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 №10**

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

**Теория:**

Шифр Плейфера использует матрицу 5х5, 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 (в английских текстах обычно опускается символ «Q», чтобы уменьшить алфавит, в других версиях «I» и «J» объединяются в одну ячейку). Ключевое слово может быть записано в верхней строке матрицы слева направо, либо по спирали из левого верхнего угла к центру. Ключевое слово, дополненное алфавитом, составляет матрицу 5х5 и является ключом шифра.

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 следующими 4 правилами, зашифровываем пары символов исходного текста:

Если два символа биграммы совпадают (или если остался один символ), добавляем после первого символа «Х», зашифровываем новую пару символов и продолжаем. В некоторых вариантах шифра Плейфера вместо «Х» используется «Q».

Если символы биграммы исходного текста встречаются в одной строке, то эти символы замещаются на символы, расположенные в ближайших столбцах справа от соответствующих символов. Если символ является последним в строке, то он заменяется на первый символ этой же строки.

Если символы биграммы исходного текста встречаются в одном столбце, то они преобразуются в символы того же столбца, находящиеся непосредственно под ними. Если символ является нижним в столбце, то он заменяется на первый символ этого же столбца.

Если символы биграммы исходного текста находятся в разных столбцах и разных строках, то они заменяются на символы, находящиеся в тех же строках, но соответствующие другим углам прямоугольника.

Для расшифровки необходимо использовать инверсию этих четырёх правил, откидывая символы «Х» (или «Q»), если они не несут смысла в исходном сообщении.

**Исходный код:**

**Преобразование ключа:**

private int Keyword\_symb\_to\_Int\_Rotate(char symb)

{

string ALF\_RU = "АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯАБВГДЕЁЖЗ";

string alf\_ru = "абвгдеёжзийклмнопрстуфхцчшщъыьэюяабвгдеёжз";

string ALF\_EN = "ABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHI";

string alf\_en = "abcdefghijklmnopqrstuvwxyzabcdefghi";

int num = -1;

bool rus = false;

if (alf\_ru.IndexOf(symb) > num) { num = alf\_ru.IndexOf(symb); rus = true; }

if (ALF\_RU.IndexOf(symb) > num) { num = ALF\_RU.IndexOf(symb); rus = true; }

if (alf\_en.IndexOf(symb) > num) num = alf\_en.IndexOf(symb);

if (ALF\_EN.IndexOf(symb) > num) num = ALF\_EN.IndexOf(symb);

if (rus)

num = num % 33;

else if (num != -1)

num = num % 26;

return num;

}

**Алгоритм зашифрования:**

private void f10\_ButtonEncrypt\_Click(object sender, EventArgs e)

{

string Source = f10\_fieldOriginal.Text;

string entered\_key = Regex.Replace(f10\_fieldKeyword.Text, "[^a-zA-Zа-яА-ЯёЁ]", "");

f10\_fieldKeyword.Text = entered\_key;

entered\_key = entered\_key.Replace("J", "I").Replace("Ё", "Е").Replace("Й", "И").Replace("Ь", "Ъ").Replace("j", "i").Replace("ё", "е").Replace("й", "и").Replace("ь", "ъ").ToLower();

if (what\_language(Source) == -1 || what\_language(entered\_key) == -1 || (what\_language(Source)\*what\_language(entered\_key)) == 2) //Разные языки

{

MessageBox.Show("Поля должны быть заполнены только на английском или только на русском языке.", "Ошибка");

goto metka\_exit;

}

Source = Source.Replace("J", "I").Replace("Ё", "Е").Replace("Й", "И").Replace("Ь", "Ъ").Replace("j", "i").Replace("ё", "е").Replace("й", "и").Replace("ь", "ъ");

//Выбор алфавита

string ALF, alf;

bool lenIsRus;

int rows, columns;

if (what\_language(Source) == 1) //Русский текст и русский/пустой ключ

{

ALF = ALF\_RU;

alf = alf\_ru;

lenIsRus = true;

rows = 5;

columns = 6;

}

else

{

ALF = ALF\_EN;

alf = alf\_en;

lenIsRus = false;

rows = 5;

columns = 5;

