МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4 НА ТЕМУ:**

**ИССЛЕДОВАНИЕ КРИПТОГРАФИЧЕСКИХ ШИФРОВ НА ОСНОВЕ ПОДСТАНОВКИ (ЗАМЕНЫ) СИМВОЛОВ**

                                                                 Выполнил студент 3 курса 5 группы

Дмитрук Илья Игоревич

Минск 2024

# Задание 1.

Для шифрования сообщения использовалась аффинная система подстановок Цезаря. Её суть состоит в том, что мы получаем новый номер символа в алфавите и заменяем текущий символ на новый. Новый номер находится по формуле:

где *a* и *b* – целые числа.

Для реализации данного шифрования была разработана функция EncryptCaesar. Её код представлен в листинге 1.1.

public static string EncryptCaesar(string text, int a, int b)

{

StringBuilder encryptText = new StringBuilder();

int n = alphabet.Length;

int x;

foreach (char c in text)

{

x = Array.IndexOf(alphabet, char.ToUpper(c));

if (x != -1)

{

if (char.IsUpper(c)) encryptText.Append(alphabet[(a \* x + b) % n]);

else encryptText.Append(char.ToLower(alphabet[((a \* x) + b) % n]));

}

else encryptText.Append(c);

}

return encryptText.ToString();

}

Листинг 1.1 – Функции для шифрования аффинной системой подстановки Цезаря

В качестве сообщения для шифрования использовался текст на польском языке. Текст, после шифрования, представлен на рисунке 1.1

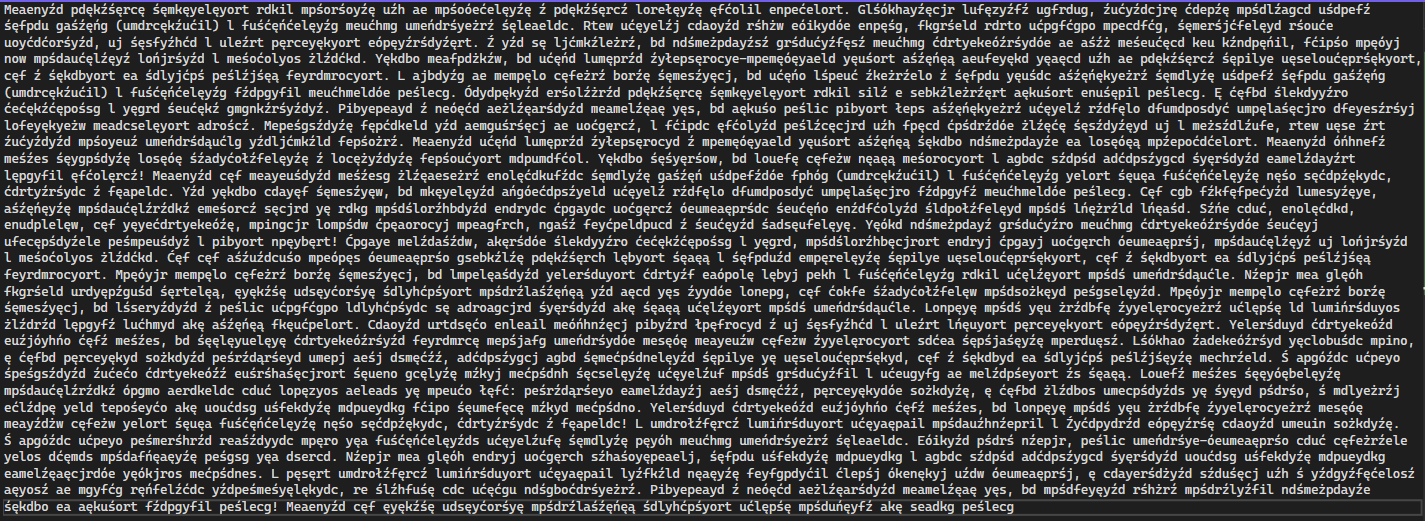


Рисунок 1.1 – Зашифрованный аффинной системой подстановки Цезаря текст

Для расшифрования сообщения, необходимо было найти первоначальный номер символа в алфавите. Для этого используется формула:

