  
Министерство науки и высшего образования Российской Федерации  
Федеральное государственное автономное образовательное  
учреждение высшего образования

**КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ**

**ОКПО 02066730, ОГРН 1021602841391, ИНН/КПП 1655018018/165502011**

**ИНСТИТУТ ФИЗИКИ**

Кремлевская ул., д.18, Казань, 420008

**Отчет по работе**

**«Шифр Рихарда Зорге»**

Выполнила студентка

2 курса группы 06-852

Крапивина Ю.Д.

Преподаватель

Шаймухаметов Р.Р.

Казань 2020

Содержание:

Шифр Рихарда Зорге ………………………………………………………....3

Описание работы ……………………………………………………………..9

Результаты работы программы …..…………………………………..….….14

Код программы …………………………………………………….…..…….17

Шифр Рихарда Зорге

Рихарду Зорге в истории нашей Родины суждено было занять особое место. Десятки книг, сотни статей, документальные и художественные фильмы, улицы, названные в честь великого разведчика. Шифр Рихарда Зорге (руководителя японской резидентуры ГРУ «Рамзай») о котором здесь пойдёт речь, нужно рассматривать как типовой образец действующих шифросистем всех советских спецслужб.

Свои телеграммы в Москву Зорге для конспирации составлял преимущественно на английском языке. Поэтому в качестве ключа для построения квадратного шифра было выбрано слово «SUBWAY».

Ключ выписывался в верхней строке квадратной таблички. А в оставшиеся клетки по порядку проставлялись буквы английского алфавита, не вошедшие в слово SUBWAY , что переводится как «подземный ход». Таким образом, мы получим следующую сетку:

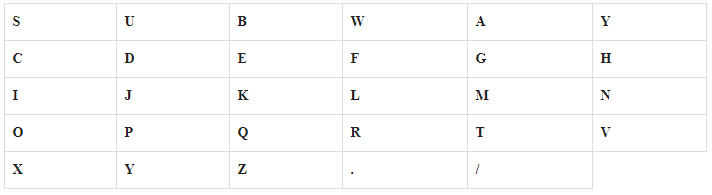


Таблица 1

В конце алфавита в таблице добавлено два знака. Это точка (.) и знак индикатора (/) — для обозначения разделителя слов.

Однако таблица в подобном виде использовалась только для придания вошедшим в нее символам новых цифровых обозначений.

Известно, что наиболее часто встречаемые в английской речи восемь букв можно представить в виде анаграммы ASINTOER (фраза «a sin to err» («грех в заблуждении») без последней буквы). Её то и использовал Зорге в качестве второго шага построения своего шифра. Для этого он нумеровал входящие в анаграмму буквы в своей табличке по порядку сверху вниз и получал новую таблицу:

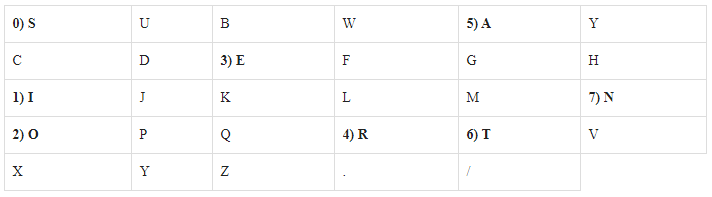


Таблица 2

Саму таблицу мы строим по горизонтали, записывая буквы рядами, а нумеруем по вертикали, столбцами. Так улучшаются перемешивающие свойства.

Далее таблица преобразуется к следующему виду: сначала в строку по столбцам записываются наиболее часто встречаемые буквы в порядке нумерации (S, I, E, …). А затем записываются и все остальные буквы, также по столбцам в строки (С, X, U, D, J, …). Конечной целью разведчиков являлось составление следующего квадратного шифра:

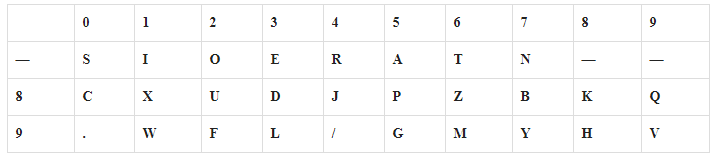


Таблица 3

В верхней строке мы видим наиболее встречаемые в английском языке буквы, которым даны цифровые обозначения от 0 до 7. В две оставшиеся строки выписаны по порядку остальные буквы из таблицы «SUBWAY» (то же сверху вниз). Они получают обозначения в виде двоичных чисел от 80 до 99. Как видно, в верхней строке конечные клетки под номерами 8 и 9 пустые. Эти цифры становятся номерами строк в ключевой таблице. Таким образом, здесь мы имеем воплощение идеи так называемого пропорционального шифра, позволяющее резко уменьшить количество входящих в шифрограмму знаков. В зависимости от размера текста это сокращение доходило до 30%. А это было очень важно для облегчения самого процесса шифровки, затруднения возможной дешифровки противником и уменьшения времени передачи радиограмм. Отделение же в тексте однозначных знаков от двузначных (конечно, при знании кодовой таблицы) не представляет никаких трудностей. Это была великолепная идея неизвестного нам советского криптолога, нашедшая затем в мировой криптографии широкое распространение.

