Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

институт

Кафедра «Информатика»

кафедра

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1**

«Симметричная криптография. Простые шифры»

тема

Преподаватель \_\_\_\_\_\_\_\_\_ В.А. Лазарева

подпись, дата инициалы, фамилия

Студент КИ16-16б, №031620303 \_\_\_\_\_\_\_\_\_ В.В. Кокташев

номер группы, зачетной книжки подпись, дата инициалы, фамилия

Красноярск 2018

**СОДЕРЖАНИЕ**

1 Цель работы…………………………………………………………...3

2 Задачи …………………………………………………………………3

3 Краткий теоретический материал……………………………………3

4 Ход работы……………………………………………………………4

5 Вывод…………………………………………...……………………18

ПРИЛОЖЕНИЕ А ……………………………………………….……19

ПРИЛОЖЕНИЕ Б…………………………………………………..…21

**1 Цель работы**

1. ознакомиться с основами симметричного шифрования;
2. ознакомиться с простыми симметричными криптографическими шифрами на основе методов подстановок, перестановок и гаммирования;
3. освоить основные этапы проектирования и реализации симметричных шифров.

# 2 Задание

1. разработать и составить в виде блок-схемы алгоритмы шифрования и дешифрования текста шифром маршрутной перестановки;
2. составить программу, которая реализует данные алгоритмы;
3. на ряде контрольных примеров (не менее 10) открытого текста, состоящего из различного количества символов, проверить правильность работы алгоритмов шифрования и дешифрования;
4. придумать оригинальный способ модификации шифра с целью повышения его криптостойкости, внести изменения в исходный алгоритм и программу, проверить работоспособность алгоритма на тестовых примерах;
5. доказать, что предложенный способ модификации действительно повышает криптостойкость.

# 3 Краткий теоретический материал

Шифр Плейфера – ручная [симметричная](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BC%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D1%87%D0%BD%D0%BE%D0%B5_%D1%88%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) техника [шифрования](https://ru.wikipedia.org/wiki/%D0%A8%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), в которой впервые использована [замена биграмм](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D1%8B%D0%B9_%D1%88%D0%B8%D1%84%D1%80). Изобретена в 1854 году английским [физиком](https://ru.wikipedia.org/wiki/%D0%A4%D0%B8%D0%B7%D0%B8%D0%BA) [Чарльзом Уитстоном](https://ru.wikipedia.org/wiki/%D0%A3%D0%B8%D1%82%D1%81%D1%82%D0%BE%D0%BD,_%D0%A7%D0%B0%D1%80%D0%BB%D1%8C%D0%B7), но названа именем [лорда](https://ru.wikipedia.org/wiki/%D0%9B%D0%BE%D1%80%D0%B4) [Лайона Плейфера](https://ru.wikipedia.org/w/index.php?title=%D0%9F%D0%BB%D0%B5%D0%B9%D1%84%D0%B5%D1%80,_%D0%9B%D0%B0%D0%B9%D0%BE%D0%BD,_1-%D0%B9_%D0%B1%D0%B0%D1%80%D0%BE%D0%BD_%D0%9F%D0%BB%D0%B5%D0%B9%D1%84%D0%B5%D1%80&action=edit&redlink=1" \o "Плейфер, Лайон, 1-й барон Плейфер (страница отсутствует)), который внёс большой вклад в продвижение использования данной системы шифрования в государственной службе. Шифр предусматривает шифрование пар символов (биграмм) вместо одиночных символов, как в [шифре подстановки](https://ru.wikipedia.org/wiki/%D0%A8%D0%B8%D1%84%D1%80_%D0%BF%D0%BE%D0%B4%D1%81%D1%82%D0%B0%D0%BD%D0%BE%D0%B2%D0%BA%D0%B8) и в более сложных системах шифрования [Виженера](https://ru.wikipedia.org/wiki/%D0%A8%D0%B8%D1%84%D1%80_%D0%92%D0%B8%D0%B6%D0%B5%D0%BD%D0%B5%D1%80%D0%B0" \o "Шифр Виженера). Шифр Плейфера использует матрицу 5х5 (для латинского алфавита, для кириллического алфавита необходимо увеличить размер матрицы до 4х8), содержащую ключевое слово или фразу. Для создания матрицы и использования шифра достаточно запомнить ключевое слово и четыре простых правила. Чтобы составить ключевую матрицу, в первую очередь нужно заполнить пустые ячейки матрицы буквами ключевого слова (не записывая повторяющиеся символы), потом заполнить оставшиеся ячейки матрицы символами алфавита, не встречающимися в ключевом слове, по порядку. Для того чтобы зашифровать сообщение, необходимо разбить его на биграммы (группы из двух символов), например, «Hello World» становится «HE LL OW OR LD», и отыскать эти биграммы в таблице. Два символа биграммы соответствуют углам прямоугольника в ключевой матрице. Определяем положения углов этого прямоугольника относительно друг друга. Затем, руководствуясь правилами, зашифровываем пары символов исходного текста.

**4 Ход работы**

1. Были созданы алгоритмы шифрования и дешифрования с помощью шифра Плейфера на языке C# и составлены соответствующие блок-схемы, представленные на рисунках 1-2. В первом случае входными данными является строка, содержащая открытый текст, во втором – строка с зашифрованным текстом. В качестве выходных данных выступает строка, к которой был применен один из алгоритмов.