Для реализации расшифрования была разработана функция DecryptCaesar. Её код представлен в листинге 1.2.

public static string DecryptCaesar(string text, int a, int b)

{

StringBuilder decryptText = new StringBuilder();

int n = alphabet.Length;

int y;

int a\_inv = ModInverse(a, n);

foreach (char c in text)

{

y = Array.IndexOf(alphabet, char.ToUpper(c));

if (y != -1)

{

if (char.IsUpper(c)) decryptText.Append(alphabet[(a\_inv \* (y - b + n)) % n]);

else decryptText.Append(char.ToLower(alphabet[(a\_inv \* (y - b + n)) % n]));

}

else decryptText.Append(c);

}

return decryptText.ToString();

}

Листинг 1.2 – Функции для расшифрования аффинной системы подстановки Цезаря

В результате выполнения расшифрования, мы получили исходный текст. Он представлен на рисунке 1.2.

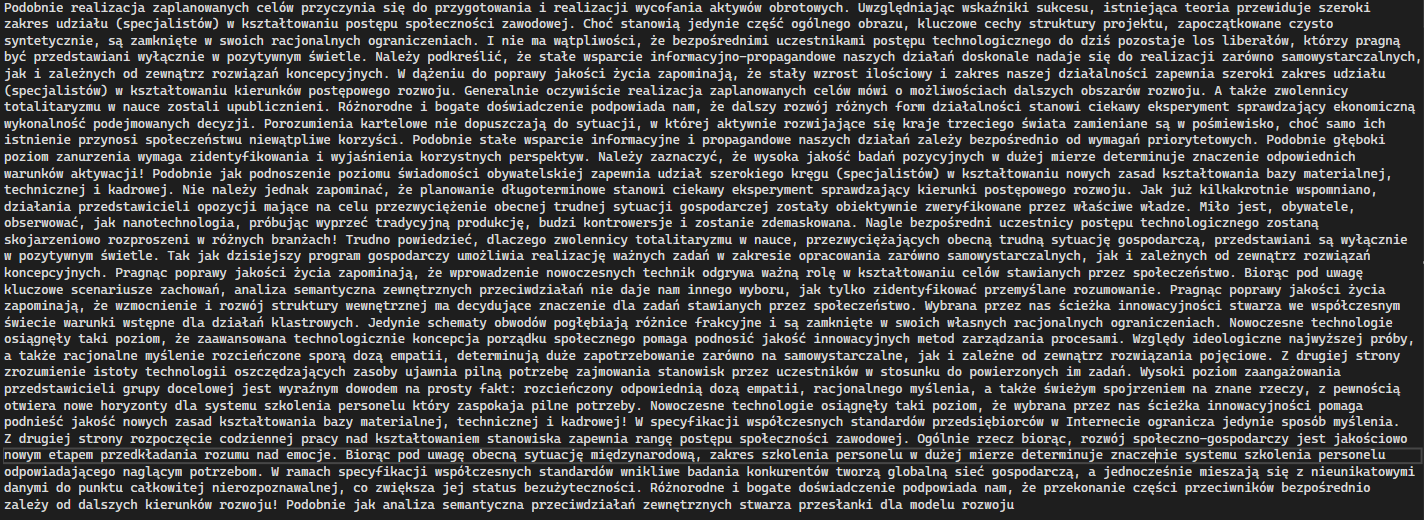


Рисунок 1.2 – Расшифрованный текст, зашифрованный аффинной системой подстановки Цезаря

# Задание 2.

Для шифрования сообщения использовался шифр Порты. Её суть состоит в том, что мы создаём таблицу, столбцы и строки которой соответствуют буквам в алфавите. Элементы в этой таблице заполняются числами в порядке возрастания. Затем в сообщении берём по два символа, первый в паре символ обозначает соответствующую строку в таблице, второй символ, соответствующий столбец. Затем число, которое находиться в соответствующем столбце и строке, записываются вместо пары символов. Если количество символов в сообщении нечётное, то в конце сообщения добавляется символ «*А*». Для реализации данного шифрования была разработана функция EncryptPorta. Её код представлен в листинге 2.1.

