Минобрнауки России

федеральное государственное бюджетное образовательное учреждение высшего образования

«Санкт-Петербургский государственный технологический институт

(технический университет)»

­­­­

**Дисциплина «Разработка ПС»**

Отчёт по лабораторной работе №2.

Знакомство с наследованием и интерфейсами

Разработка программной системы для шифрования и дешифрования текстов

Преподаватели: Корниенко Иван Григорьевич

Федин Алексей Константинович

Исполнил студент 494 группы: Гусев Антон Александрович

Санкт-Петербург

2021

Постановка задачи

Необходимо реализовать два метода шифрования и дешифрования текста. Один из алгоритмов в соответствие с вариантом, второй по выбору. Первый алгоритм шифрования - Атбаш, второй - ROT13.

В программе должны быть предусмотрены два варианта ввода текста: пользователем с клавиатуры или из файла. В работе должна присутствовать возможность сохранения исходных данных и сохранения результата работы программы, а также модульного тестирования.

Исходные данные

В качестве исходных данных программа использует: пользовательский ввод текста в специальное поле для ввода; текстовые файлы формата «txt», в которых хранится текст для шифрования/дешифрования.

Текст, который ввел пользователь, является строковой переменной (тип string в C#). Результат также представляет строковую переменную (тип string в C#).

Особые ситуации

Необходимо рассмотреть следующие особые ситуации:

1. При шифровании и дешифровании необходимо соблюдать регистр введённого текста.
2. Необходимо сохранять в тексте «нешифруемые» элементы (цифры, символы, знаки препинания)
3. Отсутствие ожидаемых программой файлов на чтение или содержание некорректных данных внутри существующих файлов.
4. Запись работы программы в уже существующий файл или создание недопустимого файла, а также проверка атрибутов существующего файла (Атрибут «Только для чтения»).

Математические методы и алгоритмы решения задач

Поставленная задача требует использования некоторых математических методов.

При реализации метода шифрования ROT13 был использован метод взятия остатка при делении (оператор % в c#).

Форматы представления данных

Формат внешних файлов, из которых производится ввод данных – строго текстовый формат txt. Внутри читаемого txt файла для корректной работы должен присутствовать текст в любом виде.

Формат файлов для сохранения результатов работы программы - только текстовый формат txt.

Структура программы

При создании данной программной системы был использован паттерн проектирования MVP (Model View Presenter). В связи с этим было создано три папки: Models, Presenters и Views, которые содержат основные классы и интерфейсы программы. Программа разбита на 7 классов, также отдельно подключен проект для тестирования и 2 интерфейса.

Основная последовательность работы программы – ожидания решения пользователя. Программа ожидает пользовательские нажатия на доступные в тот или иной момент элементы управления. После ввода корректный данных и нажатии кнопки «Запуск» программа запускает главный алгоритм. После чего результат появляется в специальном поле внизу окна программы. Затем пользователь может сохранить исходные данные и результат работы программы или же ввести текст заново без сохранения. Кнопка «Справка» открывает информацию о программе. Комплекс продолжает свою работу до тех пор, пока его не закроет пользователь в правом верхнем углу или с помощью средств операционной системы.

Таблица 1 – Основные переменные программы

|  |  |  |
| --- | --- | --- |
| **Имя** | **Тип** | **Описание** |
| InputText | string | Текст, вводимый пользователем |
| Path | string | Путь к файлу |
| smthGoesWrong | int | Код ошибки, возвращаемый программой |
| ResultText | string | Результат работы программы |