Предположим, нужно зашифровать следующую телеграмму на немецком языке: «DAL.DER SOWJETISCHE FERNE OSTEN KANN ALS SICHER VOR EINEM ANGRIFF JAPANS ERACHTET WERDEN.RAMSAY» [DAL. Советский Дальний Восток может не опасаться нападения Японии. Рамзай.] Каждая радиограмма разведчиков начиналась их «обратным адресом»: DAL. Это были начальные буквы географического названия Дальний Восток. Заменяя буквы, знаки препинания и добавляя разделитель согласно квадратного шифра Зорге, получим:





Таблица 4

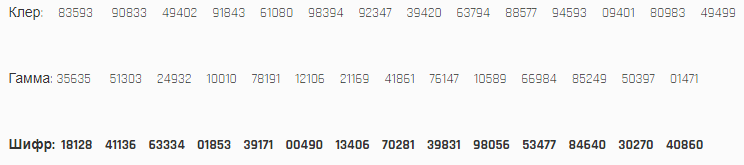
Имея ввиду, что шифротекст разведчики разбивали на 5-ти значные группы, последние цифры криптограммы или дополняли до полной пятёрки нулями, или просто удаляли.

Здесь мы подошли к главному секрету Рамзая. Первоначальная шифровка текста далее перекодировалась методом наложения на него бесконечной одноразовой цифровой гаммы по модулю 10. Способ получения её мог быть абсолютно разным: начиная от использования так называемых одноразовых шифровальных блокнотов до преобразования букв определенного книжного текста в цифры. И тот, и другой способ имели в разведке самое широкое применение и мы это еще увидим. Но для Зорге задачу значительно упростили. В качестве шифровальной книги был выбран толстенный «Немецкий статистический ежегодник за 1935 год», состоящий из сотен числовых таблиц, из которых наугад и выбиралась требуемые гаммы.

Предполагалось, что наличие у разведчиков в Японии подобного справочника никак не могло навести на подозрения. Ведь Р. Зорге был известным немецким журналистом, а его главный помощник и радист М. Клаузен – бизнесменом. Конечно, цифровые последовательности, получаемые с помощью этих таблиц, не были достаточно равномерными. В них неизбежно преобладали некоторые цифры, что вело к их повторению. Тем не менее, такие гаммы имели достаточное разнообразие, и никогда не были успешно преодолены вражескими криптоаналитиками.

Первая половина ежегодника на белой бумаге содержала статистические данные о Германии. Эта часть книги использовалась в качестве основы для кодирования шифрограмм непосредственно самой резидентурой Зорге. Во второй части справочника, на листах зеленого цвета, приводились международные статистические обзоры: ею уже пользовался московский Центр для шифровки ответных радиограмм. Это разделение делалось для предотвращения возможного наложения одинаковых гамм при шифровании текстов в Токио и Москве, что прямо могло привести к дешифровке радиограмм противником. Очевидно, что и сам Зорге и его помощник должны были делать в тексте своей кодовой книги какие-то пометки для недопущения всё того же повтора ключа. При аресте в 1941 году в квартирах Р. Зорге и М. Клаузена японской полицией были обнаружены совершенно одинаковые справочники с подозрительными отметками. Что сразу навело контрразведку на ключевую книгу пойманных шпионов.

Итак, цифры гаммы поочередно выбирались из справочника и выписывались под цифрами шифротекста, затем шло по-значковое сложение цифр клера и ключа по модулю 10. То есть, при сложении цифр во внимание принимались только единицы суммы, а десятки отбрасывались.



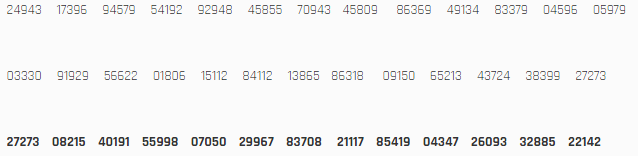


Таблица 5

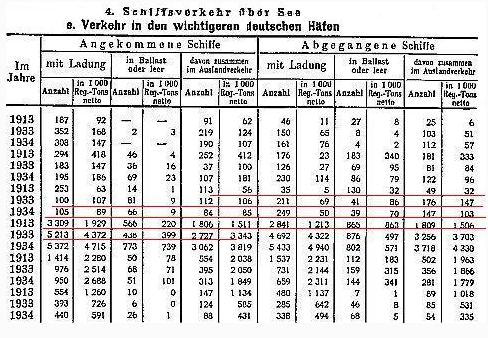


Рис.1 «Statistischen Jahrbuchs fuer das deutsche Reich» Jahrgang 1935 (фрагмент) (стр. 193, 7 строка таблицы, 5 колонка –. подчёркнуты ключевые гаммы)

Место справочника, с которого начиналась выборка очередной гаммы, обозначалось пятизначной группой и добавлялось в текст шифрограммы. Первые три цифры являлись номером страницы, следующая цифра обозначала строку в таблице на этой странице, а последняя цифра – номер колонки на странице, где располагались нужные цифры (без учёта первого столбца).