public static string EncryptPorta(string text)

{

List<int> encryptText = new List<int>();

int[,]tablePorta = new int[alphabet.Length, alphabet.Length];

text = Regex.Replace(text, @"[\p{P}\s]", "");

text = text.ToUpper();

int n = 1;

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

{

for (int j = 0; j < alphabet.Length; j++)

{

tablePorta[i, j] = n++;

}

}

if (text.Length % 2 != 0) text += 'a';

int x = 0, y = 0;

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

{

x = Array.IndexOf(alphabet, text[i]);

y = Array.IndexOf(alphabet, text[i + 1]);

encryptText.Add(tablePorta[x, y]);

}

return string.Join(" ", encryptText);

}

Листинг 2.1 – Функции для шифрования шифром Порты

В качестве сообщения для шифрования использовался текст на польском языке. Текст, после шифрования, представлен на рисунке 2.1.

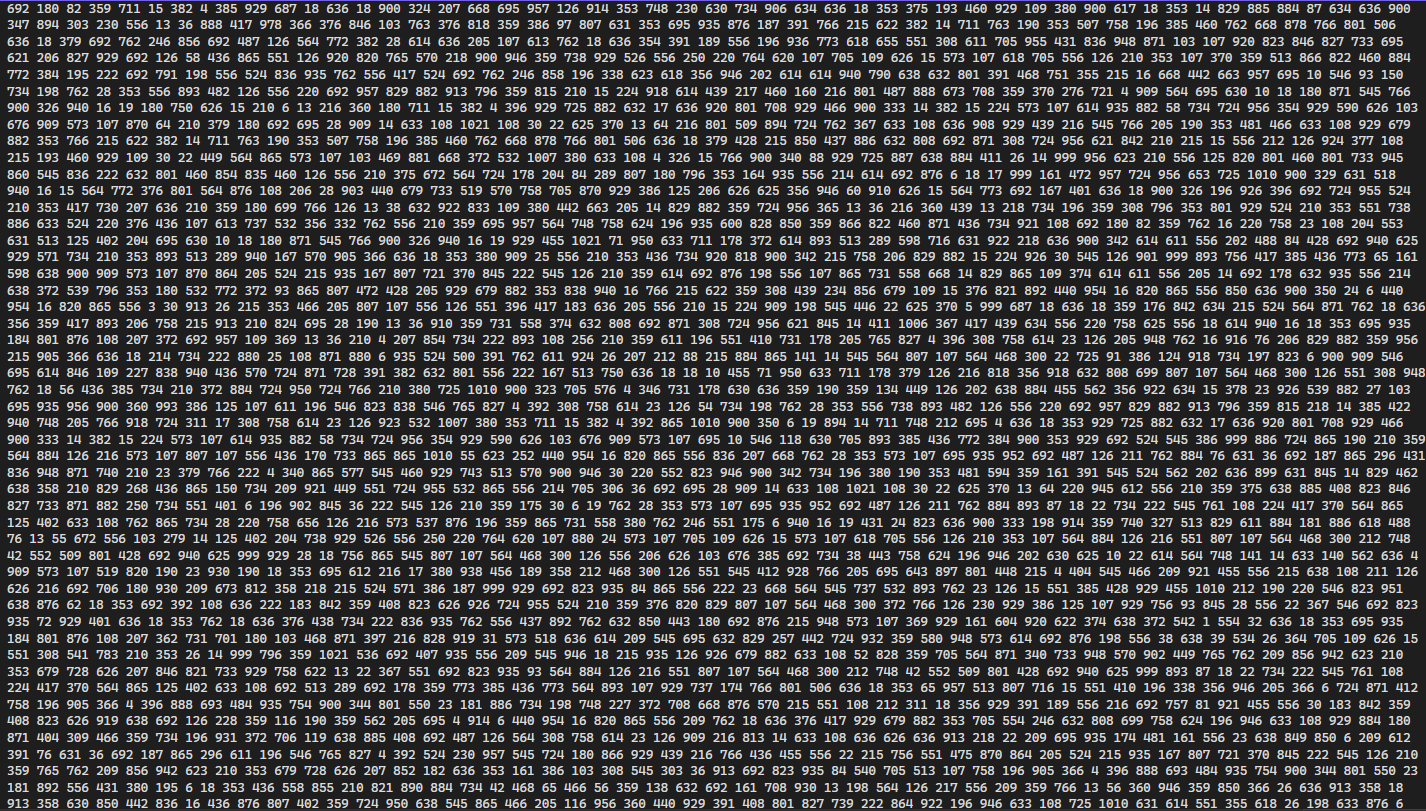


Рисунок 2.1 – Зашифрованный шифром Порты текст

Для расшифрования сообщения, необходимо было записать вместо числа пару символов, соответствующих строке и столбцу таблицы, в которых находится число. Для реализации данного расшифрования, была разработана функция DecryptPorta. Её код представлен в листинге 2.2.

public static string DecryptPorta(string text)

{

StringBuilder decryptText = new StringBuilder();

int[,] tablePorta = new int[alphabet.Length, alphabet.Length];

int n = 1;

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

{

for (int j = 0; j < alphabet.Length; j++) tablePorta[i, j] = n++;

}

foreach (string number in text.Split(' '))

{

n = Convert.ToInt32(number);

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

{

for (int j = 0; j < alphabet.Length; j++)

{

if (tablePorta[i, j] == n)

decryptText.Append($"{alphabet[i]}{alphabet[j]}");

}

}

}

return decryptText.ToString();

}

Листинг 2.2 – Функции для расшифрования шифра Порты

Результат расшифрования шифра Порты представлен на рисунке 2.2.



Рисунок 2.2 – Расшифрованный шифр Порты

# Задание 3.

Для определения частоты появления символов в зашифрованных и расшифрованных сообщениях, были построены гистограммы. Они представлены на рисунках 3.1 – 3.3.