}

//Сохраним регистр и будем работать только с нижним

int[] isUpper = new int[Source.Length];

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

{

if (ALF.IndexOf(Source[i]) >= 0)

isUpper[i] = 1;

else

isUpper[i] = 0;

}

Source = Source.ToLower();

char[] message = Source.ToCharArray();

//Формирование таблицы, на основе которой будет происходить шифрование (будет располагаться в одномерном массиве) 5 строк, 6 столбцов

int gridSize = alf.Length;

char[] PlayfairGrid = new char[gridSize];

int keyPos = 0;

int gridPos = 0;

int[] times = new int[alf.Length]; //иницилизация массива вхождений

for (int i = 0; i < times.Length; i++) times[i] = 0; //обнуление массива вхождений

while (keyPos < entered\_key.Length)

{

if (alf.IndexOf(entered\_key[keyPos]) >= 0 && times[alf.IndexOf(entered\_key[keyPos])] == 0)

{

times[alf.IndexOf(entered\_key[keyPos])]++;

PlayfairGrid[gridPos] = entered\_key[keyPos];

gridPos++;

}

keyPos++;

}

int alfPos = 0;

while (gridPos < gridSize)

{

if (times[alfPos] == 0)

{

times[alfPos]++;

PlayfairGrid[gridPos] = alf[alfPos];

gridPos++;

}

alfPos++;

}

//Добавим мусорные буквы в последовательные пары одинаковых буквы

int similar = 0;

int previousLetter = 0;

while (!char.IsLetter(message[previousLetter])) previousLetter++;

//Считаем количество пар одинаковых букв, идущих подряд.

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

{

if (message[previousLetter] == message[i]) similar++;

if (char.IsLetter(message[i])) previousLetter = i;

}

char[] tmp = new char[message.Length + similar];

int[] register = new int[message.Length + similar + 1];

int rot = 0;

char letter = '⁣';

//Добавление букв

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

{

if (char.IsLetter(message[i - rot]) && message[i - rot] == letter)

{

if (lenIsRus)

{

tmp[i] = 'х';

if (letter == 'х') tmp[i] = 'ъ';

}

else

{

tmp[i] = 'x';

if (letter == 'x') tmp[i] = 'q';

}

register[i] = 0;

i++;

rot++;

tmp[i] = message[i - rot];

register[i] = isUpper[i - rot];

}

else

{

tmp[i] = message[i - rot];

register[i] = isUpper[i - rot];

}

if (char.IsLetter(message[i - rot])) letter = message[i - rot];

}

//Добавим букву в конец, если букв нечетное количество

int amount\_letters = 0;

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

if (char.IsLetter(tmp[i])) amount\_letters++;

if (amount\_letters % 2 == 1)

{

char[] tmp2 = tmp;

tmp = new char[tmp.Length + 1];

for (int i = 0; i < tmp2.Length; i++)

tmp[i] = tmp2[i];

if (lenIsRus)

{

tmp[tmp.Length - 1] = 'х';

if (tmp[tmp.Length - 2] == 'х') tmp[tmp.Length - 1] = 'ъ';

}

else

{

tmp[tmp.Length - 1] = 'x';

if (tmp[tmp.Length - 2] == 'x') tmp[tmp.Length - 1] = 'q';

}

register[tmp.Length - 1] = 0;

}

//Парное шифрование

string playfair = new string(PlayfairGrid);

int pair\_found = 0;//сколько элементов найдено для пары

int first\_letter\_ind = 0, second\_letter\_ind = 0; //индексы элементов, составляющих пару

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

{

if (playfair.IndexOf(tmp[i]) >= 0)

{

if (pair\_found == 0)

{

first\_letter\_ind = i;

pair\_found++;

}

else

{

second\_letter\_ind = i;

char[] pair = new char[2]; //Адреса символов в исходном тексте

pair[0] = tmp[first\_letter\_ind];

pair[1] = tmp[second\_letter\_ind];

int[] x\_coord = new int[2];

x\_coord[0] = playfair.IndexOf(pair[0]) % columns; //столбец

x\_coord[1] = playfair.IndexOf(pair[1]) % columns; //столбец

int[] y\_coord = new int[2];

y\_coord[0] = playfair.IndexOf(pair[0]) / columns; //строка

y\_coord[1] = playfair.IndexOf(pair[1]) / columns; //строка

//Зашифрование пары символов

if ((y\_coord[0] == y\_coord[1]) && (x\_coord[0] != x\_coord[1])) //символы находятся в одной строке

