МИНИСТЕРСТВО науки и высшего ОБРАЗОВАНИЯ РОссИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(национальный исследовательский университет)»

Институт №3 «Системы управления, информатика и электроэнергетика»

Кафедра № 304 «Вычислительные машины, системы и сети»

Программирование

Отчет по лабораторной работе №1

Символьные данные

Выполнили студенты группы M3О-211Б-21

Плоцкий Б.А.

Раужев Ю.М.

Проверила Дмитриева Е.А.

Москва 2022 г.

Оглавление

[Задание 3](#_Toc99210030)

[Общая блок-схема программы 4](#_Toc99210031)

[Структурные схемы алгоритмов 5](#_Toc99210032)

[Функция read\_str() 5](#_Toc99210033)

[Функция first\_task() 7](#_Toc99210034)

[Функция second\_task() 9](#_Toc99210035)

[Код программы 12](#_Toc99210036)

[Тестирование программы 25](#_Toc99210037)

[Тестирование некорректных режимов 25](#_Toc99210038)

[Тест 1 25](#_Toc99210039)

[Тест 2 25](#_Toc99210040)

[Тест 3 25](#_Toc99210041)

[Тестирование корректных режимов 27](#_Toc99210042)

[Тест 1 27](#_Toc99210043)

[Тест 2 28](#_Toc99210044)

[Тест 3 29](#_Toc99210045)

[Тест 4 30](#_Toc99210046)

[Тест 5 31](#_Toc99210047)

[Тест 6 32](#_Toc99210048)

[Тест 7 33](#_Toc99210049)

[Тест 8 34](#_Toc99210050)

[Тест 9 35](#_Toc99210051)

[Тест 10 36](#_Toc99210052)

[Тест 11 37](#_Toc99210053)

[Тест 12 38](#_Toc99210054)

[Вывод 39](#_Toc99210055)

# Задание

**1 этап работы**

Разработать функции, формирующие последовательности чисел:

* упорядоченные по возрастанию,
* упорядоченные по убыванию,
* случайную последовательность,
* частично упорядоченные: «пилообразные», «синусоидальные», «ступенчатые».

для целых чисел и чисел с плавающей запятой (должно быть два комплекта функций и соответственно результатов) .

При реализации функций считать, что выделение памяти под последовательности происходит вне этих функций, функции в качестве формальных параметров получают указатель на массив, его размер и, возможно, **диапазон изменения величин и длину интервалов** (для частично упорядоченных последовательностей).

Рекомендуется использовать указатели на функцию в качестве формального параметра или массив указателей на функции для автоматизации сбора статистической информации.

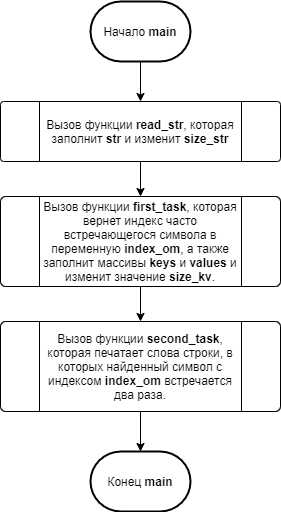
Для массивов размерностью **от 150 до 200 элементов** (размерность выбирается самостоятельно) предусмотреть вывод значений в файл.

Используя данные из файлов, **построить графики** полученных зависимостей (возможно с привлечением других программных средств) для доказательства правильности полученных функций.

**2 этап работы**

1. Оценить длительность формирования последовательностей всех типов для различных значений размеров последовательностей (5⋅105, 10⋅105, …, 50⋅105), и на основе полученных значений построить графики зависимостей длительностей формирования массивов от их размера (лучше в виде столбчатых диаграмм, можно в Microsoft Exel).
2. Составить отчет, в котором привести структурные схемы алгоритмов главной функции и какой-то одной из функций (например, реализующей частично упорядоченную последовательность), текст программы, результаты по временным интервалам для различных размерностей массивов, графики зависимостей (времени от размерностей) и выводы по полученным результатам.

# Общая блок-схема программы



# Структурные схемы алгоритмов

## Функция read\_str()

1. Назначение

выбор места считывания информации

1. Прототип функции

Void read\_str(

input\_typeinput, // считывание из файла/консоли

int&size\_str, // размермассиваstr

char\*&str // массивstr

)

1. Обращение

read\_str(in, size\_str, str)

1. Описание параметров

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификаторы | Тип | Назначение | Входной/Выходной |
| input | Перечисление | Выбор места считывания информации | входной |
| size\_str | целочисленный | размер строки | выходной |
| str | строка | строка | выходной |

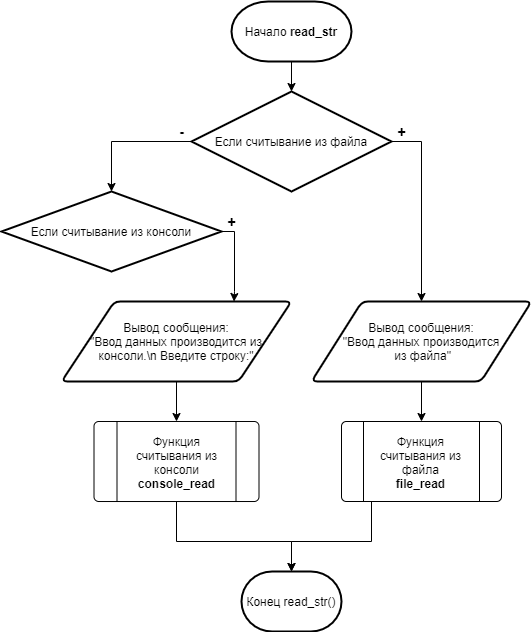
****

Рисунок 1. Структурная схема алгоритма функцииread\_str.

