**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: “Текстовые строки как массив символов”

|  |  |  |
| --- | --- | --- |
| Студент гр. 2372 |  | Карпачёв А.С. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2022

**Цель работы.**

Изучение внутреннее строение строки string, понимания, как хранится string в памяти.

**Основные теоретические положения.**

Любая строка является набором (массивом) символов. Т.е.:

“Любой текст” в виде массива символов представляется так:

'Л', 'ю', 'б', 'о', 'й', ' ', 'т', 'е', 'к', 'с', 'т', ‘\0’

В конце любой строки должен находится символ ‘\0’. Это нужно для того, чтобы случайно не прочитать лишнюю память. Строка – это массив. Но длина его нигде не записана, поэтому при чтении строки, когда выполнение дойдёт то спецсимвола, будет понятно, что строка закончилась.

Строки можно объявить разными способами:

char S1[15] = “This is text”;

char S2[] = “Пример текста”;

Если не указать размерность в скобках, то она автоматически определится исходя из присвоенной строки.

Выводить и вводить строки не очень сложно:

char Str[21] = “Это пример текста”;

cout << Str << endl;

cout << “Это текстовый литерал.” << endl;

Также можно использовать cin.getline:

const int N = 21;

char Str [N];

cin.getline (Str, N);

cout << Str << endl;

*Поиск подстроки в строке.*

Нередко возникает необходимость поиска шаблона в строке. Для решения этого вопроса я рассмотрю два алгоритма: линейний алгоритм и алгоритм Кнута-Морриса-Прата (КМП).

*Линейный алгоритм.*

Алгоритм очень прост: мы ставим шаблон на нулевую позицию, сравниваем все символы шаблона с символами строки. Если совпало, то подстрока найдена, если нет, то переставляем шаблон на следующую позицию. По достижении конца, когда длина шаблона + 1 равно длине строки, поиск заканчивается.

*Алгоритм КМП.*

Алгоритм разделён на два этапа: составление префиксной функции и сопоставление её со строкой.

Рассмотрим на примере шаблон ”ABBABCAB”:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| A | B | B | A | B | C | A | B |
| 0 | 0 | 0 | 1 | 2 | 0 | 1 | 2 |

Для каждого индекса числа мы пытаемся найти совпадающий суффикс и префикс одинаковой длины (и максимальной):

Для index = 0 префиксная функция равна нулю (или в тех случаях, когда длина суффикса равна длине строки)

**AB** – Префикс А, Суффикс В – несовпадение, длина = 0

**ABB** – Префикс А, Суффикс В – несовпадение, длина = 0

**ABBA** – Префикс А, Суффикс A – макс. длина = 1

**ABBAB** – Префикс АB, Суффикс AВ – макс. длина = 2

**ABBABC** – Префикс А, Суффикс С – несовпадение, длина = 0

**ABBABCA** – Префикс А, Суффикс А – макс. длина = 1

**ABBABCAB** – Префикс АB, Суффикс AВ – макс. длина = 2

Префиксная функция готова:

Int[] prefix = {0, 0, 0, 1, 2, 0, 1, 2};

Теперь перейдём к строке.

Установим два курсора: i, j. Один указывает на начало строки, а другой – на начало образца.

Пока символы под курсорами совпадают, они оба двигаются вперёд на единицу: i++; j++. Если второй курсор дойдёт до конца шаблона, то подстрока найдена.

Если символы под курсорами не совпадают и курсор находится в начале образца, то мы просто передвигаем курсор строки вперёд на единицу.

Если же символы под курсорами не совпадают и курсор находится НЕ в начале образца, то мы обращаемся к префикс функции. Мы устанавливаем значение курсора образца j на значение префикс функции по индексу j – 1. Таким образом, мы возвращаемся на последнее совпадение суффиксов и префиксов.

Если курсор строки I стал равен длине строки, то поиск завершён.

**Постановка задачи.**

Вся работа должна быть выполнена с использованием массивов символов.

1. С клавиатуры или с файла (по выбору) вводится последовательность, содержащая от 1 до 50 слов, в каждом из которых от 1 до 10 строчных латинских букв и цифр. Между соседними словами произвольное количество пробелов. За последним символом стоит точка.
2. Необходимо отредактировать входной текст:

- удалив лишние пробелы.

- удалить лишние знаки препинания.

