в наступному форматі Назва\_файлів | \* .розширення. Наприклад, Текстові файли (\*. txt) | \* .txt. Можна задати відразу кілька фільтрів, для цього вони поділяються вертикальною лінією |. Наприклад, Bitmap files (\* .bmp) | \* .bmp | Image files (\* .jpg) | \* .jpg) і **FileName** (повертає повне ім’я файлу, обраного в діалоговому вікні), а також метод **ShowDialog()** (для відображення діалогового вікна).

* 1. **Алгоритм задання ключа**

Цей алгоритм ще простіший за попередній. Для задання ключа нам потрібно буде створити об’єкт numericUpDown1. Потім властивостями цього об’єкту ми задамо максимальне, мінімальне і початкове значення ключа **Maximum, Minimum** і **Value** відповідно і передаватимемо значення у шифр Цезаря**.**

* 1. **Алгоритм шифру Цезаря.**

Блок-схема алгоритму Цезаря наведена в додатку «Б».

Алгоритми шифрування і дешифрування описані у програмі окремими функціями, в які передаватимуться ключ шифрування, а також назви файлів, що будуть шифруватися/дешифруватися, зберігатися після відповідних дій за допомогою модифікатора доступу **public**, який означає, що ці дані доступні з любого місця в коді.

На початку роботи алгоритму шифрування/дешифрування створюються три змінні, дві з яких рядкового типу(skey і text) і одна цілого(key). Спочатку один із рядкових типів(skey) отримує значення ключа шифрування, після чого цей рядок перетворимо до числового типу(key). Останній рядковий тип(text) буде приймати порожнє слово, щоб для подальшої роботи з ним у нього можна було записати різну інформацію. Потім після цих дій ми відкриватимемо потік для читання даних з файлу і зчитуємо усі данні, що записані у файл, які будемо записувати в рядковий тип text. Виконавши ці дії відкриваємо інший потік в якому ми будемо записувати уже в інший файл зашифрований/розшифрований текст. Для цього ми створюємо масив символів з модифікатором доступу public, після його створення заповнимо його посимвольно текстом, що записаний у рядковий тип(text). Після того як ми перетворили наш текст з рядкового типу у масив символів, ми створимо цикл який буде проходити по всіх символах для того, щоб ми могли виконати нашу заміну букв відповідно шифру. Для цього в тілі циклу ми використовуватимемо оператор множинного розгалудження. За допомогою буде проводитися перевірка чи символ в тексті відповідає великим і малим буквам, якщо так то будемо брати значення цього символу в таблиці ASCII і зміщувати його на значення в право або в ліво, залежно від того чи це шифрування чи дешифрування і записуватимемо результат посимвольно в новий файл, а якщо ні, то його так і записуватимемо. Після того як пройдемо по всьому тексту і запишемо всі дані в новий файл, нам потрібно буде закрити потік.

* 1. **Алгоритм зберігання зашифрованого/розшифрованого файлу**

Алгоритм досить простий. Після натиснення кнопки «Зашифрувати» або «Розшифрувати», всередині функції («saveEncryptedFile», «saveDecryptedFile» відповідно), виконується перевірка чи був обраний файл для шифрування/дешифрування, якщо ні, то виводиться відповідне повідомлення про те, що потрібно обрати файл, а якщо файл все-таки був обраний, то виконується зберігання файлу за допомогою об’єкту saveFileDialog1, який відкриває діалогове вікно за допомогою метода ShowDialog(), і запам’ятовується шлях та ім’я завдяки властивості FileName. Потім після цього розгалудження викликається функція шифрування/дешифрування в яку передається і шлях та ім’я нового файлу.

* 1. **Алгоритм відкриття новоствореного файлу для перевірки**

Цей алгоритм полягає в тому, що є дві кнопки «Зашифрований» і «Розшифрований», при натисненні на неї виконується конструкція виключення. Якщо все добре, то виконується відкриття файлу за допомогою класу Process і методу Start(), інакше буде відкриватися вікно попереджуючи про помилку.

* 1. **Алгоритм закриття програми**

Є кнопка «Вихід». При натисненні на неї повинна закриватися програма за допомогою класу Application і методу цього класу Exit().

