Министерство науки и высшего образования РФ

ФГАОУ ВО Пермский национальный исследовательский

политехнический университет

Кафедра «Вычислительная математика, механика и биомеханика»

Отчет по лабораторной работе №1

по дисциплине «Информационная безопасность»

по теме: «Анализ надёжности пароля»

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

Проверил: доцент каф. ВММБ Труфанов А.Н.

Пермь, 2025

**Содержание**

[Постановка задачи 3](#_Toc189564358)

[Решение задачи 4](#_Toc189564359)

[Описание алгоритма 4](#_Toc189564360)

[Работа алгоритма 5](#_Toc189564361)

[Выводы 7](#_Toc189564362)

[Приложение 8](#_Toc189564363)

# Постановка задачи

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

Программа проверки надежности пароля должна поддерживать следующие функции:

* в зависимости от использования в пароле символов определять мощность алфавита и количество возможных комбинаций;
* в зависимости от введенных вручную параметров s, m, v(s – скорость перебора паролей в секунду, m количество неправильных попыток, после которых идёт пауза в v секунд) определить время перебора всех возможных комбинаций;
* время перебора всех возможных комбинаций выводить в формате «ЛЛ лет ММ месяцев ДД дней ЧЧ часов ММ минут СС секунд», для оценки принять 30 дней в месяце, и 365 дней в году.

# Решение задачи

## Описание алгоритма

Для проверки пароля была написана функция, проверяющая строку по следующим параметрам:

* длина пароля;
* наличие символов нижнего регистра;
* наличие символов верхнего регистра;
* наличие цифр;
* наличие специальных символов, удовлетворяющих алфавиту
* отсутствие символов, не содержащихся в алфавите

Сам алгоритм устроен следующим образом, в качестве входных данных выступает строка, содержащая проверяемый пароль. Далее создаются переменные в соответствии с параметрами, описанными выше, после чего открывается цикл for и в нём посимвольно перебирается строка. Выходными данными будет массив, содержащий boolean переменные по каждому из условий. Реализация алгоритма представлена на рисунке №1.

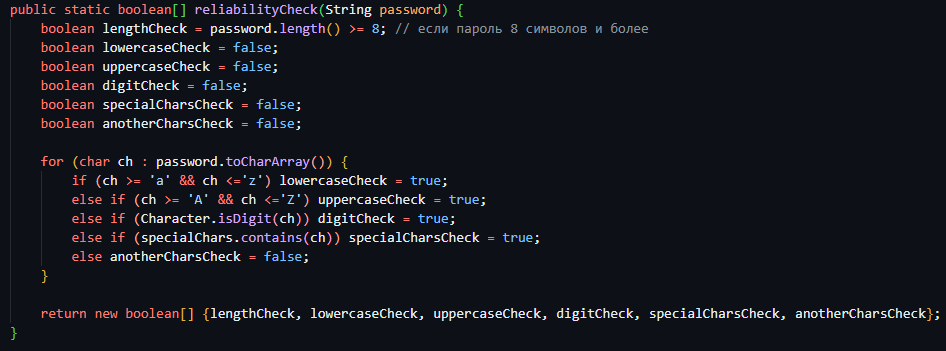


Рис. №1. Метод проверки

Далее идёт метод для подсчёта всевозможных комбинаций и времени грубого перебора, вычисление количества комбинаций происходят по следующей формуле  (где М – мощность пространства паролей, N – мощность алфавита, L – длина пароля), вычисление времени без учёта количества попыток и блока на следующий ввод происходит по следующей формуле , вычисление времени на блокировку ввода происходит в соответствии с условием на деление без остатка, само вычисление происходит по следующей формулам: 1) для деления без остатка ; 2) для деления с остатком . Входными данными является длина пароля. Выходными данными является массив, содержащий время и количество комбинаций. Реализация представлена на рисунке №2

