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 №5**

|  |
| --- |
| **Шифр Кардано** |
| *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 Кардано на любом языке программирования.

**Теория:**

Решетка — квадрат {\displaystyle N\times N}NxN клеток, некоторые из которых вырезаны (в том числе, возможно, и имеющие общую сторону или вершину). Клетки должны иметь такой размер, чтобы в каждую помещалась ровно одна буква. Вырезанные клетки должны располагаться таким образом, чтобы никакие две из них не оказывались в одном и том же месте при поворотах решетки. Чтобы зашифровать сообщение, нужно разместить решетку на бумаге и вписать часть текста в вырезанные клетки, затем повернуть решетку на 90° и вписать следующую часть и т. д. После этого в оставшиеся пустыми места на бумаге нужно вписать произвольные символы (разумеется, для лучшей маскировки нужно использовать символы из того же алфавита, что и символы, из которых состоит сообщение).

**Исходный код:**

**Генерация ключа:**

private void f5\_ButtonGenKey\_Click(object sender, EventArgs e)

{

Random rand = new Random();

int key = Convert.ToInt32(numericUpDown1.Value);

key = (key \* key) / 4;

int[] sequence = Enumerable.Range(1, key).OrderBy(y => rand.Next(key)).Take(key).ToArray();

String seq = null;

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

{

seq += sequence[i];

seq += ' ';

}

f5\_fieldKey.Text = seq;

}

**Проверка и преобразование ключа:**

private int KeyToMatrix()

{

f5\_fieldKey.Text = Regex.Replace(f5\_fieldKey.Text, "[^0-9 ]", "");

int[] sequence = f5\_fieldKey.Text.Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries).Select(x => int.Parse(x)).ToArray(); ;

int lines = Convert.ToInt32(Math.Sqrt(sequence.Length\*4));

if (lines < 4)

{

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

return -1;

}

Key\_Matrix = new int[lines, lines];

numericUpDown1.Value = lines;

foreach (int a in sequence)

{

int temp = 0;

for (int i = 0; i < lines / 2; i++)

{

for (int j = 0; j < lines / 2; j++)

{

temp++;

if (temp == a)

{

Key\_Matrix[i, j] = 1;

Key\_Matrix = Matrix\_Rotation(Key\_Matrix);

goto metka\_cont;

}

}

}

metka\_cont:

continue;

}

int[,] tmpMatrix = new int[lines, lines];

//Обозначим порядок элементов на этой итерации

int times = 0;

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

{

for (int j = 0; j < lines; j++)

{

if (Key\_Matrix[i, j] == 1)

Key\_Matrix[i, j] += times++;

}

}

times++;

//Расставим метки и повернем вспомогательную матрицу

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

{

for (int j = 0; j < lines; j++)

{

if (Key\_Matrix[i, j] > 0)

{

tmpMatrix[i, j] = times;

}

else

tmpMatrix[i, j] = 0;

}

}

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

{

//Расставим метки и повернем вспомогательную матрицу

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

{

for (int j = 0; j < lines; j++)

{

if (tmpMatrix[i, j] > 0)

{

tmpMatrix[i, j] = times;

}

else

tmpMatrix[i, j] = 0;

}

}

tmpMatrix = Matrix\_Rotation(tmpMatrix);

//Переносим значения со вспомогательной матрицы

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

{

for (int j = 0; j < lines; j++)

{

if (tmpMatrix[i, j] == times)

Key\_Matrix[i, j] = times;

}

}

int tmpTimes = 0;

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

{

for (int j = 0; j < lines; j++)

{

if (Key\_Matrix[i, j] == times)

Key\_Matrix[i, j] += tmpTimes++;

}

}

times += tmpTimes;

}

bool [] check = new bool[lines \* lines+1];

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

{

check[i] = false;

}

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

{

for (int j = 0; j < lines; j++)

{

check[Key\_Matrix[i, j]] = true;

}

}

for (int i = 1; i < check.Length; i++)

{

if (!check[i])

{

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

Key\_Matrix = null;

return -1;

}

}

x\_seq = new int[lines \* lines];

y\_seq = new int[lines \* lines];

int finding\_coords = 0;

while (finding\_coords < lines\*lines)

{

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

{

for (int j = 0; j < lines; j++)

{

if (Key\_Matrix[i, j] == finding\_coords + 1)

{

x\_seq[finding\_coords] = i;

y\_seq[finding\_coords] = j;

}

}

}

finding\_coords++;

}

return 0;

}

**Алгоритм зашифрования:**

private void f5\_ButtonEncrypt\_Click(object sender, EventArgs e)

{

string alf\_ru = "абвгдеёжзийклмнопрстуфхцчшщъыьэюя";

string alf\_en = "abcdefjhigklmnopqrstuvwxyz";

string ALF\_RU = "АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ";

string ALF\_EN = "ABCDEFJHIGKLMNOPQRSTUVWXYZ";

string alf\_numeric = "1234567890";

string Source = f5\_fieldOriginal.Text;