* 1. **Висновок до другого розділу**

Готовий програмний продукт повинен містити всі функції, що реалізовують вищезгадані алгоритми. Тобто програма повинна при натисненні на кнопку «Виберіть Файл» відкривати діалогове вікно, для того, щоб можна було обрати потрібний нам файл для шифрування/дешифрування. Також уміти як шифрувати текст, так його і дешифрувати, одночасно з цим давати можливість обирати самому куди зберігати нові файли при натисненні на відповідні кнопки «Зашифрувати», «Розшифрувати». Ще програма повинна при натисненні на кнопки «Зашифрований», «Розшифрований» відкривати відповідні збережені файли для перевірки вмісту файлу. Насамкінець, програма повинна при натисненні на кнопку «Вихід» завершати свій робочий процес і закриватися.

# РОЗДІЛ 3. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ ДЛЯ ПОБУДОВИ ОПУКЛОЇ ОБОЛОНКИ ДВОВИМІРНОЇ ТОЧКОВОЇ МНОЖИНИ

* 1. **Реалізація основних функцій програмного продукту**

Основна функція програмного продукту – це шифрування та дешифрування методом шифру Цезаря. Отже її і реалізовуємо

Лістинг 3.1 – Алгоритм шифрування

private void encryptionCaesar() //функція шифрування тексту

{

string skey = numericUpDown1.Value.ToString(); //отримуємо значення ключа шифрування

int key = Convert.ToInt32(skey); //перетворюємо отримане значення в числовий тип

string text = "";

StreamReader sr = new StreamReader(fileName, Encoding.Default); //потік для читання тфайлу

while (!sr.EndOfStream) //читається весь текст до кінця

{

text += sr.ReadLine();

}

sr.Close(); //закриваємо потік читання

StreamWriter sw = new StreamWriter(saveEncrypted); //потік, для запису у зашифрованого тексту в новий файл

textArray = text.ToCharArray(); //перетворюємо наш текст у масив символів

for (int i = 0; i < textArray.Length; i++) //проходимя по усим символам

{

if (textArray[i] >= 'a' && textArray[i] <= 'z') //перевіряємо чи символ належить множині a-z

{

char encryptedText = (char)((textArray[i] - 'a' + key) % 26 + 'a'); //зміщуємо символ на наш ключ

sw.Write(encryptedText); //записуємо посимвольно наший замінений символ у файл

}

else if (textArray[i] >= 'A' && textArray[i] <= 'Z') //перевіряємо чи символ належить множині A-Z

{

char encryptedText = (char)((textArray[i] - 'A' + key) % 26 + 'A'); //также само

sw.Write(encryptedText);

}

else

{

sw.Write(textArray[i]); //якщо наш символ не буква, то записуємо його як є

}

}

sw.Close(); // закриваємо наш потік

}

Лістинг 3.2 – Алгоритм дешифрування

private void decryptionOfCaesar() //функція дешифрування тексту, такий же алгоритм як і при шифруванні

{

string skey = numericUpDown1.Value.ToString();

int key = Convert.ToInt32(skey);

string text = "";

StreamReader sr = new StreamReader(fileName, Encoding.Default);

while (!sr.EndOfStream)

{

text += sr.ReadLine();

}

sr.Close();

StreamWriter sw = new StreamWriter(saveDecoded);

textArray = text.ToCharArray();

for (int i = 0; i < textArray.Length; i++)

{

if (textArray[i] >= 'a' && textArray[i] <= 'z')

{

char decodedText = (char)((textArray[i] - 'z' - key) % 26 + 'z');

sw.Write(decodedText);

}

else if (textArray[i] >= 'A' && textArray[i] <= 'Z')

{

char decodedText = (char)((textArray[i] - 'Z' - key) % 26 + 'Z');

sw.Write(decodedText);

}

else

{

sw.Write(textArray[i]);

}

}

sw.Close();

