Федеральное государственное бюджетное образовательное

учреждение высшего образования

«Национальный исследовательский университет «МЭИ»

|  |  |
| --- | --- |
| Институт | ИВТИ |
| Кафедра | ПМИИ |

Дисциплина: «Защита данных»

Отчет по расчетному заданию

«Программная реализация шифра Рабина»

Вариант №39

Выполнил студент группы А-13а-19

Самсонова М. А.

Преподаватель: Хорев П. Б.

2023г.

Оглавление

[Введение 3](#_Toc134929428)

[1. Цель расчетного задания 3](#_Toc134929429)

[2. Задачи расчетного задания 3](#_Toc134929430)

[3. Требования к расчетному заданию 3](#_Toc134929431)

[Глава 1 Описание алгоритма 4](#_Toc134929432)

[1. Алгоритм шифра Рабина 4](#_Toc134929433)

[1.1. Генерация пары ключей 4](#_Toc134929434)

[1.2. Шифрование 4](#_Toc134929435)

[1.3. Расшифрование 4](#_Toc134929436)

[Глава 2 Результаты проектирования 6](#_Toc134929437)

[1. Пользовательский интерфейс 6](#_Toc134929438)

[2. Программная реализация 14](#_Toc134929439)

[Глава 3 Тестирование разработанной программы 18](#_Toc134929440)

[Тестовый пример №1 18](#_Toc134929441)

[Тестовый пример №2 20](#_Toc134929442)

[Тестовый пример №3 24](#_Toc134929443)

[Заключение 29](#_Toc134929444)

[Список источников 30](#_Toc134929445)

[Исходный код программы 31](#_Toc134929446)

# Введение

## Цель расчетного задания

Целью расчетного задания является программная реализация шифра Рабина.

## Задачи расчетного задания

В процессе выполнения расчетного задания:

* Разрабатывается пользовательский интерфейс приложения;
* Реализовывается алгоритм шифрования и расшифрования текста с применением шифра Рабина;
* Интеграция алгоритма в пользовательский интерфейс;
* Тестирование и отладка программы;
* Подготовка отчёта по расчетному заданию.

## Требования к расчетному заданию

Разрабатываемый программный продукт должен поддерживать:

* возможность шифрования/расшифрования как выбираемых файлов любого типа, так и вводимых текстовых сообщений на ключе, выводимом из парольной фразы с регулируемой пользователем минимальной длиной и сложностью;
* возможность при расшифровании проверять правильность введенной парольной фразы (например, путем добавления к данным перед их шифрованием сигнатуры – специальной строки символов – с проверкой ее наличия в расшифрованных данных и удалением из них в случае успешной проверки);
* возможность удаления шифруемого (расшифровываемого) файла после выполнения криптографической операции;
* возможность сохранения зашифрованного введенного пользователем текстового сообщения в файле с выбираемым именем.

# Описание алгоритма

Криптосистема Рабина — криптографическая система с открытым ключом, безопасность которой обеспечивается сложностью поиска квадратных корней в кольце остатков по модулю составного числа. Безопасность системы обеспечивается сложностью целочисленной факторизации, аналогично RSA. Существенным недостатком криптосистемы является то, что при расшифровании получается 4 текста, из которых лишь одно является исходным текстом сообщения.

## Алгоритм шифра Рабина

### Генерация пары ключей

Криптосистема Рабина имеет открытый ключ и два закрытых ключа. Открытый ключ используется для шифрования текста сообщения. Закрытые ключи используются для расшифрования шифротекста сообщения.

Закрытые ключи (p, q) выбираются следующим образом:

* Числа p и q являются простыми
* Числа p и q большие
* Выполняется условия p = 3 (mod 4) и q = 3 (mod 4)

Открытый ключ n получается из закрытых n = p \* q

### Шифрование

Шифротекст c сообщения получается с использованием открытого ключа шифрования

### Расшифрование

Для расшифрования шифротекста сообщения используются закрытые ключи p, q.

Алгоритм расшифрования:

1. Вычисление коэффициентов и
2. Вычисление коэффициентов Безу чисел p и q -
3. Вычисление 4 текстов
4. Выбор из исходного текста сообщения

# Результаты проектирования

## Пользовательский интерфейс

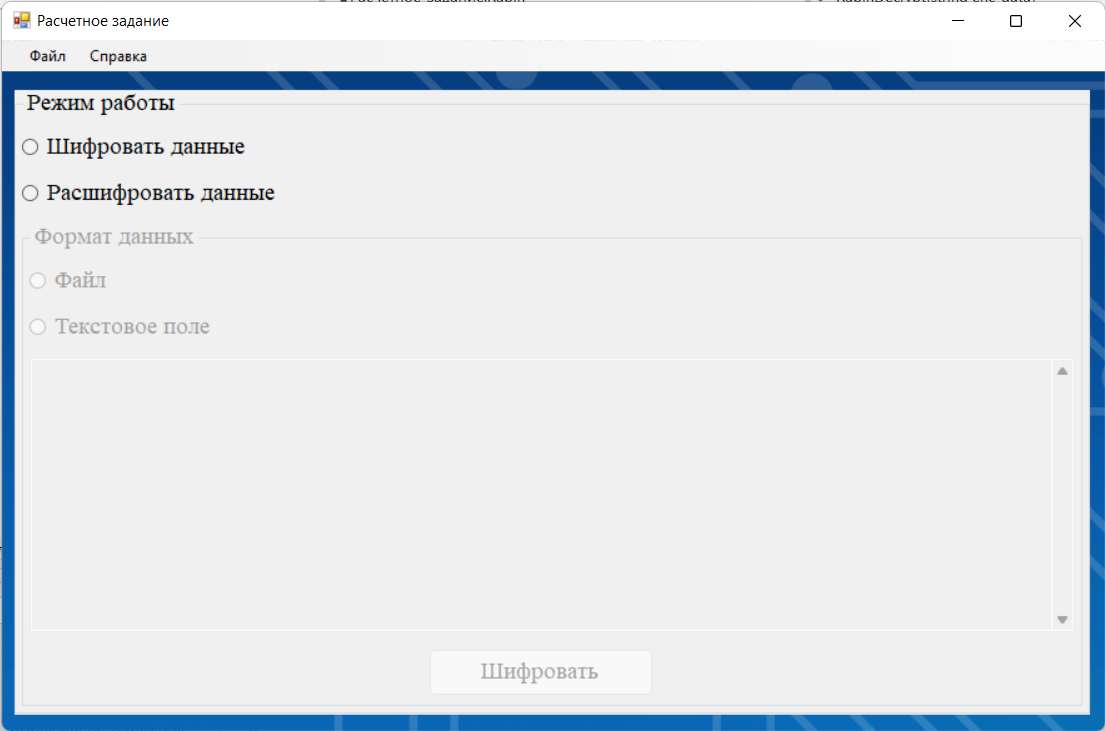


Рисунок 2‑1 Главное окно программы

Меню главного окна содержит разделы «Файл» и «Справка».

Раздел меню «Файл» содержит пункты «Задать параметры парольной фразы» и «Выход» (Рисунок 2.2).

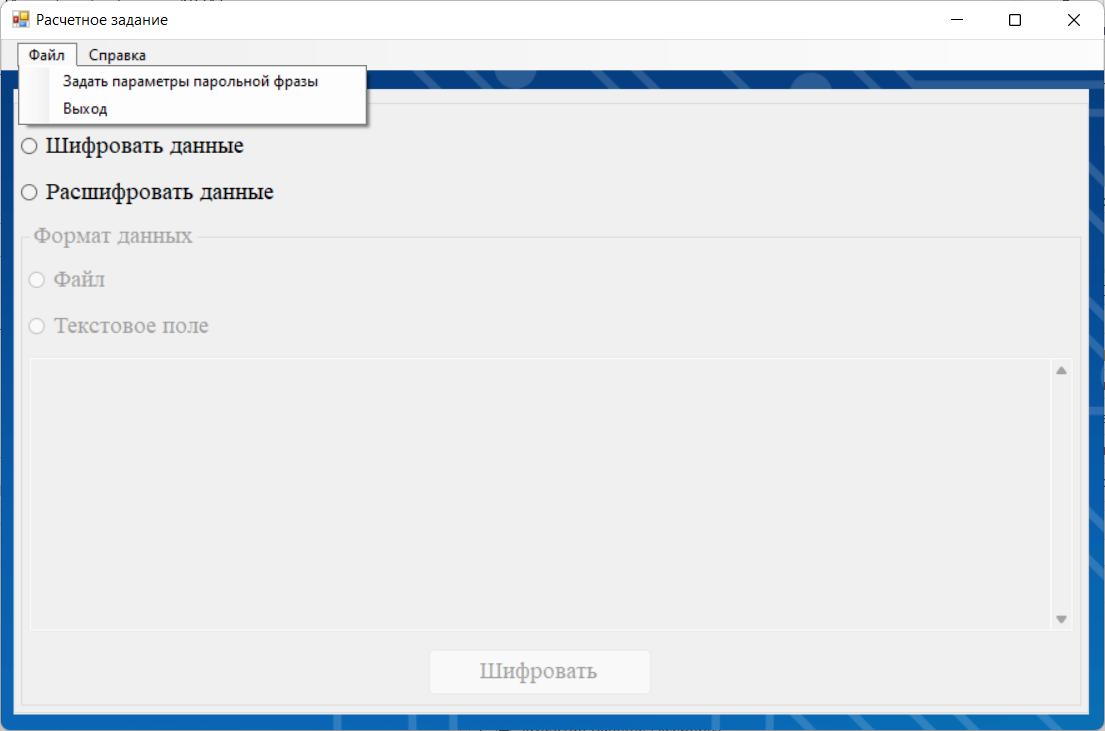


Рисунок 2‑2 Меню главного окна программы

При нажатии на пункт «Задать параметры парольной фразы» открывается окно «Параметры парольной фразы» (Рисунок 2.3).

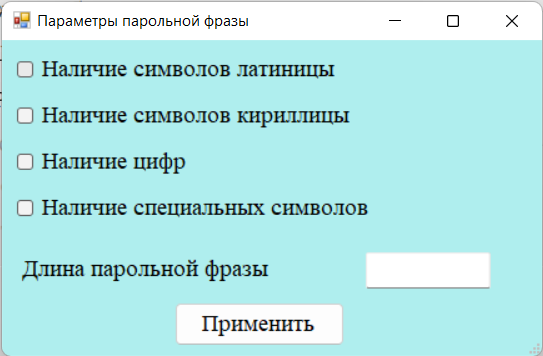


Рисунок 2‑3 Окно «Параметры парольной фразы»

При нажатии на пункт «Выход» происходит закрытие приложения.

Раздел меню «Справка» позволяет ознакомиться со сведениями программы («О программе») (Рисунок 2.4).

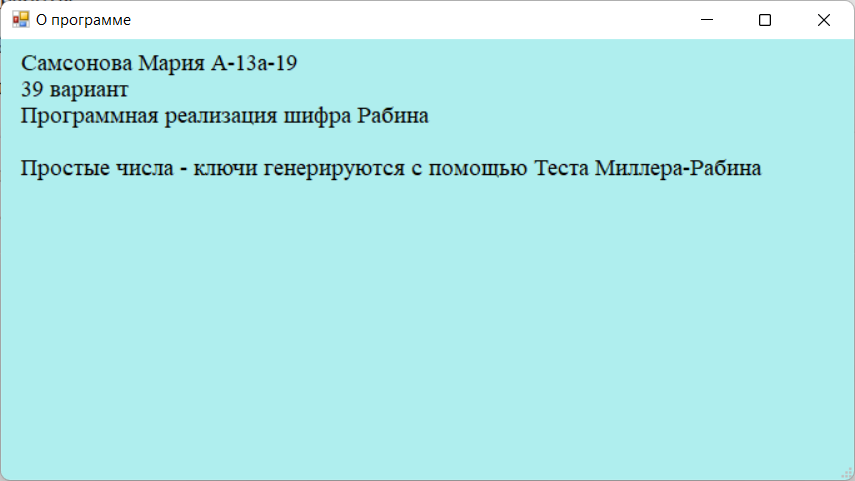


Рисунок 2‑4 Окно "О программе"

При активации одного из пунктов «Режим работы» главного окна «Шифровать данные» или «Расшифровать данные» происходит «активация» пунктов «Формат данных» (Рисунок 2.5) (Рисунок 2.6).

