ГУАП

КАФЕДРА № 42

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| доцент |  |  |  | В. А. Кузнецов |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ № 2.2 |
| АНАЛИЗ СТРОК, ХЕШИРОВАНИЕ |
| по курсу: |
| ОСНОВЫ ПРОГРАММИРОВАНИЯ |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ гр. № | 4326 |  |  |  | Г. С. Томчук |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2024

СОДЕРЖАНИЕ

[1 Постановка задачи 3](#__RefHeading___Toc801_316780651)

[2 Схема алгоритма решения 4](#__RefHeading___Toc809_316780651)

[3 Полное описание реализованных функций 5](#__RefHeading___Toc807_316780651)

[3.1 split\_words 5](#__RefHeading___Toc3907_3112333877)

[3.2 find\_indices 5](#__RefHeading___Toc3913_3112333877)

[3.3 transform\_to\_lower 6](#__RefHeading___Toc3911_3112333877)

[3.4 main 6](#__RefHeading___Toc3909_3112333877)

[4 Листинг программы 7](#__RefHeading___Toc805_316780651)

[5 Результаты тестирования программы 9](#__RefHeading___Toc803_316780651)

1. Постановка задачи

Задача: реализовать программную функцию на языке C/С++, выполняющую поставленную задачу. Вариант задания, пример входных и выходных данных представлен в таблице 1. Глобальные параметры использовать запрещено; допустимо использование дополнительных функций.

Таблица 1 – Вариант

|  |  |  |  |
| --- | --- | --- | --- |
| N | Текст задания | Вход | Выход |
| 6 | **Антиплагиат**  Даны две строки S1, S2. Разделителем между словами в строке может являться пробел « », запятая «,» или перенос строки «\n». Найти все повторяющиеся в обеих строках слова без учета регистра символов (“Антиплагиат” = “АнТиплагиаТ”). Для каждого повторяющегося слова определить все индексы его вхождения в строку S1 и строку S2. | S1 :  “Get now to get first”  S2 :  “I will get back to you” | “get” :  [0, 11],  [7] |

