Министерство науки и высшего образования РФ

ФГАОУ ВО Пермский национальный исследовательский

политехнический университет

Кафедра «Вычислительная математика, механика и биомеханика»

Отчет по лабораторной работе №6

по дисциплине «Информационная безопасность»

по теме: «Частотный криптоанализ»

Выполнил: студент группу ИСТ-22-2б Старков Д.Э.

Проверил: доцент каф. ВММБ Труфанов А.Н.

Пермь, 2025

**Содержание**

[Постановка задачи 3](#_Toc190278942)

[Описание алгоритма 5](#_Toc190278943)

[Графический интерфейс 5](#_Toc190278944)

[Внутренняя логика 5](#_Toc190278945)

[База данных 6](#_Toc190278946)

[Выводы 7](#_Toc190278947)

[Приложение 8](#_Toc190278948)

# Постановка задачи

Разработать и реализовать программу криптоанализа зашифрованного текстового файла. Для разработки использовать из перечисленных сред программирования: Turbo Pascal, Delphi, C++ Builder, Visual C++ или другие языки по согласованию с преподавателем.

1. Программа должна обеспечивать работу в двух режимах:

* Криптоанализ шифротекста, зашифрованного методом Цезаря;
* Криптоанализ шифротекста, зашифрованного методом Виженера, при условии, что длина ключа неизвестна.

1. В режиме криптоанализа шифротекста, зашифрованного методом Цезаря, должны быть реализованы следующие функции:

* На основании частотного анализа шифротекста, полученного с помощью программы из лабораторной работы №4, определить ключ шифра;
* Расшифровать текст.

1. В режиме криптоанализа шифротекста, зашифрованного методом Виженера, должны быть реализованы следующие функции:

* На основании частотного анализа или других методов определить длину ключа;
* На основании частотного анализа определить алфавит замены;
* Расшифровать текст.

1. Для выполнения задания необходимо:

* С помощью программы, написанной для выполнения лабораторной работы №4, подготовить 2 текста, зашифрованные методами Цезаря и Виженера (длина ключа — 4 символа; длина текста — не менее 2000 знаков; использовать текст только на русском языке);
* Написать вспомогательную программу определения частотных характеристик текста на русском языке (т.е. подсчета вероятностей букв, наиболее вероятных биграмм, в том числе удвоений "лл", "нн", "ее" и т.д.). Построить диаграммы, на которых отразить 10 наиболее встречаемых букв и биграмм. На основании большого текста (более 100 000 знаков, любое доступное в открытых источниках художественное произведение на русском языке) определить частотные характеристики.

# Описание алгоритма

## Шифр Цезаря

Система шифрования Цезаря является частным случаем шифра простой замены. Метод основан на замене каждой буквы сообщения на другую букву того же алфавита путем смещения от исходной буквы на К букв. Для того взломать такой шифр при помощи частотного анализа необходимо рассмотреть график частотного распределения для русского алфавита на рисунке №1, на котором видно, как на основе какого-то достаточно большого текста подведена статистика

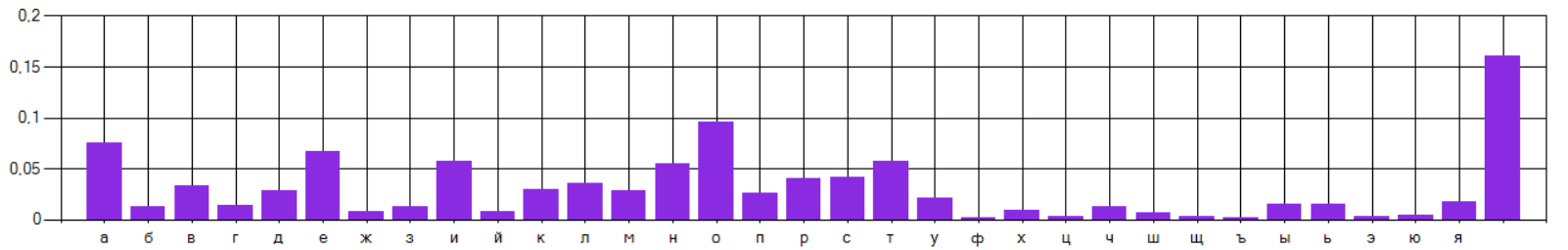


Рис. №1. График распределения без смещений

А сейчас зашифруем этот же текст шифром Цезаря со сдвигом в 5, график продемонстрирован на рисунке №2