Таблица 2 – Классы, используемые в программе и входящие в состав Models

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя** | **Описание** | **Методы** | **Описание** |
| Atbash | Реализация шифрования методом Атбаш | private string Reverse(string inputText) | Отражения алфавита для шифрования |
| private string EncryptDecrypt(char letter, string symbols, string cipher) | Алгоритм шифрующий одну букву из введенного текста |
| private bool IsLatin(char letter) | Проверка языковой принадлежности буквы |
| public string Encode(string text) | Метод шифрования |
| public string Decode(string text) | Метод дешифрования |
| ROT13 | Реализация шифрования методом ROT13 | private bool IsLatin(char letter) | Проверка языковой принадлежности буквы |
| public string Encode(string text) | Метод шифрования |
| public string Decode(string text) | Метод дешифрования |
| SaveData | Сохранение данных | public void SaveEnteredData(string path, string text) | Сохранение введенного текста |
| public void SaveResults(string path, string text, string result) | Сохранение результатов работы |

Таблица 3 – Классы, используемые в программе и входящие в состав Presenters

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя** | **Описание** | **Методы** | **Описание** |
| CipherPresenter | Реализация связующего звена между Views и Models | public CipherPresenter(ICipherAlgs view) | Конструктор класса |
| public void EncodeAtbash() | Атбаш шифрование |
| public void EncodeRot13() | ROT13 шифрование |
| public void DecodeAtbash() | Атбаш дешифрование |
| public void DecodeRot13() | ROT13 дешифрование |
| public void SaveInput() | Сохранения данных |
| public void SaveResult() | Сохранения результатов |

Таблица 4 – Интерфейсы, используемые в программе

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя** | **Описание** | **Методы / Свойства** | **Описание** |
| ICipher | Методы шифрования для алгоритмов | string Encode(string text); | Шифрование |
| string Decode(string text); | Дешифрование |
| ICipherAlgs | Свойства для Views | string InputText { get; set; } | Введенный текст |
| string ResultText { get; set; } | Результат работы |
| string Path { get; set; } | Путь к файлу |

Блок-схемы алгоритмов программы

Рисунок 1 - Блок-схема алгоритма

Описание хода выполнения лабораторной работы

В ходе лабораторной работы было создано решение (Solution) интегрированной среде разработки Microsoft Visual Studio C# 2019. В нём был создан проект.

Программа была разбита на 3 условные части: модуль работы с данными (Model), модуль взаимодействия с пользователем (Views), модуль связующего звена между пользователем, данными и интерфейсом (Presenter).

После написания основного цикла работы программы, были созданы функции пользовательского ввода, чтения данных из файла. Далее были созданы методы класса работы с текстом. После был создан алгоритм печати полученных данных на экран и сохранения во внешний файл как исходных данных, так и результатов.

В ходе работы над проектом были учтены и обработаны ошибки ввода некорректных данных, некорректное чтение файлов, а также обработаны возникающие исключения.

Также в код программы были включены модульные тесты, предназначенные для тестирования основного модуля программы. Данные для тестирования берутся из заранее подготовленных файлов. В случае удачного прохождения тестов на экран выводится сообщение об успешном их выполнении, иначе - сообщение о неудаче в ходе тестирования.

Результаты работы программы

При первом запуске программы пользователю предлагается выбрать вариант ввода текста: ручной ввод или ввод из файла.

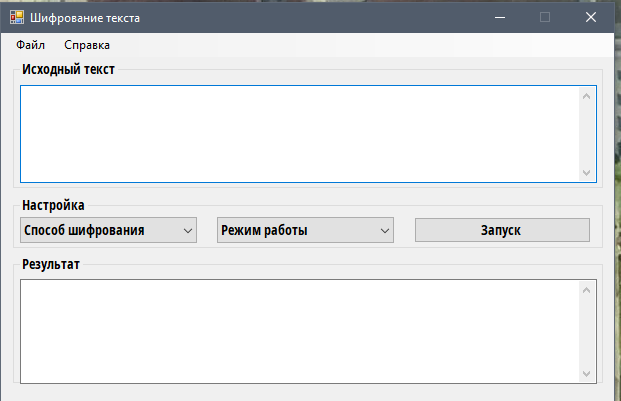


Рисунок 2 – Первый запуск программы

Изаначально перед пользователем появляется текстовое поле для ручного ввода текста, ввести текст из файла можно при помощи варианта из вспомогательного меню «Файл».

