**ЛАБОРАТОРНАЯ РАБОТА №2**

**Курс «Технологии разработки программного обеспечения»**

**Тема:** Проектирование и макетирование программного продукта.

**Цель:** Научиться проектировать простейшие системы и составлять документацию по проектированию программного продукта

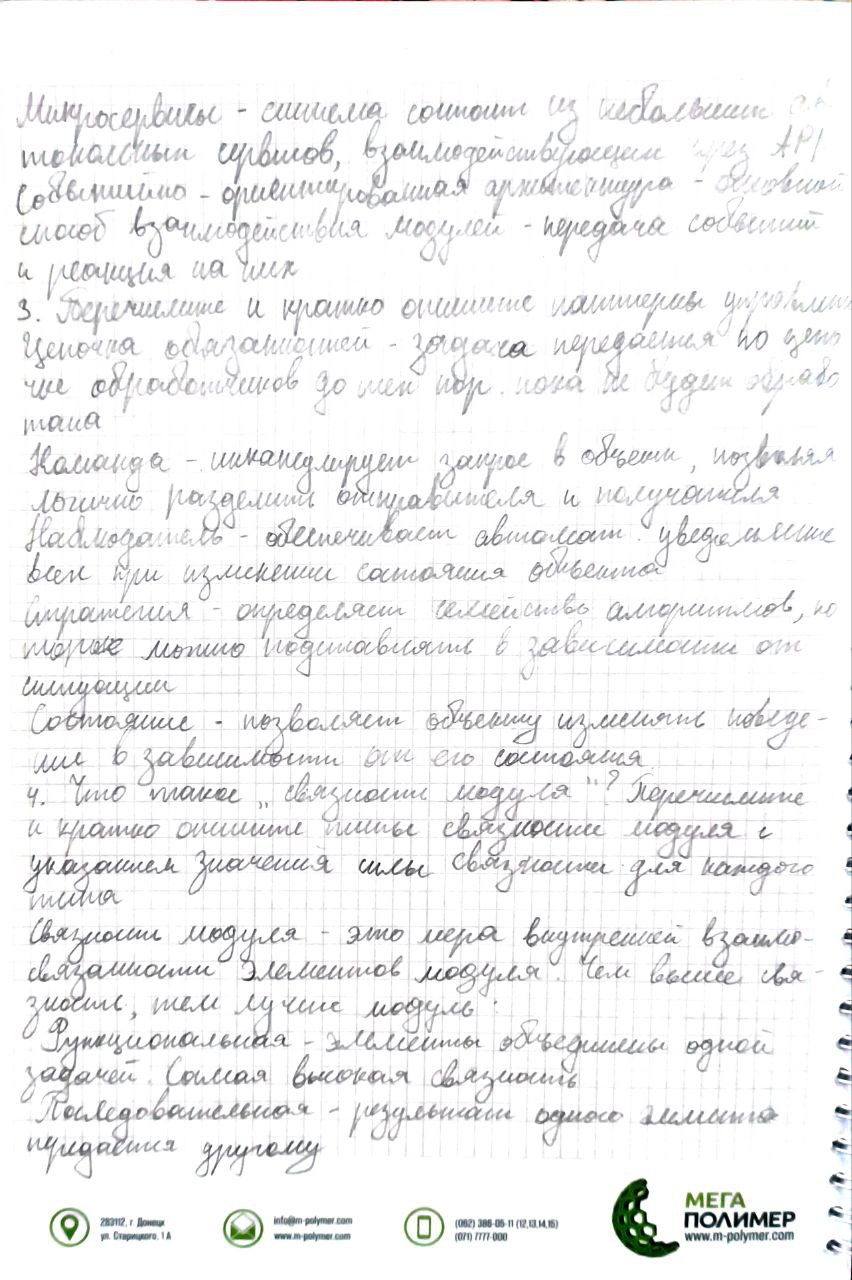
**Задание:**

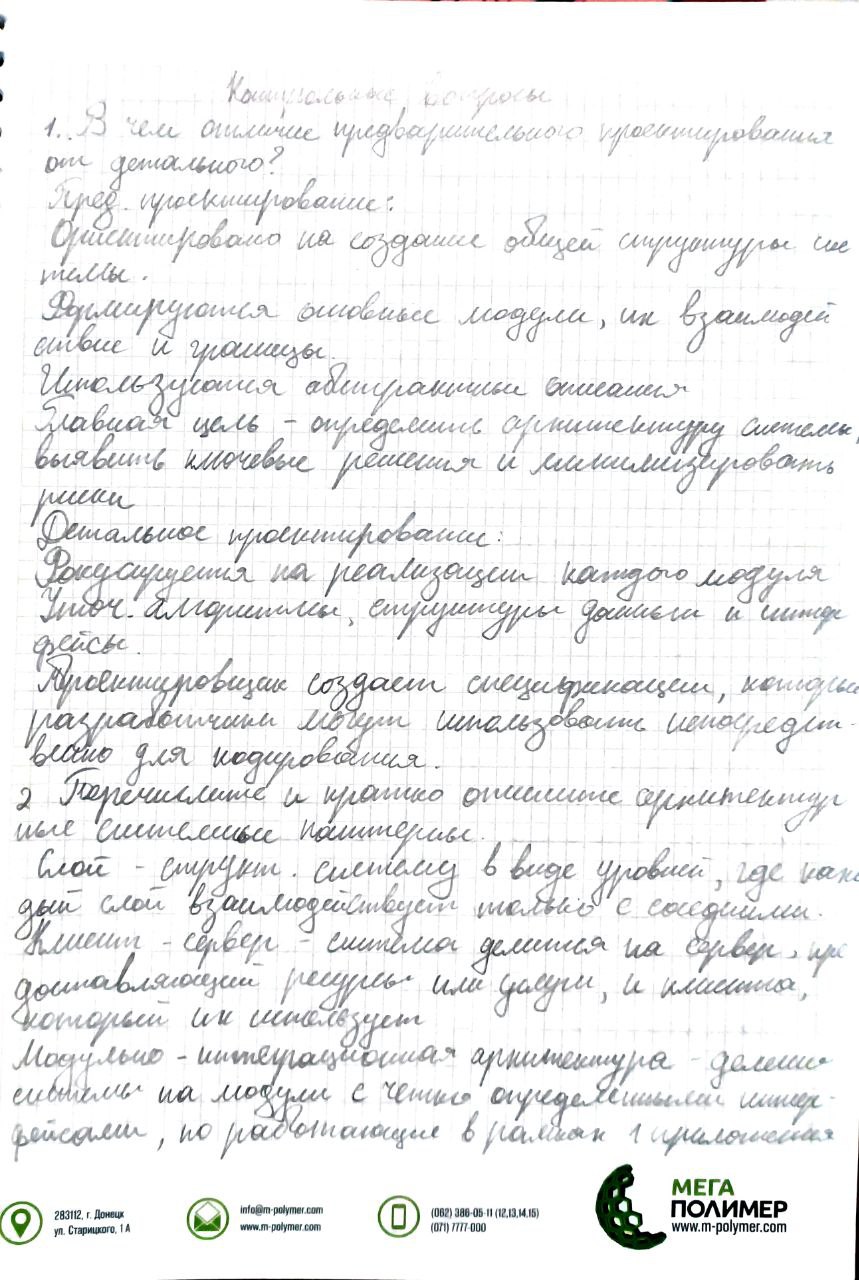
