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 №6**

|  |
| --- |
| **Шифр Ришелье** |
| *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:**

Реализовать шифр Ришелье на любом языке программирования.

**Теория:**

Шифр, использовавшийся Ришелье представляет собой шифр перестановки, при котором открытый текст разбивается на отрезки, а внутри каждого отрезка буквы переставляются в соответствии с фиксированной перестановкой.

**Исходный код:**

**Проверка ключа на корректность:**

private int Check\_key\_func(string entered\_key)

{

string alf = "123456789(),"; //Допустимые символы в ключе

if (entered\_key[0] != '(') return -1;

int opened\_breaks = 1;

for (int i = 1; i < entered\_key.Length; i++)

{

if (alf.IndexOf(entered\_key[i]) == -1) return -1;

if (entered\_key[i] == ')') opened\_breaks--;

if (entered\_key[i] == '(') opened\_breaks++;

if (opened\_breaks>1 || opened\_breaks<0) return -1;

}

if (opened\_breaks > 0) return -1;

var swaps = (entered\_key).Split(new char[] { '(', ')' }, StringSplitOptions.RemoveEmptyEntries);

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

{

var local\_swaps = (swaps[i]).Split(new char[] { ','}, StringSplitOptions.RemoveEmptyEntries);

int[] times = new int[local\_swaps.Length + 1];//Подсчет вхождений каждой цифры в каждой отдельной перестановке ключа

Array.Clear(times, 0, local\_swaps.Length + 1);

for (int j = 0; j < local\_swaps.Length; j++)

{

times[Convert.ToInt32(local\_swaps[j])]++;

if (Convert.ToInt32(local\_swaps[j]) >= 100000) return -1;

}

if (times[0] != 0) return -1;

for (int j = 1; j <= local\_swaps.Length; j++)

if (times[j] != 1) return -1;

}

return 0;

}

**Алгоритм зашифрования:**

private void f6\_ButtonEncrypt\_Click(object sender, EventArgs e)

{

var Source = f6\_fieldOriginal.Text;

if (Check\_key\_func(f6\_fieldKey.Text) == -1)

{

MessageBox.Show("Некорректный ключ.", "Ошибка");

goto metka\_exit;

}

string result = "";

var swaps = (f6\_fieldKey.Text).Split(new char[] { '(', ')' }, StringSplitOptions.RemoveEmptyEntries);

int curPos = 0;

int brackets = 0; //Номер скобок в ключе

while (curPos < Source.Length)

{

var local\_swaps = swaps[brackets].Split(new char[] { ',' }, StringSplitOptions.RemoveEmptyEntries);

if (curPos + local\_swaps.Length > Source.Length)

{

var tail = Source.Skip(curPos).Take(Source.Length - curPos).ToArray();

result += new string (tail);

break;

}

var text = Source.Skip(curPos).Take(local\_swaps.Length).ToArray();

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

for (int i = 0; i < local\_swaps.Length; i++)

Encrypted[Convert.ToInt32(local\_swaps[i]) - 1] = text[i];

result += new string(Encrypted);

curPos += local\_swaps.Length;

if (++brackets == swaps.Length) brackets = 0;

}

f6\_fieldCiphertext.Text = result;

metka\_exit:;

}

**Алгоритм расшифрования:**

private void f6\_ButtonDecrypt\_Click(object sender, EventArgs e)

{

var Source = f6\_fieldCiphertext.Text;

if (Check\_key\_func(f6\_fieldKey.Text) == -1)

{

MessageBox.Show("Некорректный ключ.", "Ошибка");

goto metka\_exit;

}

string result = "";

var swaps = (f6\_fieldKey.Text).Split(new char[] { '(', ')' }, StringSplitOptions.RemoveEmptyEntries);

int curPos = 0;

int brackets = 0; //Номер скобок в ключе

while (curPos < Source.Length)

{

var local\_swaps = swaps[brackets].Split(new char[] { ',' }, StringSplitOptions.RemoveEmptyEntries);

if (curPos + local\_swaps.Length > Source.Length)

{

var tail = Source.Skip(curPos).Take(Source.Length - curPos).ToArray();

result += new string(tail);

break;

}

var text = Source.Skip(curPos).Take(local\_swaps.Length).ToArray();

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

for (int i = 0; i < local\_swaps.Length; i++)

Encrypted[i] = text[Convert.ToInt32(local\_swaps[i]) - 1];

result += new string(Encrypted);

curPos += local\_swaps.Length;

if (++brackets == swaps.Length) brackets = 0;

