ЗВІТ З ЛАБОРАТОРНОЇ РОБОТИ № 1

за курсом “Мовні технології”

студент(а/ки) групи ПЗ-24у-1

Кондрачук Олександр Вадимович

2024/2025 н.р.

1. **Постановка задачі**

Варіант 8

Знайти у тексті до 15 слів, у складі яких є найбільша послідовність приголосних літер. Результат виконання програми: вивести знайдені слова або повідомлення, якщо у тексті немає жодного слова, у складі якого є послідовності приголосних.

1. **Опис розв’язку**

Код є програмою на C++, яка аналізує текстовий файл, підраховує кількість послідовних приголосних у слові і виводить результати в консоль і файл.

1. Функція `split\_words` поділяє текст на слова, використовуючи функцію std::isalpha для поділу за пробілами, комами, точками та іншими знаками пунктуації.

2. Функція `read` читає вміст файлу та повертає його у вигляді рядка.

3. Функція `write` записує рядок у вказаний файл.

4. У функції `main` програма зчитує текст із файлу "file\_in.txt", поділяє його на слова, підраховує кількість послідовних приголосних, а потім:

- Якщо текст є порожнім, виводить "File has no words.".

- Інакше виводить топ-15 слів з найбільшою кількістю послідовних приголосних.

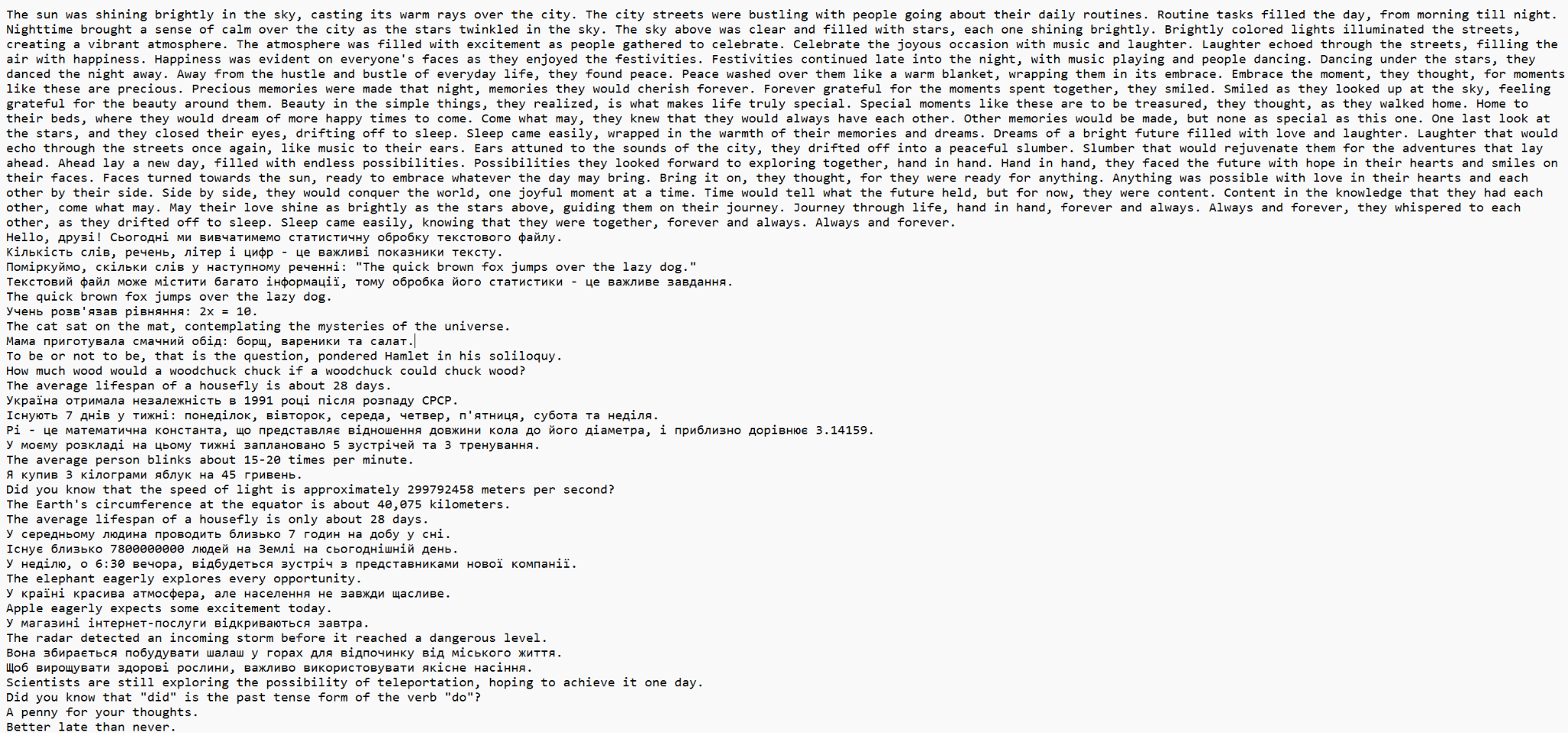
Програма також записує результати у файл "output.txt".

