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 №12**

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

**Теория:**

Для шифрования открытого текста используются шифрблокноты – обычные бумажные блокноты, на каждом листе которых написан ключ шифрования. Как у отправителя, так и у получателя должны быть два заранее составленных одинаковых шифрблокнота. Когда отправитель хочет отправить сообщение, он проделывает следующие действия: берет первую букву открытого текста и первую букву из шифрблокнота, затем побитово их складывает (с помощью операции исключающее ИЛИ) и получает первую букву шифртекста. Далее отправитель побитово складывает вторую букву открытого текста и вторую букву из шифрблокнота и получает вторую букву шифртекста. Он будет продолжать проделывать те же действия, пока весь открытый текст не будет зашифрован. Затем он посылает свой зашифрованный текст получателю по каналу связи и сразу же обязательно уничтожает использованную страницу своего шифрблокнота.

Получив шифртекст, получатель, чтобы расшифровать его, проделывает те же самые действия, что и отправитель, а именно: побитово складывает первую букву полученной шифртекста с первой буквой своего шифрблокнота, вторую - со второй, и т.д. пока не расшифрует весь текст, после чего он обязательно уничтожает использованную страницу своего шифрблокнота.

* Длина ключа должна совпадать с длиной сообщения.
* Нельзя шифровать разные сообщения одним и тем же ключом (т.е. не использовать одну и ту же страницу шифроблокнота повторно).
* Необходимо составлять страницы шифроблокнота случайным образом и составлять их без повторений цепочек символов.
* После передачи зашифрованного сообщения отправитель должен обязательно уничтожать использованную страницу шифрблокнота.

**Исходный код:**

**Информация об алфавите:**

public string alf = "abcdefghijklmnopqrstuvwxyz" + "ABCDEFGHIJKLMNOPQRSTUVWXYZ" +

"абвгдеёжзийклмнопрстуфхцчшщъыьэюя" + "АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ" +

"0123456789" + " .,?!~`@#№$;%^:&\*()-\_+=[]{}|/<>" + "„…†‡€‰‘’“”•—" +

"™¤¦§©Є«»¬®°±µ¶·£¥≠≤≥÷×∞αβπΩ∑←↑→↓↔↕↨∆∏√∟∩∫≡⌂⌠⌡◊♀♂♠♣♥♦♪♫ⱰⱭⱲⱳⱴꞈ¼½¾²³¿ˇ˅ˣˢΞΘΛΨλζεδγϕϞϬϼѡ⅓";

int bitsForOneSymbol = 8;

**Генерация ключа:**

private void f12\_ButtonGenKeyword\_Click(object sender, EventArgs e)

{

Random rnd = new Random();

string generated\_key = "";

for (int i = 0; i < f12\_fieldOriginal.TextLength; i++)

generated\_key = generated\_key + alf[rnd.Next(alf.Length)];

f12\_fieldKeyword.Text = generated\_key;

if (f12\_fieldOriginal.TextLength == 0)

{

MessageBox.Show("Сначала введите исходный текст", "Предупреждение");

f12\_fieldOriginal.Text = null;

}

}

**Функция перевода строки в последовательность битов:**

private byte [,] TextToBits (string text)

{

byte[] decimal\_code = new byte[text.Length];

for (int i = 0; i < decimal\_code.Length; i++)

decimal\_code[i] = (byte)alf.IndexOf(text[i]);

//от буквенной строки перейдем к десятичным числам - номеру символа в алфавите

byte[,] bin\_code = new byte[text.Length, bitsForOneSymbol];

//создадим ячейки под двоичный код - по 8 бит на букву

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

{

double curCode = decimal\_code[i];

for (int j = bitsForOneSymbol - 1; j >= 0; j--)

if (curCode - Math.Pow(2, j) >= 0)

{

curCode -= Math.Pow(2, j);

bin\_code[i,j] = 1;

}

else

{

bin\_code[i, j] = 0;

}

}

return bin\_code;

}

**Функция перевода последовательности битов в строку**

private string BitsToText (byte[,] bytes)

{

byte [] decimal\_code = new byte [bytes.Length / bitsForOneSymbol];

Array.Clear(decimal\_code, 0, decimal\_code.Length);

for (int i = 0; i < decimal\_code.Length; i++)

for (int j = bitsForOneSymbol - 1; j >= 0; j--)

{

decimal\_code[i] += (byte)(bytes[i,j] \* Math.Pow(2, j));

}

string result = "";

for (int i = 0; i < decimal\_code.Length; i++)

result += alf[decimal\_code[i]];

return result;