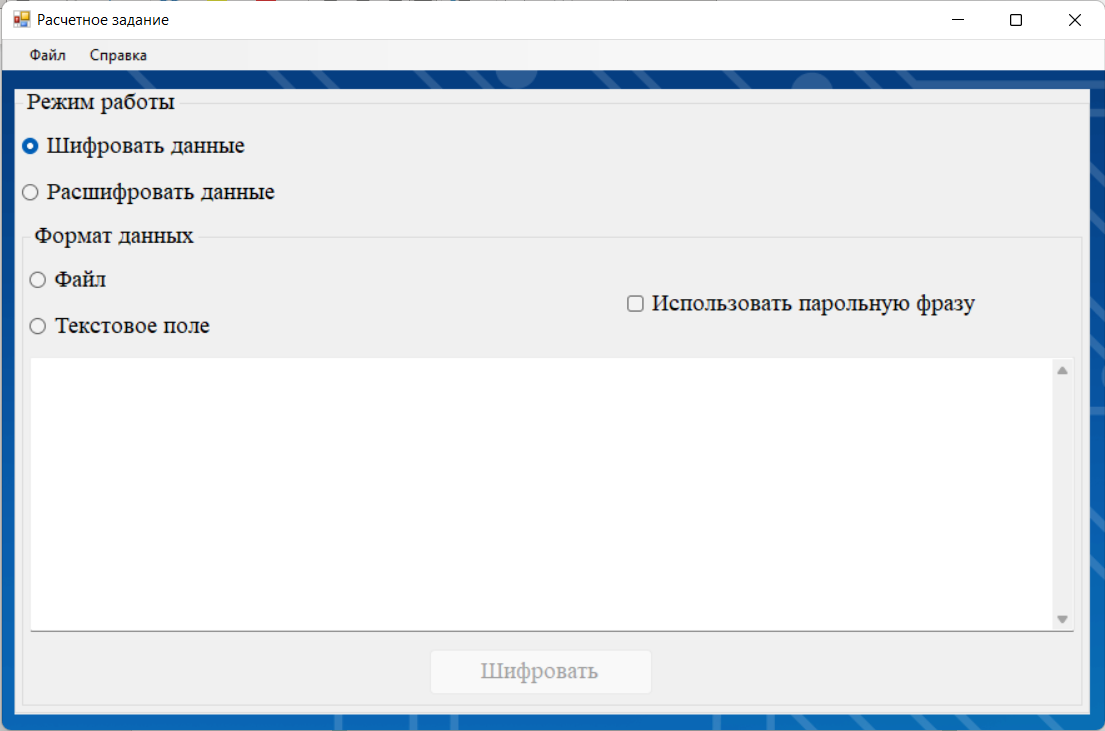


Рисунок 2‑5 «активация» пунктов «Формат данных»

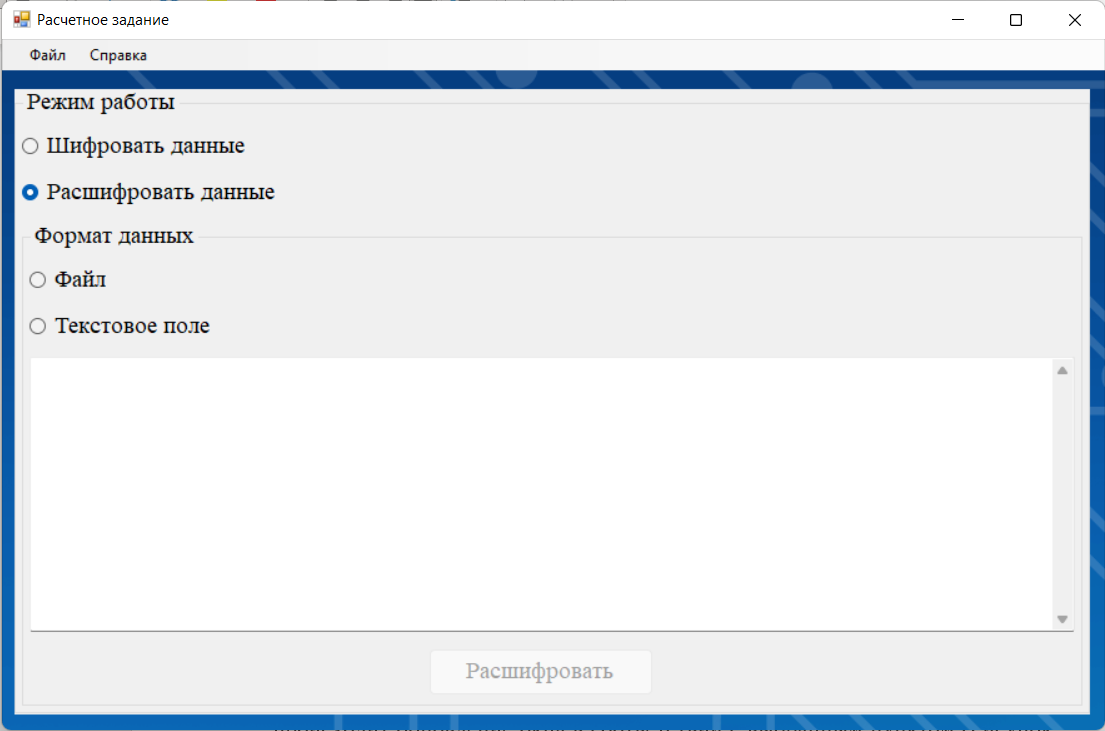


Рисунок 2‑6 «активация» пунктов «Формат данных»

При выборе одного из пунктов «Формат данных» происходит «активация» кнопки (Рисунок 2.7).

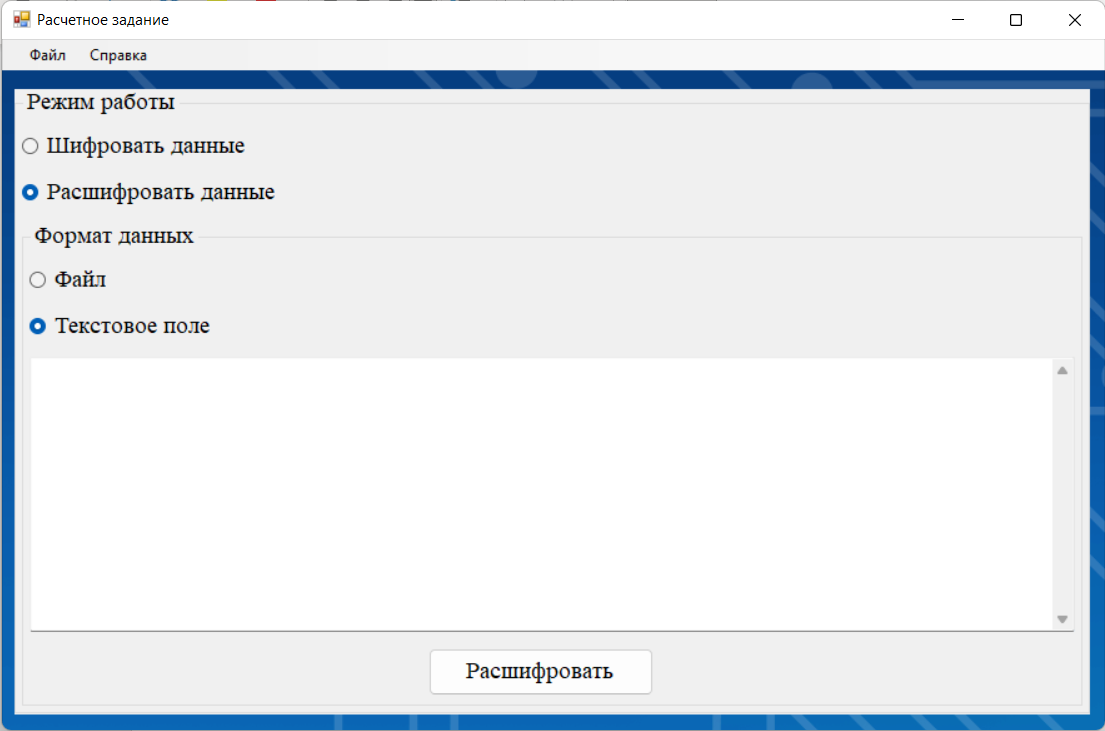


Рисунок 2‑7 «активация» кнопки

При выборе пункта «Файл» происходит открытие окна выбора файла (Рисунок 2.8).

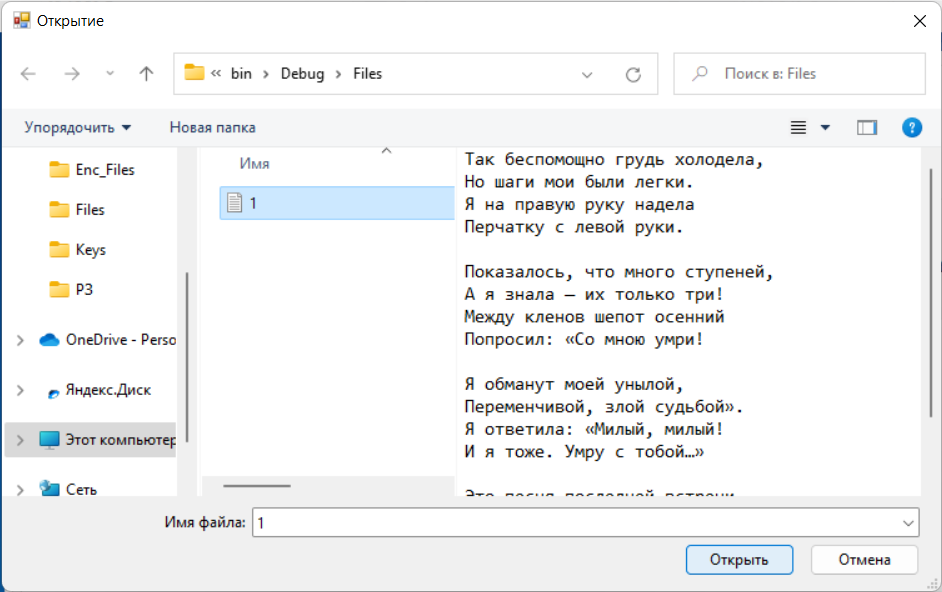


Рисунок 2‑8 Окно «Открытие»

При нажатии в Главном окне на чекбокс «Использовать парольную фразу» происходит открытие окна «Парольная фраза»

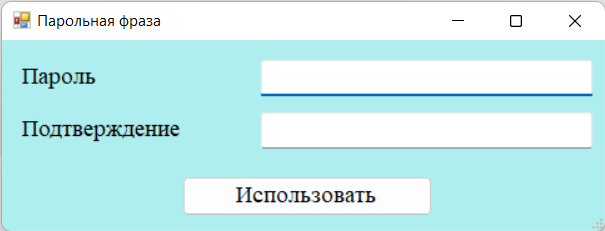


Рисунок 2‑9 Окно «Парольная фраза»

При нажатии в Главном окне на кнопку «Шифровать» или «Расшифровать» происходит последовательное открытие окон: окна оповещения (Рисунок 2.10) (Рисунок 2.11), окно сохранения или выбора контейнера ключей (Рисунок 2.12) (Рисунок 2.13), окно результата (Рисунок 2.14).

