**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра информационных систем**

отчет

**по практической работе № 4**

**по дисциплине «Программирование»**

Тема: «**Работа с текстом»**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 3373 |  | Огородникова В.С. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2023

**Цель работы.**

Изучение и организация структур; получение практических навыков работы со структурами; определение преимуществ и недостатков использования структур.

**Основные теоретические положения.**

### Определение текстовой строки

* Текстовые строки представляются с помощью одномерных массивов символов. В языке C++ текстовая строка представляет собой набор символов, обязательно заканчивающийся нулевым символом (‘\0’). Поэтому, если вы хотите создать текстовый массив для хранения 10 (N) символов, нужно выделить память под 11(N+1) символов.
* Объявленный таким образом массив может использоваться для хранения текстовых строк, содержащих не более 10 символов. Нулевой символ позволяет определить границу между содержащимся в строке текстом и неиспользованной частью строки.
* При определении строковых переменных их можно инициализировать конкретными значениями с помощью строковых литералов:

char S1[15] = “This is text”;

char S2[] = “Пример текста”;

* Последние два элемента переменной  просто не используются, а строка  автоматически подстраивается под длину инициализирующего текста.
* При работе со строками можно обращаться к отдельным символам строки как в обычном одномерном массиве с помощью индексов:

cout << S1[0]; // На экране будет выведен символ ‘T’

* Если строка формируется при помощи цикла (или иного способа), то необходимо в ее конец обязательно записать нулевой символ '\0'.
* Класс string предназначен для работы со строками типа char, которые представляют собой строчку с завершающим нулем (символ ‘\0’). Класс string был введен как альтернативный вариант для работы со строками типа char.
* Чтобы использовать возможности класса string, нужно подключить библиотеку <string> и пространство имен std. Объявление же переменной типа string осуществляется схоже с обычной переменной:
* string S1; // Переменная с именем s1 типа string
* string S2 = “Пример”; // объявление с инициализацией
* Создание нового типа string было обусловлено недостатками работы с строками символов, который показывал тип char. В сравнении с этим типом string имеет ряд основных преимуществ:
* ·        возможность использования для обработки строк стандартные операторы С++(=,+,<,==,>,+=,!=,<=,>=,[])(=,+,<,==,>,+=,!=,<=,>=,[]). Использование типа char приводило требовало написание чрезмерного программного кода;
* ·        обеспечение лучшей надежности программного кода;
* ·        обеспечение строки, как самостоятельного типа данных.

**Выполнение работы.**

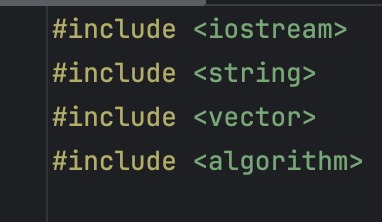


Рис.1. – подключение библиотек

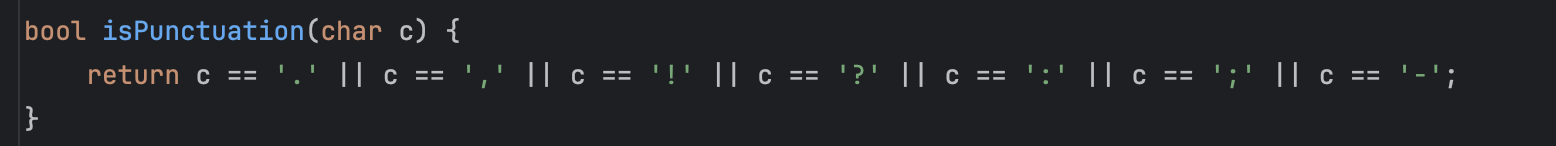


Рис.2. – функция.

