Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

|  |
| --- |
| Институт космических и информационных технологий |
| *институт* |
| Кафедра Прикладной математики и компьютерной безопасности |
| *кафедра* |

**ЛАБОРАТОРНАЯ РАБОТА №15**

|  |
| --- |
| **Криптоанализ полиалфавитных шифров** |
| *тема* |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Руководитель | | |  |  |  | В.И.Вайнштейн |
|  | |  |  | *подпись, дата* |  | *инициалы, фамилия* |
| Студент | КИ15-01 №031508683 | |  |  |  | М.С.Димаксян |
|  | *номер группы, зачетной книжки* | |  | *подпись, дата* |  | *инициалы, фамилия* |

Красноярск 2019

**Задание:**

Необходимо реализовать анализ полиалфавитного шифртекста (на любом языке программирования) на основе следующих методов.

**Теория:**

Метод Касиски

Взлом полиалфавитного шифра в [1863 году](https://ru.wikipedia.org/wiki/1863_%D0%B3%D0%BE%D0%B4) в своей книге «Тайнопись и искусство дешифрования»([нем.](https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D0%BC%D0%B5%D1%86%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *«Die Geheimschriften und die Dechiffrirkunst»*) описал отставной офицер прусской армии [Фридрих Касиски](https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D1%81%D0%B8%D1%81%D0%BA%D0%B8,_%D0%A4%D1%80%D0%B8%D0%B4%D1%80%D0%B8%D1%85) ([нем.](https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D0%BC%D0%B5%D1%86%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Friedrich Kasiski*). Данный метод применим для шифров с коротким ключевым словом, так как основан на поиске повторяющихся последовательностей символов в шифротексте. Длина ключа определяет количество вариантов, которыми может быть зашифрована одна и та же последовательность в открытом тексте. Таким образом, можно предположить, что количество букв между повторяющимися последовательностями в шифротексте кратно длине кодового слова. Далее [криптоаналитик](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B0%D0%BD%D0%B0%D0%BB%D0%B8%D1%82%D0%B8%D0%BA) определяет вероятное число символов в кодовом слове путём подбора общих [делителей](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%BB%D0%B8%D0%BC%D0%BE%D1%81%D1%82%D1%8C) для всех интервалов между повторениями. На заключительном этапе первоначальный [шифротекст](https://ru.wikipedia.org/wiki/%D0%A8%D0%B8%D1%84%D1%80%D0%BE%D1%82%D0%B5%D0%BA%D1%81%D1%82) рассматривается как несколько [моноалфавитных шифров](https://ru.wikipedia.org/wiki/%D0%A8%D0%B8%D1%84%D1%80_%D0%BF%D1%80%D0%BE%D1%81%D1%82%D0%BE%D0%B9_%D0%B7%D0%B0%D0%BC%D0%B5%D0%BD%D1%8B), к которым применяется [частотный криптоанализ](https://ru.wikipedia.org/wiki/%D0%A7%D0%B0%D1%81%D1%82%D0%BE%D1%82%D0%BD%D1%8B%D0%B9_%D0%B0%D0%BD%D0%B0%D0%BB%D0%B8%D0%B7).

Этот метод взлома полиалфавитного шифра был разработан ещё в [1854 году](https://ru.wikipedia.org/wiki/1854_%D0%B3%D0%BE%D0%B4) [Чарльзом Бэббиджем](https://ru.wikipedia.org/wiki/%D0%91%D1%8D%D0%B1%D0%B1%D0%B8%D0%B4%D0%B6,_%D0%A7%D0%B0%D1%80%D0%BB%D1%8C%D0%B7) ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Charles Babbage*). Однако в связи с тем, что он не опубликовал результаты своего труда, об этом стало известно только в [XX веке](https://ru.wikipedia.org/wiki/XX_%D0%B2%D0%B5%D0%BA). Именно поэтому такого рода способ дешифровки связывают с именем Фридриха Касиски, а также носит название «[Метод Касиски](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%9A%D0%B0%D1%81%D0%B8%D1%81%D0%BA%D0%B8)».

Индекс совпадений

В [1920 году](https://ru.wikipedia.org/wiki/1920_%D0%B3%D0%BE%D0%B4) была опубликована выдающаяся [монография](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%BD%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%8F) под названием «Индекс совпадения и его применение в криптоанализе» ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *«Index of Coincidence and Its Applications in Cryptography»*). Её автором является известный криптоаналитик того времени [Уильям Фридман](https://ru.wikipedia.org/wiki/%D0%A4%D1%80%D0%B8%D0%B4%D0%BC%D0%B0%D0%BD,_%D0%A3%D0%B8%D0%BB%D1%8C%D1%8F%D0%BC_%D0%A4%D1%80%D0%B5%D0%B4%D0%B5%D1%80%D0%B8%D0%BA) ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *William Frederick Friedman*). Он заинтересовался поиском способа вскрытия [шифра 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) со случайным длинным ключом[[](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%B8%D0%B0%D0%BB%D1%84%D0%B0%D0%B2%D0%B8%D1%82%D0%BD%D1%8B%D0%B9_%D1%88%D0%B8%D1%84%D1%80#cite_note-10). В своем анализе он впервые обратился к [статистическим](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D1%82%D0%B8%D1%81%D1%82%D0%B8%D0%BA%D0%B0) методам, рассмотрев текст, как единый объект с взаимосвязанными элементами. Благодаря работам Фридмана появилось понятие «[индекса совпадений](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D0%B4%D0%B5%D0%BA%D1%81_%D1%81%D0%BE%D0%B2%D0%BF%D0%B0%D0%B4%D0%B5%D0%BD%D0%B8%D0%B9)», а криптоанализ обогатился средствами из категории статистических исследований.