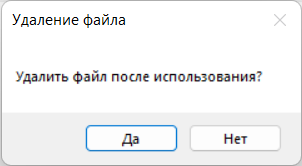


Рисунок 2‑10 Окно оповещения об удалении файла

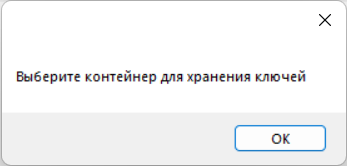


Рисунок 2‑11 Окно оповещения о выборе контейнера ключей

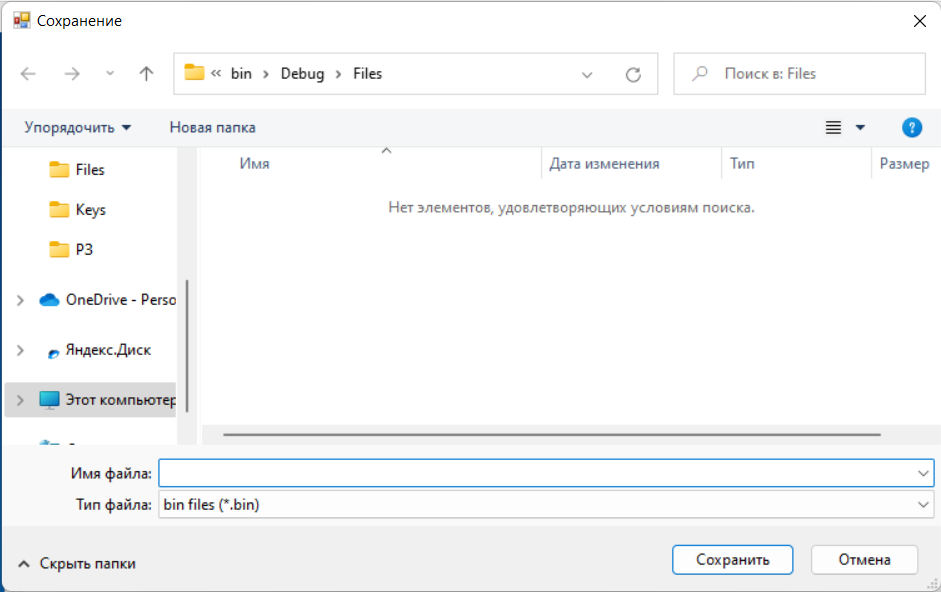


Рисунок 2‑12 Окно сохранения контейнера ключей

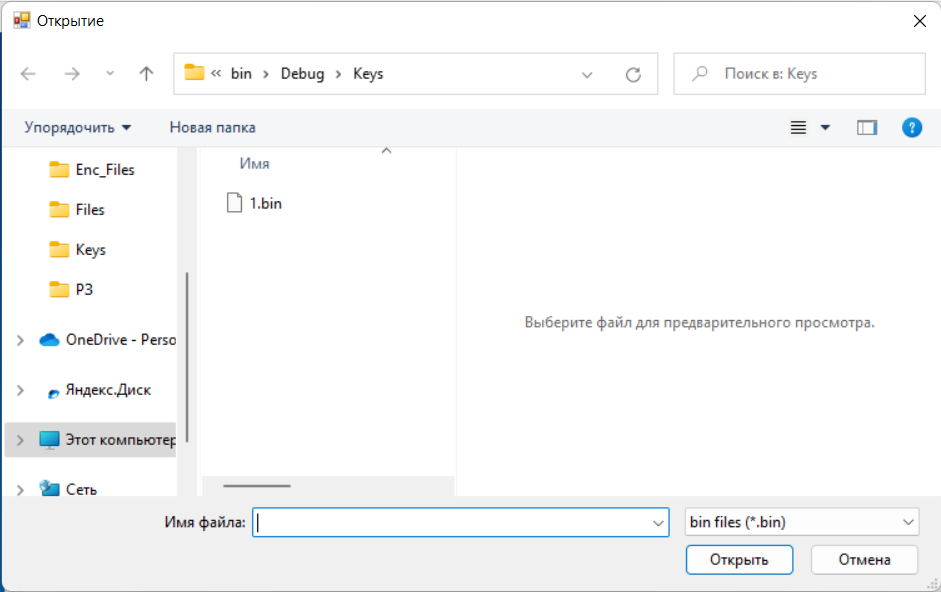


Рисунок 2‑13 Окно выбора контейнера ключей

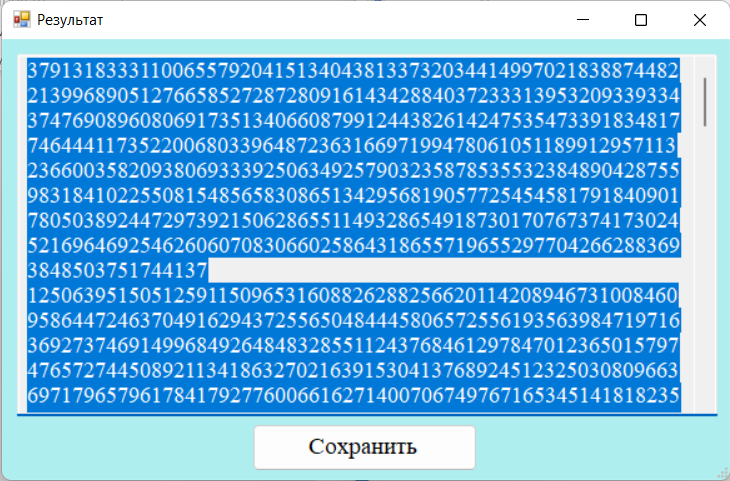


Рисунок 2‑14 Окно «Результат»

При нажатии в окне «Результат» на кнопку сохранить происходит открытие окна «Сохранить» для сохранения полученной информации в файл (Рисунок 2.15).

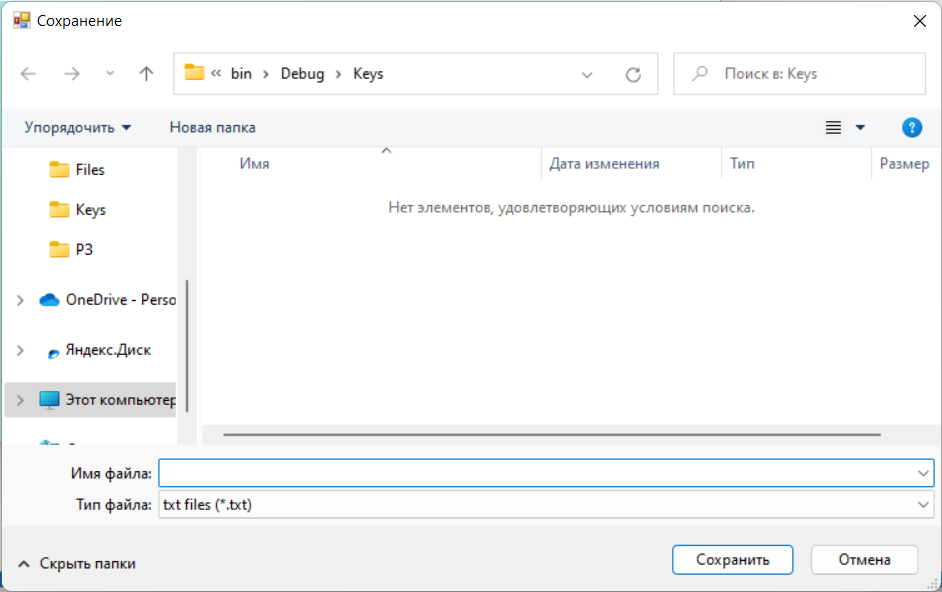


Рисунок 2‑15 Окно сохранения результата в файл

Если шифротекст использует ключи заданные с помощью парольной фразы перед выводом окна «Результат» происходит открытие окна «Проверка парольной фразы».

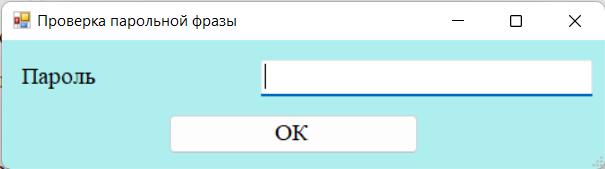


Рисунок 2‑16 Окно «Проверка парольной фразы»

При возникновении ошибок программа сигнализирует об этом пользователю (Рисунок 2.17 – Рисунок 2.28).

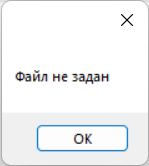


Рисунок 2‑17 Ошибка «Файл не задан»

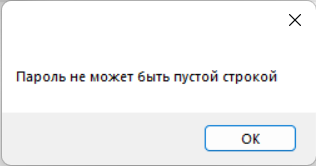


Рисунок 2‑18 Ошибка «Пароль не может быть пустой строкой»

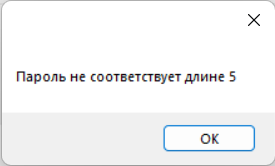


Рисунок 2‑19 Ошибка «Пароль не соответствует длине 5»

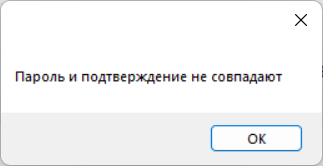


Рисунок 2‑20 Ошибка «Пароль и подтверждение не совпадают»

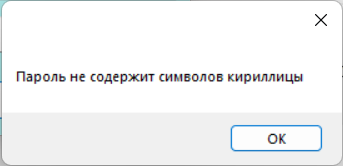


Рисунок 2‑21 Ошибка «Пароль не содержит символов кириллицы»

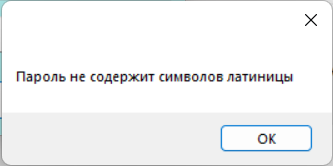


Рисунок 2‑22 Ошибка «Пароль не содержит символов латиницы»

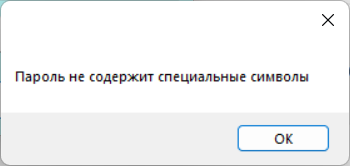


Рисунок 2‑23 Ошибка «Пароль не содержит специальные символы»

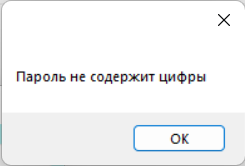


Рисунок 2‑24 Ошибка «Пароль не содержит цифры»

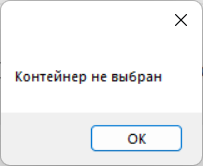


Рисунок 2‑25 Ошибка «Контейнер не выбран»

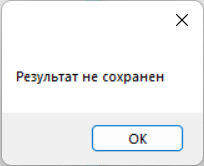


Рисунок 2‑26 Ошибка «Результат не сохранен»

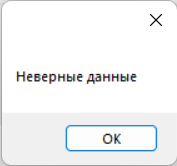


Рисунок 2‑27 Ошибка «Неверные данные»

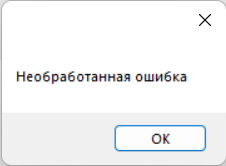


Рисунок 2‑28 Ошибка «Необработанная ошибка»

## Программная реализация

* Класс Rabin.cs
  + private BigInteger p – закрытый ключ шифра Рабина
  + private BigInteger q – закрытый ключ шифра Рабина
  + public BigInteger n – открытый ключ шифра Рабина
  + public Encoding enc\_utf8 = Encoding.UTF8 – кодировка UTF8
  + public Encoding enc\_un = Encoding.Unicode – кодировка Unicode
  + private TripleDESCryptoServiceProvider DES3 – криптопровайдер алгоритма 3DES для шифрования закрытых ключей в контейнере
  + private byte[] key – ключ для алгоритма 3DES
  + public byte[] DES3\_Enc(string data) – шифрование данных по алгоритму 3DES
  + public byte[] DES3\_Dec(byte[] enc\_data, int sizekey) – расшифрование данных по алгоритму 3DES
  + private bool RabinMillerTest(BigInteger num, int k) – тест Миллера-Рабина для проверки на простоту числа
  + private (BigInteger, BigInteger) EuclideanAlgorithm() – расширенный алгоритм Евклида для нахождения коэффициентов Безу
  + public Rabin(int p\_keySize = 1024, int q\_keySize = 1024) – конструктор класса для создания пары ключей
  + public Rabin(string filename, byte[] newkey) – конструктор класса для создания пары ключей
  + public void SaveKey(byte[] newkey) – сохранение ключей в файле
  + public string RabinEncrypt(byte[] data) – шифрование по алгоритму Рабина
  + public (byte[],byte[],byte[],byte[]) RabinDecrypt(string enc\_data) – расшифрование по алгоритму Рабина
* Класс Form1.cs
  + public bool Is\_Kir = false – параметр парольной фразы
  + public bool Is\_Lat = false – параметр парольной фразы
  + public bool Is\_Num = false – параметр парольной фразы
  + public bool Is\_Spec = false – параметр парольной фразы
  + public int Pass\_Len = 5 – параметр парольной фразы
  + Encoding enc\_cur – текущая кодировка файла
  + Encoding enc\_un = Encoding.Unicode – кодировка Unicode
  + public string password = "" – парольная фраза
  + public string message = "" – сообщение или шифротекст сообщения
  + public string cor\_password = "" – проверяемая парольная фраза
  + private bool Action\_Type – тип действия (true - шифровать false – расшифровывать)
  + private bool Format\_Info – тип информации (true - файл false - текстовое поле)
  + private OpenFileDialog ofd – открытие формы диалога
  + private string FilePath = "" – путь выбранного файла
  + private byte[] key – ключ для алгоритма 3DES
  + public void ClearForm() – очистка кода
  + public MainForm() – создание формы
  + private void оПрограммеToolStripMenuItem\_Click(object sender, EventArgs e) – открытие формы «О программе»
  + private void MainForm\_Shown(object sender, EventArgs e) - при первой загрузке окна (начальные данные)
  + private void выходToolStripMenuItem\_Click(object sender, EventArgs e) - закрытие программы при нажатии на пункт меню
  + private void задатьОграниченияПарольнойФразыToolStripMenuItem\_Click(object sender, EventArgs e) - настройка парольной фразы при нажатии на пункт меню
  + private void CB\_Password\_CheckedChanged(object sender, EventArgs e) - Чекбокс использования парольной фразы
  + private void RB\_Enc\_CheckedChanged(object sender, EventArgs e) - Радиобатон шифрования
  + private void RB\_Dec\_CheckedChanged(object sender, EventArgs e) - Радиобатон расшифрования
  + private void RB\_File\_CheckedChanged(object sender, EventArgs e) - Радиобатон использования файла
  + private void RB\_TextBox\_CheckedChanged(object sender, EventArgs e) - Радиобатон использования текстового поля
  + private void But\_Action\_Click(object sender, EventArgs e) – нажатие на кнопку шифрования или расшифрования
* Класс FormAbout.cs
* Класс FormPassword.cs
  + private void But\_Action\_Click(object sender, EventArgs e) – нажатие на кнопку Использовать
* Класс FormPasswordCorrect.cs
  + private void But\_Action\_Click(object sender, EventArgs e) - нажатие на кнопку ОК
* Класс FormResult.cs
  + private void But\_Save\_Click(object sender, EventArgs e) – нажатие на кнопку Сохранить
* Класс PasswordParams.cs
  + private void But\_Action\_Click(object sender, EventArgs e) – нажатие на кнопку Применить

# Тестирование разработанной программы

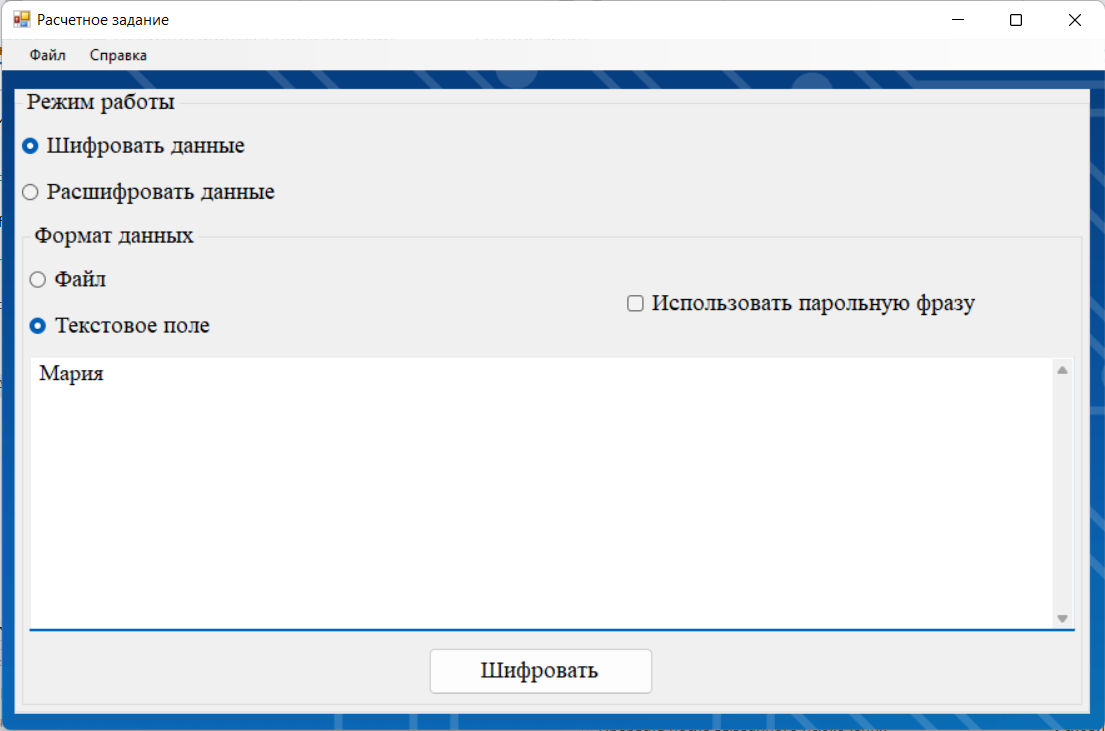
Для тестирования интерфейса приложения и алгоритма шифра Рабина опишем тестовые примеры и сравним их результаты с ожидаемыми данными.

