Министерство образования Республики Беларусь

Учреждение образования

Белорусский государственный университет информатики и

Радиоэлектроники

Факультет информационных технологий и управления

Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе №1

по курсу “Средства и методы защиты информации в интеллектуальных системах”

Вариант 3

Выполнил:

Студент гр. 321702 Рублевская Е. А.

Проверил: Захаров В. В.

Минск

2025

**Тема:** Генерация паролей

**Задание для самостоятельного выполнения:**

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

- генерация строки с заданной пользователем длиной, состоящей из символов алфавита в соответствии с вариантом задания (использовать функции rand( ), srand( ) и инициализацию от таймера);

- поверка равномерности распределения символов путем визуализации частотного распределения;

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

- Использовать строчные буквы русского алфавита

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

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

**Выполнение:**

**Код программы:**

**package org.example;**

**import java.util.\*;**

**import java.text.DecimalFormat;**

**public class Main {**

**// Функция генерации случайной строки**

**public static String generateRandomString(int length, String alphabet) {**

**StringBuilder result = new StringBuilder(length);**

**Random random = new Random();**

**for (int i = 0; i < length; i++) {**

**char randomChar = alphabet.charAt(random.nextInt(alphabet.length()));**

**result.append(randomChar);**

**}**

**return result.toString();**

**}**

**// Функция для вычисления частотного распределения символов**

**public static Map<Character, Integer> calculateFrequency(String str) {**

**Map<Character, Integer> frequency = new HashMap<>();**

**for (char ch : str.toCharArray()) {**

**frequency.put(ch, frequency.getOrDefault(ch, 0) + 1);**

**}**

**return frequency;**

**}**

**// Функция для визуализации частотного распределения**

**public static void visualizeFrequency(Map<Character, Integer> frequency) {**

**System.out.println("\nЧастотное распределение символов:");**

**for (Map.Entry<Character, Integer> entry : frequency.entrySet()) {**

**System.out.print(entry.getKey() + ": ");**

**for (int i = 0; i < entry.getValue(); i++) {**

**System.out.print("\*");**

**}**

**System.out.println(" (" + entry.getValue() + ")");**

**}**

**}**

**public static void generateCombinations(String alphabet, String password) {**

**int length = password.length();**

**int alphabetSize = alphabet.length();**

**// Массив индексов для комбинаций**

**int[] indices = new int[length];**

**long totalCombinations = (long) Math.pow(alphabetSize, length);**

**long startTime = System.nanoTime();**

**for (long i = 0; i < totalCombinations; i++) {**

**StringBuilder combination = new StringBuilder(length);**

**// Создаем строку из текущих индексов**

**for (int j = 0; j < length; j++) {**

**combination.append(alphabet.charAt(indices[j]));**

**}**

**// Проверяем, соответствует ли текущая строка паролю**

**if (combination.toString().equals(password)) {**

**long endTime = System.nanoTime();**

**// Вычисляем время в секундах**

**double timeTaken = (endTime - startTime) / 1e9;**

**// Форматируем время**

**DecimalFormat df = new DecimalFormat("#.######");**

**System.out.println("Пароль подобран: " + combination);**

**System.out.println("Время, затраченное на подбор: " + df.format(timeTaken) + " секунд");**

**return;**

**}**

**// Обновляем индексы (аналог системы счисления с основанием alphabetSize)**

**for (int j = length - 1; j >= 0; j--) {**

**indices[j]++;**

**if (indices[j] < alphabetSize) {**

**break;**

**}**

**indices[j] = 0; // Перенос "разряда"**

**}**

**}**

**System.out.println("Пароль не удалось подобрать.");**

**}**

**public static void main(String[] args) {**

**Scanner scanner = new Scanner(System.in);**

**String alphabet = "абвгдеёжзийклмнопрстуфхцчшщъыьэюя";**

**// Ввод длины строки**

**System.out.print("Введите длину строки: ");**

**int stringLength = scanner.nextInt();**

**// Генерация случайной строки**

**String randomString = generateRandomString(stringLength, alphabet);**

**System.out.println("\nСгенерированная строка: " + randomString);**

**// Подсчет частот символов и визуализация**

**Map<Character, Integer> frequency = calculateFrequency(randomString);**

**visualizeFrequency(frequency);**

**// Ввод пароля для проверки времени подбора**

**System.out.print("\nВведите пароль для вычисления среднего времени подбора: ");**

**scanner.nextLine(); // для очистки буфера**

**String password = scanner.nextLine();**

**// Подбор пароля и вывод времени**

**generateCombinations(alphabet, password);**

**}**

**}**

**Основные функции программы:**

* **String generateRandomString**:  
  Функция, которая принимает длину строки и алфавит (в данном случае – русский). Возвращает случайно сгенерированную строку из символов алфавита заданной длины. Выделяется память для новой строки через объект **StringBuilder**. Случайные символы выбираются на основе метода **Random.nextInt**, где индекс строго ограничен длиной алфавита.Каждый случайно выбранный символ добавляется в строку. Финальный результат возвращается.
* **Map<Character, Integer> calculateFrequency**:  
  Функция, которая принимает строку (например, ранее сгенерированную) и подсчитывает частоту появления каждого символа. Используется структура **HashMap** для хранения символов строки как ключей и количества их вхождений как значений. Выполняется итерация по каждому символу строки, если символ встречается впервые, он добавляется в **HashMap** с количеством 1, если символ уже существует в карте, его счётчик увеличивается.Возвращается карта с символами и их частотой.
* **void visualizeFrequency**:  
  Функция для визуального представления частотного распределения символов. На вход получает частотное распределение в виде карты (**Map<Character, Integer>).** Для каждого символа отображает символ, его количество и соответствующее количество звёздочек (\*). Звёздочки помогают визуально понять, насколько часто встречается символ. Вывод упорядочен для наглядности.
* **void generateCombinations**:  
  Функция для подбора строки, соответствующей заданному паролю, выполняющая полный перебор всех возможных комбинаций. Создаётся массив индексов для символьной строки. Эти индексы используются как "система счисления" с основанием, равным длине алфавита.Подсчитывается общее количество возможных комбинаций как **alphabetSize^passwordLength**. Фиксируется стартовое время подбора с помощью **System.nanoTime()**. Итеративный перебор всех возможных комбинаций символов, текущая комбинация формируется с использованием индексов и алфавита, выполняется проверка: если текущая комбинация совпадает с паролем — процесс завершён. Вычисляются время окончания и разница времени, если комбинация не совпадает, индексы обновляются (аналог инкремента числа в заданной системе счисления), при успешном подборе выводятся: найденный пароль и время, потраченное на подбор.