- исправить регистр букв, если это требуется.

1. После окончания ввода последовательности вывести на экран сначала все слова, содержащие только буквы, затем слова, содержащие только цифры, а потом слова, содержащие и буквы, и цифры.
2. Вывести все слова исходной последовательности на экран вертикально.
3. Необходимо найти все подстроки, которую введёт пользователь в имеющейся строке, используя алгоритм линейного поиска и КМП.

**Выполнение работы.**

Вспомогательная функция ввода в терминал и получения из него данных в переменную (принимает только указанный тип, иначе выводит ошибку):

Вспомогательные функции:

onlyLetters – содержит ли строка ТОЛЬКО буквы

onlyNumbers – содержит ли строка ТОЛЬКО цифры

onlyLettersNumbers – содержит ли строка цифры и буквы

lower – перевод символов в нижний регистр

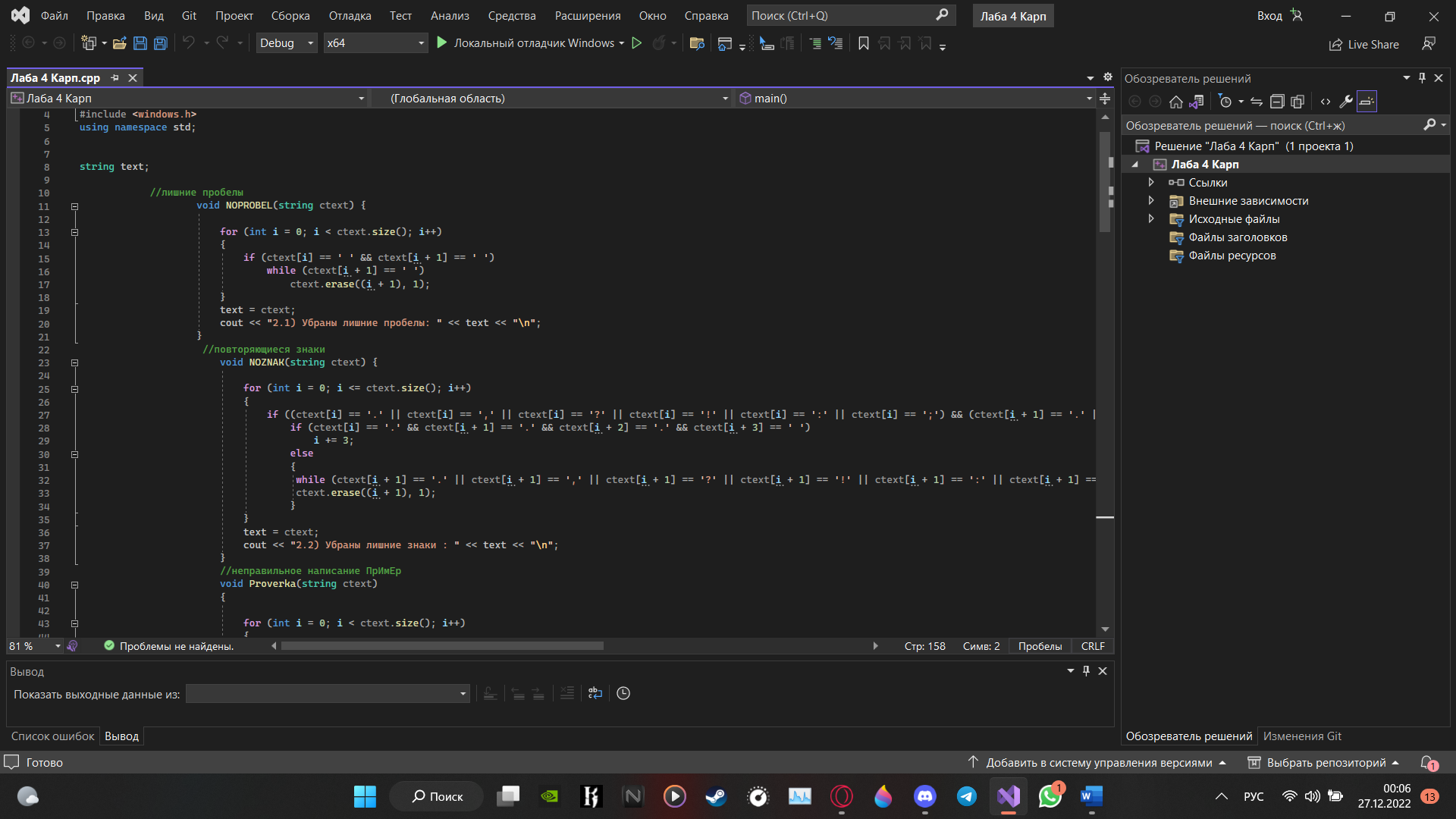
isPunctuation – является ли символ знаком пунктуацией

