Міністерство освіти і науки України
Дніпровський національний університет імені Олеся Гончара
Факультет прикладної математики і комп’ютерних технологій
Кафедра комп’ютерних технологій

ЗВІТ З ЛАБОРАТОРНОЇ РОБОТИ № 2

з курсу «Методи кібербезпеки»

на тему «Найпростіші алгоритми шифрування»

Варіант №2

Виконав:
студент гр. ПА-22-2
Овдієнко Андрій

Дніпро
2025

Зміст

1. Постановка задачі

Програмно реалізувати 2 найпростіші алгоритми шифрування.

2. Опис розв’язку

2.1 Алгоритм Цезаря

Для спрощення реалізації алгоритму припустимо, що ми працюємо з текстом англійського типу, алфавіт якого такий: ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z'].

Суть полягає в тому, що є певний текст (набір символів) і ключ шифрування, але ключ має бути цілим числом.

Якщо текст містить літеру з алфавіту, то її слід замінити на ту, що зміщена на кількість одиниць, задану ключем, а якщо алфавіт закінчився, почати з початку. Наприклад, (текст, ключ) = (“YES”, 13) – містить кожен символ алфавіту та зсуває його на 13 одиниць праворуч: “Y” – “L”; “E” – “R”; “S” – “F” – отримуємо: “LRF” – зашифроване слово “YES”. Для розшифрування достатньо використати аналогічну операцію, але зі зсувом ліворуч, якщо ми дійшли до початку, до наступного символу з кінця.

Алгоритм працює аналогічно з числами, але має свій алфавіт: ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']. Наприклад, (текст, ключ) = (“673”, 4) – містить кожен символ алфавіту та зсуває його на 4 одиниці праворуч: “6” – “0”; “7” – “1”; “3” – “7” – отримуємо: “017” – зашифроване слово “673”. Для розшифровки достатньо використати аналогічну операцію, але зі зсувом вліво, якщо ви дійшли до початку, до наступного символу з кінця.

2.2 Алгоритм XOR

Алгоритм XOR базується на мат логіці. Побудуємо таблицю істинності.

Таблиця 1 – XOR – таблиця істинності.

A	B	A^B
0	0	0
0	1	1
1	0	1
1	1	0

Можна, по-перша, ми працюємо в булевій системі (двоїчній), по-друге, якщо сиволи рівні – вони з операцією ^ дають 0, інакше 1.

Нехай у нас є текст типу: “Text.” – а ключ – “key”. Тоді сиволи в ключі повторюються стільки разів, щоб він був рівним тексту, який кодується. У данному випадку логіка ключа буде: “keyke” – бо “Text.” Має 5 символів, а ключ 3, тому він повторюється. А якщо текст був: “key”- а ключ: “Text.” – так би і залишили.

Знов нагадаю, що нехай у нас є текст типу: “Text.” – а ключ – “key”. Тоді ми маємо застосовувати операцію ^ до відповідних символів із тексту до ключа: “T” – “k”, “e” – “e”, “x” – “y”, “t” – “k”, “.” – “e”. Але перед тим слід їх перевести в двійкову систему, а потім виконати операцію ^, і повернути до типу строки.

Python спрощує нам життя, бо якщо йому дати число в типу integer, то він може сам все перевести в двійкову систему. Тому для пайтона достатньо знайти операцію, яка показує символи з char в int, а з int у char – тобто юнікод. Наприклад: “T” – “84”, “e” – “101”, “x” – “120”, “t” – “116”, “.” – “46”, “k” – “107”, “e” – “101”, “y” – “121”. Це можна зробити за допомогою функції ord(‘A’), де A – це певний символ. А щоб перетворити у символьний тип з ASCII - chr(65).

Зауважу, якщо ми маємо деякий текст типу: “Text.” – а ключ – “key”. Зробимо шифрування XOR, то отримаємо:

Рисунок 1 – Зашифроване слово “Text.” Із ключем “key”.

І якщо цю операцію повторити – вийде знову: “Text.”.

