Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

Лабораторная работа 13

По дисциплине «Основы алгоритмизации и программирования»

На тему «Обработка символьной информации»

Выполнил:

Студент 1 курса 9 группы

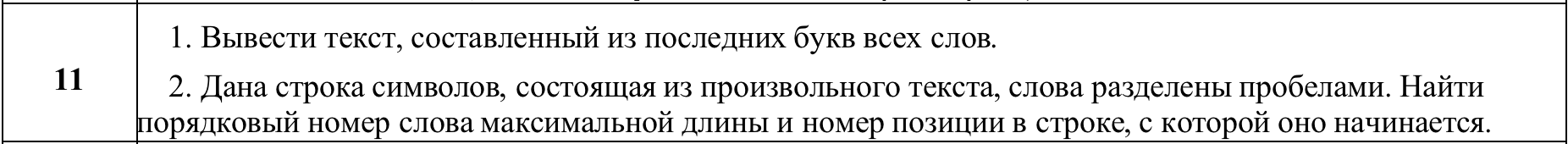
Павлович Ян Андреевич

Преподаватель: асс. Андронова М.В.

2023, Минск

Вариант 11

1. Выполнить задания из таблицы ниже, используя ***индексы*** для доступа к элементам массивов в первой программе и ***указатели*** − во второй программе. При написании программне использовать стандартные функции для строк символов.



1. Вывести текст, составленный из последних букв всех слов.

#include <iostream> //для работы с вводом/выводом

#include <cstring> //для работы со строками

**using** **namespace** std; //объявляем программе, что мы используем пространство имен std

**int** main() {

    setlocale(LC\_CTYPE, "Russian"); //возможность работы с русским языком

**int** pos = 0;

**int** **const** N = 50;

**char** str[] = "Вывести текст, составленный из последних букв всех слов.";

**char**\* str2 = **new** **char**[N]; //отделяем память для массива

    cout << "Исходный текст: вывести текст, составленный из последних букв всех слов.\n\n";

    //с помощью оператора for проходим по каждому символу текста и ищем последние буквы

**for** (**int** i = 0; i <= strlen(str); i++) {

**if** (str[i] == ' ') {

            str2[pos] = str[i - 1];

            pos++;

        }

**else** **if** (str[i] == ',') {

            str2[pos] = str[i - 1];

            pos++;

            i++;

        }

**else** **if** (str[i] == '.') {

            str2[pos] = str[i - 1];

        }

    }

    //выводим результат

    cout << "Результат: ";

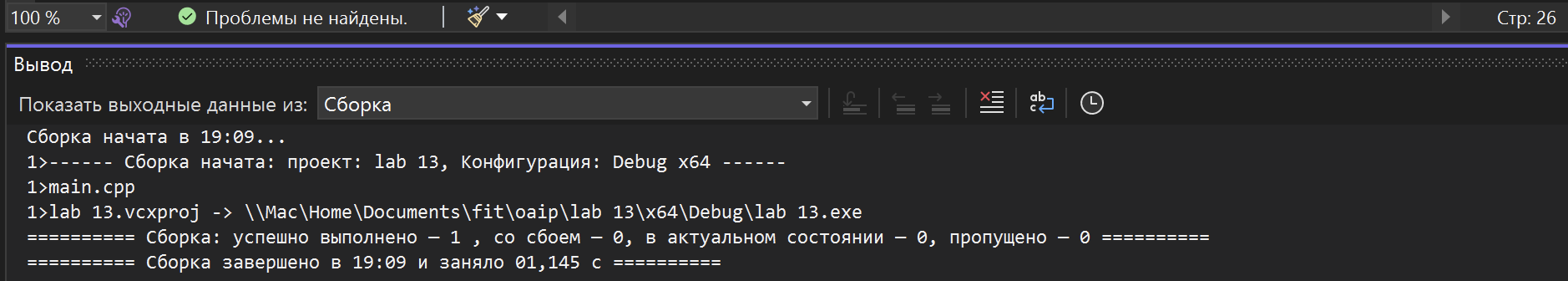
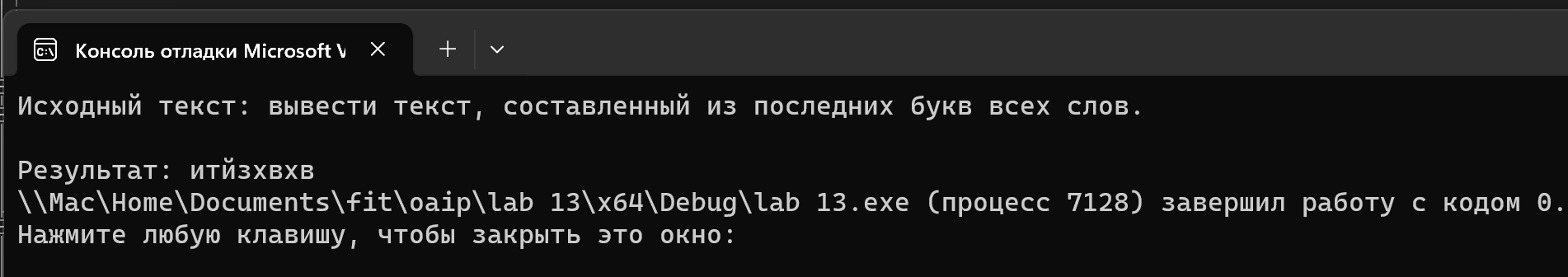
**for** (**int** i = 0; i <= pos; i++) {