{

tmp[first\_letter\_ind] = playfair[columns \* y\_coord[0] + ((x\_coord[0] + 1) % columns)];

tmp[second\_letter\_ind] = playfair[columns \* y\_coord[1] + ((x\_coord[1] + 1) % columns)];

}

if ((y\_coord[0] != y\_coord[1]) && (x\_coord[0] == x\_coord[1])) //символы находятся в одном столбце

{

tmp[first\_letter\_ind] = playfair[columns \* ((y\_coord[0] + 1) % rows) + x\_coord[0]];

tmp[second\_letter\_ind] = playfair[columns \* ((y\_coord[1] + 1) % rows) + x\_coord[1]];

}

if ((y\_coord[0] != y\_coord[1]) && (x\_coord[0] != x\_coord[1])) //символы образуют прямоугольник

{

tmp[first\_letter\_ind] = playfair[columns \* y\_coord[0] + x\_coord[1]];

tmp[second\_letter\_ind] = playfair[columns \* y\_coord[1] + x\_coord[0]];

}

//Конец зашифрования пары символов

pair\_found = 0;

}

}

}

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

if (register[i] == 1) tmp[i] = ALF[alf.IndexOf(tmp[i])];

f10\_fieldCiphertext.Text = new string(tmp);

metka\_exit:;

}

**Алгоритм расшифрования:**

private void f10\_ButtonDecrypt\_Click(object sender, EventArgs e)

{

string Source = f10\_fieldCiphertext.Text;

string entered\_key = Regex.Replace(f10\_fieldKeyword.Text, "[^a-zA-Zа-яА-ЯёЁ]", "");

f10\_fieldKeyword.Text = entered\_key;

entered\_key = entered\_key.Replace("J", "I").Replace("Ё", "Е").Replace("Й", "И").Replace("Ь", "Ъ").Replace("j", "i").Replace("ё", "е").Replace("й", "и").Replace("ь", "ъ").ToLower();

if (what\_language(Source) == -1 || what\_language(entered\_key) == -1 || (what\_language(Source) \* what\_language(entered\_key)) == 2) //Разные языки

{

MessageBox.Show("Поля должны быть заполнены только на английском или только на русском языке.", "Ошибка");

goto metka\_exit;

}

Source = Source.Replace("J", "I").Replace("Ё", "Е").Replace("Й", "И").Replace("Ь", "Ъ").Replace("j", "i").Replace("ё", "е").Replace("й", "и").Replace("ь", "ъ");

//Выбор алфавита

string ALF, alf;

int rows, columns;

if (what\_language(Source) == 1) //Русский текст и русский/пустой ключ

{

ALF = ALF\_RU;

alf = alf\_ru;

rows = 5;

columns = 6;

}

else

{

ALF = ALF\_EN;

alf = alf\_en;

rows = 5;

columns = 5;

}

//Сохраним регистр и будем работать только с нижним

int[] isUpper = new int[Source.Length];

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

{

if (ALF.IndexOf(Source[i]) >= 0)

isUpper[i] = 1;

else

isUpper[i] = 0;

}

Source = Source.ToLower();

char[] message = Source.ToCharArray();

//Формирование таблицы, на основе которой будет происходить шифрование (будет располагаться в одномерном массиве) 5 строк, 6 столбцов

int gridSize = alf.Length;

char[] PlayfairGrid = new char[gridSize];

int keyPos = 0;

int gridPos = 0;

int[] times = new int[alf.Length]; //иницилизация массива вхождений

for (int i = 0; i < times.Length; i++) times[i] = 0; //обнуление массива вхождений

while (keyPos < entered\_key.Length)

{

if (alf.IndexOf(entered\_key[keyPos]) >= 0 && times[alf.IndexOf(entered\_key[keyPos])] == 0)

{

times[alf.IndexOf(entered\_key[keyPos])]++;

