Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

 Лабораторная работа 13

По дисциплине «Основы алгоритмизации и программирования»

На тему «Обработка символьной информации»

Выполнила:

Студентка 1 курса 7 группы

Бабич Виолетта Станиславовна

Преподаватель: асс. Андронова М.В.

2023, Минск

|  |  |
| --- | --- |
| **2** | 1. Написать программу, реализующую выделение подстроки **S1** длиной **k** с позиции номер **n** из некоторой строки.  2. Дана строка, состоящая из групп нулей и единиц. Определить самую короткую группу. |

№1

#include <iostream> // Включаем заголовочный файл для ввода-вывода данных.

#include <string> // Включаем заголовочный файл для работы со строками.

using namespace std; // Используем пространство имен std (стандартной библиотеки).

string extractSubstring(const string& input, int n, int k) {

// Функция для выделения подстроки из исходной строки.

if (n < 0 || k <= 0 || n >= input.length()) {

// Проверяем, что параметры n и k

return "Некорректные параметры n и k";

// Если параметры некорректны, возвращаем сообщение об ошибке.

}

else if (n + k > input.length()) {

// Проверяем, что подстрока не выходит за границы исходной строки.

return "Подстрока выходит за границы строки";

// Если подстрока выходит за границы, возвращаем сообщение об ошибке.

}

else {

return input.substr(n, k);

// Возвращаем подстроку, начиная с позиции n и длиной k.

}

}

int main() {

setlocale(LC\_ALL, "rus");

string originalString = "Это пример строки, из которой извлекаем подстроку.";

int n = 8; // Номер элемента,с которого начнеться подстрока

int k = 7; // Кол-во элементов в подстроке

string result = extractSubstring(originalString, n, k);//выполняем функцию вычисления подстроки

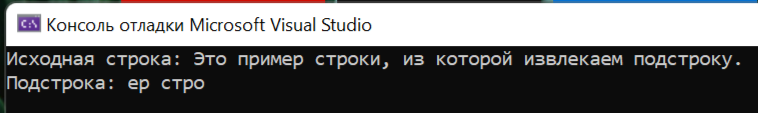
//Вывод результата

cout << "Исходная строка: " << originalString << endl;

cout << "Подстрока: " << result << endl;

return 0;

}



№2

#include <iostream>

#include <string>

using namespace std;

// Функция для поиска самой короткой группы нулей или единиц в строке.

string findShortestGroup(const std::string& input) {

// Создаем строку для хранения самой короткой группы.

string shortestGroup = "";

// Создаем строку для хранения текущей группы.

string currentGroup = "";

// Инициализируем переменную для хранения текущего символа.

char currentChar = input[0];

// Проходим по каждому символу во входной строке.

for (char c : input) {

// Если текущий символ совпадает с предыдущим, добавляем его в текущую группу.

if (c == currentChar) {

currentGroup += c;

}

else {

// Если текущая группа короче, обновляем самую короткую группу.

if (currentGroup.length() < shortestGroup.length() || shortestGroup.empty()) {

shortestGroup = currentGroup;

}

// Начинаем новую текущую группу с текущего символа.

currentGroup = c;

// Обновляем текущий символ.

currentChar = c;

}

}

// После завершения цикла, проверяем текущую группу, чтобы убедиться, что она не является самой короткой.

if (currentGroup.length() < shortestGroup.length() || shortestGroup.empty()) {

shortestGroup = currentGroup;

}

// Возвращаем самую короткую группу.

return shortestGroup;

}

int main() {

setlocale(LC\_ALL, "rus");

// Входная строка, в которой ищем самую короткую группу.

string inputString = "0011001111000";

// Вызываем функцию для поиска самой короткой группы.

