МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский авиационный институт (национальный исследовательский университет)»

ОТЧЕТ

к практической работе №4 по предмету

«Защита информации»

Институт №3 «Системы управления, информатика и электроэнергетика»

Кафедра 316 «Системное моделирование и автоматизированное проектирование»

Преподаватель:

Коновалов Кирилл Андреевич

Исполнитель:

студент группы М3О-433Б-18

Бондаренко Илья Алексеевич

Отметка: _____

Дата: _____

Москва 2021

ОГЛАВЛЕНИЕ

ЗАДАНИЕ ПРАКТИЧЕСКОЙ РАБОТЫ

Задание: создать программу для демонстрации процесса шифрования и расшифрования текстовых сообщений.

Требования к программе:

- программа должна быть реализована как оконное приложение,
- язык разработки C# или C++,
- допустимые фреймворки: .NET или Qt,
- используя простые криптоалгоритмы, шифровать и расшифровывать информацию в виде текстового сообщения,
- требуемые функции программы:
  1) ввести сообщение (шифртекст);
  2) выбрать направление изменения сообщения (шифрование или расшифрование);
  3) указать криптоалгоритм (например указать название шифра из выпадающего списка);
  4) ввести ключ шифра (текст или файл ключа);
  5) по активации соответствующей функции (сделанной, например в виде появившейся в доступе кнопки) получить сообщение в зависимости от направления изменения: либо зашифрованное, либо расшифрованное.

Требуемые к реализации шифры:

1. Одиночная перестановка;
2. Шифр Цезаря;
3. Шифр Виженера.

АЛГОРИТМ РЕШЕНИЯ

Данная работа выполнена при помощи языка программирования C#. Для создания графического интерфейса пользователя была использована библиотека GTK#.

Интерфейс приложения (GUI)

Окно приложения имеет 2 текстовых поля ввода исходной информации (сообщение и ключ), 1 текстовое поле для вывода результата, кнопку запуска, переключатель режима работы: шифрование или расшифрование, переключатель используемого криптоалгоритма: шифр Цезаря, шифр Виженера, одиночная перестановка. Внешний вид приложения представлен на рисунке 1.

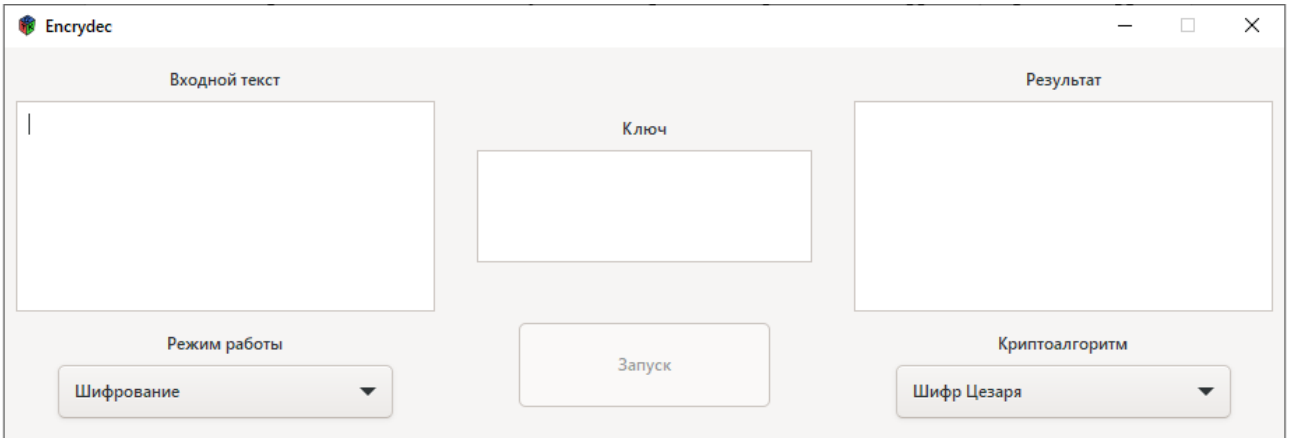

Рисунок 1 — GUI приложения.

Архитектура приложения

Для каждого метода реализован собственный класс, имплементирующий интерфейс IEncryptor (рис. 2). Подобная архитектура позволяет сохранять унифицированный вид классов, а так же реализует простую расширяемость приложения. В соответствии с логикой работы приложения каждый класс должен иметь методы, реализующие шифровку и расшифровку сообщения (в данном случае методы Encrypt и Decrypt.

