Министерство науки и высшего образования Российской Федерации

федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет инфокоммуникационных технологий

Проект

“Шифровальная машинка с применением методов хэширования”

Выполнили:

студентка 3 курса ИКТ, группа К33422

Третьякова Е.С

Проверил:

Мусаев А.А

Санкт-Петербург

2023

Суть проекта

В ходе проекта было реализовано приложение для передачи секретных сообщений.

В приложении пользователь добавляет текст, после чего происходит хэширование и получается зашифрованный текст.

Также, присутствует возможность расшифровки. Пользователь вставляет в поле зашифрованный текст, метод хэширования распознается автоматически. На выходе получается расшифровка.

Ход работы

Проект «Шифровальная машинка с применением методов хэширования» был выполнен функциональным способом. В структуре проекта содержится 6 файлов: 5 из них отвечают за то или иное хэширование или шифрование, а 1 отвечает за сборку графического интерфейса.

Для работы программы необходимо было придумать свой уникальный вид строки для передачи зашифрованного текста со всеми необходимыми данными для возможности дальнейшей дешифровки. Было решено остановиться на следующем решении. Зашифрованная строка принимает вид:
метод_хэширования@зашифрованный_ключ@хэшированный_текст.
Разберём каждый из аспектов программы подробнее.

В основу программы легли 3 основополагающих метода хэширования: умножением, делением и сложением. Выбор в их пользу был сделан в силу их относительной сложности для дешифровки со стороны и простоты использования для знающих людей. Метод хэширования умножением обозначается «###», метод хэширования делением - «%%%» и метод хэширования сложением - «***».

Далее, так как для каждого метода хэширования необходим ключ, то его также необходимо передавать в финальной строке. В файле *key_algos.py* собраны все функции, отвечающие за работу с ключом. Функция *generate_key()* отвечает за случайную генерацию ключа. Он состоит из 33 неповторяющихся букв русского алфавита в случайном порядке. Можно сказать, что для подбора нужного ключа будет необходимо перебрать 33! вариантов. Функция *cypher_key(key)* зашифровывает ключ *key* с помощью шифра Цезаря. Соответственно функция *de_cypher_key(cypher_key)* расшифровывает имеющийся зашифрованный ключ *cypher_key*.

Для каждого из трёх методов хэширования были созданы свои файлы, в которых имеются функции хэширования и получения исходного текста из хэша.

В файле *division_hashing.py* реализован метод хэширования делением. А именно, его выполняет функция *division_hash(key, text)*. В начале находится сумма ключа с помощью функции *sum_of_key(key)*, которая суммирует номер каждого символа в таблице Unicode. После этого происходит посимвольное хэширование. В хэш добавляется подстрока вида *_hash/help_hash_*, где *hash* – это остаток от деления номера хэшируемого символа в таблице Unicode на сумму ключа, а *help_hash* – целочисленное деление этих же переменных. Тогда функции *division_dehash(key, text)* не составит труда посимвольно восстановить исходную строку. Достаточно лишь умножить *help_hash* на сумму ключа и прибавить *hash.*

В файле *multi_hashing.py* реализован метод хэширования умножением. А именно, его выполняет функция *multi_hash(key, text)*. В начале находится сумма ключа с помощью функции *sum_of_key(key)*. Далее устанавливается константа $c = 0.1$. После этого происходит посимвольное хэширование. В хэш добавляется подстрока вида *_hash _*, где *hash* – это находится по формуле *int(сумма_ключа * (номер_элемента_unicode * c)*. Тогда функция *multi_dehash(key, text)* посимвольно восстанавливает исходную строку, выполняя обратные математические преобразования.

В файле *addition_hashing.py* реализован метод хэширования сложением. В силу сложности дешифровки классического метода он был немного переработан. Его выполняет функция *addition_hash(key, text)*. В начале создается список из символов ключа, далее он сортируется и переворачивается. В конце к нему добавляются различные регулярные символы (пробелы, точки, запятые и тд). После этого происходит посимвольное хэширование. В хэш добавляется подстрока вида *index*, где *index* – это индекс элемента в списке. На первый взгляд, метод хэширования может показаться слабым, но учитывая неизвестность ключа и добавление специальных символов в список, без знания алгоритма дешифровка будет занимать много времени. К сожалению, в силу особенностей шифрования поддерживается только русский язык. Тогда функция *addition_dehash(key, text)* посимвольно восстанавливает исходную строку, собирая заново список и выполняя обратные действия с индексами.