PlayfairGrid[gridPos] = entered\_key[keyPos];

gridPos++;

}

keyPos++;

}

int alfPos = 0;

while (gridPos < gridSize)

{

if (times[alfPos] == 0)

{

times[alfPos]++;

PlayfairGrid[gridPos] = alf[alfPos];

gridPos++;

}

alfPos++;

}

//Парное шифрование

string playfair = new string(PlayfairGrid);

int pair\_found = 0;//сколько элементов найдено для пары

int first\_letter\_ind = 0, second\_letter\_ind = 0; //индексы элементов, составляющих пару

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

{

if (playfair.IndexOf(message[i]) >= 0)

{

if (pair\_found == 0)

{

first\_letter\_ind = i;

pair\_found++;

}

else

{

second\_letter\_ind = i;

char[] pair = new char[2]; //Адреса символов в исходном тексте

pair[0] = message[first\_letter\_ind];

pair[1] = message[second\_letter\_ind];

int[] x\_coord = new int[2];

x\_coord[0] = playfair.IndexOf(pair[0]) % columns; //столбец

x\_coord[1] = playfair.IndexOf(pair[1]) % columns; //столбец

int[] y\_coord = new int[2];

y\_coord[0] = playfair.IndexOf(pair[0]) / columns; //строка

y\_coord[1] = playfair.IndexOf(pair[1]) / columns; //строка

//Зашифрование пары символов

if ((y\_coord[0] == y\_coord[1]) && (x\_coord[0] != x\_coord[1])) //символы находятся в одной строке

{

message[first\_letter\_ind] = playfair[columns \* y\_coord[0] + ((x\_coord[0] - 1 + columns) % columns)];

message[second\_letter\_ind] = playfair[columns \* y\_coord[1] + ((x\_coord[1] - 1 + columns) % columns)];

}

if ((y\_coord[0] != y\_coord[1]) && (x\_coord[0] == x\_coord[1])) //символы находятся в одном столбце

{

message[first\_letter\_ind] = playfair[columns \* ((y\_coord[0] - 1 + rows) % rows) + x\_coord[0]];

message[second\_letter\_ind] = playfair[columns \* ((y\_coord[1] - 1 + rows) % rows) + x\_coord[1]];

}

if ((y\_coord[0] != y\_coord[1]) && (x\_coord[0] != x\_coord[1])) //символы образуют прямоугольник

{

message[first\_letter\_ind] = playfair[columns \* y\_coord[0] + x\_coord[1]];

message[second\_letter\_ind] = playfair[columns \* y\_coord[1] + x\_coord[0]];

}

//Конец зашифрования пары символов

pair\_found = 0;

}

}

}

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

if (isUpper[i] == 1) message[i] = ALF[alf.IndexOf(message[i])];

f10\_fieldOriginal.Text = new string(message);

metka\_exit:;

}

**Результат работы программы:**

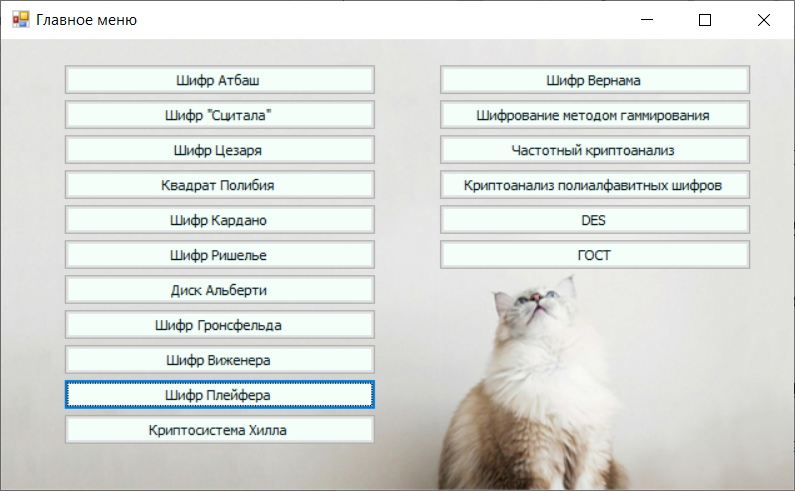


Рисунок 1. Главное меню программы.

