ГУАП

КАФЕДРА № 82

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| доцент, кандидат техн. наук |  |  |  | Е.Л. Турнецкая |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №1 |
| РАЗРАБОТКА ПРОГРАММЫ КРИПТОГРАФИЧЕСКОЙ ЗАЩИТЫ |
| по курсу: МЕТОДОЛОГИЯ И ТЕХНОЛОГИЯ ПРОЕКТИРОВАНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. | 8025М |  |  |  | А.С.Гловацкий |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2020

**Цель работы:**

Разработать программу криптографической защиты, которая выполняет функции шифрования и дешифрования текстовых данных.

**Ход работы:**

Для выполнения работы необходимо выполнить 4 шага:

1. Выбрать алгоритм шифрования
2. Спроектировать интерфейс
3. Реализация алгоритма на выбранном языке программирования
4. Объединение реализации и алгоритма

**Алгоритм шифрования**

В качестве алгоритма шифрования был выбран шифр Виженера.

Шифр Виженера – это метод полиалфавитного шифрования буквенного текста с

использованием ключевого слова. Это простой для понимания и реализации метод, но в то же время он является шифром повышенной надежности, т.к. размывает частоту букв. Это помогает противостоять частотному анализу и прочим статистическим методам дешифровки текста.

Шифрование методом Виженера производится по формуле:

*,*

Где ci – символ закодированного сообщения, pi – символ исходного сообщения, ki – символ ключа, N – мощность алфавита (количество символов в алфавите)

Дешифровка производится по формуле:

Программа способна выполнять шифрование и дешифрование русского и

английского текста. Для каждого алфавита существует две вариации: с цифрами и без цифр.

|  |  |
| --- | --- |
| **Название алфавита** | **Мощность** |
| Русский с цифрами | 54 |
| Русский без цифр | 44 |
| Английский с цифрами | 47 |
| Английский без цифр | 37 |

Таблица 1 – мощность алфавитов

В качестве ключа используется слово соответствующего языка, и если сообщение

длиннее ключа, то символы ключа будут накладываться до тех пор, пока количество символов не сравняется.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Сообщение** | **К** | **л** | **а** | **с** | **с** | **н** | **а** | **я** |  | **п** | **о** | **г** | **о** | **д** | **а** | **!** |
| *Ключ* | *к* | *о* | *т* | *к* | *о* | *т* | *к* | *о* | *т* | *к* | *о* | *т* | *к* | *о* | *т* | *к* |

Таблица 2 – пример наложение ключа

**Интерфейс приложения**

Перед проектированием интерфейса необходимо определиться с необходимым функционалом:

* Ввод данных вручную
* Заполнение текста сообщения из файла
* Сохранение результата в файл
* Преобразование текста в текстовых полях в читаемый вид
* Выбор операции
* Выбор алфавита
* Перезапуск программы

В результате был получен следующий интерфейс:

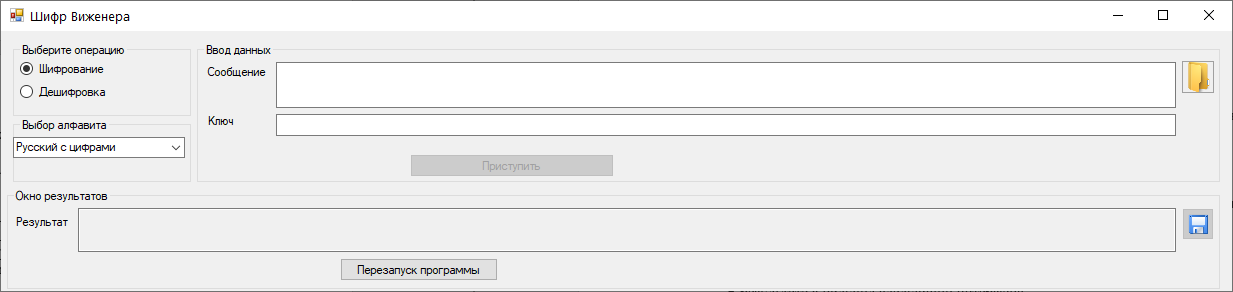


Рисунок 1 – интерфейс приложения

Для реализации приложения использовалась среда разработки Visual Studio 2019, язык программирования C#, приложение базируется на Windows Forms, соответственно было разработано приложение для рабочего стола.

В ходе разработки активно использовался репозиторий GitHub, основанный на системе контроля версий Git.

Рассмотрим методы в коде поближе:

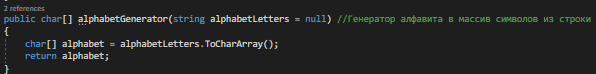


Рисунок 2 – метод для преобразования строки в массив символов

Данный метод получает строку из массива алфавитов и преобразует ее в массив символов, чтобы можно было обращаться к индексу символа в алфавите.