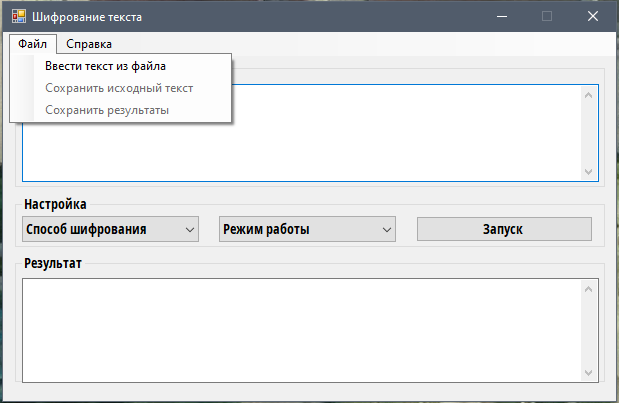


Рисунок 3 – Пример работы программы при вызове меню «Файл»

После ввода текста вручную или из файла пользователю предлагается выбрать способ шифрования (Атбаш или ROT13) и режим работы (Зашифровать или Расшифровать), после чего необходимо нажать кнопку «Запуск» для старта работы программы.

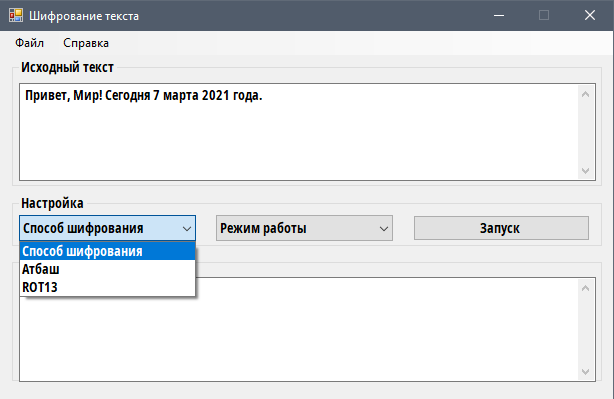


Рисунок 4 – Пример работы программы при выборе способа шифрования

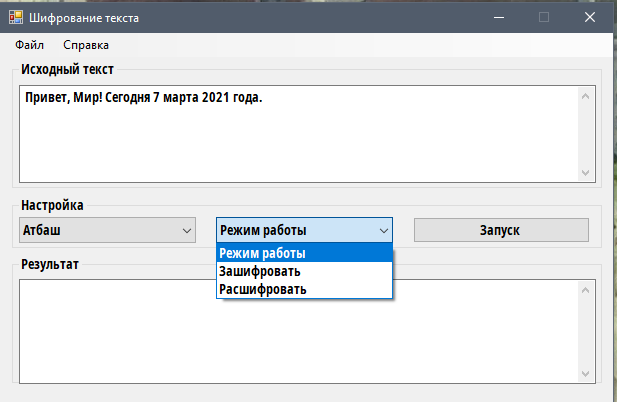
.

Рисунок 5 – Пример работы программе при выборе редима работы

Изначально кнопки меню «Файл» недосутпны для нажатий пользователя ввиду того, что исходные данные и результаты отсутсвтуют.

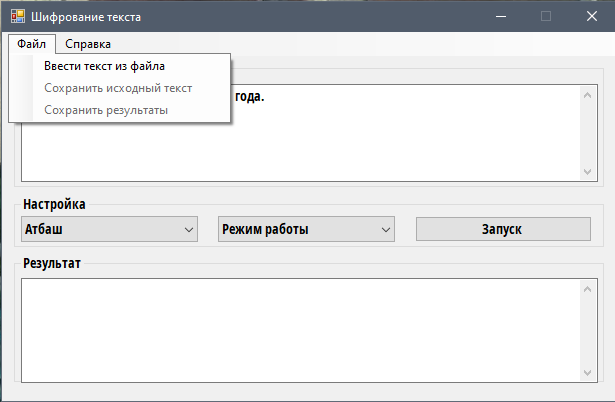


Рисунок 6 – Меню «Файл» до запуска программы

Однако после того, как был введён текст любым из доступных способов и получены результаты, пользователь может сохранить все данные, которые использовались в программе.