Например, пусть разведчики начинали выборку гаммы с 193 страницы седьмой строки пятого столбца. Обозначалось это как 19375. Для еще большей надежности они никогда не брали первые цифры, а всегда начинали шифрование с последнего знака соответствующей колонки. Но в таком виде ключевая группа не оставлялась, а проходила определенную обработку. Для этого к ней опять же по модулю 10 прибавлялась четвертая «пятерка» с начала и третья «пятерка» с конца каждой новой шифровки. Получившуюся сумму помещали в начале криптограммы, как индикатор к расшифровке всего текста.

В нашем случае:

01853  –  четвертая группа от начала криптограммы  
+ 26093  –  третья группа от конца криптограммы  
+ 19375  –  страница/строка/колонка  
36111 – ключевая группа – индикатор

Отдельно следует объяснить, как шла передача цифрового текста. Числа выделялись в шифрограммах разделителем 94 с двух сторон, а сами цифры писались сдвоенными. Например:

W H O C O M M A N D S / 5 3 / A R M Y

91 98 2 80 2 96 96 5 7 83 0 94 55 33 94 5 4 96 97

(Кто командует 53 армией?)

Перехват радиосообщений Зорге велся японской полицией в течение нескольких лет, колонки загадочных пятизначных групп аккуратно подшивались в досье не пойманных шпионов. Но до самого конца японские эксперты не смогли прочесть не единой их шифрограммы. И только арестованный радист группы Макс Клаузен осенью 1941 года прояснил контрразведке систему своего шифра. Не вдаваясь в причины этого прискорбного факта, акцентируем внимание на другом – времени его появления в арсенале разведчиков.

Описание работы

***Цель работы:***

Моделирование шифровальной машины, которая позволит зашифровать и расшифровывать текст шифром Рихарда Зорге.

***Ход работы:***

В среде разработки Microsoft Visual Studio было создано приложение Windows Forms, для написания программы использовался язык С#.

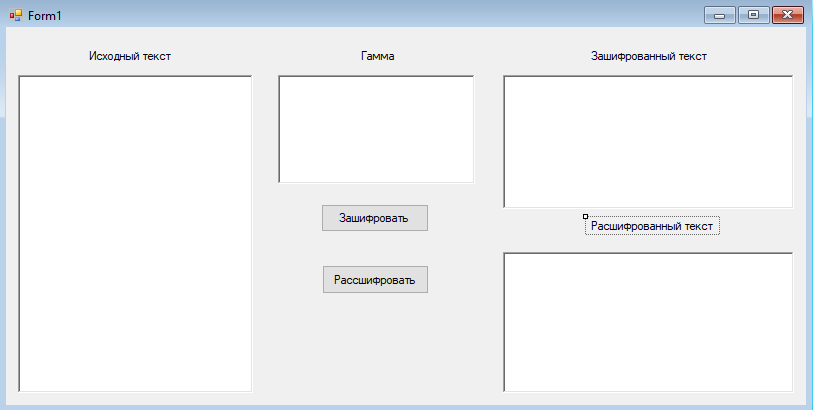


Рис.2 Интерфейс программы

Рассмотрим процесс шифрования текста поэтапно:

1. Нажатии кнопки “Зашифровать” создается двумерный массив символов **tab** в который мы помещаем ключевую Таблицу 3. Заводим массив символов **tx**, в котором будет храниться исходный текст, и строку **cip**, в которую будет записываться клер.

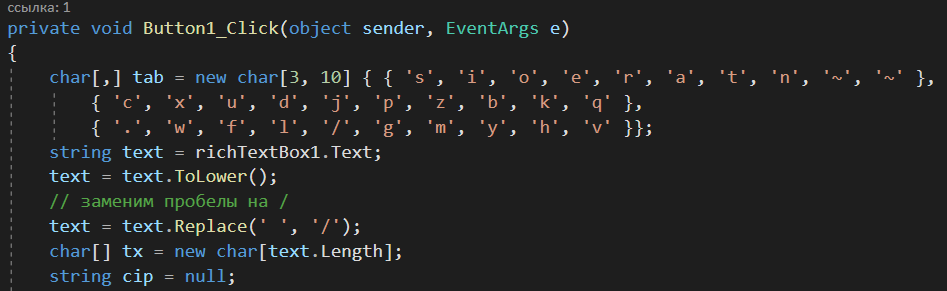


Рис. 3 Введение необходимых переменных

1. Шифруем каждый символ исходного теста. Цифры с использованием уже описанной ранее техники : цифры удваиваются, например,51 = 5511, а затем отделяются индикатором 94 с обоих сторон - 51 = 94551194.