## Тестовый пример №1

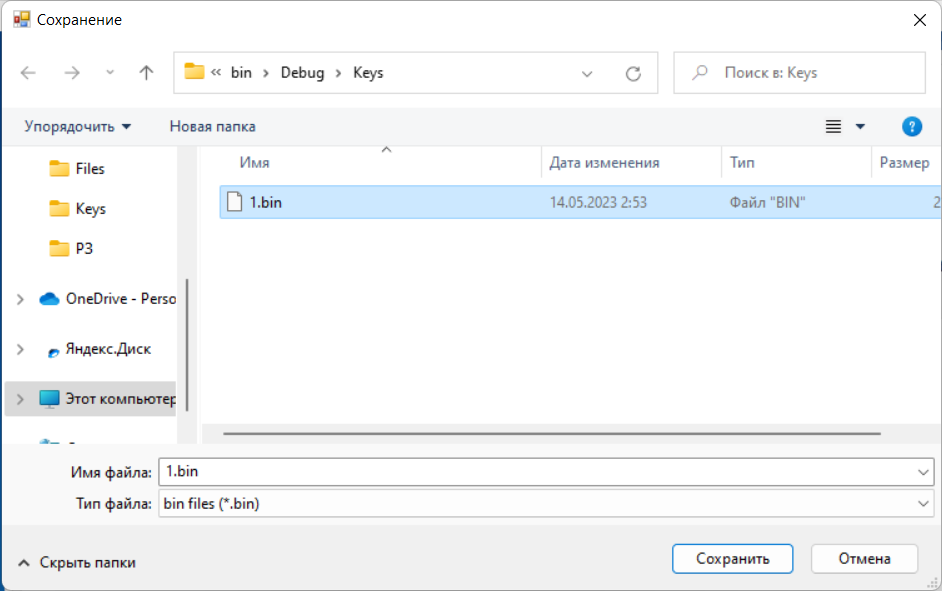
Использование текстового поля без использования парольной фразы без сохранения результата.

Шаги тестирования:

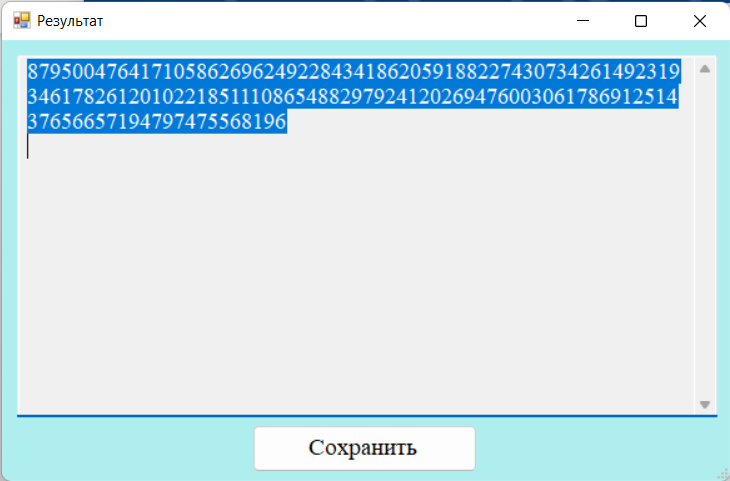
* Вписать данные в текстовое поле



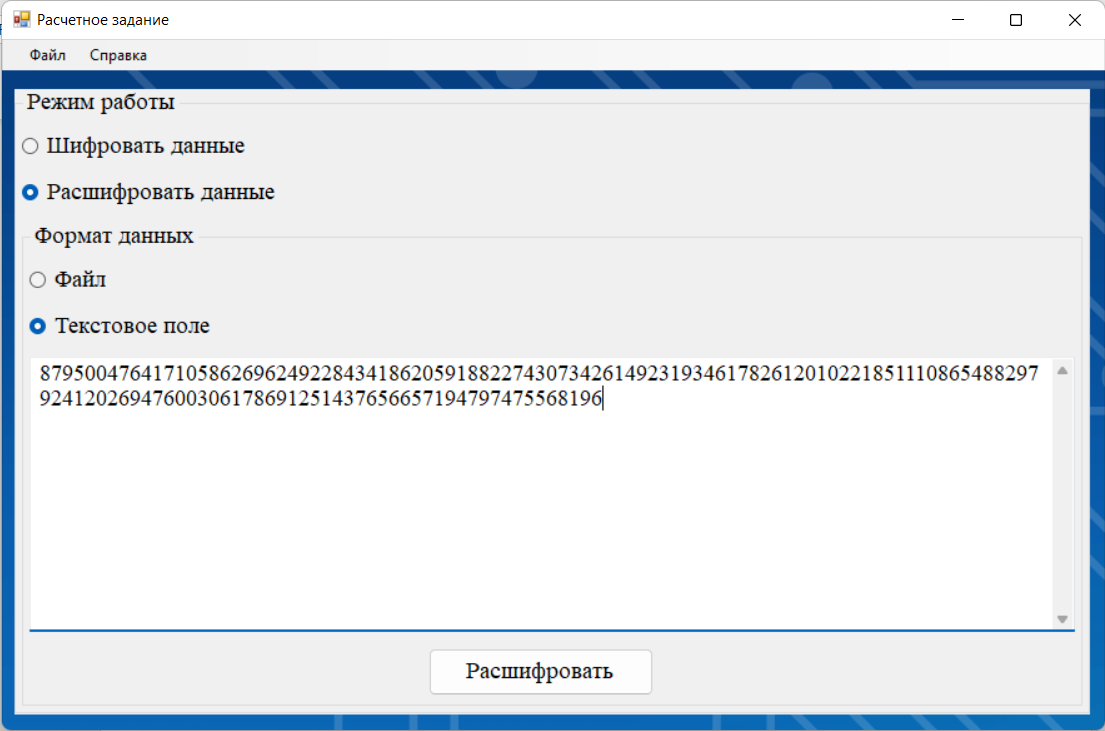
* Сохранить пару ключей в файл



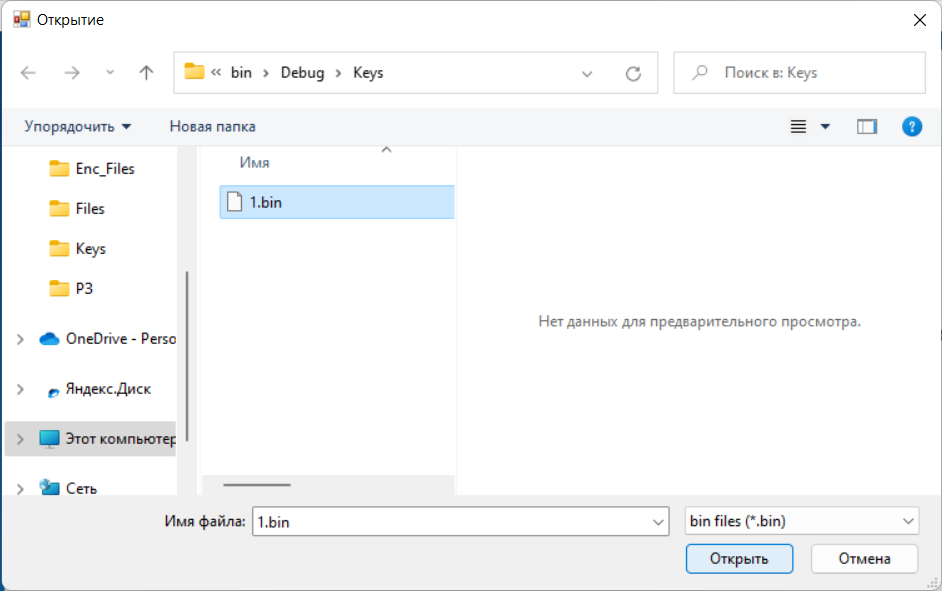
* Копирование результата



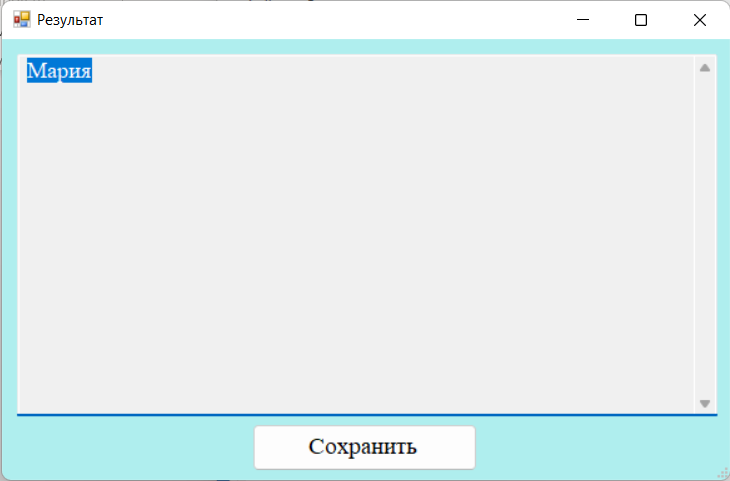
* Вставка результата в текстовое поле



* Выбор контейнера ключей



* Получение результата

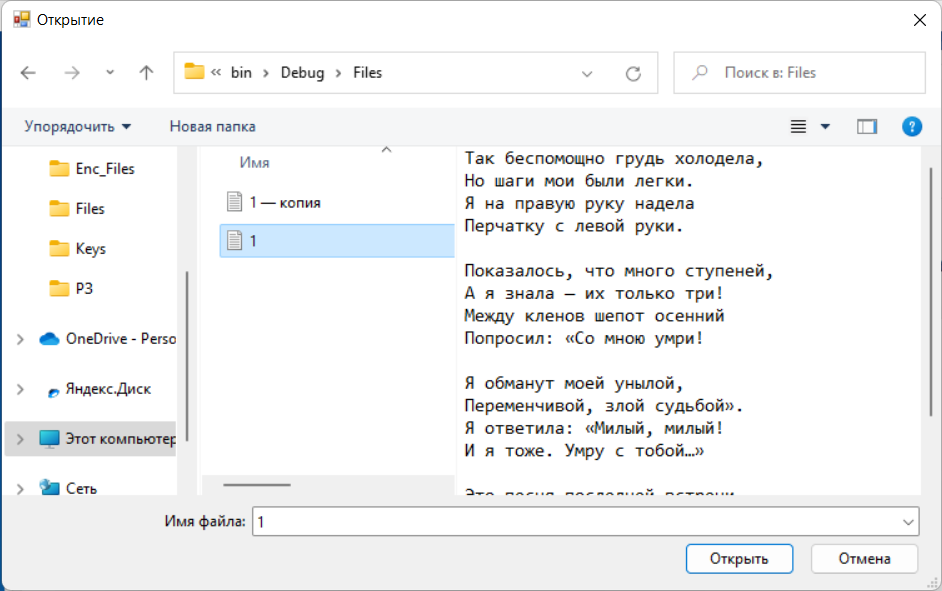


## Тестовый пример №2

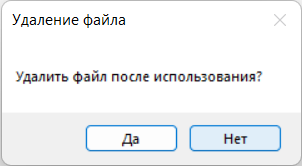
Использование файла без его удаления без использования парольной фразы с сохранением результата.

