Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

Кафедра информационных технологий автоматизированных систем

Отчет по лабораторной работе №1

по курсу «ИТОКБ»

на тему: «Генерация паролей»

Вариант 4

|  |  |
| --- | --- |
| Выполнил магистрант группы 025941: | Колесников В.Г. |
| Проверил: | Боброва Т.С. |

МИНСК

2021

**Задание 1:** Разработать программу на языке C++, реализующую следующие функции:

- генерация строки заданной пользователем длины, состоящей из символов русского алфавита (строчные и прописные);

- проверка равномерности распределения символов путем визуализации частотного распределения;

- вычисление среднего времени подбора пароля, выбираемого из сгенерированной строки.

Код программы приведен в конце лабораторной работы.

Результаты работы программы генерации пароля приведены на рисунках 1.1, 1.2 и 1.3.



Рисунок 1.1 — Результат генерации пароля на 10 символов



Рисунок 1.2 — Результат генерации пароля на 20 символов



Рисунок 1.3 — Результат генерации пароля на 30 символов

Была произведена проверка равномерности распределения на 660000 символов. Результат проверки приведен на рисунке 1.4.

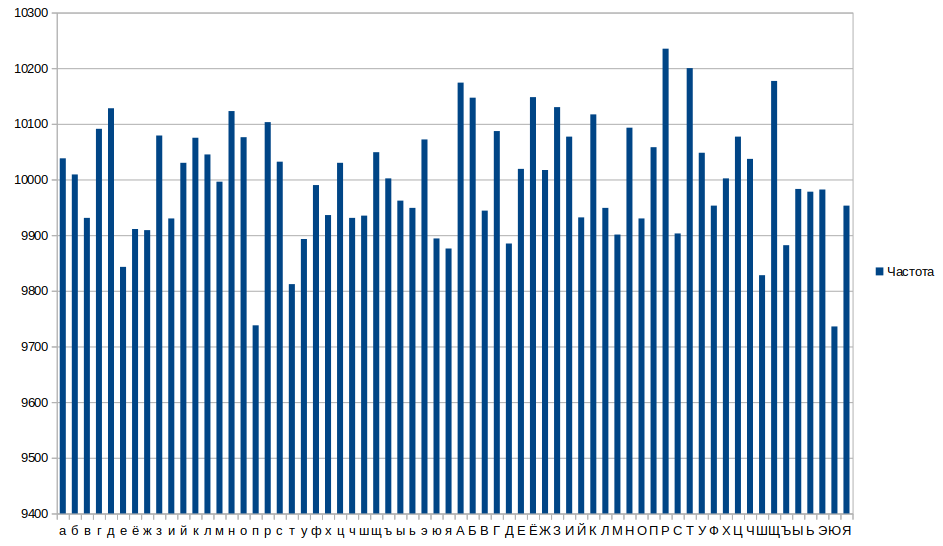


Рисунок 1.4 — Частотное распределение символов

Результаты подбора паролей на определенное количество символов приведены на рисунке 1.5.

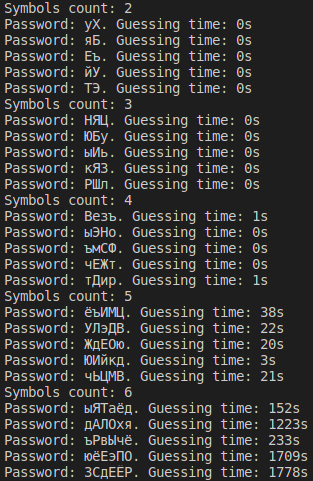


Рисунок 1.5 — Результаты подбора паролей

Среднее время подборов паролей по количеству символов приведено в таблице 1.1.

Таблица 1.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Количество символов | 2 | 3 | 4 | 5 | 6 |
| Среднее время подбора (секунд) | 0.000001 | 0.01 | 0.4 | 20.8 | 1019 |

Время на подбор остальных паролей рассчитывается вручную, т.к. оно слишком велико. Данные рассчеты приведены в таблице 1.2.

Таблица 1.2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Количество символов | 7 | 8 | 9 | 10 |
| Среднее время подбора (секунд) | 3500 | 12033 | 200000 | 9417600 |

**Задание 2:** Построить график зависимости среднего времени подбора пароля от его длины.

График зависимости среднего времени подбора пароля от его длины представлен на рисунке 2.1.

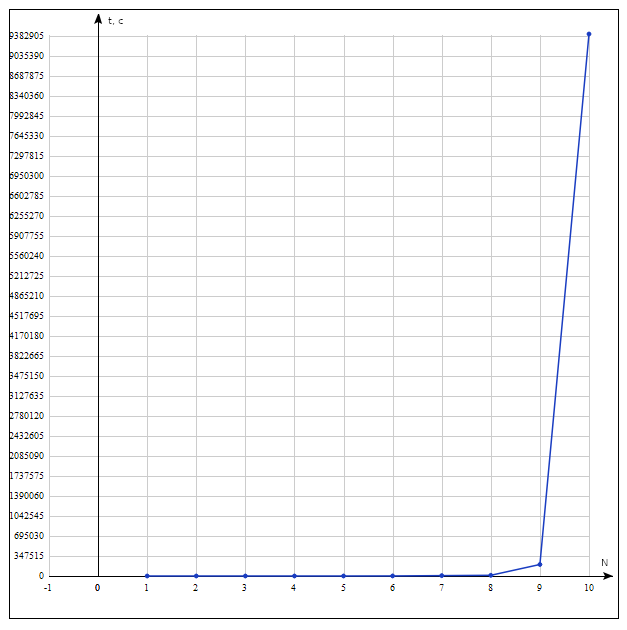


Рисунок 2.1 — График зависимости среднего времени подбора пароля от его длины

**Задание 3:** Дать практические рекомендации по выбору пароля исходя из предположений б алфавите пароля, ценности информации, доступ к которой защищается с помощью этого пароля, производительности вычислительного средства атакующего и времени атаки.