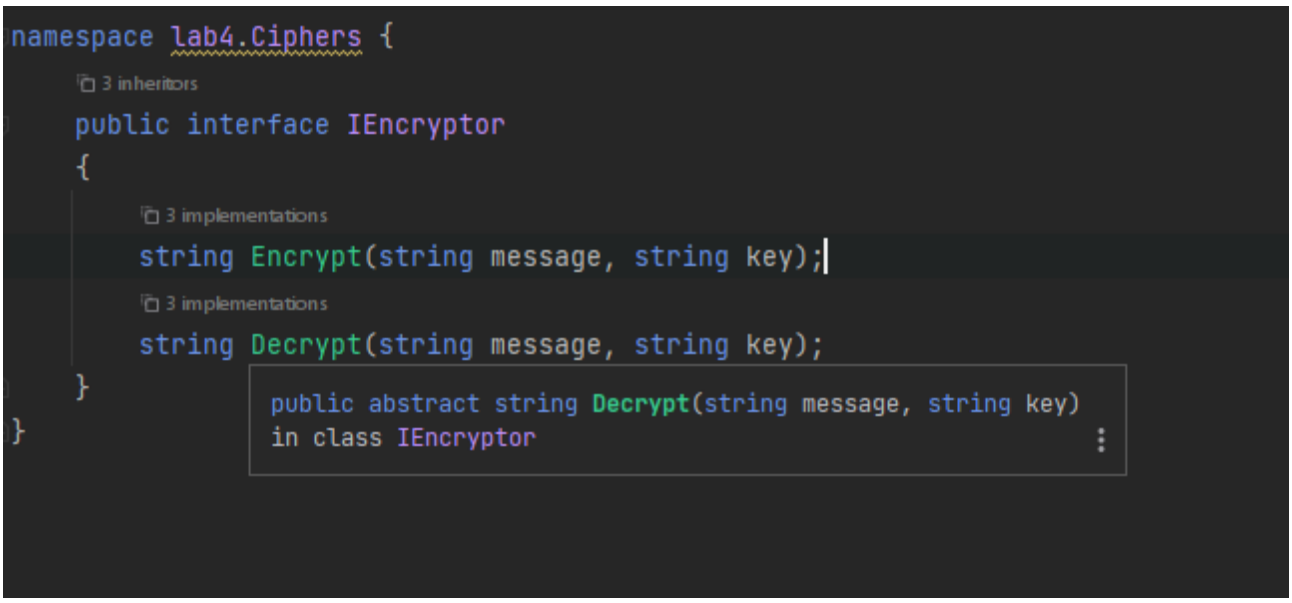

Рисунок 2 — Интерфейс шифрующих классов.

Одиночная перестановка. Реализация и примеры работы

В этом шифре также используется прямоугольная таблица, в которую сообщение записывается по строкам слева направо. Выписывается шифрограмма по вертикалям, при этом столбцы выбираются в порядке, определяемом алфавитом ключа.

На рисунке 3-4 представлен скриншот фрагмента кода, реализующего описанный выше алгоритм.

```
var width int = key.Length;
var height int = message.Length % key.Length == 0
    ? message.Length / key.Length
    :  message.Length / key.Length == 1
        ? 2
        :message.Length / key.Length +1;



var permutation = new int[key.Length];
for (int i = 0; i < permutation.Length; i++)
    permutation [i] = i;
var index = 0;
foreach (var a char in Alphabet)
{
    for (int i = 0; i < permutation.Length; i++)
    {
        if (a == permutation[i])
        {
            permutation[index] = permutation[index] ^ permutation[i];
            permutation[i] = permutation[index] ^ permutation[i];
            permutation[index] = permutation[index] ^ permutation[i];
            index++;
        }
    }
}
```

Рисунок 3 — Фрагмент кода реализации шифра перестановки, часть 1

```
var trail = "";
for (int i = 0; i < message.Length % key.Length; i++)
{
    trail += " ";
}
message = message + trail;
char[] res = new char[message.Length];
for (int i = 0; i < height; i++)
    for (int j = 0; j < width; j++)
    {
        try
        {
            res[i * width + j] = message[permutation[j] * height + i];
        }
        catch (Exception e)
        {
            return e.Message+permutation[j]+" "+i + " "+message.Length + " " + res.Length;
        }
    }


return new string(res);
```

Рисунок 4 — Фрагмент кода реализации шифра перестановки, часть 2

Примеры работы программы на шифрование и расшифровку представлены на рисунках 5 и 6 соответственно.

