

Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут ім. І. Сікорського»

Кафедра інженерії програмного забезпечення в енергетиці

Практична робота № 3
з курсу: «*Безпека програмного забезпечення*»

Виконав:
студент 4-го курсу,
групи ТВ-21
Цвігун Богдан

Київ 2025

Практична робота № 3

Завдання:

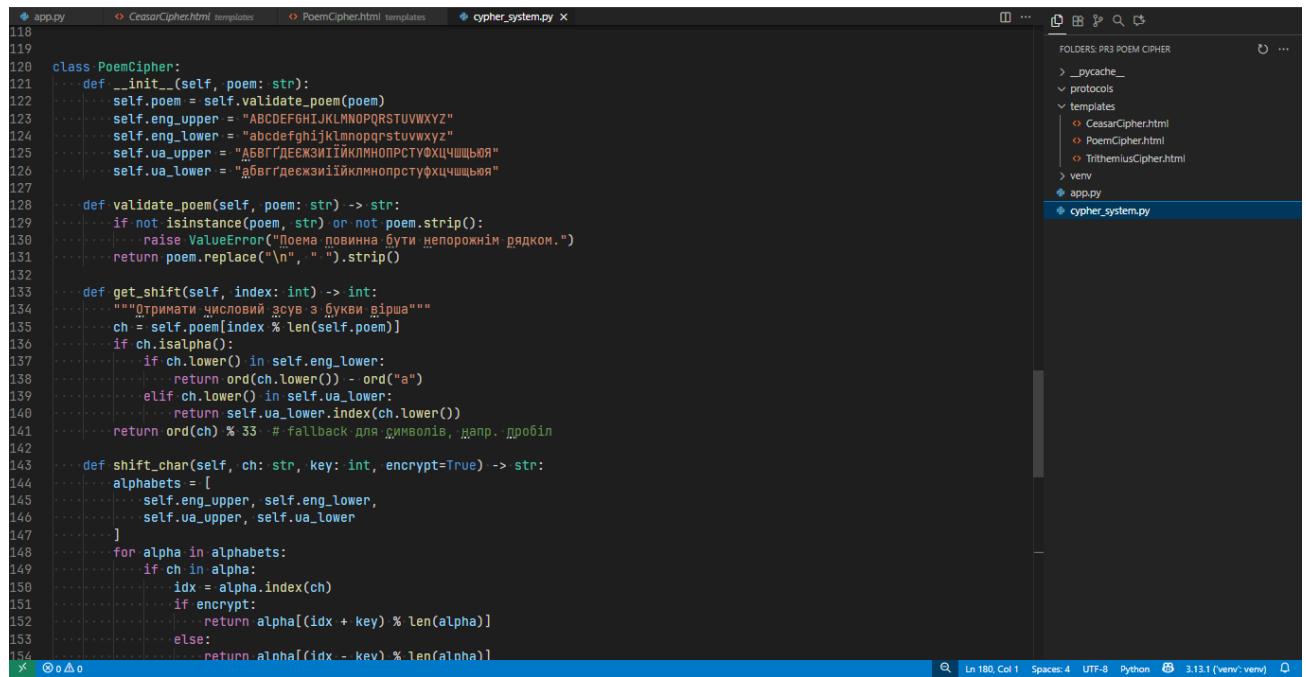
1. Розробіть інтерфейс криптографічної системи для реалізації шифрування з використанням вірша.
2. Доповніть систему класів з попередніх лабораторних робіт класами та методами, необхідними для шифрування і розшифрування віршованим шифром.
3. Виконати тестування роботи системи.

Хід виконання:

Для розробки веб-додатку було використано фреймворк Flask, який забезпечує ефективну взаємодію між серверною частиною та інтерфейсом користувача. Інтерфейс реалізовано з використанням сучасних веб-технологій, зокрема CSS-фреймворку Bulma, що надає адаптивний і естетичний дизайн, а також шаблонізатора Jinja2, який відповідає за динамічне формування вмісту HTML-сторінок.

The screenshot shows the Poem Cipher web application. At the top, there is a navigation bar with links: 'Poem Cipher' (highlighted in blue), 'Головна', 'Про розробника', and 'Вихід'. On the right side of the header is a language switcher 'Підтримка EN / UA'. The main content area has a dark background with white text. It features a form for encryption/decryption. The form includes fields for 'Текст' (Text) and 'Ключ (вірш або рядок)' (Key (verse or line)). Below these are two buttons: 'Шифрувати' (Encrypt) and 'Розшифрувати' (Decrypt). A note below the key field says: 'Ключ має бути текстом — використовується для генерації таблиці шифру.' To the right of the form is a sidebar with sections: 'Як працює ключ' (How the key works), 'Поради' (Advice), and footer information: 'Розробник: ТВ-21 Цагун Богдан', 'Poem Cipher — підтримка EN / UA залишить від реалізації шифру на сервері', and a note about leaving the key unencrypted.

Шифр Поеми є поліалфавітним методом шифрування, в якому ключ формується на основі заданого вірша або текстового фрагмента. Кожна літера ключа визначає зсув для відповідної літери відкритого тексту, а при завершенні ключа процес повторюється циклічно. Такий підхід робить шифр більш стійким до простого частотного аналізу, оскільки одна й та сама літера може бути зашифрована різними символами залежно від позиції у тексті. Попри це, через передбачуваність структури ключа та обмежену довжину вірша шифр Поеми залишається вразливим до статистичних методів розкриття і використовується переважно в навчальних цілях для демонстрації принципів поліалфавітного шифрування.



