МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Отчет по лабораторной работе №2

Исследование криптологических шифров на основе

подстановки (замены) символов

Выполнил студент: Плюто Э. В.

ФИТ 3 курса, 5 группа

Проверил: Савельева М. Г.

Минск 2024

**Практическое задание:**

**1).** Разработать авторское приложение, которое должно выполнять зашифрование/расшифрование текстовых документов, созданных на основе белорусского алфавита (шифр Цезаря с ключевым словом, ключевое слово – «інфарматыка», *а* = 2).

Для получения алфавита подстановки необходимо вначале вставить ключевое слово в указанную в задании позицию, а затем вставить символы в алфавитном порядке (без повторения символов как в ключевом слове, так и в алфавите). Для реализации получения алфавита подстановки из исходного алфавита (белорусский), ключевого слова («інфарматыка») и позиции (2) была написана функция, представленная ниже.

|  |
| --- |
| public static string GetAlphabetCeasarWithKeyWord(string alphabet, string keyWord, int position)  {  position--;  char[] encryptAlphabet = new char[alphabet.Length];  bool skip = false;  int symbol = 0;  // Вставляем ключевое слово  int insertPosition = position;  for (int i = position; i < keyWord.Length + position; i++)  {  for (int j = position; j < i; j++)  {  if (keyWord[i - position] == keyWord[j - position])  {  skip = true;  break;  }  }  if (!skip)  {  encryptAlphabet[insertPosition] = keyWord[i - position];  insertPosition++;  }  else  skip = false;  }  // Вставляем символы после ключевого слова  for (int i = 0; i < alphabet.Length; i++)  {  for (int j = position; j < keyWord.Length + position; j++)  {  if (alphabet[i] == encryptAlphabet[j])  {  skip = true;  break;  }  }  if (!skip)  {  if (insertPosition == encryptAlphabet.Length)  {  symbol = i;  break;  }  encryptAlphabet[insertPosition] = alphabet[i];  insertPosition++;  }  else  skip = false;  }  // Вставляем символы до ключевого слова  insertPosition = 0;  for (int i = symbol; i < alphabet.Length; i++)  {  for (int j = position; j < keyWord.Length + position; j++)  {  if (alphabet[i] == encryptAlphabet[j])  {  skip = true;  break;  }  }  if (!skip)  {  if (insertPosition == position)  break;  encryptAlphabet[insertPosition] = alphabet[i];  insertPosition++;  }  else  skip = false;  }  return String.Concat<char>(encryptAlphabet);  } |

Листинг 1 – Реализация функции получения алфавита подстановки

Результат выполнения данной функции представлен ниже.



Рисунок 1 – Результат получения алфавита подстановки

Для шифрования и записи файла была написана функция, представленная на листинге 2.

|  |
| --- |
| public static void EncryptFile(string text, string alphabet)  {  string encryptedText = "";  foreach (char symbol in text)  {  if (belarus.Contains(symbol))  encryptedText += alphabet[belarus.IndexOf(symbol)];  else  encryptedText += symbol;  }  File.WriteAllText(@"C:\Users\Erik\Desktop\3course\IB\LABA4\EncryptedCaesar.txt", encryptedText);  } |

Листинг 2 – Реализация функции шифрования и записи файла

Результат шифрования файла представлен на рисунке 2.

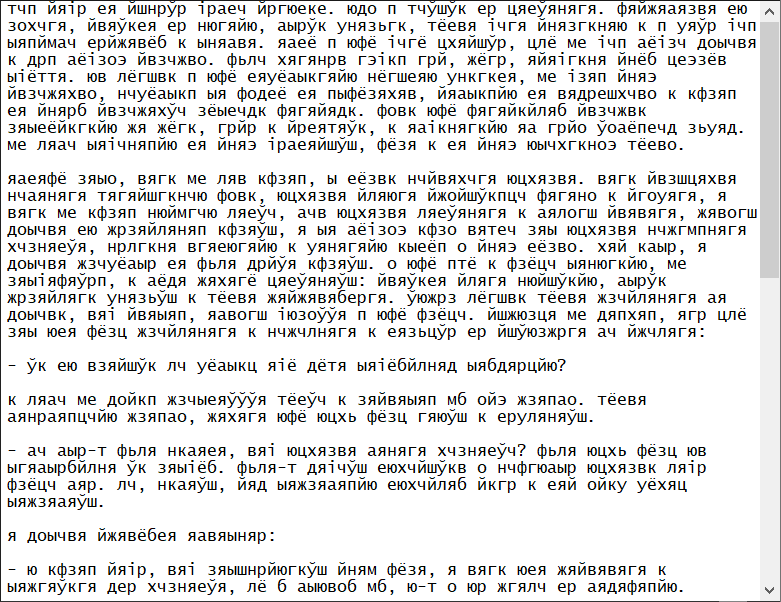


Рисунок 2 – Зашифрованный файл

Для расшифрования была написана функция, аналогичная функции выше, представленная на листинге 3.