}

Після того, як ми дізналися шлях та ім’я файлу з яким будемо працювати, починаємо роботу з цим файлом за одним із алгоритмів(шифрування або дешифрування). Алгоритм починається з того, що ми отримуємо значення ключа шифрування у рядковому типі, тому далі ми його перетворюємо у цілий тип даних. Створюється рядковий тип з пустим словом, для подальшого заповнення його даними з файлу. Після цього починається робота потоку для читання файлу, який виконує читання усього файлу і записуються ці дані у рядковий тип який ми раніше створювали. Після читання файлу відкривається потік для запису даних у файл. Для того, щоб ми могли змінювати букви ми перетворюємо наш текст у масив символів. Потім циклом проходимо по всіх елементах масиву, і якщо це маленька або велика буква, то проводимо здвиг на заданий ключ, якщо ні, то залишаємо цей символ таким як і є. І після того як ми зашифрували або розшифрували файл закриваємо наш потік.

* 1. **Реалізація користувацького інтерфейсу**

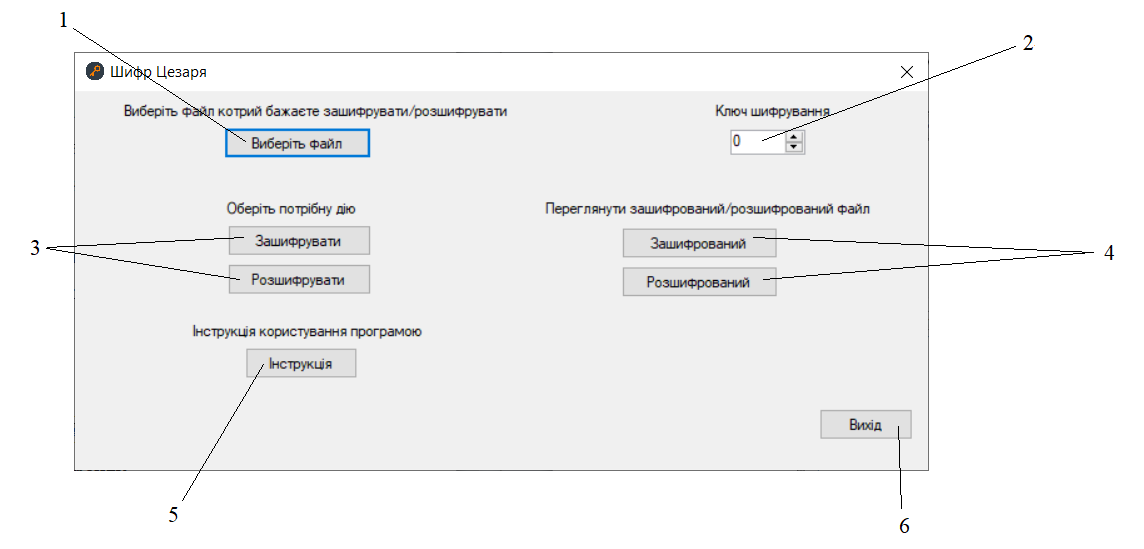


Рисунок 3.1. Вікно програми.

На рисунку зображено вікно, що з’являється на екрані, після запуску програми.

Те що позначено номером 1 – це кнопка, яка відповідає за вибір файлу з яким ми будемо працювати. Номер 2 – вибір числа з певного діапазону. Номер 3 – це кнопки, при яких відбувається шифрування/дешифрування і збереження відповідних файлів. Номер 4 – це кнопки, що відповідають за відкриття зашифрованого/розшифрованого файлу. Номер 5 – це кнопка, що відкриває інструкцію до програми в окремому вікні, тому що, можливо, не кожен користувач відразу зрозуміє, як користуватися нею. Номер 6 – це кнопка, що припиняє роботу програми.

Лістинг 3.3 – Кнопка «Виберіть файл»

private void openFile(object sender, EventArgs e) //кнопка, при натисненні на яку повинно з'явитися діалогове вікно для вибору файлу

{

openFileDialog1.Filter = "(\*.txt) | \*.txt"; //задаємо обмеження, щоб вибиралися файли тільки з таким розширенням

if (openFileDialog1.ShowDialog() == DialogResult.OK) //якщо, після закриття вікна, результат повернення дорівнює OK, то ім’я вибраного файлу зберігається у властивості FileName.