        cout << \*(str2 + i);

    }

**return** 0;

}



1. Дана строка символов, состоящая из произвольного текста, слова разделены пробелами. Найти порядковый номер слова максимальной длины и номер позиции в строке, с которой оно начинается.

#include <iostream> //для работы с вводом/выводом

#include <cstring> //для работы со строками

**using** **namespace** std; //объявляем программе, что мы используем пространство имен std

**int** main() {

    setlocale(LC\_CTYPE, "Russian"); //возможность работы с русским языком

**int** max = 0, d = 0, n = 0, max2 = 0;

**char** str[] = "Вывести текст, составленный из последних букв всех слов.";

    cout << "Исходный текст: вывести текст, составленный из последних букв всех слов.\n\n";

    //с помощью оператора for проходим по каждому символу текста и ищем самое большое слово

**for** (**int** i = 0; i <= strlen(str); i++) {

        d++;

**if** (\*(str + i) == ' ') {

            n++;

**if** (max < d) {

                max = d;

                max2 = n;

            }

            d = 0;

        }

**else** **if** (\*(str + i) == ',') {

            n++;

**if** (max < d) {

                max = d;

                max2 = n;

            }

            d = 0;

            i++;

        }

**else** **if** (\*(str + i) == '.') {

            n++;

**if** (max < d) {

                max = d;

                max2 = n;

            }

            d = 0;

        }

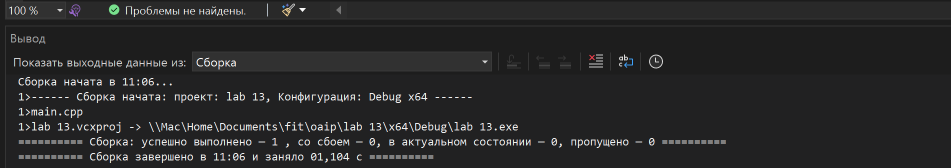
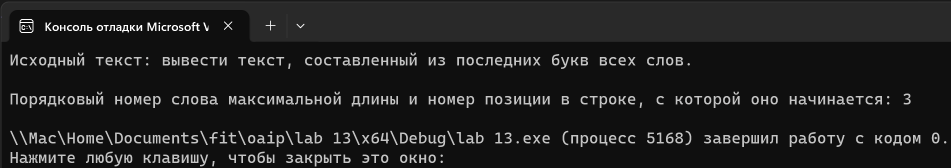
    }

    //выводим результат

    cout << "Порядковый номер слова максимальной длины и номер позиции в строке, с которой оно начинается: " << max2 << endl;

**return** 0;

}



**Дополнительные задачи**

1. Из заданного предложения удалить те слова, которые уже встречались в предложении раньше.

#include <iostream> //для работы с вводом/выводом

**using** **namespace** std; //объявляем программе, что мы используем пространство имен std

// Функция для проверки, содержится ли слово в массиве

**bool** isWordEncountered(**char** word[][50], **int** numEncounteredWords, **char**\* currentWord) {

**for** (**int** i = 0; i < numEncounteredWords; ++i) {

**if** (strcmp(word[i], currentWord) == 0) {

**return** **true**;

        }

    }

**return** **false**;

}

**int** main() {

    setlocale(LC\_CTYPE, "Russian"); //для работы с русским языком

    // Произвольное предложение

**char** sentence[] = "Программирование - это интересно. Программирование - это творчество.";

    // Массив для хранения слов, которые уже встречались

**char** encounteredWords[100][50];

**int** numEncounteredWords = 0;

    // Флаг для определения, находимся ли мы внутри слова

**bool** inWord = **false**;

    // Текущее слово

**char** currentWord[50];

**int** currentWordIndex = 0;

    // Обработка каждого символа в предложении

**for** (**int** i = 0; i <= strlen(sentence); ++i) {

**char** currentChar = sentence[i];

        // Проверка на пробел или конец строки

**if** (currentChar == ' ' || currentChar == '\0') {

            // Завершаем текущее слово

            currentWord[currentWordIndex] = '\0';