}

f6\_fieldOriginal.Text = result;

metka\_exit:;

}

**Результат работы программы:**

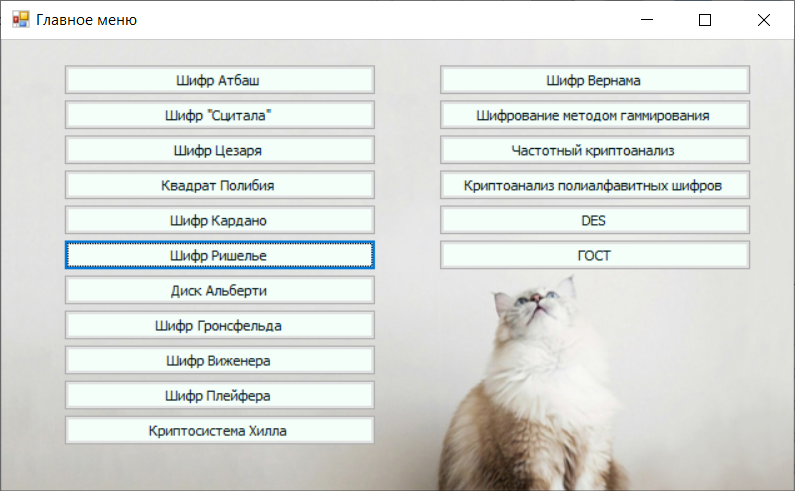


Рисунок 1. Главное меню программы.

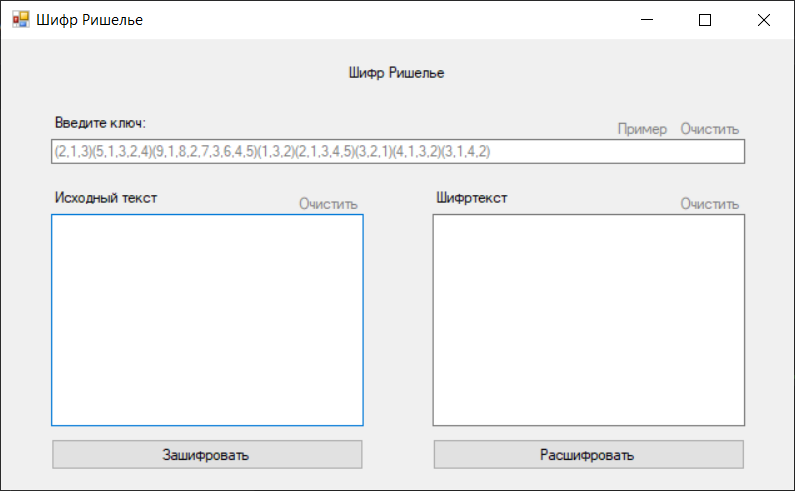


Рисунок 2. Окно шифра Ришелье.

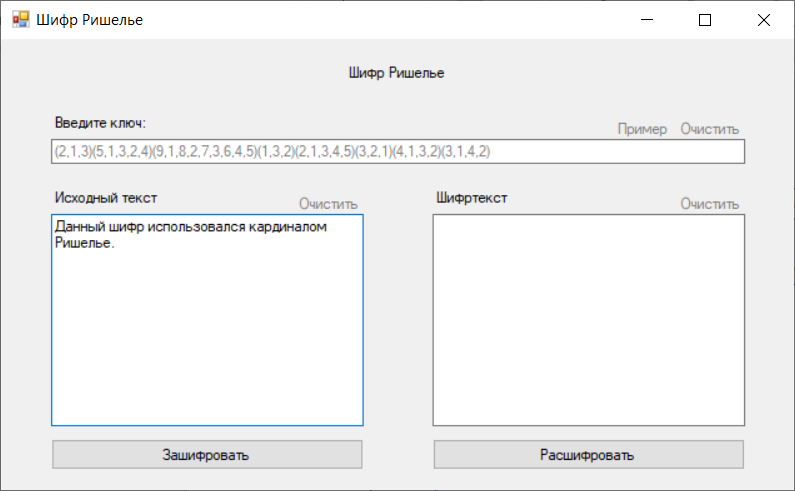


Рисунок 3. Исходный текст.

