Учреждение образования

«Белорусский государственный технологический университет»

**Кафедра информационных систем и технологий**

**«Отчёт по лабораторной работе 5»**

“ИССЛЕДОВАНИЕ КРИПТОГРАФИЧЕСКИХ ШИФРОВ НА ОСНОВЕ ПЕРЕСТАНОВКИ СИМВОЛОВ”

**Выполнила:** студентка 3 курса

4 группы специальности ПОИТ

Сапегина Екатерина Игоревна,

Вариант 11 (1)

Минск 2023

***Цель:*** изучение и приобретение практических навыков разработки и использования приложений для реализации перестановочных шифров (работа рассчитана на 4 часа аудиторных занятий).

***Задачи:***

1. Закрепить теоретические знания по алгебраическому описанию, алгоритмам реализации операций зашифрования/расшифрования и оценке криптостойкости перестановочных шифров.

2. Ознакомиться с особенностями реализации и свойствами различных перестановочных шифров на основе готового программного средства (L\_LUX).

3. Разработать приложение для реализации указанных преподавателем методов перестановочного зашифрования/расшифрования.

4. Выполнить исследование криптостойкости шифров на основе статистических данных о частотах появления символов в исходном и зашифрованном сообщениях.

5. Оценить скорость зашифрования/расшифрования реализованных способов шифров.

6. Результаты выполнения лабораторной работы оформить в виде описания разработанного приложения, методики выполнения экспериментов с использованием приложения и результатов

эксперимента.

***Практическая часть:***

|  |  |  |
| --- | --- | --- |
| 1 вариант | Белорусский алфавит | 1. Маршрутная перестановка (маршрут: запись – по строкам, считывание – по столбцам таблицы; параметры таблицы – по указанию преподавателя)  2. Множественная перестановка, ключевые слова – собственные имя и фамилия |

1. Разработать авторское приложение в соответствии с целью лабораторной работы. Приложение должно реализовывать следующие операции:

• выполнять зашифрование/расшифрование текстовых документов (объемом не менее 500 знаков), созданных на основе алфавита языка в соответствии с таблицей вариантов задания; при этом следует использовать шифры из третьего столбца данной таблицы (варианты задания в табл. выше);

• формировать гистограммы частот появления символов для исходного и зашифрованного сообщений;