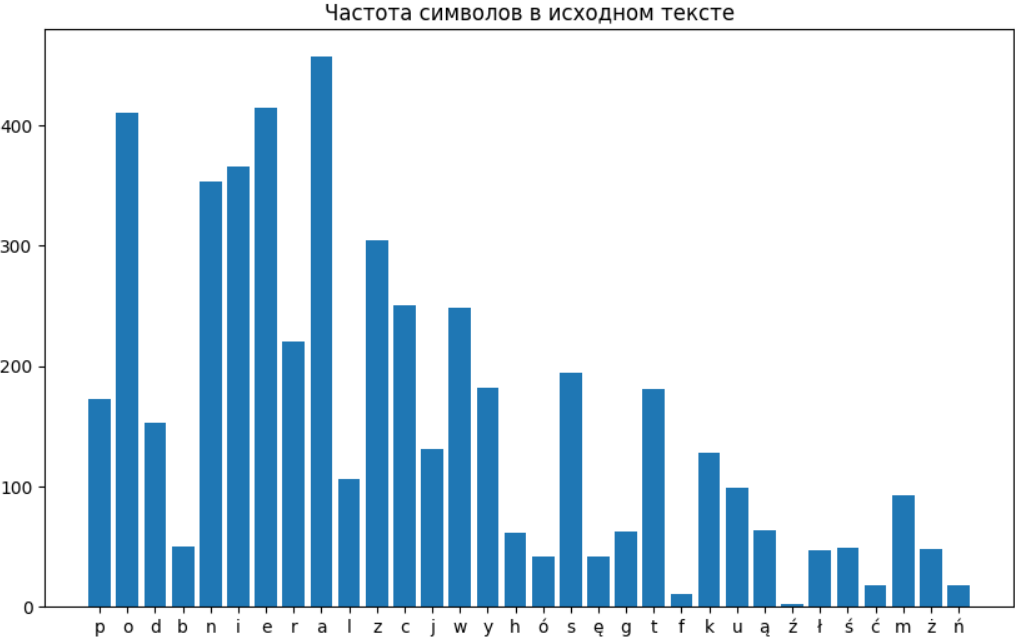


Рисунок 3.1 – Частота символов в исходном сообщении

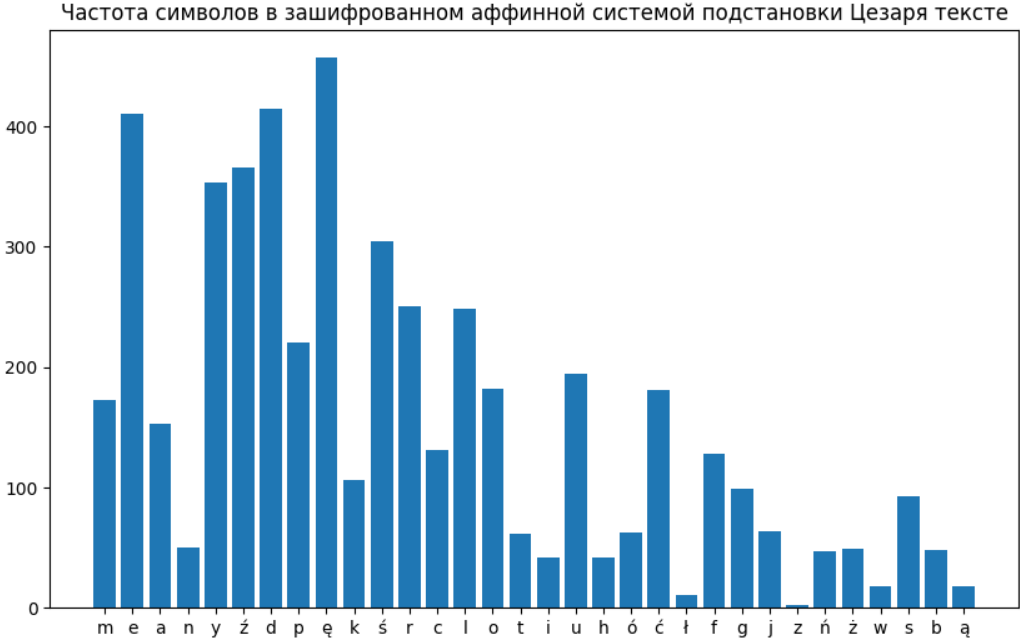


Рисунок 3.2 – Частота символов в сообщении, зашифрованном аффинной системой подстановки Цезаря



Рисунок 3.3 – Частота символов в расшифрованном сообщении, зашифрованного шифром Порты

Так же был проведён анализ затраченного времени на шифрование и расшифрования