**Исходный код:**

**Информация об алфавитах:**

public string alf\_ru = "абвгдеёжзийклмнопрстуфхцчшщъыьэюя";

public string alf\_RU = "АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ";

public string alf\_en = "abcdefghijklmnopqrstuvwxyz";

public string alf\_EN = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

public double[] freqs\_ru = {0.062,0.014,0.038,0.013,0.025,0.072,0.0001,0.007,0.016,0.062,0.01,0.028,0.035,0.026,0.053,0.09,0.023,0.04,0.045,0.053,0.021,0.002,0.009,0.004,0.012,0.006,0.003,0.0004,0.016,0.014,0.003,0.006,0.018};

public double[] freqs\_en = {0.081,0.016,0.032,0.036,0.123,0.023,0.016,0.051,0.071,0.001,0.005,0.04,0.022,0.072,

0.079,0.023,0.002,0.06,0.066,0.096,0.031,0.009,0.02,0.002,0.019,0.001};

public string alf = "", ALF = "";

**Основная функция поиска длины ключа:**

private void f15\_buttonFindKeyLength\_Click(object sender, EventArgs e)

{

f15\_keyLengthUpDown.Value = 0; //Сброс предыдущего результата

if (f15\_comboBoxMethods.Text == "Метод индекса совпадений" && f15\_fieldCiphertext.TextLength > 0)

{

int possibleLength = 1;

while (possibleLength < 50)

{

string Source = (f15\_fieldCiphertext.Text).ToLower();

string everyXsymb = ""; // Формируем строку БЕЗ УЧЕТА РАЗДЕЛИТЕЛЕЙ из каждого х-того символа, где х = possibleLength

int j = 0;

for (int i = 0; i < Source.Length; i++)

{

if (j % possibleLength == 0 && (alf.IndexOf(Source[i]) != -1 || ALF.IndexOf(Source[i]) != -1)) //буква, нужная по номеру

{

everyXsymb += Source[i];

j++;

}

else if (j % possibleLength != 0 && (alf.IndexOf(Source[i]) != -1 || ALF.IndexOf(Source[i]) != -1)) //буква, не нужная по номеру

{

j++;

}

}

//Подсчитаем индекс для текста без пробелов

double index = 0;

int[] times = new int[alf.Length];

Array.Clear(times, 0, alf.Length);

//Вычислим количество вхождений каждой буквы

int amount = 0;

for (int i = 0; i < everyXsymb.Length; i++)

{

if (alf.IndexOf(everyXsymb[i]) >= 0)

{

amount++;

times[alf.IndexOf(everyXsymb[i])]++;

}

}

for (int i = 0; i < times.Length; i++)

{

index += (times[i]) \* (times[i] - 1);

}

index = index / (double)(amount \* (amount - 1));

if (index > 0.0553)

{

f15\_keyLengthUpDown.Value = possibleLength;

goto metka\_exit;

}

possibleLength++;

}

metka\_exit:;

}

if (f15\_comboBoxMethods.Text == "Автокорреляционный метод" && f15\_fieldCiphertext.TextLength > 0)

{

string Source = (f15\_fieldCiphertext.Text).ToLower();

for (int i = 0; i < Source.Length; i++) //Удаляем разделители

{

if (alf.IndexOf(Source[i]) < 0)

{

string tmpDel = "";

tmpDel += Source[i];

Source = Source.Replace(tmpDel, "");

}

}

for (int possibleLength = 1; possibleLength < 50; possibleLength++)

{

string copySource = Source.Substring(possibleLength) + Source.Substring(0, possibleLength);

int coincidences = 0;

for (int i = 0; i < Source.Length; i++)

{

if (Source[i] == copySource[i]) coincidences++;

}

double index = coincidences / (double) Source.Length;

if (index > 0.0553)

{

f15\_keyLengthUpDown.Value = possibleLength;

break;

}

}

}

if (f15\_comboBoxMethods.Text == "Тест Казиски" && f15\_fieldCiphertext.TextLength > 0)

{

string Source = (f15\_fieldCiphertext.Text).ToLower();

for (int i = 0; i < Source.Length; i++) //Удаляем разделители

{

if (alf.IndexOf(Source[i]) < 0)

{

string tmpDel = "";

tmpDel += Source[i];

Source = Source.Replace(tmpDel, "");

}

}

if (Source.Length < 10) goto metka\_exit; //Для работы нужен достаточно длинный текст

string checked\_sublines = ""; //В эту строку через пробел будем записывать проверенные строки

List<int> possibleLengths = new List<int>();

//если индекс в ней -1 то можно проверять, иначе - не надо

for (int i = 0; i < Source.Length/3 - 1; i++)

{

string subline = Source.Substring(i, 3);

if (checked\_sublines.IndexOf(subline) == -1)

{

//Считаем количество вхождений подстроки

int times = 0;

int j = 0;

while (Source.IndexOf(subline, j) != -1)

{

j = Source.IndexOf(subline, j) + 1; //Следующий поиск начнем после текущего вхождения

times++;