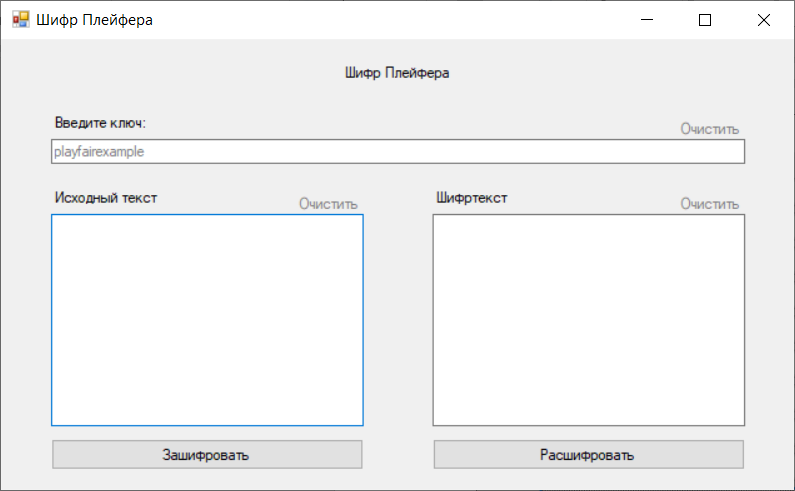
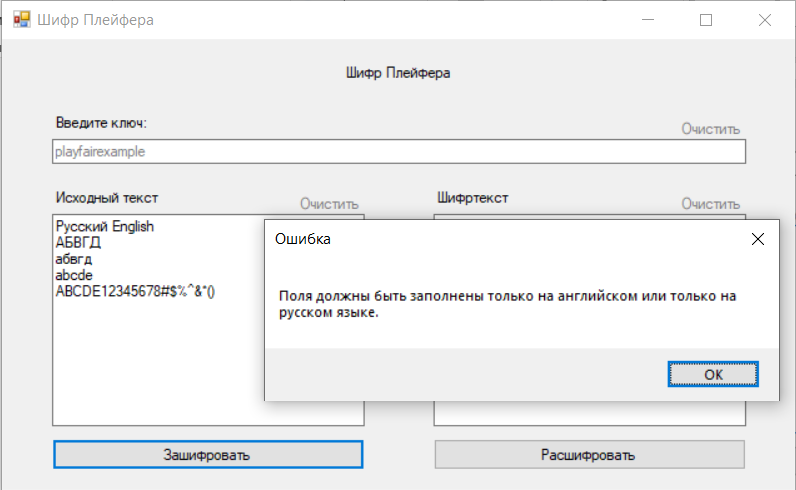


Рисунок 2. Окно шифра Плейфера.

Рисунок 3. Формат входных данных.

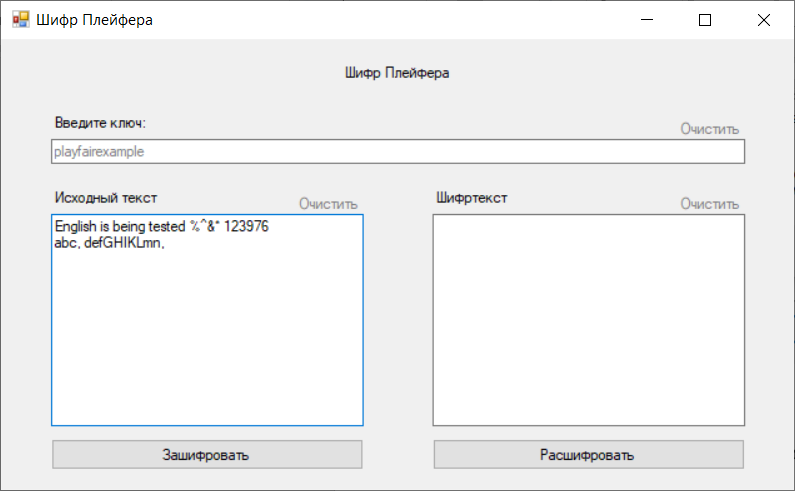


Рисунок 4. Исходный текст.