## Функция first\_task()

1. Назначение

функция, решающаяпервуюзадачу

1. Прототип функции

intfirst\_task(

int&size\_kv, // размермассива keys

char\*& keys, // массив keys

int\*& values, // массив values

int&size\_str, // размермассиваstr

char\*&str, // массивstr

input\_typeinput // чтениеизфайла/консоли

)

1. Обращение

index\_om = first\_task(size\_kv, keys, values, size\_str, str, in);

1. Описание параметров

|  |  |  |  |
| --- | --- | --- | --- |
| Индикатор | Тип | Назначение | Входной/Выходной |
| input | Перечисление | Выбор места считывания информации | входной |
| size\_str | целочисленный | размер строки | входной |
| str | строка | строка | входной |
| size\_kv | целочисленный | размер | выходной |
| keys | строка | строка | выходной |
| values | Целочисленный массив | массив | выходной |

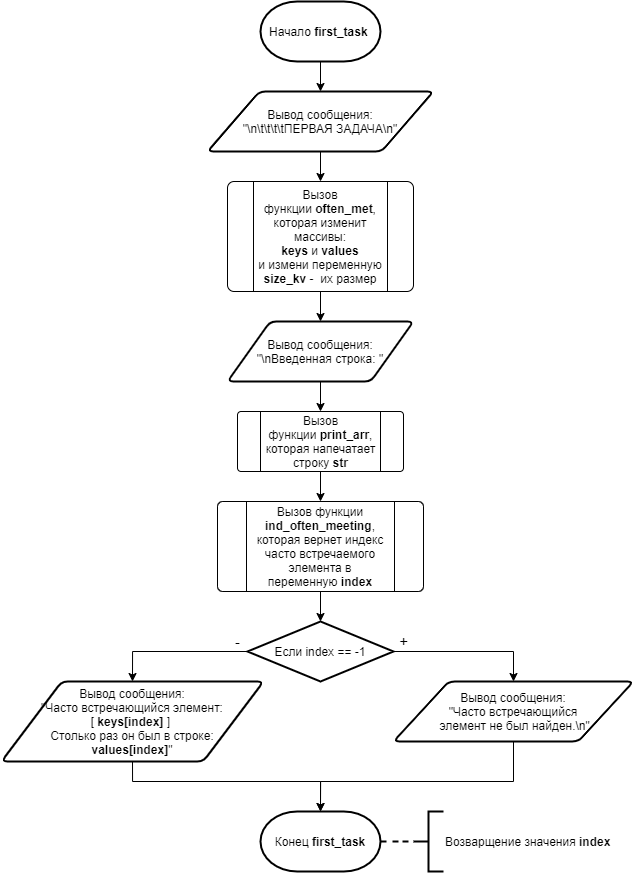


Рисунок 2. Структурная схема алгоритма функции first\_task().

## Функция second\_task()

1. Назначение

функция, решающаявторуюзадачу

1. Прототип функции

Voidsecond\_task(

int&size\_str, // размермассива str

char\*&str, // массив str

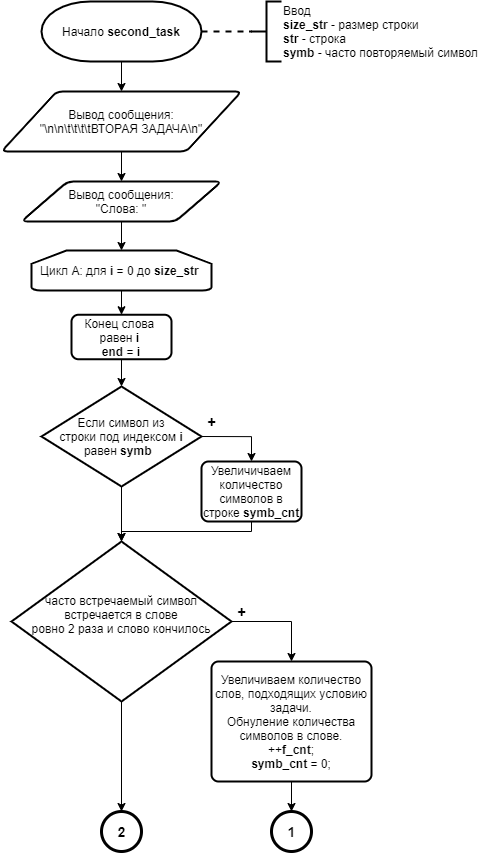
char&symb // массив symb

)