Эта функция isPunctuation проверяет, является ли переданный символ знаком пунктуации. Она возвращает true, если символ является одним из знаков пунктуации

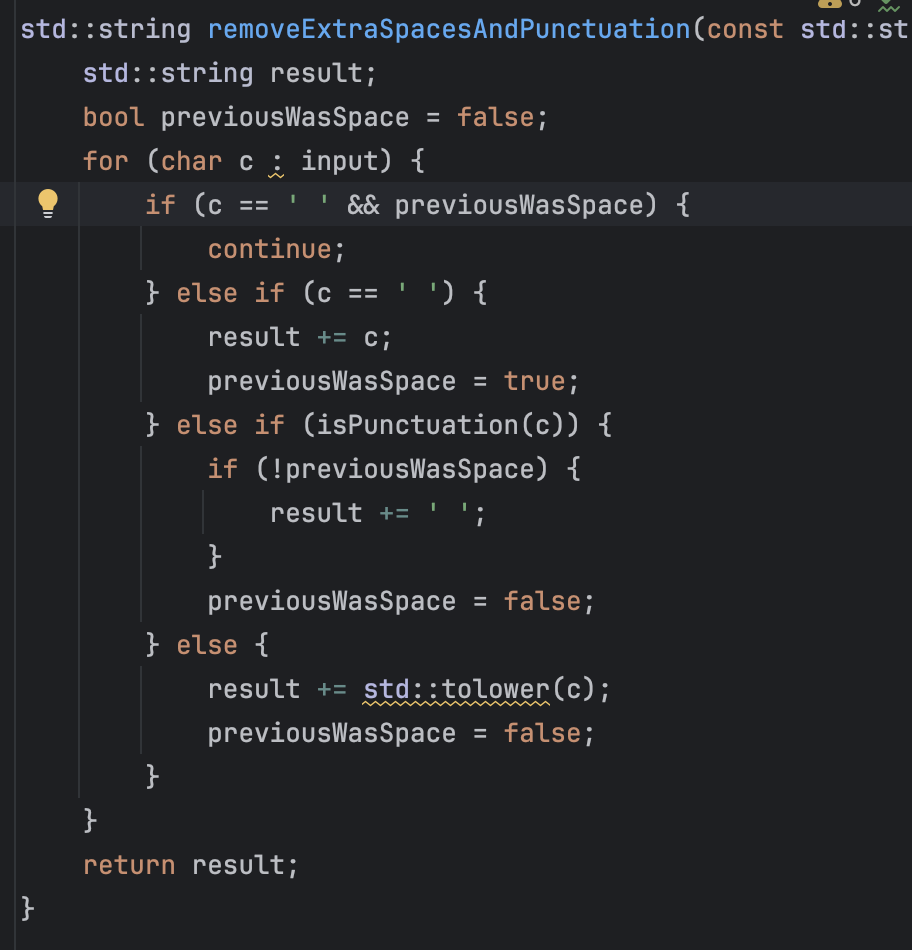


Рис.3. –основной код

Эта функция removeExtraSpacesAndPunctuation принимает строку в качестве аргумента и возвращает новую строку без лишних пробелов и знаков пунктуации. Она также преобразует все символы в нижний регистр. Она использует функцию isPunctuation для проверки наличия знаков пунктуации.