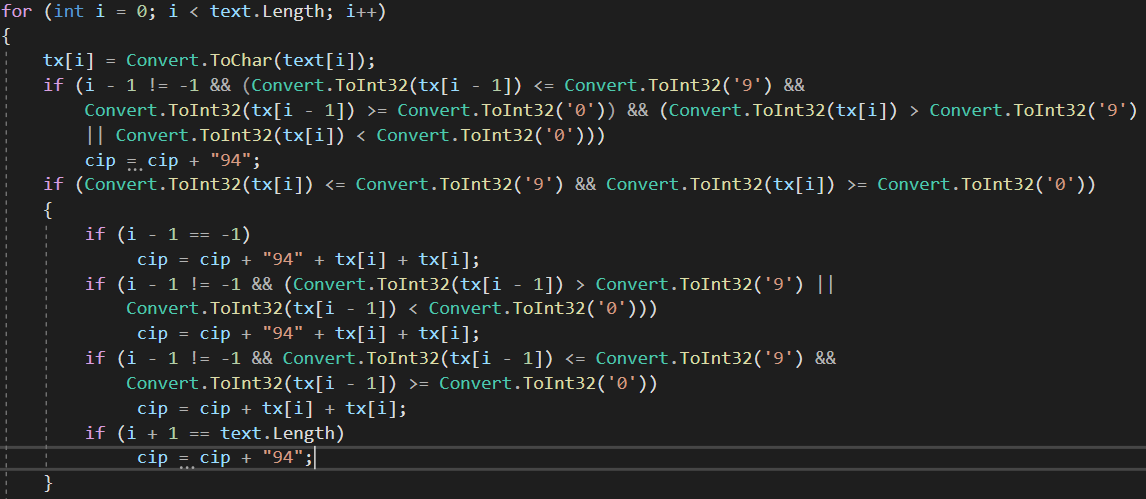


Рис. 4 Шифрование цифр

1. Шифрование букв и символов с использованием ключевой таблицы **tab**.

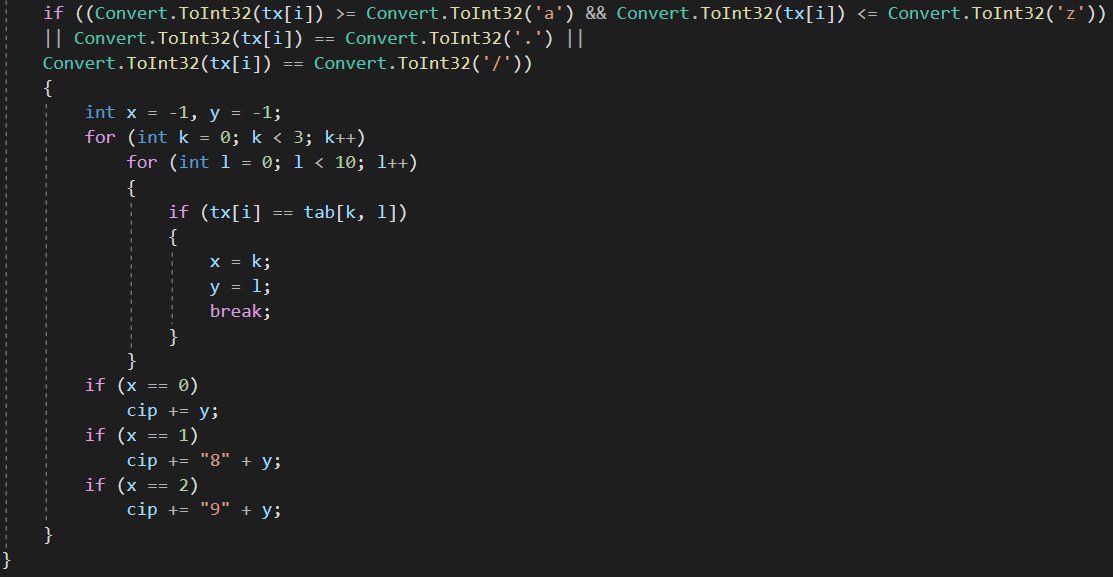


Рис. 5 Шифрование символов

1. Гамма записывается в соответствующее поле на форме. Заводим переменные **cip1** и **cip2** и помещаем в них клер и гамму соответственно. Выполняем сложение по модулю 10 и записываем результат в переменную **cip3**, который уже является конечным шифром. Помещаем его в соответствующее поле на форме.