//Source = Regex.Replace(f5\_fieldOriginal.Text, "[^a-zA-Zа-яА-ЯёЁ]", ""); //разрешить на вход только буквы

if (Source.Length == 0 && f5\_fieldOriginal.TextLength > 0)

{

MessageBox.Show("Исходный текст должен содержать символы русского или английского алфавита. Прочие символы недопустимы и будут удалены.", "Ошибка"); //Выводится только при запрете ввода символов

goto metka\_exit;

}

string result = "";

if (KeyToMatrix() == -1) goto metka\_exit;

//Определение алфавита для мусорных символов

int ru\_symbols = 0, en\_symbols = 0, RU\_symbols = 0, EN\_symbols = 0, num\_symbols = 0;

for (int i = 0; i < Source.Length; i++) //Подсчет количества входящих в исходный текст символов каждого алфавита

{

if (alf\_ru.IndexOf(Source[i]) >= 0) ru\_symbols++;

if (ALF\_RU.IndexOf(Source[i]) >= 0) RU\_symbols++;

if (alf\_en.IndexOf(Source[i]) >= 0) en\_symbols++;

if (ALF\_EN.IndexOf(Source[i]) >= 0) EN\_symbols++;

if (alf\_numeric.IndexOf(Source[i]) >= 0) num\_symbols++;

}

//Создание решетки Кардано

int sizeOfGrid = Convert.ToInt32(numericUpDown1.Value);

char[,] CardanoGrid = new char[sizeOfGrid, sizeOfGrid];

Random rnd = new Random();

int amount\_of\_encrypted\_symbols = 0; //Количество уже зашифрованных символов

while (amount\_of\_encrypted\_symbols < Source.Length)

{

//Подготовка решетки (заполнение мусором)

if (en\_symbols >= ru\_symbols && en\_symbols >= EN\_symbols && en\_symbols >= RU\_symbols && en\_symbols >= num\_symbols) //заполняем строчным английским мусором

{

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

for (int j = 0; j < sizeOfGrid; j++)

CardanoGrid[i, j] = alf\_en[rnd.Next(26)];

}

else if (ru\_symbols >= EN\_symbols && ru\_symbols >= RU\_symbols && ru\_symbols >= num\_symbols) //заполняем строчным русским мусором

{

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

for (int j = 0; j < sizeOfGrid; j++)

CardanoGrid[i, j] = alf\_ru[rnd.Next(33)];

}

else if (RU\_symbols >= EN\_symbols && RU\_symbols >= num\_symbols) //заполняем заглавным русским мусором

{

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

for (int j = 0; j < sizeOfGrid; j++)

CardanoGrid[i, j] = ALF\_RU[rnd.Next(33)];

}

else if (EN\_symbols >= num\_symbols) //заполняем заглавным английским мусором

{

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

for (int j = 0; j < sizeOfGrid; j++)

CardanoGrid[i, j] = ALF\_EN[rnd.Next(26)];

}

else //Заполняем мусорными цифрами

{

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

for (int j = 0; j < sizeOfGrid; j++)

CardanoGrid[i, j] = alf\_numeric[rnd.Next(alf\_numeric.Length)];

}

//Заполнение решетки исходным текстом

int curSymb = 0;

while (curSymb < Source.Length - amount\_of\_encrypted\_symbols && curSymb < (sizeOfGrid \* sizeOfGrid))

{

CardanoGrid[x\_seq[curSymb], y\_seq[curSymb]] = Source[curSymb+amount\_of\_encrypted\_symbols];

curSymb++;

}

//Считывание шифртекста с решетки

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

for (int j = 0; j < sizeOfGrid; j++)

{

result = result + CardanoGrid[i, j];

}

amount\_of\_encrypted\_symbols += (sizeOfGrid \* sizeOfGrid);

}

f5\_fieldCiphertext.Text = result;

metka\_exit:;

}

**Алгоритм расшифрования:**

private void f5\_ButtonDecrypt\_Click(object sender, EventArgs e)

{

string Source = f5\_fieldCiphertext.Text;

int sizeOfGrid = Convert.ToInt32(numericUpDown1.Value);

//Source = Regex.Replace(f5\_fieldCiphertext.Text, "[^a-zA-Zа-яА-ЯёЁ]", ""); //разрешить на вход только буквы

if ((Source.Length == 0 || Source.Length % (sizeOfGrid \* sizeOfGrid) > 0)) //Если шифртекст не содержит натуральное число блоков

{

MessageBox.Show("Шифртекст должен содержать как минимум 1 блок из n\*n символов.", "Ошибка");

goto metka\_exit;