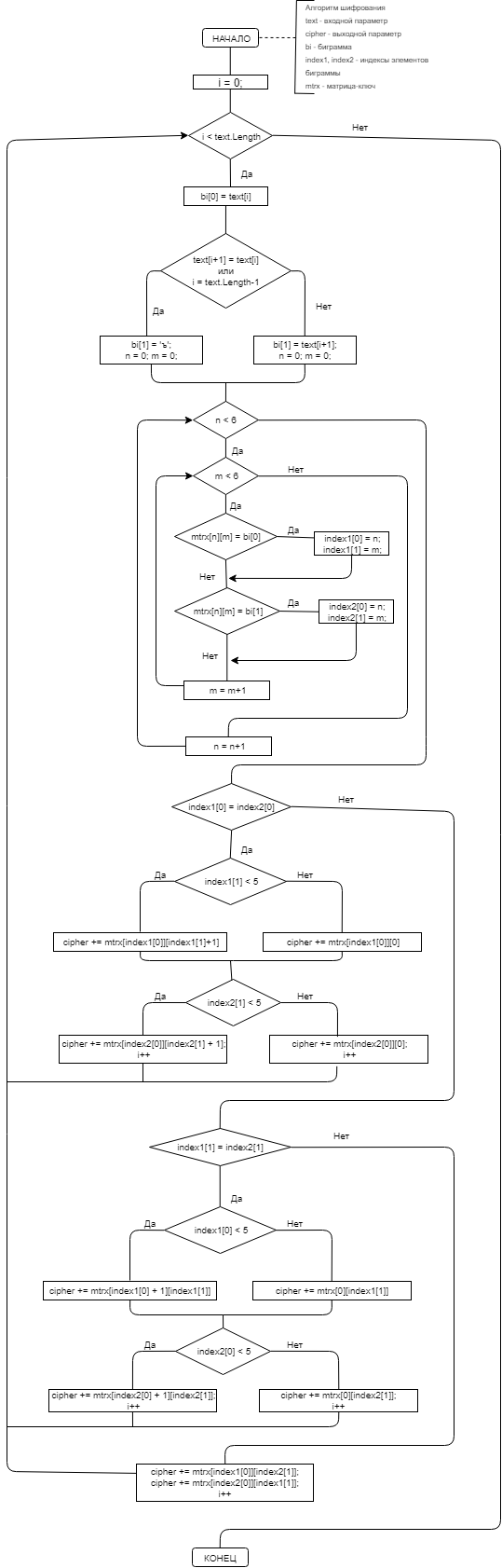


Рисунок 1 – Блок-схема алгоритма шифрования, лист 1

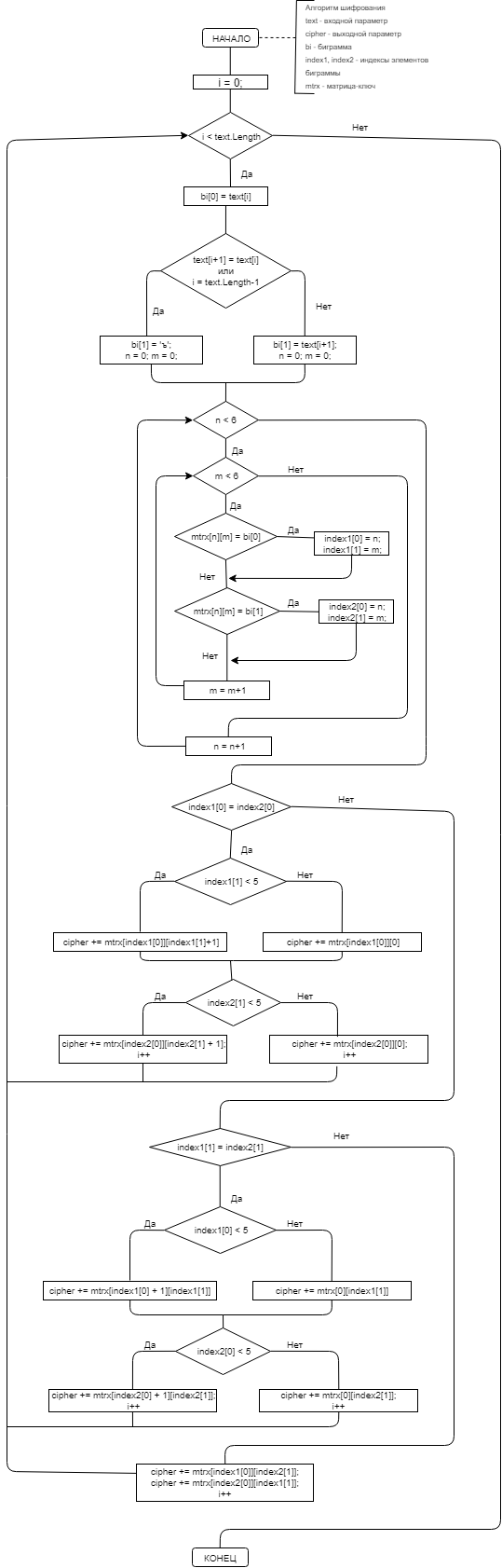


Рисунок 1, лист 2

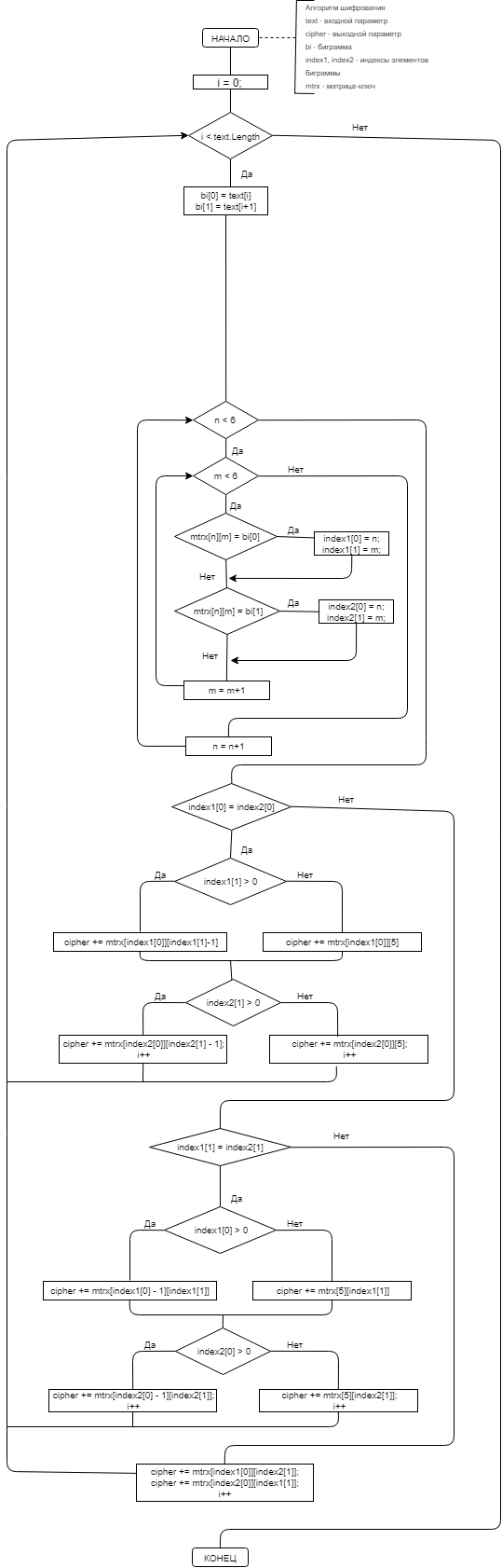


Рисунок 2 – Блок-схема алгоритма дешифрования, лист 1