            // Проверяем, встречалось ли слово раньше

**if** (!isWordEncountered(encounteredWords, numEncounteredWords, currentWord)) {

                cout << currentWord << " ";

                strcpy\_s(encounteredWords[numEncounteredWords], currentWord);

                ++numEncounteredWords;

            }

            // Сбрасываем индекс текущего слова

            currentWordIndex = 0;

            inWord = **false**;

        }

**else** {

            // Добавляем символ в текущее слово

            currentWord[currentWordIndex] = currentChar;

            ++currentWordIndex;

            inWord = **true**;

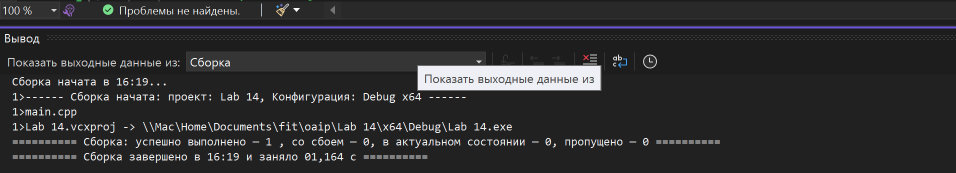
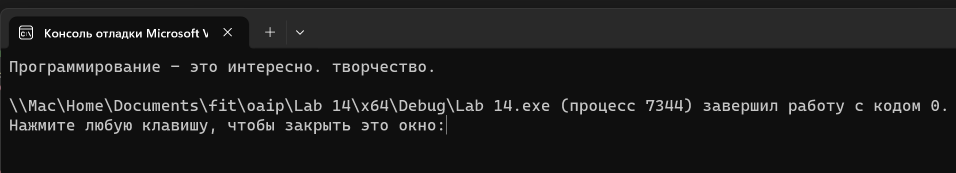
        }

    }

    cout << endl;

**return** 0;

}



2. В имеющемся словаре найти группы слов, записанных одними и теми же буквами и отличающиеся только порядком расположения.

#include <iostream> //для работы с вводом/выводом

#include <algorithm> //algorithm содержит функцию sort()

**using** **namespace** std; //объявляем программе, что мы используем пространство имен std

**int** main() {

    setlocale(LC\_CTYPE, "Russian"); //возможность работы с русским языком

    string dict[] = { "word", "orwd", "dord", "gord" }; //строковый массив - словарь слов

    //для наглядности выводим все слова массива на экран

**for** (**int** i = 0; i < **sizeof**(dict) / **sizeof**(dict[0]); i++) {

        cout << dict[i] << " ";

    }

    cout << endl;

    //вычисляем слова записанные одними и теми же буквами

**for** (**int** i = 0; i < **sizeof**(dict) / **sizeof**(dict[0]); i++) {

        //если их длина одинакова, продолжаем

**if** (dict[i].size() == dict[i + 1].size()) {

            //соритруем наши слова по номерам в кодировке

            sort(dict[i].begin(), dict[i].end());

            sort(dict[i + 1].begin(), dict[i + 1].end());

        }

        //если после сортировки слова идентичны, то они записаны одними и теми же буквами

**if** (dict[i] == dict[i + 1]) {

            cout << "Слово под номером " << i + 1 << " и слово под номером " << i + 2 << " записаны одними и теми же буквами.";

        }

    }

}

1. В заданной последовательности слов найти все слова, имеющие заданное окончание.