}

**Алгоритм зашифрования:**

private void f12\_ButtonEncrypt\_Click(object sender, EventArgs e)

{

string Source = f12\_fieldOriginal.Text;

string entered\_key = f12\_fieldKeyword.Text;

//Составим строку из символов текста и ключа, отсутствующих в алфавите

string prohibited\_symbs = "";

for (int i = Source.Length - 1; i >= 0; i--)

if (alf.IndexOf(Source[i]) < 0)

prohibited\_symbs += Source[i];

for (int i = entered\_key.Length - 1; i >= 0; i--)

if (alf.IndexOf(entered\_key[i]) < 0)

prohibited\_symbs += entered\_key[i];

//Удалим все встретившиеся запрещенные символы

for (int i = 0; i < prohibited\_symbs.Length; i++)

{

Source = Regex.Replace(Source,Convert.ToString(prohibited\_symbs[i]),"");

entered\_key = Regex.Replace(entered\_key, Convert.ToString(prohibited\_symbs[i]), "");

}

//Убедимся, что размер ключа не меньше длины исходного текста

if (entered\_key.Length < Source.Length)

{

MessageBox.Show("Слишком короткий ключ", "Ошибка");

goto metka\_exit;

}

//Отразим очищенные от запрещенных символов данные в полях

f12\_fieldOriginal.Text = Source;

f12\_fieldKeyword.Text = entered\_key;

//Переведем данные в последовательность битов

byte[,] mes\_bin = TextToBits(Source);

byte[,] key\_bin = TextToBits(entered\_key);

//Соберем битовые строки просто посмотреть на них

string key\_bin\_str = "";

string mes\_bin\_str = "";

for (int i = 0; i < key\_bin.Length/bitsForOneSymbol; i++)

for (int j = bitsForOneSymbol - 1; j >= 0; j--)

{

key\_bin\_str += Convert.ToString(key\_bin[i,j]);

}

for (int i = 0; i < mes\_bin.Length / bitsForOneSymbol; i++)

for (int j = bitsForOneSymbol - 1; j >= 0; j--)

{

mes\_bin\_str += Convert.ToString(mes\_bin[i, j]);

}

//Побитовое сложение по модулю 2

byte[,] result = new byte[mes\_bin.Length / bitsForOneSymbol, bitsForOneSymbol];

for (int i = 0; i < mes\_bin.Length / bitsForOneSymbol; i++)

for (int j = bitsForOneSymbol - 1; j >= 0; j--)

result[i, j] = (byte)((mes\_bin[i, j] + key\_bin[i, j]) % 2);

f12\_fieldCiphertext.Text = BitsToText(result);

metka\_exit:;

}

**Алгоритм расшифрования:**

private void f12\_ButtonDecrypt\_Click(object sender, EventArgs e)

{

string Source = f12\_fieldCiphertext.Text;

string entered\_key = f12\_fieldKeyword.Text;

//Составим строку из символов текста и ключа, отсутствующих в алфавите

string prohibited\_symbs = "";

for (int i = Source.Length - 1; i >= 0; i--)

if (alf.IndexOf(Source[i]) < 0)

prohibited\_symbs += Source[i];

for (int i = entered\_key.Length - 1; i >= 0; i--)

if (alf.IndexOf(entered\_key[i]) < 0)

prohibited\_symbs += entered\_key[i];

//Удалим все встретившиеся запрещенные символы

for (int i = 0; i < prohibited\_symbs.Length; i++)

{

Source = Regex.Replace(Source, Convert.ToString(prohibited\_symbs[i]), "");

entered\_key = Regex.Replace(entered\_key, Convert.ToString(prohibited\_symbs[i]), "");

}

//Убедимся, что размер ключа не меньше длины исходного текста

if (entered\_key.Length < Source.Length)

{

MessageBox.Show("Слишком короткий ключ", "Ошибка");

goto metka\_exit;

}

//Отразим очищенные от запрещенных символов данные в полях

f12\_fieldCiphertext.Text = Source;

f12\_fieldKeyword.Text = entered\_key;

//Переведем данные в последовательность битов

byte[,] mes\_bin = TextToBits(Source);

byte[,] key\_bin = TextToBits(entered\_key);

//Соберем битовые строки просто посмотреть на них

string key\_bin\_str = "";

string mes\_bin\_str = "";

for (int i = 0; i < key\_bin.Length / bitsForOneSymbol; i++)

for (int j = bitsForOneSymbol - 1; j >= 0; j--)

{

key\_bin\_str += Convert.ToString(key\_bin[i, j]);