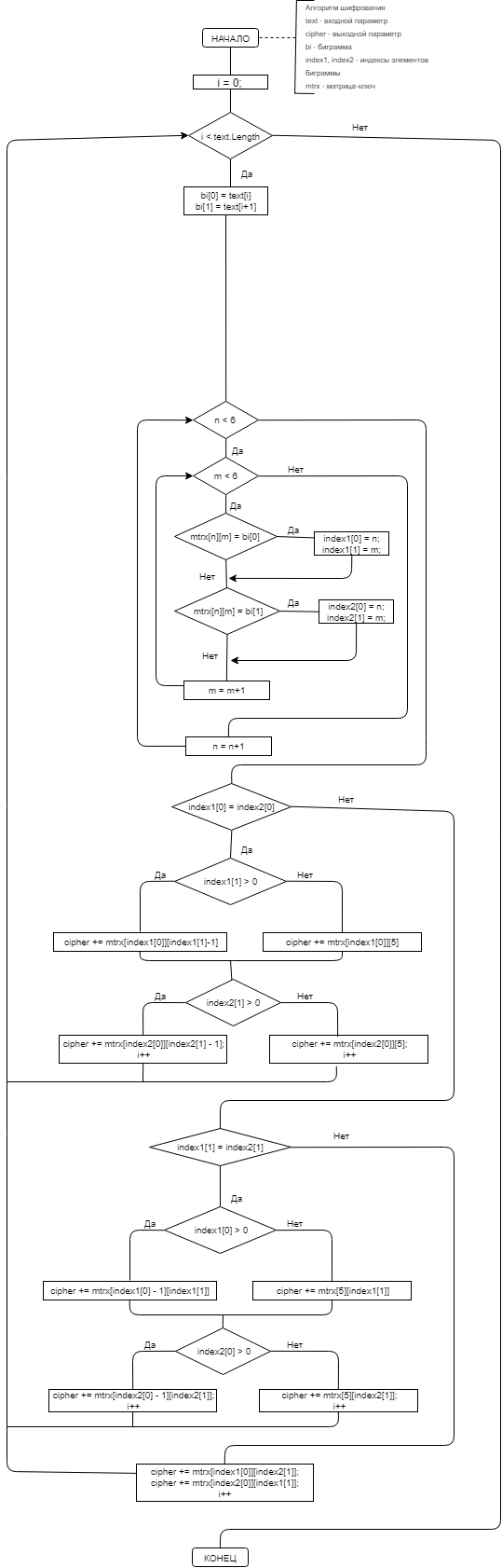


Рисунок 2, лист 2

1. Была разработана программа с графическим интерфейсом, реализующая данный алгоритм. Примеры её работы продемонстрированы на рисунках 3-7.