**Результат программы:**

**Задание 1.1:**

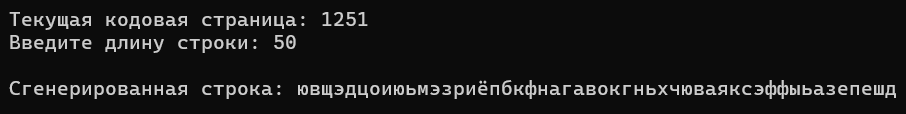


Рисунок 1. Генерация пароля

**Задание 1.2:**

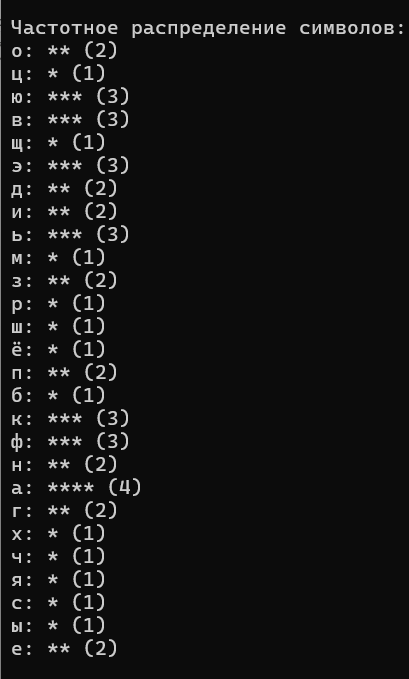


Рисунок 2. Вывод частотного распределения символов

**Задание 1.3:**

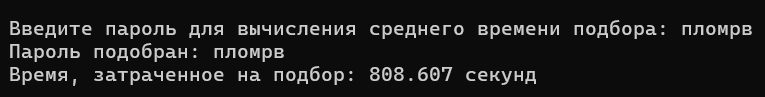


Рисунок 3. Подбор пароля и определение времени подбора

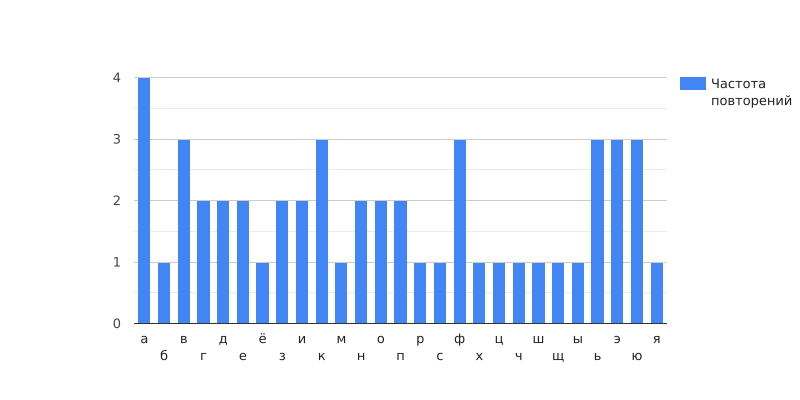


Рисунок 4. График частотного распределения символов в пароле

**Задание 2:**

Рисунок 5. График среднего времени подбора пароля

**Задание 3:**

Если ваш пароль ограничен в символах и может содержать только строчные символы русского алфавита, то он будет существенно ограничиваться, а также будет менее стойким к различным атакам, т. к. имеет всего 33 символа. В данном случае основной рекомендацией для создания пароля только из строчных русских букв будет увеличение длины пароля, поскольку:

* Пароль длиной 8 символов будет иметь:

комбинаций;

* Пароль длиной 12 символов будет иметь:

комбинаций.

Также при составлении пароля рекомендуется начальные и конечные символы пароля выбирать из середины алфавита, поскольку это минимизирует шанс подбора пароля перебором как с начала алфавита, так и с конца, что стабилизирует ваш пароль относительно атак. Старайтесь использовать минимум 12 символов для составления вашего пароля. Если ваша информация обладает высокой ценностью, рекомендуется использовать минимум 16 символов для составления пароля. Рекомендуется использовать генераторы случайных паролей для создания сложных и безопасных комбинаций. Недопустимо использовать личные данные, такие как дата рождения, ФИО, или простые слова, которые могут быть легко угаданы или взломаны.

Если ваш пароль может включать прописные буквы, буквы других алфавитов, а также цифры и различные другие символы, то рекомендуется одновременно различные символы для усложнения пароля.

**Вывод:**

В результате выполнения лабораторной работы была разработана программа на языке программирования С++, которая может генерировать случайный пароль определенной длины из строчных символов русского алфавита. Также данная программа способна отобразить частотное распределение символов в пароле. Еще одной функцией программы является подбор пароля и определение среднего времени подбора пароля.  