1. Схема алгоритма решения

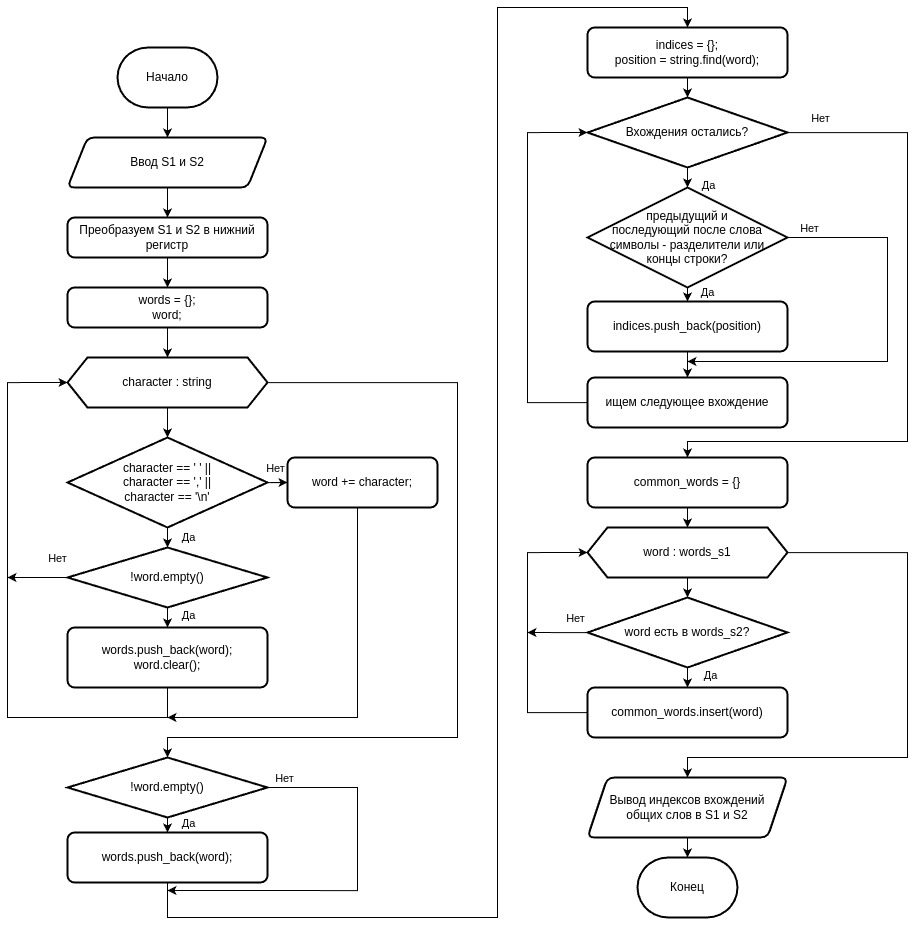


Рисунок 1 – Блок-схема алгоритма

1. Полное описание реализованных функций
   1. split\_words

Функция split\_words разделяет строку на слова, используя пробелы, запятые и переносы строк в качестве разделителей. Принимает следующие аргументы:

1. const std::string &string — строка, которую нужно разделить на слова.

Возвращает std::vector<std::string> — вектор строк, каждая из которых является словом из исходной строки. Работа функции происходит следующим образом:

1. Создаётся пустой вектор words и временная строка word.
2. Для каждого символа в строке проверяется, является ли он разделителем (пробел, запятая, перенос строки).
3. Если символ является разделителем и word не пуст, добавляем word в words и очищаем word.
4. Если символ не является разделителем, добавляем его к word.
5. В конце добавляем последнее слово в words, если оно не пустое.
   1. find\_indices

Функция find\_indices находит все индексы вхождения слова в строку. Принимает следующие аргументы:

1. const std::string &string — строка, в которой ищем вхождения слова.
2. const std::string &word — слово, индексы вхождения которого ищем.

Возвращает std::vector<int> — вектор индексов, где слово встречается в строке. Работа функции происходит следующим образом

1. Создаётся пустой вектор indices.
2. Используем string.find для поиска первого вхождения слова в строке.
3. Пока находятся вхождения, проверяем, является ли предыдущий и последующий после слова символы разделителями или концом строки.
4. Если условие выполняется, добавляем индекс начала слова в indices.
5. Ищем следующее вхождение, начиная с позиции position + 1.
   1. transform\_to\_lower

Функция transform\_to\_lower преобразует все символы строки в нижний регистр. Принимает следующие аргументы:

1. const std::string &string — строка, которую нужно преобразовать.

Возвращает std::string — строка, преобразованная в нижний регистр. Работа функции происходит следующим образом:

1. Копируем исходную строку в новую строку lower.
2. Используем std::transform для преобразования каждого символа строки в нижний регистр с помощью функции ::tolower.
   1. main
3. Программа запрашивает ввод строк S1 и S2 от пользователя.
4. Преобразуем S1 и S2 в нижний регистр с помощью transform\_to\_lower.
5. Разделяем S1 и S2 на слова с помощью функции split\_words.
6. Для каждого слова в S1 и S2 находим индексы вхождений в исходные строки, используя функцию find\_indices.
7. Находим слова, которые есть в обеих строках, используя unordered\_set.
8. Для каждого общего слова выводим индексы их вхождений в S1 и S2.
9. Листинг программы

Листинг 1

#include <iostream>  
#include <vector>  
#include <unordered\_map>  
#include <unordered\_set>  
#include <algorithm>  
  
std::vector<std::string> split\_words(const std::string &string) {  
 std::vector<std::string> words;  
 std::string word;  
 for (char character : string) {  
 if (character == ' ' || character == ',' || character == '\n') {  
 if (!word.empty()) {  
 words.push\_back(word);  
 word.clear();  
 }  
 } else {  
 word += character;  
 }  
 }  
 if (!word.empty()) { // Добавляем последнее слово  
 words.push\_back(word);  
 }  
 return words;  
}  
  
std::vector<int> find\_indices(const std::string &string, const std::string &word) {  
 std::vector<int> indices;  
 size\_t position = string.find(word); // Находим первое вхождение  
 while (position != std::string::npos) {  
 char start\_character = string[position - 1];  
 char end\_character = string[position + word.size()];  
 // Проверяем начало и окончание слова  
 if (end\_character == ' ' || end\_character == ',' || end\_character == '\n' || end\_character == '\0')  
 if (start\_character == ' ' || start\_character == ',' || start\_character == '\n' || start\_character == '\0')  
 indices.push\_back((int) position); // Добавляем индекс в вектор  
 position = string.find(word, position + 1); // Ищем следующее вхождение, начиная с позиции position + 1  
 }  
 return indices;  
}  
  
std::string transform\_to\_lower(const std::string &string) {  
 std::string lower = string;  
 std::transform(lower.begin(), lower.end(), lower.begin(), ::tolower);  
 return lower;  
}  
  
int main() {  
 std::string S1;  
 std::string S2;

