Задание по технологиям защиты информации

На тему: Шифр Цезаря

Горшков Ян

**Задание:**

Построить графическую программу, позволяющую шифровать исходный текст шифром Цезаря на определенный сдвиг, и дешифровать текст. Текст можно вводить как вручную в текстовое поле, так и читать из готового текстового файла. Результат может быть также записан в файл.

**Теоретические сведения:**

Шифр Цезаря, также известный как шифр сдвига, код Цезаря или сдвиг Цезаря — один из самых простых и наиболее широко известных методов шифрования. Он представляет собой вид шифра подстановки, в котором каждый символ в открытом тексте заменяется символом, находящимся на некотором постоянном числе позиций левее или правее него в алфавите. Схема шифра изображена на рисунке 1.

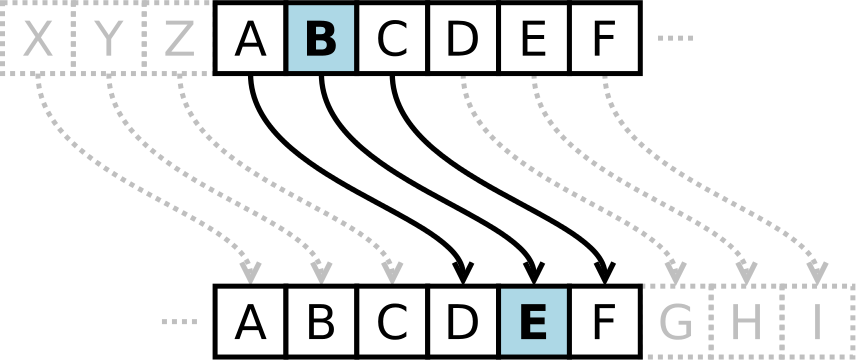


Рисунок 1 – Схема шифра

Как и все моноалфавитные шифры, шифр Цезаря легко взламывается и не имеет почти никакого применения на практике.

Если сопоставить каждому символу алфавита его порядковый номер (нумеруя с 0), то шифрование и дешифрование можно выразить формулами модульной арифметики:

y= (z+k) mod n

x=(y-k+n) mod n,

где x — символ открытого текста, y — символ шифрованного текста, n — мощность алфавита, а k — ключ.

**Разработанная программа:**

Программа разрабатывалась на языке C# при помощи Windows Forms. Интерфейс программы изображен на рисунке 2.

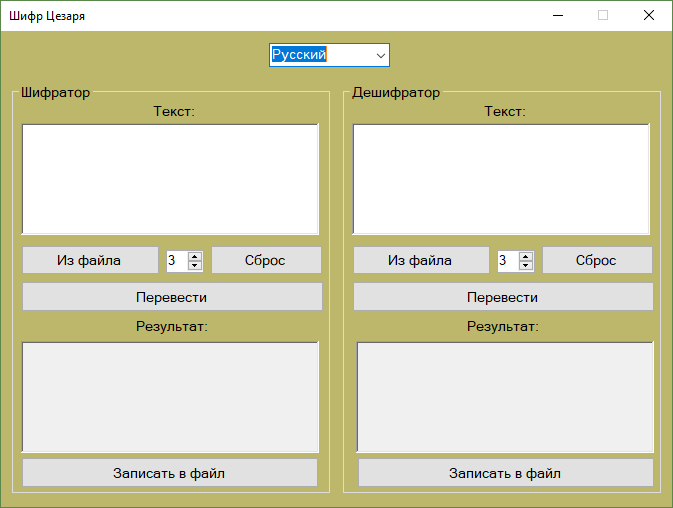


Рисунок 2 – Интерфейс программы

Программа позволяет шифровать и дешифровать текст на двух языках: на русском и на английском. Выбор языка осуществляется в верхней части окна в выпадающем списке. В зависимости от выбора языка, будет изменяться и язык интерфейса. Графический интерфейс поделен на 2 группы: шифратор и дешифратор. Это позволяет работать с двумя компонентами одновременно. Шифруемый и дешифруемый текст может быть введен как вручную в текстовое поле, так и прочитан из текстового файла. В обоих случаях указывается ключ шифра (сдвиг), и при нажатии кнопки “Перевести” в нижнем поле будет выведен результат операции шифрования или дешифрования. Для сохранения результата предусмотрена кнопка “Записать в файл”. Кнопка “Сброс” позволяет очистить текстовое поле.

**Вывод:**

В данной практической работе была разработана программа шифратор-дешифратор, работающая по принципу шифра Цезаря. Данная работа позволила понять принцип шифрования, реализовать его при помощи языка программирования, и проверить работоспособность на практике. Данный шифр является простым моноалфавитным шифром, который, на сегодняшний день, не находит себе применения из-за простоты взлома.