|  |
| --- |
| public static void DecryptFile(string path, string alphabet)  {  string text = GetStringFromFile(path);  string decryptedText = "";  foreach (char symbol in text)  {  if (alphabet.Contains(symbol))  decryptedText += belarus[alphabet.IndexOf(symbol)];  else  decryptedText += symbol;  }  File.WriteAllText(@"C:\Users\Erik\Desktop\3course\IB\LABA4\DecryptedCaesar.txt", decryptedText);  } |

Листинг 3 – Реализация функции расшифрования и записи файла

Время выполнения шифрования и расшифрования файла представлено на рисунке 3.

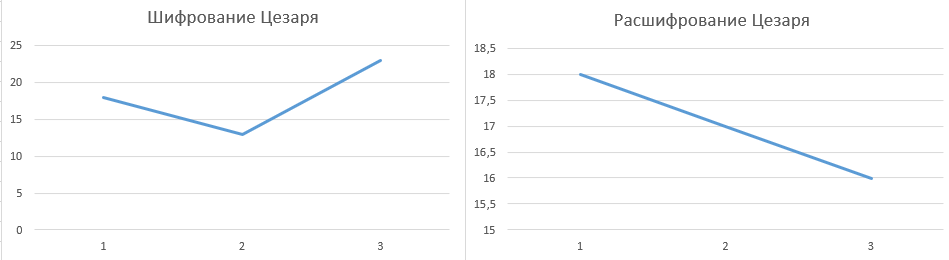


Рисунок 3 – Время выполнения операций шифрования и расшифрования

Гистограмма частоты появления символов исходного файла представлена ниже.

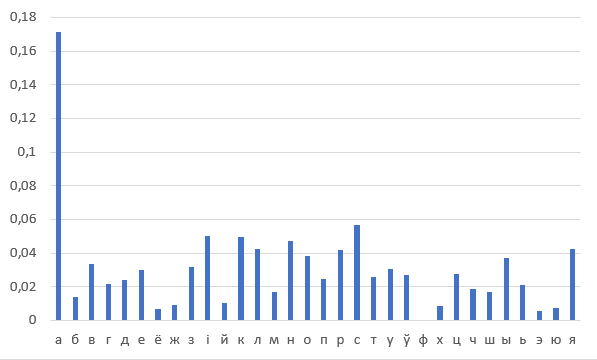


Рисунок 4 – Частота появления символов исходного файла

Гистограмма частоты появления символов зашифрованного файла представлена на рисунке 5.

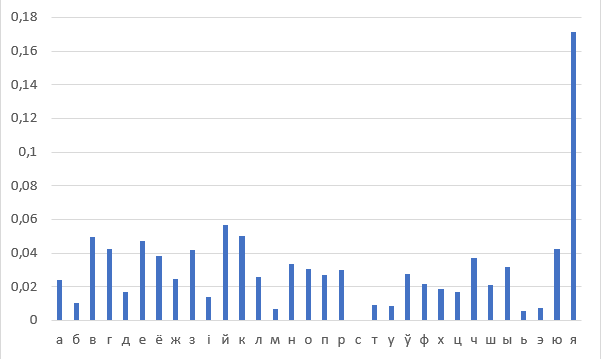


Рисунок 5 – Частота появления символов зашифрованного файла

**2).** Разработать авторское приложение, которое должно выполнять зашифрование/расшифрование текстовых документов, созданных на основе белорусского алфавита (таблица Трисемуса, ключевое слово – собственное имя).

Для шифрования файла с помощью таблицы Трисемуса была написана функция, которая составляет данную таблицу. В качестве аргументов данная функция принимает алфавит, составленный с помощью функции, описанной в задании выше, и количество столбцов.

|  |
| --- |
| public static char[,] CreateTable(string alphabet, int n)  {  char[,] table = new char[alphabet.Length/n, n];  int position = 0;  for (int i = 0; i < alphabet.Length/n; i++)  {  for (int j = 0; j < n; j++)  {  table[i,j] = alphabet[position];  position++;  }  }  return table;  } |

Листинг 4 – Реализация функции создания таблицы Трисемуса

Результат выполнения данной функции представлен ниже.