Шаги тестирования:

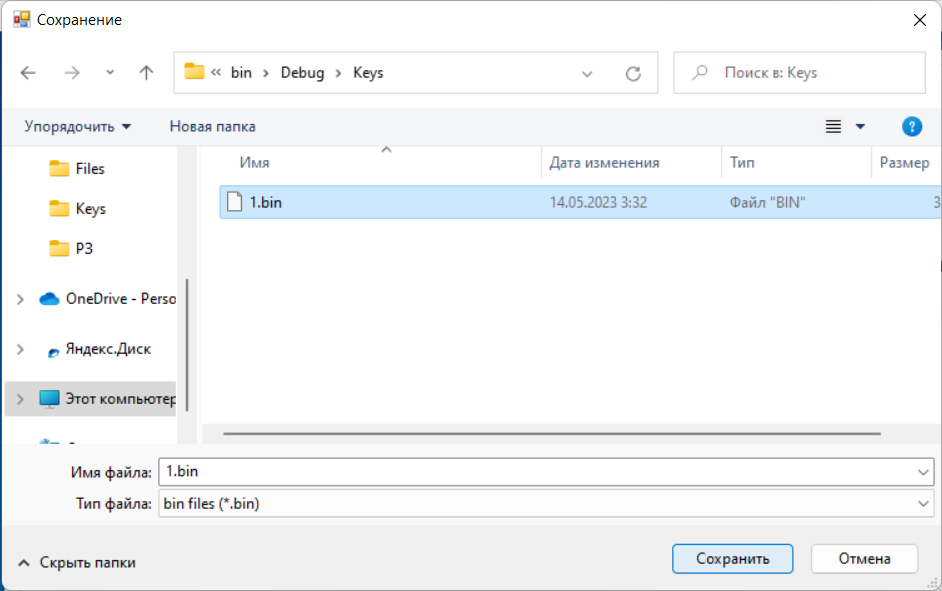
* Выбор файла



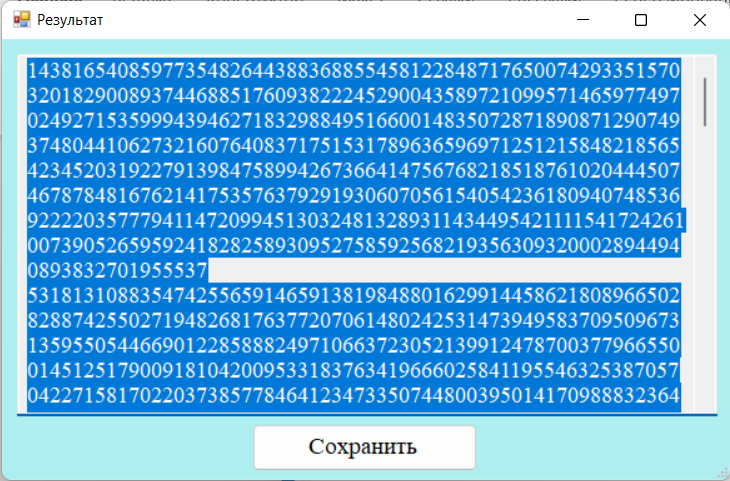
* Отказ от удаления файла



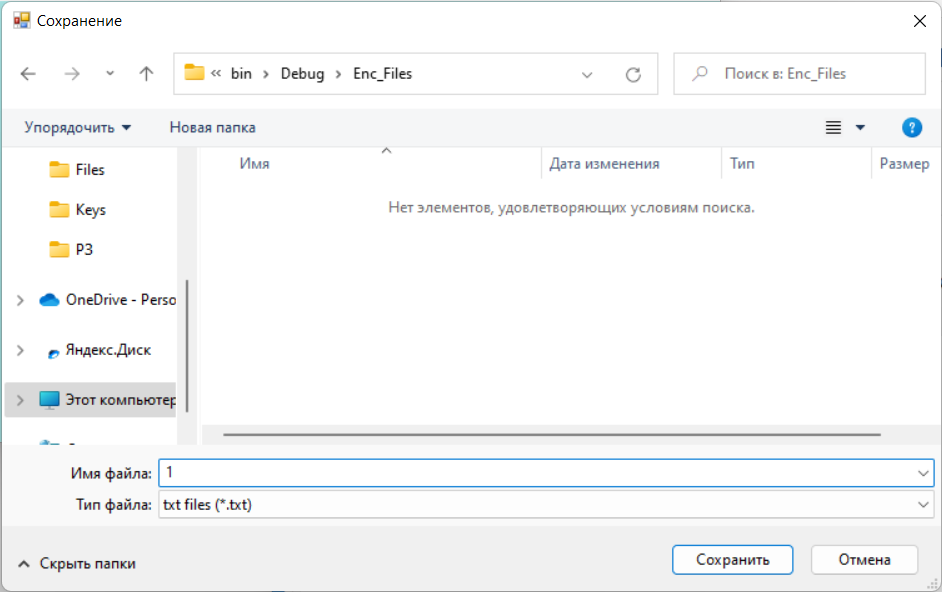
* Сохранение пары ключей в файл



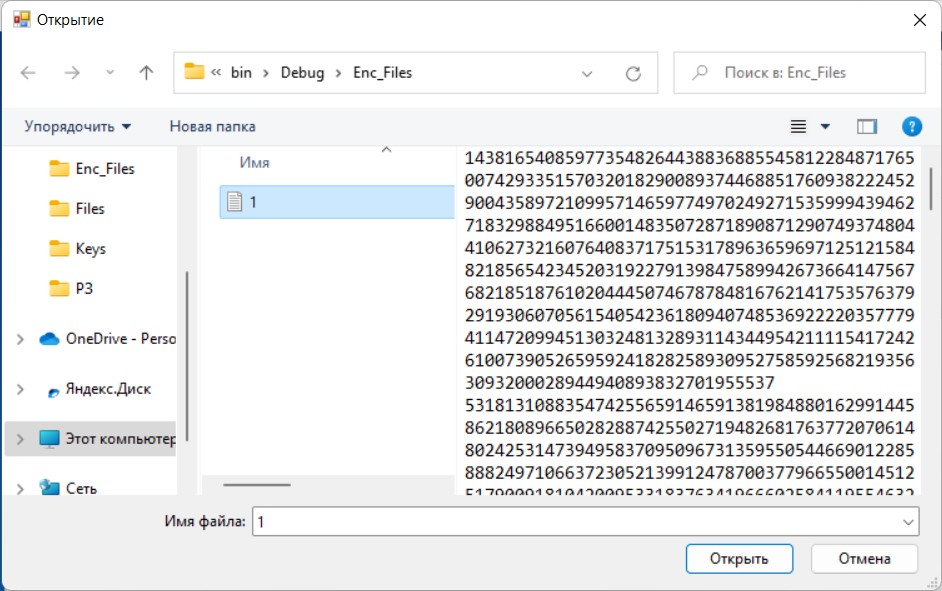
* Получение результата



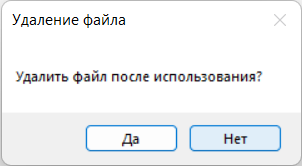
* Сохранение результата



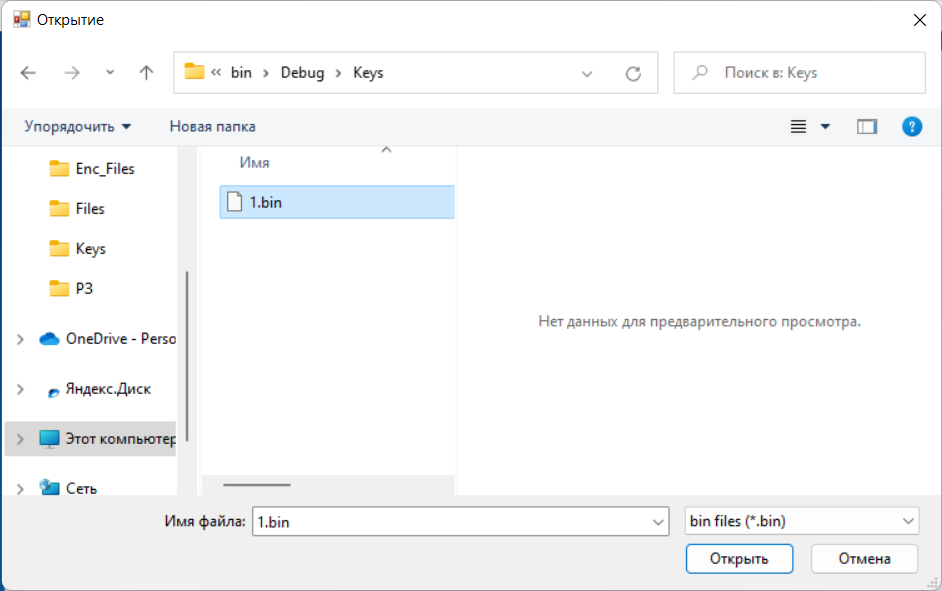
* Выбор файла



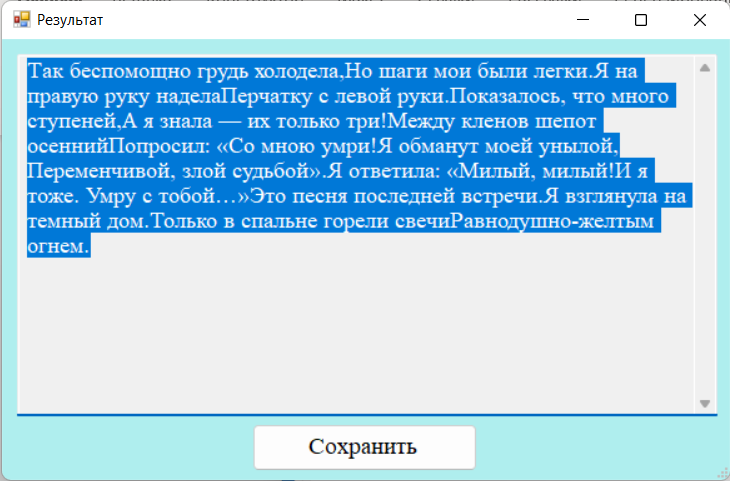
* Отказ от удаления



* Выбор пары ключей



* Получение результата

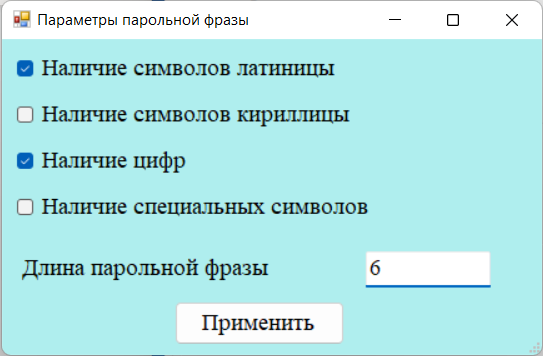


## Тестовый пример №3

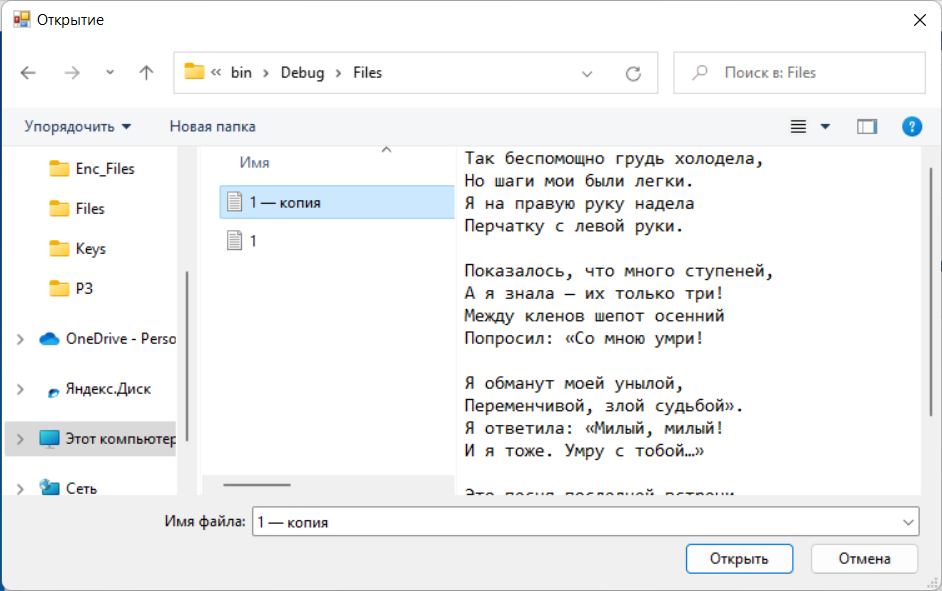
Использование файла с его удалением с использованием парольной фразы без сохранения результата.