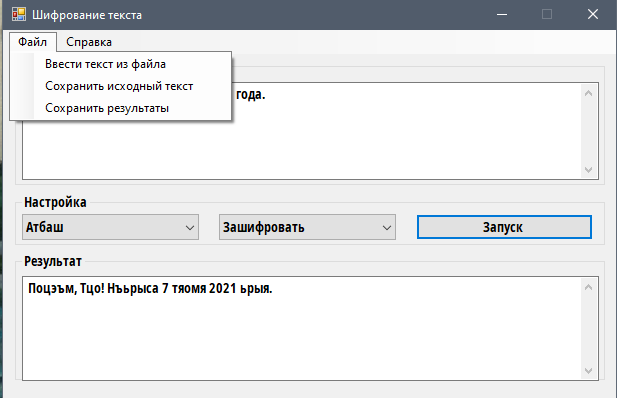


Рисунок 7 – Доступные пункты меню «Файл»

Входные данные, то есть изначально введённый текст, сохраняется так же, как того требует программа при открытии файлов. Таким образом сохраненные данные могут быть использованны повторно в данной программе. Результат сохраняется вместе с введённым текстом, способом шифрования и результатом шифрования и дешифрования.

При нажатии кнопки меню «О проекте» появляется окно с информацией об авторе данной программы и ее задаче.

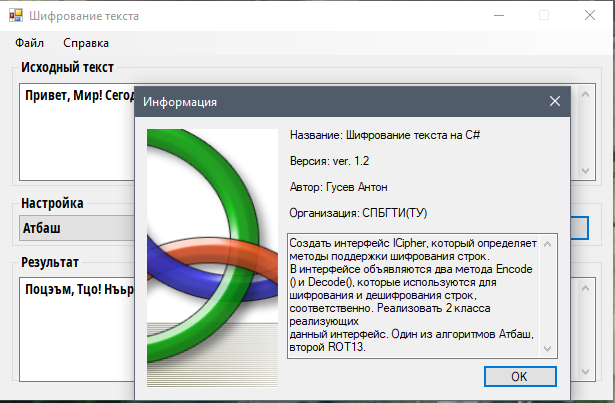


Рисунок 8 – Окно «Информация»

Исходный текст программы

[Начало программы ---]

[Начало Program.cs ---]

using System;

using System.Windows.Forms;

namespace firstLab

{

static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new MainWindow());

}

}

}

[Конец Program.cs --- ]

[Начало MainWindow.cs ---]

using System.Windows.Forms;

using Lab2.Views;

using Lab2.Presenters;

using System;

using System.IO;

using System.Linq;

namespace Lab2

{

public partial class MainWindow : Form, ICipherAlgs

{

public MainWindow()

{

InitializeComponent();

MaximizeBox = false;

saveFileDialog.Filter = @"Text files(\*.txt)|\*.txt"; // фильтр файлов при сохранении

comboBoxCipher.SelectedIndex = 0;

comboBoxMode.SelectedIndex = 0;

}

public string InputText

{

get => textBoxInput.Text;

set => textBoxInput.Text = value;

}

public string ResultText

{

get => textBoxResult.Text;

set => textBoxResult.Text = value;

}

public string Path { get; set; }

public string CipherMethod { get; set; }

private void ButtonStart\_Click\_1(object sender, EventArgs e)

{

if (!string.IsNullOrEmpty(InputText))

{

if (comboBoxCipher.Text == @"Атбаш" && comboBoxMode.Text == @"Зашифровать")

{

var presenter = new CipherPresenter(this);

presenter.EncodeAtbash();

CipherMethod = comboBoxCipher.Text;

}

if (comboBoxCipher.Text == @"Атбаш" && comboBoxMode.Text == @"Расшифровать")

{

var presenter = new CipherPresenter(this);

presenter.DecodeAtbash();

CipherMethod = comboBoxCipher.Text;