The screenshot shows a code editor interface with several tabs open. The main tab contains Python code for a 'PoemCipher' class. The code defines various methods for validating poems, getting shifts based on characters, and shifting characters. It uses four alphabets: English uppercase, English lowercase, Ukrainian uppercase, and Ukrainian lowercase. The code is annotated with comments explaining its logic. The sidebar shows project files including 'CeasarCipher.html', 'PoemCipher.html', 'TrithemiusCipher.html', and 'app.py'. The bottom status bar indicates the file is 180 lines long, in Col 1, with 4 spaces, in UTF-8 encoding, and version 3.13.1 ('venv' venv).

```
118
119
120 class PoemCipher:
121     def __init__(self, poem: str):
122         self.poem = self.validate_poem(poem)
123         self.eng_upper = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
124         self.eng_lower = "abcdefghijklmnopqrstuvwxyz"
125         self.ua_upper = "АБВГДЕЖЗИЙКЛМНОРСТУФЧЦЩЬЯ"
126         self.ua_lower = "абвгдежзиийклмнопрстуфчцищъя"
127
128     def validate_poem(self, poem: str) -> str:
129         if not isinstance(poem, str) or not poem.strip():
130             raise ValueError("Поема повинна бути непорожнім рядком.")
131         return poem.replace("\n", " ").strip()
132
133     def get_shift(self, index: int) -> int:
134         """Отримати зсув з букві вірша"""
135         ch = self.poem[index % len(self.poem)]
136         if ch.isalpha():
137             if ch.lower() in self.eng_lower:
138                 return ord(ch.lower()) - ord("a")
139             elif ch.lower() in self.ua_lower:
140                 return self.ua_lower.index(ch.lower())
141             return ord(ch) % 33 # fallback для символів, напр. пробіл
142
143     def shift_char(self, ch: str, key: int, encrypt=True) -> str:
144         alphabets = [
145             self.eng_upper, self.eng_lower,
146             self.ua_upper, self.ua_lower
147         ]
148         for alpha in alphabets:
149             if ch in alpha:
150                 idx = alpha.index(ch)
151                 if encrypt:
152                     return alpha[(idx + key) % len(alpha)]
153                 else:
154                     return alpha[(idx - key) % len(alpha)]
```

Повний вихідний код проєкту доступний у репозиторії, що буде додано до завдання.

Клас PoemCipher реалізує симетричний метод поліалфавітного шифрування, у якому ключем виступає вірш або довільний текстовий фрагмент. Кожна літера ключа визначає числовий зсув, який використовується для шифрування відповідного символу у вхідному повідомленні. Коли кінець ключа досягається, зсуви повторюються циклічно, що забезпечує рівномірне застосування шаблону шифрування до всього тексту.

Під час ініціалізації виконується перевірка валідності вхідного ключа-вірша та підготовка алфавітів — українського та англійського у верхньому й нижньому регістрах. Метод `get_shift()` обчислює величину зсуву для певної позиції, перетворюючи символ вірша у числове значення. Процедура `shift_char()` виконує безпосереднє зміщення літери у відповідному алфавіті під час шифрування або розшифрування, залишаючи неалфавітні символи (цифри, пробіли, розділові знаки) без змін.

Методи `encrypt()` і `decrypt()` реалізують двосторонній процес перетворення тексту, синхронно використовуючи символи ключа для кожної позиції вхідного рядка. Такий підхід забезпечує різні зсуви для однакових літер залежно від їхнього розташування у тексті, що підвищує криптографічну стійкість порівняно з простими моноалфавітними шифрами.

The screenshot shows the Poem Cipher web application interface. At the top, there is a navigation bar with links: Poem Cipher, Головна, Про розробника, Вихід, and Підтримка EN / UA. The main content area has a title "Poem Cipher" and a subtitle "Шифрування та розшифрування за ключем «Вірш»". Below this, there are two input fields: "Текст" containing placeholder text "Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s", and "Ключ (вірш або рядок)" containing the key "KEYWORD". There are two buttons: "Шифрувати" (Encrypt) and "Розшифрувати" (Decrypt). The "Розшифрувати" button is highlighted in blue. Below the buttons is a "Результат" (Result) section showing the encrypted text: "Vspraa Zscyk eg jiwjtu rpw ralik rp xfa dliixgju rqn xwjsjhdxgiu zqnuqfrp. Oyvi Wgvqf twg shor rds zqnuqfrp v cxiyjrun hsap wobr ajvu cmlys kko 1500w". To the right of the result is a sidebar with sections "Як працює ключ" (How the key works), "Поради" (Advice), and developer information: "Розробник: ТВ-21 Цецич Богдан", "Poem Cipher — підтримка EN / UA залишить від реалізації шифру на сервері".

This screenshot shows the same Poem Cipher application interface, but the "Розшифрувати" (Decrypt) button is now highlighted in blue. The result section displays the decrypted text: "Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s". The sidebar remains the same, providing developer information.

Висновок:

У результаті виконання роботи було розроблено веб-застосунок, який реалізує шифрування та розшифрування текстів за допомогою віршованого (Poem) шифру. Система побудована на основі фреймворку Flask із використанням шаблонізатора Jinja2 та CSS-фреймворку Bulma, що забезпечило зручний інтерфейс і коректну взаємодію між клієнтською та серверною частинами. Реалізований клас PoemCipher продемонстрував принципи поліалфавітного шифрування, а проведене тестування підтвердило працездатність і правильність реалізації методів шифрування та розшифрування.