Время подбораэкспоненциально зависит от мощности алфавита и длины подбираемого пароля. Дольше всего будет подбираться пароль, состоящий полностью из последней буквы в алфавите (Z), быстрее всего – из первой (a).

Даже в условиях сильно ограниченного времени, 121 секунды должно хватить на подбор пароля по модели «скринсейвера», когда оператор машины временно отсутствует. Так что пароль рекомендуется делать не менее 6-ти символов.

Если в систему имеется доступ извне и нет ограничений по вводу паролей, нежелательно делать пароль короче, чем 8 символов.

Если система обладает высокой ценностью и пароль на ней меняется редко, желательно использовать пароли от 9 и более символов.

Существует несколько видов атак, которые упрощают задачу подбора пароля. Для этого во многих IT-компаниях введен стандарт — от 15 символов, состоящих из букв верхнего и нижнего регистра, цифр, а также нестандартных для алфавита знаков и специальных символов. Кроме того, рекомендуется менять пароль не реже чем через каждые три месяца для исключения его компрометации.

Для улучшенной защиты данных также могут использоваться такие методы, как двухфакторная аутентификация и одноразовые пароли.

**Код программы:**

#include <iostream>

#include <time.h>

const int createPassword(const wchar\_t \*alphabet, const int lettersCount)

{

int symbCount;

wprintf(L"Enter symbols count: ");

wscanf(L"%d", &symbCount);

wchar\_t \*pass = new wchar\_t[symbCount + 1];

for (int i = 0; i < symbCount; i++)

{

const int letterNumber = rand() % lettersCount;

// wprintf(L"Next number: %d\n", letterNumber);

pass[i] = alphabet[letterNumber];

}

pass[symbCount] = L'\0';

wprintf(L"Generated password: %ls\n", pass);

delete pass;

return 0;

}

const int createDistribution(const wchar\_t \*alphabet, const int lettersCount)

{

int \*lettersDistribution = new int[lettersCount];

for (int i = 0; i < lettersCount \* 10000; i++)

{

const int letterNumber = rand() % lettersCount;

lettersDistribution[letterNumber]++;

}

for (int i = 0; i < lettersCount; i++)

{

wprintf(L"%lc: %ld\n", alphabet[i], lettersDistribution[i]);

}

delete lettersDistribution;

return 0;

}

const int calcAvgGuessTime(const wchar\_t \*alphabet, const int lettersCount)

{

const int countOfTries = 5;

const int symbolCounts[countOfTries] = {2, 3, 4, 5, 6};

for (int symbCountNumber = 0; symbCountNumber < countOfTries; symbCountNumber++)

{

const long symbCount = symbolCounts[symbCountNumber];

wprintf(L"Symbols count: %ld\n", symbCount);

for (int tryNumb = 1; tryNumb < countOfTries + 1; tryNumb++)

{

// Password generation here

wchar\_t \*pass = new wchar\_t[symbCount + 1];

pass[symbCount] = L'\0';

for (int i = 0; i < symbCount; i++)

{

const int letterNumber = rand() % lettersCount;

pass[i] = alphabet[letterNumber];

}

wprintf(L"Password: %ls. ", pass);

// Brute force

time\_t start = time(0);

int \*bfSymbNumbers = new int[symbCount];

for (int i = 0; i < symbCount; i++)

{

bfSymbNumbers[i] = 0;

}

int bfSymbNumbersSum = 0;

bool increaseNext = false;

bool isPassBruteForced = false;

do

{

for (int i = 0; i < symbCount; i++)

{

if (increaseNext)

{

bfSymbNumbers[i]++;

}

if (bfSymbNumbers[i] > lettersCount - 1)

{

bfSymbNumbers[i] = 0;

increaseNext = true;

}

else if (i == 0)

{

bfSymbNumbers[i]++;

}

else

{

increaseNext = false;

}

}

int similarCount = 0;

for (int i = 0; i < symbCount; i++)

{

if (alphabet[bfSymbNumbers[i]] == pass[i])

{

similarCount++;

}

}

if (similarCount == symbCount)

{

break;

}

bfSymbNumbersSum++;

} while (true);

time\_t end = time(0);

// Time calculation

const long int startInt = static\_cast<long int>(start);

const long int endInt = static\_cast<long int>(end);

const long int elapsed = endInt - startInt;

wprintf(L"Guessing time: %lds\n", elapsed);

delete pass;

delete bfSymbNumbers;

}

}

return 0;

}

int main(int argc, char \*\*argv)

{

setlocale(LC\_ALL, "");

srand(time(NULL));

const wchar\_t \*alphabet = L"абвгдеёжзийклмнопрстуфхцчшщъыьэюяАБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ";

const int lettersCount = wcslen(alphabet);

// а

// - 1

createPassword(alphabet, lettersCount);

// - 2

createDistribution(alphabet, lettersCount);

// - 3;

calcAvgGuessTime(alphabet, lettersCount);

return 0;

}