Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт Космических и информационных технологий
институт
Кафедра «Информатика»
кафедра

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №1

Простые симметричные шифры
тема

Вариант 1

Преподаватель _____ А.А. Сидарас
подпись, дата инициалы, фамилия

Студент КИ18-17/1б 031831229 _____ В.А. Прекель
номер группы, зачетной книжки подпись, дата инициалы, фамилия

Красноярск 2020

СОДЕРЖАНИЕ

1 Цель работы с постановкой задачи

1.1 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.

1.2 Задача работы

Согласно персональному варианту или индивидуальному заданию преподавателя разработать и составить в виде блок-схемы алгоритмы шифрования и дешифрования текста. Убедиться в правильности составления алгоритмов и затем на языке программирования составить программу, которая реализует данные алгоритмы.

На ряде контрольных примеров (не менее 10) открытого текста, состоящего из различного количества символов, проверить правильность работы алгоритмов шифрования и дешифрования.

Самостоятельно или с помощью преподавателя придумать оригинальный способ модификации шифра с целью повышения его криптостойкости. Внести изменения в исходный алгоритм и программу. Проверить работоспособность алгоритма на тестовых примерах.

Доказать, что предложенный Вами способ модификации действительно повышает криптостокость.

Разработанная Вами программа должна содержать графический интерфейс пользователя.

Вариант 1 – Шифр на основе «магических» квадратов размерностью $N \times N$.

2 Теоретические сведения

Для шифрования методом магических квадратов требуется квадратная матрица $N \times N$, заполненная N^2 различными натуральными числами от 1 до N таким образом, что сумма чисел в каждой строке, каждом столбце и на обеих диагоналях одинакова [1]. Это магический квадрат, и он будет ключом шифрования. При шифровании буквы открытого текста необходимо вписать в магический квадрат в соответствии с нумерацией его клеток. Для получения шифротекста считывают содержимое заполненной таблицы по строкам. [2]

Для взлома шифрования с помощью магических квадратов требуется определить размер магического квадрата, основываясь на длине шифротекста и перебрать магические квадраты этого размера. Поэтому для усложнения взлома можно добавить функцию перевода букв текста в верхний регистр, удаления пробелов или всех не-букв, а также возможность поворота ключа (от чего магичность ключа не меняется). Так же для шифрования достаточно использовать не только магические квадраты, а просто заполненные N^2 различными натуральными числами от 1 до N.

3 Ход работы

Алгоритм реализован на языке C# 8 используя фреймворк .NET Core 3.1 и библиотеку для графического интерфейса AvaloniaUI.

Рисунок 1 – Блок-схема алгоритма шифрования

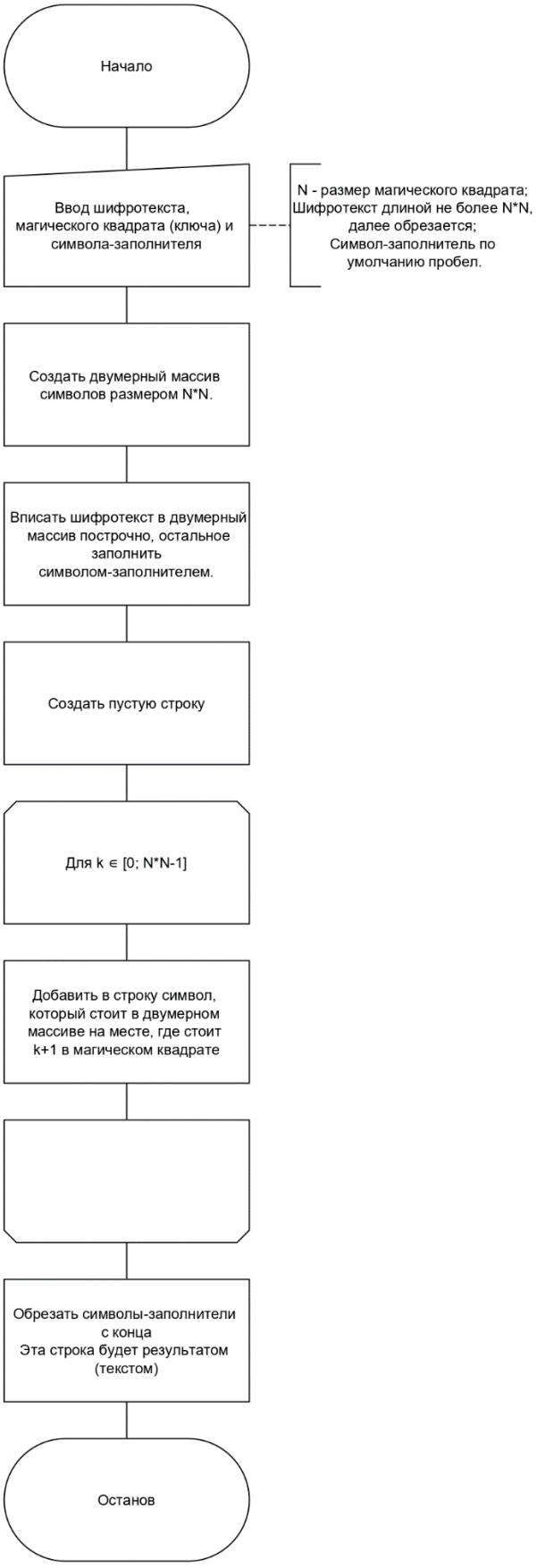

Рисунок 2 – Блок-схема алгоритма расшифровки

Листинг 1 – Lab_01/Lab_01.Core/MagicSquare.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;

namespace Lab_01.Core
{
    /// <summary>
    ///     Магический квадрат.
    /// </summary>
    public class MagicSquare : IEnumerable<int>
    {
        /// <summary>
        ///     Значения магического квадрата.
        /// </summary>
        private readonly int[,] _data;

        /// <summary>
        ///     Инициализирует новый экземпляр магического квадрата, заполняя нулями.
        /// </summary>
        /// <param name="n">Размер квадрата.</param>
        public MagicSquare(int n)
        {
            Count = n;
            _data = new int[n, n];
        }

        /// <summary>
        ///     Инициализирует новый экземпляр магического квадрата, заполняя значения из строки.
        ///     Перевод строки между строками, пробел между рядами.
        /// </summary>
        /// <param name="s">Магический квадрат в виде строки.</param>
        public MagicSquare(string s)
        {
            var rows = s.Trim('\n').Split('\n');
            Count = rows.Length;
            _data = new int[Count, Count];
            for (var i = 0; i < Count; i++)
            {
                var col = rows[i].Split();
                for (var j = 0; j < Count; j++)
                {
                    this[i, j] = Int32.Parse(col[j]);
                }
            }
        }

        /// <summary>
        ///     Размер квадрата (N).
        /// </summary>
        public int Count { get; }

        public int this[int i, int j]
        {
            get => _data[i, j];
            set => _data[i, j] = value;
        }

        /// <summary>
        ///     Является ли квадрат магическим.
        /// </summary>
        public bool IsMagic => IsMagicSum && IsDistinct;
```

```
/// <summary>
///     Правда ли, что сумма чисел в каждой строке, каждом столбце и на обеих диагоналях одинакова.
/// </summary>
public bool IsMagicSum =>
    Enumerable.Range(0, Count - 1)
        .Select(SumRow)
        .Concat(Enumerable.Range(0, Count - 1)
            .Select(SumColumn))
        .Append(SumMainDiagonal())
        .Append(SumAntiDiagonal())
        .Distinct()
        .Count()
        .Equals(1);