#include <iostream> //для работы с вводом/выводом

#include <cstring> //для работы со строками

**using** **namespace** std; //объявляем программе, что мы используем пространство имен std

**int** main() {

    setlocale(LC\_CTYPE, "Russian"); //возможность работы с русским языком

**int** wordcount = 0; **char** tmp[100], end[10]; //wordcount - количество слов последовательности, tmp - буфер, end - окончание

    size\_t endc = 0, match = 0, wordc = 0; //счетчики

    cout << "Введите окончание: ";

    cin >> end;

    cout << ("Введите количество слов в последовательности: ");

    cin >> wordcount;

    cout << "Введите слова: ";

    //вводим слова и проверяем наличие в них заданного окончания

**for** (**int** j = 0; j < wordcount; j++) {

        cin >> tmp;

        //ищем среди символов слова окончание

**for** (**int** i = 0; i < strlen(tmp); i++) {

**if** (tmp[i] == end[endc]) {

                match++; endc++;

                //если есть соответствие окончанию, выводим текст об этом и аннулируем счетчики

**if** (match == strlen(end)) {

                    cout << "Слово имеет заданное окончание\n";

                    endc = 0; match = 0;

**break**;

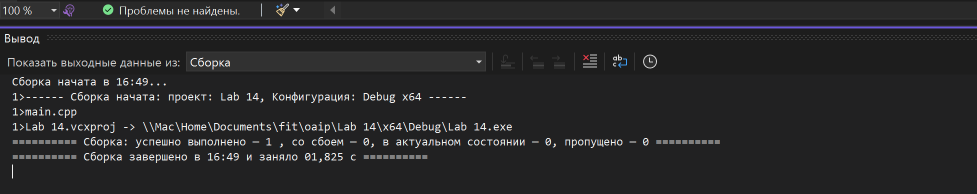
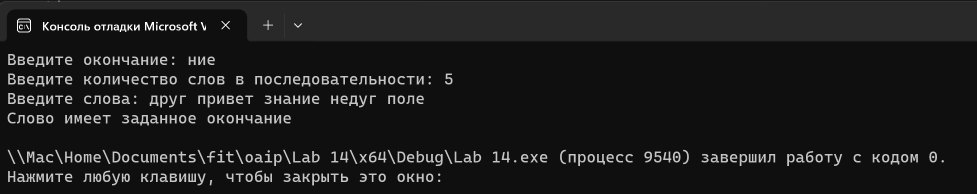
                }

            }

        }

    }

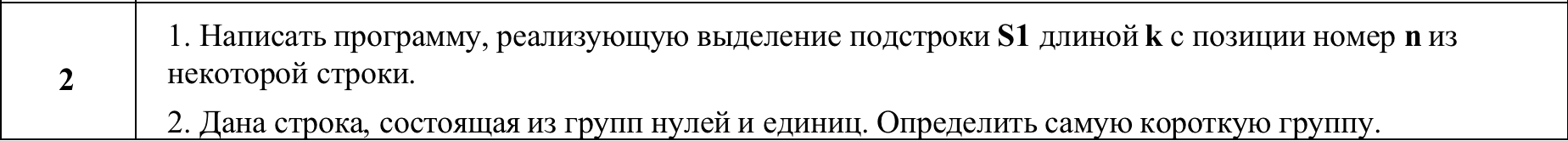
}



Другие варианты сделаны из-за невнимательности (не заметил сразу дополнительные задания)

Вариант 2

6. Выполнить задания из таблицы ниже, используя ***индексы*** для доступа к элементам массивов в первой программе и ***указатели*** − во второй программе. При написании программне использовать стандартные функции для строк символов.



1. Написать программу, реализующую выделение подстроки **S1** длиной **k** с позиции номер **n** из некоторой строки.

#include <iostream> //для работы с вводом/выводом

#include <cstring> //для работы со строками

**using** **namespace** std; //объявляем программе, что мы используем пространство имен std

**int** main() {

    setlocale(LC\_CTYPE, "Russian"); //возможность работы с русским языком

**int** k, n, pos;

**int** **const** N = 50;

**char** str[] = "строкаизкотороймывозьмемнекоторыйтекст";

**char**\* str2 = **new** **char**[N]; //отделяем память для массива

    cout << "Исходная строка: строкаизкотороймывозьмемнекоторыйтекст.\n";

    cout << "Введите позицию: ";

    cin >> n;

    cout << "Введите длину: ";

    cin >> k;

    pos = n - 1;

    //с помощью оператора for отделяем нужные символы

**for** (**int** i = 0; i < k ; i++) {

        \*(str2 + i) = \*(str + pos);

        pos++;

    }

    //выводим результат

    cout << "Результат: ";