{

fileName = openFileDialog1.FileName; //запам'ятовуємо шлях та ім'я файлу в змінній fileName

}

}

Для того, щоб відкрити файл треба добавити об’єкт openFileDialog1 на форму. Так як ми працюємо з текстом, то ставимо обмеження на те, щоб обиралися тільки файли текстових форматів. Після вибору потрібного нам файлу, ми будемо зберігати його шлях та ім’я у змінній fileName, так як у подальшій роботі нам ці дані будуть потрібні.

Лістинг 3.4 – Кнопка «Зашифрувати»

private void saveEncryptedFile(object sender, EventArgs e) //функція для збереження зашифрованого файлу

{

if (fileName == null) //перевіряємо чи користувач обрав файл для шифрування

{

MessageBox.Show("Оберіть файл котрий хочете зашифрувати.");

}

else

{

saveFileDialog1.Filter = "(\*.txt) | \*.txt"; //задаємо обмеження, щоб зберігалися файли тільки з текстовим розширенням

if (saveFileDialog1.ShowDialog() == DialogResult.OK) //якщо, після закриття вікна, результат повернення дорівнює OK, то ім’я вибраного файлу зберігається у властивості FileName.

{

saveEncrypted = saveFileDialog1.FileName; //запам'ятовуємо шлях та ім'я файлу в змінній saveEncrypted

MessageBox.Show("Файл зашифровано"); //повідомляємо про те, що операція пройшла успішно

}

encryptionCaesar(); //функція, що виконує шифрування тексту

}

}

При натисканні на кнопку «Зашифрувати» відбувається перевірка чи користувач вибрав файл, якщо ні, то попереджаємо його про це. Далі якщо був вибраний файл, то відкривається діалогове вікно, яке дозволяє зберегти файл у потрібному користувачі місці. Зберігається файл у текстовому форматі і запам’ятовується ім’я та адреса. Після цього викликається функція «encryptionCaesar» в якій використовуються змінні fileName і saveEncrypted.

Лістинг 3.5 – Кнопка «Розшифрувати»

private void saveDecryptedFile(object sender, EventArgs e) //функція для збереження розшифрованого файлу, поясненя такі ж

{

if (fileName == null)

{

MessageBox.Show("Оберіть файл котрий хочете розшифрувати.");

}

else

{

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

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

{

saveDecoded = saveFileDialog1.FileName;

MessageBox.Show("Файл розшифровано");

}

decryptionOfCaesar(); //функція, що виконує дешифровку тексту

}

}

При натисканні на цю кнопку відбуваються такі самі дії, але викликається функція «decryptionOfCaesar» в якій використовуються змінні fileName і saveDecoded.

Лістинг 3.6 – Кнопка «Зашифрований»

private void openAnEncryptedFile(object sender, EventArgs e) //перегляд зашифрованого тексту

{

try //пробуємо відкрити файл

{

System.Diagnostics.Process.Start(saveEncrypted); //відкриваємо наш файл

}

catch(Exception exp) //якщо файлу неіснує, виводимо повідомлення про це

{

MessageBox.Show(exp.Message);

}

}

При натисканні на кнопку «Зашифрований», відбувається відкриття зашифрованого файлу, якщо такого файлу неіснує, то програма про це повідомляє.

Лістинг 3.7 – Кнопка «Розшифрований»

private void openDecryptedFile(object sender, EventArgs e) //перегляд розшифрованого тексту, такий же алгоритм

{

try

{

System.Diagnostics.Process.Start(saveDecoded);

}

catch (Exception exp)

{

MessageBox.Show(exp.Message);

}

}

При натисканні на кнопку «Розшифрований», відбувається відкриття зашифрованого файлу, якщо такого файлу неіснує, то програма про це повідомляє.