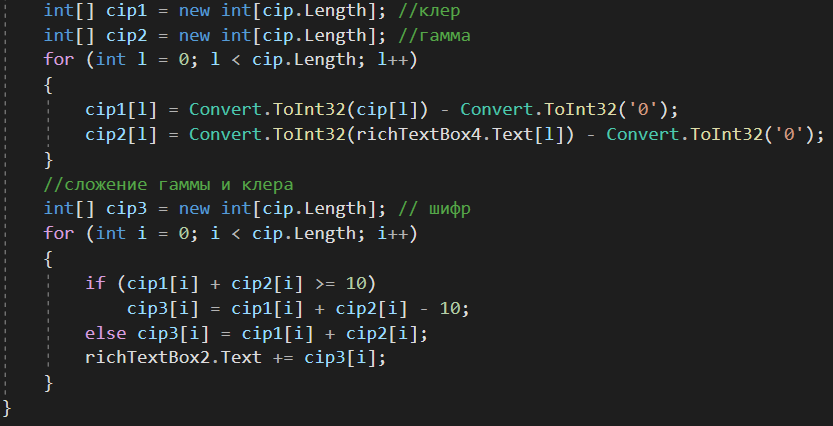


Рис. 6 Сложение гаммы и клера

Теперь рассмотрим поэтапно процесс расшифровки:

1. Так же как при шифровке создаем ключевую таблицу **tab** и переменные **cip1**, **cip2**, **cip3**, куда будут помещаться клер, гамма и шифр соответственно.

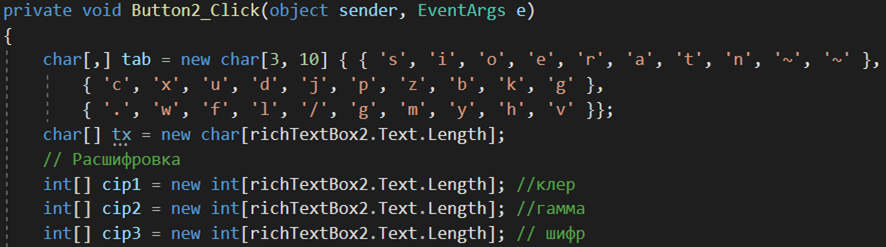


Рис. 7 Введение переменных

1. Выполняем операцию обратную к операции сложение по модулю 10.

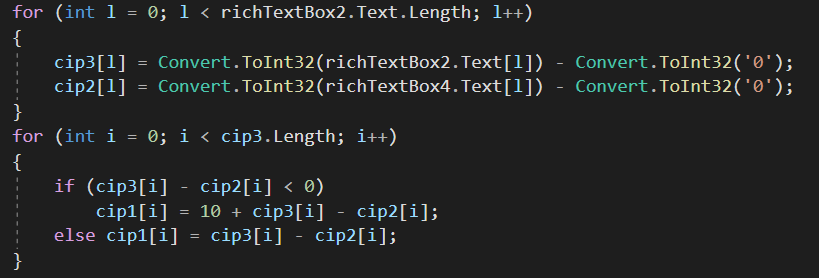


Рис. 8 Вычитание по модулю 10

1. Так как и пробел, и индикатор чисел кодируется как “94”, то для расшифровки необходимо определить, что именно обозначает в данном случае «94». Для этого значения массива **cip1** запишем в строку **str1,** затем заменим все вхождения «94» на «\*» и создадим из полученной строки символьный массив **str**, для дальнейшего удобства работы.

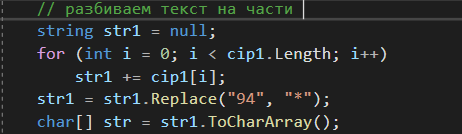


Рис. 9 Замена индикатора

В получившемся массиве проверяем повторение цифр между «\*», чтобы отличить пробел от индикатора чисел, последний помечаем «#», пробелы так и остаются «\*».

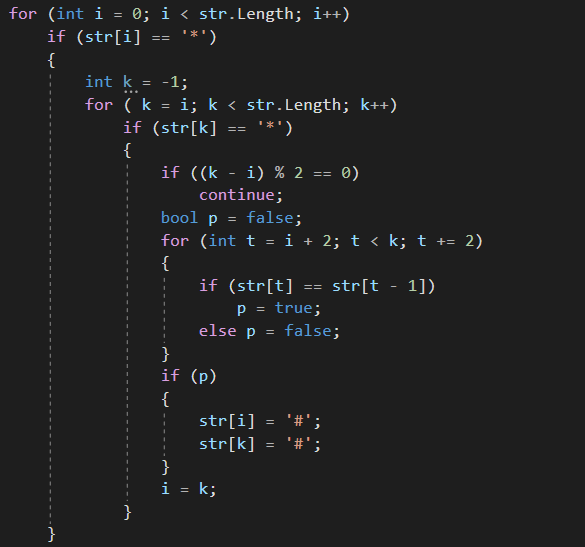


Рис. 10 Определение пробелов и индикаторов

1. Полученный клер **str,** расшифровываем в соответствии с ключевой таблицей **tab**.В переменную **s** будем записывать расшифрованный текст.