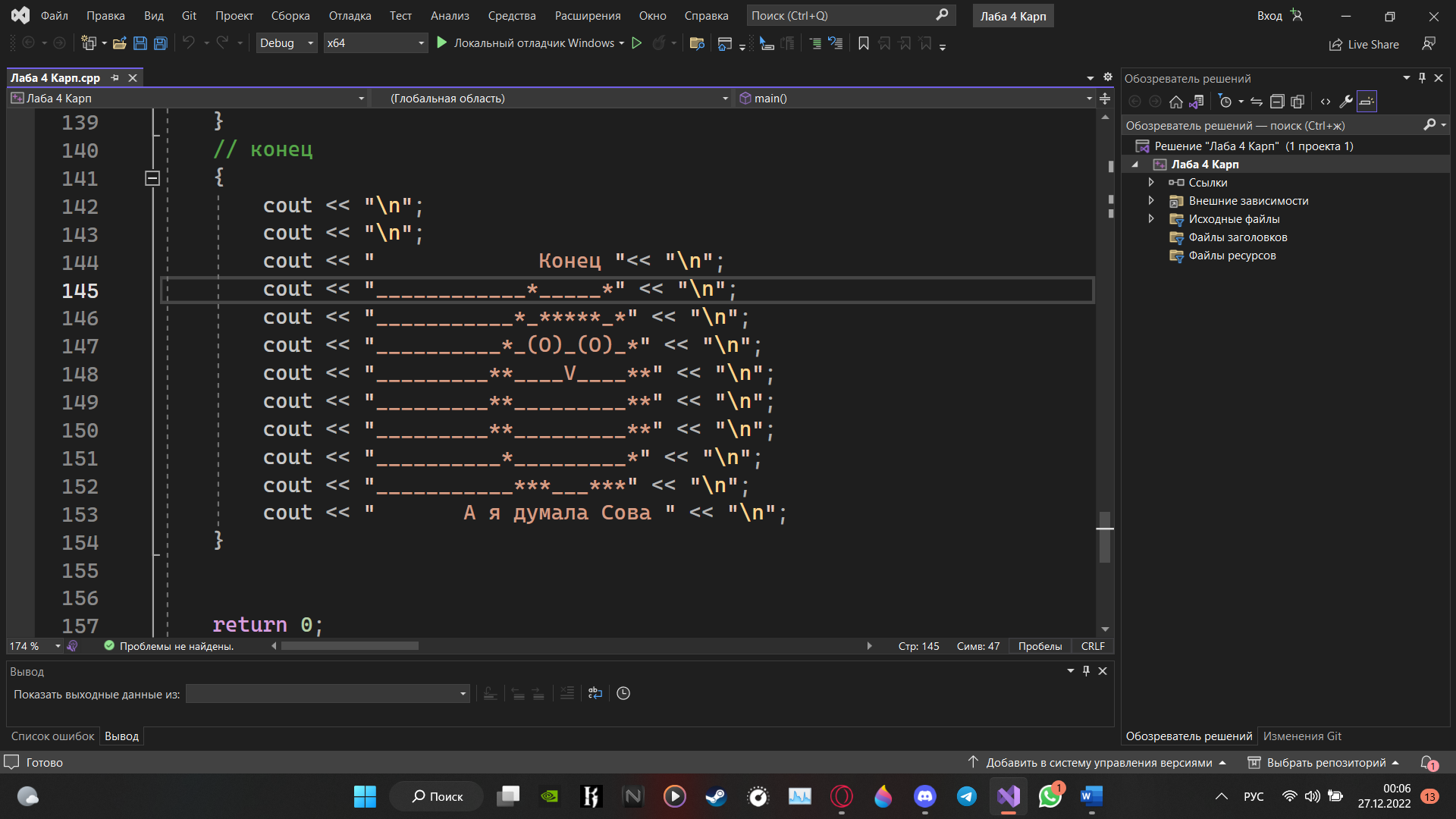