};

if (times >= 3)

{

checked\_sublines += subline + " ";

List<int> positionsDifference = new List<int>();

int prev = Source.IndexOf(subline); //первое вхождение

j = prev + 1; //начнем поиск со второго

while (Source.IndexOf(subline, j) != -1)

{

int curIndex = Source.IndexOf(subline, j);

positionsDifference.Add(curIndex - 1);

prev = curIndex;

j = curIndex + 1;

};

int curGCD = positionsDifference[0];

for (int k = 1; k < positionsDifference.Count; k++)

{

curGCD = GCD(curGCD, positionsDifference[k]);

}

if (curGCD > 1) possibleLengths.Add(curGCD);

}

}

}

if (possibleLengths.Count > 0)

{

possibleLengths.Sort();

int[] possibility = new int[50];

int i = 0;

while (i < possibleLengths.Count() && possibleLengths[i] < 50)

{

possibility[possibleLengths[i]]++;

i++;

}

for (i = 0; i < 50; i++)

possibility[i] = possibility[i] \* i; //Уменьшим значимость ложных срабатываний

for (i = 0; i < 50; i++)

if (possibility[i] == possibility.Max())

f15\_keyLengthUpDown.Value = i;

}

metka\_exit:;

}

}

**Основная функция поиска ключа:**

private void f15\_buttonFindKeys\_Click(object sender, EventArgs e)

{

int keyLen = (int) f15\_keyLengthUpDown.Value;

string Source = (f15\_fieldCiphertext.Text).ToLower();

for (int i = 0; i < Source.Length; i++) //Удаляем разделители

{

if (alf.IndexOf(Source[i]) < 0)

{

string tmpDel = "";

tmpDel += Source[i];

Source = Source.Replace(tmpDel, "");

}

}

if (keyLen < 1 || Source.Length < f15\_keyLengthUpDown.Value)

goto metka\_exit;

//Разобьем текст на keyLen шифров цезаря

string[] everyXsymb = new string[keyLen];

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

{

string tmp = "";

int j = k;

while (j < Source.Length)

{

tmp += Source[j];

j += keyLen;

}

everyXsymb[k] = tmp;

}

string result = "";

//Найдем ключ для каждого Цезаря отдельно

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

{

//Подсчитаем количество вхождений каждой буквы

int[] times = new int[alf.Length];

Array.Clear(times, 0, times.Length);

for (int i = 0; i < everyXsymb[k].Length; i++)

{

if (alf.IndexOf(everyXsymb[k][i]) >= 0)

{

times[alf.IndexOf(everyXsymb[k][i])]++;

}

}

//Найдем самый часто встречающийся элемент, запишем его индекс

int index = -1;

for (int i = 0; i < times.Length; i++)

{

if (times[i] == times.Max()) index = i;

}

//Запишем номер самой встречающейся статистичеки буквы в переменную standard

int standard = alf.IndexOf('e'); //английская

if (alf.IndexOf('о') >= 0) standard = alf.IndexOf('о'); //русская

if (alf.IndexOf('\_') >= 0) standard = alf.IndexOf('\_'); //пробел в виде подчеркивания

if (alf.IndexOf(' ') >= 0) standard = alf.IndexOf(' '); //пробел

result += alf[((index - standard) + alf.Length) % alf.Length];

}

f15\_foundedKey.Text = result;

metka\_exit:;

}

**Функция отображения результата дешифрования в окне:**

private void f15\_fieldKey\_TextChanged(object sender, EventArgs e)

{

if (f15\_fieldKey.TextLength > 0)

{

string res = "";

int j = 0;

for (int i = 0; i < f15\_fieldCiphertext.TextLength; i++)

{

int curSymbNum = alf.IndexOf(f15\_fieldCiphertext.Text[i]);

int rotSymbNum = alf.IndexOf(f15\_fieldKey.Text[j]);

if (curSymbNum != -1)

{//строчная буква

res += alf[((curSymbNum - rotSymbNum) + alf.Length) % alf.Length];

j++;

j = j % f15\_fieldKey.TextLength;

}

else

{

curSymbNum = ALF.IndexOf(f15\_fieldCiphertext.Text[i]);

if (curSymbNum != -1)//заглавная буква

{

res += ALF[((curSymbNum - rotSymbNum) + ALF.Length) % ALF.Length];

j++;

j = j % f15\_fieldKey.TextLength;

}

else

{

//Символы, которых нет в алфавите, просто переносятся \

//и не меняют позицию в ключе

res += f15\_fieldCiphertext.Text[i];

/\*j++;

j = j % f15\_fieldKey.TextLength; \*/

}

}

}

f15\_fieldOriginal.Text = res;

}

else f15\_fieldOriginal.Text = null;

}

**Функция заполнения таблицы и диаграммы:**

private void f15\_Table\_Chart\_func()

{

int[] times\_ru = new int[alf\_ru.Length]; //Массив подсчета вхождений каждой буквы

int[] times\_en = new int[alf\_en.Length];

Array.Clear(times\_ru, 0, alf\_ru.Length);

Array.Clear(times\_en, 0, alf\_en.Length);

string analyzedText = f15\_fieldOriginal.Text.ToLower();

//Подсчитаем количество русских и английских букв, количество их вхождений

int amount\_ru = 0, amount\_en = 0;

for (int i = 0; i < analyzedText.Length; i++)