}

string result = "";

if (KeyToMatrix() == -1) goto metka\_exit;

//Создание решетки Кардано

sizeOfGrid = Convert.ToInt32(numericUpDown1.Value);

char[,] CardanoGrid = new char[sizeOfGrid, sizeOfGrid];

Random rnd = new Random();

int amount\_of\_decrypted\_symbols = 0;//Количество уже расшифрованных символов

while (amount\_of\_decrypted\_symbols < Source.Length)

{

//Заполнение решетки шифртекстом

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

for (int j = 0; j < sizeOfGrid; j++)

{

CardanoGrid[i, j] = Source[amount\_of\_decrypted\_symbols+i\*sizeOfGrid+j];

}

//Считывание с решетки исходного текста

int curSymb = 0;

while (curSymb < (sizeOfGrid \* sizeOfGrid))

{

result += CardanoGrid[x\_seq[curSymb], y\_seq[curSymb]];

curSymb++;

}

amount\_of\_decrypted\_symbols += (sizeOfGrid \* sizeOfGrid);

}

f5\_fieldOriginal.Text = result;

metka\_exit:;

}

**Результат работы программы:**

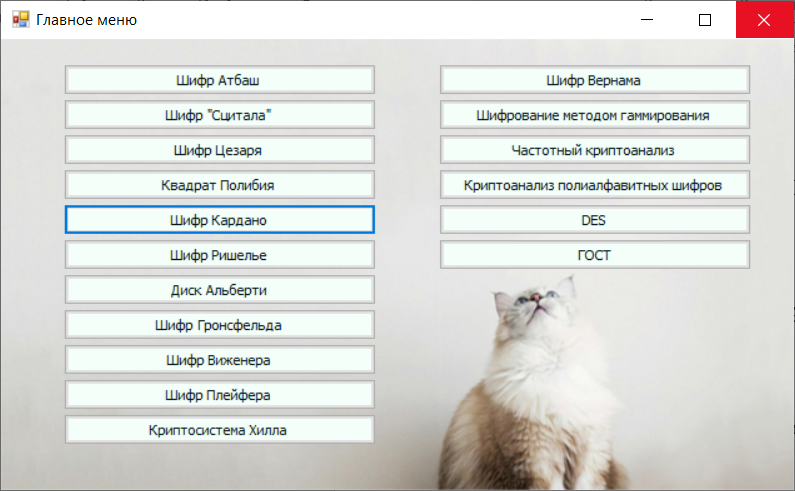


Рисунок 1. Главное меню программы.

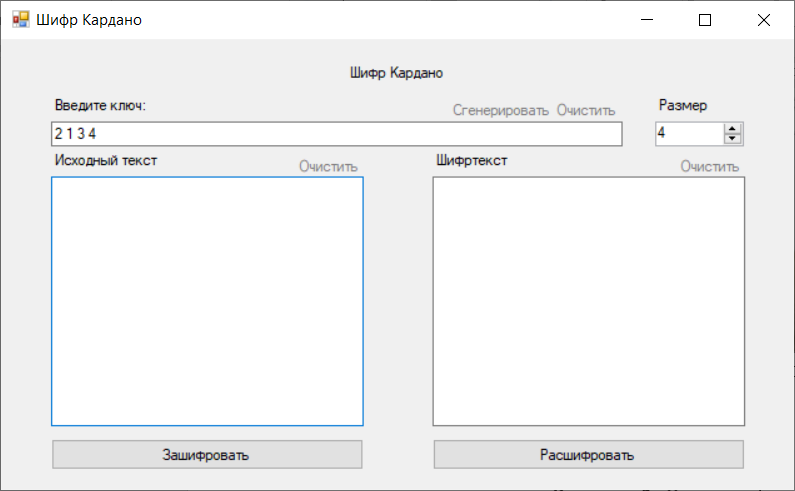


Рисунок 2. Окно шифра Кардано.

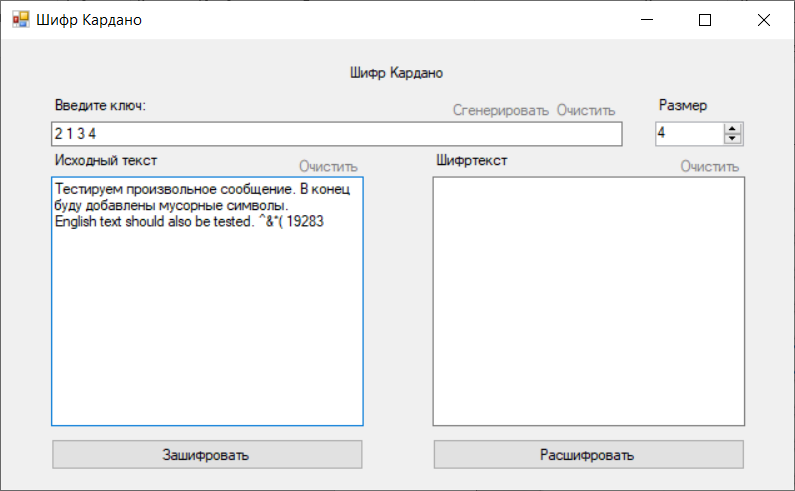


Рисунок 3. Исходный текст.