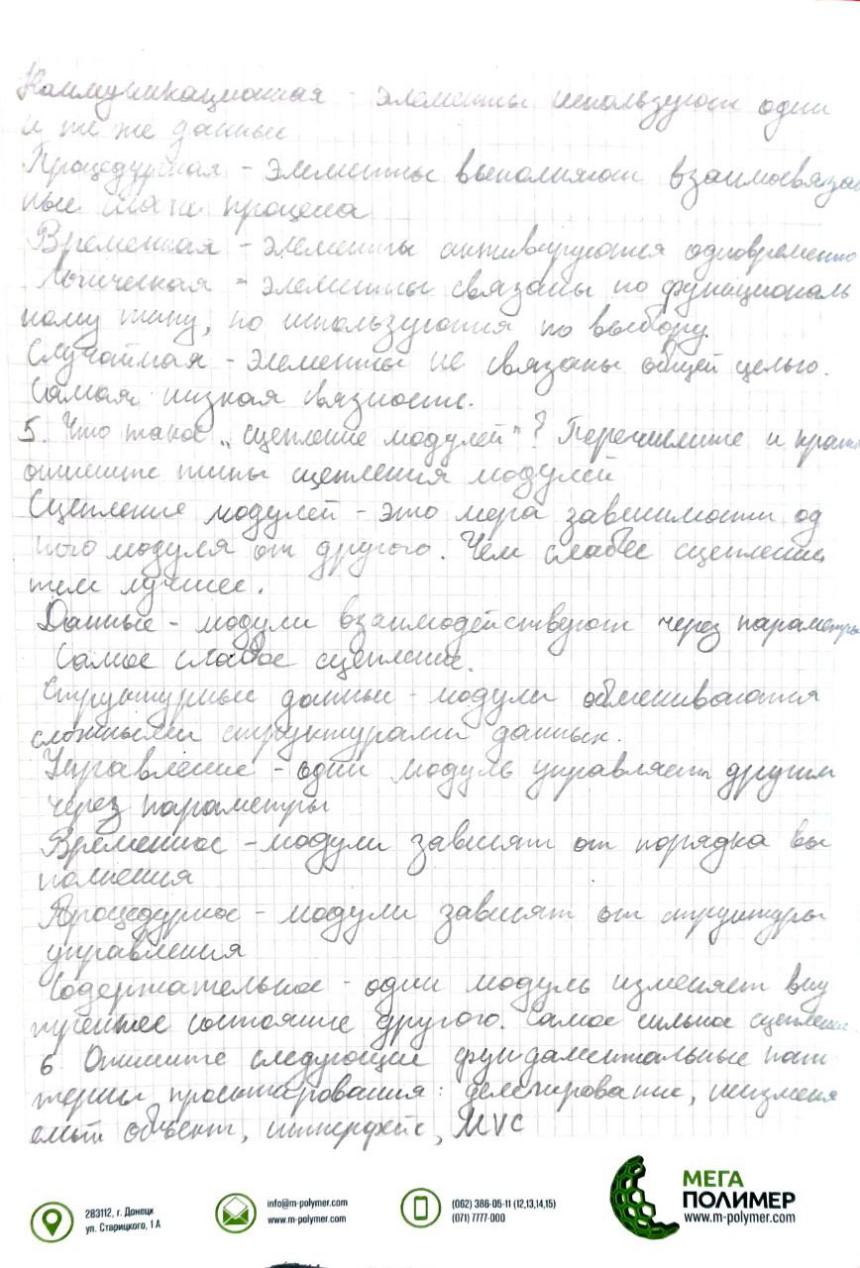
1. Спроектировать общую архитектуру приложения на основе спецификаций требований, составленных в лабораторной работе №1.
2. Разработать макет интерфейса программного продукта для демонстрации заказчику. Проанализировать и учесть его замечания и пожелания.
3. Оформить разделы документа SDD, относящиеся к предварительному проектированию (приложение 2.1).
4. Программно реализовать первую версию спроектированной системы (создать программные модули и закодировать основные классы / структуры и связи между ними).
5. Оформить отчет, включающий ответы на контрольные вопросы, программный код системы и первую версию SDD-документа.

