Министерство науки и высшего образования РФ

ФГАОУ ВО Пермский национальный исследовательский

политехнический университет

Кафедра «Вычислительная математика, механика и биомеханика»

Отчет по лабораторной работе №5

по дисциплине «Информационная безопасность»

по теме: «Анализ устойчивости парольной защиты»

Выполнил: студент группу ИСТ-22-2б Субботин М.О.

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 анализа устойчивости парольной аутентификации методу «полного перебора». Для разработки использовать из перечисленных сред программирования: Turbo Pascal, Delphi, C++ Builder, Visual C++ или другие языки по согласованию с преподавателем.

Требования к программе:

Программа должна обеспечивать работу в двух режимах:

1. Проверка надежности выбранного пароля (лабораторная № 1).
2. Подбор пароля при известном логине ADMIN.

В режиме проверки надежности пароля программа должна поддерживать функции, перечисленные в лабораторной работе №1. Необходим для оценки времени, которое займет "полный перебор" паролей.

1. В режиме анализа устойчивости выбранного пароля к методу "полного перебора" программа должна поддерживать следующие функции:

* Подбор пароля по словарю (при условии, что пароль набран русскими буквами в латинской раскладке, например, пароль "пароль" = "gflqknn");
* Подбор пароля перебором всех возможных вариантов, не превышающих заданной пользователем длины;
* В процессе подбора пароля вывести среднюю скорость перебора паролей в секунду.

1. Для выполнения задания необходимо:

* Доработать программы лабораторных №1 и №3 для обеспечения возможности перебора паролей в соответствии с заданием;
* Сформировать словарь (или использовать готовый из открытых источников) слов русского языка объемом не менее 2000 слов.

# Описание алгоритма

Для корректного выполнения работы необходимо реализовать программу в двух режима:

* с помощью словаря;
* с помощью перебора всевозможных комбинаций

Для реализации метода с использованием словаря необходим дополнительной метод, который будет переводить русское слово в английское в соответствии с раскладкой qwerty. Сам метод продемонстрирован на рисунке №1 и представляет собой нахождение индекса символа в строке с русским алфавитом и добавление, соответствующего символа из строки с английским алфавитом к итоговому результату.

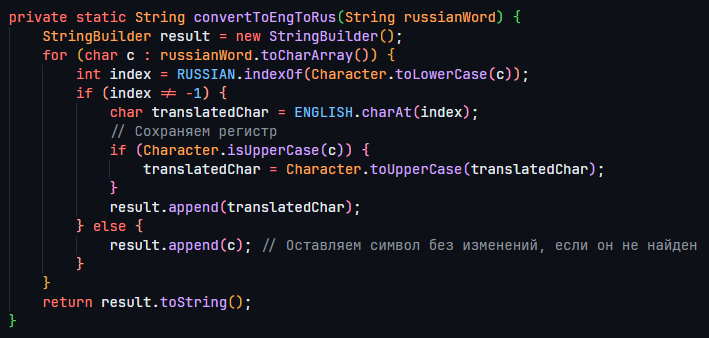


Рис. №1. Метод для перевода строки

Сам метод для перебора словарём представляет собой чтение слова из заранее подготовленного файла, перевод его в английский текст и сравнение, метод представлен на рисунке №2

