Міністерство освіти і науки України

НТУУ «КПІ ім. Ігоря Сікорського»

Навчально-науковий інститут атомної та теплової енергетики

Кафедра цифрових технологій в енергетиці

Лабораторна робота №1

з дисципліни «Безпека інформаційних систем»

«Шифр Цезаря»

Варіант № 22

Виконав: Студент групи ТР-12

Ковальов Олександр

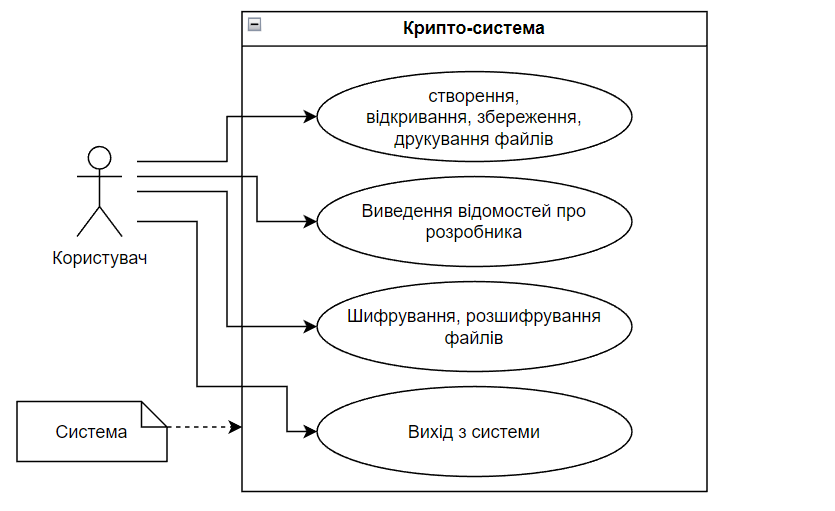
Перевірив: доцент, к.ф.-м.н.

Тарнавський Ю. А.

КИЇВ 2023

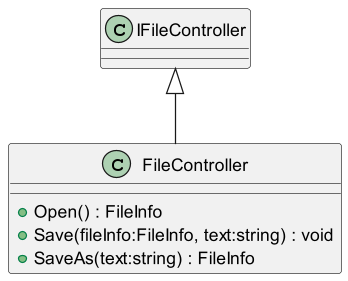
**Мета роботи.** Розробити криптосистему на основі шифру Цезаря.

**Діаграма прецедентів.**

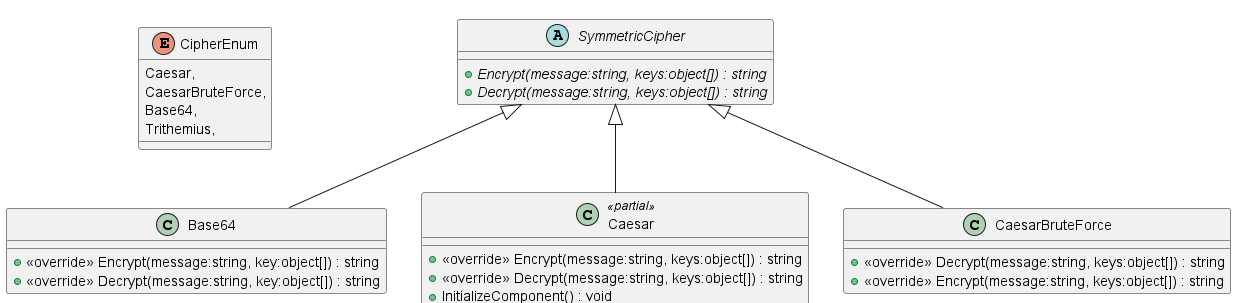


**Діаграма класів.**

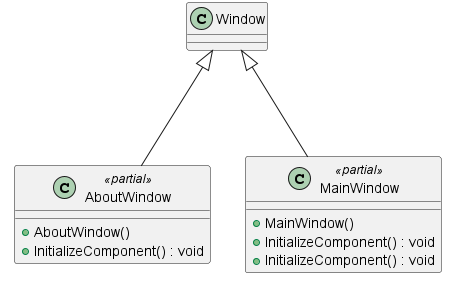
Контроллер файлів – наслідується від інтерфейсу з визначеним API (Applied Program Interface): методи Open, Save, SaveAs.