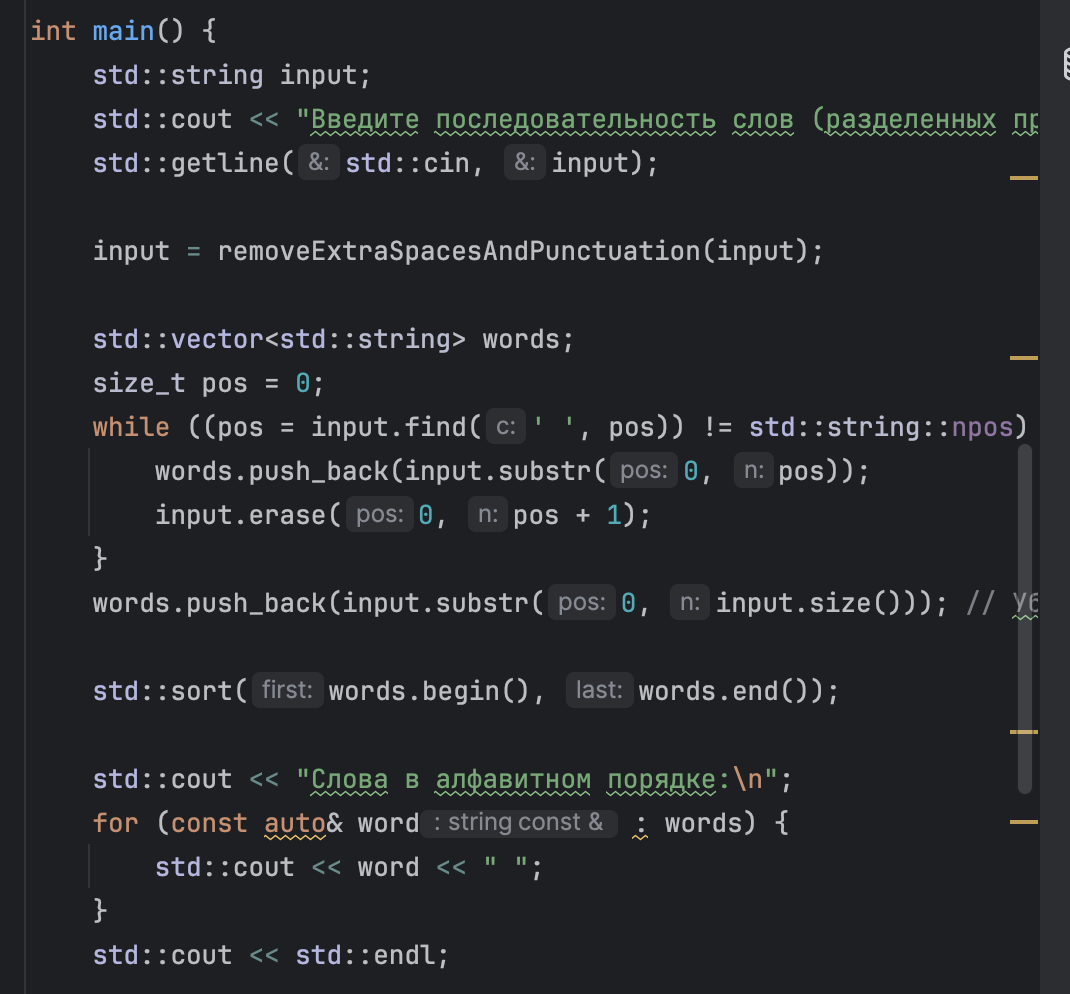


Рис.4. – запуск и вывод

Рабочий код А:

#include <iostream>  
#include <string>  
#include <vector>  
#include <algorithm>  
  
bool isPunctuation(char c) {  
 return c == '.' || c == ',' || c == '!' || c == '?' || c == ':' || c == ';' || c == '-';  
}  
  
std::string removeExtraSpacesAndPunctuation(const std::string& input) {  
 std::string result;  
 bool previousWasSpace = false;  
 for (char c : input) {  
 if (c == ' ' && previousWasSpace) {  
 continue;  
 } else if (c == ' ') {  
 result += c;  
 previousWasSpace = true;  
 } else if (isPunctuation(c)) {  
 if (!previousWasSpace) {  
 result += ' ';  
 }  
 previousWasSpace = false;  
 } else {  
 result += std::tolower(c);  
 previousWasSpace = false;  
 }  
 }  
 return result;  
}  
  
int main() {  
 std::string input;  
 std::cout << "Введите последовательность слов (разделенных пробелами, заканчивающихся точкой): ";  
 std::getline(std::cin, input);  
  
 input = removeExtraSpacesAndPunctuation(input);  
  
 std::vector<std::string> words;  
 size\_t pos = 0;  
 while ((pos = input.find(' ', pos)) != std::string::npos) {  
 words.push\_back(input.substr(0, pos));  
 input.erase(0, pos + 1);  
 }  
 words.push\_back(input.substr(0, input.size())); // Убрать точку в конце  
  
 std::sort(words.begin(), words.end());  
  
 std::cout << "Слова в алфавитном порядке:\n";  
 for (const auto& word : words) {  
 std::cout << word << " ";  
 }  
 std::cout << std::endl;  
  
 std::cout << "Слова исходной последовательности:\n";  
 for (const auto& word : words) {  
 std::cout << word << std::endl;  
 }  
  
 return 0;  
}

**Вывод.**

Программа работает исправно

Не исключайте то, что при копировании кода он просто не заработает, может выдавать ошибки.