Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт Космических и информационных технологий

институт

Кафедра «Информатика»

кафедра

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1**

Преподаватель \_\_\_\_\_\_\_\_\_\_\_ А.Н. Шниперов

подпись, дата инициалы, фамилия

Студент КИ15–17Б, 031510065 \_\_\_\_\_\_\_\_\_\_\_ Т.В. Радионов

номер группы, зачетной книжки подпись, дата инициалы, фамилия

Красноярск 2017

1. **Цель работы**

* ознакомиться с основами симметричного шифрования;
* ознакомиться с простыми симметричными криптографическими шифрами на основе методов подстановок, перестановок и гаммирования;
* освоить основные этапы проектирования и реализации симметричных шифров;

1. **Задание к работе**

Разработать и составить алгоритмы шифрования и дешифрования текста по методу Плейфера и реализовать программу на их основе. Модифицировать реализованный алгоритм с целью повышения его криптостойкости.

1. **Описание алгоритма**

Данный алгоритм шифрования является симметричным. Для его реализации используется алфавитная матрица.

На первом этапе происходит перемешивание в матрице: ключ помещается в начало данной матрицы (без повторяющихся символов), а затем алфавит (без символов, которые уже есть в ключе), а далее остальные символы. В данном алгоритме реализовано свойство рассеивания, так как ключ напрямую влияет на шифротекст.

На втором этапе открытый текст делится на биграммы, а если у последнего символа нет пары, то к нему добавляется символ, не использующийся в открытом тексте, но присутствующий в матрице (для дальнейшего разъяснения назовем этот символ «$»).

На третьем этапе каждая биграмма шифруется по правилу алгоритма Плейфера:

1. Если два символа биграммы совпадают (или если остался один символ), добавляем после первого символа «$», зашифровываем новую пару символов и продолжаем.
2. 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.
3. 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.
4. 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.

Расшифрование происходит инверсией описанных выше пунктов с исключением символа «$», который не несет никакой информации в открытом тексте.

Шифр алгоритма Плейфера является блочным. Уязвимостью для его криптостойкости является то, что символы биграммы лишь заменяются другими, то есть это шифр перестановки, который с помощью одного из приемов криптоанализа, а именно сопоставлением вероятности появления отдельных символов и сопоставления с аналогичными значениями алфавита используемого при шифровании языка, можно легко осуществить дешифрование.

1. **Блок-схема алгоритма**
2. Вспомогательные алгоритмы

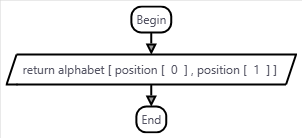
char FindCharacter(int [] position) // ф-ция поиска символа по его позиции в матрице

Output

FindCharacter(position)

// position – 2-х элементный массив, где 0 – i-ая строка, 1 – j-ый столбец

// alphabet – массив символов матрицы



private int[] FindCharacterPosition(char ch) // ф-ция поиска позиции символа в матрице по значению

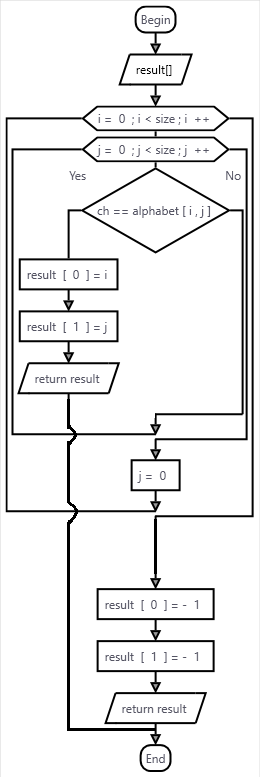
Output result

FindCharacterPosition(ch)