3. Код програми

```
alphabet = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z']
numbers = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']


class Main:
    def __init__(self, value="",key=""):
        self.__value = value
        self.__key = key

    def get_value(self):
        return self.__value

    def get_key(self):
        return self.__key



    def set_value(self, value):
        self.__value = value

    def set_key(self, value):
        self.__key = value

    def __str__(self):
        return f"The text:\"{self.__value}\".\nThe key:\"{self.__key}\"."


def input_text(value):
```

```
    temp = input("Enter the text: ")
    value.set_value(temp)

def input_key(value):
    temp = input("Enter the key: ")
    value.set_key(temp)

def encrypt(value):
    temp = str(input("Use Caesar cipher or XOR? (c/x):")).lower().replace(' ', '')
    if temp == 'c':
        in_caesar(value)
    elif temp == "x":
        in_xor(value)
    else:
        print("\nYour choice is wrong!")

def decrypt(value):
    temp = str(input("Use Caesar cipher or XOR? (c/x):")).lower().replace(' ', '')
    if temp == 'c':
        out_caesar(value)
    elif temp == "x":
        out_xor(value)
    else:
        print("\nYour choice is wrong!")

def menu(value):

    while True:
        print("\n\nMenu:\n")
        print("1.Enter the text.")
```

```
        print("2.Enter the key.")
        print("3.Encrypt.")
        print("4.Decrypt.")
        print("5.Show the text and the key.")
        print("6.Exit")

        try:
            choice = int(input("\nYour choice: "))
        except Exception:
            print("\nInvalid choice.")
            continue


        if choice == 1:
            input_text(value)
        if choice == 2:
            input_key(value)
        if choice == 3:
            encrypt(value)
        if choice == 4:
            decrypt(value)
        if choice == 5:
            print(f"\n{value}")
        if choice == 6:
            print("\n\nExit...\n\n")
            break

def in_caesar(value):

    try:
```

```
            value.set_key(int(str(value.get_key()).lower().replace(' ','")))
        except Exception:
            print("\nThe key is invalid. The key must be an integer.")
            value.set_key(0)

        if value.get_key() == 0:
            print(value)
            return

        new_text = ""
        for temp in value.get_value():
            if temp in numbers:
                for i, char in enumerate(numbers):
                    if char == temp:
                        new_text += f"{numbers[(i + value.get_key()) % len(numbers)]}"


            elif temp.upper() in alphabet:
                    if temp.isupper():

                        for i, char in enumerate(alphabet):
                            if char == temp:
                                new_text += f"{alphabet[(i + value.get_key()) % len(alphabet)]}"

                    else:

                        temp = temp.upper()

                        for i, char in enumerate(alphabet):
```

```
                    if char == temp:
                        new_text += f"{str(alphabet[(i + value.get_key()) % len(alphabet)]).lower()}"

                else: new_text += temp


            value.set_value(new_text)
            print(value)


        def in_xor(value):
            text = value.get_value()
            key = value.get_key()

            result = ""

            i=0
            for char in text:
                if i == len(key):
                    i = 0

                result += f"{chr(ord(char)^ord(key[i]))}"
                i+=1


            value.set_value(result)
            print(value)

        def out_caesar(value):
```

```
        try:
            value.set_key(int(str(value.get_key()).lower().replace(' ', '')))
        except Exception:
            print("\nThe key is invalid. The key must be an integer.")
            value.set_key(0)

        if value.get_key() == 0:
            print(value)
            return

        value.set_key(-value.get_key())

        new_text = ""
        for temp in value.get_value():
            if temp in numbers:
                for i, char in enumerate(numbers):
                    if char == temp:
                        new_text += f"{numbers[(i + value.get_key()) % len(numbers)]}"
            elif temp.upper() in alphabet:
                if temp.isupper():

                    for i, char in enumerate(alphabet):
                        if char == temp:
                            new_text += f"{alphabet[(i + value.get_key()) % len(alphabet)]}"

                else:

                    temp = temp.upper()
```

```
                    for i, char in enumerate(alphabet):
                        if char == temp:
                            new_text += f"{str(alphabet[(i + value.get_key()) % len(alphabet)]).lower()}"

                else:
                    new_text += temp

            value.set_key(-value.get_key())
            value.set_value(new_text)
            print(value)

        def out_xor(value):
            in_xor(value)

        if name == "__main__":
            main = Main()
            menu(main)
```

4. Опис інтерфейса

Після запуска кода можна побачити головне меню.

