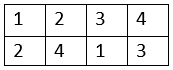
Шифр перестановки: алгоритм, реализация на C#

**Шифр перестановки** (перестановочный шифр) – это один из видов блочного шифра. В статье разберем алгоритм шифрования и реализуем его на языке C# в проекте Windows Forms.

Алгоритм шифра перестановки

Исходное сообщение разбивается на блоки длины m, где m – это длина ключа.

Ключ в шифре перестановки имеет следующий вид:

[](https://vscode.ru/wp-content/uploads/2017/09/Klyuch-shifra-perestanovki.png)

В первой строке таблицы указаны номера символов блока по порядку, а во второй строке указаны номера позиций, которые должны занимать указанные символы в зашифрованном блоке текста.

Кодирование осуществляется перестановкой букв. Таким образом первый символ из исходного блока должен быть переставлен на второе место, второй на четвертое, третий на первое, четвертый на третье.

Если данным ключом зашифровать слово кофе, то получится слово фкео.

Дешифрование производится в обратном порядке. На примере указанного ключа: второй символ из зашифрованного блока ставим на первое место, четвертый на второе, первый на третье, третий на четвертое.

При использовании любого блочного шифра (шифр перестановки не исключение), может возникнуть ситуация, когда текст не делится на равные блоки длины m. То есть остаток от деления длины текста n на длину ключа m не равен нулю.

В таких случаях длину исходного сообщения увеличивают на m – (n % m) символов, чтобы оно делилось на равные блоки длины m.

Реализация шифра перестановки на C#

Напишем шифр перестановки на языке C#. Реализуем его функциональность в виде отдельного класса Transposition.

Приведем код класса, а затем подробно разберем все его компоненты.

C#

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64 | class Transposition  {      private int[] key = null;        public void SetKey(int[] \_key)      {          key = new int[\_key.Length];            for (int i = 0; i < \_key.Length; i++)              key[i] = \_key[i];      }        public void SetKey(string[] \_key)      {          key = new int[\_key.Length];            for (int i = 0; i < \_key.Length; i++)              key[i] = Convert.ToInt32(\_key[i]);      }        public void SetKey(string \_key)      {          SetKey(\_key.Split(' '));      }        public string Encrypt(string input)      {          for (int i = 0; i < input.Length % key.Length; i++)              input += input[i];            string result = "";            for (int i = 0; i < input.Length; i += key.Length)          {              char[] transposition = new char[key.Length];                for (int j = 0; j < key.Length; j++)                  transposition[key[j] - 1] = input[i + j];                for (int j = 0; j < key.Length; j++)                  result += transposition[j];          }            return result;      }        public string Decrypt(string input)      {          string result = "";            for (int i = 0; i < input.Length; i += key.Length)          {              char[] transposition = new char[key.Length];                for (int j = 0; j < key.Length; j++)                  transposition[j] = input[i + key[j] - 1];                for (int j = 0; j < key.Length; j++)                  result += transposition[j];          }            return result;      }  } |

Сначала идет объявление ссылочной переменной под одномерный массив, содержащий ключ (вторую строчку из таблицы ключа). Пока присвоим ей значение null.

C#

|  |  |
| --- | --- |
| 1 | private int[] key = null; |

Затем идут три версии перегруженного метода SetKey, который сохраняет в экземпляр класса значение ключа.

Первая версия – входным аргументом является массив целых чисел:

C#

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | public void SetKey(int[] \_key)  {      key = new int[\_key.Length];        for (int i = 0; i < \_key.Length; i++)          key[i] = \_key[i];  } |

Заметим, что мы не присваиваем ссылку на массив \_key в key, а создаём копию массива \_key.

Таким образом мы страхуемся от внешних воздействий на массив ключа, поскольку ссылка на него приходит в объект класса извне, значит внешний код имеет возможность воздействовать на массив ключа во время работы класса Transposition, а внешний код из класса шифрования мы контролировать не можем.

Поэтому лучшим вариантом является создание копии массива с ключом для шифра перестановки.

Следующая перегруженная версия метода SetKey принимает на вход массив ключа, элементы которого представлены строками. Метод идентичен предыдущему, за исключением того, что строки конвертируются в целые числа с помощью статического класса Convert и его метода ToInt32():

C#

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | public void SetKey(string[] \_key)  {      key = new int[\_key.Length];        for (int i = 0; i < \_key.Length; i++)          key[i] = Convert.ToInt32(\_key[i]);  } |

Третья версия метода принимает на вход строку, в которой содержится ключ. Элементы ключа должны быть отделены друг от друга символом пробела. С помощью метода Split разделяем строку на массив строк по разделителю – пробелу и вызываем для данного массива метод с сигнатурой public void SetKey(int[] \_key).

C#

|  |  |
| --- | --- |
| 1  2  3  4 | public void SetKey(string \_key)  {      SetKey(\_key.Split(' '));  } |

Далее идет метод Encrypt, который непосредственно реализует шифрование с помощью перестановочного шифра. Аргумент метода – строка с исходным текстом.