Сделав все основные действия, мы можем описать функции из файла *main_functions.py*. Функция *choose_algo()* возвращает случайный метод хэширования из трёх. Функция *make_hash(text)* возвращает строку вида *метод_хэширования@зашифрованный_ключ@хэшированный_текст*, вызывая для этого функции выбора алгоритма, генерации и шифрования

ключа и хэширования текста выбранным методом. А функция *make_dehash(user_text)* выполняет обратные действия. Она разбивает строку на три по символу «@» и вызывает функции дешифровки ключа и дешифровки текущего метода хэширования, возвращая исходную строку пользователя.

В файле *main.py* реализован графический интерфейс с помощью библиотеки tkinter. Реализовано окно с текстом, двумя полями ввода, кнопками и меню с дополнительной функцией очистки форм. После хэширования текста строка автоматически копируется в буфер обмена пользователя. И для более красивого оформления был добавлен стиль из библиотеки ttkthemes.

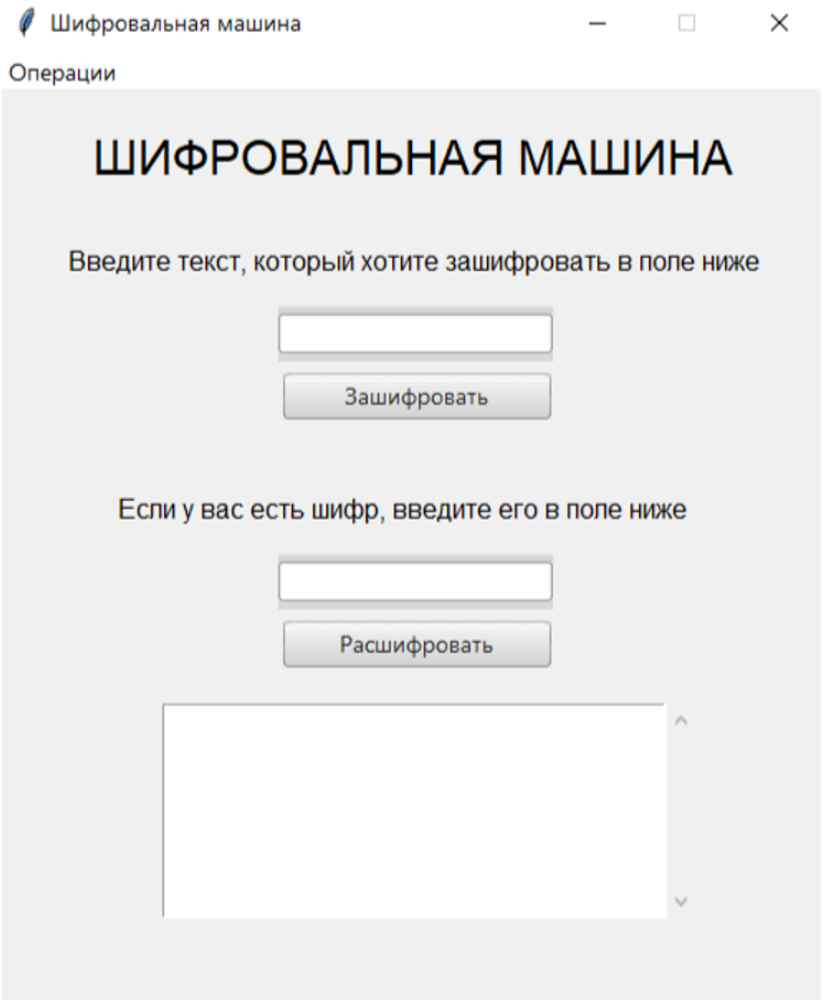

Рисунок 1 – Главное меню программы

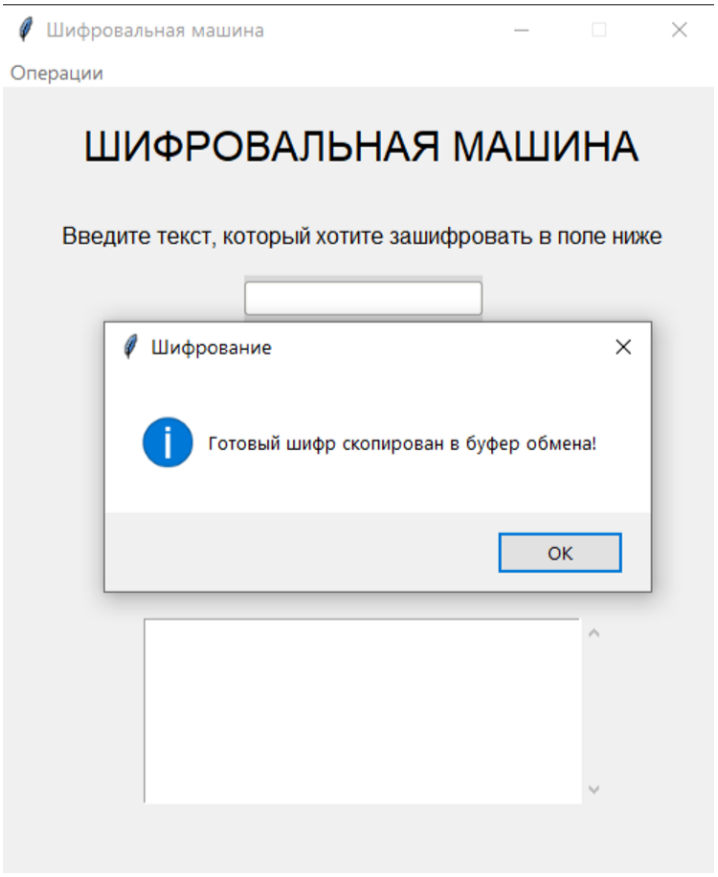

Рисунок 2 – Хэширование введенной строки

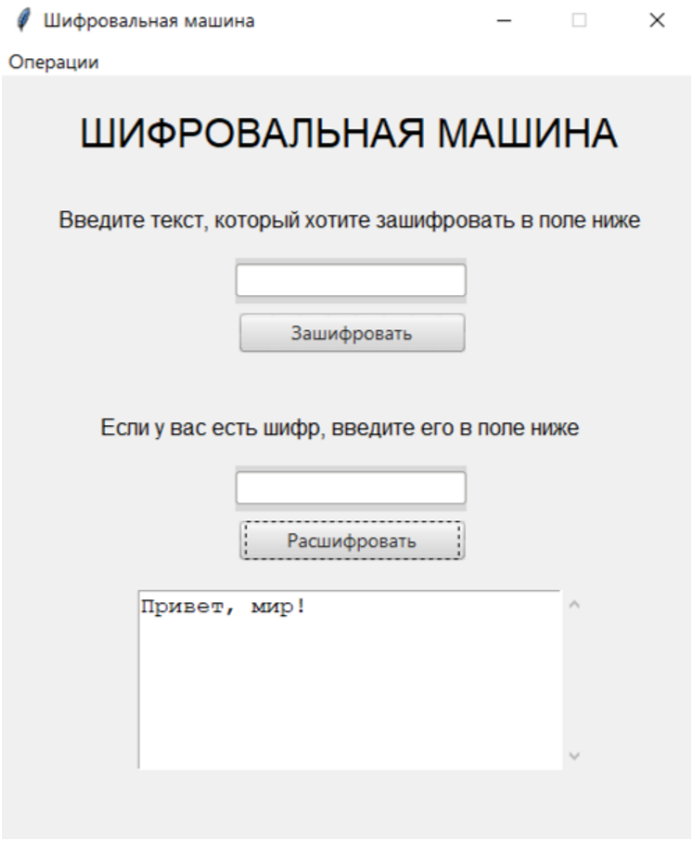

Рисунок 3 – Расшифровка введенной строки

Развитие проекта

Для дальнейшего развития проекта присутствуют разные варианты. Добавление большего количества методов шифрования, например, rsa или Виженер. Также, добавление возможно шифровать и дешифровать текст на английском языке, так как сейчас можно делать это только на русском языке. Более того, можно добавить функционал, при котором пользователь сможет выбрать метод шифрования и дешифрования сам, так как сейчас он выбирается автоматически, но при этом оставить возможность автоматического выбора и распознания метода.