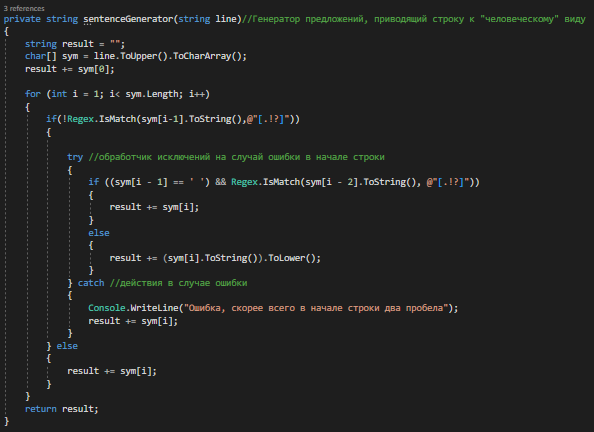


Рисунок 3 – генератор предложений

Данный метод приводит строку к читаемому виду, например строка – «В ТРАВЕ сИДел кузнеЧИК! оН бЫл зеЛёНый.», станет – «В траве сидел кузнечик! Он был зелёный.»

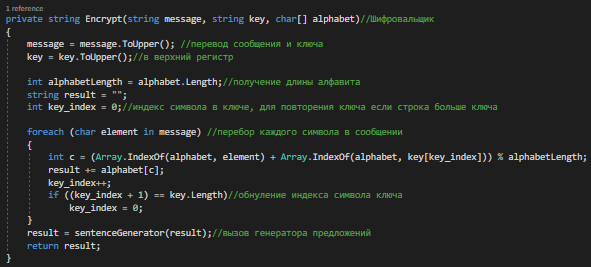


Рисунок 4 – шифровальщик

Данный метод отвечает за шифрование сообщения, для удобства сообщение и ключ переводятся в верхний регистр. Метод получает длину алфавита и начинает перебор каждого символа в сообщении и шифрует по формуле. Итоговая строка прогоняется через генератор предложений.

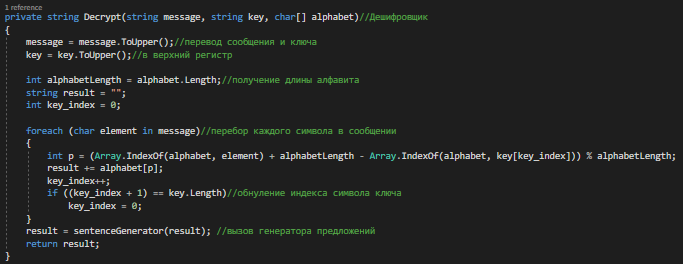


Рисунок 5 – дешифровщик

Данный метод аналогичен методу шифрования, за исключением используемой формулы.

Приложение было протестировано:

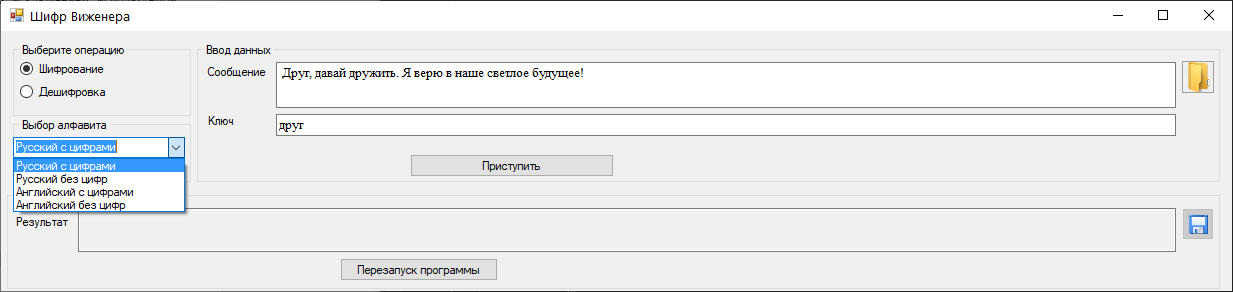


Рисунок 6 – тестирование шифрования с ручным вводом сообщения

Кнопка «приступить» заблокирована до тех пор, пока не введен валидный ключ (язык сообщения и ключа должен совпадать, а также не должно быть цифр и специальных символов), кнопка «сохранить» заблокирована до проведения операции, так как сохранять нечего.