Лістинг 3.8 – Кнопка «Інструкція»

private void Instruction(object sender, EventArgs e) //кнопка інструкції до програми

{

MessageBox.Show("Програма для шифрування і дешифрування тексту англійської мови шифром Цезаря.\nПринцип дії полягає в тому, щоб циклічно зсунути алфавіт, а ключ — це кількість літер, на які робиться зсув.\n" +

" 1.Спочатку ви повинні обрати файл котрий бажаєте зашифрувати або дешифрувати.\n" +

" 2.Потім задайте значення ключа шифрування.\n" +

" 3.Оберіть потрібну вам операцію.\n" +

" 4.По бажанню можете переглянути вміст файлу який був зашифрований або розшифрований.\n" +

"Вдалого вам користування!!!", "Інструкція");

}

При натисканні на кнопку «Інструкція» відкривається додаткове вікно в яуому наведено інформацію про те, як користуватися цією програмою.

Лістинг 3.9 – Кнопка «Вихід»

private void closeTheProgram(object sender, EventArgs e) //закриття програми

{

Application.Exit(); //виконує власне закриття нашої програми

}

Після того, як користувач натисне на кнопку «Вихід», програма закриється.

* 1. **Тестування програмного продукту**

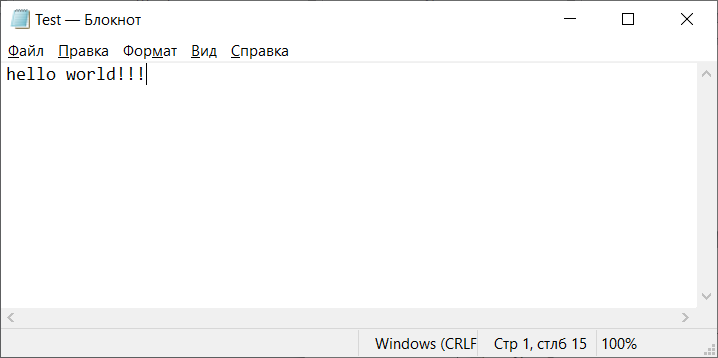


Рисунок 3.2. Початковий вміст файлу, котрий ми будемо зашифровувати.

Запустивши програму, на екрані з’являється вікно як на рисунку 4.

Після цього можна починати роботу. Натиснувши кнопку відкрити файл відкривається ось таке діалогове вікно.

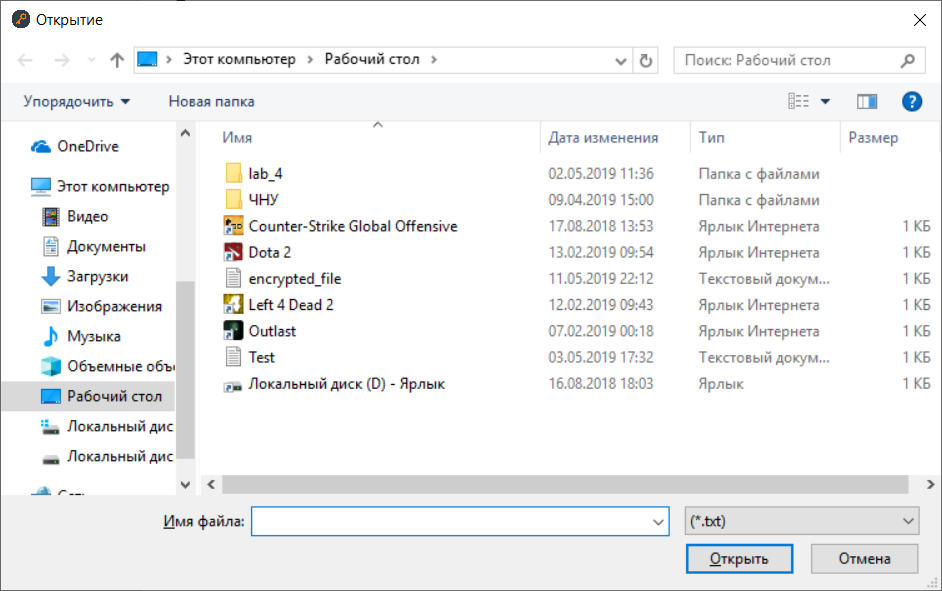


Рисунок 3.3. Вибір файлу.

Після того як обрали потрібний нам файл, задаємо значення ключа за яким будемо зашифровувати файл. Обравши ключ з яким будемо зашифровувати файл потрібно натиснути одну з кнопку «Зашифрувати», щоб виконати відповідну дію. Знову відкриється діалогове вікно, але на цей раз ми повинні задати ім’я файлу і натиснути кнопку «Сохранить».