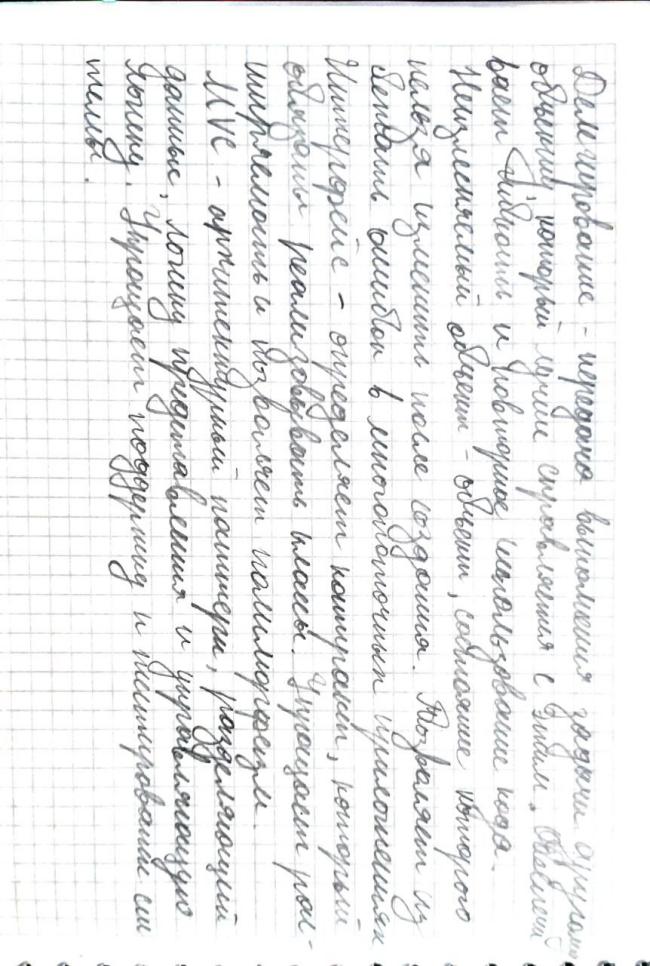
**Контрольные вопросы:**

* 1. В чем отличие предварительного проектирования от детального?
  2. Перечислите и кратко опишите архитектурные системные паттерны.
  3. Перечислите и кратко опишите паттерны управления.
  4. Что такое «связность модуля»? Перечислите и кратко опишите типы связности модуля с указанием значения силы связности для каждого типа.
  5. Что такое «сцепление модулей»? Перечислите и кратко опишите типы сцепления модулей.
  6. Опишите следующие фундаментальные паттерны проектирования: делегирование, неизменяемый объект, интерфейс, MVC.

**Ответы на контрольные вопросы:**







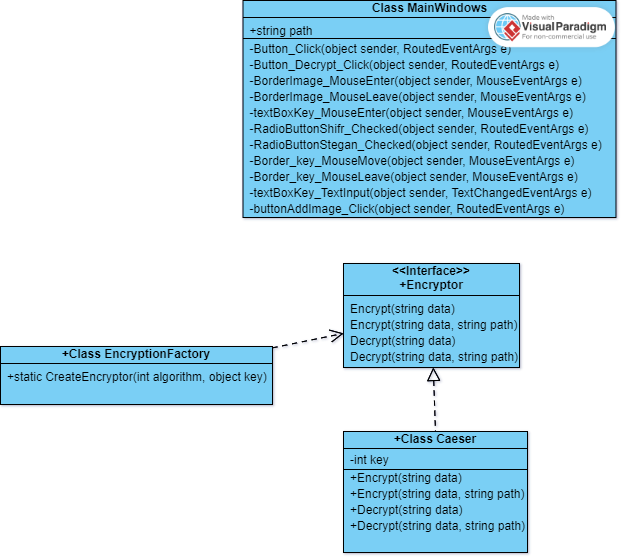
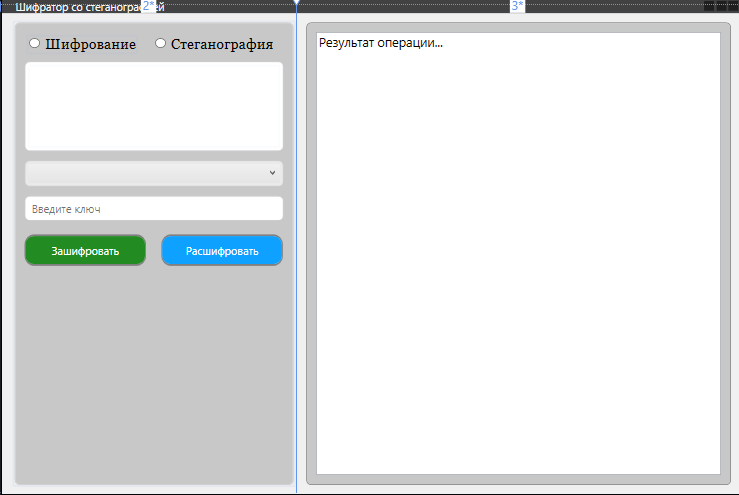
1. Общая архитектура приложения