 Данная программа может быть полезной, поскольку она может быть использована как генератор случайных паролей. А также она помогает определить среднее время подбора пароля, на основе чего можно создать статистику защиты паролей.

В результате тестирования данной программы можно заметить, что среднее время на подборку пароля увеличивается при увеличении длины пароля.

Для теоретического определения времени подбора пароля используется формула:

Где:

* Т – время подбора пароля;
* R – скорость подбора (количество проверяемых паролей в секунду);
* А – размер алфавита (количество возможных символов, которые могут быть использованы в пароле);
* L – длина пароля.

В данной лабораторной работе были изучены такие понятия, как субъект доступа, объект доступа, а также идентификация и аутентификация.

Идентификация обеспечивает функции, а именно:

* установление подлинности и определение полномочий субъекта при его входе в систему;
* контроль установленных полномочий в процессе сеанса работы;
* регистрация и учет действий.

Аутентификация обычно выделяет три фактора:

* Аутентификация с использованием паролей – наиболее распространенный и привычный метод. Обычно имеют одноразовые пароли и многоразовые. Большинство устройств имеют многоразовые пароли. Однако банки для повышения безопасности чаще используют одноразовые пароли, которые после использования обновляются
* Аутентификация с использованием уникальных предметов – метод, предполагающий предъявление субъектов различных предметов (ключ от замка, файл данных и тд.)
* Аутентификация с использованием биометрических данных – способ, при котором свойствами субъекта для аутентификации являются лицо, рисунок радужной оболочки глаз, отпечаток пальцев и тд.

Однако ни один метод не является безопасным на 100%, т. к. имеет свои недостатки. Аутентификация с использованием паролей может быть перехвачена атакующим и использована для несанкционированного получения прав доступа. Аутентификация с использованием уникальных предметов также не является полностью безопасной, поскольку существует риск потери определенного предмета, выхода его из строя, а также кражи. Аутентификация с использованием биометрических данных обладает особой уникальностью, однако также не имеет 100% безопасности, поскольку возможны ошибки распознавания данных, а также шанс подделки.