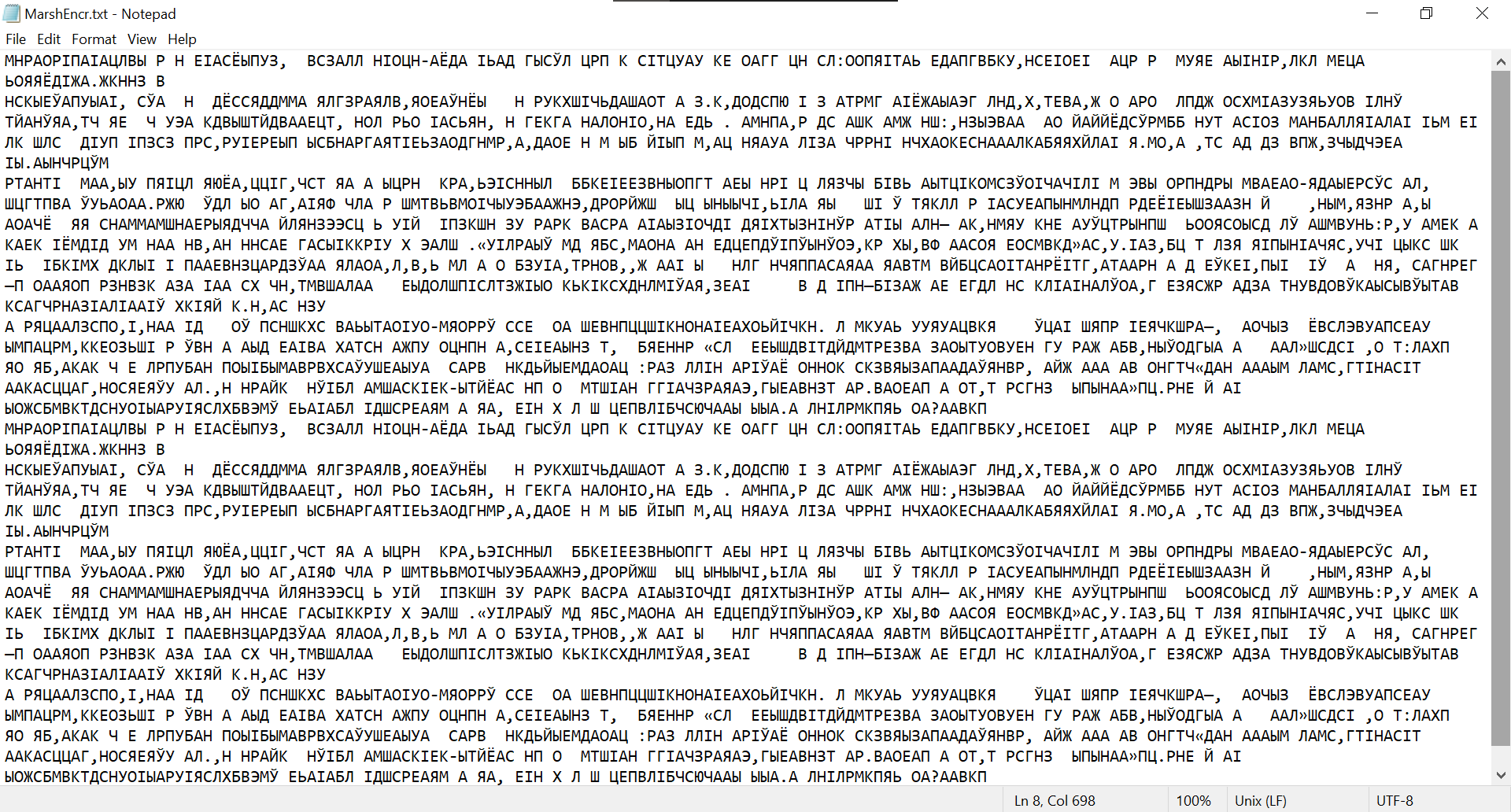
• оценивать время выполнения операций зашифрования/расшифрования (напоминание: во многих языках программирования есть встроенные методы для замеров времени; при отсутствии такового в используемом языке можно воспользоваться разностью двух дат (например, в миллисекундах: время после выполнения программы – время до начала выполнения преобразования))