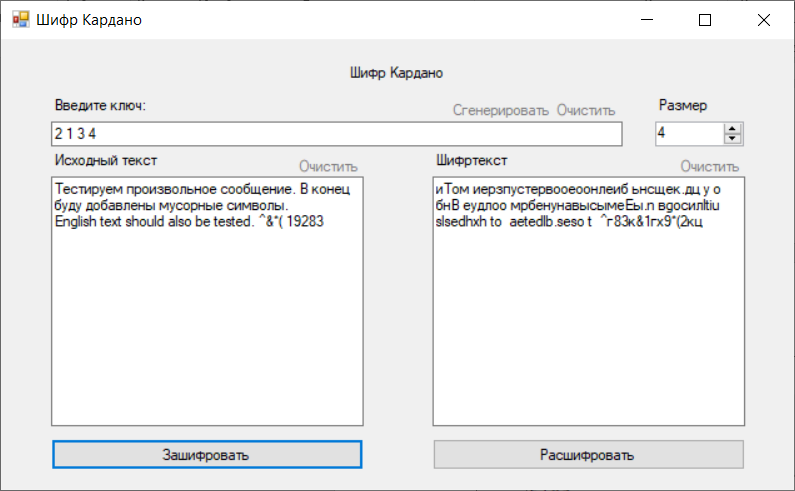


Рисунок 4. Шифртекст (результат зашифрования).

Очистим поле Исходный текст и расшифруем сообщение, оставив ключ неизменным.

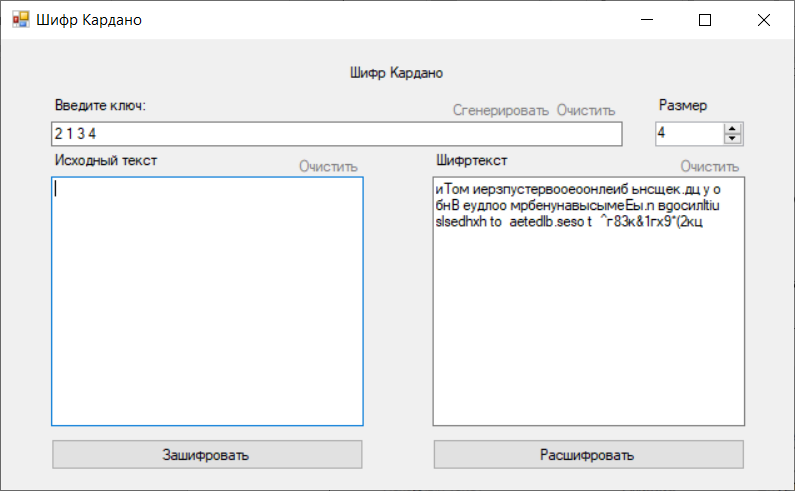


Рисунок 5. Очистка поля Исходный текст.

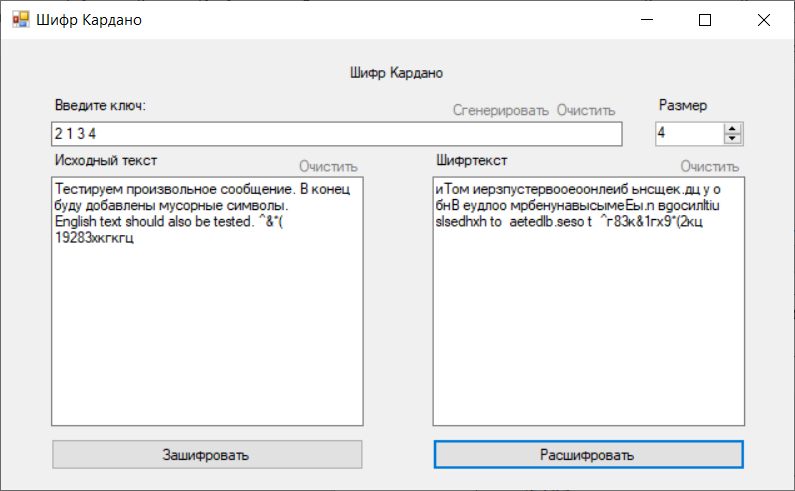


Рисунок 6. Исходный текст с мусорными символами в конце (результат расшифрования).

**Вывод:**

В ходе выполнения лабораторной работы я ознакомилась с алгоритмом шифра Кардано, 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 Кардано.