/// <summary>
///     Являются ли элементы в квадрате от 1 до N*N.
/// </summary>
public bool IsDistinct =>
    this.OrderBy(p => p)
        .SequenceEqual(Enumerable.Range(1, Count * Count));

public IEnumerator<int> GetEnumerator() => _data.Cast<int>().GetEnumerator();

IEnumerator IEnumerable.GetEnumerator() => GetEnumerator();

/// <summary>
///     Сумма строки с индексом i.
/// </summary>
/// <param name="i">Индекс строки.</param>
/// <returns>Сумма строки.</returns>
public int SumRow(int i)
{
    var s = 0;
    for (var j = 0; j < Count; j++)
    {
        s += this[i, j];
    }

    return s;
}

/// <summary>
///     Сумма ряда с индексом j.
/// </summary>
/// <param name="j">Индекс ряда.</param>
/// <returns>Сумма ряда.</returns>
public int SumColumn(int j)
{
    var s = 0;
    for (var i = 0; i < Count; i++)
    {
        s += this[i, j];
    }

    return s;
}

/// <summary>
///     Сумма главной диагонали.
/// </summary>
/// <returns>Сумма главной диагонали.</returns>
public int SumMainDiagonal()
{
    var s = 0;
    for (var i = 0; i < Count; i++)
    {
        s += this[i, i];
    }
```

```
                return s;
            }

            /// <summary>
            ///     Сумма побочной диагонали.
            /// </summary>
            /// <returns>Сумма побочной диагонали.</returns>
            public int SumAntiDiagonal()
            {
                var s = 0;
                for (var i = 0; i < Count; i++)
                {
                    s += this[i, Count - i - 1];
                }

                return s;
            }

            /// <summary>
            ///     Преобразовывает в строку вставляя между строк перевод строки, а между рядом пробелы.
            /// </summary>
            /// <returns>Квадрат в виде строки.</returns>
            public override string ToString()
            {
                return String.Join("\n",
                    Enumerable.Range(0, Count)
                        .Select(p =>
                            {
                                var a = new int[Count];
                                for (var i = 0; i < Count; i++)
                                {
                                    a[i] = this[p, i];
                                }

                                return String.Join(" ", a.Select(u => u.ToString()));
                            }
                        )
                );
            }
        }
    }
```

Листинг 2 – Lab_01/Lab_01.Core/MagicSquareCipher.cs

```
using System.Text;

namespace Lab_01.Core
{
    /// <summary>
    ///     Шифрования методом магических квадратов.
    /// </summary>
    public class MagicSquareCipher
    {
        /// <summary>
        ///     Инициализирует новый экземпляр класса шифрования методом магических квадратов.
        /// </summary>
        /// <param name="key">Ключ (магический квадрат).</param>
        /// <param name="emptyChar">Символ, используемый на месте пустых ячеек.</param>
        public MagicSquareCipher(MagicSquare key, char emptyChar = ' ')
        {
            Key = key;
            CipherTextMatrix = new char[Key.Count, Key.Count];
            EmptyChar = emptyChar;
        }

        /// <summary>
```

```
///     Максимальная и рекомендуемая длина текста, возможная шифрованием данным ключом.
/// </summary>
public int MaxLength => Key.Count * Key.Count;

/// <summary>
///     Ключ (магический квадрат).
/// </summary>
public MagicSquare Key { get; }

/// <summary>
///     Матрица шифротекста (здесь шифротекст хранится).
/// </summary>
private char[,] CipherTextMatrix { get; }

/// <summary>
///     Текст.
/// </summary>
public string Text { get; private set; }

/// <summary>
///     Символ, используемый на месте пустых ячеек.
/// </summary>
public char EmptyChar { get; set; }

/// <summary>
///     Шифротекст (вычисляется из матрицы шифротекста).
/// </summary>
public string CipherText
{
    get
    {
        var sb = new StringBuilder();
        for (var i = 0; i < Key.Count; i++)
        {
            for (var j = 0; j < Key.Count; j++)
            {
                sb.Append(CipherTextMatrix[i, j]);
            }
        }

        return sb.ToString();
    }

    private set
    {
        for (var i = 0; i < Key.Count; i++)
        {
            for (var j = 0; j < Key.Count; j++)
            {
                var index = i * Key.Count + j;
                if (index >= value.Length)
                {
                    CipherTextMatrix[i, j] = EmptyChar;
                }
                else
                {
                    CipherTextMatrix[i, j] = value[index];
                }
            }
        }
    }
}

/// <summary>
///     Зашифровывает текст.
/// </summary>
/// <param name="text">Текст для шифрования.</param>
/// <returns>Шифротекст.</returns>
public string Crypt(string text)
```

```
        {
            Text = text;

            for (var i = 0; i < Key.Count; i++)
            {
                for (var j = 0; j < Key.Count; j++)
                {
                    if (Key[i, j] - 1 >= text.Length)
                    {
                        CipherTextMatrix[i, j] = EmptyChar;
                    }
                    else
                    {
                        CipherTextMatrix[i, j] = text[Key[i, j] - 1];
                    }
                }
            }

            return CipherText;
        }

        /// <summary>
        ///     Расшифровывает текст.
        /// </summary>
        /// <param name="cipherText">Шифротекст для расшифрованиая.</param>
        /// <returns>Текст.</returns>
        public string Encrypt(string cipherText)
        {
            CipherText = cipherText;

            var sb = new StringBuilder(MaxLength);
            for (var k = 0; k < MaxLength; k++)
            {
                for (var i = 0; i < Key.Count; i++)
                {
                    for (var j = 0; j < Key.Count; j++)
                    {
                        if (Key[i, j] == k + 1)
                        {
                            sb.Append(CipherTextMatrix[i, j]);
                        }
                    }
                }
            }