{

if (alf\_ru.IndexOf(analyzedText[i]) >= 0)

{

amount\_ru++;

times\_ru[alf\_ru.IndexOf(analyzedText[i])]++;

}

if (alf\_en.IndexOf(analyzedText[i]) >= 0)

{

amount\_en++;

times\_en[alf\_en.IndexOf(analyzedText[i])]++;

}

}

f15\_dataGridView.ColumnCount = 4;

f15\_dataGridView.RowCount = 1;

int real\_index = 0;

for (int i = 0; i < times\_ru.Length; i++)

{

if (times\_ru[i] > 0)

{

f15\_dataGridView.RowCount++;

f15\_dataGridView.Rows[real\_index].Cells[0].Value = alf\_ru[i];

f15\_dataGridView.Rows[real\_index].Cells[1].Value = times\_ru[i];

f15\_dataGridView.Rows[real\_index].Cells[2].Value = Math.Round((double)times\_ru[i] / amount\_ru, 6); //округлим до 6 знаков после ,

f15\_dataGridView.Rows[real\_index].Cells[3].Value = freqs\_ru[i];

real\_index++;

}

}

for (int i = 0; i < times\_en.Length; i++)

{

if (times\_en[i] > 0)

{

f15\_dataGridView.RowCount++;

f15\_dataGridView.Rows[real\_index].Cells[0].Value = alf\_en[i];

f15\_dataGridView.Rows[real\_index].Cells[1].Value = times\_en[i];

f15\_dataGridView.Rows[real\_index].Cells[2].Value = Math.Round((double)times\_en[i] / amount\_en, 6); //округлим до 6 знаков после ,

f15\_dataGridView.Rows[real\_index].Cells[3].Value = freqs\_en[i];

real\_index++;

}

}

//Заполнение гистограммы

f15\_chart.Series[0].Points.Clear(); //Очистка старых точек

if (amount\_ru > 0)

{

for (int i = 0; i < times\_ru.Length; i++)

{

f15\_chart.Series[0].Points.Add(times\_ru[i]);

}

}

if (amount\_en > 0)

{

for (int i = 0; i < times\_en.Length; i++)

{

f15\_chart.Series[0].Points.Add(times\_en[i]);

}

}

}

**Функция проверки введенного шифртекста текста:**

private void f15\_fieldCiphertext\_TextChanged(object sender, EventArgs e)

{

f15\_fieldOriginal.Text = null;

f15\_foundedKey.Text = null;

f15\_keyLengthUpDown.Value = 0;

//Подсчитаем количество русских и английских букв, количество их вхождений

int amount\_ru = 0, amount\_en = 0;

for (int i = 0; i < f15\_fieldCiphertext.TextLength; i++)

{

if (alf\_ru.IndexOf(f15\_fieldCiphertext.Text[i]) >= 0 || alf\_RU.IndexOf(f15\_fieldCiphertext.Text[i]) >= 0)

{

amount\_ru++;

}

if (alf\_en.IndexOf(f15\_fieldCiphertext.Text[i]) >= 0 || alf\_EN.IndexOf(f15\_fieldCiphertext.Text[i]) >= 0)

{

amount\_en++;

}

}

if (amount\_ru > 0) { alf = alf\_ru; ALF = alf\_RU; }

if (amount\_en > 0) { alf = alf\_en; ALF = alf\_EN; }

f15\_alf.Text = alf;

f15\_alfExtended.Text = "";

if (amount\_en \* amount\_ru > 0)

{

f15\_fieldCiphertext.Text = null;

MessageBox.Show("Пожалуйста, используйте только один язык.", "Ошибка");

}

string tmp = f15\_fieldKey.Text;

f15\_fieldKey.Text = null;

f15\_fieldKey.Text = tmp;

}

**Функция расширения алфавита**

private void f15\_alfExtended\_TextChanged(object sender, EventArgs e)

{

if (f15\_checkBox\_alfEx.Checked)

alf = f15\_alfExtended.Text + f15\_alf.Text;

else

alf = f15\_alf.Text + f15\_alfExtended.Text;

ALF = alf.ToUpper();

string tmp = f15\_fieldKey.Text;

f15\_fieldKey.Text = null;

f15\_fieldKey.Text = tmp;

}

**Работа с файлами:**

private void f15\_ButtonReadCiphertext\_Click(object sender, EventArgs e)

{

f15\_fieldOriginal.Text = null;

f15\_foundedKey.Text = null;

f15\_fieldKey.Text = null;

f15\_keyLengthUpDown.Value = 0;

string textFromFile = "";

Stream myStream = null;

OpenFileDialog myDialog = new OpenFileDialog();

myDialog.InitialDirectory = "D:\\Учеба\\Крипта\\Labs\\Labs\\bin\\Debug\\TextFiles";

myDialog.DefaultExt = "txt";

myDialog.FileName = "15\_rus.txt";

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

myDialog.FilterIndex = 2;

myDialog.RestoreDirectory = true;

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

{

try

{

if ((myStream = myDialog.OpenFile()) != null)

{

textFromFile = new StreamReader(myStream, Encoding.GetEncoding(1251)).ReadToEnd();

myStream.Close();

}

}

catch (Exception ex)

{

MessageBox.Show("Error: Could not read file from disk. Original error: " + ex.Message);

}

}

//Подсчитаем количество русских и английских букв, количество их вхождений

int amount\_ru = 0, amount\_en = 0;

for (int i = 0; i < textFromFile.Length; i++)