Рис.1 Архитектура приложения

1. Макет интерфейса программного продукта

Рис.2 Макет интерфейса программного продукта. (При нажатой кнопкой «Шифрования»)

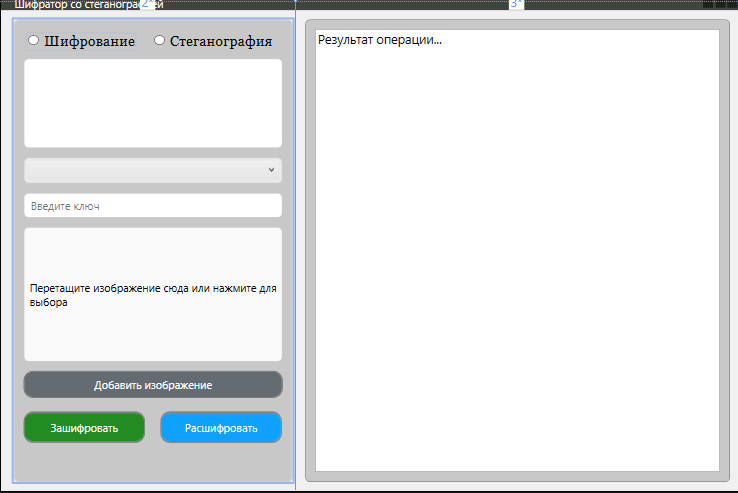


Рис.3 Макет интерфейса программно продукта. (При нажатой кнопкой «Стеганография»)

## Software Design Document (SDD)

**Описание особенности системы шифрования со стеганографией**

**Версия:** 1.0  
**Дата:** 08.12.2024

### ****СОДЕРЖАНИЕ****

1. Примечания
2. Структура проекта
3. Особенности реализации и использования классов
4. Описание Control Flow основных частей проекта
5. Форматы данных
6. **ПРИМЕЧАНИЯ**

* Код оформлен с использованием стандартов C# и WPF.
* Архитектура приложения построена с использованием интерфейса и шаблонной фабрики для упрощения добавления новых алгоритмов.
* Внедрение данных в BMP осуществляется с использованием метода модификации младших битов пикселей.
* Планируется возможность расширения функционала, включая поддержку дополнительных форматов изображений и шифров.

1. **СТРУКТУРА ПРОЕКТА**

**Основные модули:**

**Логика приложения (классы):**

Caesar — шифр Цезаря.

EncryptionFactory — шаблонная фабрика для создания объектов шифрования.

Encryptor — интерфейс, обеспечивающий унифицированный доступ к шифрам.

**GUI-элементы (классы):**

MainWindow — главное окно приложения с текстовыми полями и кнопками для шифрования и стеганографии.

BorderImage — контейнер для отображения загруженного изображения.

ComboBox1 — элемент выбора алгоритма шифрования.

**Служебные файлы приложения:**

App.xaml и MainWindow.xaml — файлы конфигурации интерфейса.

App.xaml.cs и MainWindow.xaml.cs — файлы логики приложения.

1. **ОСОБЕННОСТИ РЕАЛИЗАЦИИ И ИСПОЛЬЗОВАНИЯ КЛАССОВ**

*Encryptor* **(интерфейс)**

Определяет стандартный набор методов:

Encrypt(string data) — шифрование строки.

Decrypt(string data) — дешифрование строки.

Методы Encrypt(string data, string path) и Decrypt(string data, string path) — для работы с BMP-файлами.

*Caesar* **(реализация шифра Цезаря)**

Использует сдвиг символов латинского алфавита на заданное значение key.

Дополнительно поддерживает внедрение зашифрованного текста в BMP.

Примеры формата вызова:

Encrypt("example", "image.bmp") — внедрение текста в изображение.