            Text = sb.ToString().TrimEnd(EmptyChar);
            return Text;
        }
    }
}
```

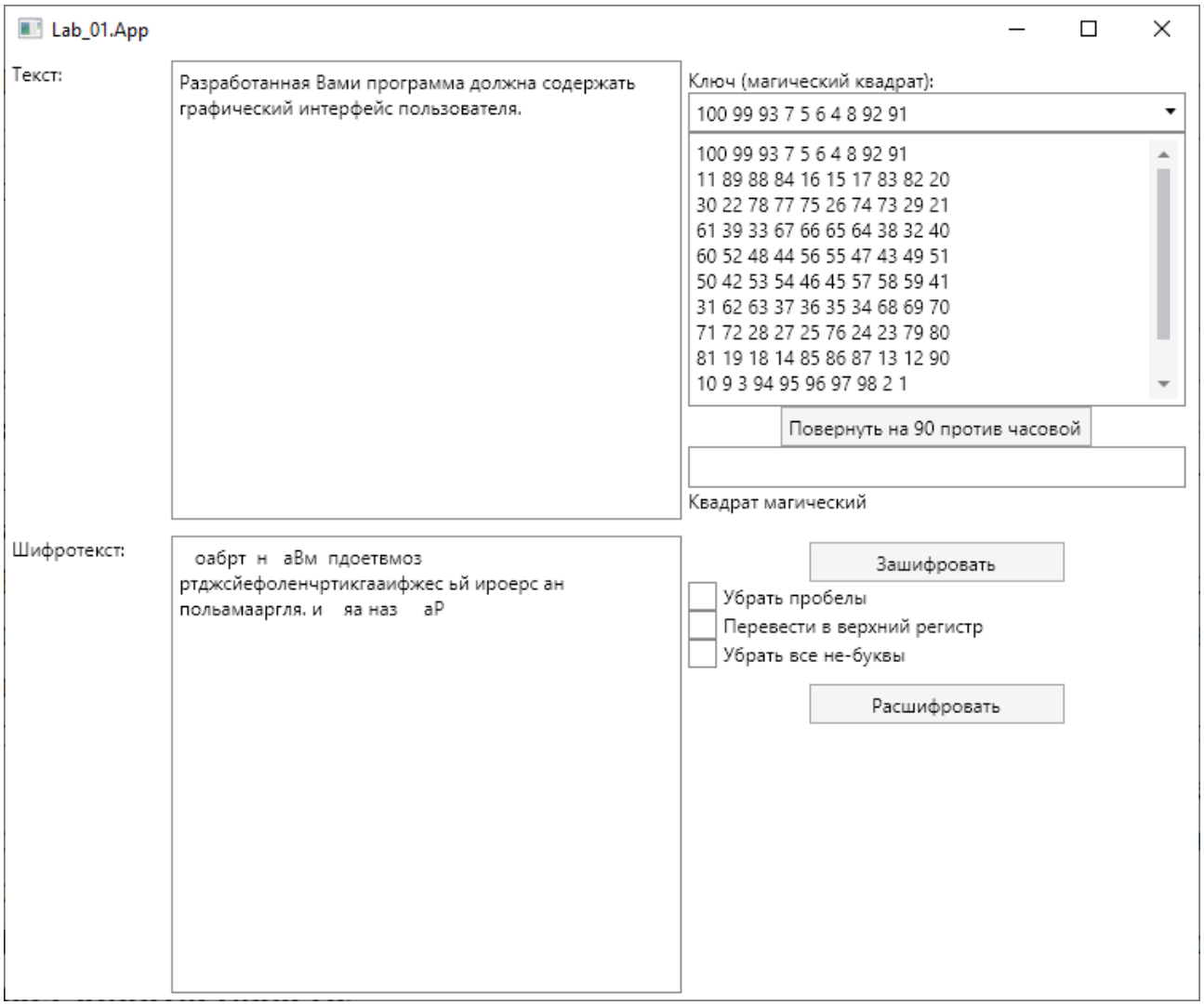

Рисунок 3 – Пример работы программы (шифрование)

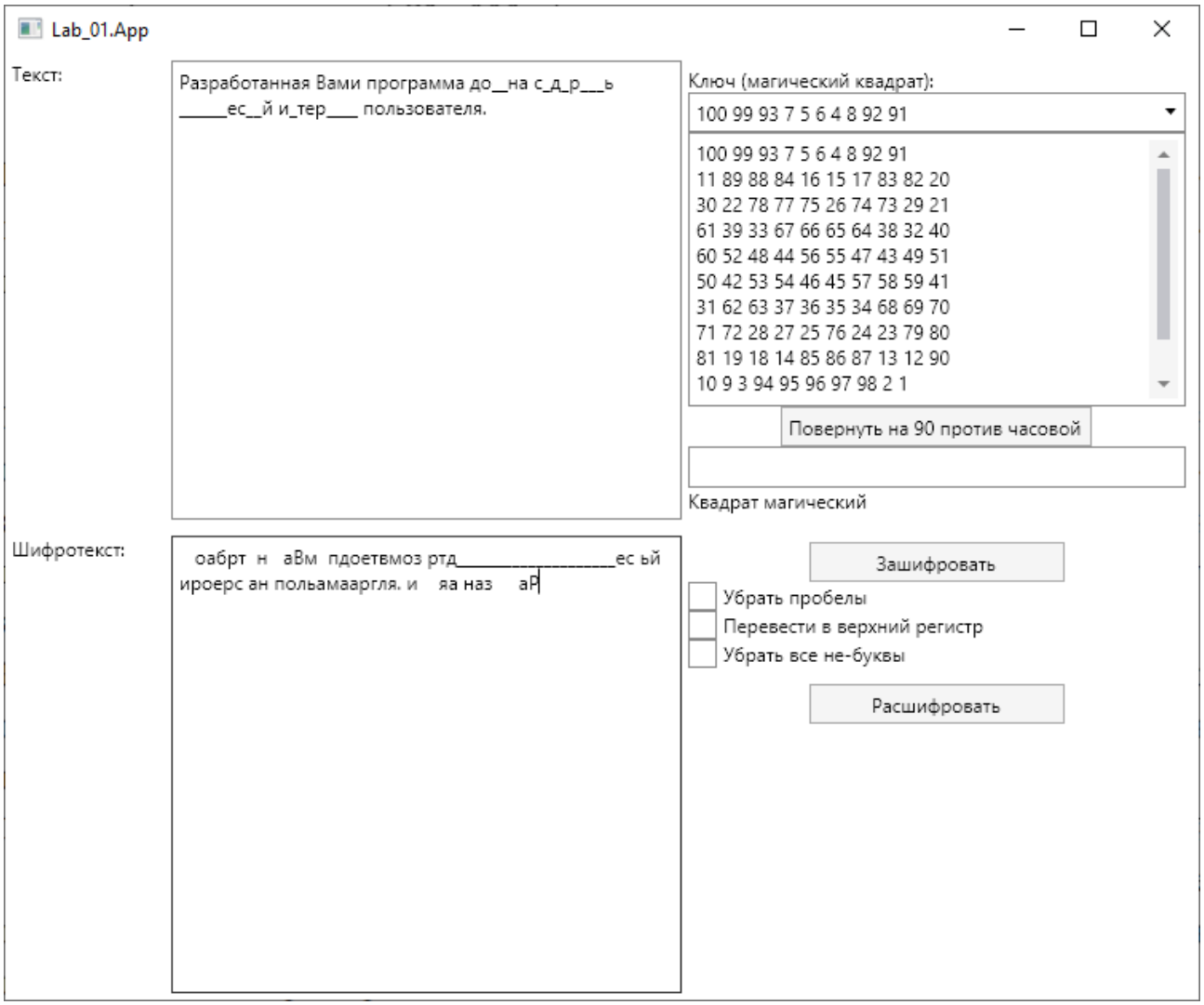

Рисунок 4 – Пример работы программы (дешифрование, часть символов в шифротексте заменена на подчёркивание – и на соответствующих местах в тексте появились подчёркивания)