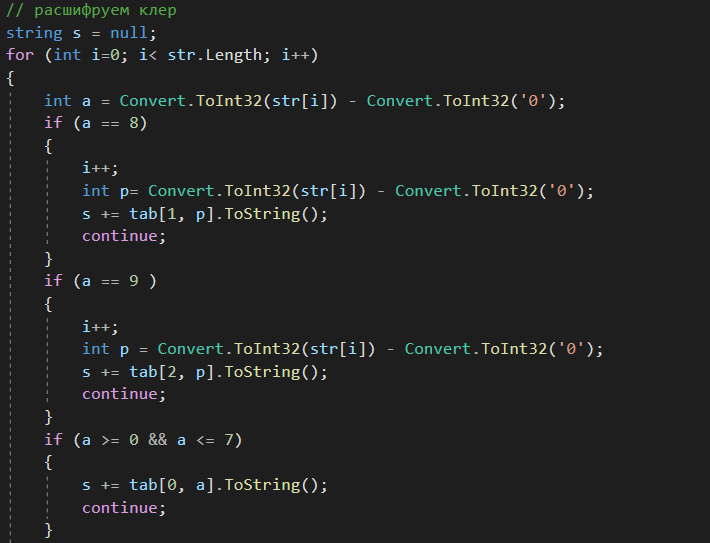


Рис.11 Расшифровка символов исходного текста

Используя разграничительные символы «\*» и «#» расшифровываем числа и пробелы. Выводим расшифрованный текст на форму.

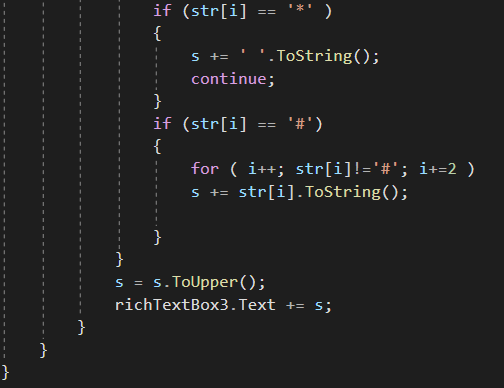


Рис.12 Расшифровка чисел и пробелов

Результаты работы программы

1. Проверим работу программы на разобранном выше примере с посланием на немецком языке: «DAL.DER SOWJETISCHE FERNE OSTEN KANN ALS SICHER VOR EINEM ANGRIFF JAPANS ERACHTET WERDEN.RAMSAY» [DAL. Советский Дальний Восток может не опасаться нападения Японии. Рамзай.]

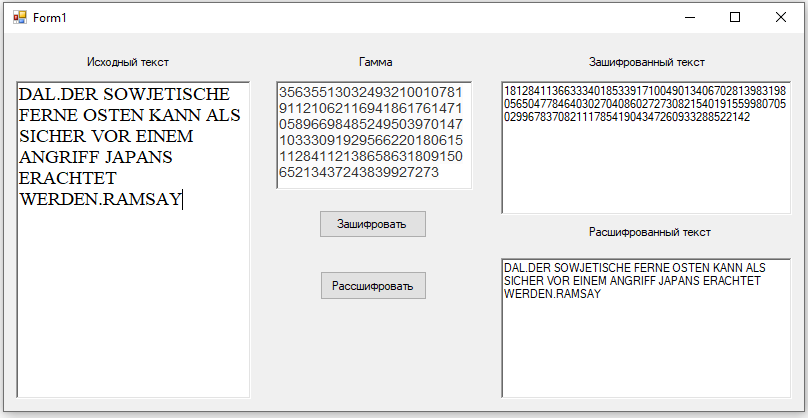


Рис. 13 Результат работы программы

Видим, что используя ту же гамму, что и в примере, мы получаем шифр, который сходится с теоретически рассчитанным в Таблице 5.

1. Зашифруем другое сообщение с использованием новой гаммы.

Возьмем исходный текст:

Mr. X will fly tomorrow.

Расставим слеши для разделения слов:

Mr./X/will/fly/tomorrow.

Зашифруем текс используя ключевую таблицу:

9649094 81 94  911 93939492 9397946 29624 429190

M R. / X / W I L L / F L Y / T OM OR ROW .

Теперь из «Немецкого статистического ежегодника» возьмем произвольно выранную199-ю страницу, четвертую строку, четвертый столбец.