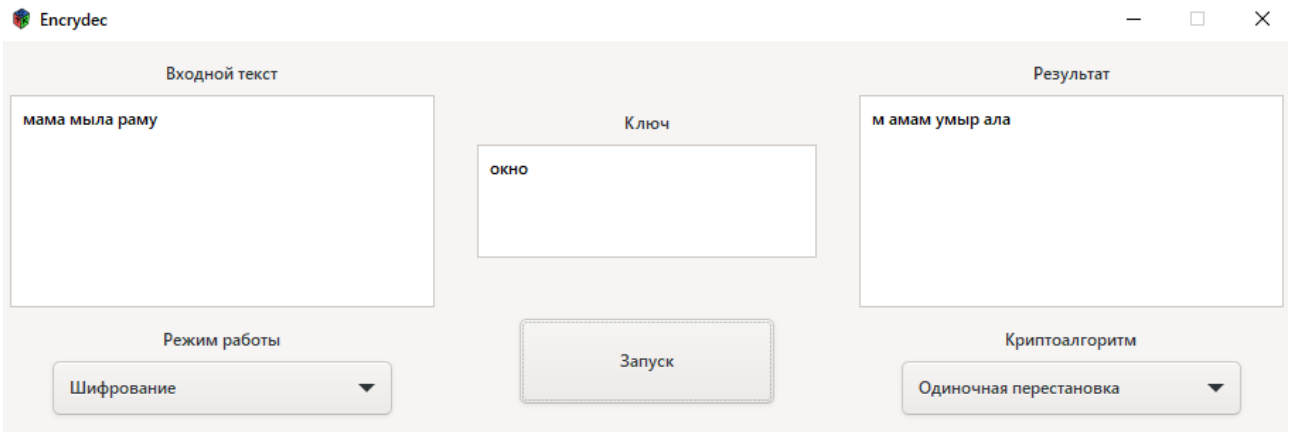

Рисунок 5 — Пример шифрования методом одиночной перестановки

Рисунок 6 — Пример расшифровки методом одиночной перестановки

Шифр Цезаря. Реализация и примеры работы

Алгоритм использует ключ для сдвига символов сообщения на key позиций в алфавите. Сдвиг работает с переполнением.

На рисунке 7 представлен скриншот фрагмента кода, реализующего описанный выше алгоритм.

```
private string Do(string message, int key)
{
    var fullAlfabet :string = Alphabet + Alphabet.ToLower();
    var letterQty :int = fullAlfabet.Length;
    var retVal = "";
    for (int i = 0; i < message.Length; i++)
    {
        var c :char = message[i];
        var index :int = fullAlfabet.IndexOf(c);
        if (index < 0)
        {
            retVal += c.ToString();
        }
        else
        {
            var codeIndex :int = (letterQty + index + key) % letterQty;
            retVal += fullAlfabet[codeIndex];
        }
    }
```

Рисунок 7 — Фрагмент кода реализации шифра Цезаря

Ключ для расшифровки используется в обратном виде.

Примеры работы программы на шифрование и расшифровку представлены на рисунках 8 и 9 соответственно.

Рисунок 8 — Пример шифрования методом Цезаря

Рисунок 9 — Пример расшифровки методом Цезаря

Шифр Виженера. Реализация и примеры работы

Алгоритм основывается на вычислении индекса зашифрованного символа в сообщении со сдвигом на позицию символа в алфавите в соответствии с изначальной позицией шифруемого символа

На рисунке 10 представлен скриншот фрагмента кода, реализующего описанный выше алгоритм.

```
private string Do(string message, string key, bool encrypting)
{
    key = key.ToUpper();
    var retValue = "";
    message = message.ToUpper();
    for (int i = 0; i < message.Length; i++)
    {
        var alphabetIndex :int = Alphabet.IndexOf(message[i]);
        var codeIndex :int = Alphabet.IndexOf(key[i%key.Length]);

        retValue += message[i]==' '? ' '
            : encrypting
            ? Alphabet[(alphabetIndex + codeIndex) % Alphabet.Length].ToString().ToLower()
            : alphabetIndex < codeIndex
                ? Alphabet[alphabetIndex - codeIndex + Alphabet.Length].ToString().ToLower()
                : Alphabet[(alphabetIndex - codeIndex) % Alphabet.Length].ToString().ToLower();

    }

    return retValue;
}
```

Рисунок 10 —Фрагмент кода реализации шифра Виженера

Расшифровка осуществляется аналогичным методу Цезаря образом, т.е. вычитанием. Алгоритм так же учитывает переполнение алфавита.

