**МИНИСТЕРСТВО**

**ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ  
ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ  
 УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 1

по дисциплине: «**Основы защиты информации»**

на тему: «Шифрование и дешифрование на основе классических криптосистем»

Выполнил: студент гр. ИТП-31

Расшивалов Н.И.  
 Принял: доцент

Кравченко О.А.

Гомель 2021

**Цель работы:** освоить основные понятия криптологии, научиться применять простейшие криптосистемы.

**Вариант 19**

**Задание 1**

Цифровая система тайнописи (соответствие цифр и букв выбрать самостоятельно).

**Задание 2**

Квадрат Полибия (для латинского алфавита).

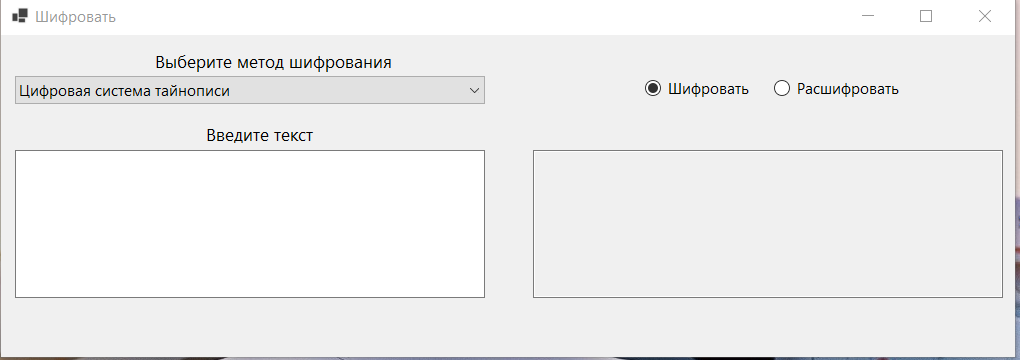


Рисунок 1 – Графический интерфейс программы

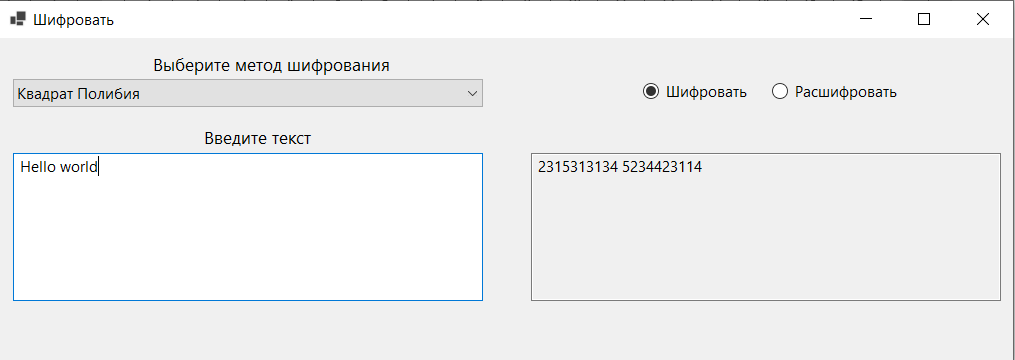


Рисунок 2 – Квадрат Полибия (для латинского алфавита)

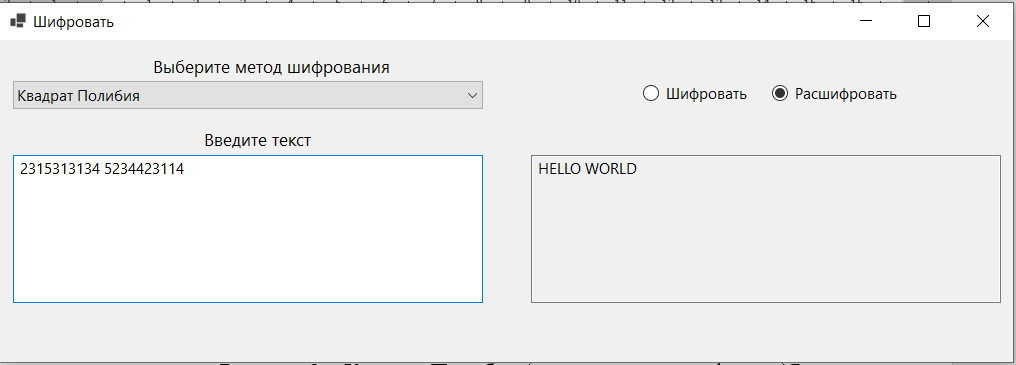


Рисунок 3 – Дешифрование Квадратом Полибия (для латинского алфавита)