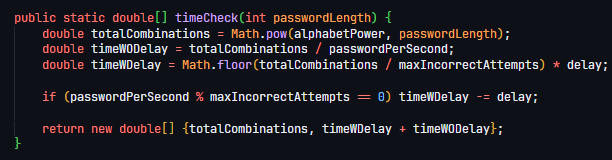


Рис. №2. Метод вычисления

Далее идёт метод для пересчёта времени. Алгоритм продемонстрирован на рисунке №3

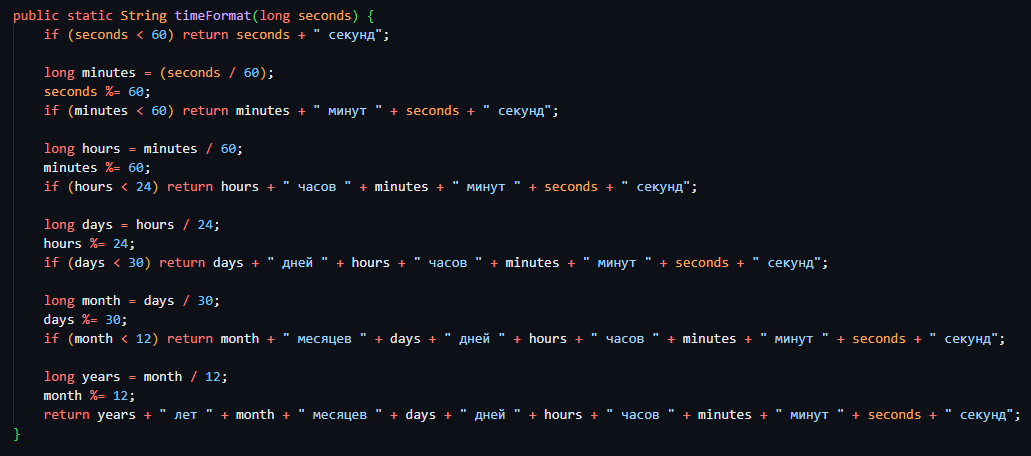


Рис. №3. Метод пересчёта времени

## Работа алгоритма

|  |  |  |  |
| --- | --- | --- | --- |
| № | Мощность алфавита | Длина пароля | Количество комбинаций |
| 1 | 10 | 8 | 1.0E8 |
| 2 | 26 | 8 | 2.08827064576E11 |
| 3 | 36 | 7 | 7.8364164096E10 |
| 4 | 95 | 6 | 7.35091890625E11 |

Вывод программы продемонстрирован на рисунке №4

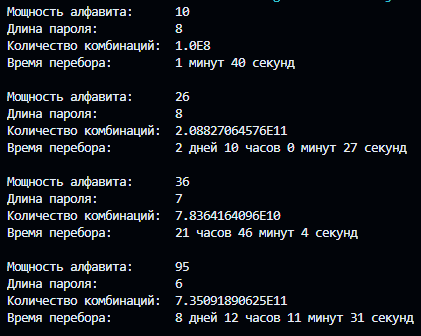


Рис. №4. Вывод программы

# Выводы

* Были изучены основные метода криптографической защиты
* Была реализована программы для проверки надежности пароля

# Приложение

import java.util.HashSet;

import java.util.Scanner;

import java.util.Set;

public class Main {

    private static int alphabetPower;               // Мощность алфавита: a-z + A-Z = 52.  | N = 52

    private static double passwordPerSecond = 10;  // Скорость перебора паролей в секунду | S

    private static int maxIncorrectAttempts = 5;    // Количество неправильных попыток     | T

    private static int delay = 10;                   // Пауза, после T неправильных попыток | V

    private static final Set<Character> specialChars = new HashSet<>();

    // символ двойные ковычки ["] и обратный слэш [\] нужно экранировать -> ["] = [\"], [\] -> [\\]

    static {

        for (char ch : " !\"#$%&'()\*+,-./:;<=>?@[\\]^\_`{|}~".toCharArray()) {

            specialChars.add(ch);

        }

    }

    public static void main(String[] args) {

        print(10, 8);

        print(26, 8);

        print(36, 7);

        print(95, 6);

        Scanner s = new Scanner(System.in);

        System.out.println("Введите ваш пароль:\n");