// ch - входной символ

// size - размерность матрицы

// alphabet – массив символов матрицы



1. Шифрование

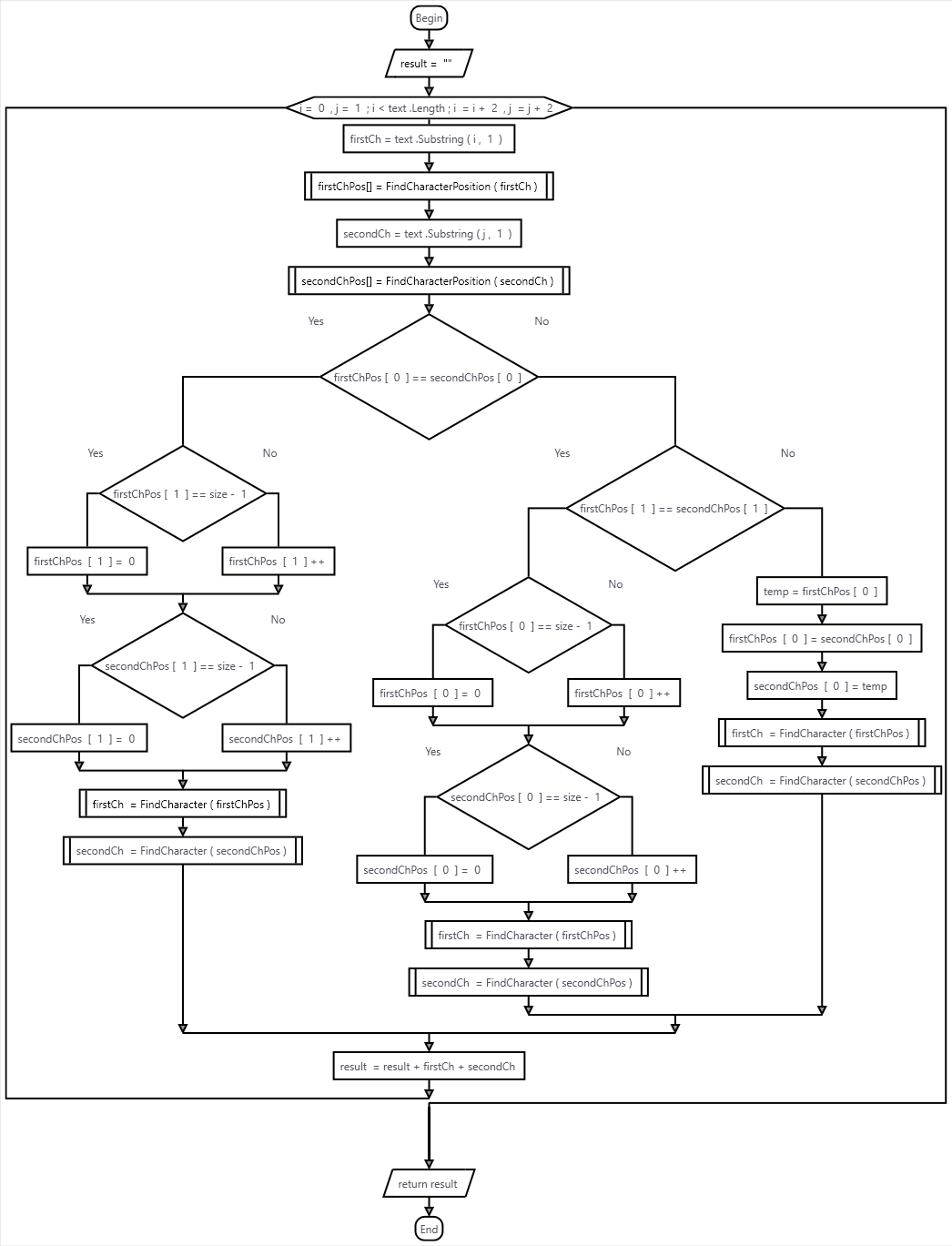
private string Encryption(string text)

Output result

Encryption(text)

// text - входная строка

// size - размерность матрицы



1. Расшифрование

private string Decryption(string text)