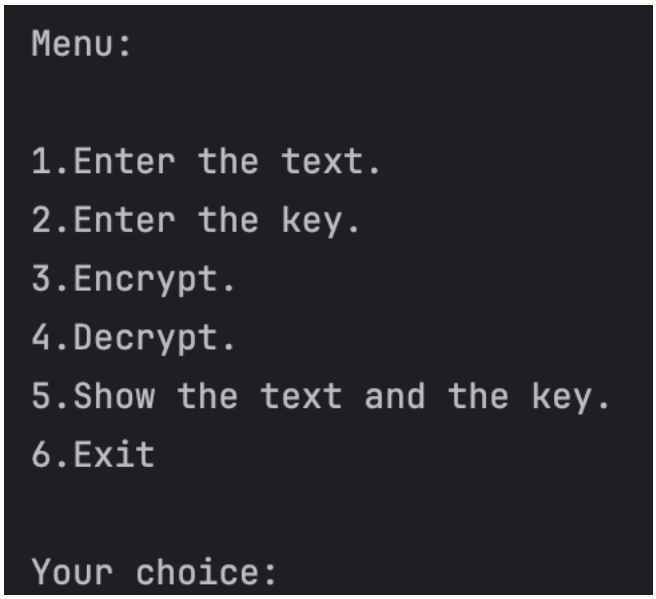

Рисунок 2 – Головне меню програми.

1. У першому пункті вводиться сам текст.
2. У другому пункті вводиться ключ.
3. У третьому обирається тип шифрування (Цезаря/XOR) та відбувається шифрування, виводиться результат.
4. У четвертому обирається тип дешифрування (Цезаря/XOR) та відбувається дешифрування, виводиться результат.
5. У п’ятому пункті просто виводиться текст і ключ, який є на данний момент, бо шифрування і дешифрування змінюють сам текст, а якщо у алгоритмі цезаря ключ не є числом – то ключ становиться нулем.
6. Шостий пункт – вихід.

5. Приклад роботи програми

Відкриємо головне меню програми та введемо текст і ключ.

```
Menu:

1.Enter the text.
2.Enter the key.
3.Encrypt.
4.Decrypt.
5.Show the text and the key.
6.Exit

Your choice: 1
Enter the text: My name is Andrew. I'm 20.


Menu:

1.Enter the text.
2.Enter the key.
3.Encrypt.
4.Decrypt.
5.Show the text and the key.
6.Exit

Your choice: 2
Enter the key: 113


Menu:

1.Enter the text.
2.Enter the key.
3.Encrypt.
4.Decrypt.
5.Show the text and the key.
6.Exit

Your choice: 5

The text:"My name is Andrew. I'm 20.".
The key:"113".
```

Рисунок 3 – Вводимо текст та ключ. Виводимо дані.

Зашифруємо текст методом Цезаря.

```
Menu:

1.Enter the text.
2.Enter the key.
3.Encrypt.
4.Decrypt.
5.Show the text and the key.
6.Exit

Your choice: 3
Use Caesar cipher or XOR? (c/x):c
The text:"Vh wjvn rb Jwmanf. R'v 53.".
The key:"113".


Menu:

1.Enter the text.
2.Enter the key.
3.Encrypt.
4.Decrypt.
5.Show the text and the key.
6.Exit

Your choice: 5

The text:"Vh wjvn rb Jwmanf. R'v 53.".
The key:"113".
```

Рисунок 4 - Шифруємо методом Цезаря. Виводимо дані.

Розшифруємо текст методом Цезаря.

```
Menu:

1.Enter the text.
2.Enter the key.
3.Encrypt.
4.Decrypt.
5.Show the text and the key.
6.Exit

Your choice: 4
Use Caesar cipher or XOR? (c/x):c
The text:"My name is Andrew. I'm 20.".
The key:"113".


Menu:

1.Enter the text.
2.Enter the key.
3.Encrypt.
4.Decrypt.
5.Show the text and the key.
6.Exit

Your choice: 5

The text:"My name is Andrew. I'm 20.".
The key:"113".
```

Рисунок 5 - Розшифруємо методом Цезаря. Виводимо дані.