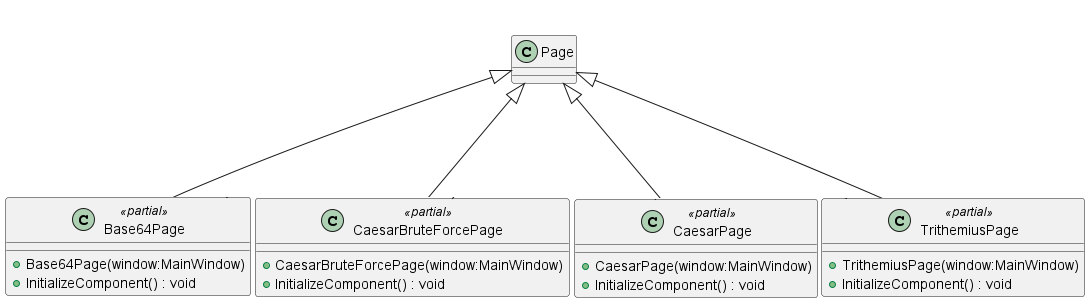
В просторі імен Cryptography знаходяться всі шифри, які наслідуються від класу SymmetricCipher. Також там є перелік CipherEnum.



Вікна знаходяться в просторі імен View. MainWindow – головне, AboutWindow призначене для виведення інформації про розробника.



Сторінки знаходяться в тому ж просторі імен. Вони відповідають за кнопки «Шифрування» та «Розшифрування», можливий ключ.



**Фрагмент коду з реалізацією алгоритму шифрування/розшифрування.**

public override string Encrypt(string message, params object[] keys){

var key = Validate(keys);

return CaesarCipher(message, key);

}

public override string Decrypt(string message, params object[] keys){

var key = Validate(keys);

return CaesarCipher(message, -key);

}

private string CaesarCipher(string message, int key){

var sb = new StringBuilder();

foreach (var c in message)

{

if (EscapeSequence.Contains(c))

{

sb.Append(c);

continue;

}

sb.Append((char) ((c + key) % UnicodeCardinal));

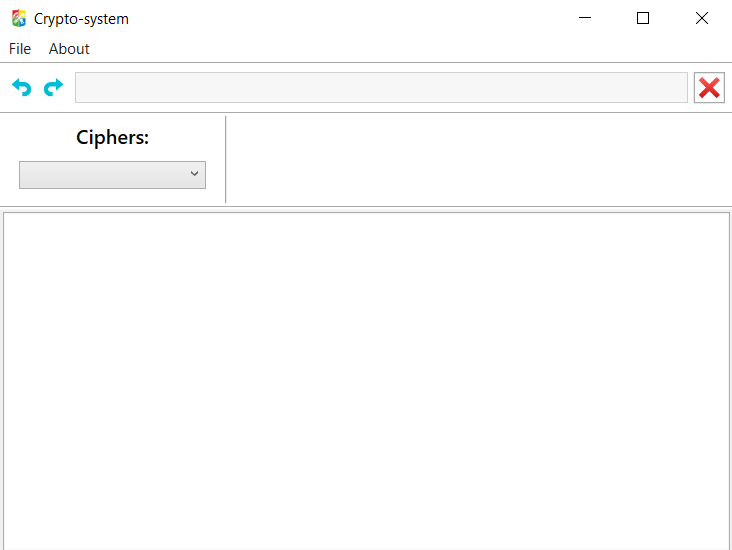
}

return sb.ToString();