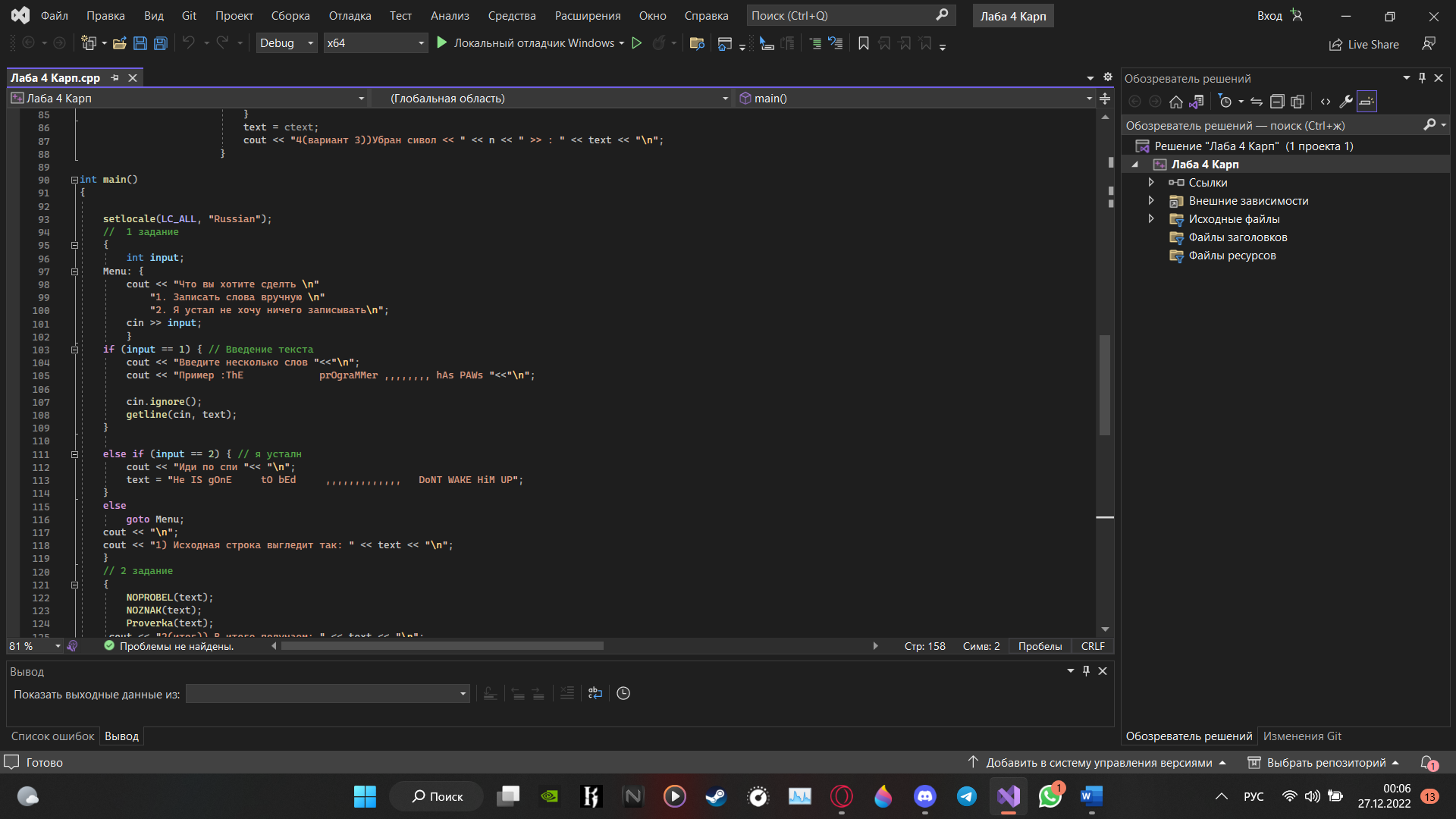