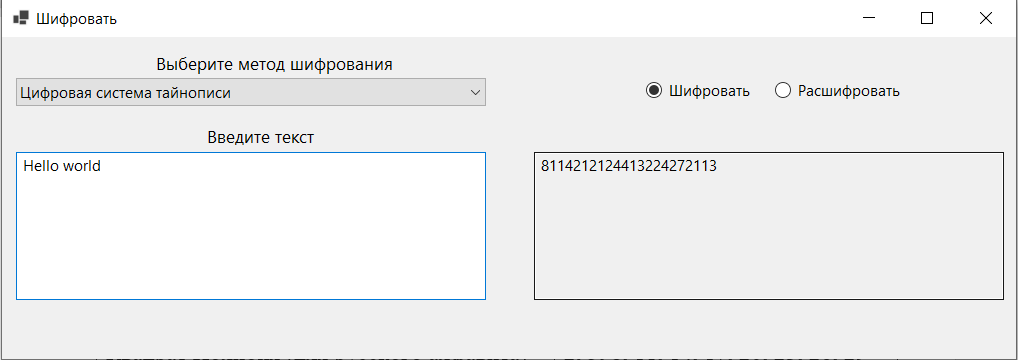


Рисунок 4 – Шифрование методом тайнописи

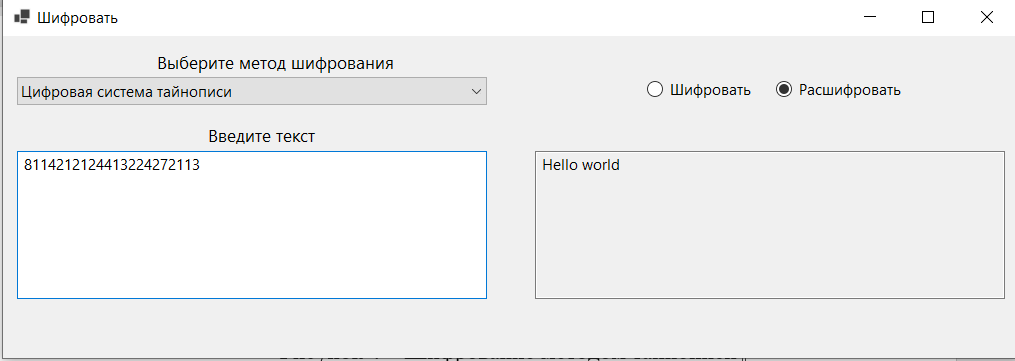


Рисунок 5 – Дешифрование методом тайнописи

Листинг программы представлен в приложении А.

**Вывод:** в ходе выполнения лабораторной работы, были изучены простейшие методы шифрования и дешифрования данных.

**ПРИЛОЖЕНИЕ А**

**Листинг программы**

/// <summary>

   /// Класс реализующий цифровую систему тайнопими

   /// </summary>

   public class **DigitalCipherStrategy** : CipherStrategy

   {

       Dictionary<char, int> tokens;

       Dictionary<int, char> inversedTokens;

       public DigitalCipherStrategy()

       {

           tokens = new Dictionary<char, int>();

           inversedTokens = new Dictionary<int, char>();

           for (int i = 41; i < 100; i++) // Добавляю в словарь символы и их числовое представление

           {

               tokens.Add((char)(i - 9), i);

               inversedTokens.Add(i, (char)(i - 9));

           }

           for(int i = 10; i < 37; i++)

           {

               tokens.Add((char)(i + 87), i);

               inversedTokens.Add(i, (char)(i + 87));

           }

       }

       /// <summary>

       /// Метод дешифрования

       /// </summary>

       /// <param name="textToDecrypt">зашифрованный текст</param>

       /// <returns></returns>

       public string Decrypt(string textToDecrypt)

       {

           StringBuilder builder = new StringBuilder(textToDecrypt.Length);

           for(int i = 0; i < textToDecrypt.Length; i += 2)

           {

               if(i == textToDecrypt.Length - 1)

               {

                   builder.Append(i.ToString());

               }

               try

               {

                   int num = int.Parse(textToDecrypt.Substring(i, 2));

                   builder.Append(inversedTokens[num]);

               }

               catch

               {

                   builder.Append(textToDecrypt[i]);

                   i -= 1;

               }

           }

           return builder.ToString();

       }

       /// <summary>

       /// Метод шифрования

       /// </summary>

       /// <param name="textToEncrypt"> исходный текст</param>

       /// <returns></returns>

       public string Encrypt(string textToEncrypt)

       {

           StringBuilder builder = new StringBuilder(textToEncrypt.Length \* 2);

           foreach (char c in textToEncrypt)

           {

               try

               {

                   builder.Append(tokens[c].ToString());

               }

               catch

               {

                   builder.Append(c);

               }

           }

           return builder.ToString();

       }

   }

//Классс реализующий квадрат Полибия для латинского алфавита

   public class **PolybiusSquareStrategy** : CipherStrategy

   {

       char[,] charMatrix;

       public PolybiusSquareStrategy()

       {

           charMatrix = new char[,]

           {

               {'A', 'B', 'C', 'D', 'E' },

               {'F', 'G', 'H', 'I', 'K' },

               {'L', 'M', 'N', 'O', 'P' },

               {'Q', 'R', 'S', 'T', 'U' },

               {'V', 'W', 'X', 'Y', 'Z' }

           };

       }

       //Метод дешифрования

       public string Decrypt(string textToDecrypt)

       {

           StringBuilder builder = new StringBuilder(textToDecrypt.Length);

           for (int i = 0; i < textToDecrypt.Length; i += 2)

           {

               if (i == textToDecrypt.Length - 1)

               {

                   builder.Append(i.ToString());

               }

               try

               {

                   int row = int.Parse(textToDecrypt.Substring(i, 1));

                   int column = int.Parse(textToDecrypt.Substring(i + 1, 1));

                   builder.Append(charMatrix[row - 1, column - 1]);

               }

               catch

               {

                   builder.Append(textToDecrypt[i]);

                   i -= 1;

               }

           }

           return builder.ToString();

       }

       //Метод шифрования

       public string Encrypt(string textToEncrypt)

       {

           textToEncrypt = textToEncrypt.ToUpper();

           StringBuilder builder = new StringBuilder();

           foreach (char c in textToEncrypt)

           {

               if (c == 'J')

                   builder.Append("24");

               string coords = getSymbolCoordsInStringFormat(c);

               if (coords.Equals(""))

               {

                   builder.Append(c);

               }

               else

               {

                   builder.Append(coords);

               }

           }

           return builder.ToString();

       }

       //Получение координаты символа

       private string getSymbolCoordsInStringFormat(char c)

       {

           for (int i = 0; i < 5; i++)

               for (int j = 0; j < 5; j++)

               {

                   if (c == charMatrix[i, j])

                       return ((i + 1) \* 10 + (j + 1)).ToString();

               }

           return "";

       }

   }