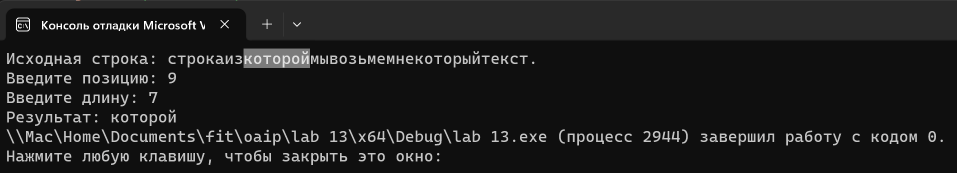
**for** (**int** i = 0; i < k; i++) {

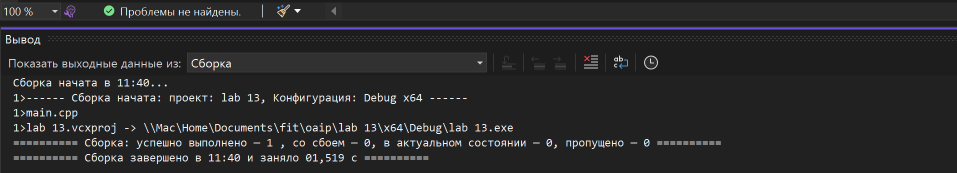
        cout << \*(str2 + i);

    }

**return** 0;

}





2. Дана строка, состоящая из групп нулей и единиц. Определить самую короткую группу.

#include <iostream> //для работы с вводом/выводом

#include <cstring> //для работы со строками

**using** **namespace** std; //объявляем программе, что мы используем пространство имен std

**int** main() {

    setlocale(LC\_CTYPE, "Russian"); //возможность работы с русским языком

**int** pos, sum = 1, min = 15, min2 = 15;

**int** **const** N = 50;

**char** str[] = "0100011011100000101";

**char**\* str2 = **new** **char**[N]; //отделяем память для массива

    cout << "Исходная строка: 0100011011100000101\n";

    //с помощью оператора for отделяем нужные символы

**for** (**int** i = 0; i <= strlen(str) ; i++) {

**int** p = i + 1;

**if** (\*(str + i) == '0') {

**for** (p; p <= strlen(str); p++) {

**if** (\*(str + p) == '0') {

                    sum++;

                }

**else** {

**if** (sum < min) {

                        min = sum;

                        sum = 1;

                    }

**break**;

                }

            }

        }

**else** {

**for** (p; p <= strlen(str); p++) {

**if** (\*(str + p) == '1') {

                    sum++;

                }

**else** {

**if** (sum < min2) {

                        min2 = sum;

                        sum = 1;

                    }

**break**;

                }

            }

        }

    }

    //выводим результат

cout << "Самая короткая группа: ";

    //группа - это минимум 2 символа

**if** (min == 1) {

        min++;

    }

**if** (min2 == 1) {

        min2++;

    }

    //выводим самую короткую группу

**if** (min < min2) {

**for** (**int** i = 0; i < min; i++) {

            cout << 0;

        }

    }

**else** {

**for** (**int** i = 0; i < min2; i++) {

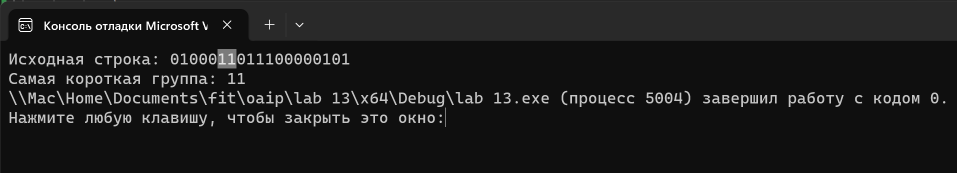
            cout << 1;