**Листинг:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.IO;

namespace Ceaser

{

public partial class MainForm : Form

{

public MainForm()

{

InitializeComponent();

}

private int words = 33;

private int upperASCII = 1040;

private int lowerASCII =1072;

private void comboBox\_Language\_SelectedValueChanged(object sender, EventArgs e)

{

if (comboBox\_Language.Text == "English")

{

this.Text = "Cipher of Caesar";

groupBox\_Coder.Text = "Coder";

groupBox\_Decoder.Text = "Decoder";

label\_CoderText.Text = "Text";

label\_DecoderText.Text = "Text";

button\_CoderClear.Text = "Clear";

button\_DecoderClean.Text = "Clear";

button\_CoderFile.Text = "From file";

button\_DecoderFile.Text = "From file";

label\_CoderResult.Text = "Result";

label\_DecoderResult.Text = "Result";

button\_CoderGo.Text = "Translate";

button\_DecoderGo.Text = "Translate";

button\_DecoderOutFIle.Text = "Write in File";

button\_CoderOutFile.Text = "Write in File";

numericUpDown\_CoderCount.Maximum = 26;

numericUpDown1\_DecoderCount.Maximum = 26;

words = 26;

upperASCII = 65;

lowerASCII = 97;

}

else

{

this.Text = "Шифр Цезаря";

groupBox\_Coder.Text = "Шифратор";

groupBox\_Decoder.Text = "Дешифратор";

label\_CoderText.Text = "Текст";

label\_DecoderText.Text = "Текст";

button\_CoderClear.Text = "Сброс";

button\_DecoderClean.Text = "Сброс";

button\_CoderFile.Text = "Из файла";

button\_DecoderFile.Text = "Из файла";

label\_CoderResult.Text = "Результат";

label\_DecoderResult.Text = "Результат";

button\_CoderGo.Text = "Перевести";

button\_DecoderGo.Text = "Перевести";

button\_DecoderOutFIle.Text = "Записать в файл";

button\_CoderOutFile.Text = "Записать в файл";

numericUpDown\_CoderCount.Maximum = 33;

numericUpDown1\_DecoderCount.Maximum = 33;

words = 33;

upperASCII = 1040;

lowerASCII = 1072;

}

}

private void button\_CoderClear\_Click(object sender, EventArgs e)

{

richTextBox\_CoderIn.Text = String.Empty;

richTextBox\_CoderOut.Text = String.Empty;

}

private void button\_DecoderClean\_Click(object sender, EventArgs e)

{

richTextBox\_DecoderIn.Text = String.Empty;

richTextBox\_DecoderOut.Text = String.Empty;

}

private void button\_CoderFile\_Click(object sender, EventArgs e)

{

OpenFileDialog open = new OpenFileDialog();

try

{

if (open.ShowDialog() == DialogResult.OK)

richTextBox\_CoderIn.Text = File.ReadAllText(open.FileName, Encoding.Default);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void button\_DecoderFile\_Click(object sender, EventArgs e)

{

OpenFileDialog open = new OpenFileDialog();

try

{

if (open.ShowDialog() == DialogResult.OK)

richTextBox\_DecoderIn.Text = File.ReadAllText(open.FileName, Encoding.Default);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void button\_CoderGo\_Click(object sender, EventArgs e)

{

string result = String.Empty;

foreach (char item in richTextBox\_CoderIn.Text)

{

if (!Char.IsLetter(item))

{

result += item;

continue;

}

if (Char.IsUpper(item))

result += Char.ConvertFromUtf32(upperASCII+(int)(item- upperASCII + numericUpDown\_CoderCount.Value) % words);

else

result += Char.ConvertFromUtf32(lowerASCII + (int)(item - lowerASCII + numericUpDown\_CoderCount.Value) % words);

}

richTextBox\_CoderOut.Text = result;

}

private void button\_DecoderGo\_Click(object sender, EventArgs e)

{

string result = String.Empty;

foreach (char item in richTextBox\_DecoderIn.Text)

{

if (!Char.IsLetter(item))

{

result += item;

continue;

}

if (Char.IsUpper(item))

result += Char.ConvertFromUtf32(upperASCII + (int)(item - upperASCII - numericUpDown1\_DecoderCount.Value + words) % words);

else

result += Char.ConvertFromUtf32(lowerASCII + (int)(item - lowerASCII - numericUpDown1\_DecoderCount.Value + words) % words);

}

richTextBox\_DecoderOut.Text = result;

}

private void button\_CoderOutFile\_Click(object sender, EventArgs e)

{

SaveFileDialog save = new SaveFileDialog();

save.DefaultExt = "txt";

try

{

if (save.ShowDialog() == DialogResult.OK)

{

File.WriteAllText(save.FileName, richTextBox\_CoderOut.Text,Encoding.Unicode);

MessageBox.Show("File was saved succesfully!");

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message,"Error",MessageBoxButtons.OK,MessageBoxIcon.Error);

}

}

private void button\_DecoderOutFIle\_Click(object sender, EventArgs e)

{

SaveFileDialog save = new SaveFileDialog();

save.DefaultExt = "txt";

try

{

if (save.ShowDialog() == DialogResult.OK)

{

File.WriteAllText(save.FileName, richTextBox\_DecoderOut.Text,Encoding.Default);

MessageBox.Show("File was saved succesfully!");

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}