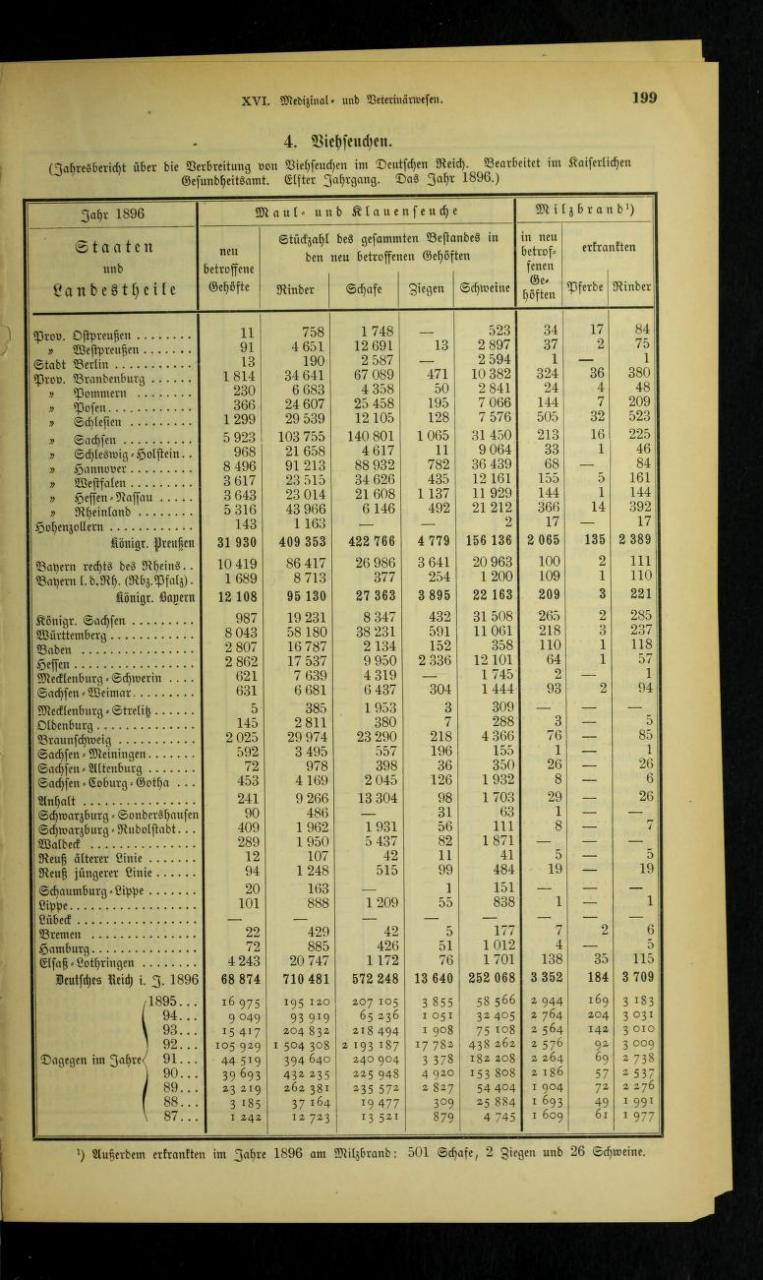


Рис. 14 Немецкий статистический ежегодник

Отсюда начнется нужная гамма:

324 36 380 230 6683 4358 50 2841

Складываем гамму и клер:

96490+324, 94819+36, 49119+380, 39394+230, 92939+6683, 79462+4358, 96244+50, 29190+2841

Получаем зашифрованный текст:

96714 94845 49499 39524 98512 73710 96294 21931.

Проверим работу программы с данными текстом и гаммой.

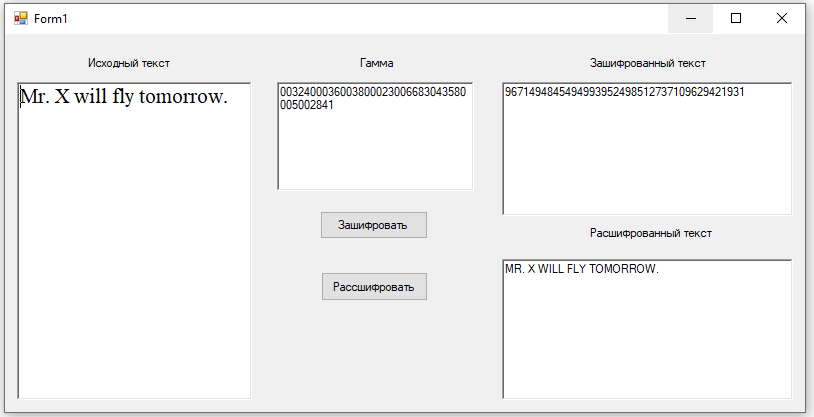


Рис.15 Результат работы программы

Видим, что практический результат полностью соответствует теоретически рассчитанному результату.

Код программы

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 WindowsFormsApp2

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void Button1\_Click(object sender, EventArgs e)

{

char[,] tab = new char[3, 10] { { 's', 'i', 'o', 'e', 'r', 'a', 't', 'n', '~', '~' },

{ 'c', 'x', 'u', 'd', 'j', 'p', 'z', 'b', 'k', 'q' },

{ '.', 'w', 'f', 'l', '/', 'g', 'm', 'y', 'h', 'v' }};

string text = richTextBox1.Text;

text = text.ToLower();

// заменим пробелы на /

text = text.Replace(' ', '/');

char[] tx = new char[text.Length];

string cip = null;

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

{

tx[i] = Convert.ToChar(text[i]);

if (i - 1 != -1 && (Convert.ToInt32(tx[i - 1]) <= Convert.ToInt32('9') &&

Convert.ToInt32(tx[i - 1]) >= Convert.ToInt32('0')) && (Convert.ToInt32(tx[i]) > Convert.ToInt32('9')

|| Convert.ToInt32(tx[i]) < Convert.ToInt32('0')))

cip = cip + "94";

if (Convert.ToInt32(tx[i]) <= Convert.ToInt32('9') && Convert.ToInt32(tx[i]) >= Convert.ToInt32('0'))