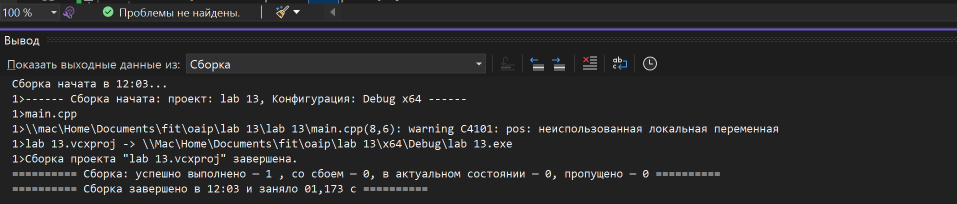
        }

    }

**return** 0;

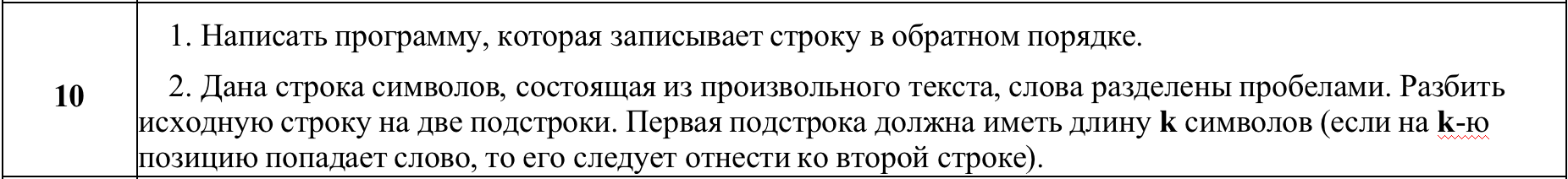
}





Вариант 10

6. Выполнить задания из таблицы ниже, используя ***индексы*** для доступа к элементам массивов в первой программе и ***указатели*** − во второй программе. При написании программне использовать стандартные функции для строк символов.



1. Написать программу, которая записывает строку в обратном порядке.

#include <iostream> //для работы с вводом/выводом

#include <cstring> //для работы со строками

**using** **namespace** std; //объявляем программе, что мы используем пространство имен std

**int** main() {

    setlocale(LC\_CTYPE, "Russian"); //возможность работы с русским языком

**int** pos = 0;

**int** **const** N = 50;

**char** str[] = "0100011011100000101";

**char**\* str2 = **new** **char**[N]; //отделяем память для массива

    cout << "Исходная строка: 0100011011100000101\n";

    //с помощью оператора for записываем символы в обратном порядке

**for** (**int** i = strlen(str); i >= 0; i--) {

        \*(str2 + pos) = \*(str + i);

        pos++;

    }

    cout << "Результат: ";

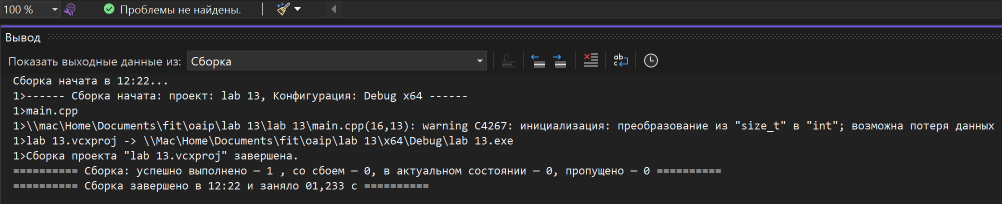
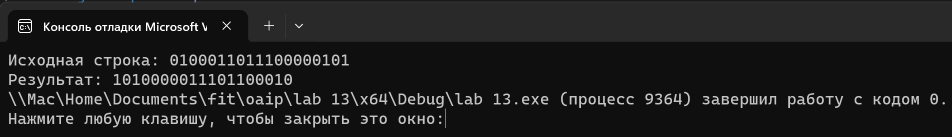
**for** (**int** i = 0; i < pos; i++) {

        cout << \*(str2 + i);

    }

**return** 0;

}



2. Дана строка символов, состоящая из произвольного текста, слова разделены пробелами. Разбить исходную строку на две подстроки. Первая подстрока должна иметь длину **k** символов (если на **k**-ю позицию попадает слово, то его следует отнести ко второй строке).

#include <iostream> //для работы с вводом/выводом

#include <cstring> //для работы со строками

**using** **namespace** std; //объявляем программе, что мы используем пространство имен std

**int** main() {

    setlocale(LC\_CTYPE, "Russian"); //возможность работы с русским языком

**char** inputString[] = "Дана строка символов, состоящая из произвольного текста, слова разделены пробелами.";

    cout << "Исходная строка: Дана строка символов, состоящая из произвольного текста, слова разделены пробелами.\n";

    // Ввод длины первой подстроки

    cout << "Введите длину первой подстроки (k): ";

**int** k;

    cin >> k;

    // Переменные для хранения результатов

**char** firstSubstring[1000] = "";

**char** secondSubstring[1000] = "";

    // Индексы для прохода по исходной строке

**int** currentIndex = 0;

**int** firstIndex = 0;

**int** secondIndex = 0;

**bool** wordMoved = **false**;

    // Проход по исходной строке

**while** (currentIndex < strlen(inputString) && inputString[currentIndex] != '\0') {

        // Если текущий символ - не пробел

**if** (inputString[currentIndex] != ' ') {

            // Если первая подстрока еще не заполнена

**if** (firstIndex < k) {

                firstSubstring[firstIndex++] = inputString[currentIndex];