Примеры работы программы на шифрование и расшифровку представлены на рисунках 11 и 12 соответственно.

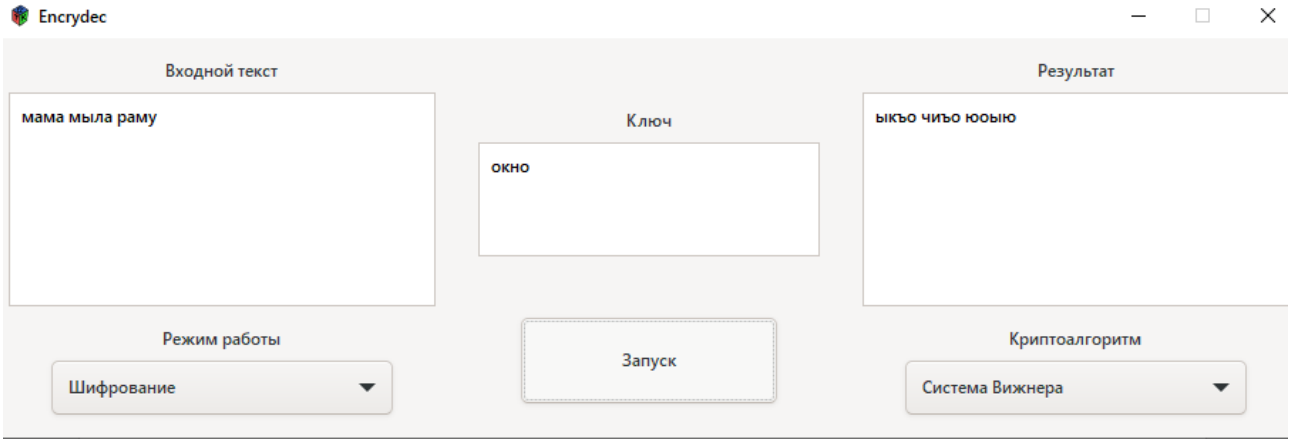

Рисунок 11 —Пример шифрования методом Виженера

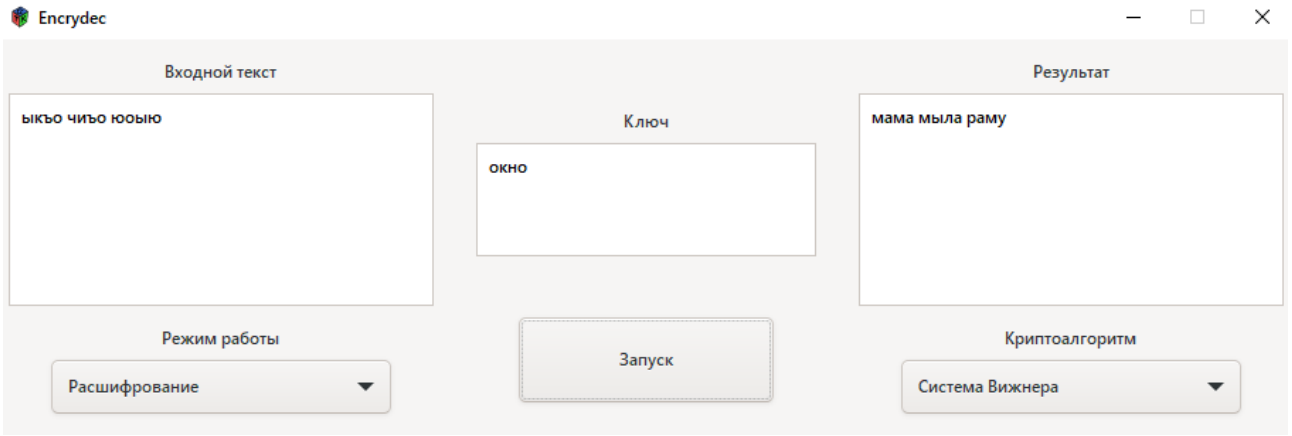

Рисунок 12 —Пример расшифровки методом Виженера

ЗАКЛЮЧЕНИЕ

В результате проделанной работы было разработано приложение, демонстрирующее процесс шифрования и расшифрования сообщений. Были реализованы следующие шифры: шифр Цезаря, шифр Виженера и одиночная перестановка. Интерфейс приложения был разработан с использованием библиотеки GTK#.