Шаги тестирования:

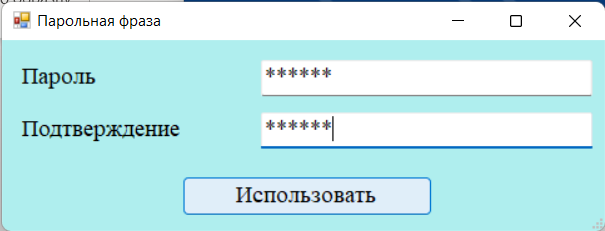
* Выбор параметров парольной фразы



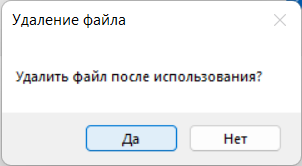
* Выбор файла



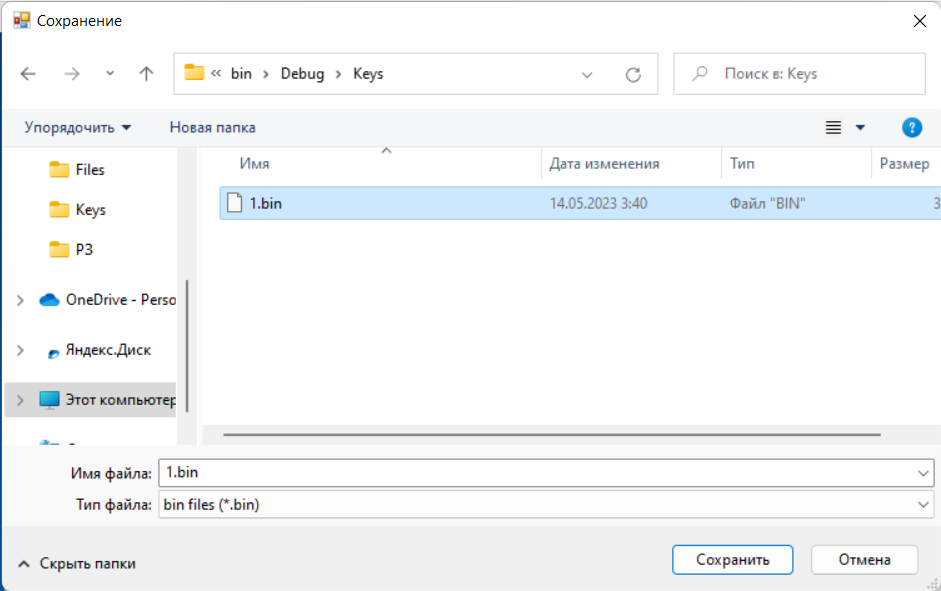
* Ввод парольной фразы



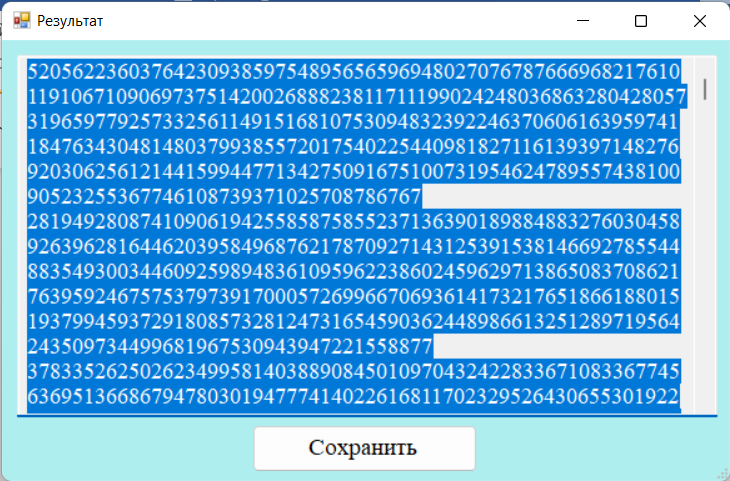
* Согласие с удалением файла



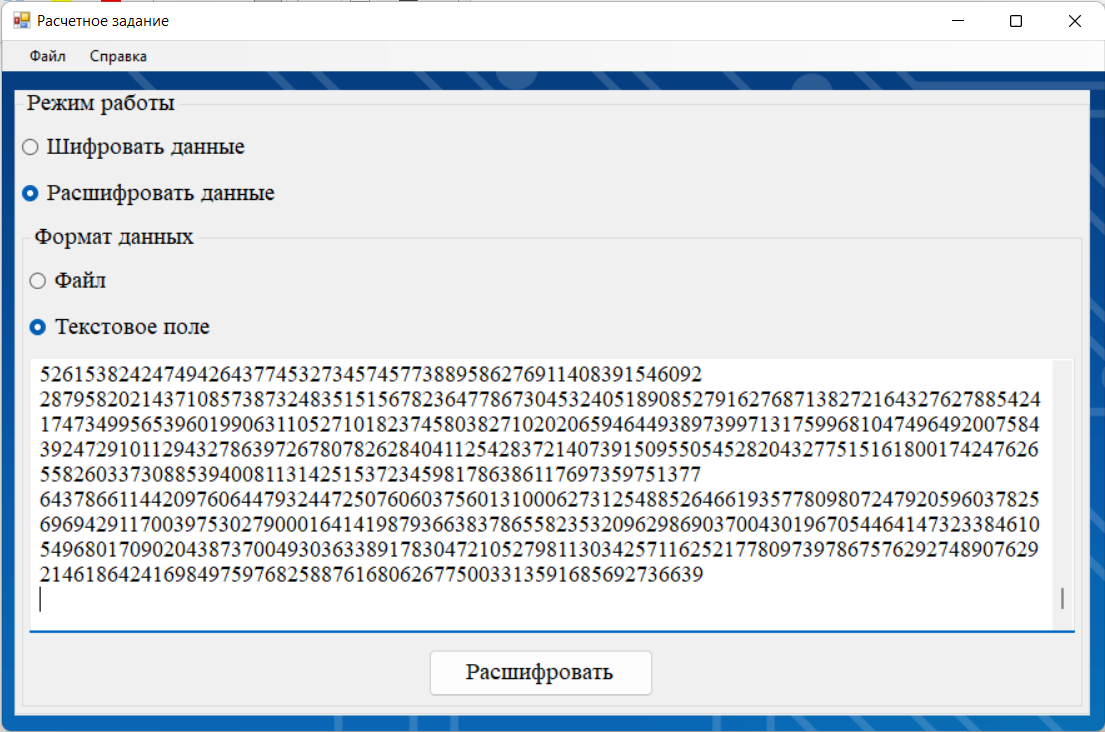
* Сохранение пары ключей в файл



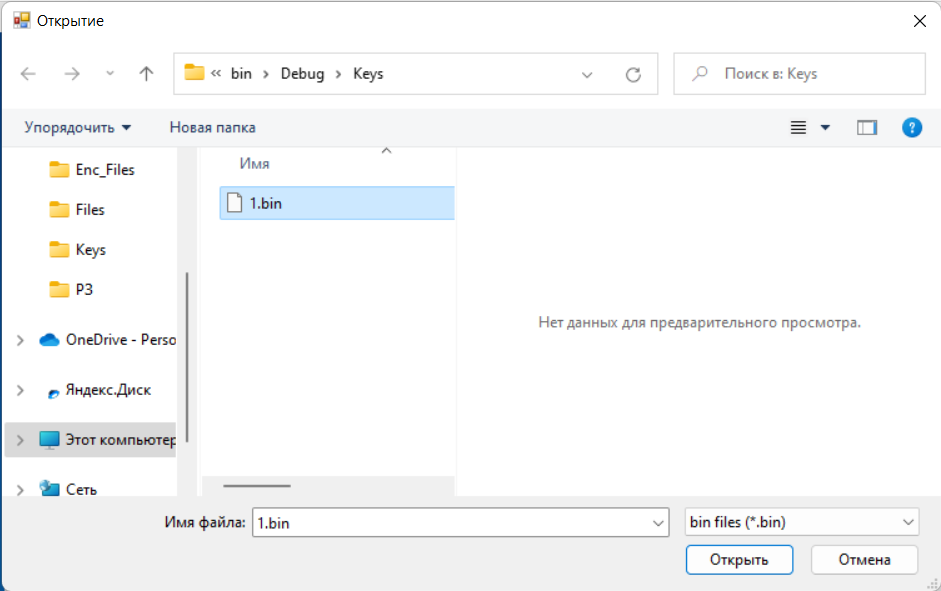
* Копирование результата



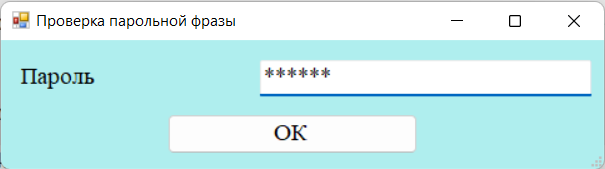
* Ввод в текстовое поле данных результата



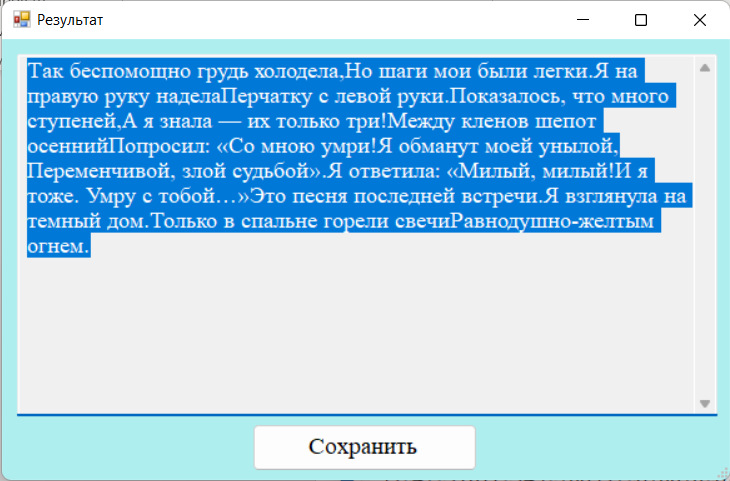
* Выбор контейнера ключей



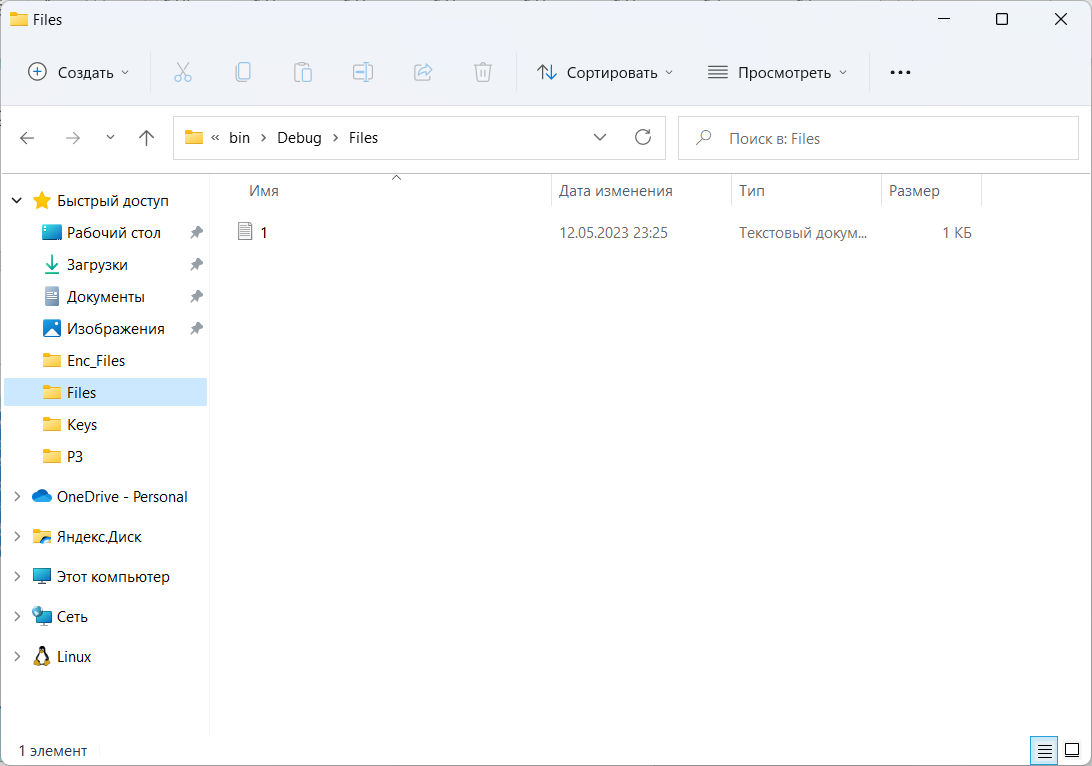
* Ввод парольной фразы



* Вывод результата



* Проверка отсутствия файла



# Заключение

В ходе выполнения расчетного задания по заданной теме:

* Изучен и реализован шифр Рабина
* Спроектирован и разработан интерфейс приложения с помощью языка C# и платформы Windows Forms
* Произведено тестирование приложения по тестовым примерам
* Составлен отчет по расчетному заданию

# Список источников

1. Rabin cryptosystem

URL: https://en.wikipedia.org/wiki/Rabin\_cryptosystem

1. Тест Миллера — Рабина

URL: https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D1%81%D1%82\_%D0%9C%D0%B8%D0%BB%D0%BB%D0%B5%D1%80%D0%B0\_%E2%80%94\_%D0%A0%D0%B0%D0%B1%D0%B8%D0%BD%D0%B0

1. Соотношение Безу

URL: https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D0%BE%D1%82%D0%BD%D0%BE%D1%88%D0%B5%D0%BD%D0%B8%D0%B5\_%D0%91%D0%B5%D0%B7%D1%83

# Исходный код программы

Rabin.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Numerics;

using System.Security.Cryptography;

using System.IO;

using System.Windows.Forms;

namespace Расчетное\_задание

{

class Rabin

{

private BigInteger p;

private BigInteger q;

public BigInteger n;

public Encoding enc\_utf8 = Encoding.UTF8;

public Encoding enc\_un = Encoding.Unicode;