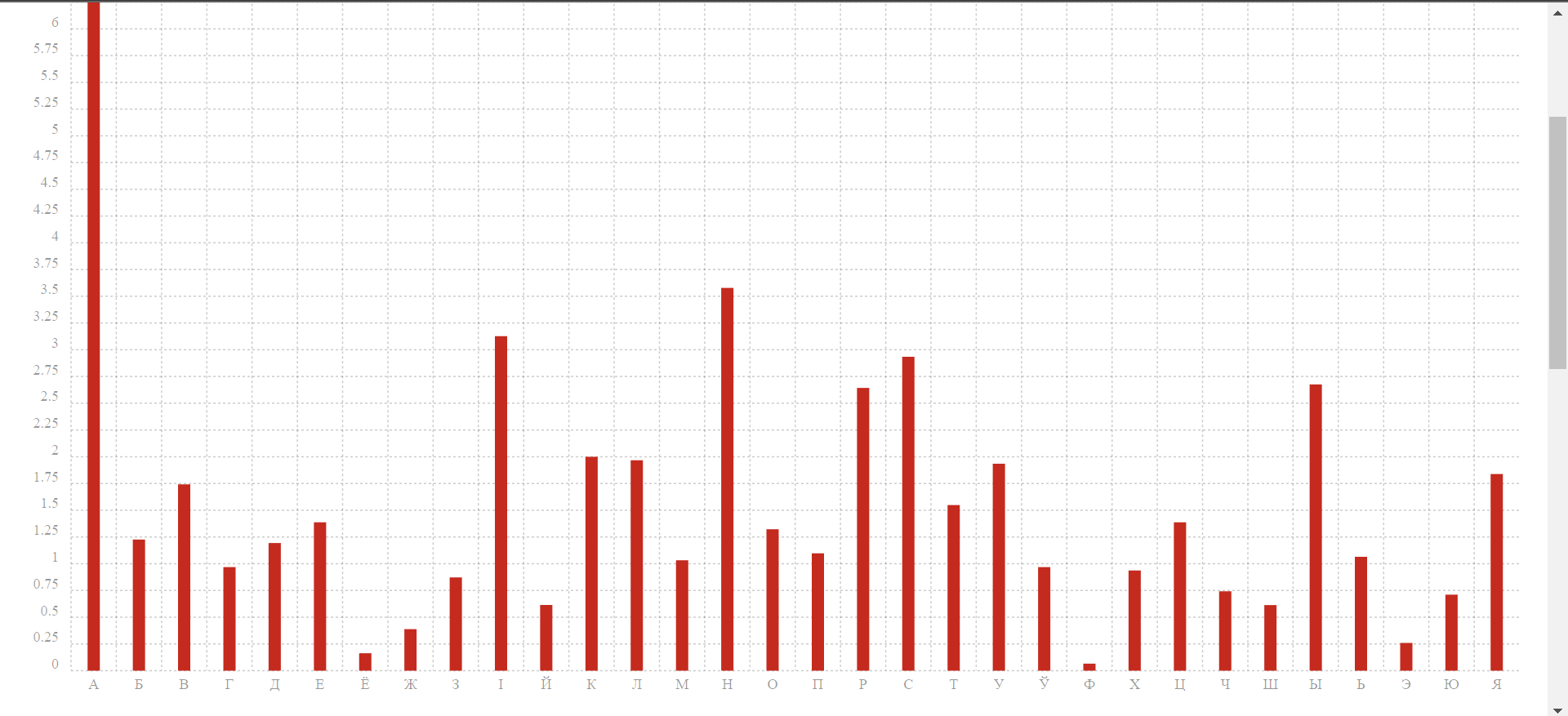
***Код реализации:***

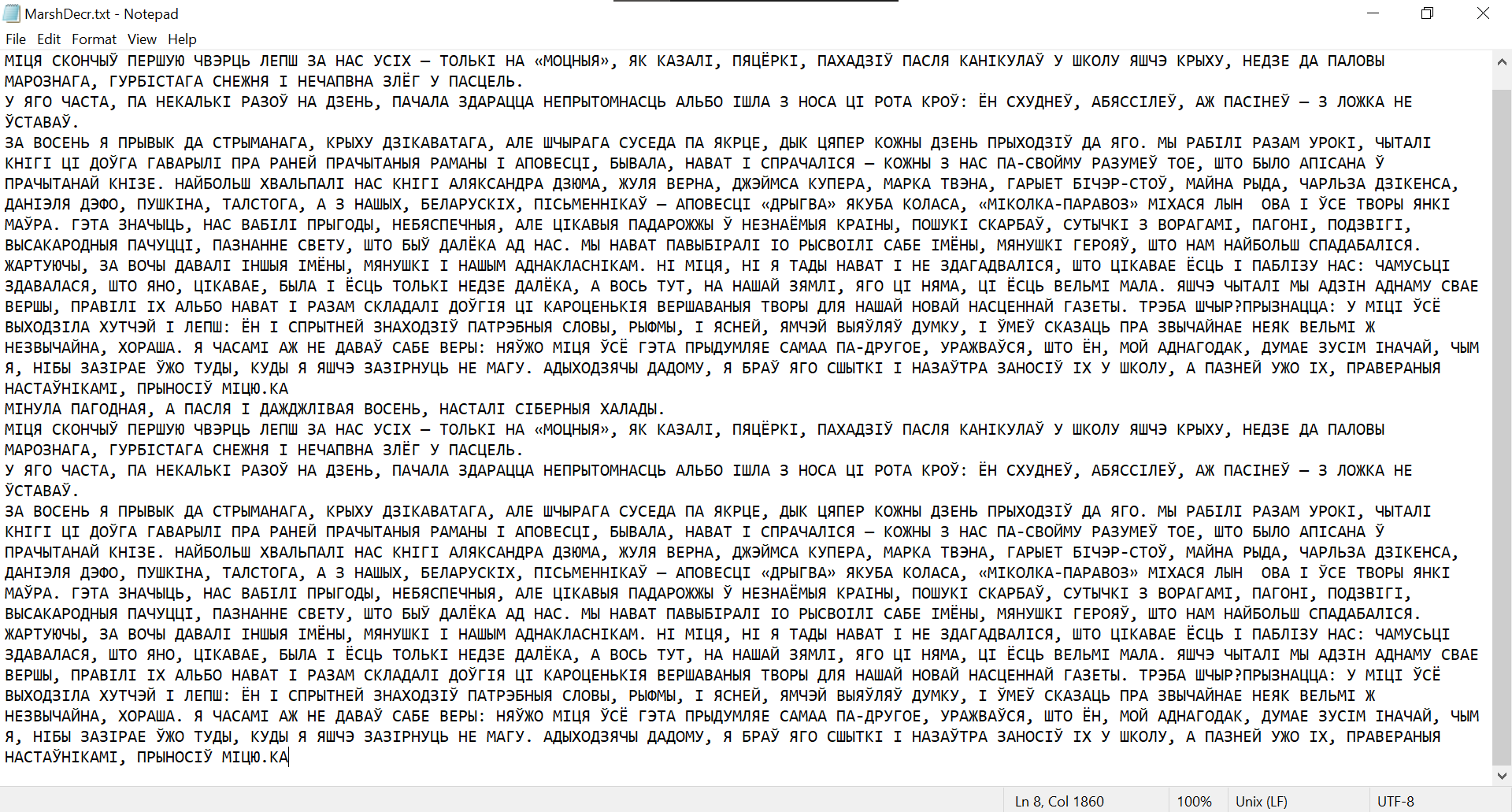
*Маршрутная перестановка****:***

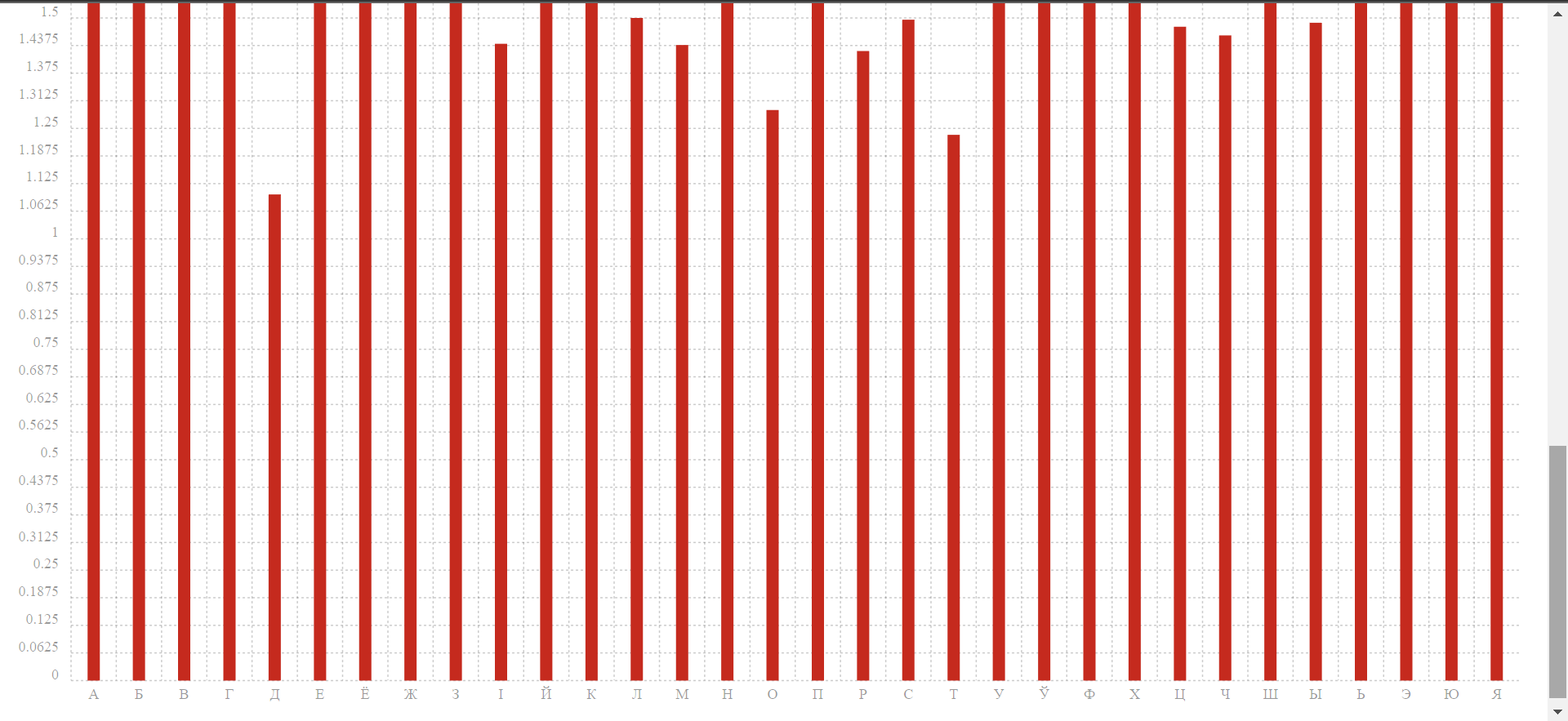
|  |
| --- |
| *const belAl = ['А', 'Б', 'В', 'Г', 'Д', 'Е', 'Ё', 'Ж', 'З', 'І', 'Й', 'К', 'Л', 'М', 'Н', 'О', 'П', 'Р', 'С', 'Т',*  *'У', 'Ў', 'Ф', 'Х', 'Ц', 'Ч', 'Ш', 'Ы', 'Ь', 'Э', 'Ю', 'Я'];*  *let probability = [];*  *belAl.forEach((e,int) => {*  *probability[int] = 0*  *})*    *let getText = (lang) => fetch(lang, {*  *method: "get",*  *cache: "no-cache",*  *mode: "no-cors"*  *}).catch(err => console.log(err))*  *let EncryptMarsh = async (s) => {*  *let response = await