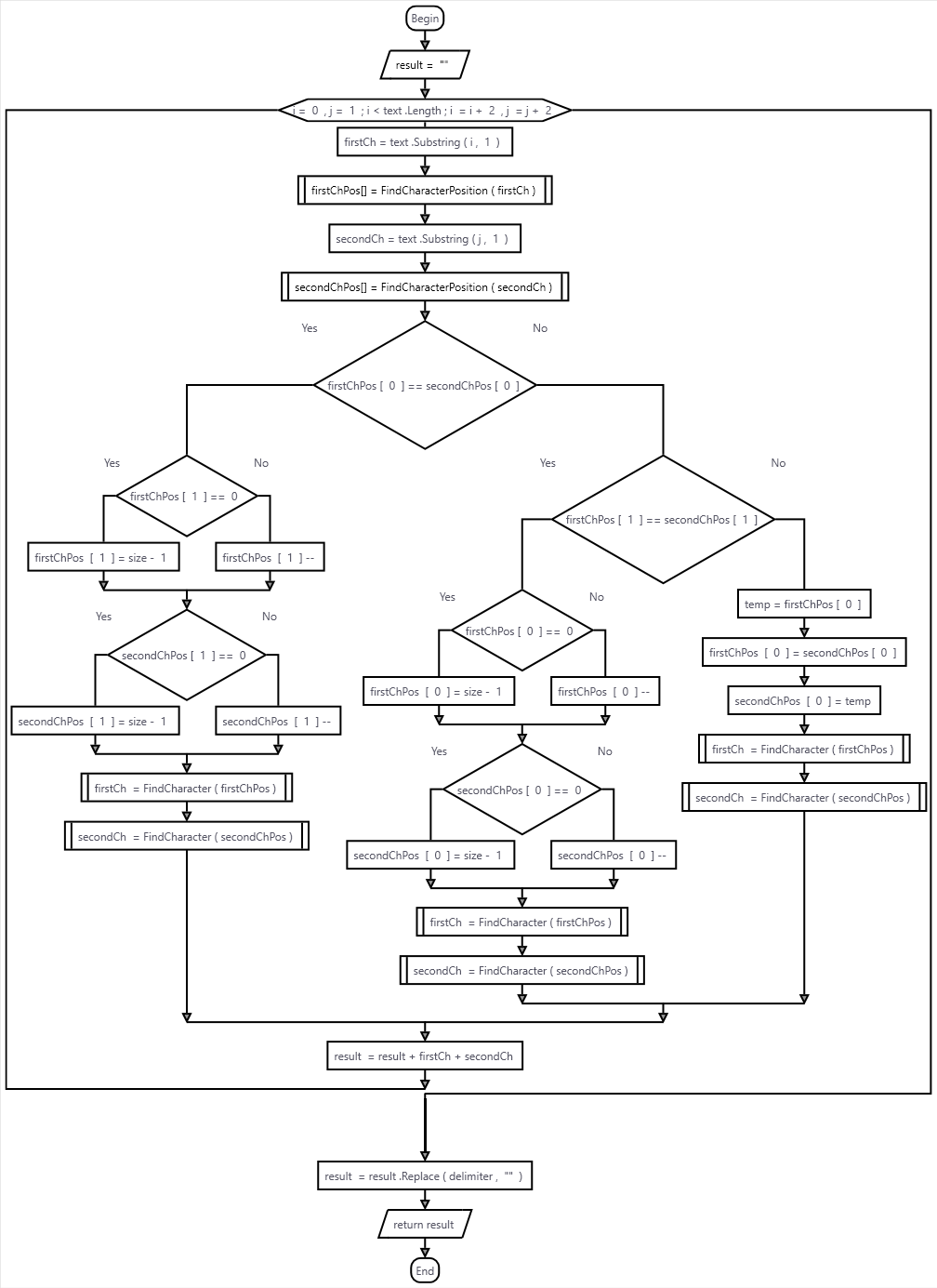
Output result

Здесь необходимо избавиться от спец. символа delimiter перед возвратом строки

Decryption(text)

// text - входная строка

// size - размерность матрицы



1. **Код алгоритма**
2. Шифрование

private string Encryption(string text)

{

string result = "";

for (int i = 0, j = 1; i < text.Length; i = i + 2, j = j + 2)

{

char firstCh = Convert.ToChar(text.Substring(i, 1));

int[] firstChPos = FindCharacterPosition(firstCh); // [0] - строка, [1] - столбец

char secondCh = Convert.ToChar(text.Substring(j, 1));

int[] secondChPos = FindCharacterPosition(secondCh); // [0] - строка, [1] - столбец

// Если символы находятся на одной строке в таблице (шаг 2)

if (firstChPos[0] == secondChPos[0])

{

// Если первый символ находится в крайнем столбце, то присваиваем значения символа в первом столбце

if (firstChPos[1] == size - 1)

{

firstChPos[1] = 0;

}

// Иначе смещаем на один столбец вправо

else

{

firstChPos[1]++;

}

// Если второй символ находится в крайнем столбце, то присваиваем значения символа в первом столбце

if (secondChPos[1] == size - 1)

{

secondChPos[1] = 0;

}

// Иначе смещаем на один столбец вправо

else

{

secondChPos[1]++;

}

firstCh = FindCharacter(firstChPos);

secondCh = FindCharacter(secondChPos);

}

// Если символы находятся на одном столбце в таблице (шаг 3)

else if (firstChPos[1] == secondChPos[1])

{

// Если первый символ находится на нижней строке, то присваиваем значения символа на первой строке

if (firstChPos[0] == size - 1)

{

firstChPos[0] = 0;

}

// Иначе смещаем на строку вниз

else

{

firstChPos[0]++;