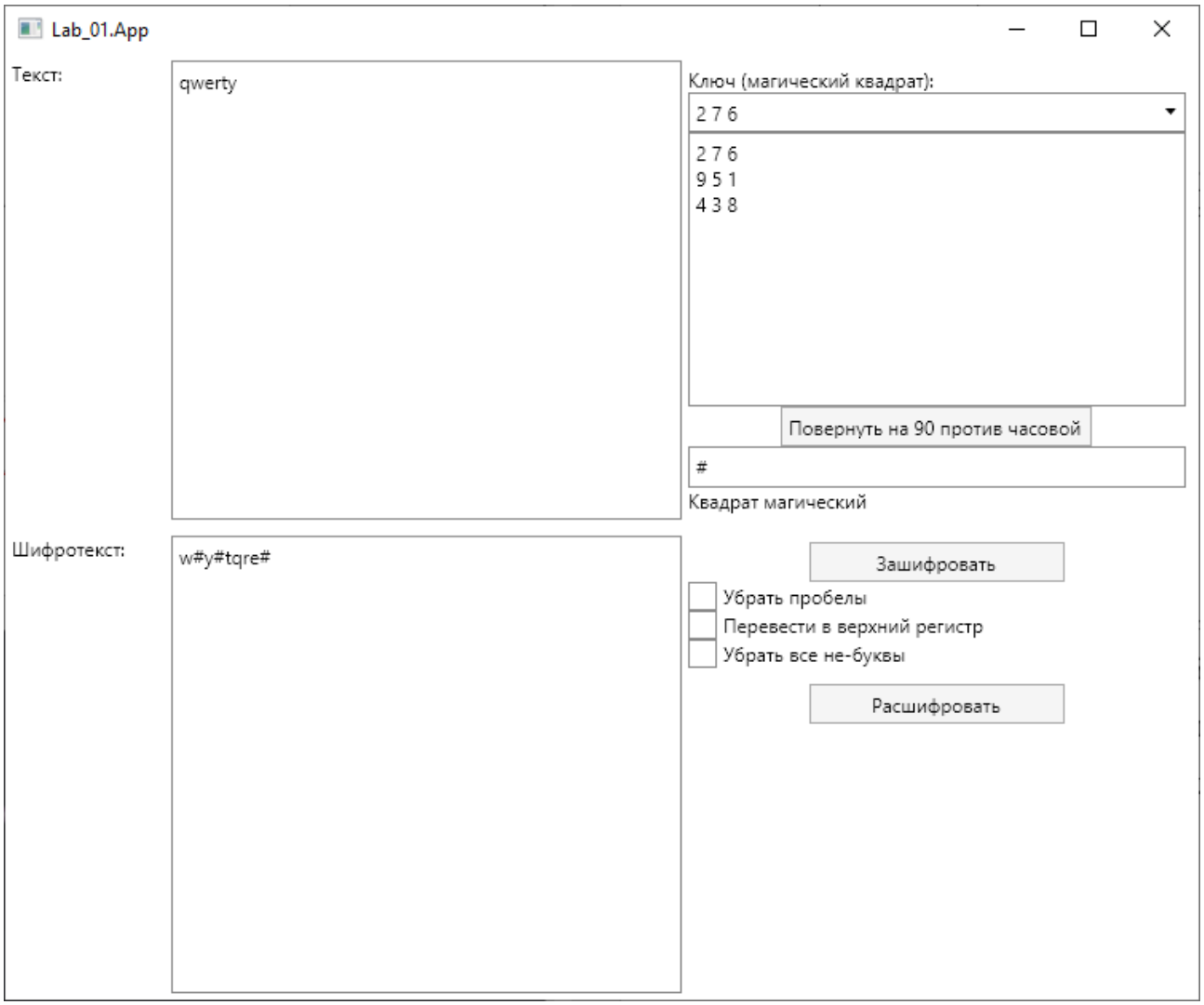

Рисунок 5 – Шифрование, используя ключ 3*3, как символ-заполнитель используется решётка

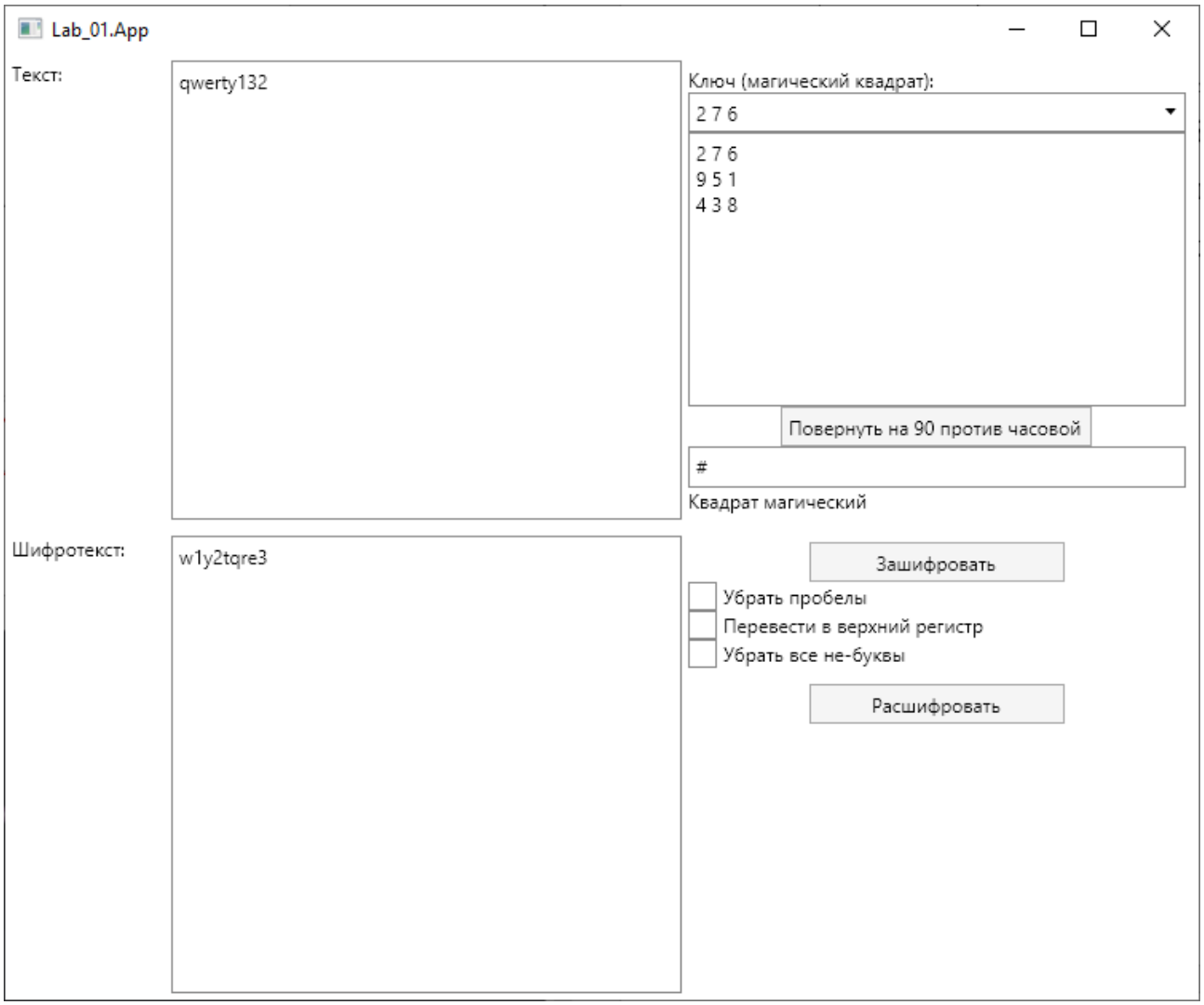

Рисунок 6 – Дешифрование, на месте символов-заполнителей из прошлого примера поставлены цифры; текст стал длиннее и цифрами в конце

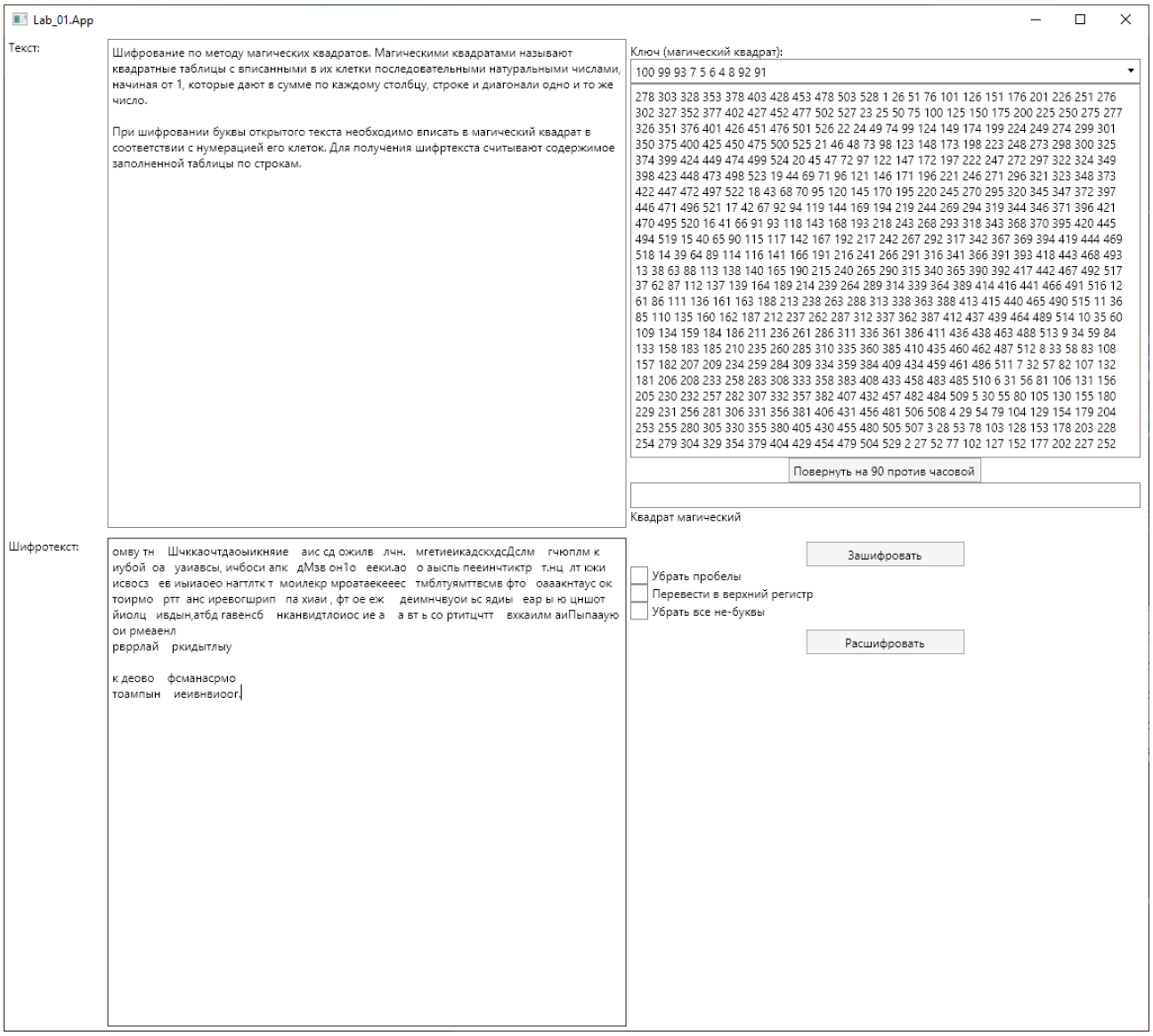