getText('text');*  *let text = (await response.text()).split('');*  *let encr = 'MarshEncr --- ';*  *let k = Math.floor((text.length - 1)/s) + 1;*  *while (text.includes('\t'))*  *text.splice(text.indexOf('\t'), 1);*  *for (i = 0; i <= k; i++) {*  *for (j = 0; j <= s; j++) {*  *if (text[k\*j + i] != undefined)*  *{*  *if (belAl.includes(text[k\*j + i])) probability[belAl.indexOf(text[k\*j + i])] += 1;*  *encr += text[k\*j + i]*  *text[k\*j + i] = undefined*  *}*  *}*  *}*  *fetch('tofile', {*  *method: "post",*  *cache: "no-cache",*  *mode: "no-cors",*  *body: encr*  *})*  *.catch(err => console.log(err))*  *}*  *let DecryptMarsh = async (s) => {*  *let response = await getText('textEn');*  *let text = (await response.text()).split('');*  *while (text.includes('\t'))*  *text.splice(text.indexOf('\t'), 1);*  *let encr = 'MarshDecr --- ';*  *let k = Math.floor((text.length - 1)/s) + 1;*  *k += 1;*  *for (j = 0; j <= s; j++) {*  *for (i = 0; i <= k; i++) {*  *if (text[s\*i + j] != undefined)*  *{*  *if (belAl.includes(text[s\*i + j])) probability[belAl.indexOf(text[s\*i + j])] += 1;*  *encr += text[s\*i + j]*  *text[s\*i + j] = undefined*  *}*  *}*  *}*  *fetch('tofile', {*  *method: "post",*  *cache: "no-cache",*  *mode: "no-cors",*  *body: encr*  *})*  *.catch(err => console.log(err))*  *}* |