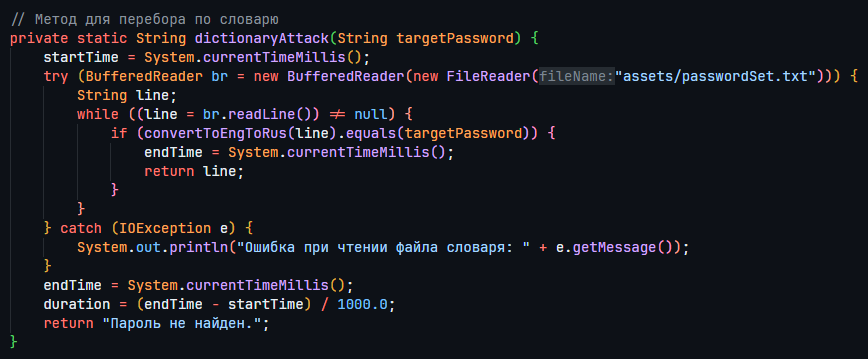


Рис.№2. Метод перебора словарём

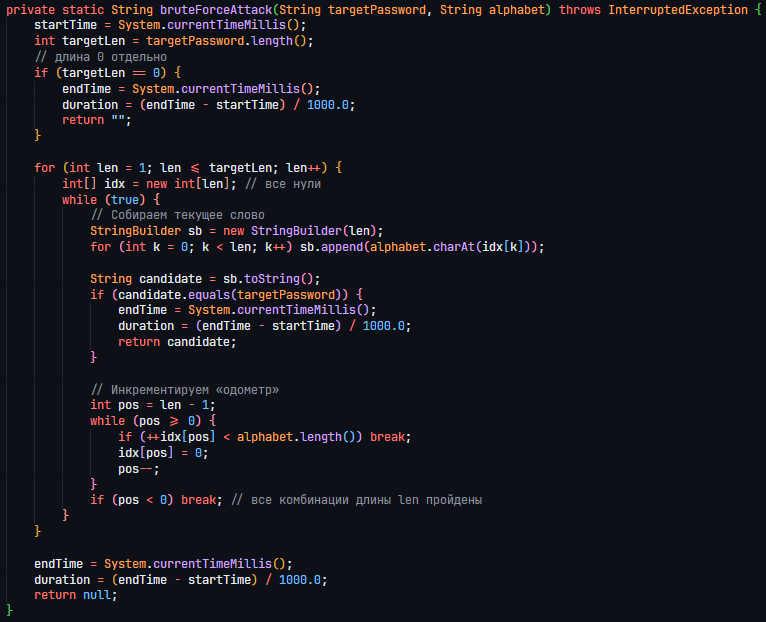
Для реализации альтернативного варианта метода полного перебора используется **итеративный алгоритм «одометр»**. Алгоритм поддерживает массив индексов длины L (где L — текущая длина генерируемых строк). Массив ведёт себя как одометр: при каждой итерации он инкрементируется в младшей позиции и при переполнении переносится в старшую. Текущая комбинация формируется по этим индексам путём выбора символов из заданного алфавита, после чего она немедленно проверяется на совпадение с целевым паролем. При попадании выполнение прекращается — найденная комбинация возвращается и измеряется время.

Рис. №3. Метод «полного перебора»

# Выводы

* Бали получены базовые знания парольной системы аутентификации
* Были получены практические навыки создания ПО для оценки устойчивости парольной защиты к атаке методом «полного перебора»

# Приложение

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

import java.util.ArrayDeque;

import java.util.Queue;

public class Main {

    private static final String RUSSIAN = "йцукенгшщзхъфывапролджэячсмитьбю.ё";

    private static final String ENGLISH = "qwertyuiop[]asdfghjkl;'zxcvbnm,./`";

    private static long startTime;

    private static long endTime;

    private static double duration;

    public static void main(String[] args) throws InterruptedException {

        String currentPassword = "qwerty";

        System.out.println(dictionaryAttack(currentPassword));

        System.out.println(bruteForceAttack(currentPassword, ENGLISH + "1234567890"));

        System.out.println(duration);

    }

    private static String convertToEngToRus(String russianWord) {

        StringBuilder result = new StringBuilder();

        for (char c : russianWord.toCharArray()) {

            int index = RUSSIAN.indexOf(Character.toLowerCase(c));

            if (index != -1) {

                char translatedChar = ENGLISH.charAt(index);

                // Сохраняем регистр

                if (Character.isUpperCase(c)) {

                    translatedChar = Character.toUpperCase(translatedChar);

                }

                result.append(translatedChar);

            } else {

                result.append(c); // Оставляем символ без изменений, если он не найден

            }

        }

        return result.toString();

    }

    // Метод для перебора по словарю

    private static String dictionaryAttack(String targetPassword) {

        startTime = System.currentTimeMillis();

        try (BufferedReader br = new BufferedReader(new FileReader("assets/passwordSet.txt"))) {

            String line;

            while ((line = br.readLine()) != null) {

                if (convertToEngToRus(line).equals(targetPassword)) {

                    endTime = System.currentTimeMillis();

                    return line;

                }

            }

        } catch (IOException e) {

            System.out.println("Ошибка при чтении файла словаря: " + e.getMessage());

        }

        endTime = System.currentTimeMillis();

        duration = (endTime - startTime) / 1000.0;

        return "Пароль не найден.";

    }

    private static String bruteForceAttack(String targetPassword, String alphabet) throws InterruptedException {

        startTime = System.currentTimeMillis();

        int targetLen = targetPassword.length();

        // длина 0 отдельно

        if (targetLen == 0) {

            endTime = System.currentTimeMillis();

            duration = (endTime - startTime) / 1000.0;

            return "";

        }

        for (int len = 1; len <= targetLen; len++) {

            int[] idx = new int[len]; // все нули

            while (true) {

                // Собираем текущее слово

                StringBuilder sb = new StringBuilder(len);

                for (int k = 0; k < len; k++) sb.append(alphabet.charAt(idx[k]));

                String candidate = sb.toString();

                if (candidate.equals(targetPassword)) {

                    endTime = System.currentTimeMillis();

                    duration = (endTime - startTime) / 1000.0;

                    return candidate;

                }

                // Инкрементируем «одометр»

                int pos = len - 1;

                while (pos >= 0) {

                    if (++idx[pos] < alphabet.length()) break;

                    idx[pos] = 0;

                    pos--;

                }

                if (pos < 0) break; // все комбинации длины len пройдены

            }

        }

        endTime = System.currentTimeMillis();

        duration = (endTime - startTime) / 1000.0;

        return null;

    }

}