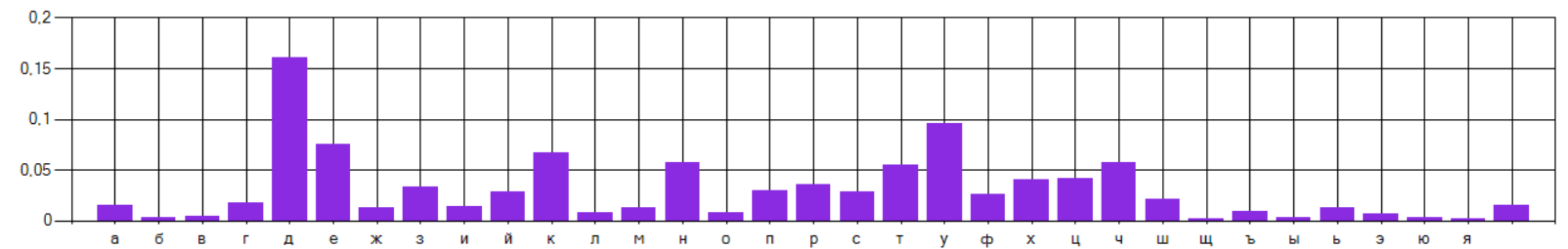


Рис. №2. График распределения со смещением

На втором графике можно увидеть, как все значения просто сместились на 5 значений вправо, значит для того, чтобы взломать этот шифр можно перебирать всевозможные сдвиги в алфавите, пока не прийдём к средним показателям. В качестве средних показателей были взяты параметры, продемонстрированные на рисунке №3

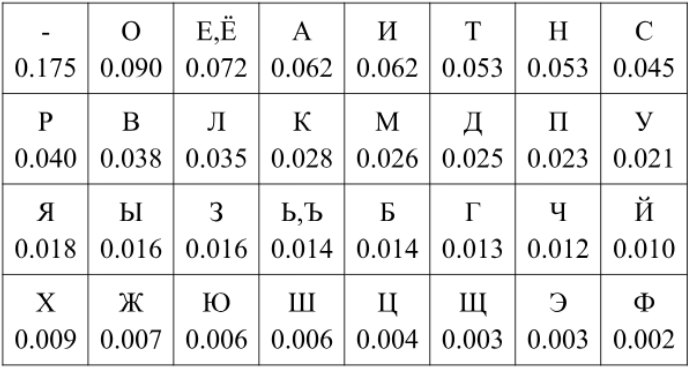


Рис. №3. Частотное распределение букв в алфавите

В алгоритме происходит следующее:

1. Считается ожидаемое количество встретить каждую букву, исходя из вероятностей и длины текста
2. Считается хи – квадрат для каждого смещения
3. Находится наименьший результат, он и будет конечным ответом



Рис. №4. Алгоритм взлома шифра Цезаря

## Шифр Виженера

Шифр Виженера можно рассмотреть, как несколько шифров Цезаря, но для того, чтобы корректно определить как много шифров Цезаря надо взять необходимо узнать длину ключа, а далее можно будет поделить на блоки. Для того чтобы узнать длину ключа воспользуемся индексом совпадений, а далее останется только разделить текст на блоки, и применить к ним метод взлома шифра Цезаря



Рис. №5. Алгоритм взлома шифра Виженера

# Выводы

* Ознакомился с методом частотного анализа
* Были проучены практически кие навыки создания ПО для криптоанализа зашифрованного текста

# Приложение

## Взлом шифра Цезаря

import java.util.Arrays;

public class Lab6Caesar {

    private static final String ABC = "АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ ";

    private static final double[] russianLettersProbabilities = {

            0.062, 0.014, 0.038, 0.013, 0.025, 0.072,  0.072,

            0.007, 0.016, 0.062, 0.010,  0.028, 0.035,  0.026,

            0.053, 0.090, 0.023, 0.040, 0.045, 0.053, 0.021,

            0.002,  0.009, 0.004, 0.012, 0.006, 0.003, 0.014,

            0.016, 0.014, 0.003, 0.006, 0.018, 0.175

    };

    public static void main(String[] args) {

        String cipher = Lab4.encryptCaesar("ПРИВЕТ МИР", 24);