{

if (i - 1 == -1)

cip = cip + "94" + tx[i] + tx[i];

if (i - 1 != -1 && (Convert.ToInt32(tx[i - 1]) > Convert.ToInt32('9') ||

Convert.ToInt32(tx[i - 1]) < Convert.ToInt32('0')))

cip = cip + "94" + tx[i] + tx[i];

if (i - 1 != -1 && Convert.ToInt32(tx[i - 1]) <= Convert.ToInt32('9') &&

Convert.ToInt32(tx[i - 1]) >= Convert.ToInt32('0'))

cip = cip + tx[i] + tx[i];

if (i + 1 == text.Length)

cip = cip + "94";

}

if ((Convert.ToInt32(tx[i]) >= Convert.ToInt32('a') && Convert.ToInt32(tx[i]) <= Convert.ToInt32('z'))

|| Convert.ToInt32(tx[i]) == Convert.ToInt32('.') ||

Convert.ToInt32(tx[i]) == Convert.ToInt32('/'))

{

int x = -1, y = -1;

for (int k = 0; k < 3; k++)

for (int l = 0; l < 10; l++)

{

if (tx[i] == tab[k, l])

{

x = k;

y = l;

break;

}

}

if (x == 0)

cip += y;

if (x == 1)

cip += "8" + y;

if (x == 2)

cip += "9" + y;

}

}

int[] cip1 = new int[cip.Length]; //клер

int[] cip2 = new int[cip.Length]; //гамма

for (int l = 0; l < cip.Length; l++)

{

cip1[l] = Convert.ToInt32(cip[l]) - Convert.ToInt32('0');

cip2[l] = Convert.ToInt32(richTextBox4.Text[l]) - Convert.ToInt32('0');

}

//сложение гаммы и клера

int[] cip3 = new int[cip.Length]; // шифр

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

{

if (cip1[i] + cip2[i] >= 10)

cip3[i] = cip1[i] + cip2[i] - 10;

else cip3[i] = cip1[i] + cip2[i];

richTextBox2.Text += cip3[i];

}

}

private void Button2\_Click(object sender, EventArgs e)

{

char[,] tab = new char[3, 10] { { 's', 'i', 'o', 'e', 'r', 'a', 't', 'n', '~', '~' },

{ 'c', 'x', 'u', 'd', 'j', 'p', 'z', 'b', 'k', 'g' },

{ '.', 'w', 'f', 'l', '/', 'g', 'm', 'y', 'h', 'v' }};

char[] tx = new char[richTextBox2.Text.Length];

// Расшифровка

int[] cip1 = new int[richTextBox2.Text.Length]; //клер

int[] cip2 = new int[richTextBox2.Text.Length]; //гамма

int[] cip3 = new int[richTextBox2.Text.Length]; // шифр

for (int l = 0; l < richTextBox2.Text.Length; l++)

{

cip3[l] = Convert.ToInt32(richTextBox2.Text[l]) - Convert.ToInt32('0');

cip2[l] = Convert.ToInt32(richTextBox4.Text[l]) - Convert.ToInt32('0');

}

for (int i = 0; i < cip3.Length; i++)

{

if (cip3[i] - cip2[i] < 0)

cip1[i] = 10 + cip3[i] - cip2[i];

else cip1[i] = cip3[i] - cip2[i];

}

// разбиваем текст на части

string str1 = null;

for (int i = 0; i < cip1.Length; i++)

str1 += cip1[i];

str1 = str1.Replace("94", "\*");

char[] str = str1.ToCharArray();

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

if (str[i] == '\*')

{

int k = -1;

for ( k = i; k < str.Length; k++)

if (str[k] == '\*')

{

if ((k - i) % 2 == 0)

continue;

bool p = false;

for (int t = i + 2; t < k; t += 2)

{

if (str[t] == str[t - 1])

p = true;

else p = false;

}

if (p)

{

str[i] = '#';

str[k] = '#';

}

i = k;

}

}

// расшифруем клер

string s = null;

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

{

int a = Convert.ToInt32(str[i]) - Convert.ToInt32('0');

if (a == 8)

{

i++;

int p= Convert.ToInt32(str[i]) - Convert.ToInt32('0');

s += tab[1, p].ToString();

continue;

}

if (a == 9 )

{

i++;

int p = Convert.ToInt32(str[i]) - Convert.ToInt32('0');

s += tab[2, p].ToString();

continue;

}

if (a >= 0 && a <= 7)

{

s += tab[0, a].ToString();

continue;

}

if (str[i] == '\*' )

{

s += ' '.ToString();

continue;

}

if (str[i] == '#')

{

for ( i++; str[i]!='#'; i+=2 )

s += str[i].ToString();

}

}

s = s.ToUpper();

richTextBox3.Text += s;

}

}

}