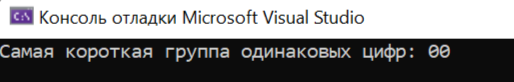
string shortestGroup = findShortestGroup(inputString);

// Выводим результат.

cout << "Самая короткая группа одинаковых цифр: " << shortestGroup << endl;

return 0;

}



**Дополнительные задания**

№1 В заданной последовательности слов найти все слова, имеющие заданное окончание

#include <iostream>

#include <vector>

#include <string>

using namespace std;

int main() {

setlocale(LC\_ALL, "rus");

// Создаем вектор, в котором будут храниться слова для поиска окончания.

vector<std::string> words = { "красивый", "савременный", "молодая", "молодой", "зелёный", "стеклянный", "жёлтая" };

// Определяем заданное окончание, которое будем искать.

string targetSuffix = "ый";

// Выводим заголовок для найденных слов.

cout << "Слова с окончанием '" << targetSuffix << "':" << endl;

// Проходим по каждому слову в векторе.

for (const string& word : words) {

// Проверяем, достаточно ли символов в текущем слове для сравнения с окончанием.

if (word.size() >= targetSuffix.size()) {

// Выделяем подстроку с окончанием из текущего слова.

string wordSuffix = word.substr(word.size() - targetSuffix.size());

// Сравниваем окончание текущего слова с заданным окончанием.

if (wordSuffix == targetSuffix) {

// Если окончание совпадает, выводим это слово.

cout << word << endl;

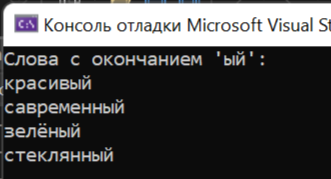
}

}

}

return 0;

}



№3 Из заданного предложения удалить те слова, которые уже встречались в предложении раньше

#include <iostream>

#include <string>

#include <vector>

#include <sstream>

#include <algorithm>

using namespace std;

int main() {

setlocale(LC\_ALL, "rus");

// Создаем строку для хранения входной строки

string sentence;

// Вводим предложение с помощью getline на английском

cout << "Введите предложение: ";

getline(cin, sentence);

// Создаем вектор для хранения слов

vector<string> words;

// Создаем строковый поток для разбиения предложения на слова

istringstream iss(sentence);

string word;

// Разбиваем предложение на слова и сохраняем их в векторе

while (iss >> word) {

words.push\_back(word);

}

vector<string> uniqueWords;

string resultSentence;

// Перебираем слова в порядке их появления в предложении

for (const string& word : words) {

// Если слово еще не встречалось ранее, добавляем его в результат

if (find(uniqueWords.begin(), uniqueWords.end(), word) == uniqueWords.end()) {

resultSentence += word + " ";

uniqueWords.push\_back(word);