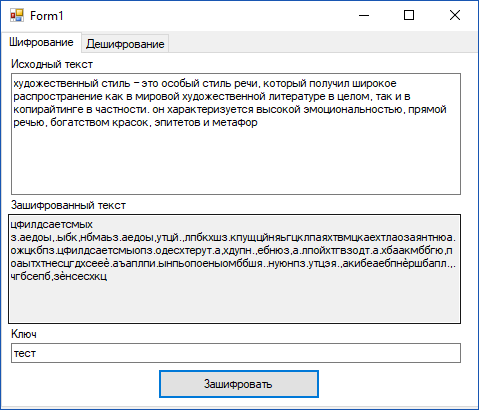


Рисунок 3 – Пример работы программы 1

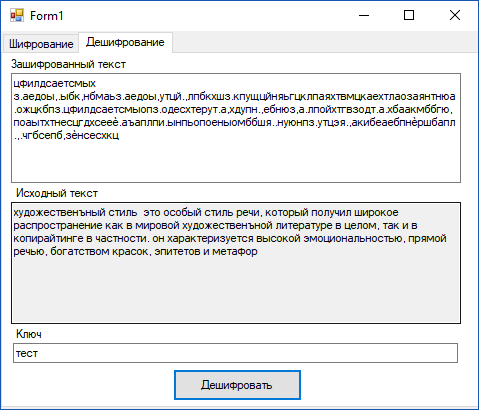


Рисунок 4 – Пример работы программы 2

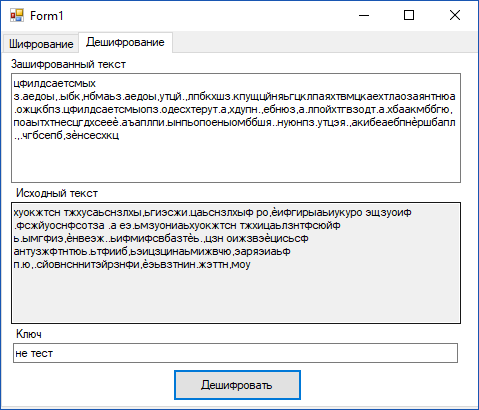


Рисунок 5 – Пример работы программы 3

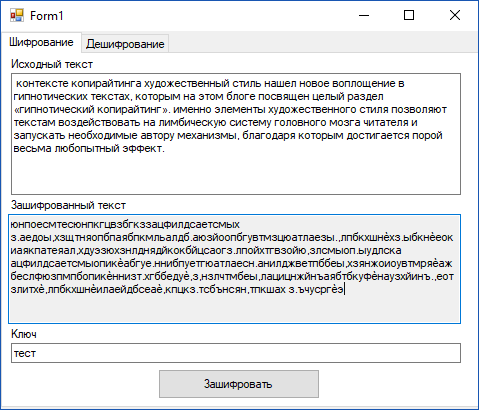


Рисунок 6 – Пример работы программы 4

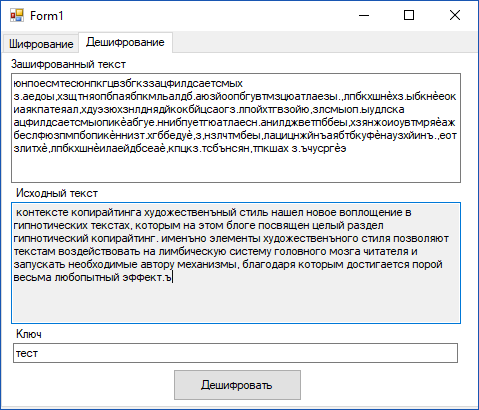


Рисунок 7 – Пример работы программы 5

1. Шифр Плейфера не устойчив к частотному анализу – методе криптографии, основывающемся на частотных характеристиках букв. Если злоумышленник обладает данными о статическом распределении отдельных символов или их групп, то процесс раскрытий шифра для него не окажется трудным. В связи с этим была предложена следующая модификация алгоритма шифрования: в биграммы добавляются случайные символы, никак не относящиеся к тексту. Поэтому из одного и того же открытого текста можно получить разные шифротексты. Блок-схемы алгоритмов шифрования и дешифрования изображены на рисунках 8-9 (отличающиеся от оригинала блоки имеют красную границу), примеры работы программы с внесенными изменениями – на рисунках 10-14.