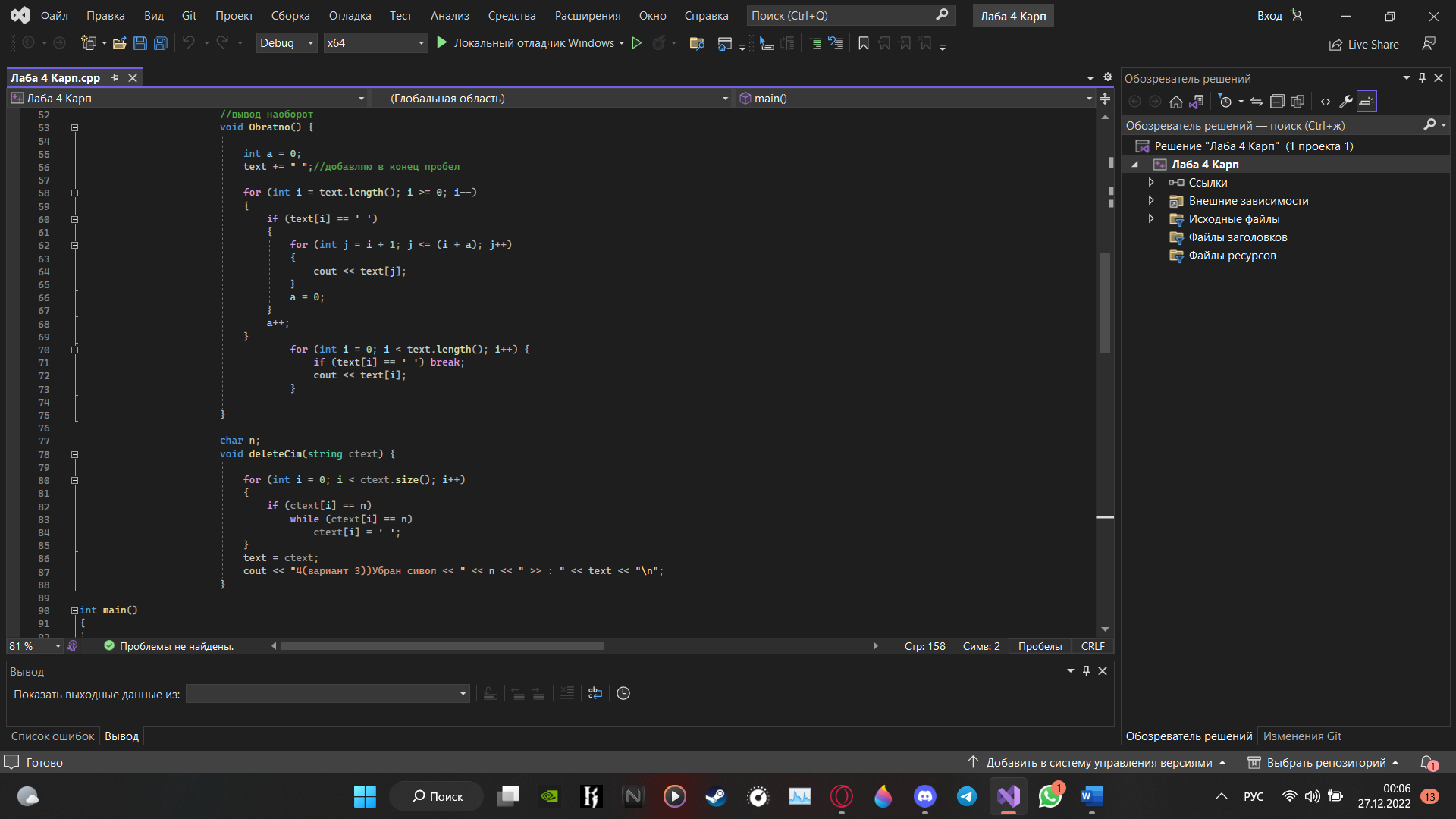
slen – найти длину строки