Продолжение листинга 1

std::cout << "S1: ";  
 std::getline(std::cin >> std::ws, S1);  
  
 std::cout << "S2: ";  
 std::getline(std::cin >> std::ws, S2);  
  
 std::string lower\_s1 = transform\_to\_lower(S1);  
 std::string lower\_s2 = transform\_to\_lower(S2);  
  
 std::unordered\_map<std::string, std::vector<int>> word\_indices\_s1;  
 std::unordered\_map<std::string, std::vector<int>> word\_indices\_s2;  
  
 std::vector<std::string> words\_s1 = split\_words(S1);  
 std::vector<std::string> words\_s2 = split\_words(S2);  
  
 // Находим индексы  
 for (const std::string &word : words\_s1) {  
 std::string lower\_word = transform\_to\_lower(word);  
 word\_indices\_s1[lower\_word] = find\_indices(lower\_s1, lower\_word);  
 }  
 for (const std::string &word : words\_s2) {  
 std::string lower\_word = transform\_to\_lower(word);  
 word\_indices\_s2[lower\_word] = find\_indices(lower\_s2, lower\_word);  
 }  
  
 // Находим общие слова  
 std::unordered\_set<std::string> common\_words;  
 for (const auto &pair : word\_indices\_s1) {  
 if (word\_indices\_s2.find(pair.first) != word\_indices\_s2.end()) {  
 common\_words.insert(pair.first);  
 }  
 }  
  
 // Выводим индексы вхождений  
 for (const std::string &word : common\_words) {  
 std::vector<int> current\_indices\_s1 = word\_indices\_s1[word];  
 std::vector<int> current\_indices\_s2 = word\_indices\_s2[word];  
  
 std::cout << "\"" << word << "\": ";  
 std::cout << "[";  
 for (size\_t i = 0; i < current\_indices\_s1.size(); i++) {  
 std::cout << current\_indices\_s1[i];  
 if (i < current\_indices\_s1.size() - 1) std::cout << ", ";  
 }  
 std::cout << "], [";  
 for (size\_t i = 0; i < current\_indices\_s2.size(); i++) {  
 std::cout << current\_indices\_s2[i];  
 if (i < current\_indices\_s2.size() - 1) std::cout << ", ";  
 }  
 std::cout << "]" << std::endl;  
 }  
  
 return 0;  
}

1. Результаты тестирования программы

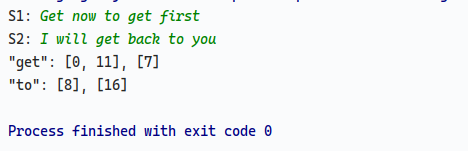


Рисунок 2

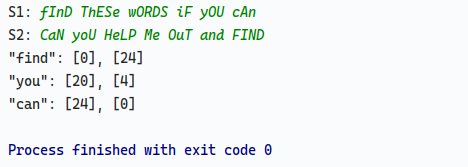


Рисунок 3

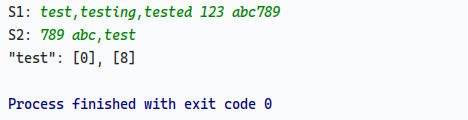


Рисунок 4

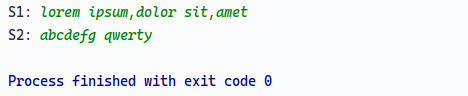


Рисунок 5

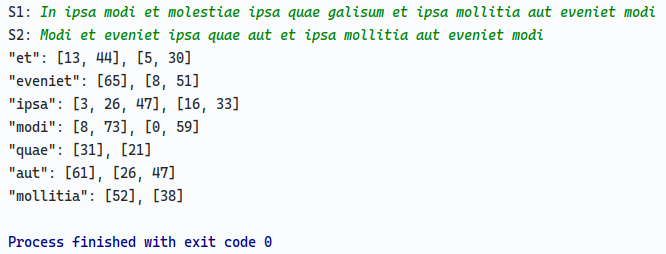


Рисунок 6