}

// Если второй символ находится на нижней строке, то присваиваем значения символа на первой строке

if (secondChPos[0] == size - 1)

{

secondChPos[0] = 0;

}

// Иначе смещаем на строку вниз

else

{

secondChPos[0]++;

}

firstCh = FindCharacter(firstChPos);

secondCh = FindCharacter(secondChPos);

}

// Если символы находятся на разных столбцах и строках (шаг 4), то замещаем по "прямоугольнику" (см. алгоритм Плейфера)

else

{

int temp = firstChPos[0];

firstChPos[0] = secondChPos[0];

secondChPos[0] = temp;

firstCh = FindCharacter(firstChPos);

secondCh = FindCharacter(secondChPos);

}

result = result + firstCh.ToString() + secondCh.ToString();

}

return result;

}

1. Расшифрование

private string Decryption(string text)

{

string result = "";

for (int i = 0, j = 1; i < text.Length; i = i + 2, j = j + 2)

{

char firstCh = Convert.ToChar(text.Substring(i, 1));

int[] firstChPos = FindCharacterPosition(firstCh); // [0] - строка, [1] - столбец

char secondCh = Convert.ToChar(text.Substring(j, 1));

int[] secondChPos = FindCharacterPosition(secondCh); // [0] - строка, [1] - столбец

// Если символы находятся на одной строке в таблице (шаг 2)

if (firstChPos[0] == secondChPos[0])

{

// Если первый символ находится в первом столбце, то присваиваем значение символа в крайнем столбце

if (firstChPos[1] == 0)

{

firstChPos[1] = size - 1;

}

// Иначе смещаем на один столбец влево

else

{

firstChPos[1]--;

}

// Если второй символ находится в первом столбце, то присваиваем значение символа в крайнем столбце

if (secondChPos[1] == 0)

{

secondChPos[1] = size - 1;

}

// Иначе смещаем на один столбец влево

else

{

secondChPos[1]--;

}

firstCh = FindCharacter(firstChPos);

secondCh = FindCharacter(secondChPos);

}

// Если символы находятся на одном столбце в таблице (шаг 3)

else if (firstChPos[1] == secondChPos[1])

{

// Если первый символ находится на первой строке, то присваиваем значение символа на крайней строке

if (firstChPos[0] == 0)

{

firstChPos[0] = size - 1;

}

// Иначе смещаем на строку вверх

else

{

firstChPos[0]--;

}

// Если второй символ находится на первой строке, то присваиваем значение символа на крайней строке

if (secondChPos[0] == 0)

{

secondChPos[0] = size - 1;

}

// Иначе смещаем на строку вверх

else

{

secondChPos[0]--;

}

firstCh = FindCharacter(firstChPos);

secondCh = FindCharacter(secondChPos);

}

// Если символы находятся на разных столбцах и строках (шаг 4), то замещаем по "прямоугольнику" (см. алгоритм Плейфера)

else

{

int temp = firstChPos[0];

firstChPos[0] = secondChPos[0];

secondChPos[0] = temp;

firstCh = FindCharacter(firstChPos);

secondCh = FindCharacter(secondChPos);

}

result = result + firstCh.ToString() + secondCh.ToString();

}

result = result.Replace(delimiter, "");

return result;