{

if (alf\_ru.IndexOf(textFromFile[i]) >= 0)

{

amount\_ru++;

}

if (alf\_en.IndexOf(textFromFile[i]) >= 0)

{

amount\_en++;

}

}

if (amount\_ru > 0) { alf = alf\_ru; ALF = alf\_RU; }

if (amount\_en > 0) { alf = alf\_en; ALF = alf\_EN; }

f15\_alf.Text = alf;

f15\_alfExtended.Text = "";

f15\_fieldCiphertext.Text = textFromFile;

if (amount\_en \* amount\_ru > 0)

{

f15\_fieldCiphertext.Text = null;

MessageBox.Show("Пожалуйста, используйте только один язык.", "Ошибка");

}

}

private void f15\_ButtonSaveOriginal\_Click(object sender, EventArgs e)

{

StreamWriter myStream = null;

SaveFileDialog myDialog = new SaveFileDialog();

myDialog.InitialDirectory = "D:\\Учеба\\Крипта\\Labs\\Labs\\bin\\Debug\\TextFiles";

myDialog.DefaultExt = "txt";

myDialog.FileName = "15\_out.txt";

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

myDialog.FilterIndex = 2;

myDialog.RestoreDirectory = true;

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

{

if ((myStream = new StreamWriter(myDialog.OpenFile(), Encoding.GetEncoding(1251))) != null)

{

myStream.WriteLine(Convert.ToString(f15\_fieldOriginal.Text));

myStream.Close();

}

}

}

**Результат работы программы:**

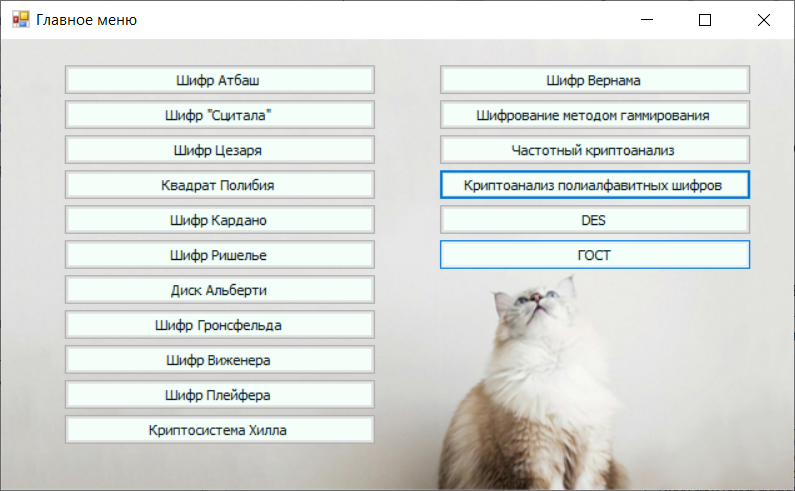


Рисунок 1. Главное меню программы.

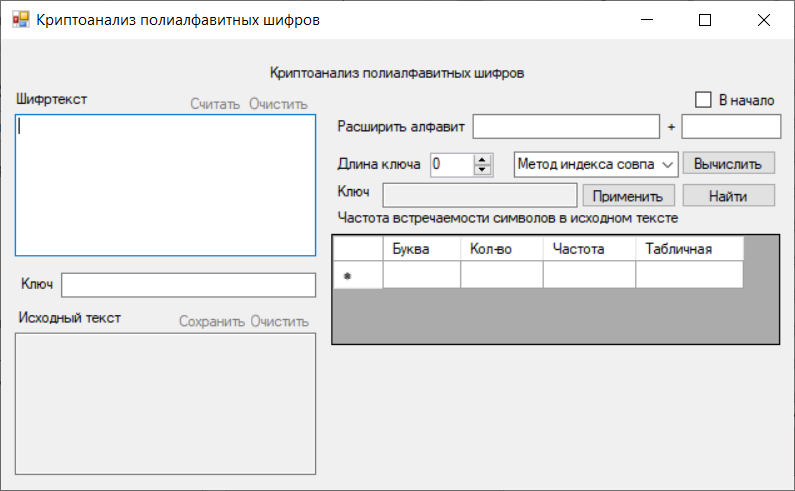


Рисунок 2. Окно криптоанализа полиалфавитных шифров.

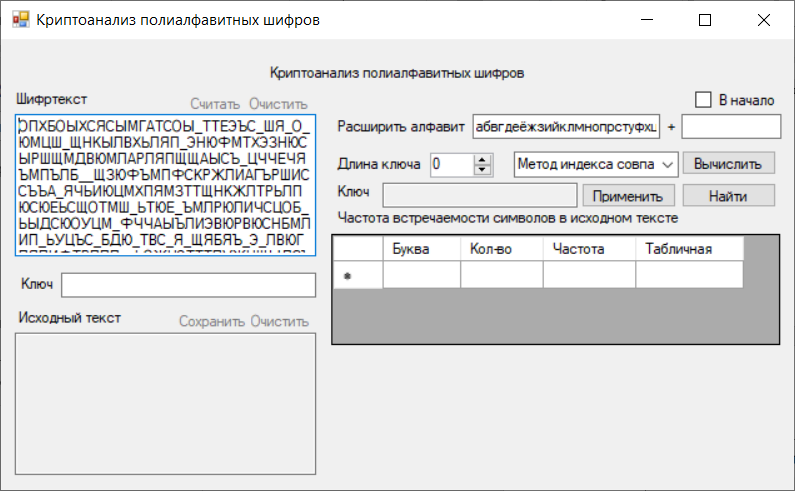


Рисунок 3. Загрузка шифртекста для анализа.