***Результат:***



******

******

******

*Множественная перестановка:*

|  |
| --- |
| class Program  {  static void Main(string[] args)  {  Console.OutputEncoding = Encoding.UTF8;  string firstKey = "САПЕГІНА";  string secondKey = "КАЦЯРЫНА";  string stringUser = "НАД ШЭРАЙ КУЧАЙ ЗЯМЛI ПАД ПЛОТАМ ПЕРАСТАЛI ЎЗЛЯТАЦЬ УГОРУ ДРОБНЫЯ КАМЯКI I " +  "ПАКАЗАЛАСЯ АБЛЕЗЛАЯ ПIЛОТКА I КIРПАТЫ, АПЕЧАНЫ СОНЦАМ ТВАР";  char[,] matrix = new char[secondKey.Length, firstKey.Length];  int countSymbols = 0;  char[] charsFirstKey = firstKey.ToCharArray();  char[] charsSecondKey = secondKey.ToCharArray();  char[] charStringUser = stringUser.ToCharArray();  List<CharNum> listCharNumFirst =  new List<CharNum>(firstKey.Length);  List<CharNum> listCharNumSecond =  new List<CharNum>(secondKey.Length);  listCharNumFirst = FillListKey(charsFirstKey);  listCharNumSecond = FillListKey(charsSecondKey);  listCharNumFirst = FillingSerialsNumber(listCharNumFirst);  listCharNumSecond = FillingSerialsNumber(listCharNumSecond);  ShowKey(listCharNumFirst, "First key: ");  ShowKey(listCharNumSecond, "Second key: ");  for (int i = 0; i < listCharNumSecond.Count; i++)  {  for (int j = 0; j < listCharNumFirst.Count; j++)  {  matrix[i, j] = charStringUser[countSymbols++];  }  }  ShowMatrix(matrix, "First value: ");  countSymbols = 0;  for (int i = 0; i < listCharNumSecond.Count; i++)  {  for (int j = 0; j < listCharNumFirst.Count; j++)  {  matrix[listCharNumSecond[i].NumberInWord,  listCharNumFirst[j].NumberInWord] = charStringUser[countSymbols++];  }  }  ShowMatrix(matrix, "Encrypted value: ");  Console.ReadKey();  }  #region Methods  public static int GetNumberInThealphabet(char s)  {  string str = @"АБВГДЕЁЖЗІЙКЛМНОПРСТУЎФХЦЧШЫЬЭЮЯ";  int number = str.IndexOf(s);  return number;  }  public static List<CharNum> FillListKey(char[] chars)  {  List<CharNum> listKey = new List<CharNum>(chars.Length);  for (int i = 0; i < chars.Length; i++)  {  CharNum charNum = new CharNum()  {  Ch = chars[i],  NumberInWord = GetNumberInThealphabet(chars[i])  };  listKey.Add(charNum);  }  return listKey;  }  public static void ShowKey(List<CharNum> listCharNum, string message)  {  Console.WriteLine(message);  foreach (var i in listCharNum)  {  Console.Write(i.Ch + " ");  }  Console.WriteLine();  foreach (var i in listCharNum)  {  Console.Write(i.NumberInWord + " ");  }  Console.WriteLine();  Console.WriteLine();  }  public static List<CharNum> FillingSerialsNumber(  List<CharNum> listCharNum)  {  int count = 0;  var result = listCharNum.OrderBy(a =>  a.NumberInWord);  foreach (var i in result)  {  i.NumberInWord = count++;  }  return listCharNum;  }  public static void ShowMatrix(char[,] matrix, string message)  {  Console.WriteLine(message);  for (int i = 0; i < matrix.GetLength(0); i++)  {  for (int j = 0; j < matrix.GetLength(1); j++)  {  Console.Write(matrix[i, j] + " ");  }  Console.WriteLine();  }  Console.WriteLine();  Console.WriteLine();  }  #endregion Methods |

***Результат:***