scopy – копирование одной строки в другую

sappend – копирование одной строки в конец другой

sequal – проверка на равенство двух строк





**Выводы.**

Я научился работать с массивами строк, изучил их строение.

Приложение А

рабочий код

#include <iostream>

#include <fstream>

#include <string>

#include <windows.h>

using namespace std;

string text;

//лишние пробелы

void NOPROBEL(string ctext) {

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

{

if (ctext[i] == ' ' && ctext[i + 1] == ' ')

while (ctext[i + 1] == ' ')

ctext.erase((i + 1), 1);

}

text = ctext;

cout << "2.1) Убраны лишние пробелы: " << text << "\n";

}

//повторяющиеся знаки

void NOZNAK(string ctext) {

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

{

if ((ctext[i] == '.' || ctext[i] == ',' || ctext[i] == '?' || ctext[i] == '!' || ctext[i] == ':' || ctext[i] == ';') && (ctext[i + 1] == '.' || ctext[i + 1] == ',' || ctext[i + 1] == '?' || ctext[i + 1] == '!' || ctext[i + 1] == ':' || ctext[i + 1] == ';'))

if (ctext[i] == '.' && ctext[i + 1] == '.' && ctext[i + 2] == '.' && ctext[i + 3] == ' ')

i += 3;

else

{

while (ctext[i + 1] == '.' || ctext[i + 1] == ',' || ctext[i + 1] == '?' || ctext[i + 1] == '!' || ctext[i + 1] == ':' || ctext[i + 1] == ';')

ctext.erase((i + 1), 1);

}

}

text = ctext;

cout << "2.2) Убраны лишние знаки : " << text << "\n";

}

//неправильное написание ПрИмЕр

void Proverka(string ctext)

{

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

{

if (ctext[i] == ' ')

i += 2;

ctext[i] = tolower(ctext[i]);

}

text = ctext;

cout << "2.3) Исправлена запись слова : " << text << "\n";

}

//вывод наоборот

void Obratno() {

int a = 0;

text += " ";//добавляю в конец пробел

for (int i = text.length(); i >= 0; i--)

{

if (text[i] == ' ')

{

for (int j = i + 1; j <= (i + a); j++)

{

cout << text[j];

}

a = 0;

}

a++;

}

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

if (text[i] == ' ') break;

cout << text[i];

}

}

char n;

void deleteCim(string ctext) {

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

{

if (ctext[i] == n)

while (ctext[i] == n)

ctext[i] = ' ';

}

text = ctext;

cout << "4(вариант 3))Убран сивол << " << n << " >> : " << text << "\n";

}

int main()

{

setlocale(LC\_ALL, "Russian");

// 1 задание

{

int input;

Menu: {

cout << "Что вы хотите сделть \n"

"1. Записать слова вручную \n"

"2. Я устал не хочу ничего записывать\n";

cin >> input;

}

if (input == 1) { // Введение текста

cout << "Введите несколько слов "<<"\n";

cout << "Пример :ThE prOgraMMer ,,,,,,,, hAs PAWs "<<"\n";

cin.ignore();

getline(cin, text);

}

else if (input == 2) { // я усталн

cout << "Иди по спи "<< "\n";

text = "He IS gOnE tO bEd ,,,,,,,,,,,,, DoNT WAKE HiM UP";

}

else

goto Menu;

cout << "\n";

cout << "1) Исходная строка выгледит так: " << text << "\n";

}

// 2 задание

{

NOPROBEL(text);

NOZNAK(text);

Proverka(text);

cout << "2(итог)) В итоге получаем: " << text << "\n";

}

// 3 задание

{

cout << "3(вариант 1)) Cтрока в обратном порядке: ";

Obratno();

}

// 4 Задание

{

cout << "\n";

cout << "4(вариант 3))Введите символ который хотите удалить : ";

cin >> n;

cout << "\n";

deleteCim(text);

}

// конец

{

cout << "\n";

cout << "\n";

cout << " Конец "<< "\n";

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\*\_\_\_\_\_\*" << "\n";

cout << "\_\_\_\_\_\_\_\_\_\_\_\*\_\*\*\*\*\*\_\*" << "\n";

cout << "\_\_\_\_\_\_\_\_\_\_\*\_(O)\_(O)\_\*" << "\n";

cout << "\_\_\_\_\_\_\_\_\_\*\*\_\_\_\_V\_\_\_\_\*\*" << "\n";

cout << "\_\_\_\_\_\_\_\_\_\*\*\_\_\_\_\_\_\_\_\_\*\*" << "\n";

cout << "\_\_\_\_\_\_\_\_\_\*\*\_\_\_\_\_\_\_\_\_\*\*" << "\n";

cout << "\_\_\_\_\_\_\_\_\_\_\*\_\_\_\_\_\_\_\_\_\*" << "\n";

cout << "\_\_\_\_\_\_\_\_\_\_\_\*\*\*\_\_\_\*\*\*" << "\n";

cout << " А я думала Сова " << "\n";

}

return 0;

}