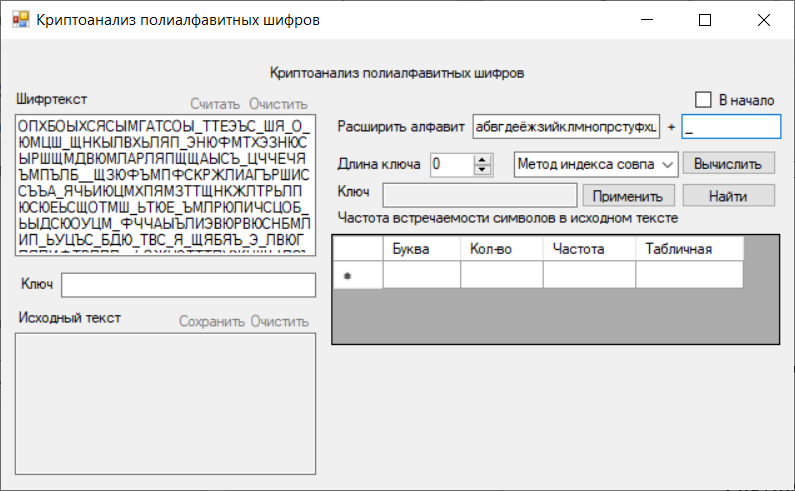


Рисунок 4. Расширение алфавита.

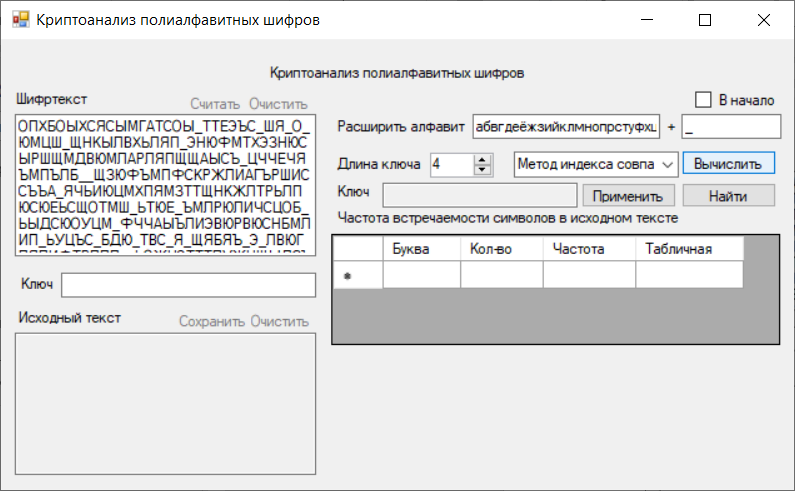


Рисунок 5. Расчет длины ключа с использованием метода индекса совпадений.

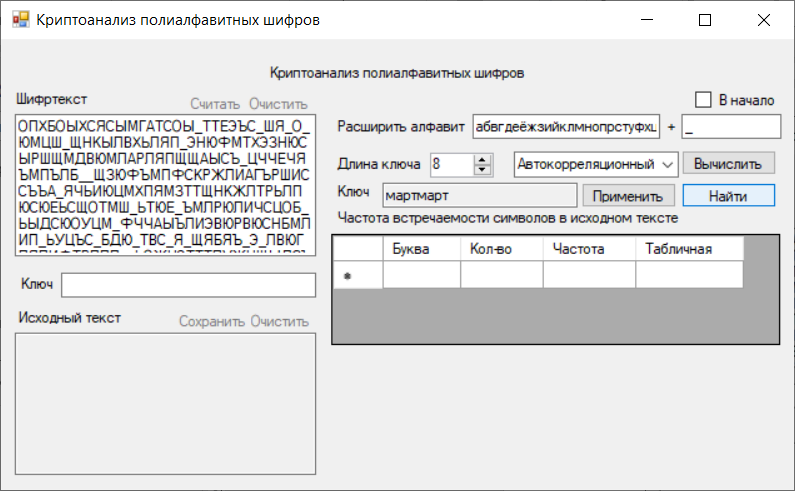


Рисунок 6. Расчет длины ключа с использованием автокорреляционного метода и поиск ключа.

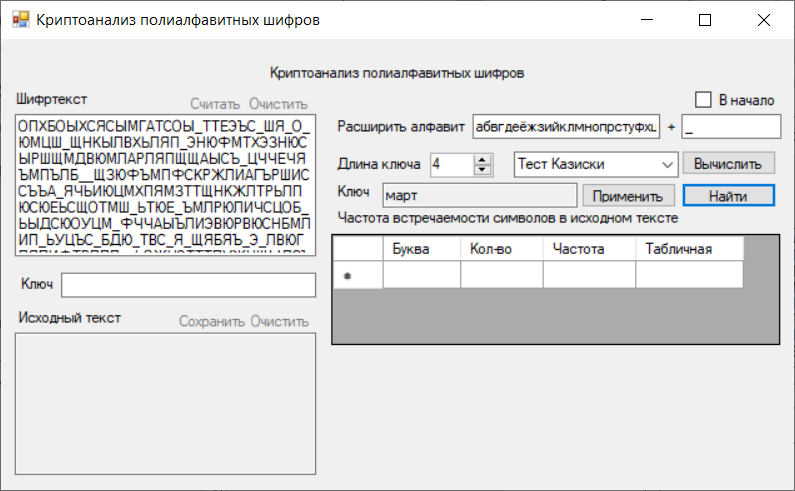


Рисунок 7. Расчет длины ключа с использованием теста Казиски и поиск ключа.