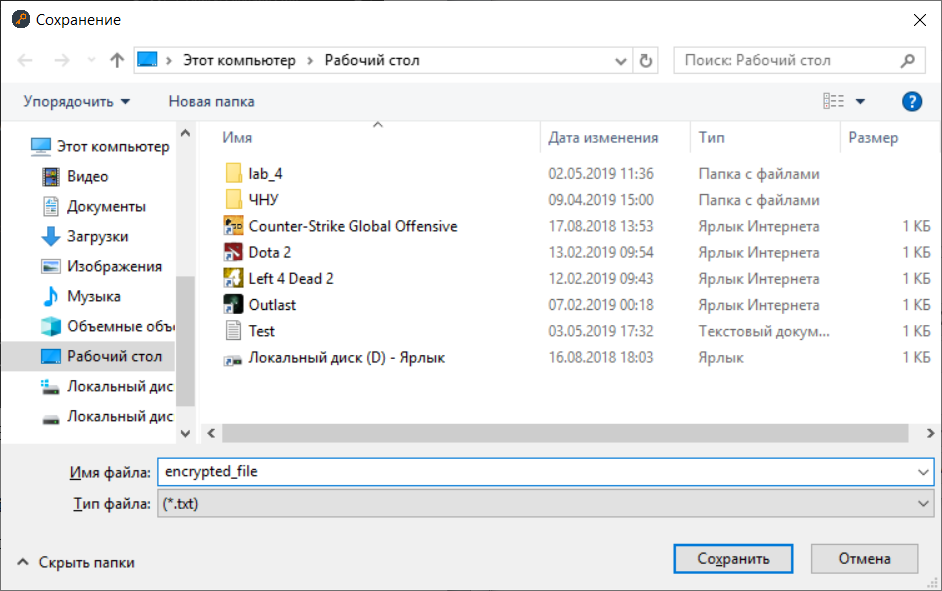


Рисунок 3.4. Збереження файлу

Після того як натиснули кнопку «Сохранить» програма повідомляє про те, що операція здійснена успішно.

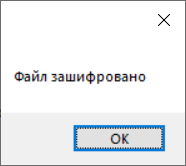


Рисунок 3.5. Повідомляюче вікно про виконану операцію.

Тепер можемо переглянути вміст зашифрованого файлу натиснувши кнопку «Зашифрований».

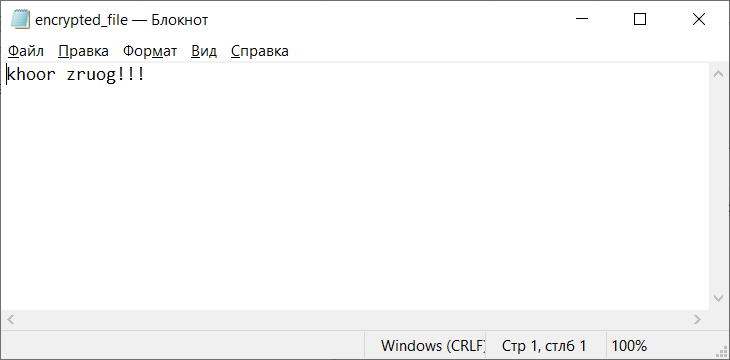


Рисунок 3.6. Вміст зашифрованого файлу

* 1. **Висновок**

Для програмного виконання основних поставлених задач необхідно використати всі алгоритми описані в першому розділі. Створена програма дає користувачеві можливість легко працювати з шифруванням або дешифруванням файлу, а також відкривати новостворений файл для перевірки. Також створений дружній інтерфейс до даної програми, оскільки є кнопка «Інструкція» в котрій описується як працювати з цією програмою.

Також варто додати, що була зроблена спроба обробки помилок. Наприклад, якщо користувач натиснув на кнопку «Розшифрований», а файл ми ще не зашифровували, то на екрані з’явиться повідомлення, яке інформує користувача, що спочатку потрібно зашифрувати потрібний користувачеві файл, а вже потім відкривати розшифрований файл, або користувач вирішує обрати файл не текстового розширення, то також буде з’являтися повідомлення, яке інформує про те, що обраний файл не відповідного розширення.