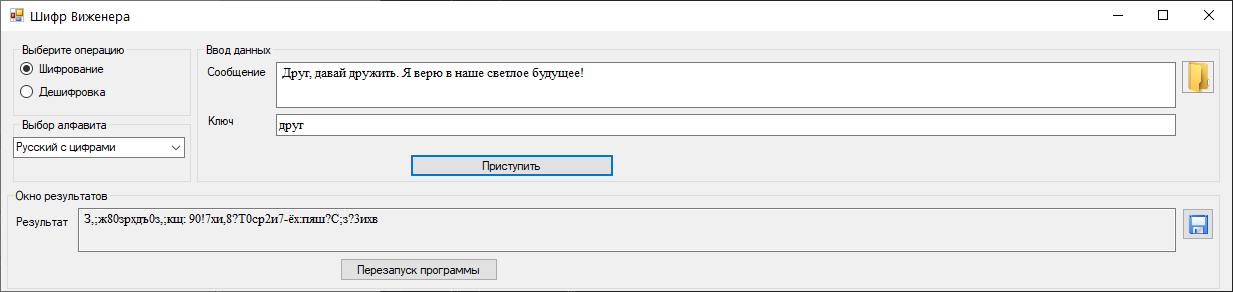


Рисунок 7 – тестирование шифрования с ручным вводом сообщения, результат

Теперь после выполнения шифрования, можно сохранить результат, что мы и сделаем.

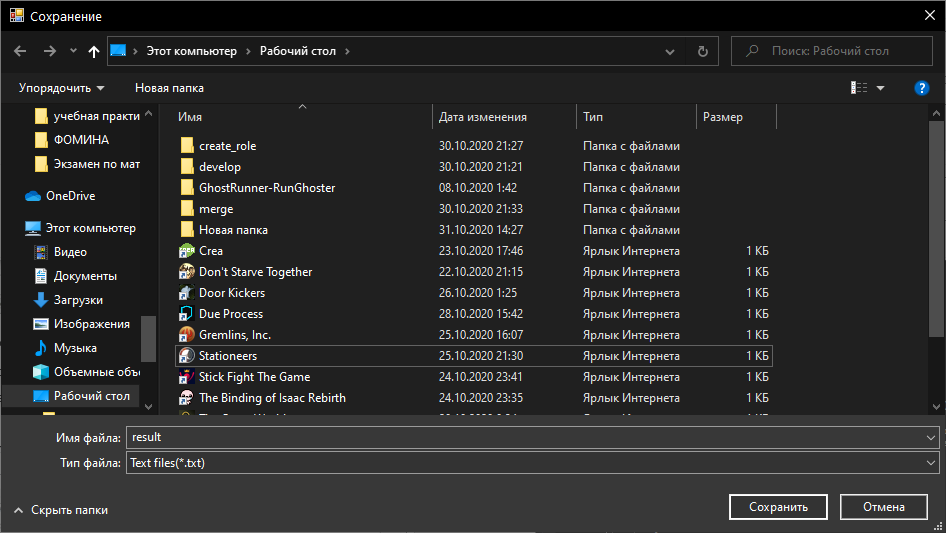


Рисунок 8 – тестирование шифрования с ручным вводом сообщения, сохранение результата

Откроем ранее созданный файл и дешифруем его.

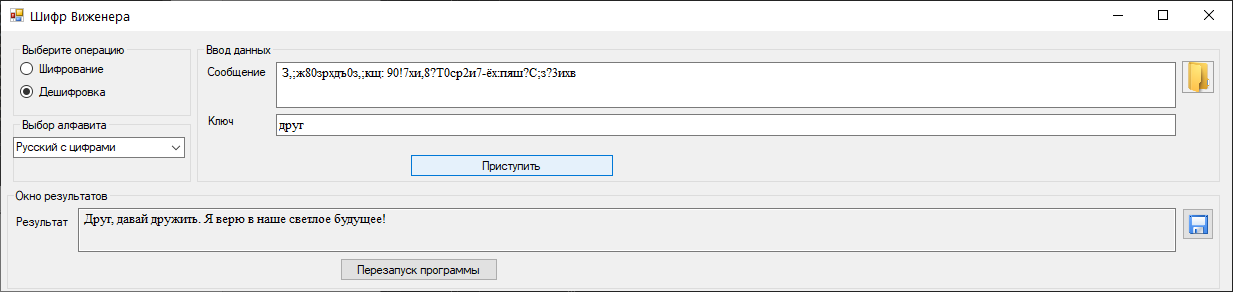


Рисунок 9 – тестирование дешифровки с внесением сообщения из файла

В результате было получено исходное сообщение.

**Вывод:**

Были проанализованы существующие шифры и их алгоритмы, были закреплены навыки разработки приложений для рабочего стола. Была реализована программа для шифрования и дешифрования текста.

Весь исходный код хранится в репозитории по ссылке <https://github.com/kagex/IS_Glovatskiy_8025M>