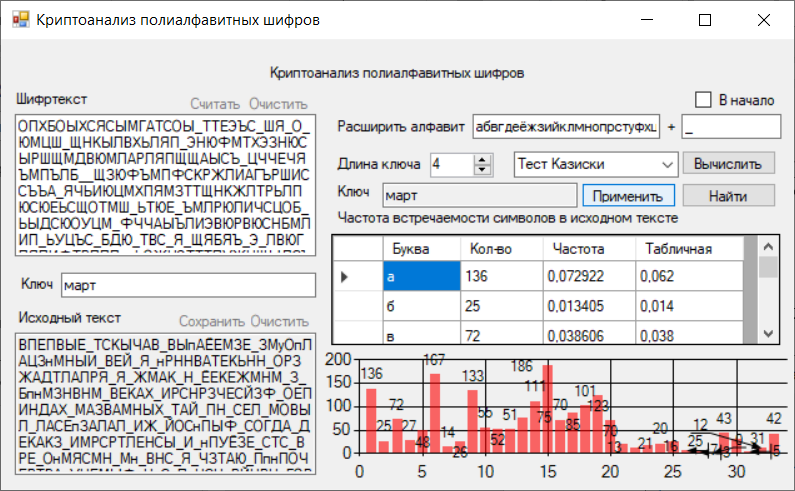


Рисунок 8. Применим найденный ключ к шифртексту.

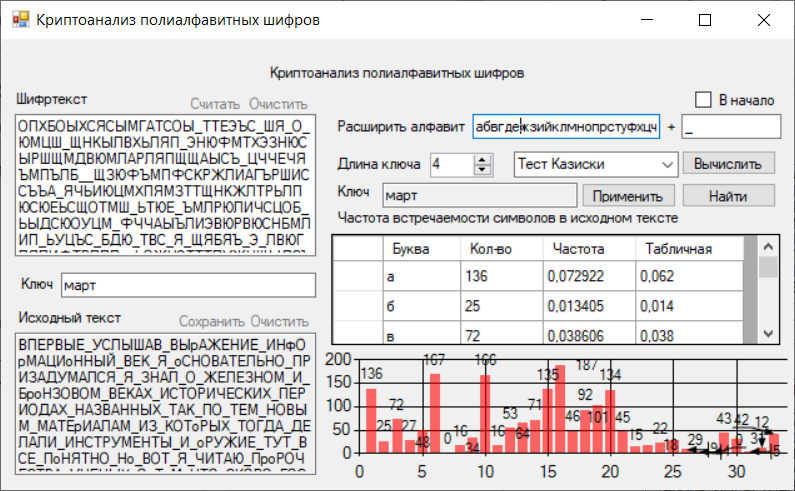


Рисунок 9. Скорректируем алфавит.

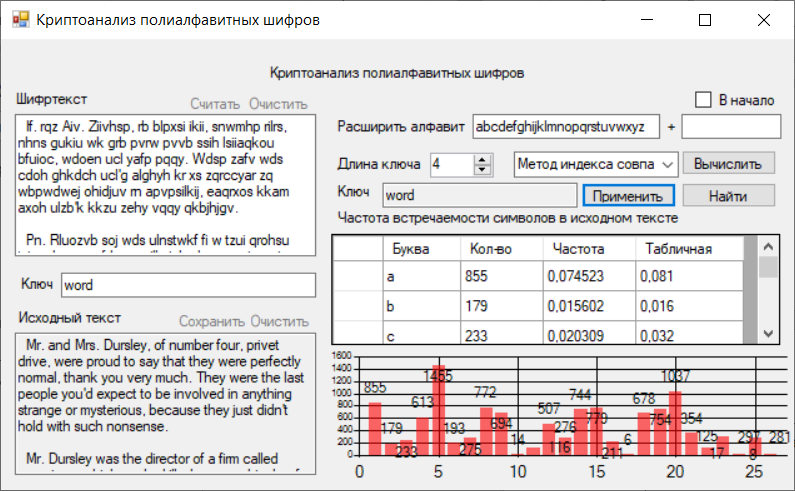


Рисунок 10. Загрузим другой текст, найдем и применим ключ.

