Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Пермский национальный** **исследовательский политехнический университет»**

Факультет: Прикладной математики и механики

Кафедра: Вычислительной математики, механики и биомеханики

Направление: 09.03.02 «Информационные системы и технологии»

Профиль бакалавриата: «Информационные системы и технологии»

**Отчет по лабораторной работе**

по дисциплине

«**Информационная безопасность и защита информации**»

Лабораторная работа

**«Анализ надежности пароля»**

Выполнил студент гр. ИСТ-17-1б

Мальцев К.А.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Ф.И.О.)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

*(подпись)*

**Принял (преподаватель по дисциплине):**

доцент Труфанов А. Н.

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(должность, ФИО)*

*\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(оценка) (подпись)*

*\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(дата)*

**Пермь, 2020**

**Цель работы:** ознакомиться с основными методами криптографической защиты информации. Получить практические навыки создания программного модуля оценки надежности выбранного пароля.

**Описание алгоритма:**

Использованный алгоритм проверки надежности пароля можно разделить на несколько этапов:

1. Сбор данных с формы. Пользователь вводит необходимые значения полей: пароль (пароль, надежность которого необходимо проверить) и 3 поля необходимые для расчёта времени перебора всех возможных вариантов (s, m, v).
2. Передача данных в модель, реализующую логику, описанную ниже.
3. Передача результатов на форму.

Мощность алфавита определяется следующим образом: каждый символ пароля в цикле проверяется на принадлежность одной из заранее сформированных групп символов, затем в зависимости от использованных групп символов складывается мощность алфавита.

Количество возможных вариантов определяется формулой .

Весь описанный алгоритм повторяется каждый раз, когда пользователь изменяет значение поля пароль. А также, если пользователь изменяет одно из полей необходимых для расчёта времени, производится только перерасчет времени по формуле , где

NumOP – количество возможных операций,   
s – скорость перебора паролей в секунду,   
P – кол-во пауз за время перебора,

v – задержка в секундах во время паузы.

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Мощность алфавита** | **Длина пароля** | **Кол-во комбинаций** |
| 1 | 10 | 6 | 1000000 |
| 2 | 10 | 7 | 10000000 |
| 3 | 10 | 8 | 100000000 |
| 4 | 26 | 8 | 208827064576 |
| 5 | 36 | 7 | 78364164096 |
| 6 | 95 | 8 | 6,63420431289063E+15 |
| 7 | 95 | 6 | 735091890625 |

**Листинг программы**

namespace lab1

{

class PassAnalysis

{

int[] passPowerGr = { 0, 10, 26, 26, 33 }; // Мощность символов по группам

int alphPower = 0;

string password = "";

int s = 1000; // Скорость перебора паролей в секунду

int m = 0; // Неправ. попытки

int v = 0; // Пауза (сек) после m

public string Password // Установка пароля

{

set

{

password = value;

}

}

public int S // Установка s

{

set

{

s = value;

}

}

public int M // Установка m

{

set

{

m = value;

}

}

public int V // Установка v

{

set

{

v = value;

}

}

public double NumOp // Расчет и выдача кол-ва возможных операций

{

get

{

if (password.Length == 0)

return 0;

else

return Math.Pow(alphPower, password.Length);

}

}

public string TimeOp // Расчет и выдача времени перебора

{

get

{

double t;

if (s == 0)

t = 0;

else if (m == 0 || v == 0)

t = NumOp / s;

else

t = (NumOp / s) + Math.Floor(NumOp / m) \* v;

double sec = t % 60;

t = Math.Floor(t / 60);

double min = t % 60;

t = Math.Floor(t / 60);

double hours = t % 24;

t = Math.Floor(t / 24);

double days = t % 30;

t = Math.Floor(t / 30);

double months = t % 12;

double years = Math.Floor(t / 12);

return string.Format($"{years} лет {months} месяцев {days} дней {hours} часов {min} минут {sec} секунд");

}

}

public int AlphPower // Расчет и выдача мощности алфавита

{

get

{

PowerAnalysis();

return alphPower;

}

}

int PowerAnalysis() // Расчет мощности алфавита

{

bool[] isbelongToGr = new bool[5];

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

{

int sw = PowerChar(password[i]);

switch (sw)

{

case 0:

alphPower = 0;

return 0;

default:

isbelongToGr[sw] = true;

break;

}

}

alphPower = 0;

for (int i = 1; i < passPowerGr.Length; i++)

{

if (isbelongToGr[i])

alphPower += passPowerGr[i];

}

return alphPower;

}

int PowerChar(char ch) // Расчет мощности символа

{

if (ch >= 48 && ch <= 57)

{

// Цифры +10

return 1;

}

if (ch >= 65 && ch <= 90)

{

// Пропись +26

return 2;

}

if (ch >= 97 && ch <= 122)

{

// Строчные +26

return 3;

}

if (ch >= 32 && ch <= 126)

{

// Спец. символы +33

return 4;

}

else

{

// Неверный ввод

return 0;

}

}

}

}