}

1. **Примеры работы программы**

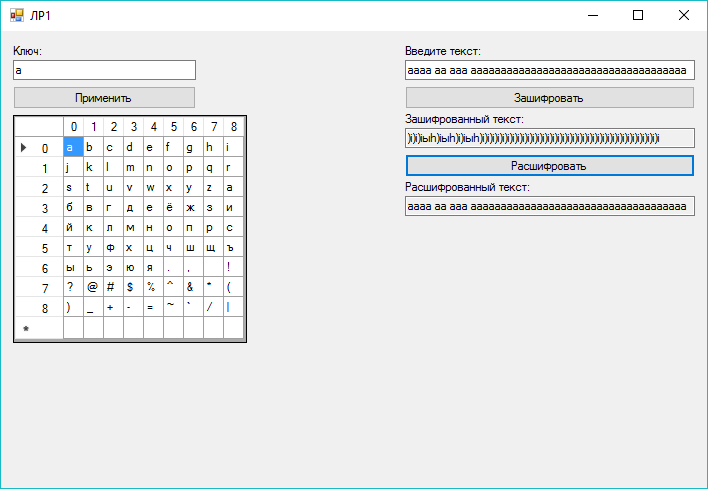


Рисунок 1 – Пример программы

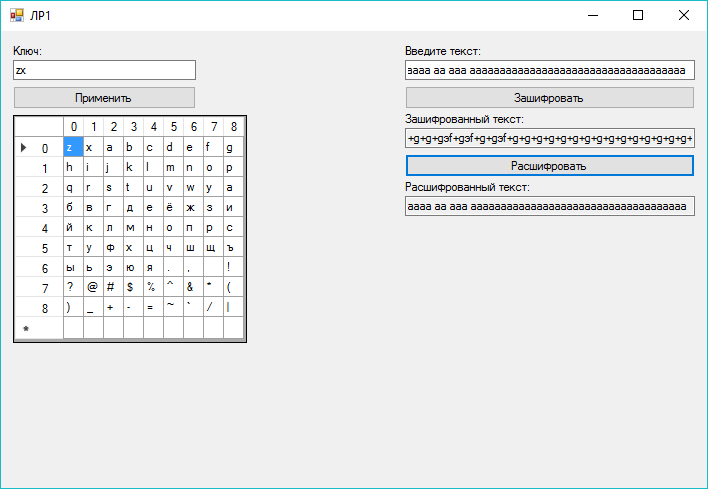


Рисунок 2 – Пример программы

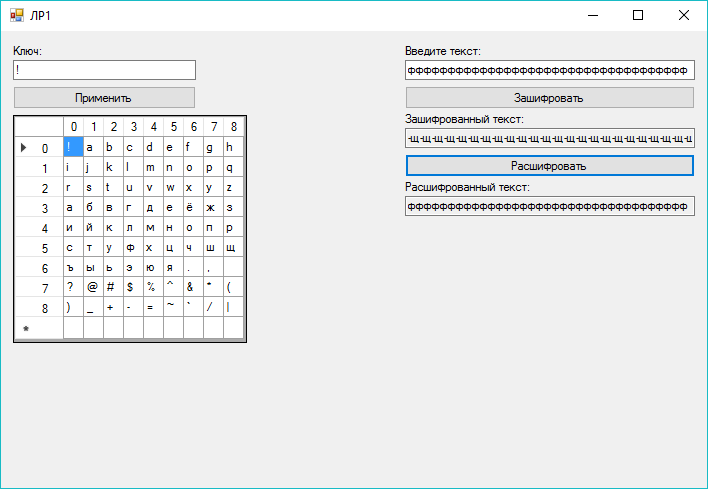


Рисунок 3 – Пример программы

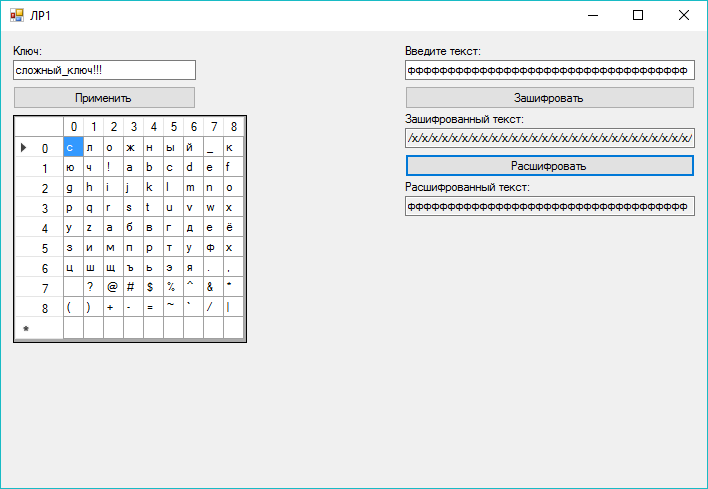


Рисунок 4 – Пример программы

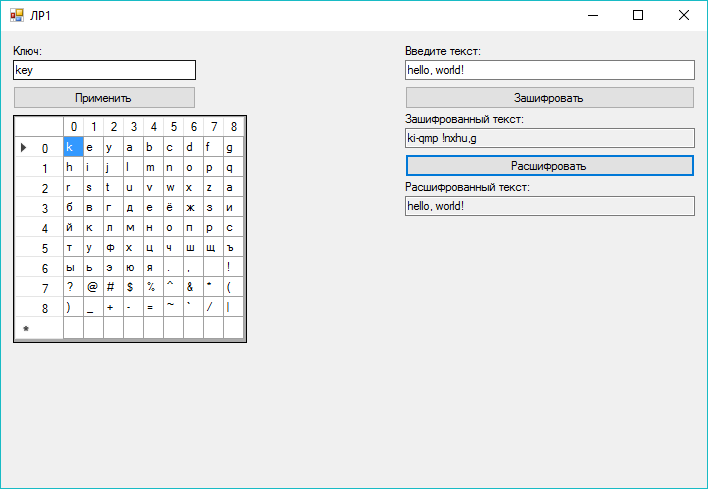


Рисунок 5 – Пример программы

1. **Описание модифицированного алгоритма**

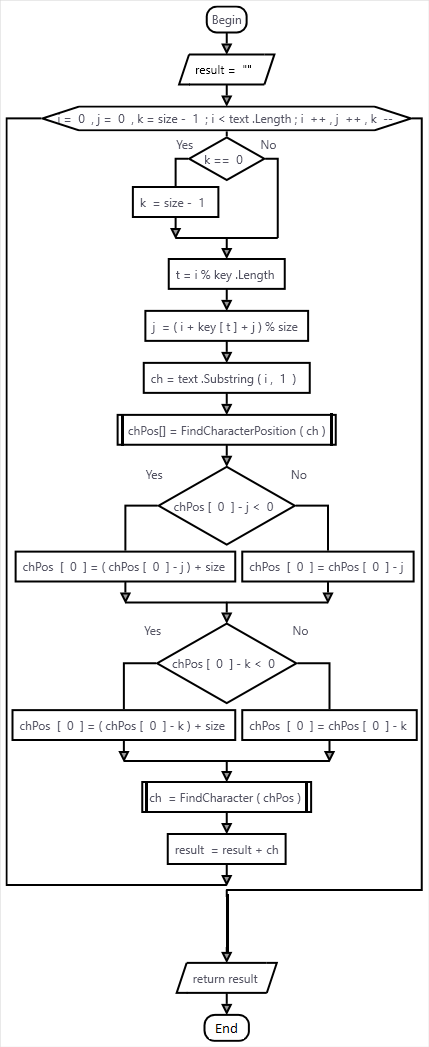
Решением по усилению криптостойкости является использование дополнительного алгоритма с полиалфавитным шифром и использованием ключа в процессе шифрования. Принцип работы алгоритма реализован следующим образом: внутри цикла при каждом проходе два счетчика i и j принимают различные значения по некоторому правилу, зависящему от ключа. В дальнейшем значения i и j используются для усовершенствованной, непредсказуемой для злоумышленника, подстановки в открытом тексте. Затем применяется алгоритм Плейфера, что лишь повышает криптостойкость системы.

