Analiza Haseł

Zasady zaliczenia i wybór oceny

Oceny zdobywa się schodkowo (aby zaliczyć poziom wyższy, musisz mieć zrobione poziomy niższe):

Ocena	Wymagany zakres zadań	Opis poziomu
2	Poziom 2 (Zadania 2.1 – 2.2)	Wczytanie pliku, badanie długości napisów
3	Poziom 2 + Poziom 3 (Zadania 3.1 – 3.3)	Min/max długości, filtrowanie tylko cyfr, średnia
4	Poziomy 2, 3 + Poziom 4 (Zadania 4.1 – 4.2)	Polityka silnego hasła oraz hasła-palindromy
5	Poziomy 2, 3, 4 + Poziom 5 (Zadania 5.1 – 5.2)	Wykrywanie potrójnych powtórzeń i ciągów znaków w ASCII
6	Wszystkie poziomy + Poziom 6 (Zadania 6.1 – 6.2)	Wyszukiwanie anagramów i analiza statystyczna prefiksów

Dane do zadania

W pliku `hasla.txt` znajduje się lista haseł użytkowników (po jednym haśle w każdym wierszu). Hasła składają się wyłącznie z liter alfabetu łacińskiego (`a-z`, `A-Z`) oraz cyfr (`0-9`).

```
12345
Kajak
pass
SuperTajneHaslo2024
kotek
tokek
987654321
aaa123
Abcdef1
SekretnaBaza999
Abcd1234
radar
Adm007
Adm99
Admin123
```

TREŚĆ ZADAŃ

- **POZIOM 2 (Ocena: Dopuszczający)**
  Cel: Wczytanie danych tekstowych i podstawowe filtrowanie długości.
  - **Zadanie 2.1:** Podaj, ile łącznie haseł znajduje się w pliku.
  - **Zadanie 2.2:** Zlicz:
    1. ile haseł jest krótkich (posiada mniej niż 8 znaków),
    2. ile haseł spełnia rekomendację minimalnej długości (posiada co najmniej 8 znaków).
- **POZIOM 3 (Ocena: Dostateczny)**
  Cel: Wyszukiwanie wartości skrajnych, średnia i weryfikacja typów znaków.
  - **Zadanie 3.1**: Znajdź długość najkrótszego oraz najdłuższego hasła w pliku. Wypisz pierwsze napotkane hasło o najmniejszej długości oraz pierwsze o największej.
  - **Zadanie 3.2**: Ile haseł w pliku składa się wyłącznie z cyfr (tzw. kody PIN)?
  - **Zadanie 3.3:** Oblicz średnią długość hasła w pliku (wynik podaj z zaokrągleniem do 2 miejsc po przecinku).
- **POZIOM 4 (Ocena: Dobry)**
  Cel: Wielowarunkowe badanie struktury tekstu i operacje lustrzane.
  - **Zadanie 4.1 (Silne hasła):**
    Hasło uznajemy za złożone, jeśli spełnia jednocześnie 4 warunki:
    1. Ma co najmniej 8 znaków,
    2. Zawiera co najmniej jedną cyfrę (0-9),
    3. Zawiera co najmniej jedną małą literę (a-z),
    4. Zawiera co najmniej jedną wielką literę (A-Z).

    Ile takich złożonych haseł znajduje się w pliku?
  - **Zadanie 4.2 (Palindromy):**
    Podaj, ile haseł w pliku o długości co najmniej 2 to palindromy (hasło czyta się tak samo od początku i od końca, bez rozróżniania wielkości liter, np. Kajak, radar, 12321). Wypisz pierwsze trzy takie hasła.
- **POZIOM 5 (Ocena: Bardzo dobry)**
  Analiza sąsiedztwa znaków, sekwencje i kody ASCII.
  - **Zadanie 5.1 (Zabronione powtórzenia):**
    Dobra polityka bezpieczeństwa zabrania wpisywania tego samego znaku wielokrotnie obok siebie. Znajdź hasła, w których ten sam znak występuje bezpośrednio po sobie co najmniej 3 razy (np. aaa, 999, BBB).
    Ile jest takich haseł w pliku? Wypisz pierwsze dwa z nich.