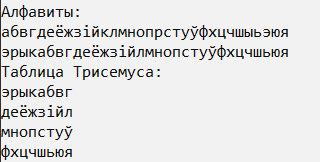


Рисунок 6 – Таблица Трисемуса

Для шифрования файла была написана функция, представленная на листинге 5.

|  |
| --- |
| public static void EncryptFileTrysemus(string text, char[,] table)  {  string encryptedText = "";  foreach (char symbol in text)  {  if (belarus.Contains(symbol))  {  for (int i = 0; i < table.GetUpperBound(0) + 1; i++)  {  for (int j = 0; j < table.Length / (table.GetUpperBound(0) + 1); j++)  {  if (table[i,j] == symbol)  {  if (i == table.GetUpperBound(0))  encryptedText += table[0, j];  else  encryptedText += table[i + 1,j];  }  }  }  }  else  encryptedText += symbol;  }  File.WriteAllText(@"C:\Users\Erik\Desktop\3course\IB\LABA4\EncryptedTrysemus.txt", encryptedText);  } |

Листинг 5 – Реализация функции шифрования и записи файла

Результат выполнения данной функции представлен ниже.

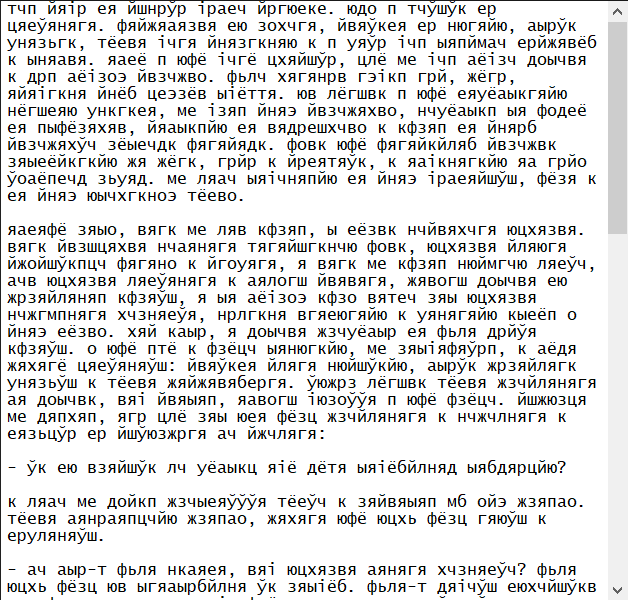


Рисунок 7 – Результат шифрования

Функция расшифрования файла была выполнена аналогичным способом. Время выполнения шифрования и расшифрования представлено ниже.

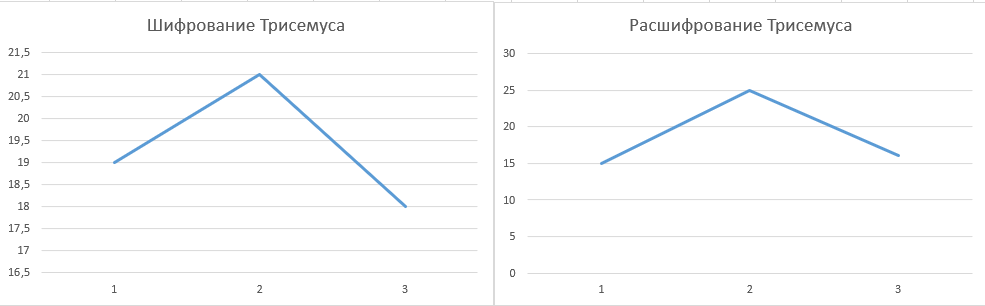


Рисунок 8 – Время выполнения шифрования и расшифрования

Гистограмма частоты появления символов зашифрованного файла представлена на рисунке 9.

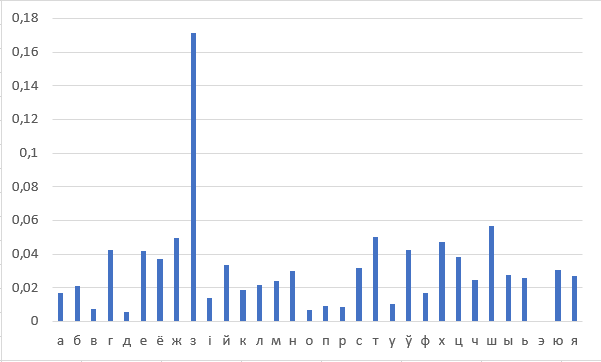


Рисунок 9 – Частота появления символов зашифрованного файла

**Вывод:**

Были приобретены практические навыки разработки и использования приложений для реализации подстановочных шифров. Были реализованы программы для выполнения шифра цезаря с подстановкой ключевого слова и таблицы Трисемуса, построены гистограммы частот появления символов исходного и зашифрованных файлов.