Decrypt("", "image.bmp") — извлечение текста из изображения.

*EncryptionFactory* **(фабрика)**

Предоставляет методы для создания шифров:

CreateEncryptor(0, key) — шифр Цезаря.

### ****ОПИСАНИЕ CONTROL FLOW ОСНОВНЫХ ЧАСТЕЙ ПРОЕКТ****

#### ****Шифрование текста****

1. Пользователь вводит текст и ключ.
2. Выбирает алгоритм в ComboBox1.
3. Нажимает кнопку "Шифровать" или "Дешифровать".
4. Результат отображается в текстовом поле textBox\_res.

#### ****Работа с BMP-файлами****

1. Пользователь выбирает BMP-файл через кнопку "Добавить изображение".
2. Выполняется операция шифрования или дешифрования.
3. Зашифрованные данные внедряются или извлекаются через младшие биты пикселей.

#### ****Обработка ошибок****

* При выборе неподдерживаемого файла (Encrypt/Decrypt) отображается сообщение об ошибке.
* Некорректный ключ блокируется проверкой в textBoxKey.

1. **ФОРМАТЫ ДАННЫХ**

**BMP-файл**

Внедрение данных: младшие биты пикселей модифицируются для кодирования текста.

Ограничитель: символ ~ добавляется для обозначения конца текста.

Пример структуры данных:

Изображение:

* Заголовок (54 байта) — служебная информация.
* Данные пикселей — содержат модифицируемые биты.

Текст:

* Символы текста преобразуются в байты.
* Каждый бит байта внедряется в младший бит байта пикселя.

**Программный код**

using Microsoft.Win32;

using System;

using System.Collections.Generic;

using System.IO;

using System.Text;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

using System.Windows.Media;

namespace Kurs\_2

{

public interface Encryptor

{

string Encrypt(string data);

string Encrypt(string data, string path);

string Decrypt(string data);

string Decrypt(string data, string path);

}

/// <summary>

/// Шаблонная фабрика

/// </summary>

public class EncryptionFactory

{

public static Encryptor CreateEncryptor(int algorithm, object key)

{

switch (algorithm)

{

case 0:

return new Caesar(Convert.ToInt32(key));

case 1:

return new Atbash((string)key);

default:

throw new ArgumentException("Неизвестный алгоритм шифрования");

}

}

}

/// <summary>

/// Шифр Цезаря

/// </summary>

public class Caesar : Encryptor

{

private int key = 0;

public Caesar(int key) { this.key = key; }

public string Encrypt(string data)

{

char[] alf = { 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z' };

data = data.ToUpper();

string res = "";

foreach(char el in data)

{

if("0123456789 ,.?!@#$%^&\*()\_+=-".IndexOf(el) >= 0)

res += el;

else

res += alf[((el+key-13)%26)];

}

return res.ToLower();

}

public string Decrypt(string data)

{

char[] alf = { 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z' };

data = data.ToUpper();

string res = "";

foreach (char el in data)

{

if ("0123456789 ,.?!@#$%^&\*()\_+=-".IndexOf(el) >= 0)

res += el;

else

res += alf[((el - key - 13) % 26)];

}

return res.ToLower();

}

public string Encrypt(string data, string path)

{

string res = Encrypt(data) + "~"; // Знак ~ является концом строки, которое добавлено в BMP файл, чтобы знать, когда остановиться программе

byte[] bmpBytes = File.ReadAllBytes(path);

if (bmpBytes[0] != 'B' || bmpBytes[1] != 'M') MessageBox.Show("Файл не является BMP изображением.", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

int pixelDataOffset = BitConverter.ToInt32(bmpBytes, 10);

byte[] resBytes = Encoding.UTF8.GetBytes(res);

int resIndex = 0;

int bitIndex = 0;

for (int i = pixelDataOffset; i < bmpBytes.Length; i++)

{

// Внедряем только если данные ещё остались

if (resIndex < resBytes.Length)

{

// Заменяем последний бит текущего байта

bmpBytes[i] = (byte)((bmpBytes[i] & ~1) | ((resBytes[resIndex] >> (7 - bitIndex)) & 1));

bitIndex++;

if (bitIndex == 8) // Переходим к следующему байту данных

{

bitIndex = 0;

resIndex++;