            }

            // Если первая подстрока уже заполнена, добавляем второй подстроке

**else** {

                // Добавляем пробел между словами во второй подстроке, если это не первое слово

**if** (secondIndex > 0 && !wordMoved) {

                    secondSubstring[secondIndex++] = ' ';

                }

                // Переносим целые слова на вторую строку

**while** (inputString[currentIndex] != ' ' && inputString[currentIndex] != '\0') {

                    secondSubstring[secondIndex++] = inputString[currentIndex++];

                }

                wordMoved = **true**;

                currentIndex--; // Вернемся на пробел или нулевой символ

            }

        }

        // Если текущий символ - пробел

**else** {

            // Добавляем пробел к первой подстроке, если первая подстрока еще не заполнена

**if** (firstIndex < k) {

                firstSubstring[firstIndex++] = ' ';

            }

            wordMoved = **false**; // Сбрасываем флаг, так как встретился пробел

        }

        // Увеличиваем индекс текущего символа

        currentIndex++;

    }

    // Завершаем строки нулевыми символами

    firstSubstring[firstIndex] = '\0';

    secondSubstring[secondIndex] = '\0';

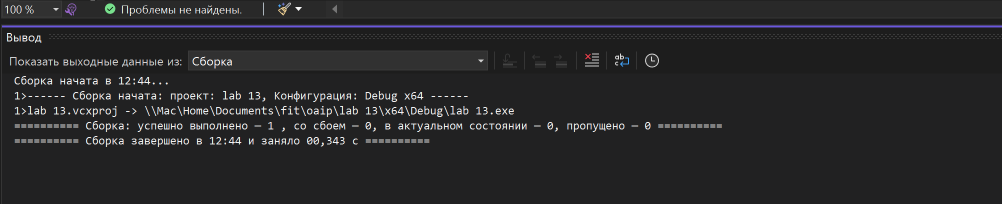
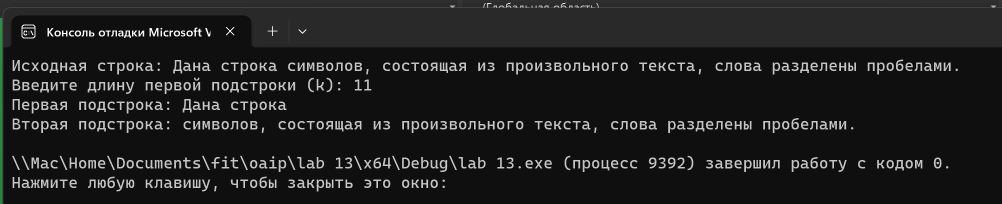
    // Вывод результатов

    cout << "Первая подстрока: " << firstSubstring << endl;

    cout << "Вторая подстрока: " << secondSubstring << endl;

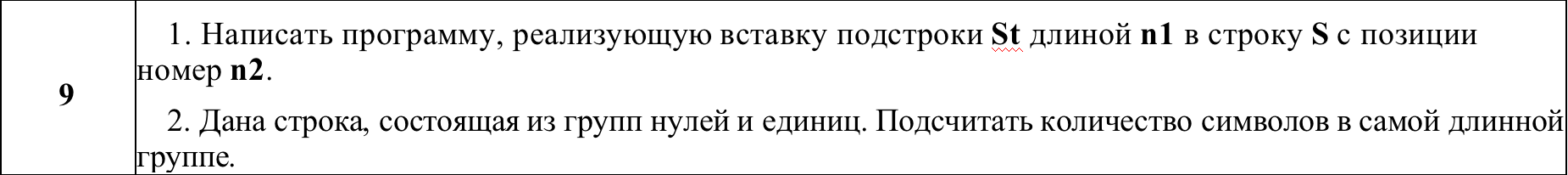
**return** 0;

}



Вариант 9

6. Выполнить задания из таблицы ниже, используя ***индексы*** для доступа к элементам массивов в первой программе и ***указатели*** − во второй программе. При написании программне использовать стандартные функции для строк символов.



1. Написать программу, реализующую вставку подстроки **St** длиной **n1** в строку **S** с позиции номер **n2**.

