**ЛАБОРАТОРНАЯ РАБОТА №1**

“Основы шифрования данных”

по дисциплине

‘Информационная безопасность’

Вариант 5

***Выполнил:***

Соболев Иван Александрович

**Группа:** P34312

***Преподаватель:***

Маркина Татьяна Анатольевна

Санкт-Петербург, 2024

# Цель работы

Изучение основных принципов шифрования информации, знакомство с широко известными алгоритмами шифрования, приобретение навыков их программной реализации.

# Программные и аппаратные средства

Для выполнения лабораторной работы был использован компьютер со следующими характеристиками:

* Процессор: Apple M2
* Видеокарта: Apple M2
* Объем оперативной памяти: 8GB
* Использована операционная система: macOS 14.4.1
* Версия Python: 3.12

# Задание

Реализовать в программе шифрование и дешифрацию файла с использованием квадрата Кардано размером 4х4.

# Листинг разработанной программы

main.py

from encrypt import encrypt, decrypt  
from io\_utils import read\_from\_file, read\_key\_from\_console, decide\_action, write\_to\_file, DECRYPT\_ACTION, ENCRYPT\_ACTION  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 action = decide\_action()  
 if action == ENCRYPT\_ACTION:  
 s = read\_from\_file(action)  
 if s is not None:  
 encrypted, key = encrypt(s)  
 write\_to\_file('encrypted.txt', encrypted)  
 print(f'Результат: "{encrypted}"')  
 print(f'Ключ: "{key}"')  
 elif action == DECRYPT\_ACTION:  
 s = read\_from\_file(action)  
 if s is not None:  
 k = read\_key\_from\_console()  
 if k is not None:  
 decrypted = decrypt(s, k)  
 print(f'Результат: "{decrypted}"')

io\_utils.py

import os  
from typing import Optional, Any  
  
ENCRYPT\_ACTION = "enc"  
DECRYPT\_ACTION = "dec"  
  
"""Функция для вывода ошибок"""  
def print\_red(message: str) -> None:  
 print(f"\033[91m{message}\033[0m")  
  
"""Функция для вывода ключа"""  
def print\_green(message: str) -> None:  
 print(f"\033[92m{message}\033[0m")

"""Функция для вывода ошибок"""  
def print\_red(message: str) -> None:  
 print(f"\033[91m{message}\033[0m")  
  
"""Функция для вывода ключа"""  
def print\_green(message: str) -> None:  
 print(f"\033[92m{message}\033[0m")

"""Функция для вывода ошибок"""  
def print\_red(message: str) -> None:  
 print(f"\033[91m{message}\033[0m")  
  
"""Функция для вывода ключа"""  
def print\_green(message: str) -> None:  
 print(f"\033[92m{message}\033[0m")  
  
"""Функция записи в файл"""  
def write\_to\_file(filename, content):  
 try:  
 with open(filename, 'w', encoding='utf-8') as file:  
 file.write(content)  
 print\_green(f"Данные успешно записаны в файл: {filename}")  
 except IOError as e:  
 print\_red(f"Ошибка записи в файл {filename}: {e}")  
 except Exception as e:  
 print\_red(f"Произошла ошибка: {e}")  
  
"""Функция чтения из файла"""  
def read\_from\_file(action) -> Optional[str]:  
 try:  
 if action == 'enc':  
 filename = input("Введите названия файла с текстом для шифрования: ")  
 else:  
 filename = input("Введите названия файла с текстом для дешифрования: ")  
  
 if not filename.strip():  
 print\_red("Ошибка: имя файла не должно быть пустым.")  
 return None  
  
 if not os.path.isfile(filename):  
 print\_red(f"Ошибка: файл '{filename}' не найден.")  
 return None  
 with open(filename, "r") as f:  
 buf = f.read().split('\n')[0]  
 if not buf.strip():  
 print\_red("Ошибка: файл пуст.")  
 return None  
 return buf  
 except Exception as e:  
 print\_red(f'Ошибка чтения из файла: {str(e)}')  
 return None

"""Функция чтения ключа из консоли"""  
def read\_key\_from\_console() -> Optional[str]:  
 k = input("Введите ключ шифрования: ")  
 if not k.strip():  
 print\_red("Ошибка: ключ не должен быть пустым.")  
 return None  
 return k

"""Функция выбора действия"""  
def decide\_action() -> Any | None:  
 print('Выберите действие: \n'  
 '1. Зашифровать текст с помощью ключа\n'  
 '2. Расшифровать текст с помощью ключа')  
 try:  
 option = int(input("Введите номер: "))  
 if option not in [1, 2]:  
 print\_red(f'Ошибка: опция {option} недоступна!')  
 return None  
 if option == 1:  
 return ENCRYPT\_ACTION  
 elif option == 2:  
 return DECRYPT\_ACTION  
 else:  
 return None  
 except Exception:  
 print\_red(f'Ошибка: неверный ввод. Необходимо ввести целое число.')  
 return None

# transformer.py

from typing import Tuple  
  
"""Функция поворота квадрата на 90 градусов"""  
def rotate\_90(row: int, col: int, n: int) -> Tuple[int, int]:  
 return col, n - 1 - row  
  
"""Функция поворота квадрата на 180 градусов"""  
def rotate\_180(row: int, col: int, n: int) -> Tuple[int, int]:  
 return n - 1 - row, n - 1 - col  
  
"""Функция поворота квадрата на 270 градусов"""  
def rotate\_270(row: int, col: int, n: int) -> Tuple[int, int]:  
 return n - 1 - col, row

"""Функция поворота ключа на 90 градусов"""  
def rotate\_key\_90(key: list[int], side\_len: int) -> list[int]:  
 return [get\_position(\*rotate\_90(\*get\_coordinates(i, side\_len), side\_len), side\_len) for i in key]  
  