C#

|  |  |
| --- | --- |
| 1  2  3  4 | public string Encrypt(string input)  {      ...  } |

В начале тела метода мы доводим длину входной строки до числа без остатка делящегося на длину ключа, посредством прибавления в конец строки нужного количества символов из ее начала.

C#

|  |  |
| --- | --- |
| 1  2 | for (int i = 0; i < input.Length % key.Length; i++)      input += input[i]; |

Условие выхода из цикла хоть и не эквивалентно i < m – (n % m), но тем не менее работает и является более элегантным – цикл продолжается до тех пор, пока длина сообщения не будет нацело делиться на длину ключа.

После каждой итерации значение input.Length меняется, поскольку длина строки увеличивается на один символ. Выражение input.Length % key.Length всегда показывает сколько символов остается в остатке на момент вызова, и оно всегда будет больше, чем i (если остаток не равен нулю). Когда деление будет выполняться нацело (т.е. i < 0 даст ложь), произойдет выход из цикла.

Объявляем строковую переменную, в которой будем хранить результат – зашифрованный текст:

C#

|  |  |
| --- | --- |
| 1 | string result = ""; |

Далее происходит непосредственно шифрование с помощью шифра перестановки:

C#

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | for (int i = 0; i < input.Length; i += key.Length)  {      char[] transposition = new char[key.Length];        for (int j = 0; j < key.Length; j++)          transposition[key[j] - 1] = input[i + j];        for (int j = 0; j < key.Length; j++)          result += transposition[j];  } |

В цикле (строки 1-10) проходимся по всем блокам исходного сообщения длиной в ключ.

На каждой итерации этого цикла создается символьный массив transposition (строка 3), в который будем заносить зашифрованный текст (переставленные символы) – символы в новом порядке исходя из ключа.

Затем во внутреннем цикле (строки 5-6) перебираем все символы блока и происходит непосредственно шифрование (перестановка) по ключу (строка 6).

После в цикле (строки 8-9) посимвольно прибавляем зашифрованный блок к концу строковой переменной с результатом (строка 9).

На этом шифрование перестановками завершено. Возвращаем из метода результат:

C#

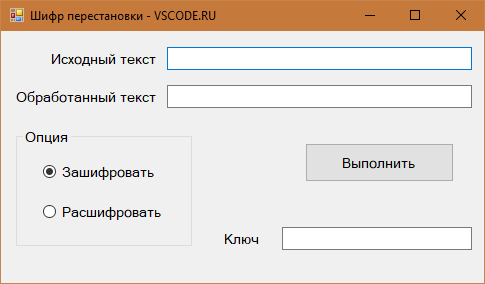
|  |  |
| --- | --- |
| 1 | return result; |

Метод дешифрования Decrypt по структуре идентичен методу шифрования Encrypt, за исключением того, что в нем отсутствует код, доводящий входную строку до нужно длины, поскольку зашифрованный текст уже имеет необходимую длину. И, конечно, в данном методе происходит дешифрование, поэтому код по перестановке символов другой:

C#

|  |  |
| --- | --- |
| 1 | transposition[j] = input[i + key[j] - 1]; |

Создадим проект Windows Forms, к котором продемонстрируем работу перестановочного шифра. На форме разместим нужные компоненты следующим образом:

[](https://vscode.ru/wp-content/uploads/2017/09/Interfeys-programmyi.png)

В коде класса Form1 объявим поле со ссылкой на экземпляр класса Transposition, а в конструкторе класса создадим данный объект:

C#

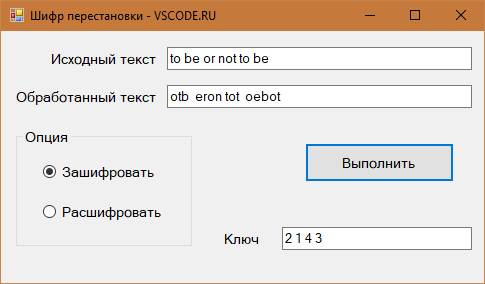
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | public partial class Form1 : Form  {      Transposition t;        public Form1()      {          InitializeComponent();            t = new Transposition();      }  } |

В обработчике события нажатия кнопки “Выполнить” вызовем метод установки ключа, а также реализуем логику работы шифрования и дешифрования в зависимости от выбранного на форме переключателя radioButton:

C#

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | private void startButton\_Click(object sender, EventArgs e)  {      t.SetKey(keyTextBox.Text);        if (encryptRadioButton.Checked)          outputTextBox.Text = t.Encrypt(inputTextBox.Text);      else          outputTextBox.Text = t.Decrypt(inputTextBox.Text);  } |

Продемонстрируем программу “Шифр перестановки” в действии:

[](https://vscode.ru/wp-content/uploads/2017/09/Rabota-programmyi.png)