Зашифруємо текст методом XOR.

```
Menu:

1.Enter the text.
2.Enter the key.
3.Encrypt.
4.Decrypt.
5.Show the text and the key.
6.Exit

Your choice: 3
Use Caesar cipher or XOR? (c/x):x
The text:"|H _P^T ZB r_UATF x \ AA ".
The key:"113".


Menu:

1.Enter the text.
2.Enter the key.
3.Encrypt.
4.Decrypt.
5.Show the text and the key.
6.Exit

Your choice: 5

The text:"|H _P^T ZB r_UATF x \ AA ".
The key:"113".
```

Рисунок 6 - Шифруємо методом XOR. Виводимо дані.

Розшифруємо текст методом XOR.

```
Menu:

1.Enter the text.
2.Enter the key.
3.Encrypt.
4.Decrypt.
5.Show the text and the key.
6.Exit

Your choice: 4
Use Caesar cipher or XOR? (c/x):x
The text:"My name is Andrew. I'm 20.".
The key:"113".


Menu:

1.Enter the text.
2.Enter the key.
3.Encrypt.
4.Decrypt.
5.Show the text and the key.
6.Exit

Your choice: 5

The text:"My name is Andrew. I'm 20.".
The key:"113".
```

Рисунок 7 - Розшифруємо методом XOR. Виводимо дані.

Вийдемо з програми.

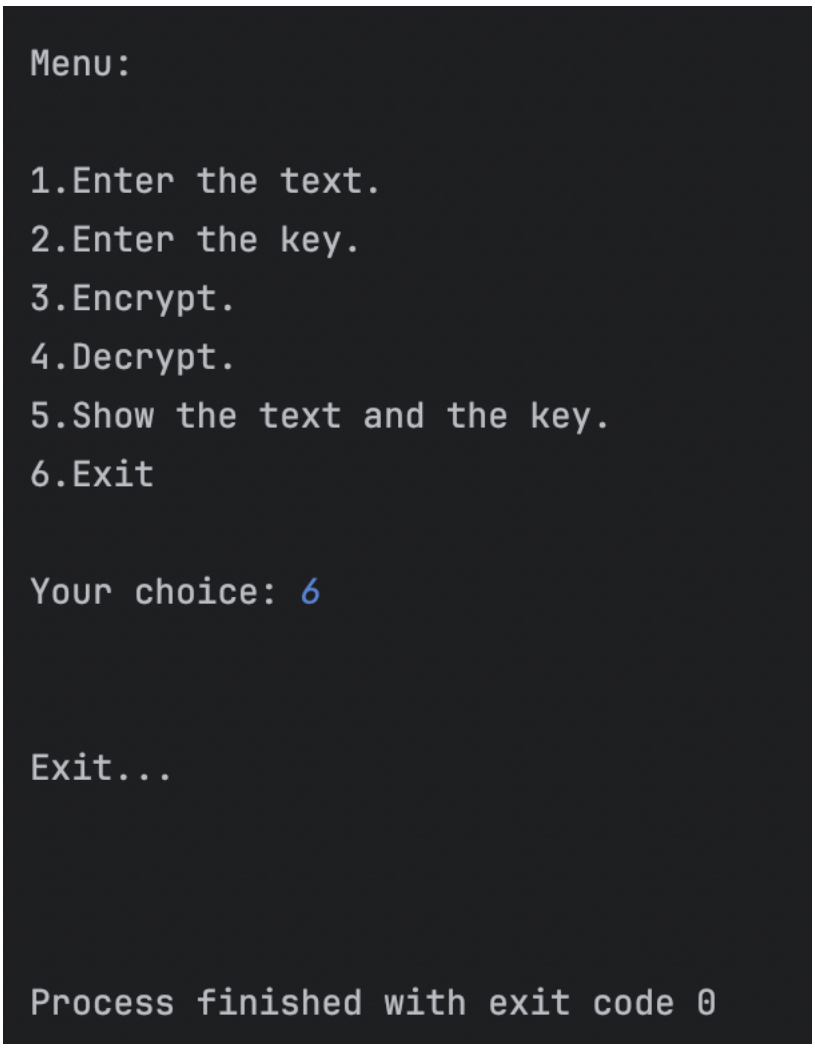

Рисунок 8 – Вихід.

6. Висновки

Вивчили суть методів шифрування. Виявили, що краще шифрувати ключем, який сам по собі має бути складним. Зрозуміли, що для злому методу Цезаря достатньо перебрати ключі від 1 до (довжина алфавіту мінус 1), якщо лише числа або лише символи. Реалізували методи шифрування: Цезар, XOR - а також дешифрування.