        String userPassword = s.nextLine();

        s.close();

        int[] userPasswordInfo = userPasswordCheck(userPassword);

        print(userPasswordInfo[0], userPasswordInfo[1]);

    }

    public static void print(int alphabetSize, int passwordLength) {

        alphabetPower = alphabetSize;

        double[] temp = timeCheck(passwordLength);

        System.out.println("Мощность алфавита:\t" + alphabetPower);

        System.out.println("Длина пароля:\t\t" + passwordLength);

        System.out.println("Количество комбинаций:\t" + temp[0]);

        System.out.println("Время перебора:\t\t" + timeFormat((long) temp[1]));

        System.out.println();

    }

    public static boolean[] reliabilityCheck(String password) {

        boolean lengthCheck = password.length() >= 8; // если пароль 8 символов и более

        boolean lowercaseCheck = false;

        boolean uppercaseCheck = false;

        boolean digitCheck = false;

        boolean specialCharsCheck = false;

        boolean anotherCharsCheck = false;

        for (char ch : password.toCharArray()) {

            if (ch >= 'a' && ch <='z') lowercaseCheck = true;

            else if (ch >= 'A' && ch <='Z') uppercaseCheck = true;

            else if (Character.isDigit(ch)) digitCheck = true;

            else if (specialChars.contains(ch)) specialCharsCheck = true;

            else anotherCharsCheck = false;

        }

        return new boolean[] {lengthCheck, lowercaseCheck, uppercaseCheck, digitCheck, specialCharsCheck, anotherCharsCheck};

    }

    public static double[] timeCheck(int passwordLength) {

        double totalCombinations = Math.pow(alphabetPower, passwordLength);               // M = N^L - мощность пространства паролей

        double timeWODelay = totalCombinations / passwordPerSecond;                       // количество секунд для проверки всех паролей

        double timeWDelay = Math.floor(totalCombinations / maxIncorrectAttempts) \* delay; // количество секунд для проверки всех паролей с учётом паузы при неправильных попытках

        if (passwordPerSecond % maxIncorrectAttempts == 0) timeWDelay -= delay;

        return new double[] {totalCombinations, timeWDelay + timeWODelay};

    }

    public static String timeFormat(long seconds) {

        if (seconds < 60) return seconds + " секунд";

        long minutes = (seconds / 60);

        seconds %= 60;

        if (minutes < 60) return minutes + " минут " + seconds + " секунд";

        long hours = minutes / 60;

        minutes %= 60;

        if (hours < 24) return hours + " часов " + minutes + " минут " + seconds + " секунд";

        long days = hours / 24;

        hours %= 24;

        if (days < 30) return days + " дней " + hours + " часов " + minutes + " минут " + seconds + " секунд";

        long month = days / 30;

        days %= 30;

        if (month < 12) return month + " месяцев " + days + " дней " + hours + " часов " + minutes + " минут " + seconds + " секунд";

        long years = month / 12;

        month %= 12;

        return years + " лет " + month + " месяцев " + days + " дней " + hours + " часов " + minutes + " минут " + seconds + " секунд";

    }

    public static int[] userPasswordCheck(String password) {

        alphabetPower = 0;

        boolean lowercaseCheck = false;

        boolean uppercaseCheck = false;

        boolean digitCheck = false;

        boolean specialCharsCheck = false;

        for (char ch : password.toCharArray()) {

            if (ch >= 'a' && ch <='z') lowercaseCheck = true;

            else if (ch >= 'A' && ch <='Z') uppercaseCheck = true;

            else if (Character.isDigit(ch)) digitCheck = true;

            else if (specialChars.contains(ch)) specialCharsCheck = true;

            else throw new IllegalArgumentException("Недопустимый символ в пароле: '" + ch + "'");

        }

        if (lowercaseCheck) alphabetPower += 26;

        if (uppercaseCheck) alphabetPower += 26;

        if (digitCheck) alphabetPower += 10;

        if (specialCharsCheck) alphabetPower += 33;

        return new int[] {alphabetPower, password.length()};

    }

}