Расшифрование основывается на схожем алгоритме, лишь с разницей от шифрующего в работе счетчиков i и j: они в обратном направлении принимают новые значения, которые соответствуют ранее прежним значениям в открытом тексте.

В качестве доказательства надежности криптосистемы можно привести один наглядный пример: в алгоритме Плейфера открытый текст с одинаковым набором символов «аааааа» заменяется перестановкой лишь на другой набор одинаковых символов той же длины. В модифицированном же алгоритме даже одинаковый набор символов шифруется по различному благодаря усовершенствованной перестановке.

1. **Блок-схема модифицированного алгоритма**
2. Шифрование

private string AdditionEncryption(string text)

****

Output result

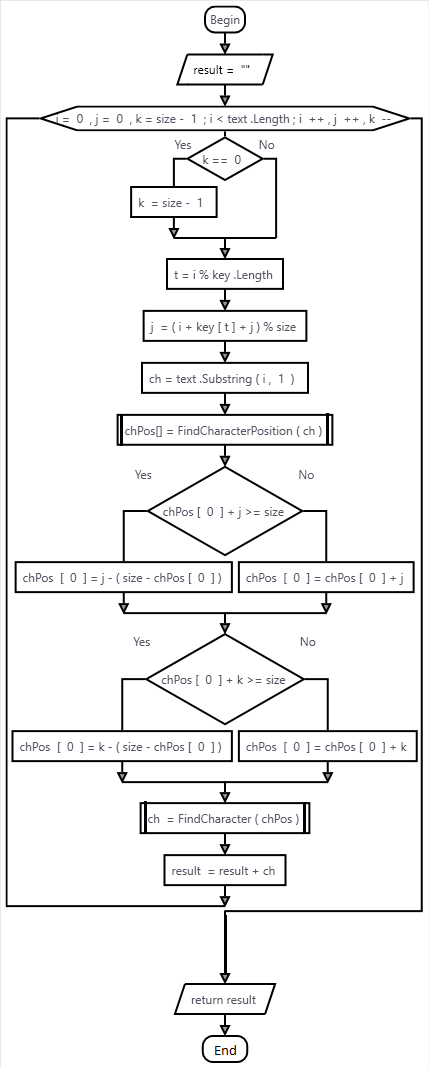
AdditionEncryption(text)