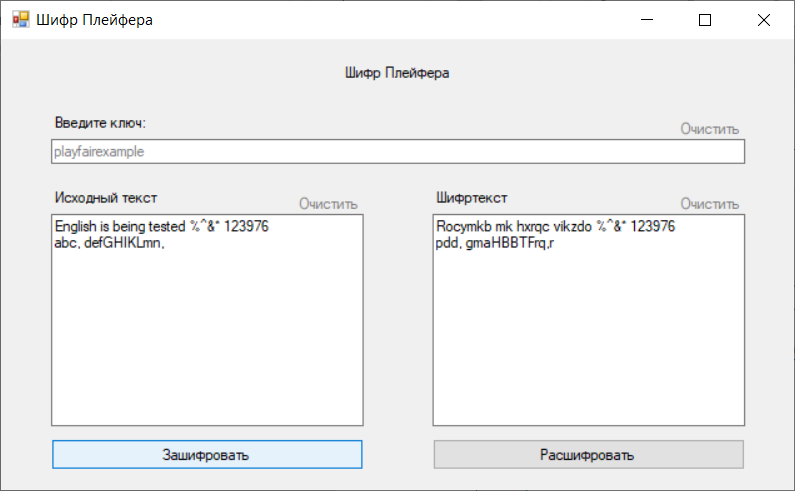


Рисунок 5. Шифртекст (результат зашифрования).

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

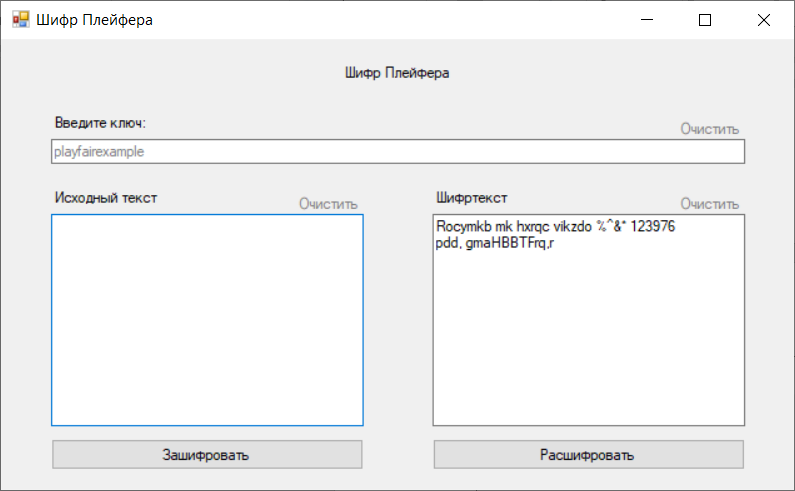


Рисунок 6. Очистка поля Исходный текст.

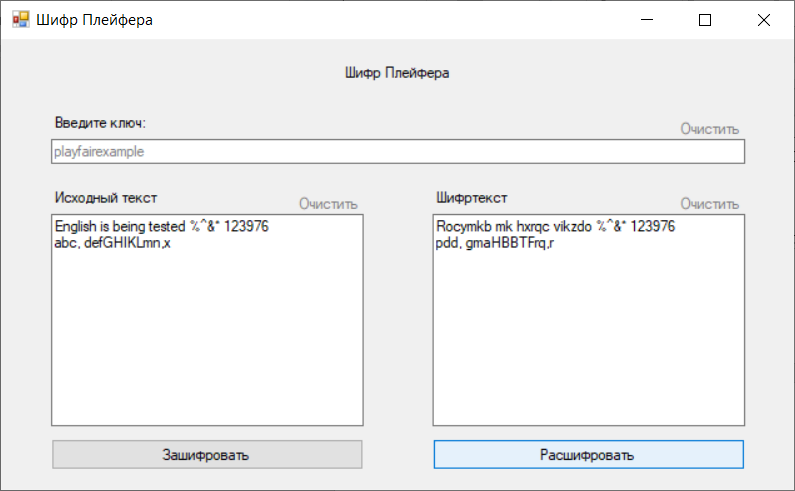


Рисунок 7. Исходный текст (результат расшифрования).

**Вывод:**

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