}

for (int i = 0; i < mes\_bin.Length / bitsForOneSymbol; i++)

for (int j = bitsForOneSymbol - 1; j >= 0; j--)

{

mes\_bin\_str += Convert.ToString(mes\_bin[i, j]);

}

//Побитовое сложение по модулю 2

byte[,] result = new byte[mes\_bin.Length / bitsForOneSymbol, bitsForOneSymbol];

for (int i = 0; i < mes\_bin.Length / bitsForOneSymbol; i++)

for (int j = bitsForOneSymbol - 1; j >= 0; j--)

result[i, j] = (byte)(((mes\_bin[i, j] - key\_bin[i, j]) + 2) % 2);

f12\_fieldOriginal.Text = BitsToText(result);

metka\_exit:;

}

**Результат работы программы:**

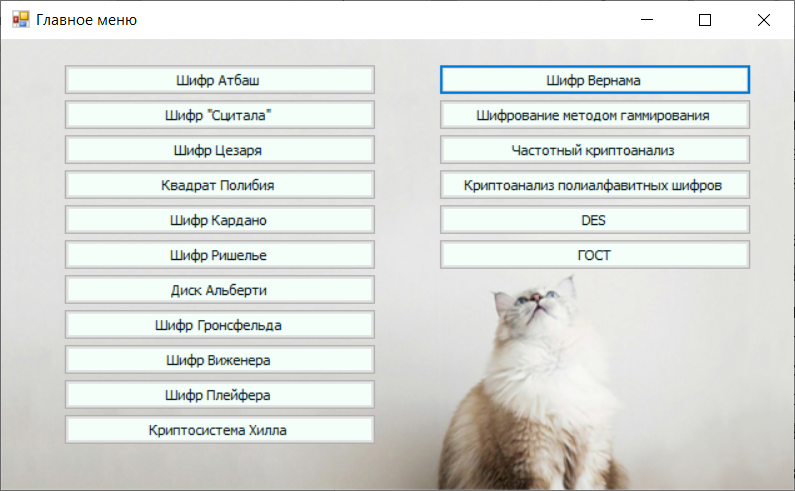


Рисунок 1. Главное меню программы.

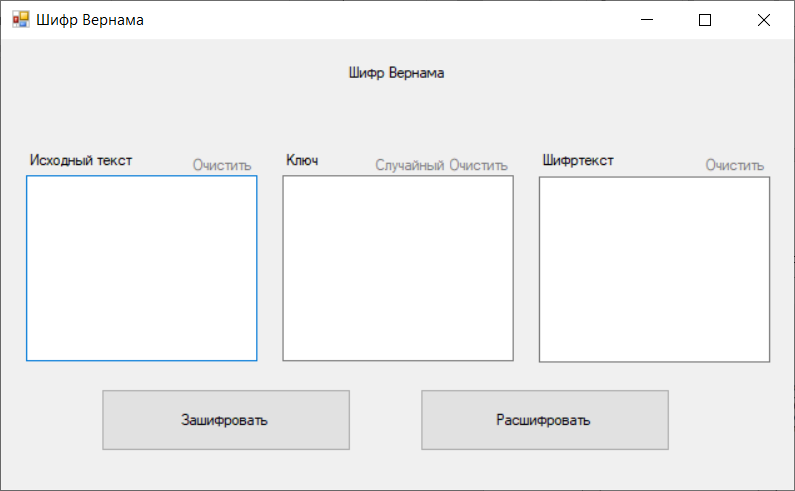


Рисунок 2. Окно шифра Вернама.

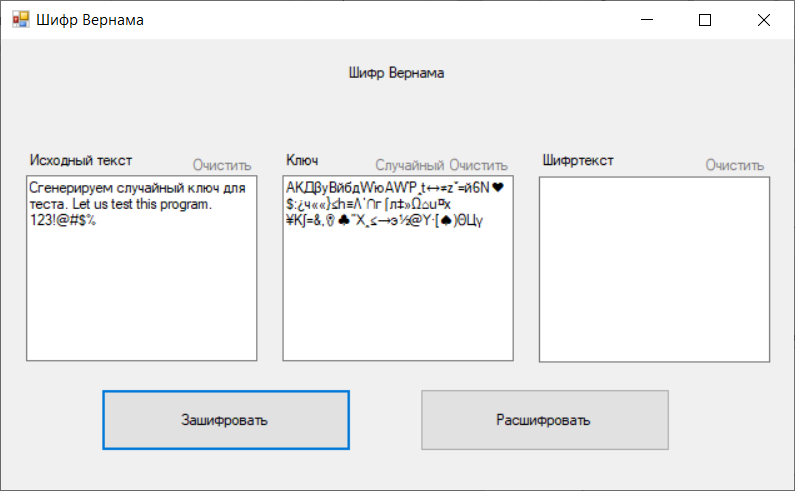


Рисунок 3. Исходный текст.