}

if (comboBoxCipher.Text == @"ROT13" && comboBoxMode.Text == @"Зашифровать")

{

var presenter = new CipherPresenter(this);

presenter.EncodeRot13();

CipherMethod = comboBoxCipher.Text;

}

if (comboBoxCipher.Text == @"ROT13" && comboBoxMode.Text == @"Расшифровать")

{

var presenter = new CipherPresenter(this);

presenter.DecodeRot13();

CipherMethod = comboBoxCipher.Text;

}

}

else if (string.IsNullOrEmpty(InputText))

{

MessageBox.Show(@"Вы ввели пустую строку. Попробуйте еще раз.",

@"Внимание",

MessageBoxButtons.OK,

MessageBoxIcon.Exclamation);

}

if (!string.IsNullOrEmpty(InputText))

{

SaveInputToolStripMenuItem.Enabled = true;

}

if (!string.IsNullOrEmpty(ResultText))

{

SaveResToolStripMenuItem.Enabled = true;

}

}

private void InputFromFileToolStripMenuItem\_Click(object sender, EventArgs e)

{

if (openFileDialog.ShowDialog() == DialogResult.Cancel)

{

Path = string.Empty;

return; // обработка закрытия окна выбора файла

}

Path = openFileDialog.FileName;

InputText = File.ReadAllText(Path);

Path = string.Empty;

}

private void SaveInputToolStripMenuItem\_Click(object sender, EventArgs e)

{

if (saveFileDialog.ShowDialog() == DialogResult.Cancel)

return;

Path = saveFileDialog.FileName;

var presenter = new CipherPresenter(this);

presenter.SaveInput();

Path = string.Empty;

}

private void SaveResToolStripMenuItem\_Click(object sender, EventArgs e)

{

if (saveFileDialog.ShowDialog() == DialogResult.Cancel)

return;

Path = saveFileDialog.FileName;

var presenter = new CipherPresenter(this);

presenter.SaveResult();

Path = string.Empty;

}

private void InfoToolStripMenuItem\_Click(object sender, EventArgs e)

{

var infoWindow = new About();

if (!Application.OpenForms.OfType<About>().Any())

{

infoWindow.Show();

infoWindow.Focus();

}

}

}

}

[Конец MainWindow.cs --- ]

[Начало Atbash.cs ---]

namespace Lab2.Models

{

public class Atbash : ICipher

{

private const int SmthGoesWrong = -1;

public string Text { get; set; }

private const string AlphabetEng = @"abcdefghijklmnopqrstuvwxyz";

private const string AlphabetRus = @"абвгдеёжзийклмнопрстуфхцчшщъыьэюя";

private string Reverse(string inputText)

{

var reversedText = string.Empty;

foreach (var s in inputText)

{

reversedText = s + reversedText; //добавляем символ в начало строки

}

return reversedText;

}

private string EncryptDecrypt(char letter, string symbols, string cipher)

{

var outputText = string.Empty;

var index = symbols.IndexOf(letter); //поиск позиции символа в строке алфавита

if (index >= 0)

{

outputText += cipher[index].ToString(); //замена символа на шифр

}

return outputText;

}

private bool IsLatin(char letter)

{

if (AlphabetEng.IndexOf(char.ToLower(letter)) == SmthGoesWrong)

return false;

return true;

}

public string Encode(string text)

{

var result = string.Empty;

var lettersText = text.ToCharArray();

for (var i = 0; i < lettersText.Length; i++)

{

if (char.IsLetter(lettersText[i]))

{

var isLowerCase = char.IsLower(lettersText[i]);

lettersText[i] = char.ToLower(lettersText[i]);

if (IsLatin(lettersText[i]))

if (isLowerCase)

{

result += EncryptDecrypt(lettersText[i], AlphabetEng, Reverse(AlphabetEng));

}

else

{

var tempRes = EncryptDecrypt(lettersText[i], AlphabetEng, Reverse(AlphabetEng));

result += tempRes.ToUpper();