}

}

else break; // Если данные закончились, выходим из цикла

}

File.WriteAllBytes(path, bmpBytes);

return "Успешно!";

}

public string Decrypt(string data, string path)

{

// Чтение файла BMP

string res = "";

byte[] bmpBytes = File.ReadAllBytes(path);

if (bmpBytes[0] != 'B' || bmpBytes[1] != 'M') MessageBox.Show("Файл не является BMP изображением.", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

int pixelDataOffset = BitConverter.ToInt32(bmpBytes, 10);

List<byte> extractedBytes = new List<byte>();

int bitIndex = 0;

byte currentByte = 0;

for (int i = pixelDataOffset; i < bmpBytes.Length; i++)

{

// Извлечение младшего бита текущего байта

int lsb = bmpBytes[i] & 1;

// Добавляем этот бит в текущий собираемый байт

currentByte = (byte)(currentByte \* 2 + lsb);

bitIndex++;

// Если собрали 8 бит, добавляем байт в список

if (bitIndex == 8)

{

extractedBytes.Add(currentByte);

bitIndex = 0;

currentByte = 0;

// Если встретили два нулевых байта подряд, выходим

if (extractedBytes[extractedBytes.Count - 1] == '~')

break;

}

}

if (extractedBytes.Count > 0 && extractedBytes[extractedBytes.Count - 1] == '~')

extractedBytes.RemoveAt(extractedBytes.Count - 1);

return res = Decrypt(Encoding.UTF8.GetString(extractedBytes.ToArray()).TrimEnd('\0'));

}

}

/// <summary>

/// Шифр Атбаша

/// </summary>

public class Atbash : Encryptor

{

private string key = "";

public Atbash(string key) { this.key = key; }

public string Encrypt(string data)

{

return data + "Atbash";

}

public string Encrypt(string data, string path)

{

return "Успешно!";

}

public string Decrypt(string data)

{

return data + "Atbash";

}

public string Decrypt(string data, string path)

{

return data + "Atbash";

}

}

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public string path;

public MainWindow()

{

InitializeComponent();

}

/// <summary>

/// Шифрование текста

/// </summary>

private void Button\_Click(object sender, RoutedEventArgs e)

{

if (Radiobutton\_Shifr.IsChecked != true && RadioButton\_Steg.IsChecked != true) { MessageBox.Show("Выберите действие\nШифрование - Стеганография", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Warning); return; };

if (ComboBox1.SelectedIndex == 0)

{

switch (Radiobutton\_Shifr.IsChecked)

{

case true:

{

Encryptor encry = EncryptionFactory.CreateEncryptor(0, textBoxKey.Text);

textBox\_res.Text = encry.Encrypt(Client\_text.Text);

return;

}

case false:

{

Encryptor encry = EncryptionFactory.CreateEncryptor(0, textBoxKey.Text);

textBox\_res.Text = encry.Encrypt(Client\_text.Text, path);

return;

}

}

}

else if(ComboBox1.SelectedIndex == 1)

{

switch (Radiobutton\_Shifr.IsChecked)

{

case true:

{

Encryptor encry = EncryptionFactory.CreateEncryptor(1, textBoxKey.Text);

textBox\_res.Text = encry.Encrypt(Client\_text.Text);

return;

}

case false:

{

Encryptor encry = EncryptionFactory.CreateEncryptor(1, textBoxKey.Text);

textBox\_res.Text = encry.Encrypt(Client\_text.Text, textBlockImage.Text);

return;

}

}

}

}

/// <summary>

/// Дешифрование текста

/// </summary>

private void Button\_Decrypt\_Click(object sender, RoutedEventArgs e)

{

if (Radiobutton\_Shifr.IsChecked != true && RadioButton\_Steg.IsChecked != true) { MessageBox.Show("Выберите действие\nШифрование - Стеганография", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Warning); return; };

if (ComboBox1.SelectedIndex == 0)

{

switch (Radiobutton\_Shifr.IsChecked)