// text - входная строка

// size - размерность матрицы

1. Расшифрование

private string AdditionDecryption(string text)



Output result

AdditionDecryption(text)

// text - входная строка

// size - размерность матрицы

1. **Код модифицированного алгоритма**
2. Шифрование

private string AdditionEncryption(string text)

{

string result = "";

for (int i = 0, j = 0, k = size - 1; i < text.Length; i++, j++, k--)

{

if (k == 0)

k = size - 1;

int t = i % key.Length;

j = (i + key[t] + j) % size;

char ch = Convert.ToChar(text.Substring(i, 1));

int[] chPos = FindCharacterPosition(ch); // [0] - строка, [1] - столбец

// Если строка со смещением на j выходит за пределы размерности матрицы, то

if (chPos[0] + j >= size)

{

chPos[0] = j - (size - chPos[0]);

}

// Иначе просто смещаем

else

{

chPos[0] = chPos[0] + j;

}

// Если строка со смещением на k выходит за пределы размерности матрицы, то

if (chPos[0] + k >= size)

{

chPos[0] = k - (size - chPos[0]);

}

// Иначе просто смещаем

else

{

chPos[0] = chPos[0] + k;

}

ch = FindCharacter(chPos);

result = result + ch.ToString();

}

return result;

}

1. Расшифрование

private string AdditionDecryption(string text)

{

string result = "";

for (int i = 0, j = 0, k = size - 1; i < text.Length; i++, j++, k--)

{

if (k == 0)

k = size - 1;

int t = i % key.Length;

j = (i + key[t] + j) % size;

char ch = Convert.ToChar(text.Substring(i, 1));

int[] chPos = FindCharacterPosition(ch); // [0] - строка, [1] - столбец

// Если строка со смещением на j выходит за пределы размерности матрицы, то

if (chPos[0] - j < 0)

{

chPos[0] = (chPos[0] - j) + size;

}

// Иначе просто смещаем

else

{

chPos[0] = chPos[0] - j;

}

// Если строка со смещением на k выходит за пределы размерности матрицы, то

if (chPos[0] - k < 0)

{

chPos[0] = (chPos[0] - k) + size;

}

// Иначе просто смещаем

else

{

chPos[0] = chPos[0] - k;

}

ch = FindCharacter(chPos);

result = result + ch.ToString();

}

return result;