"""Функция поворота ключа на 180 градусов"""  
def rotate\_key\_180(key: list[int], side\_len: int) -> list[int]:  
 return [get\_position(\*rotate\_180(\*get\_coordinates(i, side\_len), side\_len), side\_len) for i in key]

"""Функция поворота ключа на 270 градусов"""  
def rotate\_key\_270(key: list[int], side\_len: int) -> list[int]:  
 return [get\_position(\*rotate\_270(\*get\_coordinates(i, side\_len), side\_len), side\_len) for i in key]  
  
"""Функция вычисления позииции ячейки в квадрате по ее координатам"""  
def get\_position(row: int, col: int, n: int) -> int:  
 return row \* n + col  
  
"""Функция вычисления координат ячейки в квадрате по ее позиции"""  
def get\_coordinates(pos: int, n: int) -> Tuple[int, int]:  
 return pos // n, pos % n  
  
"""Функция вычисления позиций поворотов ячейки"""  
def find\_opposites(side\_len: int, window: int) -> list[int]:  
 opposites = [window]  
 row, col = get\_coordinates(window, side\_len)  
 w1 = get\_position(\*rotate\_90(row, col, side\_len), side\_len)  
 w2 = get\_position(\*rotate\_180(row, col, side\_len), side\_len)  
 w3 = get\_position(\*rotate\_270(row, col, side\_len), side\_len)  
 # Если повороты одинаковые, нет смысла их добавлять (выбранное окно - центр квадрата, размер квадрата нечетный).  
 if window != w1:  
 opposites += [w1, w2, w3]  
 return opposites

encrypt.py

import random  
from typing import Tuple  
from transformer import find\_opposites, rotate\_key\_90, rotate\_key\_180, rotate\_key\_270  
  
  
def encrypt\_message(text: str) -> Tuple[str, str]:  
 square\_len = nearest\_square\_greater\_than(len(text))  
 padded\_text = text.ljust(square\_len \*\* 2)  
 square = [i for i in range(square\_len \*\* 2)]  
 key = []  
 # Генерация квадрата Кардано и ключа  
 while len(square) != 0:  
 empty\_cell = random.choice(square)  
 key.append(empty\_cell)  
 windows = find\_opposites(square\_len, empty\_cell)  
 for w in windows:  
 square.remove(w)  
  
 encrypted = []  
 for si in key:  
 encrypted.append(padded\_text[si])  
 for si in rotate\_key\_90(key, square\_len):  
 encrypted.append(padded\_text[si])  
 for si in rotate\_key\_180(key, square\_len):  
 encrypted.append(padded\_text[si])

for si in rotate\_key\_270(key, square\_len):  
 encrypted.append(padded\_text[si])  
 return ''.join(encrypted), '-'.join(map(str, key))  
  
  
def decrypt\_message(encrypted: str, key: str) -> str:  
 size = int(len(encrypted) \*\* 0.5)  
 key = list(map(int, key.split('-')))  
 decrypted = [' '] \* (size \*\* 2)  
  
 quarter = len(encrypted) // 4  
 rotations = [  
 key,  
 rotate\_key\_90(key, size),  
 rotate\_key\_180(key, size),  
 rotate\_key\_270(key, size)  
 ]  
  
 for i, rotation in enumerate(rotations):  
 for j, pos in enumerate(rotation):  
 decrypted[pos] = encrypted[i \* quarter + j]  
  
 return ''.join(decrypted)  
  
  
def nearest\_square\_greater\_than(size: int) -> int:  
 n = 1  
 while n \*\* 2 < size:  
 n += 1  
 return n

# Результаты работы программы

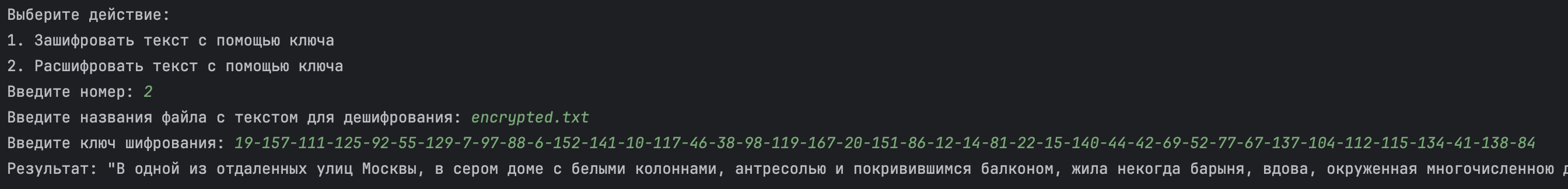
Исходный текст:  
В одной из отдаленных улиц Москвы, в сером доме с белыми колоннами, антресолью и покривившимся балконом, жила некогда барыня, вдова, окруженная многочисленною дворней.

Шифрование текста:  
A black screen with white text

Description automatically generated

Результат шифрования:  
ыюе сив лвйла еека хсвтапулно нлю ж кдкмершдблн д,куибнядн.ас мжн имос ослин,янагмвирчокдьорор реМ енат оноониоео,воамовм ы дкринсниызииы бл йаоенц,аоВмяо, р ворлео гонр

Ключ: 19-157-111-125-92-55-129-7-97-88-6-152-141-10-117-46-38-98-119-167-20-151-86-12-14-81-22-15-140-44-42-69-52-77-67-137-104-112-115-134-41-138-84

Дешифрование текста:  


Дешифрованный текст:  
В одной из отдаленных улиц Москвы, в сером доме с белыми колоннами, антресолью и покривившимся балконом, жила некогда барыня, вдова, окруженная многочисленною дворней.

Отличие – два пробела в конце, так как длина фразы 167, ближайший больший квадрат 169.