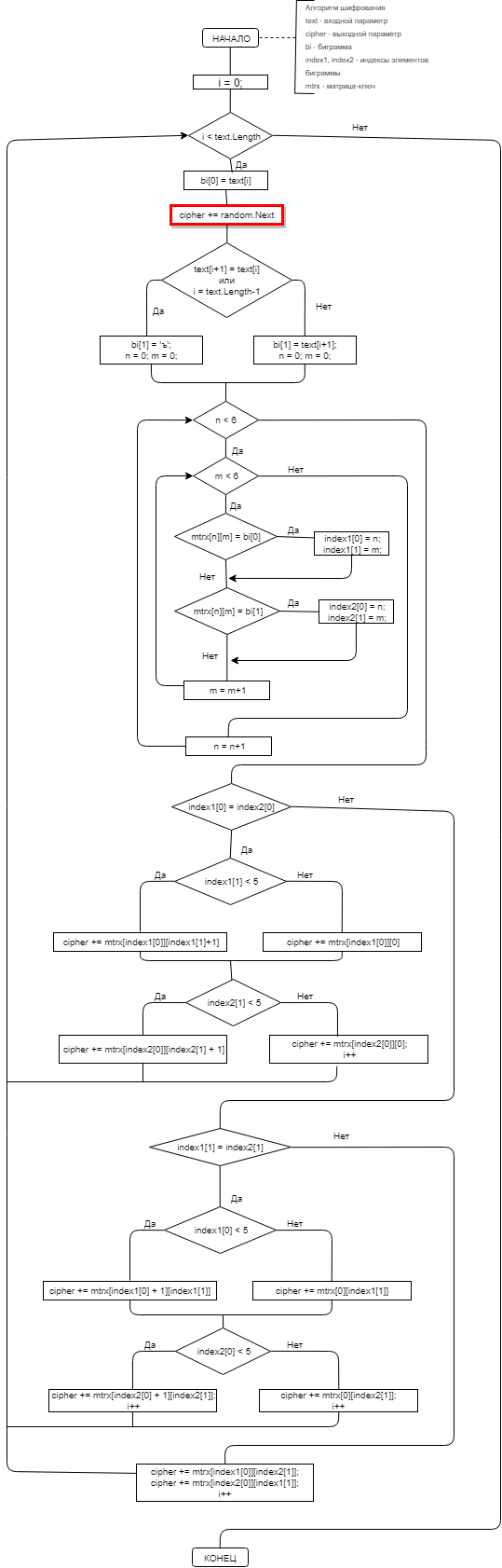


Рисунок 8 – Блок-схема модифицированного алгоритма шифрования, лист 1

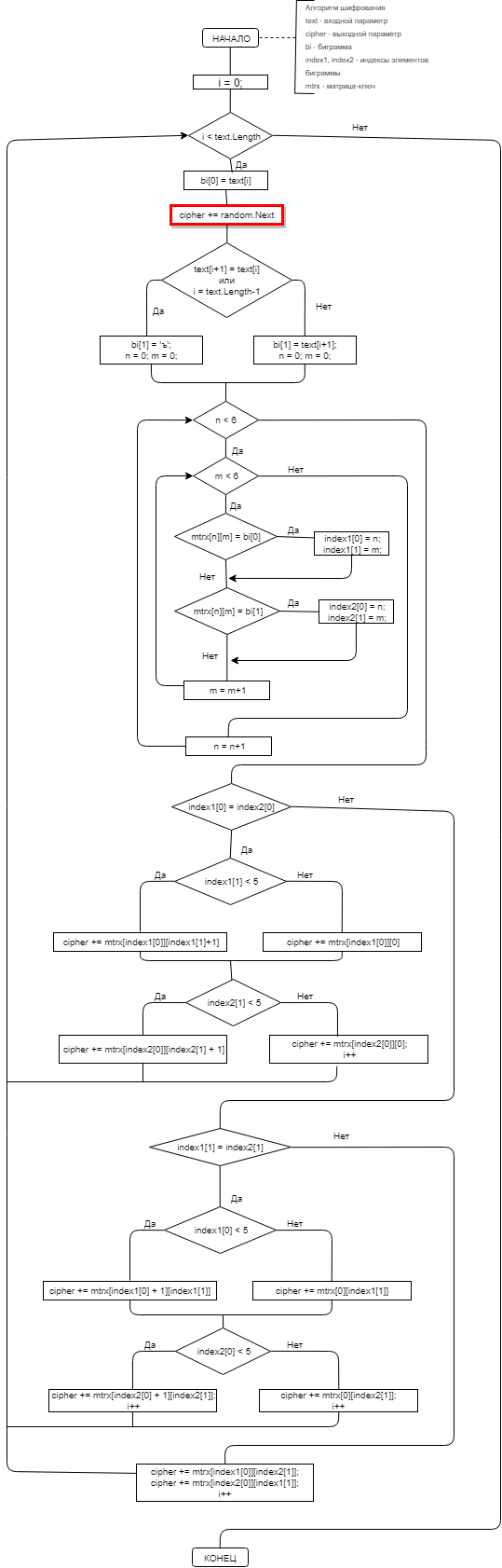


Рисунок 8 лист 2

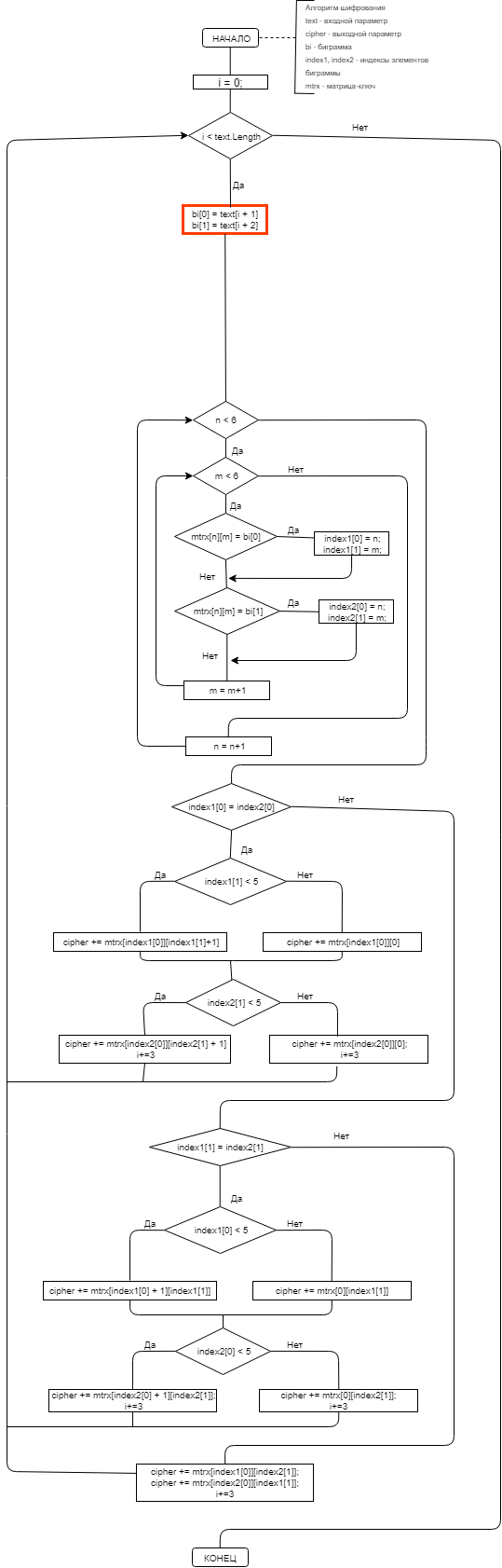


Рисунок 9 – Блок-схема модифицированного алгоритма дешифрования, лист 1

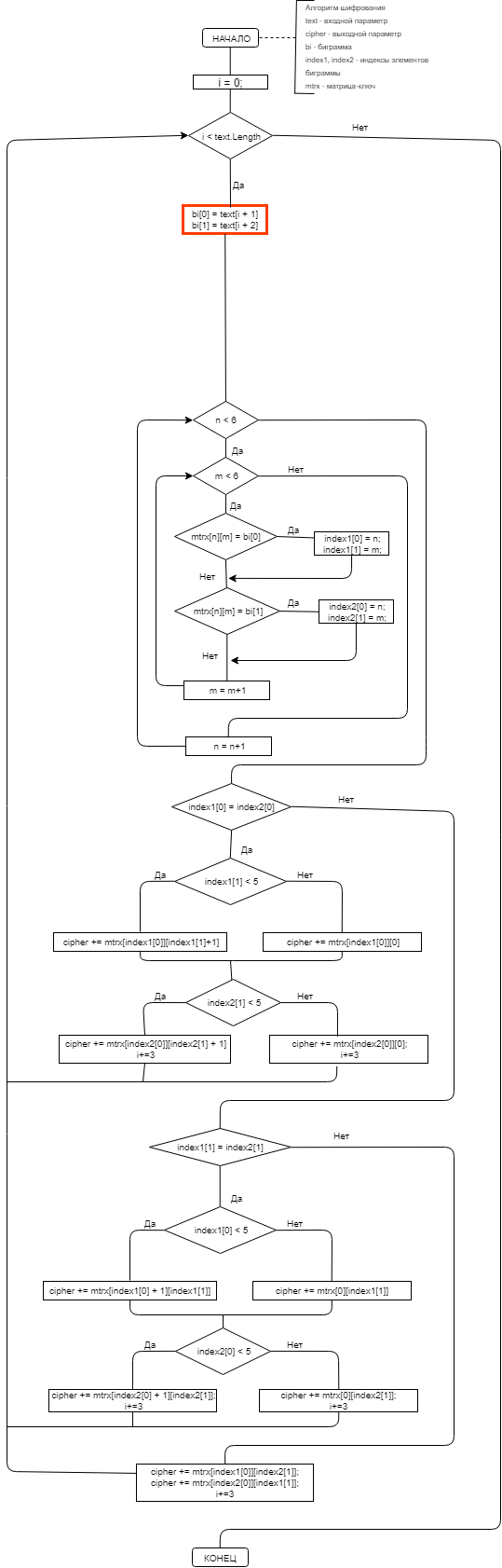


Рисунок 9 лист 2

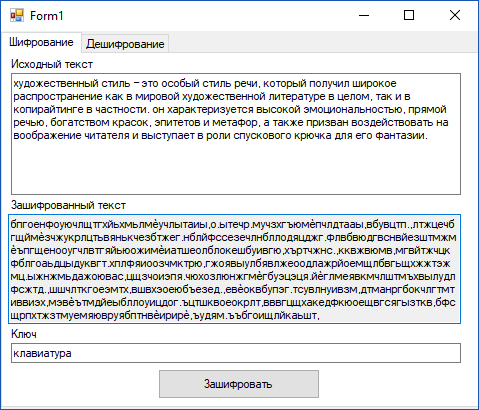


Рисунок 10 – Пример работы программы 7

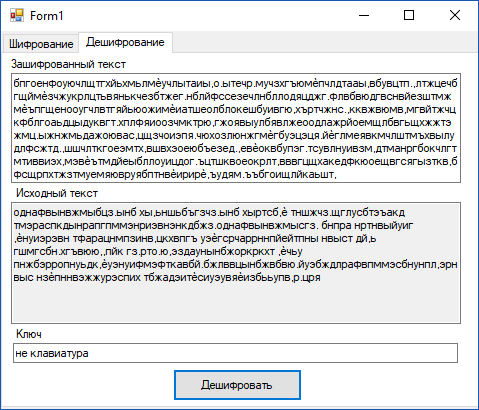


Рисунок 11 – Пример работы программы 8

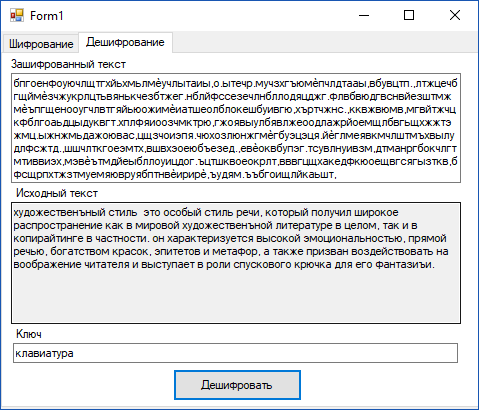