        System.out.println(breakCipher(cipher));

    }

    public static int breakCipher (String text) {

        double[] expectedLettersFrequencies = Arrays.stream(russianLettersProbabilities)

                .map(probability -> probability \* text.length())

                .toArray();

        double[] chiSquares = new double[34];

        for (int i = 0; i < chiSquares.length; i++) {

            String decipheredMessage = Lab4.decryptCaesar(text, i);

            long[] lettersFrequencies = observedLettersFrequencies(decipheredMessage);

            double chiSquare = chiSquare(expectedLettersFrequencies, lettersFrequencies);

            chiSquares[i] = chiSquare;

        }

        int probableOffset = 0;

        for (int i = 0; i < chiSquares.length; i++) {

            // System.out.println((String.format("Chi-Square for offset %d: %.2f", i, chiSquares[i])));

            if (chiSquares[i] < chiSquares[probableOffset]) {

                probableOffset = i;

            }

        }

        return probableOffset;

    }

    static long[] observedLettersFrequencies(String message) {

        return ABC.chars()

                .mapToLong(letter -> countLetter((char) letter, message))

                .toArray();

    }

    static long countLetter(char letter, String message) {

        return message.chars()

                .filter(character -> character == letter)

                .count();

    }

    static double chiSquare(double[] expected, long[] observed) {

        double sum = 0.0;

        for (int i = 0; i < expected.length; i++) {

            if (expected[i] > 0) {

                double diff = observed[i] - expected[i];

                sum += (diff \* diff) / expected[i];

            }

        }

        return sum;

    }

}

## Взлом шифра Виженера

import java.util.stream.Collectors;

import java.util.stream.IntStream;

public class Lab6Vigener {

    private static final String ABC = "АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ ";

    public static void main(String[] args) {

        String text = Lab4.read("input.txt").toUpperCase();

        String cipher = Lab4.encryptVigener(text, "АДИДАС");

        System.out.println(keyLenght(cipher));

        System.out.println(breakCipherVigener(cipher));

    }

    static String breakCipherVigener (String text) {

        int lenght = keyLenght(text);

        StringBuilder stringBuilder = new StringBuilder();

        for (int i = 0; i < lenght; i++) {

            int groupIndex = i;

            String group = IntStream.range(0, text.length())

                    .filter(k -> k % lenght == groupIndex)

                    .mapToObj(text::charAt)

                    .map(Object::toString)

                    .collect(Collectors.joining());

            int offset = Lab6Caesar.breakCipher(group);

            stringBuilder.append(ABC.charAt(offset));

            //System.out.println(ABC.charAt(offset));

        }

        return stringBuilder.toString();

    }

    static int keyLenght (String text) {

        double[] indexes = new double[10];

        for (int i = 1; i <= 10; i++) {

            int possibleLength = i;

            double sumIC = 0;

            // Разделяем текст на possibleLength групп

            for (int j = 0; j < possibleLength; j++) {

                int groupIndex = j;

                String group = IntStream.range(0, text.length())

                        .filter(k -> k % possibleLength == groupIndex)

                        .mapToObj(text::charAt)

                        .map(Object::toString)

                        .collect(Collectors.joining());

                // Вычисляем индекс совпадений для группы

                sumIC += indexOfMatch(group);

            }

            // Усредняем индекс совпадений для текущей длины ключа

            indexes[i - 1] = sumIC / possibleLength;

        }

        double average = 0, max = 0;

        for (double value : indexes) {

            average += value;

            if (value > max) {

                max = value;

            }

        }

        average /= indexes.length;

        average = (max + average) / 2;

        if((max - average) < 0.0035) return 1;

        int length = 0;

        for (int i = 0; i < indexes.length; i++) {

            if (indexes[i] > average) {

                length = i + 1;

                break;

            }

        }

        return length;

    }

    static double indexOfMatch (String text) {

        int length = text.length();

        if (length == 1) {

            return 0;

        }

        double sum = 0;

        long[] lettersFrequencies = Lab6Caesar.observedLettersFrequencies(text);

        for (int i = 0; i < lettersFrequencies.length; i++) {

            if (lettersFrequencies[i] == 0) {

                continue;

            }

            sum += Math.pow((double) lettersFrequencies[i] / length, 2);

        }

        return sum;

    }