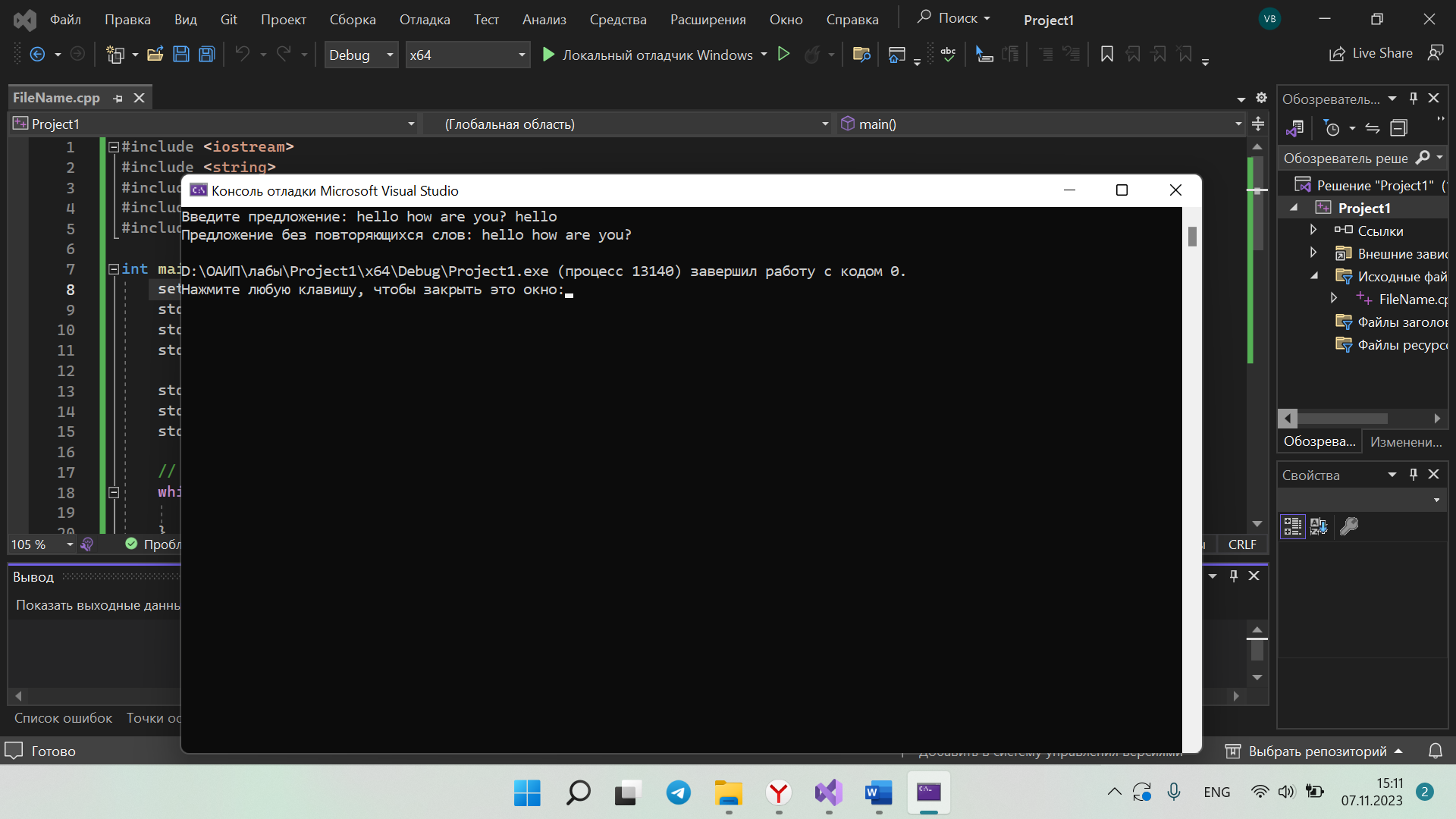
}

}

cout << "Предложение без повторяющихся слов: " << resultSentence << endl;

return 0;

}



№4 Отредактировать заданное предложение, удаляя из него все слова с нечетными номерами и переворачивая слова с четными номерами. Пример: HOW DO YOU DO преобразовать в OD OD.

#include <iostream>

#include <vector>

#include <string>

using namespace std;

int main() {

setlocale(LC\_ALL, "rus");

vector <string> a; // Создаем вектор строк для хранения слов

char s[500];

gets\_s(s); // Вводим предложение с учётом безопасности от переполнения буфера

int i = 0, num = 1;

string k = ""; // Объявляем строку k для временного хранения слова

// Запускаем цикл, который обрабатывает введенное предложение

while (s[i] != '\0') { // Пока не достигнут конец строки

if (num % 2 == 0) { // Если номер слова четный

while (s[i] != ' ' and s[i] != '\0') { // Пока не найден пробел или конец строки

k = s[i] + k; // Добавляем символ в начало строки k

i++;

}

a.push\_back(k); // Добавляем слово k в вектор a

k = ""; // Очищаем строку k для следующего слова

num += 1;

i++;

}

else if (num % 2 != 0) { // Если номер слова нечетный

while (s[i] != ' ' and s[i + 1] != '\0') { // Пока не найден пробел или предпоследний символ строки

i++;

}

num += 1;

i++;

}

}

// Выводим отредактированные слова, хранящиеся в векторе a

for (int i = 0; i < a.size(); i++) {

cout << a[i] << ' '; // Выводим слово с пробелом в качестве разделителя

}

}