Рисунок 12 – Пример работы программы 9

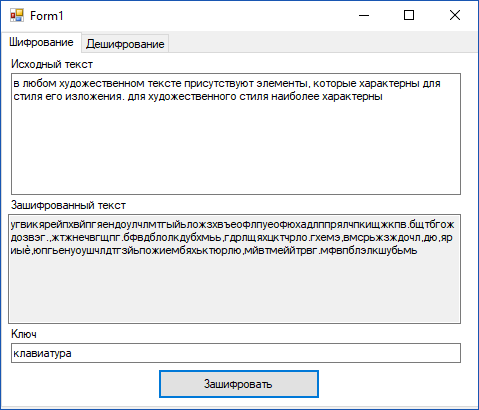


Рисунок 13 – Пример работы программы 10

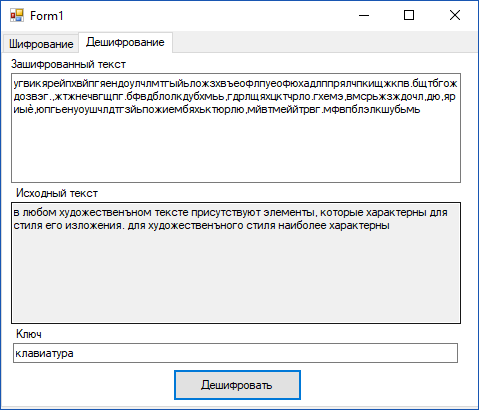


Рисунок 14 – Пример работы программы 11

# 5 Вывод

В ходе выполнения лабораторной работы были приобретены умения проектирования симметричных шифров с помощью реализации алгоритма Шифра Плейфера. Также получен опыт повышения криптостойкости существующих шифров (добавление в шифротекст случайныx символов с целью сокрытия биграмм). Закреплены навыки программирования на языке C# и платформе .NET Framework.