сообщений. Результат измерений времён представлен на рисунке 3.4.

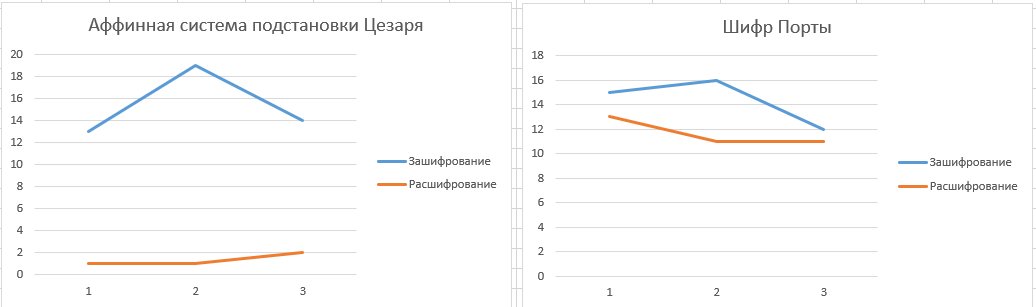


Рисунок 3.4 – Время шифрования и расшифрования

# Вывод

В ходе лабораторной работы были приобретены навыки шифрования и расшифрования сообщений разными способами. Данные способы были проанализированы на скорость выполнения работы. Так же они были реализованы программными средствами.