1. **Опис інтерфейсу**

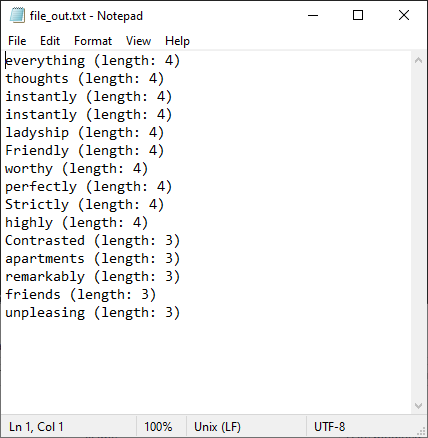
Користувач повинен підготувати файл з текстом та далі отримує файл з результатами.

1. **Тестування програми**

вхідний файл:

****

файл з результатами:

****

1. **Аналіз помилок**

немає

1. **Додаток**

#include <iostream>

#include <fstream>

#include <sstream>

#include <string>

#include <vector>

#include <locale>

#include <algorithm>

#include <windows.h>

const std::string InputFile = "file\_in.txt";

const std::string OutputFile = "file\_out.txt";

const std::string vowels = "aeiouAEIOU";

std::string read() {

std::ifstream file(InputFile, std::ios::binary);

if (!file.is\_open())

{

return "";

}

std::ostringstream buffer;

buffer << file.rdbuf();

std::string text = buffer.str();

return text;

}

void write(const std::string& data) {

std::ofstream file(OutputFile, std::ios::binary);

if (!file.is\_open())

{

return;

}

file.write(data.c\_str(), data.size());

}

bool is\_vowel(const char character) {

return vowels.find(character) != std::string::npos;

}

std::vector<std::string> split\_words(const std::string& text)

{

std::vector<std::string> words;

std::string current;

for (const char character : text)

{

if (std::isalpha(character))

{

current += character;

}

else if (!current.empty())

{

words.push\_back(current);

current.clear();

}

}

if (!current.empty())

{

words.push\_back(current);

}

return words;

}

int max\_consonant\_sequence(const std::string& word)

{

int max\_length = 0;

int current\_length = 0;

for (const char character : word)

{

if (!is\_vowel(character))

{

current\_length++;

max\_length = max(max\_length, current\_length);

}

else

{

current\_length = 0;

}

}

return max\_length;

}

int main()

{

const std::string text = read();

if (text.empty())

{

std::cout << "Can't read file.\n";

return 1;

}

auto words = split\_words(text);

if (words.empty())

{

std::cout << "File has no words.\n";

return 0;

}

std::vector<std::pair<std::string, int>> results;

for (const auto& word : words)

{

int length = max\_consonant\_sequence(word);

if (length > 0)

{

results.emplace\_back(word, length);

}

}

if (results.empty())

{

std::cout << "There are no needed words\n";

return 0;

}

std::sort(results.begin(), results.end(), [](const auto& a, const auto& b)

{

return a.second > b.second;

});

std::string result = "";

int count = 0;

for (const auto& data : results)

{

count++;

if (count > 15)

{

break;

}

result += data.first + " (length: " + std::to\_string(data.second) + ")\n";

}

std::cout << result;

write(result);

return 0;}