Рисунок 7 – Шифрование более длинного текста с самостоятельно введённым ключом (сгенерирован на [3])

Программу можно улучшить, позволив пользователю поворачивать ключ и модифицировать текст.

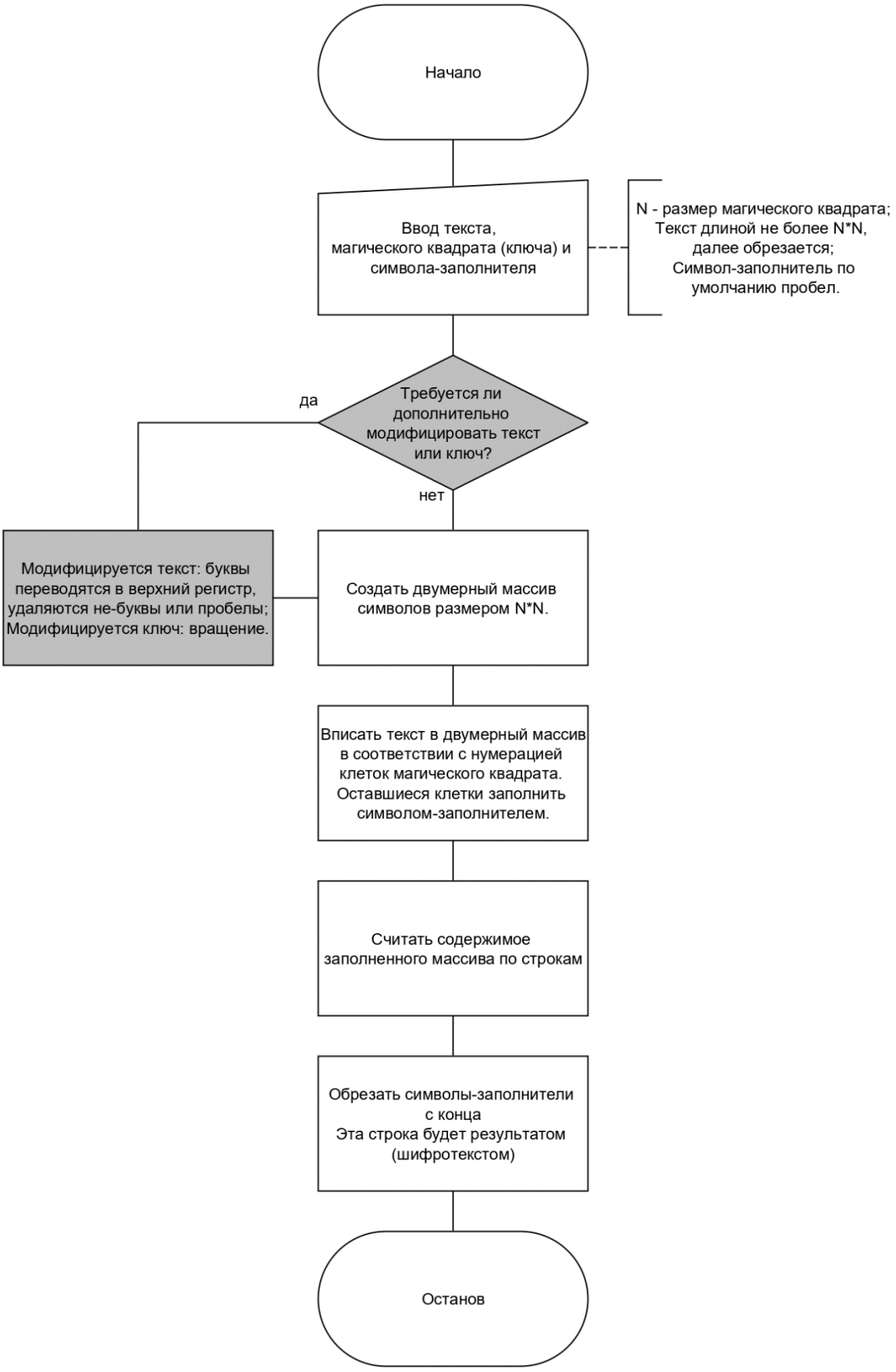

Рисунок 8 – Улучшенный алгоритм шифрования

Листинг 3 – Новый метод RotateAntiClockwise() в классе MagicSquare.