**ПРИЛОЖЕНИЕ А**

**Листинг исходного кода алгоритма шифра Плейфера**

Form1.cs:

|  |
| --- |
| using System;  using System.Windows.Forms;  namespace KLab\_1  {  public partial class Form1 : Form  {  char[][] mtrx = new char[6][];  char[] text;  char[] cipher = new char[0];  char[] bi = new char[2];  int[] index1 = new int[2], index2 = new int[2];  public Form1()  {  InitializeComponent();  mtrx[0] = new char[6] { 'ч', 'е', 'б', 'у', 'р', 'к' };  mtrx[1] = new char[6] { 'а', 'в', 'г', 'д', 'ё', 'ж' };  mtrx[2] = new char[6] { 'з', 'и', 'й', 'л', 'м', 'н' };  mtrx[3] = new char[6] { 'о', 'п', 'с', 'т', 'ф', 'х' };  mtrx[4] = new char[6] { 'ц', 'ш', 'щ', 'ъ', 'ы', 'ь' };  mtrx[5] = new char[6] { 'э', 'ю', 'я', ' ', '.', ',' };  }  private void button1\_Click(object sender, EventArgs e)  {  textBox1.Text = textBox1.Text.ToLower();  text = textBox1.Text.ToCharArray();  textBox2.Clear();  for (int i = 0; i < text.Length; i++)  if ((text[i] < 1072 || text[i] > 1105) && text[i] != 32 && text[i] != 44 && text[i] != 46)  {  for (int j = i; j < text.Length - 1; j++)  text[j] = text[j + 1];  Array.Resize(ref text, text.Length - 1);  i--;  }  for (int i = 0; i < text.Length; i++)  {  bi[0] = text[i];  if (i == text.Length - 1 || text[i] == text[i + 1])  bi[1] = 'ъ';  else  {  bi[1] = text[i + 1];  i++;  }  for (int n = 0; n < 6; n++)  for (int m = 0; m < 6; m++)  {  if (mtrx[n][m] == bi[0])  {  index1[0] = n;  index1[1] = m;  }  if (mtrx[n][m] == bi[1])  {  index2[0] = n;  index2[1] = m;  }  }  Array.Resize(ref cipher, cipher.Length + 2);  if (index1[0] == index2[0])  {  if (index1[1] < 5)  cipher[cipher.Length - 2] = mtrx[index1[0]][index1[1] + 1];  else  cipher[cipher.Length - 2] = mtrx[index1[0]][0];  if (index2[1] < 5)  cipher[cipher.Length - 1] = mtrx[index2[0]][index2[1] + 1];  else  cipher[cipher.Length - 1] = mtrx[index2[0]][0];  }  else  if (index1[1] == index2[1])  {  if (index1[0] < 5)  cipher[cipher.Length - 2] = mtrx[index1[0] + 1][index1[1]];  else  cipher[cipher.Length - 2] = mtrx[0][index1[1]];  if (index2[0] < 5)  cipher[cipher.Length - 1] = mtrx[index2[0] + 1][index2[1]];  else  cipher[cipher.Length - 1] = mtrx[0][index2[1]];  }  else  {  cipher[cipher.Length - 2] = mtrx[index1[0]][index2[1]];  cipher[cipher.Length - 1] = mtrx[index2[0]][index1[1]];  }  }  for (int i = 0; i < cipher.Length; i++)  textBox2.Text += cipher[i];  Array.Resize(ref text, 0);  Array.Resize(ref cipher, 0);  }  private void button2\_Click(object sender, EventArgs e)  {  textBox3.Text.ToLower();  textBox4.Clear();  if (textBox3.Text.Length % 2 == 1)  MessageBox.Show("Неверные данные!");  else  {  text = textBox3.Text.ToCharArray();  for (int i = 0; i < text.Length; i++)  if ((text[i] < 1072 || text[i] > 1105) && text[i] != 32 && text[i] != 44 && text[i] != 46)  {  for (int j = i; j < text.Length - 1; j++)  text[j] = text[j + 1];  Array.Resize(ref text, text.Length - 1);  i--;  }  for (int i = 0; i < text.Length; i += 2)  {  bi[0] = text[i];  bi[1] = text[i + 1];  for (int n = 0; n < 6; n++)  for (int m = 0; m < 6; m++)  {  if (mtrx[n][m] == bi[0])  {  index1[0] = n;  index1[1] = m;  }  if (mtrx[n][m] == bi[1])  {  index2[0] = n;  index2[1] = m;  }  }  Array.Resize(ref cipher, cipher.Length + 2);  if (index1[0] == index2[0])  {  if (index1[1] > 0)  cipher[cipher.Length - 2] = mtrx[index1[0]][index1[1] - 1];  else  cipher[cipher.Length - 2] = mtrx[index1[0]][5];  if (index2[1] > 0)  cipher[cipher.Length - 1] = mtrx[index2[0]][index2[1] - 1];  else  cipher[cipher.Length - 1] = mtrx[index2[0]][5];  }  else  if (index1[1] == index2[1])  {  if (index1[0] > 0)  cipher[cipher.Length - 2] = mtrx[index1[0] - 1][index1[1]];  else  cipher[cipher.Length - 2] = mtrx[5][index1[1]];  if (index2[0] > 0)  cipher[cipher.Length - 1] = mtrx[index2[0] - 1][index2[1]];  else  cipher[cipher.Length - 1] = mtrx[5][index2[1]];  }  else  {  cipher[cipher.Length - 2] = mtrx[index1[0]][index2[1]];  cipher[cipher.Length - 1] = mtrx[index2[0]][index1[1]];  }  }  for (int i = 0; i < cipher.Length; i++)  textBox4.Text += cipher[i];  Array.Resize(ref text, 0);  Array.Resize(ref cipher, 0);  }  }  }  } |