{

case true:

{

Encryptor encry = EncryptionFactory.CreateEncryptor(0, textBoxKey.Text);

textBox\_res.Text = encry.Decrypt(Client\_text.Text);

return;

}

case false:

{

Encryptor encry = EncryptionFactory.CreateEncryptor(0, textBoxKey.Text);

textBox\_res.Text = encry.Decrypt(Client\_text.Text, path);

return;

}

}

}

else if (ComboBox1.SelectedIndex == 1)

{

Encryptor encryptor = EncryptionFactory.CreateEncryptor(1, textBoxKey.Text);

textBox\_res.Text = encryptor.Decrypt(Client\_text.Text);

}

}

/// <summary>

/// Выключение видимости текста внутри BorderImage

/// </summary>

private void BorderImage\_MouseEnter(object sender, MouseEventArgs e)

{

textBlockImage.Visibility = Visibility.Collapsed;

}

/// <summary>

/// Включение видимости текста внутри BorderImage

/// </summary>

private void BorderImage\_MouseLeave(object sender, MouseEventArgs e)

{

textBlockImage.Visibility = Visibility.Visible;

}

/// <summary>

/// Редактирование при вводе ключа

/// </summary>

private void textBoxKey\_MouseEnter(object sender, MouseEventArgs e)

{

if(textBoxKey.Text == "Введите ключ" || textBoxKey.Text == "")

{

textBoxKey.Text = "";

textBoxKey.Background = new SolidColorBrush(Color.FromRgb(245, 245, 245));

}

else

textBoxKey.Background = new SolidColorBrush(Color.FromRgb(245, 245, 245));

}

/// <summary>

/// Выключение видимости BorderImage для работы с ним

/// </summary>

private void RadioButtonShifr\_Checked(object sender, RoutedEventArgs e)

{

BorderImage.Visibility = Visibility.Collapsed;

buttonAddImage.Visibility = Visibility.Collapsed;

}

/// <summary>

/// Включение видимости BorderImage для работы с ним

/// </summary>

private void RadioButtonStegan\_Checked(object sender, RoutedEventArgs e)

{

BorderImage.Visibility= Visibility.Visible;

buttonAddImage.Visibility = Visibility.Visible;

}

/// <summary>

/// Включение textbox при входе в объект

/// </summary>

private void Border\_key\_MouseMove(object sender, MouseEventArgs e)

{

textBoxKey.IsEnabled = true;

if (textBoxKey.Text == "Введите ключ" || textBoxKey.Text == "")

{

textBoxKey.Text = "";

textBoxKey.Background = new SolidColorBrush(Color.FromRgb(247, 247, 247));

}

}

/// <summary>

/// Отключение textbox при выходе за объект

/// </summary>

private void Border\_key\_MouseLeave(object sender, MouseEventArgs e)

{

if (textBoxKey.Text != "")

{

textBoxKey.IsEnabled = false;

textBoxKey.Background = Brushes.White;

}

else

{

textBoxKey.IsEnabled = false;

textBoxKey.Text = "Введите ключ";

textBoxKey.Background = Brushes.White;

}

}

/// <summary>

/// Редактирование неправильно вводимых ключей, для каждого шифра с своим ключом

/// </summary>

private void textBoxKey\_TextInput(object sender, TextChangedEventArgs e)

{

string text = textBoxKey.Text;

if (ComboBox1.SelectedIndex == 0)

if (int.TryParse(textBoxKey.Text, out \_) == false)

if (textBoxKey.Text.Length != 0 && textBoxKey.Text != "Введите ключ")

{

textBoxKey.Text = text.Remove(textBoxKey.Text.Length - 1, 1);

textBoxKey.CaretIndex = textBoxKey.Text.Length;

}

}

/// <summary>

/// Кнопка добавления фотки

/// </summary>

private void buttonAddImage\_Click(object sender, RoutedEventArgs e)

{

OpenFileDialog ofd = new OpenFileDialog();

ofd.Filter = "All Image files|\*.jpg;\*.bmp;\*.png" ;

if (ofd.ShowDialog() != true) return;

path = ofd.FileName;

}

}

}