//шифрование открытых ключей

private TripleDESCryptoServiceProvider DES3;

private byte[] key;

public byte[] DES3\_Enc(string data)

{

DES3 = new TripleDESCryptoServiceProvider();

DES3.Padding = PaddingMode.None;

DES3.Mode = CipherMode.CFB; // установка режима блочного шифрования

byte[] IV = new byte[8] { 0, 0, 0, 0, 0, 0, 0, 0 };

DES3.Key = key;

byte[] bdata = enc\_un.GetBytes(data);

ICryptoTransform encryptor = DES3.CreateEncryptor(DES3.Key, IV); // создание объекта шифрования

MemoryStream mems = new MemoryStream();

CryptoStream crypts = new CryptoStream(mems, encryptor, CryptoStreamMode.Write);

crypts.Write(bdata, 0, bdata.Length);

crypts.Close();

byte[] result = mems.ToArray();

mems.Close();

DES3.Clear();

return result;

}

public byte[] DES3\_Dec(byte[] enc\_data, int sizekey)

{

DES3 = new TripleDESCryptoServiceProvider();

DES3.Padding = PaddingMode.None;

DES3.Mode = CipherMode.CFB; // установка режима блочного шифрования

byte[] IV = new byte[8] { 0, 0, 0, 0, 0, 0, 0, 0 };

DES3.Key = key;

ICryptoTransform decryptor = DES3.CreateDecryptor(DES3.Key, IV);

MemoryStream mems = new MemoryStream(enc\_data);

CryptoStream crypts = new CryptoStream(mems, decryptor, CryptoStreamMode.Read);

byte[] result = new byte[sizekey];

crypts.Read(result, 0, result.Length);

crypts.Close();

mems.Close();

return result;

}

private bool RabinMillerTest(BigInteger num, int k)

{

if (num == 2 || num == 3 || num == 5 || num == 7)

return true;

if (num < 2 || num % 2 == 0)

return false;

BigInteger t = num - 1;

BigInteger s = 0;

while (t % 2 == 0)

{

t = t / 2;

s++;

}

//количество проверок

for (int i = 0; i < k; i++)

{

var rng = new RNGCryptoServiceProvider(); //криптографический генератор случайных чисел

byte[] bytes = new byte[num.ToByteArray().Length];

rng.GetBytes(bytes);

BigInteger a = new BigInteger(bytes);

while (a < 2 || a > num - 2)

{

rng.GetBytes(bytes);

a = new BigInteger(bytes);

}

BigInteger x = BigInteger.ModPow(a, t, num); //вычет

if (x == 1 || x == num - 1)

continue;

for (BigInteger r = 1; r < s; r++)

{

x = BigInteger.ModPow(x, 2, num);

if (x == 1)

return false;

if (x == num - 1)

break;

}

if (x != num - 1)

return false;

}

return true;

}

private (BigInteger, BigInteger) EuclideanAlgorithm()

{

BigInteger p\_ = p;

BigInteger q\_ = q;

BigInteger help = q\_;

BigInteger t = 1;

BigInteger s = 0;

BigInteger help1 = s;

while (q\_ != 0)

{

BigInteger koef = p\_ / q\_;

q\_ = p\_ - koef \* q\_;

p\_ = help;

help = q\_;

s = t - koef \* s;

t = help1;

help1 = s;

}

help1 = t;

if (q != 0)

t = (p\_ - t \* p) / q;

else

t = 0;

return (help1, t);

}

public Rabin(int p\_keySize = 1024, int q\_keySize = 1024)

{

var rng = new RNGCryptoServiceProvider(); //криптографический генератор случайных чисел

byte[] p\_bytes = new byte[p\_keySize];

rng.GetBytes(p\_bytes);

p = new BigInteger(p\_bytes);

while (!RabinMillerTest(p, 50) || p % 4 != 3)

{

p\_bytes = new byte[p\_keySize];

rng.GetBytes(p\_bytes);

p = new BigInteger(p\_bytes);

}

byte[] q\_bytes = new byte[q\_keySize];

rng.GetBytes(q\_bytes);

q = new BigInteger(q\_bytes);

while (!RabinMillerTest(q, 50) || q % 4 != 3)

{

q\_bytes = new byte[q\_keySize];

rng.GetBytes(q\_bytes);

q = new BigInteger(q\_bytes);

}

n = p \* q;

Console.WriteLine("Закрытые ключи");

Console.WriteLine(p);

Console.WriteLine(q);

Console.WriteLine("Открытый ключ");

Console.WriteLine(n);

}

public Rabin(string filename, byte[] newkey)

{

key = newkey;

using (StreamReader sr = new StreamReader(filename, enc\_utf8))

{

int p\_len\_enc = Convert.ToInt32(sr.ReadLine());

char[] p\_enc = new char[p\_len\_enc];

sr.Read(p\_enc, 0, p\_len\_enc);

sr.ReadLine();

int q\_len\_enc = Convert.ToInt32(sr.ReadLine());

char[] q\_enc = new char[q\_len\_enc];

sr.Read(q\_enc, 0, q\_len\_enc);

sr.ReadLine();

int p\_len = Convert.ToInt32(sr.ReadLine());

int q\_len = Convert.ToInt32(sr.ReadLine());

int n\_len = Convert.ToInt32(sr.ReadLine());

char[] n\_ = new char[n\_len];

sr.Read(n\_, 0, n\_len);

byte[] p\_ = DES3\_Dec(BigInteger.Parse(new string(p\_enc)).ToByteArray(), p\_len);

byte[] q\_ = DES3\_Dec(BigInteger.Parse(new string(q\_enc)).ToByteArray(), q\_len);

p = BigInteger.Parse(enc\_un.GetString(p\_));

q = BigInteger.Parse(enc\_un.GetString(q\_));

n = BigInteger.Parse(new string(n\_));

}

}

public void SaveKey(byte[] newkey)

{

MessageBox.Show("Выберите контейнер для хранения ключей");

SaveFileDialog sfd = new SaveFileDialog();

sfd.Filter = "bin files (\*.bin)|\*.bin";

sfd.RestoreDirectory = true;

if (sfd.ShowDialog() == DialogResult.Cancel)

{

MessageBox.Show("Контейнер не выбран");

return;

}

else

{

string namefile = sfd.FileName;

key = newkey;

string p\_ = p.ToString();

string strp = "";

while (p\_.Length % 8 != 0)

{

p\_ = '0' + p\_;

strp += '0';

}

string q\_ = q.ToString();

string strq = "";

while (q\_.Length % 8 != 0)

{

q\_ = '0' + q\_;

strq += '0';

}

byte[] enc\_p = DES3\_Enc(p\_);

byte[] enc\_q = DES3\_Enc(q\_);

using (StreamWriter sw = new StreamWriter(namefile, false, enc\_utf8))

{

sw.WriteLine(new BigInteger(enc\_p).ToString().Length);

sw.WriteLine(new BigInteger(enc\_p));

sw.WriteLine(new BigInteger(enc\_q).ToString().Length);

sw.WriteLine(new BigInteger(enc\_q));

sw.WriteLine(enc\_un.GetBytes(strp + p.ToString()).Length);

sw.WriteLine(enc\_un.GetBytes(strq + q.ToString()).Length);

sw.WriteLine(n.ToString().Length);

sw.WriteLine(n);

}

}

}

public string RabinEncrypt(byte[] data)

{

BigInteger int\_data = new BigInteger(data);

BigInteger enc\_data = BigInteger.ModPow(int\_data, 2, n);

Console.WriteLine("Данные для шифрования");

Console.WriteLine(int\_data);

Console.WriteLine("Данные для расшифрования");

Console.WriteLine(enc\_data);

return enc\_data.ToString();

}

public (byte[],byte[],byte[],byte[]) RabinDecrypt(string enc\_data)

{

BigInteger m\_p;

BigInteger m\_q;

BigInteger int\_enc\_data = BigInteger.Parse(enc\_data);

m\_p = BigInteger.ModPow(int\_enc\_data, ((p + 1) / 4), p);

m\_q = BigInteger.ModPow(int\_enc\_data, ((q + 1) / 4), q);

BigInteger y\_p;

BigInteger y\_q;

(y\_p, y\_q) = EuclideanAlgorithm();

BigInteger r1;

BigInteger r2;

BigInteger r3;

BigInteger r4;

r1 = (y\_p \* p \* m\_q + y\_q \* q \* m\_p) % n < 0 ? n + (y\_p \* p \* m\_q + y\_q \* q \* m\_p) % n : (y\_p \* p \* m\_q + y\_q \* q \* m\_p) % n;

r2 = n - r1;

r3 = (y\_p \* p \* m\_q - y\_q \* q \* m\_p) % n < 0 ? n + (y\_p \* p \* m\_q - y\_q \* q \* m\_p) % n : (y\_p \* p \* m\_q - y\_q \* q \* m\_p) % n;

r4 = n - r3;

//Console.WriteLine("Данные для расшифрования");

//Console.WriteLine(int\_enc\_data);

//Console.WriteLine("Коэффициенты");

//Console.WriteLine(m\_p);

//Console.WriteLine(m\_q);

//Console.WriteLine("Коэффициенты Безу");

//Console.WriteLine(y\_p);

//Console.WriteLine(y\_q);

//Console.WriteLine("Результат дешифрования");

//Console.WriteLine(r1);

//Console.WriteLine(r2);

//Console.WriteLine(r3);

//Console.WriteLine(r4);

return (r1.ToByteArray(), r2.ToByteArray(), r3.ToByteArray(), r4.ToByteArray());

}

}

}

Form1.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.IO;

namespace Расчетное\_задание

{

public partial class MainForm : Form

{

public bool Is\_Kir = false;

public bool Is\_Lat = false;

public bool Is\_Num = false;

public bool Is\_Spec = false;

public int Pass\_Len = 5;

Encoding enc\_cur;

Encoding enc\_un = Encoding.Unicode;

public string password = "";

public string message = "";

public string cor\_password = "";

private bool Action\_Type; // true - шифровать false - расшифровывать

private bool Format\_Info; // true - файл false - текстовое поле

private OpenFileDialog ofd;

private string FilePath = "";

private byte[] key;

public void ClearForm()

{

RB\_Enc.Checked = false;

RB\_Dec.Checked = false;

RB\_File.Checked = false;

RB\_TextBox.Checked = false;

GB\_Info.Enabled = false;

But\_Action.Enabled = false;

Lab\_File.Visible = false;

CB\_Password.Visible = false;

CB\_Password.Checked = false;

TB\_Info.Text = "";

Lab\_File.Text = "";

FilePath = "";

password = "";

cor\_password = "";

}

public MainForm()

{

InitializeComponent();

}

private void оПрограммеToolStripMenuItem\_Click(object sender, EventArgs e) // о программе

{

FormAbout fa = new FormAbout();

fa.ShowDialog();

}

private void MainForm\_Shown(object sender, EventArgs e) // при первой загрузке окна (начальные данные)

{

RB\_Enc.Checked = false;

RB\_Dec.Checked = false;

RB\_File.Checked = false;

RB\_TextBox.Checked = false;

GB\_Info.Enabled = false;

But\_Action.Enabled = false;

Lab\_File.Visible = false;

CB\_Password.Visible = false;

key = new byte[24];

int i = 0;

while (i < 24)

{

byte[] k = enc\_un.GetBytes("Мария");

int j = 0;

while(j < k.Length && i < 24)

{

key[i] = k[j];

i++;

j++;

}

}

}

private void выходToolStripMenuItem\_Click(object sender, EventArgs e) // закрытие программы при нажатии на пункт меню

{

Close();

}

private void задатьОграниченияПарольнойФразыToolStripMenuItem\_Click(object sender, EventArgs e) // настройка парольной фразы при нажатии на пункт меню

{

PasswordParams FormPP = new PasswordParams();

FormPP.mf = this;

FormPP.ShowDialog();

}

private void CB\_Password\_CheckedChanged(object sender, EventArgs e) // Чекбокс использования парольной фразы

{

if (CB\_Password.Checked)

{

FormPassword FormPass = new FormPassword();

FormPass.mf = this;

FormPass.ShowDialog();

}

}

private void RB\_Enc\_CheckedChanged(object sender, EventArgs e) // Радиобатон шифрования

{

if (RB\_Enc.Checked)

{

Action\_Type = true;

GB\_Info.Enabled = true;

CB\_Password.Visible = true;

But\_Action.Text = Action\_Type ? "Шифровать" : "Расшифровать";

}

}

private void RB\_Dec\_CheckedChanged(object sender, EventArgs e) // Радиобатон дешифрования

{

if (RB\_Dec.Checked)

{

Action\_Type = false;

GB\_Info.Enabled = true;

CB\_Password.Visible = false;

But\_Action.Text = Action\_Type ? "Шифровать" : "Расшифровать";

}

}

private void RB\_File\_CheckedChanged(object sender, EventArgs e) // Радиобатон использования файла

{

if (RB\_File.Checked)

{

using (ofd = new OpenFileDialog())

{

ofd.RestoreDirectory = true;

if (ofd.ShowDialog() == DialogResult.OK)

{

FilePath = ofd.FileName;

Format\_Info = true;

Lab\_File.Visible = true;

Lab\_File.Text = Path.GetFileName(FilePath);