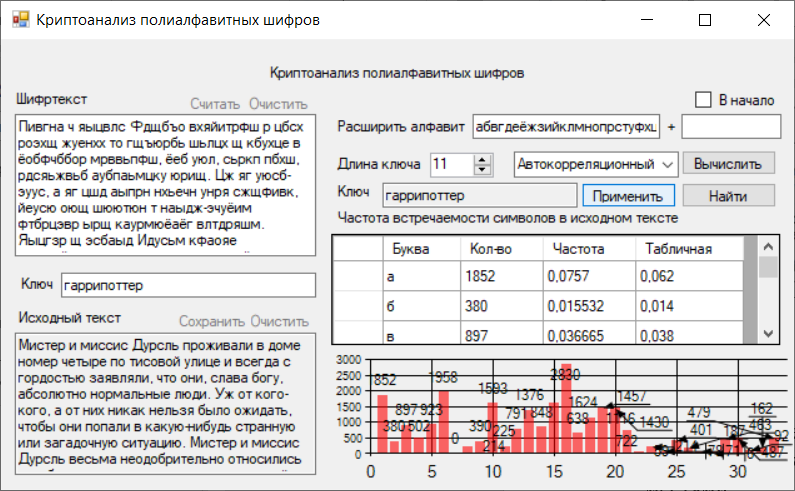


Рисунок 11. Загрузим другой текст, найдем и применим ключ.

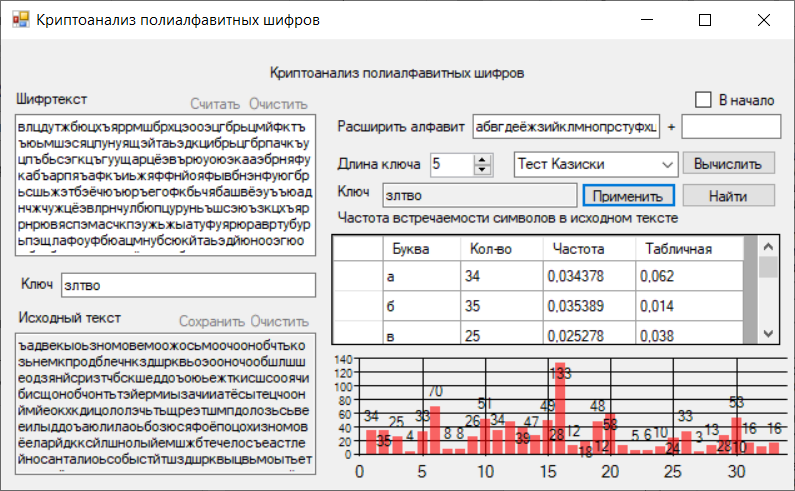


Рисунок 12. Загрузим текст из задания к лабораторной, найдем и применим ключ.

Заметим, что программа не смогла выполнить полное дешифрование из-за недостаточной длины текста. В данном случае необходимо продолжить анализ вручную. Текст не может начинаться с «ъ», следовательно, первая буква ключа определена неверно.

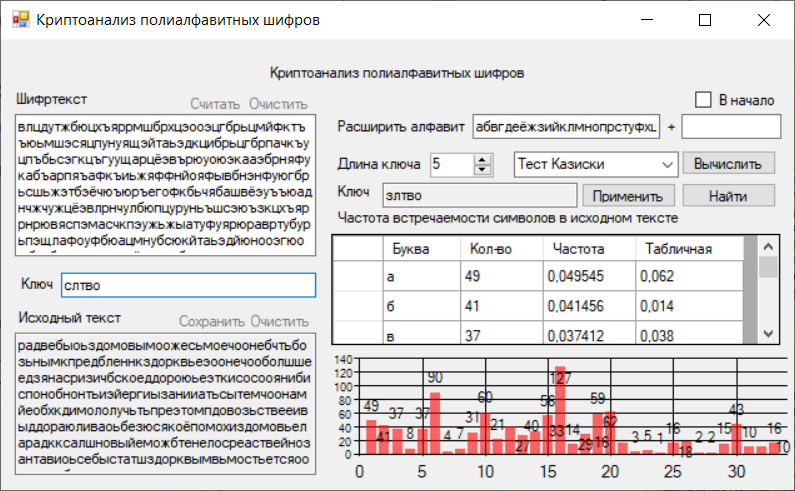


Рисунок 13. Анализ вручную.

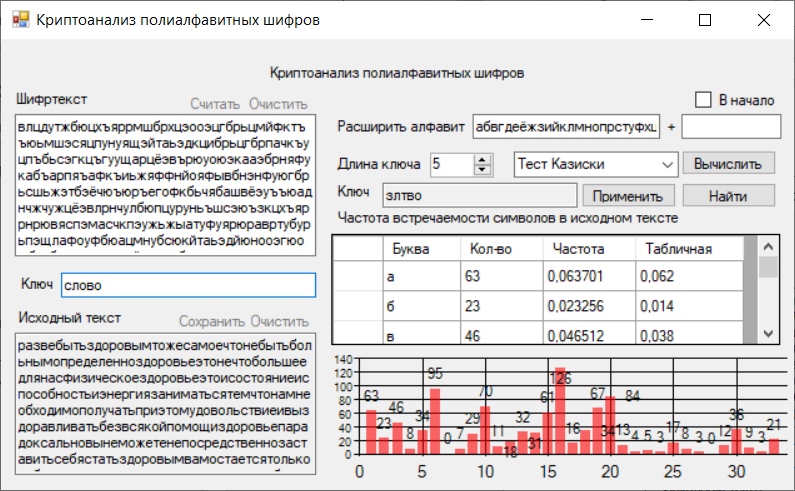


Рисунок 14. Успешное дешифрование.

**Вывод:**

В ходе выполнения лабораторной работы я ознакомилась с методами криптоанализа полиалфавитных шифров, реализовала на практике программу по дешифрованию данными методами на языке C#. В реализованной программе есть возможность выбирать используемый метод и анализировать сложные случаи вручную.