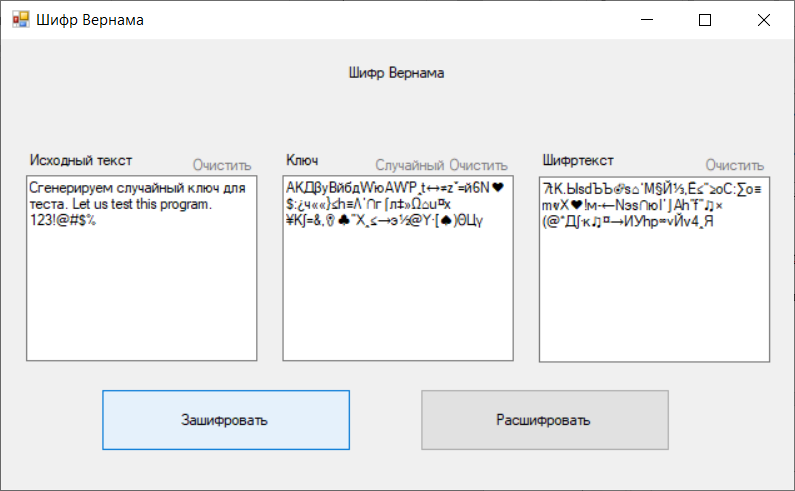


Рисунок 4. Шифртекст (результат зашифрования).

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

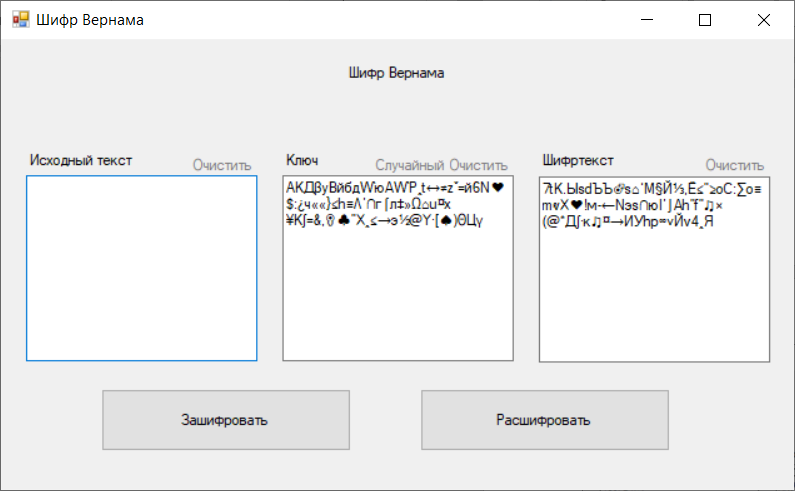


Рисунок 5. Очистка поля Исходный текст.

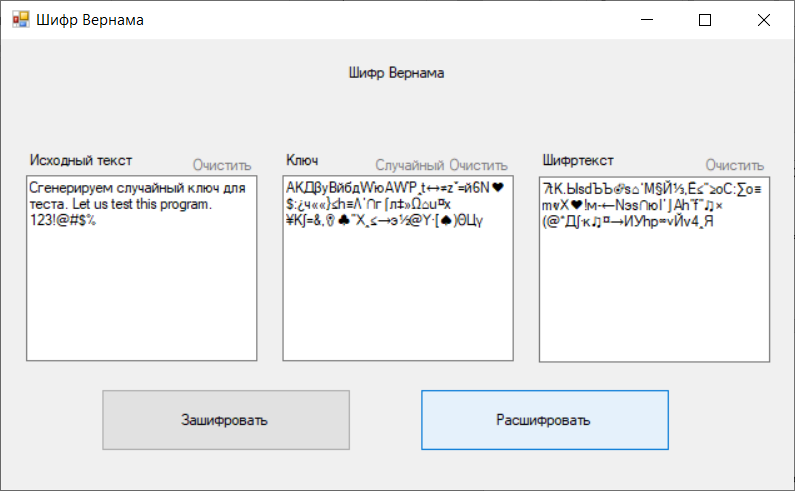


Рисунок 6. Исходный текст (результат расшифрования).

**Вывод:**

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