}

else

{

if (isLowerCase)

{

result += EncryptDecrypt(lettersText[i], AlphabetRus, Reverse(AlphabetRus));

}

else

{

var tempRes = EncryptDecrypt(lettersText[i], AlphabetRus, Reverse(AlphabetRus));

result += tempRes.ToUpper();

}

}

}

else

{

result += lettersText[i];

}

}

return string.Join("", result);

}

public string Decode(string text)

{

var result = string.Empty;

var lettersText = text.ToCharArray();

for (var i = 0; i < lettersText.Length; i++)

{

if (char.IsLetter(lettersText[i]))

{

var isLowerCase = char.IsLower(lettersText[i]);

lettersText[i] = char.ToLower(lettersText[i]);

if (IsLatin(lettersText[i]))

if (isLowerCase)

{

result += EncryptDecrypt(lettersText[i], Reverse(AlphabetEng), AlphabetEng);

}

else

{

var tempRes = EncryptDecrypt(lettersText[i], Reverse(AlphabetEng), AlphabetEng);

result += tempRes.ToUpper();

}

else

{

if (isLowerCase)

{

result += EncryptDecrypt(lettersText[i], Reverse(AlphabetRus), AlphabetRus);

}

else

{

var tempRes = EncryptDecrypt(lettersText[i], Reverse(AlphabetRus), AlphabetRus);

result += tempRes.ToUpper();

}

}

}

else

{

result += lettersText[i];

}

}

return string.Join("", result);

}

}

}

[Конец Atbash.cs --- ]

[Начало ICipher.cs ---]

namespace Lab2.Models

{

public interface ICipher

{

string Encode(string text);

string Decode(string text);

}

}

[Конец ICipher.cs --- ]

[Начало ROT13.cs ---]

namespace Lab2.Models

{

public class ROT13 : ICipher

{

private const int SmthGoesWrong = -1;

public string Text { get; set; }

private const string AlphabetEng = @"abcdefghijklmnopqrstuvwxyz";

private const string AlphabetRus = @"абвгдеёжзийклмнопрстуфхцчшщъыьэюя";

private bool IsLatin(char letter)

{

if (AlphabetEng.IndexOf(char.ToLower(letter)) == SmthGoesWrong)

return false;

return true;

}

public string Encode(string text)

{

var letters = text.ToCharArray();

for (var i = 0; i < letters.Length; i++)

if (char.IsLetter(letters[i]))

{

var isLowerCase = char.IsLower(letters[i]);

if (IsLatin(letters[i]))

letters[i] = AlphabetEng[(AlphabetEng.IndexOf(char.ToLower(letters[i])) + 13) % AlphabetEng.Length];

else

letters[i] = AlphabetRus[(AlphabetRus.IndexOf(char.ToLower(letters[i])) + 13) % AlphabetRus.Length];

if (!isLowerCase)

letters[i] = char.ToUpper(letters[i]);

}

return string.Join("", letters);

}

public string Decode(string text)

{

var letters = text.ToCharArray();

for (var i = 0; i < letters.Length; i++)

if (char.IsLetter(letters[i]))

{

var isLowerCase = char.IsLower(letters[i]);

if (IsLatin(letters[i]))

letters[i] = AlphabetEng[(AlphabetEng.Length + AlphabetEng.IndexOf(char.ToLower(letters[i])) - 13) % AlphabetEng.Length];

else

letters[i] = AlphabetRus[(AlphabetRus.Length + AlphabetRus.IndexOf(char.ToLower(letters[i])) - 13) % AlphabetRus.Length];

if (!isLowerCase)

letters[i] = char.ToUpper(letters[i]);

}

return string.Join("", letters);

}

}

}

[Конец ROT13.cs --- ]

[Начало SaveData.cs --- ]

using System;

using System.IO;

namespace Lab2.Models

{

public class SaveData

{

public void SaveEnteredData(string path, string text)

{

File.WriteAllText(path, string.Empty);

File.AppendAllText(path, text);

}

public void SaveResults(string path, string text, string result, string method)