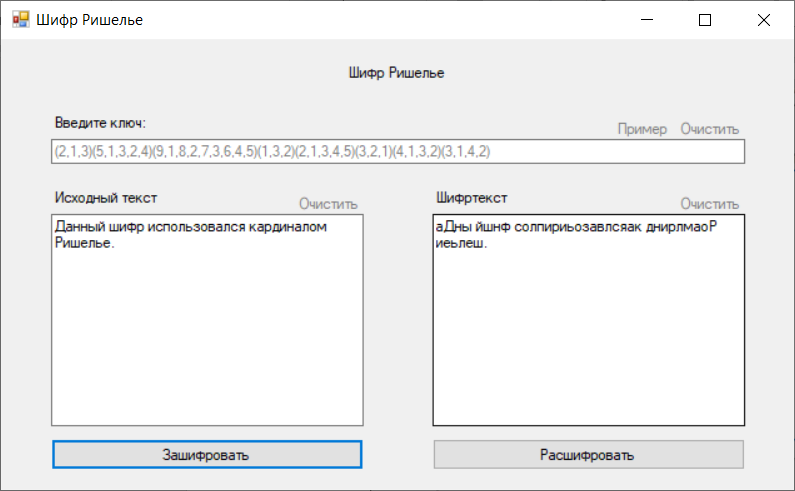


Рисунок 4. Шифртекст (результат зашифрования).

Очистим поле Исходный текст и расшифруем сообщение. Ключ необходимо оставить неизменным.

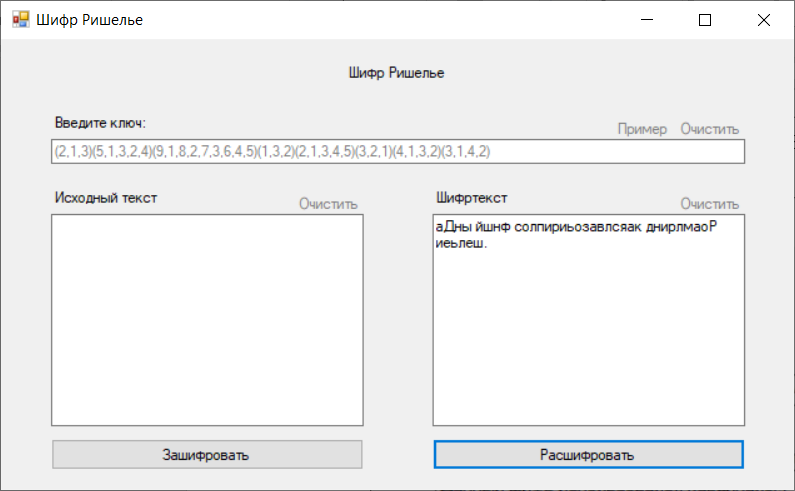


Рисунок 5. Очистка поля Исходный текст.

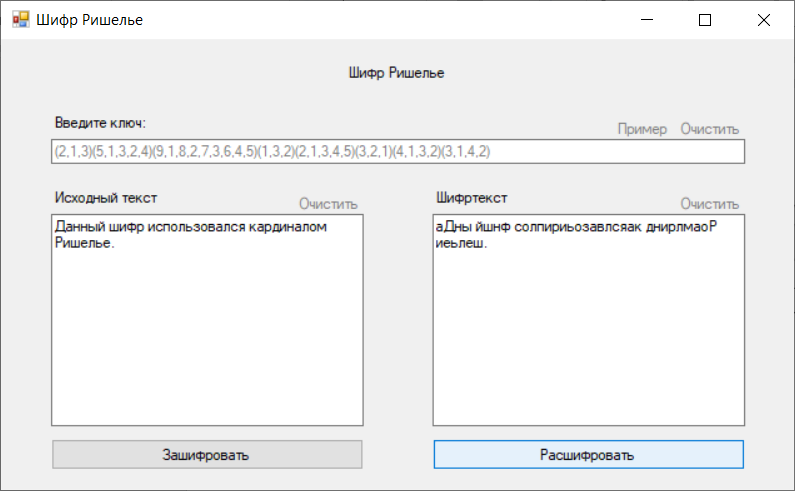


Рисунок 6. Исходный текст (результат расшифрования).

**Вывод:**

В ходе выполнения лабораторной работы я ознакомилась с алгоритмом шифра Ришелье, реализовала на практике программу по зашифрованию и расшифрованию данным шифром на языке C#. Данные навыки я могу применить при реализации других шифров и при дальнейшем использовании шифра Ришелье.