```
/// <summary>
///     Поворачивает квадрат на 90 градусов против часовой.
/// </summary>
public void RotateAntiClockwise()
{
    for (var x = 0; x < Count / 2; x++)
    {
        for (var y = x; y < Count - x - 1; y++)
        {
            var temp = this[x, y];
            this[x, y] = this[y, Count - 1 - x];
            this[y, Count - 1 - x] = this[Count - 1 - x, Count - 1 - y];
            this[Count - 1 - x, Count - 1 - y] = this[Count - 1 - y, x];
            this[Count - 1 - y, x] = temp;
        }
    }
}
```

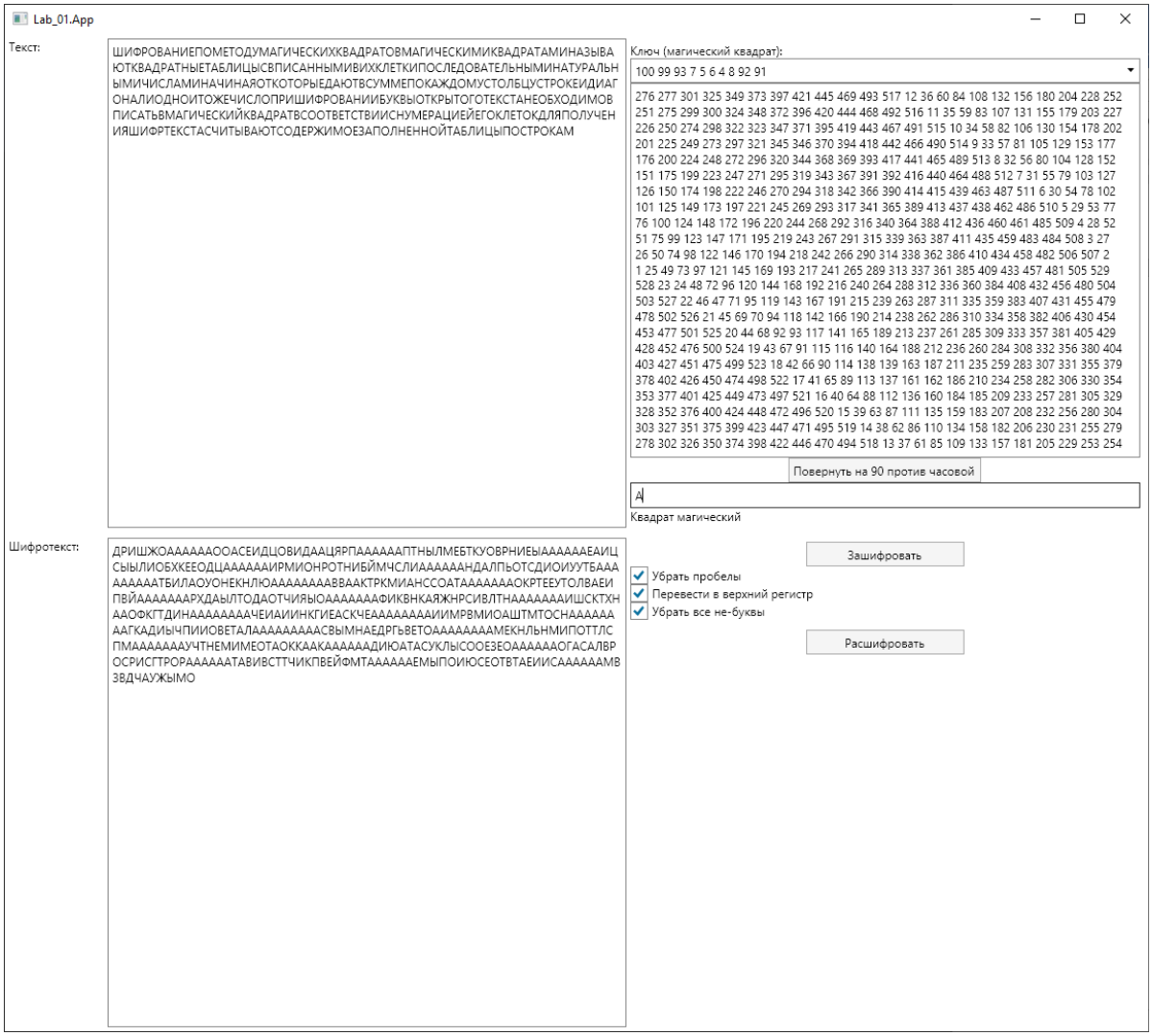

Рисунок 9 – Зашифрованный и расшифрованный текст из прошлого примера, переведённый в верхний регистр и без не-букв, с повёрнутый на 90 градусов против часовой ключом из прошлого примера, «А» как заполнитель