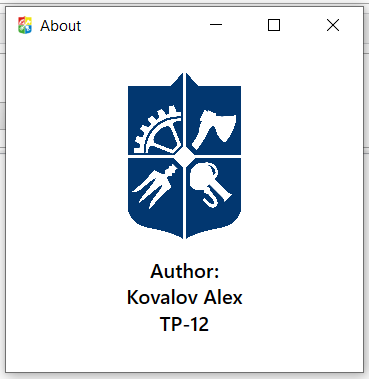
}

**Скріншоти програми.**

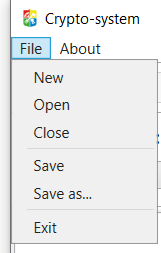
Початкове вікно:



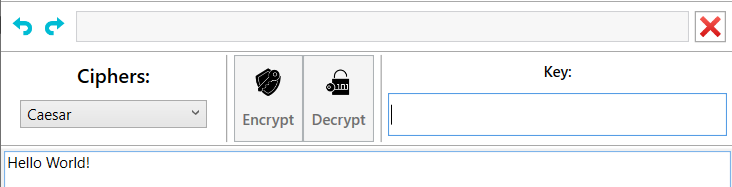
Відомості про розробника:



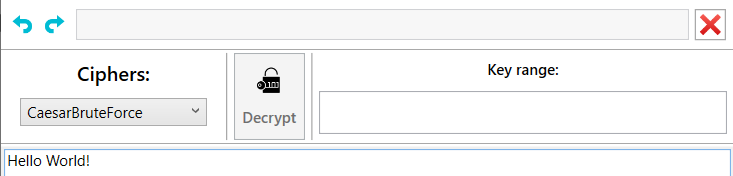
Меню роботи з файлом:



Панель для роботи з шифром Цезаря:



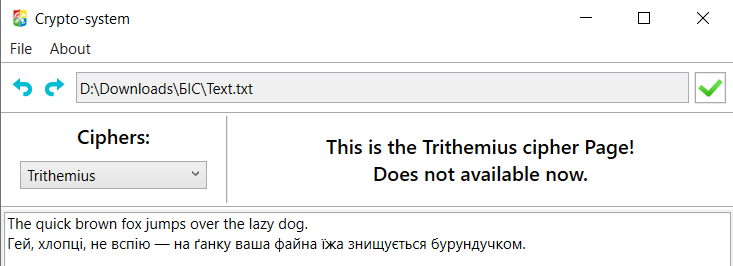
Панель для роботи з декодуванням шифра Цезаря методом «грубої сили»:



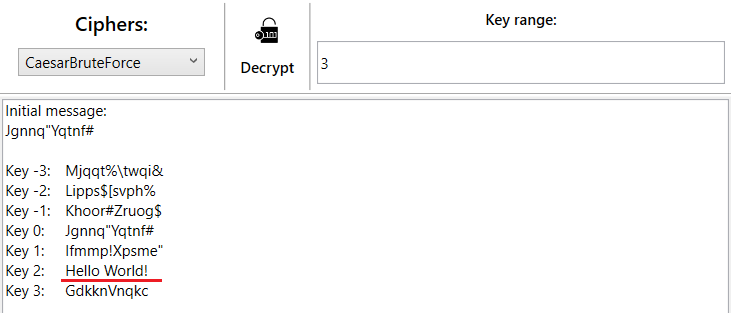
Панель для роботи з кодуванням Base64:



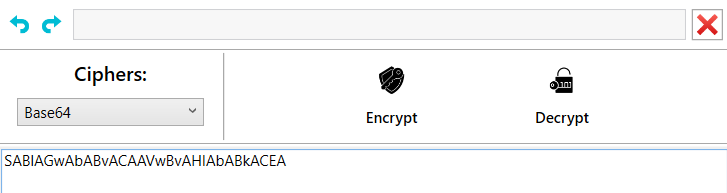
Приклад завантаження тексту з файлу:



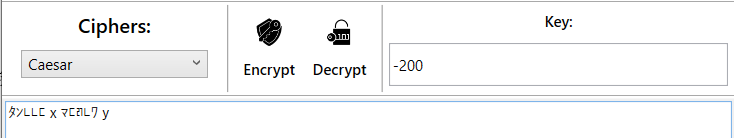
Приклад «брутфорсу»:



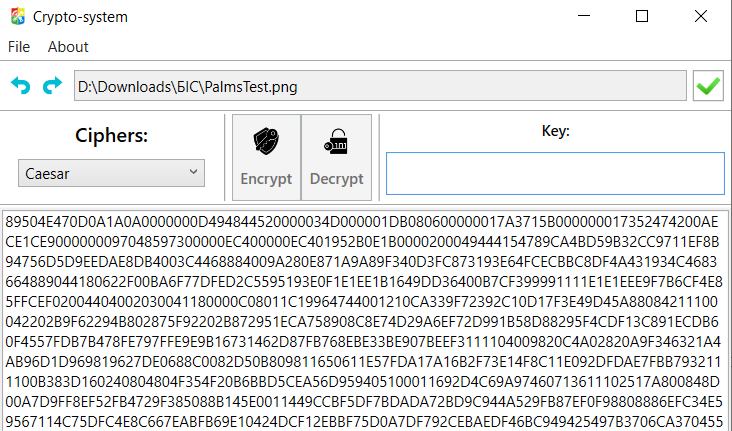
“Hello World!” в кодуванні Base64:



“Hello World!” зі здвигом у -200 символів (Шифр Цезаря):



Приклад відкритого зображення з розширенням .png у програмі.

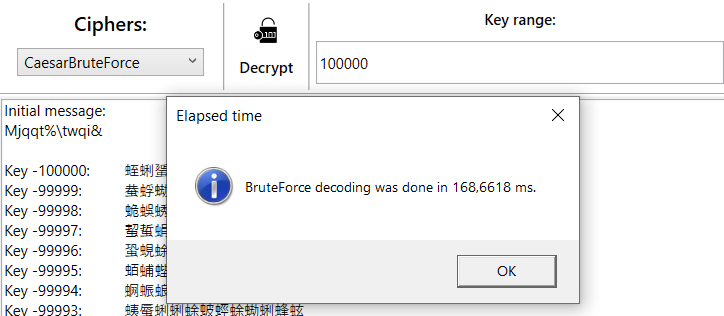


**Висновок:** за результатами виконання цієї лабораторної роботи було ознайомлено з принципом роботи шифру Цезаря. Також, була додатково реалізована можливість кодування тексту та файлів за допомогою кодування Base64. Окрім цього, можна розшифрувати повідомлення закодоване шифром Цезаря за допомогою методу «грубої сили».

З цікавого:

Під час написання алгоритму шифрування/розшифрування було виявлено, що алфавіт Unicode містить 1 114 112 кодових символів. Велика частина з них не відображається, так як є допоміжними.

Також, при розробці був зроблений приблизний аналіз перебору методом грубої сили. Він дуже невигідний для використання – його складність як мінімум , що є неприйнятним. Для прикладу, дешифрування зашифрованої фрази “Hello World” з діапазоном ключів займає 168 мс. Для великих об’ємів інформації дуже затратно по часу обробляти такі масиви інформації.



Окремо можна відмітити, що так як використовується кодування Unicode, то один символ може займати від одного до чотирьох байтів. Тобто, зберігати текст в такому форматі також невигідно. Інформація зберігається в кучі, а саме в SOH – Small Object Heap. Це особливість середовища CLR (платформа .NET). Програма може вилітати, якщо зберігати там досить багато інформації. Якщо використовувати конкатенацію рядків, то швидко заб’ється і пам’ять, і пройде багато часу – тому що кожного разу буде створюватись новий екземпляр об’єкту. Тому потрібно використовувати StringBuilder – цей тип даних дозволяє зберігати в собі рядки більш економно. Але це не рятує від переповнення.

Изображение выглядит как текст

Автоматически созданное описание

Також було виявлено, що деякі послідовності можуть втрачатись при послідовному шифруванні та розшифруванні. Тому, не можна таким чином обробляти будь-що, окрім звичайного тексту. Для зображень та інших типів файлів потрібно використовувати такий алгоритм: файл потрібно прочитати побайтово, байти перевести в шістнадцяткову систему числення, а потім в текст. Після цього його можна шифрувати, а потім і розшифрувати. Так файли не втрачають цілісність.