}

1. **Пример работы программы с модифицированным алгоритмом**

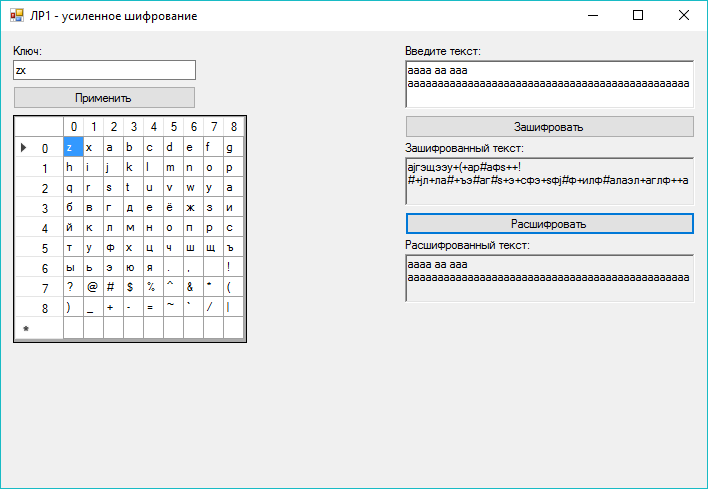


Рисунок 6 – Пример программы

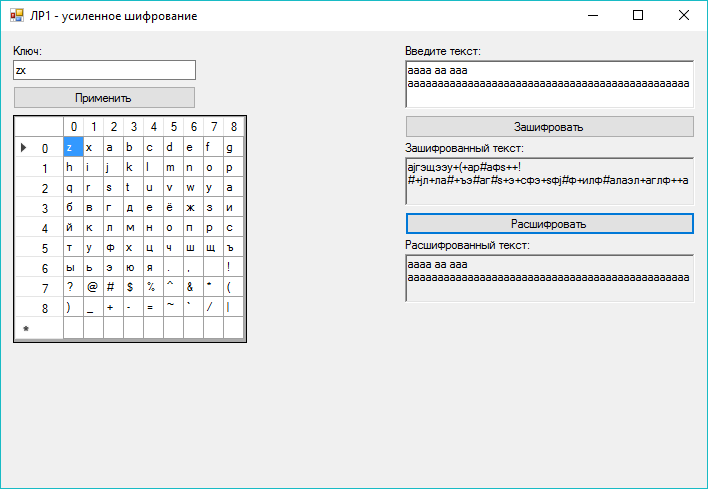


Рисунок 7 – Пример программы

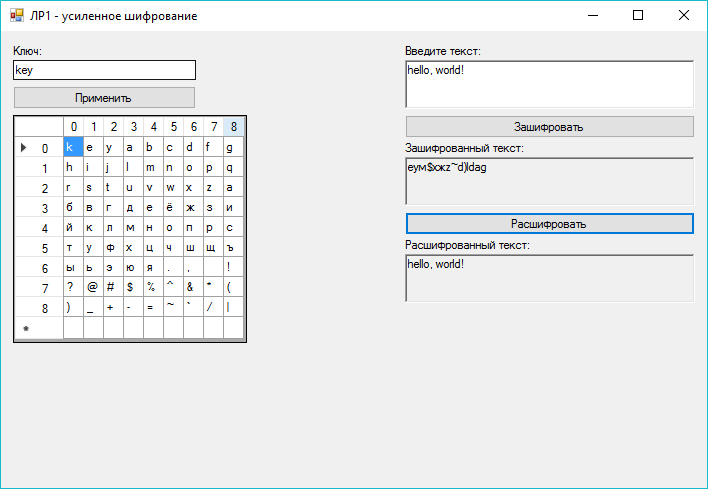


Рисунок 8 – Пример программы

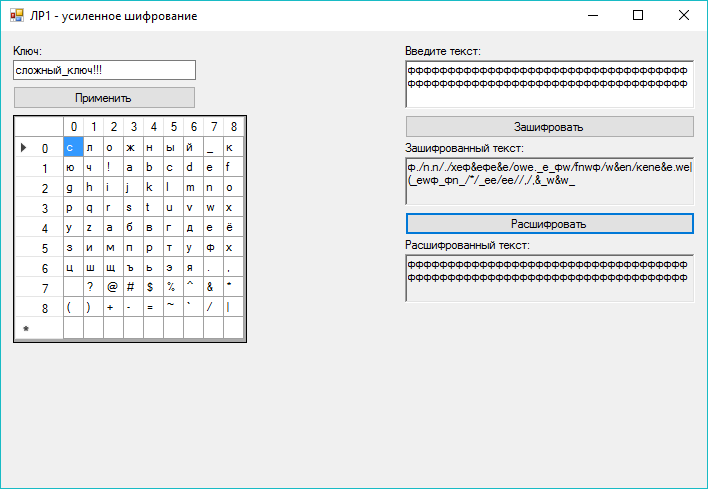


Рисунок 9 – Пример программы

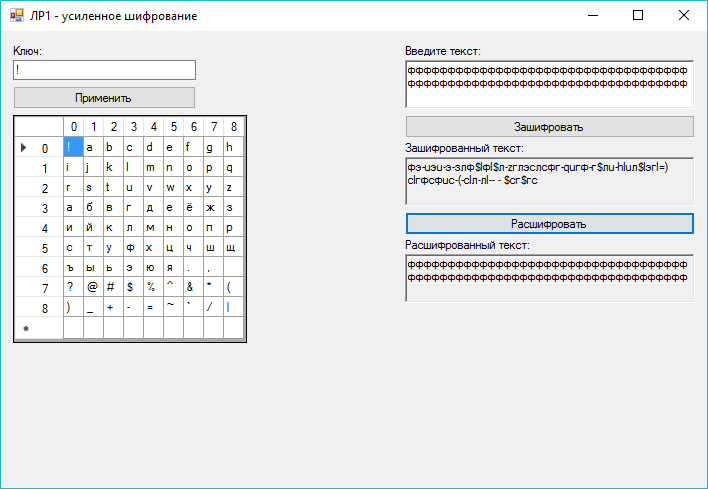


Рисунок 10 – Пример программы

1. **Выводы**

В результате проделанной работы были изучены преимущества и недостатки алгоритма Плейфера, был разработан усиленный алгоритм шифрования на его основе. Было установлено преимущество шифра с усовершенствованной подстановкой, а также замечена эффективность использования нескольких методов шифрования одновременно.