**ПРИЛОЖЕНИЕ Б**

**Листинг исходного кода модифицированного алгоритма шифра Плейфера**

Form1.cs:

|  |
| --- |
| using System;  using System.Windows.Forms;  namespace KLab\_1  {  public partial class Form1 : Form  {  char[][] mtrx = new char[6][];  char[] text;  char[] cipher = new char[0];  char[] bi = new char[2];  int[] index1 = new int[2], index2 = new int[2];  public Form1()  {  InitializeComponent();  mtrx[0] = new char[6] { 'ч', 'е', 'б', 'у', 'р', 'к' };  mtrx[1] = new char[6] { 'а', 'в', 'г', 'д', 'ё', 'ж' };  mtrx[2] = new char[6] { 'з', 'и', 'й', 'л', 'м', 'н' };  mtrx[3] = new char[6] { 'о', 'п', 'с', 'т', 'ф', 'х' };  mtrx[4] = new char[6] { 'ц', 'ш', 'щ', 'ъ', 'ы', 'ь' };  mtrx[5] = new char[6] { 'э', 'ю', 'я', ' ', '.', ',' };  }  private void button1\_Click(object sender, EventArgs e)  {  textBox1.Text = textBox1.Text.ToLower();  text = textBox1.Text.ToCharArray();  textBox2.Clear();  Random random = new Random();  for (int i = 0; i < text.Length; i++)  if ((text[i] < 1072 || text[i] > 1105) && text[i] != 32 && text[i] != 44 && text[i] != 46)  {  for (int j = i; j < text.Length - 1; j++)  text[j] = text[j + 1];  Array.Resize(ref text, text.Length - 1);  i--;  }  for (int i = 0; i < text.Length; i++)  {  bi[0] = text[i];  if (i == text.Length - 1 || text[i] == text[i + 1])  bi[1] = 'ъ';  else  {  bi[1] = text[i + 1];  i++;  }  for (int n = 0; n < 6; n++)  for (int m = 0; m < 6; m++)  {  if (mtrx[n][m] == bi[0])  {  index1[0] = n;  index1[1] = m;  }  if (mtrx[n][m] == bi[1])  {  index2[0] = n;  index2[1] = m;  }  }  Array.Resize(ref cipher, cipher.Length + 3);  cipher[cipher.Length - 3] = Convert.ToChar(random.Next(1072, 1104));  if (index1[0] == index2[0])  {  if (index1[1] < 5)  cipher[cipher.Length - 2] = mtrx[index1[0]][index1[1] + 1];  else  cipher[cipher.Length - 2] = mtrx[index1[0]][0];  if (index2[1] < 5)  cipher[cipher.Length - 1] = mtrx[index2[0]][index2[1] + 1];  else  cipher[cipher.Length - 1] = mtrx[index2[0]][0];  }  else  if (index1[1] == index2[1])  {  if (index1[0] < 5)  cipher[cipher.Length - 2] = mtrx[index1[0] + 1][index1[1]];  else  cipher[cipher.Length - 2] = mtrx[0][index1[1]];  if (index2[0] < 5)  cipher[cipher.Length - 1] = mtrx[index2[0] + 1][index2[1]];  else  cipher[cipher.Length - 1] = mtrx[0][index2[1]];  }  else  {  cipher[cipher.Length - 2] = mtrx[index1[0]][index2[1]];  cipher[cipher.Length - 1] = mtrx[index2[0]][index1[1]];  }  }  for (int i = 0; i < cipher.Length; i++)  textBox2.Text += cipher[i];  Array.Resize(ref text, 0);  Array.Resize(ref cipher, 0);  }  private void button2\_Click(object sender, EventArgs e)  {  textBox3.Text.ToLower();  textBox4.Clear();  if (textBox3.Text.Length % 2 == 0)  MessageBox.Show("Неверные данные!");  else  {  text = textBox3.Text.ToCharArray();  for (int i = 0; i < text.Length; i++)  if ((text[i] < 1072 || text[i] > 1105) && text[i] != 32 && text[i] != 44 && text[i] != 46)  {  for (int j = i; j < text.Length - 1; j++)  text[j] = text[j + 1];  Array.Resize(ref text, text.Length - 1);  i--;  }  for (int i = 0; i < text.Length; i += 3)  {  bi[0] = text[i + 1];  bi[1] = text[i + 2];  for (int n = 0; n < 6; n++)  for (int m = 0; m < 6; m++)  {  if (mtrx[n][m] == bi[0])  {  index1[0] = n;  index1[1] = m;  }  if (mtrx[n][m] == bi[1])  {  index2[0] = n;  index2[1] = m;  }  }  Array.Resize(ref cipher, cipher.Length + 2);  if (index1[0] == index2[0])  {  if (index1[1] > 0)  cipher[cipher.Length - 2] = mtrx[index1[0]][index1[1] - 1];  else  cipher[cipher.Length - 2] = mtrx[index1[0]][5];  if (index2[1] > 0)  cipher[cipher.Length - 1] = mtrx[index2[0]][index2[1] - 1];  else  cipher[cipher.Length - 1] = mtrx[index2[0]][5];  }  else  if (index1[1] == index2[1])  {  if (index1[0] > 0)  cipher[cipher.Length - 2] = mtrx[index1[0] - 1][index1[1]];  else  cipher[cipher.Length - 2] = mtrx[5][index1[1]];  if (index2[0] > 0)  cipher[cipher.Length - 1] = mtrx[index2[0] - 1][index2[1]];  else  cipher[cipher.Length - 1] = mtrx[5][index2[1]];  }  else  {  cipher[cipher.Length - 2] = mtrx[index1[0]][index2[1]];  cipher[cipher.Length - 1] = mtrx[index2[0]][index1[1]];  }  }  for (int i = 0; i < cipher.Length; i++)  textBox4.Text += cipher[i];  Array.Resize(ref text, 0);  Array.Resize(ref cipher, 0);  }  }  }  } |