- **Zadanie 5.2 (Rosnące sekwencje ASCII):**
  Podatne hasła często zawierają ciągi kolejnych znaków klawiatury/alfabetu. Znajdź hasła, które zawierają w sobie spójny fragment złożony z co najmniej 4 kolejnych znaków rosnących w kodzie ASCII (gdzie każdy kolejny znak ma kod o dokładnie 1 większy od poprzedniego, np. 1234, bcde czy 5678).
  Ile haseł posiada taki fragment? Wypisz pierwsze napotkane hasło i wskaż znaleziony 4-znakowy fragment.

- **POZIOM 6 (Ocena: Celujący)**

  Algorytmy słownikowe, zliczanie prefiksów oraz detekcja permutacji (anagramów).

  - **Zadanie 6.1 (Pary anagramów):**
    Dwa hasła są anagramami, jeśli składają się z dokładnie takiego samego zestawu liter i cyfr w innej kolejności (np. kotek i tokek, Koder2026 i 2026rekoD), ignorując wielkość liter.
    Ile unikalnych par haseł będących swoimi anagramami znajduje się w pliku? Wypisz przynajmniej jedną taką parę.
  - **Zadanie 6.2 (Najczęstszy prefiks 3-znakowy):**
    Pobierz pierwsze 3 znaki każdego hasła (dla haseł o długości co najmniej 3). Jaki 3-znakowy początek występuje najczęściej w pliku i ile dokładnie razy się pojawił? (Wielkość liter ma znaczenie, np. Adm to nie to samo co adm).

JAK SIĘ ZA TO ZABRAĆ?

Wskazówki algorytmiczne:

- **Tylko cyfry (Zadanie 3.2):**
  - Python: `haslo.isdigit()`
  - JS: `/^\d+$/.test(haslo)` lub sprawdzanie każdego znaku `c >= '0' && c <= '9'`
  - C++: `all_of(s.begin(), s.end(), ::isdigit)`
- **Złożone hasło (Zadanie 4.1):**
  Użyj trzech zmiennych logicznych (flag): `ma_cyfre = false`, `ma_mala = false`, `ma_wielka = false`. Przejdź pętlą po znakach hasła i przestawiaj flagi na `true`. Na koniec sprawdź: `dlugosc >= 8 && ma_cyfre && ma_mala && ma_wielka`.
- **Palindrom case-insensitive (Zadanie 4.2):**
  Przed porównaniem zamień cały napis na małe litery!
  - Python: `s = haslo.lower(); if len(s) >= 2 and s == s[::-1]: ...`
  - JS: const s = `haslo.toLowerCase(); if (s.length >= 2 && s === s.split('').reverse().join('')) ...`
- **Trzy jednakowe znaki z rzędu (Zadanie 5.1):**
  W pętli `od i = 0 do dlugosc - 3`:
  Sprawdzaj warunek: `haslo[i] == haslo[i+1] && haslo[i+1] == haslo[i+2]`. Jeśli prawda → zlicz hasło i przerwij sprawdzanie tego hasła (`break`).
- **Rosnące kody ASCII (Zadanie 5.2):**
  W pętli `od i = 0 do dlugosc - 4`:
  Sprawdzaj, czy kody ASCII kolejnych znaków różnią się o 1:
  - Python: `ord(s[i+1]) == ord(s[i]) + 1 and ord(s[i+2]) == ord(s[i+1]) + 1 ...`
  - C++ / JS: `s[i+1] == s[i] + 1 && s[i+2] == s[i+1] + 1 && s[i+3] == s[i+2] + 1`
- **Anagramy (Zadanie 6.1):**
  Dwa słowa są anagramami, jeśli po posortowaniu ich liter są identyczne!
  - Python: `klucz = "".join(sorted(haslo.lower()))`
  - JS: const `klucz = haslo.toLowerCase().split('').sort().join('');`
  - Użyj słownika/mapy (obiektu), aby grupować hasła o tym samym kluczu.

Kod startowy

Python

```python
with open("hasla_przyklad.txt", "r", encoding="utf-8") as f:
    hasla = [linia.strip() for linia in f if linia.strip()]

print(f"Wczytano {len(hasla)} haseł.")
# Tutaj pisz swój kod...
```

C++

```cpp
#include <iostream>
#include <fstream>
#include <vector>
#include <string>

using namespace std;

int main() {
    ifstream plik("hasla_przyklad.txt");
    vector<string> hasla;
    string h;

    while (plik >> h) {
        hasla.push_back(h);
    }
    plik.close();

    cout << "Wczytano " << hasla.size() << " hasel." << endl;
    // Tutaj pisz swój kod...
    return 0;
}
```