# ВИСНОВКИ

Основні функціональні можливості цієї програми:

1. Дозволяє користувачеві обирати користувачеві потрібний файл для шифрування/дешифрування формату txt.
2. Дозволяє обрати користувачеві ключ в межах від 1 до 1000.
3. Шифрує і дешифрує текст у файлі записаний англійською мовою, як користувач натиснув кнопку «Зашифрувати» або «Розшифрувати» відповідно.
4. Відкриває зашифрований або розшифрований файл, після натиснення кнопки «Зашифрований» і «Розшифрований» відповідно.
5. Оброблює помилкові ситуації і сповіщає користувача про це.
6. Має кнопку «Інструкція» для ознайомлення користувача в роботі з програмою.

**Короткий опис програмних рішень:**

Програма поділена на функції і вбудовані класи, кожна з яких відповідає за всю частину роботи. Кожна функція викликається якоюсь подією. Наприклад, при натисненні на кнопку «Зашифрувати» викликається функція «encryptionCaesar()», в якій відбувається шифрування файлу. При натисненні «Вихід» викликається клас «Application», який за допомогою методу «Exit()» закриває програму.

**Практичне застосування:**

В даний час шифр Цезаря, як і всі інші шифри простої заміни, легко дешифрується і не має практичного застосування (крім використання в інтернет-форумах як приховування конфіденційної інформації від випадкового прочитання, наприклад відповіді на загадки, спойлери, образи).

**Перспективи розвитку:**

При модернізації програми можна буде виконувати більш складне шифрування. Але в цілому дана програма немає практично ніякого застосування на практиці.

**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

# Криптография: Базовые знания о науке шифрования. [Електронний ресурс]. Режим доступу: <http://www.furfur.me/furfur/culture/culture/166567-kriptografiya>. Перевірено: 05.05.2019.

1. Крптографія. [Електронний ресурс]. Режим доступу: <https://uk.wikipedia.org/wiki/Криптографія#Шифр_Цезаря>. Перевірено: 05.05.2019.
2. Шифр Цезаря. [Електронний ресурс]. Режим доступу: <https://uk.wikipedia.org/wiki/Шифр_Цезаря>. Перевірено: 05.05.2019.
3. Программа ‘Шифр Цезаря’. [Електронний ресурс]. Режим доступу: <https://www.bibliofond.ru/view.aspx?id=865956#text>. Перевірено: 05.05.2019.
4. Франчук В.М. Захист інформаційних ресурсів: криптографічні та стеганографічні методи захисту даних. Посібник для викладачів, вчителів та студентів інформатичних спеціальностей. – К.: НПУ імені М.П. Драгоманова, 2012. – 120 с. [Електронний ресурс]. Режим доступу: <http://www.vfranchuk.npu.edu.ua/images/files/statty/32_ZIR_cript.pdf>. Перевірено: 06.05.2019.
5. Методы защиты информации. Шифр Цезаря. [Електронний ресурс]. Режим доступу: <http://5fan.ru/wievjob.php?id=41653>. Перевірено: 06.05.2019.
6. Шифр Цезаря з ключевым словом. [Електронний ресурс]. Режим доступу: <https://studopedia.ru/19_326576_shifr-tsezarya-s-klyuchevim-slovom.html>. Перевірено: 06.05.2019.
7. 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» та 040303 – «Системний аналіз» усіх форм навчання. – Черкаси: Вид. від. ЧНУ імені Богдана Хмельницького, 2015. – 32 с.
8. C Sharp. [Електронний ресурс]. Режим доступу: <https://uk.wikipedia.org/wiki/C_Sharp>. Перевірено: 07.05.2019.
9. Java [Електронний ресурс]. Режим доступу: <https://uk.wikipedia.org/wiki/Java>. Перевірено: 07.05.2019.
10. Сравнение C Sharp и Java. [Електронний ресурс]. Режим доступу: <https://ru.wikipedia.org/wiki/Сравнение_C_Sharp_и_Java>. Перевірено: 07.05.2019.