ЛАБОРАТОРНАЯ РАБОТА № 1

НАЗВАНИЕ ЛАБОРАТОРНОЙ РАБОТЫ

ВЫПОЛНИЛ: ст. гр. 4411 Гайфуллин Д. Р.

ВАРИАНТ № 2

*Цель работы* – исследование парольных подсистем аутентификации пользователей. Реализация простейшего генератора паролей, обладающего требуемой стойкостью к взлому.

*P*= 10-5

*V*= 3 паролей/мин

*T*= 10 дней

*S\**= =

В качестве алфавита символов, используемых при генерации пароля, были использованы следующие наборы:

1. Латинские прописные буквы (A-Z)

2. Русские строчные буквы (а-я)

Мощность данного набора *A* =59

При минимальном значении *L*= 6 выполняется условие *S\**≤*S*=*AL*.

Примеры паролей, сгенерированных программой Simple Passwords:

1) ZLиWлэ

2) Bвбсбщ

3) ZDEнйт

4) EдгFDь

5) яаьыLф

6) мHUйWW

7) лWDCфW

8) вXBCHY

9) IBTVгP

10) эAтLKо

11) иыSIET

12) тLJJQб

13) жлйфэI

14) тFRBгй

15) оиODHу

16) щHцHею

17) XBVBмX

18) зJGIаB

19) LIPAъф

20) BNгыбй

ТЕКСТ ПРОГРАММЫ

main.js

import rls from 'readline-sync';

import calculate from './calculate.js';

import utils from './utils/utils.js';

import consts from './utils/consts.js';

*const* {calculateA, calculateSL, calculateL} = calculate;

*const* {generateAlphabet, generatePasswords, printPasswords, strToInt} = utils;

*const* {P, V, T, intervalEngUpper, intervalRusLower} = consts;

function caseGeneratePasswords(L, alphabet) {

*var* res = rls.question('Enter the number of passwords: ');

    res = strToInt(res);

    if (res > 0){

        printPasswords(L, generatePasswords(res, L, alphabet));

    } else {

        console.log('Incorrect input.');

    }

};

function workCycle(L, alphabet) {

*var* run = true;

    while (run) {

*var* answer = rls.question('0 - generate password/passwords;\n1 - exit.\ncommand: ');

        switch(answer) {

            case '0':

                caseGeneratePasswords(L, alphabet);

                break;

            case '1':

                run = false;

                break;

            default:

                console.log('Incorrect input.')

                break;

        }

    }

};

function init() {

*const* A = calculateA(intervalEngUpper, intervalRusLower);

*const* SL = calculateSL(P, V, T);

*const* L = calculateL(A, SL);

*const* alphabet = generateAlphabet(intervalEngUpper, intervalRusLower);

*return* {L, alphabet};

};

function main() {

*const* {L, alphabet} = init();

    workCycle(L, alphabet);

};

main();

utils.js

function getRandInt(min, max) {

*return* Math.floor(Math.random() \* (max - min + 1) + min);

};

function getSymbols(firstASCII, lastASCII) {

*var* symbols = [];

    for (*var* i = firstASCII; i <= lastASCII; i++) {

        symbols.push(String.fromCharCode(i));

    };

*return* symbols;

}

function generateAlphabet(**...**intervals) {

*return* intervals.reduce(

        (alphabet, interval) => alphabet.concat(getSymbols(interval[0], interval[1])), []);

};

function generatePassword(passwordLength, alphabet) {

*var* password = '';

    for (*var* i = 0; i < passwordLength; i++) {

        password += alphabet[getRandInt(0, alphabet.length - 1)];

    }

*return* password;

};

function generatePasswords(passwordsQuantity, passwordLength, alphabet) {

*var* passwords = [];

    for (*var* i = 0; i < passwordsQuantity; i++) {

        passwords.push(generatePassword(passwordLength, alphabet));

    }

*return* passwords;

};

function printPasswords(L, passwords) {

    console.log(

        `Calculated password length (L): ${L}\nPasswords:`

    );

    passwords.forEach(password => {

        console.log(`${password}`);

    })

};

function checkStrIsInt(data) {

*return* !isNaN(data) && Number.isInteger(+data);

};

function strToInt(data) {

*return* checkStrIsInt(data) ? +data : 0;

};

export default {

    getRandInt, getSymbols, generateAlphabet, generatePassword, generatePasswords, printPasswords, checkStrIsInt, strToInt

}

calculate.js

*/\*Вычисление A (мощность алфовита)\*/*

function calculateA (**...**intervals) {

*return* intervals.reduce((result, interval) => result + interval[1] - interval[0] + 1, 1);

};

*/\*Вычисление SL (нижняя граница чисола всевозможных паролей)\*/*

function calculateSL(P, V, T) {

*return* Math.ceil(V \* T / P);

};

*/\*Вычисление L (длина пароля)\*/*

function calculateL (A, SL) {

*var* L = 1;

    while (!(A \*\* L >= SL)) {

        L++;

    };

*return* L;

};

export default {

    calculateA, calculateSL, calculateL

};

consts.js

Примеры сгенерированных программой паролей:

1. яDKкеи
2. вьвдвV
3. фзJNэS
4. NEлцFK
5. эамчоP
6. уLлдVс
7. KнFEкд
8. ззWFаь
9. жBюKнт
10. уBNмBя
11. эHоинW
12. HGэуаL
13. цSыWзM
14. свмYJо
15. зXвдEы
16. Cшуэду
17. шзHуиQ
18. щLоцгP
19. AлшьзT
20. пгжBUв