{

File.WriteAllText(path, string.Empty);

File.AppendAllText(path, @"Вы ввели следующий текст: " + Environment.NewLine + text + Environment.NewLine + Environment.NewLine);

File.AppendAllText(path, @"Способ шифрования: " + Environment.NewLine + method + Environment.NewLine);

File.AppendAllText(path,

Environment.NewLine + @"Результат шифрования/дешифрования " + Environment.NewLine + result +

Environment.NewLine);

}

}

}

[Конец SaveData.cs --- ]

[Начало CipherPresenter.cs --- ]

using Lab2.Models;

using Lab2.Views;

using System.Windows.Forms;

namespace Lab2.Presenters

{

public class CipherPresenter

{

ICipherAlgs cipherView;

public CipherPresenter(ICipherAlgs view)

{

cipherView = view;

}

public void EncodeAtbash()

{

var atbash = new Atbash();

atbash.Text = cipherView.InputText;

cipherView.ResultText = atbash.Encode(atbash.Text);

}

public void EncodeRot13()

{

var rot13 = new ROT13();

rot13.Text = cipherView.InputText;

cipherView.ResultText = rot13.Encode(rot13.Text);

}

public void DecodeAtbash()

{

var atbash = new Atbash();

atbash.Text = cipherView.InputText;

cipherView.ResultText = atbash.Decode(atbash.Text);

}

public void DecodeRot13()

{

var rot13 = new ROT13();

rot13.Text = cipherView.InputText;

cipherView.ResultText = rot13.Decode(rot13.Text);

}

public void SaveInput()

{

var savingInput = new SaveData();

savingInput.SaveEnteredData(cipherView.Path, cipherView.ResultText);

MessageBox.Show(@"Данные сохранены",

@"Информация");

}

public void SaveResult()

{

var savingRes = new SaveData();

savingRes.SaveResults(cipherView.Path, cipherView.InputText, cipherView.ResultText, cipherView.CipherMethod);

MessageBox.Show(@"Результаты сохранены",

@"Информация");

}

}

}

[Конец CipherPresenter.cs --- ]

[Начало ICipherAlgs.cs --- ]

namespace Lab2.Views

{

public interface ICipherAlgs

{

string InputText { get; set; }

string ResultText { get; set; }

string Path { get; set; }

string CipherMethod { get; set; }

}

}

[Конец ICipherAlgs.cs --- ]

[Начало UnitTest.cs --- ]

using System;

using Lab2.Models;

using Microsoft.VisualStudio.TestTools.UnitTesting;

using Lab2.Presenters;

using Lab2.Views;

namespace UnitTestProject

{

[TestClass]

public class UnitTest

{

[TestMethod]

public void TestMethodAtbashUsualText()

{

string text = "Привет, Мир!";

string expected = "Поцэъм, Тцо!";

var atbash = new Atbash();

Assert.AreEqual(expected, atbash.Encode(text));

}

[TestMethod]

public void TestMethodAtbashNumbers()

{

string text = "Привет123!";

string expected = "Поцэъм123!";

var atbash = new Atbash();

Assert.AreEqual(expected, atbash.Encode(text));

}

[TestMethod]

public void TestMethodAtbashDecodeUsual()

{

string text = "Поцэъм, Тцо!";

string expected = "Привет, Мир!";

var atbash = new Atbash();

Assert.AreEqual(expected, atbash.Decode(text));

}

[TestMethod]

public void TestMethodRot13Usual()

{

string text = "Привет, Мир!123";

string expected = "Ьэхося, Щхэ!123";

var rot13 = new ROT13();

Assert.AreEqual(expected, rot13.Encode(text));

}

[TestMethod]

public void TestMethodRot13Decode()

{

string text = "Ьэхося, Щхэ!123";

string expected = "Привет, Мир!123";

var rot13 = new ROT13();

Assert.AreEqual(expected, rot13.Decode(text));

}

}

}

[Конец UnitTest.cs --- ]

[Конец программы --- ]