#include <iostream> //для работы с вводом/выводом

#include <cstring> //для работы со строками

#include <Windows.h> //для работы с функциями windows

**using** **namespace** std; //объявляем программе, что мы используем пространство имен std

**int** main() {

    //установка кодовых страниц

    SetConsoleCP(1251);

    SetConsoleOutputCP(1251);

    setlocale(LC\_CTYPE, "Russian"); //возможность работы с русским языком

    // Ввод строки S

    cout << "Введите строку S: ";

**char** S[100];  // предполагаем максимальную длину строки

    cin.getline(S, **sizeof**(S));

    // Ввод подстроки St

    cout << "Введите подстроку St: ";

**char** St[50];  // предполагаем максимальную длину подстроки

    cin.getline(St, **sizeof**(St));

    // Ввод позиции n2

    cout << "Введите позицию n2: ";

    size\_t n2;  // используем size\_t для предотвращения проблемы с преобразованием

    cin >> n2;

    // Вычисление длины строк

    size\_t lenS = strlen(S);

    size\_t lenSt = strlen(St);

    // Проверка корректности позиции n2

**if** (n2 > lenS) {

        cout << "Ошибка: Некорректная позиция вставки.\n";

**return** 1;  // завершение программы с ошибкой

    }

    // Перемещение символов вправо для создания места для подстроки

**for** (size\_t i = lenS; i > n2; --i) {

        S[i + lenSt] = S[i];

    }

    // Копирование подстроки St в строку S на позицию n2

**for** (size\_t i = 0; i < lenSt; ++i) {

        S[n2 + i] = St[i];

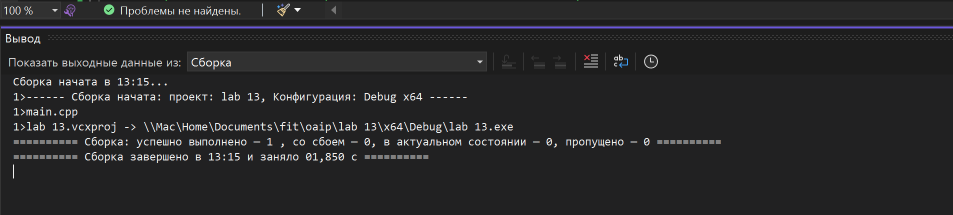
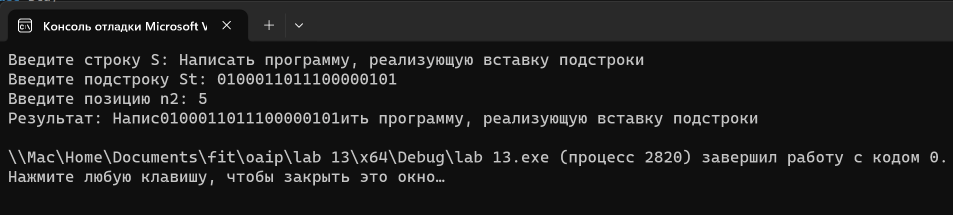
    }

    // Вывод измененной строки

    cout << "Результат: " << S << endl;

**return** 0;

}



2. Дана строка, состоящая из групп нулей и единиц. Подсчитать количество символов в самой длинной группе.

#include <iostream> //для работы с вводом/выводом

#include <cstring> //для работы со строками

**using** **namespace** std; //объявляем программе, что мы используем пространство имен std

**int** main() {

    setlocale(LC\_CTYPE, "Russian"); //возможность работы с русским языком

**int** pos, sum = 1, max = 1, max2 = 1;

**int** **const** N = 50;

**char** str[] = "0100011011100000101";

**char**\* str2 = **new** **char**[N]; //отделяем память для массива

    cout << "Исходная строка: 0100011011100000101\n";

    //с помощью оператора for отделяем нужные символы

**for** (**int** i = 0; i <= strlen(str); i++) {

**int** p = i + 1;

**if** (\*(str + i) == '0') {

**for** (p; p <= strlen(str); p++) {

**if** (\*(str + p) == '0') {

                    sum++;

                    p++;

                }

**else** {

**if** (sum > max) {

                        max = sum;

                        sum = 1;

                    }

**break**;

                }

            }

        }

**else** {

**for** (p; p <= strlen(str); p++) {

**if** (\*(str + p) == '1') {

                    sum++;

                    p++;

                }

**else** {

**if** (sum > max2) {

                        max2 = sum;

                        sum = 1;

                    }

**break**;

                }

            }

        }

    }

    //выводим результат

    cout << "Самая короткая группа: ";

    //выводим самую короткую группу

**if** (max > max2) {

**for** (**int** i = 0; i < max2; i++) {

            cout << 0;

        }

    }

**else** {

**for** (**int** i = 0; i < max; i++) {

            cout << 1;

        }

    }

**return** 0;

}