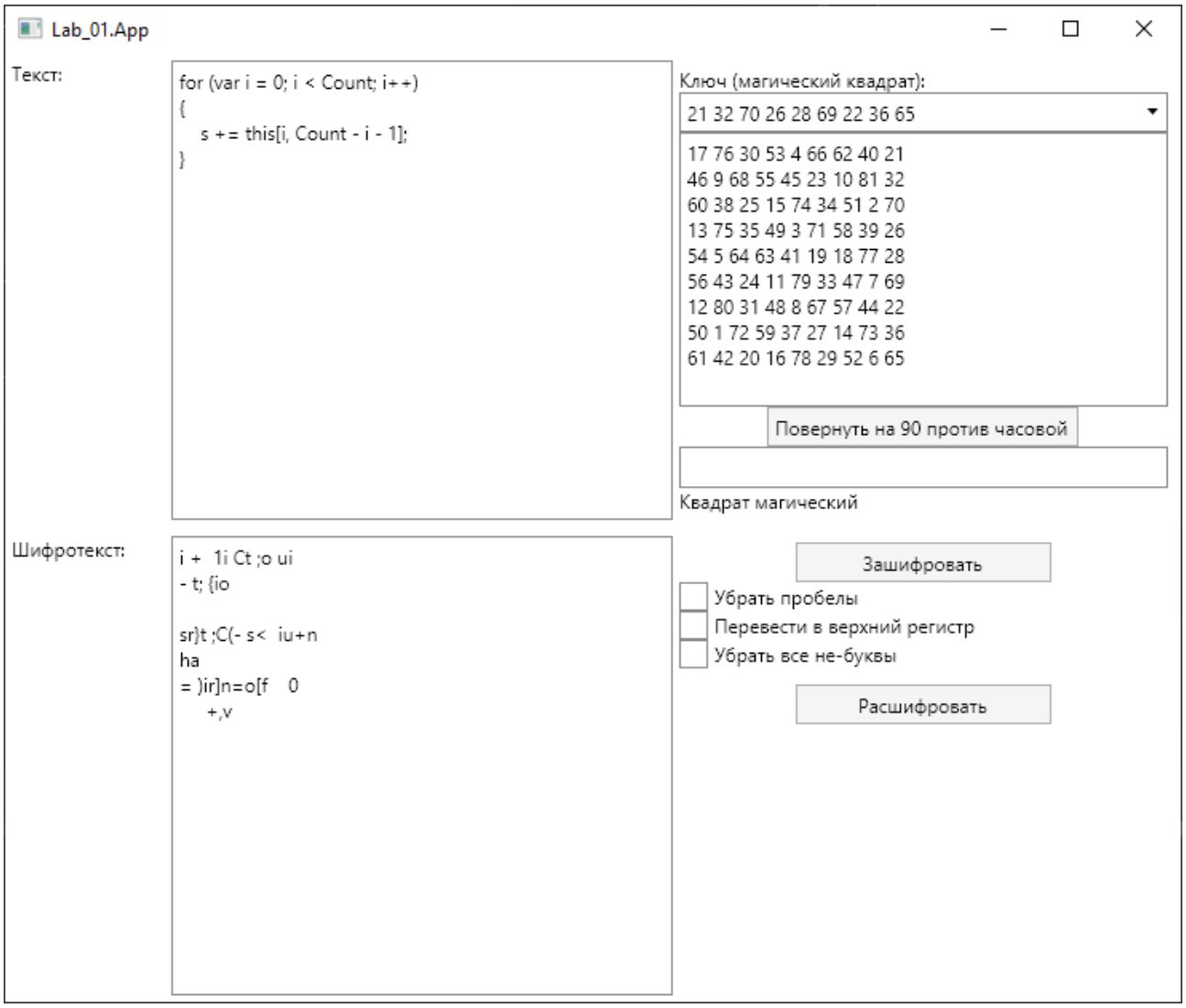

Рисунок 10 – Шифрование используя один из предложенных ключей, повёрнутый на 270 градусов против часовой

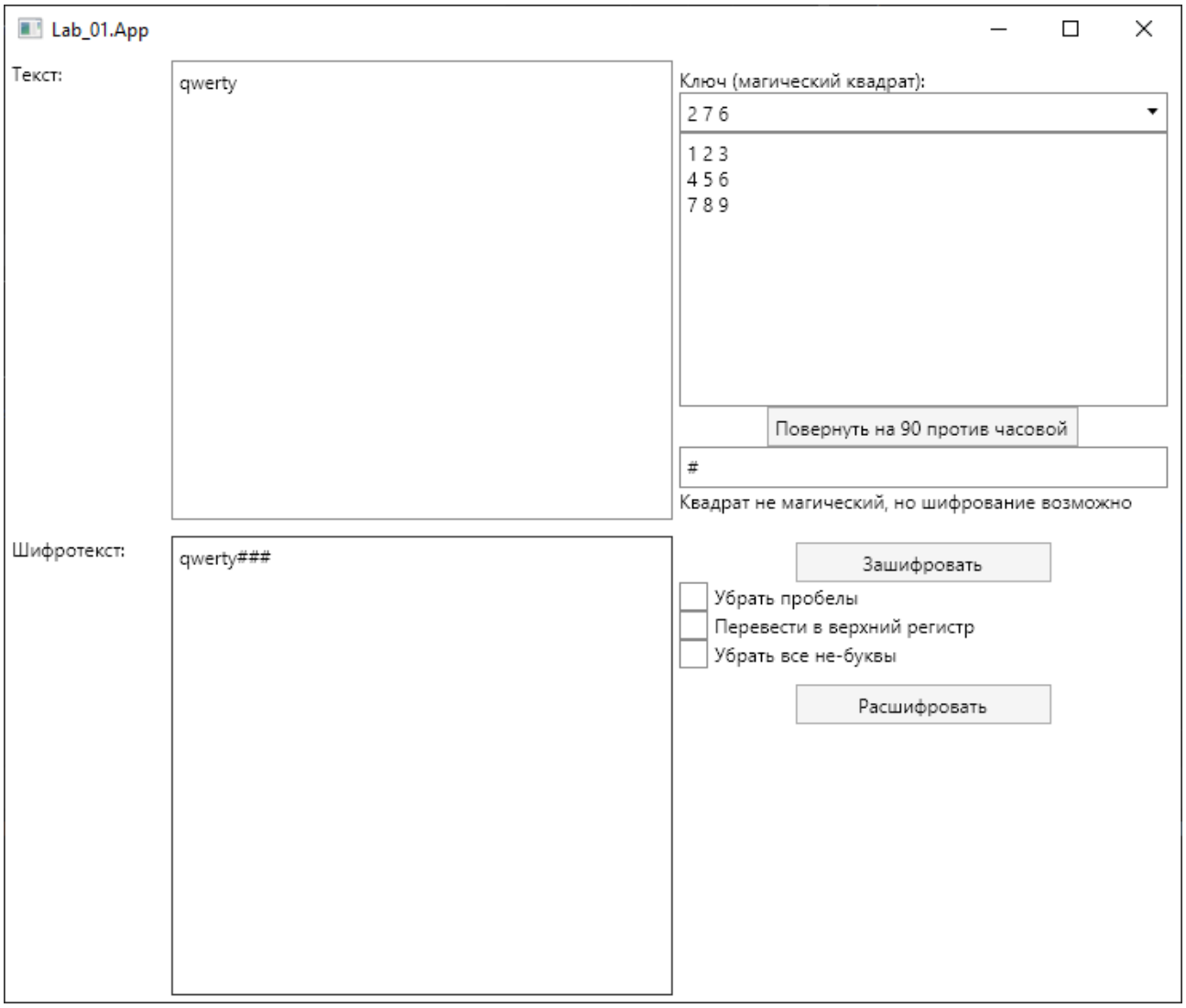

Рисунок 11 – Шифрование, используя не магический квадрат как ключ (пользователь предупреждён, что квадрат не магический), а просто квадрат, заполненный различными натуральными числами от 1

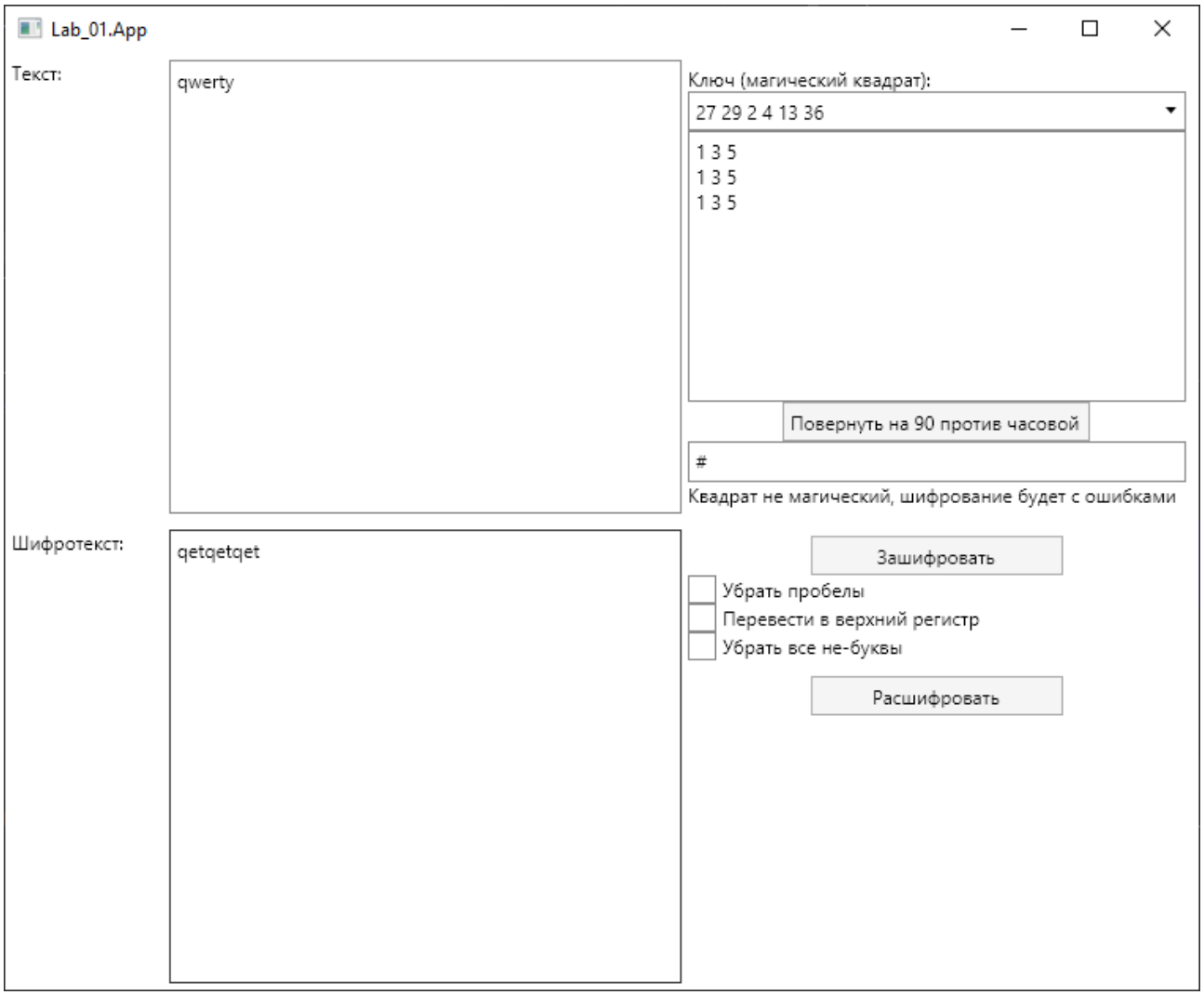

Рисунок 12 – Шифрование, используя ошибочный ключ (пользователь предупреждён об ошибках)

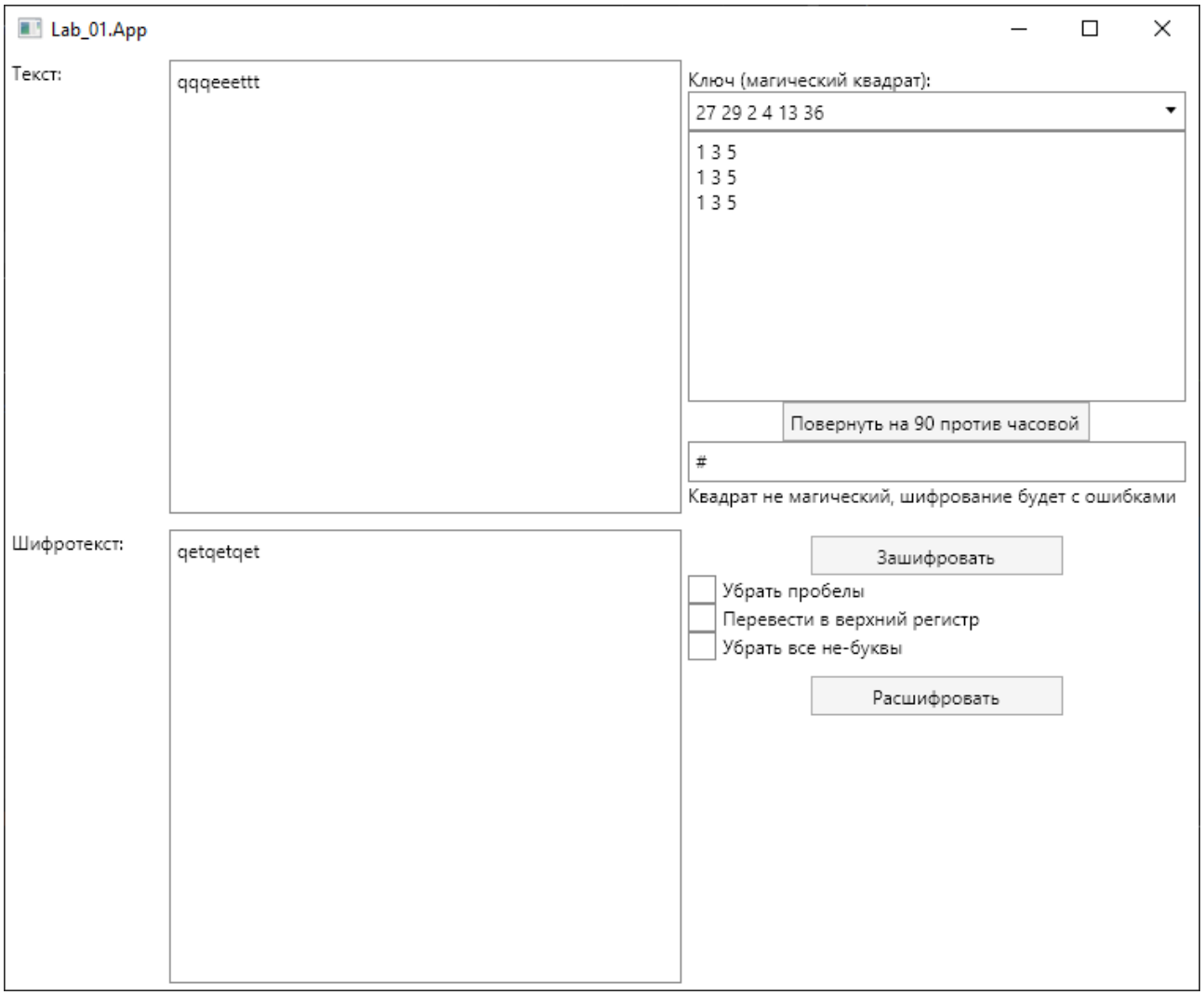

Рисунок 13 – Дешифровка ошибочного шифрования ошибочным ключом

4 Вывод

В входе данной лабораторной работы были освоены основные этапы проектирования и реализации простых симметричных криптографических шифров на основе методов подстановок, перестановок и гаммирования. Реализована программа для шифрования и дешифрования шифрования с помощью магических квадратов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Магический квадрат — Википедия [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Магический_квадрат (Дата обращения: 22.04.2020).

2. Шифрование по методу магических квадратов [Электронный ресурс]. URL: http://neudoff.net/info/informatika/shifrovanie-po-metodu-magicheskix-kvadratov/ (Дата обращения: 23.04.2020).

3. Magic Square Generator - 3, 4, 5, 6, 7, ... - Online Software Tool [Электронный ресурс]. URL: https://www.dcode.fr/magic-square (Дата обращения: 24.04.2020).