But\_Action.Enabled = true;

}

else

{

FilePath = "";

MessageBox.Show("Файл не задан");

return;

}

}

}

}

private void RB\_TextBox\_CheckedChanged(object sender, EventArgs e) // Радиобатон использования текстового поля

{

if (RB\_TextBox.Checked)

{

Format\_Info = false;

But\_Action.Enabled = true;

}

}

private void But\_Action\_Click(object sender, EventArgs e)

{

try

{

if (Action\_Type) // шифрование

{

if (CB\_Password.Checked) // с парольной фразой

{

if (password != "")

{

if (Format\_Info)

{

if (FilePath == "")

{

MessageBox.Show("Файл не задан");

return;

}

else

{

using (StreamReader sr = new StreamReader(FilePath, true))

{

enc\_cur = sr.CurrentEncoding;

}

using (StreamReader sr = new StreamReader(FilePath, enc\_cur, true))

{

message = sr.ReadToEnd();

}

DialogResult dr = MessageBox.Show("Удалить файл после использования?", "Удаление файла", MessageBoxButtons.YesNo);

switch (dr)

{

case DialogResult.Yes:

File.Delete(FilePath);

break;

case DialogResult.No:

break;

}

}

}

else

{

message = TB\_Info.Text;

}

int KeySize = password.Length \* 10 + (Is\_Kir ? 1 : 0) + (Is\_Lat ? 2 : 1) + (Is\_Num ? 3 : 2) + (Is\_Spec ? 4 : 3);

Rabin r = new Rabin(KeySize, KeySize);

r.SaveKey(key);

FormResult fr = new FormResult();

int mes\_len = message.Length;

int blocks = r.n.ToString().Length / 8;

blocks -= 26 + password.Length;

Console.WriteLine(blocks);

int i = 0;

while (mes\_len > 0)

{

string sub\_message = "Rabin" + "Password" + password + "Password" + message.Substring(i \* blocks, mes\_len > blocks ? blocks : mes\_len) + "Rabin";

byte[] enc\_un\_sub\_data = enc\_un.GetBytes(sub\_message);

string sub\_data = r.RabinEncrypt(enc\_un\_sub\_data);

fr.TB\_Result.Text += sub\_data + "\r\n";

mes\_len -= mes\_len > blocks ? blocks : mes\_len;

i++;

}

fr.ShowDialog();

ClearForm();

}

else

{

MessageBox.Show("Парольная фраза не задана");

}

}

else

{

if (Format\_Info)

{

if (FilePath == "")

{

MessageBox.Show("Файл не задан");

return;

}

else

{

using (StreamReader sr = new StreamReader(FilePath, true))

{

enc\_cur = sr.CurrentEncoding;

}

using (StreamReader sr = new StreamReader(FilePath, enc\_cur, true))

{

message = sr.ReadToEnd();

}

DialogResult dr = MessageBox.Show("Удалить файл после использования?", "Удаление файла", MessageBoxButtons.YesNo);

switch (dr)

{

case DialogResult.Yes:

File.Delete(FilePath);

break;

case DialogResult.No:

break;

}

}

}

else

{

message = TB\_Info.Text;

}

int KeySize = 100;

Rabin r = new Rabin(KeySize, KeySize);

r.SaveKey(key);

FormResult fr = new FormResult();

int mes\_len = message.Length;

int blocks = r.n.ToString().Length / 9;

blocks -= 10;

Console.WriteLine(blocks);

int i = 0;

while (mes\_len > 0)

{

string sub\_message = "Rabin" + message.Substring(i \* blocks, mes\_len > blocks ? blocks : mes\_len) + "Rabin";

byte[] enc\_un\_sub\_data = enc\_un.GetBytes(sub\_message);

string sub\_data = r.RabinEncrypt(enc\_un\_sub\_data);

fr.TB\_Result.Text += sub\_data + "\r\n";

mes\_len -= mes\_len > blocks ? blocks : mes\_len;

i++;

}

fr.ShowDialog();

ClearForm();

}

}

if (!Action\_Type) // дешифрование

{

if (Format\_Info)

{

if (FilePath == "")

{

MessageBox.Show("Файл не задан");

return;

}

else

{

using (StreamReader sr = new StreamReader(FilePath, true))

{

enc\_cur = sr.CurrentEncoding;

}

using (StreamReader sr = new StreamReader(FilePath, enc\_cur, true))

{

message = sr.ReadToEnd();

}

DialogResult dr = MessageBox.Show("Удалить файл после использования?", "Удаление файла", MessageBoxButtons.YesNo);

switch (dr)

{

case DialogResult.Yes:

File.Delete(FilePath);

break;

case DialogResult.No:

break;

}

}

}

else

{

message = TB\_Info.Text;

}

MessageBox.Show("Выберите контейнер ключей");

string filename;

OpenFileDialog ofd = new OpenFileDialog();

ofd.Filter = "bin files (\*.bin)|\*.bin";

ofd.RestoreDirectory = true;

if (ofd.ShowDialog() == DialogResult.Cancel)

{

MessageBox.Show("Контейнер не выбран");

return;

}

else

{

filename = ofd.FileName;

}

Console.WriteLine(filename);

Rabin r = new Rabin(filename, key);

FormResult fr = new FormResult();

bool istruebin = true;

while (istruebin && message != "")

{

string sub\_message = message.Substring(0, message.IndexOf("\r\n"));

if (!sub\_message.Equals("\n") && !sub\_message.Equals("\r") && !sub\_message.Equals("\r\n") && !sub\_message.Equals(""))

{

(byte[] data1, byte[] data2, byte[] data3, byte[] data4) = r.RabinDecrypt(sub\_message);

string mesdat1 = enc\_un.GetString(data1);

string mesdat2 = enc\_un.GetString(data2);

string mesdat3 = enc\_un.GetString(data3);

string mesdat4 = enc\_un.GetString(data4);

//Console.WriteLine(mesdat1);

//Console.WriteLine(mesdat2);

//Console.WriteLine(mesdat3);

//Console.WriteLine(mesdat4);

if (mesdat1.Length > 10 && mesdat1.Substring(0, 5).Equals("Rabin"))

{

string message1 = mesdat1.Substring(5, mesdat1.Length - 10);

if (message1.Length > 16 && message1.Substring(0, 8).Equals("Password"))

{

message1 = message1.Substring(8);

string password1 = message1.Substring(0, message1.IndexOf("Password"));

if (cor\_password.Equals(""))

{

FormPasswordCorrect fpc = new FormPasswordCorrect();

fpc.mf = this;

fpc.ShowDialog();

}

if (!password1.Equals(cor\_password))

{

istruebin = false;

}

message1 = message1.Substring(message1.IndexOf("Password") + 8);

}

fr.TB\_Result.Text += message1;

Console.Write(message1);

}

else if (mesdat2.Length > 10 && mesdat2.Substring(0, 5).Equals("Rabin"))

{

string message2 = mesdat2.Substring(5, mesdat2.Length - 10);

if (message2.Length > 16 && message2.Substring(0, 8).Equals("Password"))

{

message2 = message2.Substring(8);

string password2 = message2.Substring(0, message2.IndexOf("Password"));

if (cor\_password.Equals(""))

{

FormPasswordCorrect fpc = new FormPasswordCorrect();

fpc.mf = this;

fpc.ShowDialog();

}

if (!password2.Equals(cor\_password))

{

istruebin = false;

}

message2 = message2.Substring(message2.IndexOf("Password") + 8);

}

fr.TB\_Result.Text += message2;

Console.Write(message2);

}

else if (mesdat3.Length > 10 && mesdat3.Substring(0, 5).Equals("Rabin"))

{

string message3 = mesdat3.Substring(5, mesdat3.Length - 10);

if (message3.Length > 16 && message3.Substring(0, 8).Equals("Password"))

{

message3 = message3.Substring(8);

string password3 = message3.Substring(0, message3.IndexOf("Password"));

if (cor\_password.Equals(""))

{

FormPasswordCorrect fpc = new FormPasswordCorrect();

fpc.mf = this;

fpc.ShowDialog();

}

if (!password3.Equals(cor\_password))

{

istruebin = false;

}

message3 = message3.Substring(message3.IndexOf("Password") + 8);

}

fr.TB\_Result.Text += message3;

Console.Write(message3);

}

else if (mesdat4.Length > 10 && mesdat4.ToString().Substring(0, 5).Equals("Rabin"))

{

string message4 = mesdat4.Substring(5, mesdat4.Length - 10);

if (message4.Length > 16 && message4.Substring(0, 8).Equals("Password"))

{

message4 = message4.Substring(8);

string password4 = message4.Substring(0, message4.IndexOf("Password"));

if (cor\_password.Equals(""))

{

FormPasswordCorrect fpc = new FormPasswordCorrect();

fpc.mf = this;

fpc.ShowDialog();

}

if (!password4.Equals(cor\_password))

{

istruebin = false;

}

message4 = message4.Substring(message4.IndexOf("Password") + 8);

}

fr.TB\_Result.Text += message4;

Console.Write(message4);

}

else

{

istruebin = false;

}

}

message = message.Substring(message.IndexOf("\r\n") + 2);

}

if (istruebin)

{

fr.ShowDialog();

}

else

{

MessageBox.Show("Неверные данные");

}

ClearForm();

Console.WriteLine("");

}

}

catch

{

MessageBox.Show("Необработанная ошибка");

}

}

}

}

FormPassword.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Text.RegularExpressions;

namespace Расчетное\_задание

{

public partial class FormPassword : Form

{

public MainForm mf;

public FormPassword()

{

InitializeComponent();

}

private void But\_Action\_Click(object sender, EventArgs e)

{

string password = TB\_Pas.Text.Trim();

string password\_conf = TB\_Conf.Text.Trim();

if(password == password\_conf)

{

if(password != "")

{

if (password.Length == mf.Pass\_Len)

{

if (mf.Is\_Kir)

{

Regex regex = new Regex(@"[А-Яа-яЁё]");

MatchCollection matches = regex.Matches(password);

if (matches.Count == 0)

{

MessageBox.Show("Пароль не содержит символов кириллицы");

return;

}

}

if (mf.Is\_Lat)

{

Regex regex = new Regex(@"[A-Za-z]");

MatchCollection matches = regex.Matches(password);

if (matches.Count == 0)

{

MessageBox.Show("Пароль не содержит символов латиницы");

return;

}

}

if (mf.Is\_Num)

{

Regex regex = new Regex(@"[0-9]");

MatchCollection matches = regex.Matches(password);

if (matches.Count == 0)

{

MessageBox.Show("Пароль не содержит цифры");

return;

}

}

if (mf.Is\_Spec)

{

Regex regex = new Regex(@"[\\!$%&'()\*+,\-./:;<=>?@[ \]^\_`{|}~# ]");

MatchCollection matches = regex.Matches(password);

if (matches.Count == 0)

{

MessageBox.Show("Пароль не содержит специальные символы");

return;

}

}

}

else

{

MessageBox.Show("Пароль не соответствует длине " + mf.Pass\_Len);

return;

}

}

else

{

MessageBox.Show("Пароль не может быть пустой строкой");

return;

}

}

else

{

MessageBox.Show("Пароль и подтверждение не совпадают");

return;

}

mf.password = password;

MessageBox.Show("Пароль задан");

Close();

}

}

}

FormPasswordCorrect.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Расчетное\_задание

{

public partial class FormPasswordCorrect : Form

{

public MainForm mf;

public FormPasswordCorrect()

{

InitializeComponent();

}

private void But\_Action\_Click(object sender, EventArgs e)

{

mf.cor\_password = TB\_Pas.Text;

Close();

}

}

}

FormResult.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.IO;

namespace Расчетное\_задание

{

public partial class FormResult : Form

{

public Encoding enc\_un = Encoding.Unicode;

public FormResult()

{

InitializeComponent();

}

private void But\_Save\_Click(object sender, EventArgs e)

{

SaveFileDialog sfd = new SaveFileDialog();

sfd.Filter = "txt files (\*.txt)|\*.txt";

sfd.RestoreDirectory = true;

if (sfd.ShowDialog() == DialogResult.Cancel)

{

MessageBox.Show("Результат не сохранен");

return;

}

else

{

string namefile = sfd.FileName;

using (StreamWriter sw = new StreamWriter(namefile, false, enc\_un))

{

sw.WriteLine(TB\_Result.Text);

Close();

}

}

}

}

}

PasswordParams.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Расчетное\_задание

{

public partial class PasswordParams : Form

{

public MainForm mf;

public PasswordParams()

{

InitializeComponent();

}

private void But\_Action\_Click(object sender, EventArgs e)

{

if (int.TryParse(TB\_PasswordLen.Text, out int result))

{

if (Convert.ToInt32(TB\_PasswordLen.Text) >= 5)

{

mf.Is\_Kir = CB\_Kir.Checked;

mf.Is\_Lat = CB\_Lat.Checked;

mf.Is\_Num = CB\_Num.Checked;

mf.Is\_Spec = CB\_Spec.Checked;

mf.Pass\_Len = Convert.ToInt32(TB\_PasswordLen.Text);

MessageBox.Show("Параметры парольной фразы установлены");

Close();

}

else

{

MessageBox.Show("Длина парольной фразы должна быть больше или равна 5");

}

}

else

{

MessageBox.Show("Длина пароля - число типа int");

}

}

}

}