ЗВІТ З ЛАБОРАТОРНОЇ РОБОТИ № 2

за курсом “Мовні технології”

студент(а/ки) групи ПЗ-24у-1

Кондрачук Олександр Вадимович

2024/2025 н.р.

1. **Постановка задачі**

Варіант 8

1. Реалізувати програму на С++, яка виконує попереднє опрацювання введеного користувачем тексту. Текст вводиться з клавіатури або зчитується з файлу input.txt.

2. Основний функціонал програми повинен включати:

* Зчитування тексту з файлу або введення користувачем.
* Перетворення тексту у нижній регістр.
* Видалення розділових знаків і спеціальних символів. Видалення зайвих пробілів.
* Токенізацію тексту (розбиття на слова).
* Видалення стоп-слів (список слів задається у файлі або в коді).
* Виведення результату (списку токенів) у консоль або запис у файл.

1. **Опис розв’язку**

Код є програмою на C++, яка аналізує текстовий файл, опрацьовує текст, розбиває його на слова і виводить результати в консоль і файл.

1. Функція `transformString` попередньо опрацьовує текст (видаляє розділові знаки та зайві пробіли, переводить усі символи у нижній регістр та позбувається стоп-слів).

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

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

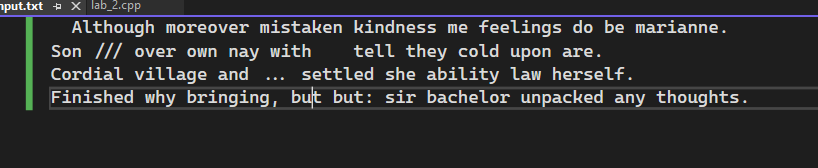
4. У функції `main` програма зчитує текст із файлу "input.txt", опрацьовує його, поділяю на слова, підраховує кількість унікальних слів та кількість повторень слів.

**Опис інтерфейсу**

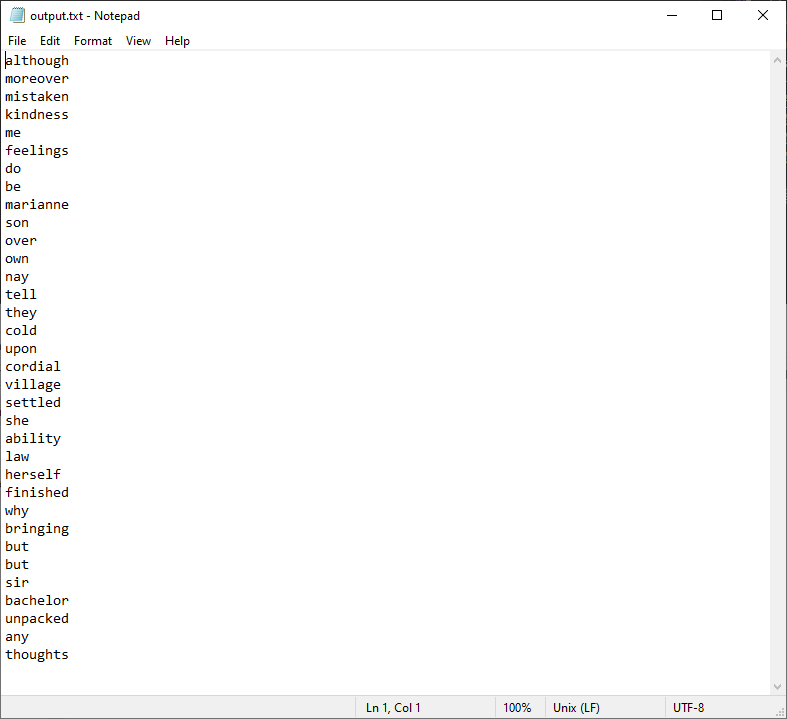
Користувач повинен підготувати файл з текстом та далі отримує файл з результатами.

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 <cctype>

#include <map>

#include <set>

const std::string InputFile = "input.txt";

const std::string OutputFile = "output.txt";

const std::string StopWordsFile = "stop\_words.txt";

std::string read(const std::string& filename)

{

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

if (!file.is\_open()) {

std::cerr << "Failed to open file: " << filename << '\n';

return "";

}

std::ostringstream buffer;

buffer << file.rdbuf();

return buffer.str();

}

void write(const std::vector<std::string>& tokens)

{

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

if (!file.is\_open()) {

std::cerr << "Failed to open file for writing: " << OutputFile << '\n';

return;

}

for (const auto& token : tokens) {

file << token << "\n";

}

}

std::string readText()

{

return read(InputFile);

}

std::set<std::string> prepareStopWords()

{

return {

"and", "the", "is", "are", "a", "an",

"of", "in", "to", "it", "this",

"that", "for", "on", "with", "as",

"at", "by"

};

}

std::vector<std::string> tokenize(const std::string& text)

{

std::istringstream iss(text);

std::vector<std::string> tokens;

std::string word;

while (iss >> word)

{

tokens.push\_back(word);

}

return tokens;

}

std::set<std::string> getUniqueWords(const std::vector<std::string>& tokens)

{

std::set<std::string> uniqueWords(tokens.begin(), tokens.end());

return uniqueWords;

}

std::map<std::string, int> getWordFrequencies(const std::vector<std::string>& tokens)

{

std::map<std::string, int> frequency;

for (const auto& token : tokens)

{

frequency[token]++;

}

return frequency;

}

char transformChar(const char original)

{

if (!isalpha(original))

{

return ' ';

}

return std::tolower(original);

}

void removeWord(std::string& text, std::string& word, int wordStart)

{

for (int i = 0; i < word.length(); i++)

{

text[i + wordStart] = ' ';

}

}

void removeStopWords(std::string& text, const std::set<std::string>& stopWords)

{

int wordStart = -1;

for (int i = 0; i < text.length(); i++)

{

char c = text[i];

if (c != ' ')

{

wordStart = wordStart == -1 ? i : wordStart;

continue;

}

if (wordStart == -1)

{

continue;

}

std::string word(text, wordStart, i - wordStart);

if (stopWords.find(word) != stopWords.end())

{

removeWord(text, word, wordStart);

}

wordStart = -1;

}

}

std::string transformString(std::string& text)

{

for (int i = 0; i < text.length(); i++)

{

text[i] = transformChar(text[i]);

}

removeStopWords(text, prepareStopWords());

return text;

}

int main()

{

std::string text = read(InputFile);

if (text.empty())

{

std::cerr << "Input text is empty." << std::endl;

return 1;

}

text = transformString(text);

std::vector<std::string> tokens = tokenize(text);

std::cout << "Processed Tokens:\n";

for (const auto& token : tokens)

{

std::cout << token << " ";

}

std::cout << "\n\n";

write(tokens);

std::set<std::string> uniqueWords = getUniqueWords(tokens);

std::cout << "Number of unique words: " << uniqueWords.size() << "\n\n";

std::map<std::string, int> wordFrequency = getWordFrequencies(tokens);

std::cout << "Word Frequencies:\n";

for (const auto& pair : wordFrequency)

{

std::cout << pair.first << ": " << pair.second << "\n";

}

return 0;

}