1. Обращение
2. second\_task(size\_str, str, keys[index\_om]);

Описание параметров

|  |  |  |  |
| --- | --- | --- | --- |
| Индикатор | Тип | Назначение | Входной/Выходной |
| symb | символ | самый часто встречаемый символ | входной |
| size\_str | целочисленный | размер строки | входной |
| str | строка | строка | входной |



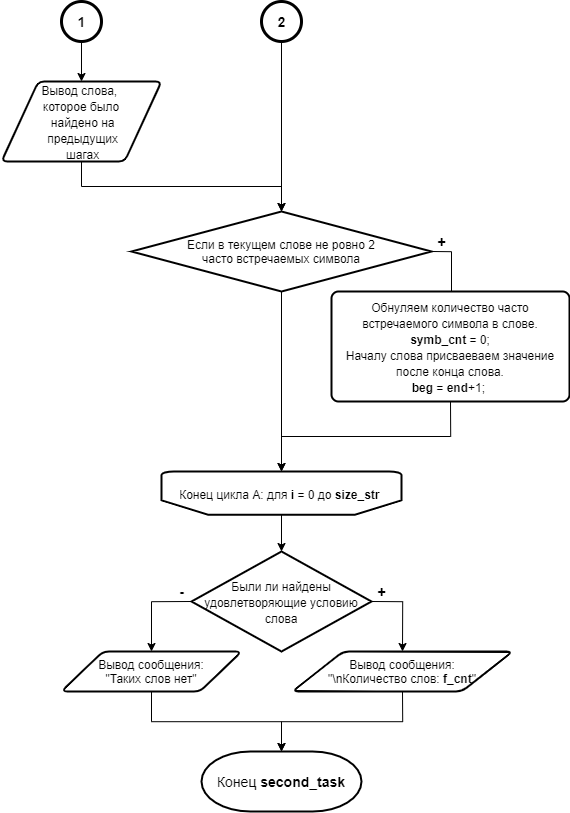


Рисунок 3. Структурная схема алгоритма функции second\_task().

# Кодпрограммы

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\  
\*                    кафедра № 304 1 курс 2 семестр информатика         \*  
\*-----------------------------------------------------------------------\*  
\*   Projecttype :solution                                             \*  
\*   Projectname : Simvolnuyedanniye                                   \*  
\*   Filename    :Simvolnuye danniye.cpp                               \*  
\*   Language     : c/c++                                                \*  
\*   Programmers  : Плоцкий Б.А. Раужев Ю. М.                            \*  
\*   Created      :  9/03/22                                             \*  
\*   Lastrevision:  23/03/22                                            \*  
\*   Comment(s)   : Вариант № 10                                         \*  
\*                                                                       \*  
\*       Исходные данные представляют собой строку символов.             \*  
\*   Написать программу, включающую две процедуры,                       \*  
\*   выполняющие следующие действия:                                     \*  
\*   1) выявление символа наиболее часто встречающегося в                \*  
\*       исходной строке, печать числа вхождений;                        \*  
\*   2) печать тех слов строки, в которых найденный символ               \*  
\*       встречается два раза.                                           \*  
\*       Чтение данных из файла производить с использованием функций     \*  
\*   ввода/вывода языка C++.                                             \*  
\*       Алгоритм должен быть параметризирован; обмен данными с          \*  
\*   подпрограммой должен осуществляться только через параметры;         \*  
\*   каждый из наборов исходных данных хранится в отдельном файле.       \*  
\*                                                                       \*  
\\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
   
#include<iostream>  
#include<fstream>  
#include<iomanip>  
   
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\*      И М Е Н А  Ф А Й Л О В  С  И Н Ф О Р М А Ц И Е Й         \*  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
   
///////////////////////////////////  
// Файлы с некорректными данными //  
///////////////////////////////////  
   
//      отсутствие файла****const******char****\* file\_name = "test/incorrect/1.txt";  
//      отсутствие данных в файле  
//constchar\* file\_name = "test/incorrect/2.txt";  
   
/////////////////////////////////  
// Файлы с корректными данными //  
/////////////////////////////////  
   
//      выполнение всех условий задачи  
//constchar\* file\_name = "test/correct/2.txt";  
//  
//      отсутствие слов, в которых часто  
//      встречаемый символ встречается два раза  
//constchar\* file\_name = "test/correct/1.txt";  
//  
//      нет повторяющихся элементов  
//constchar\* file\_name = "test/correct/4.txt";  
//  
//      всего один символ в строке  
//constchar\* file\_name = "test/correct/3.txt";  
//  
//      несколько символов с одинаковым (максимальным)  
//      числом повторений  
//constchar\* file\_name = "test/correct/5.txt";  
//  
//      все символы с одинаковым  
//      числом повторений  
//constchar\* file\_name = "test/correct/6.txt";  
//constchar\* file\_name = "test/correct/7.txt";  
   
   
// класс перечислений необходимый  
// для выбора места считывания:  
// file - считывание из файла  
// console - считывание из консоли****enum******class****input\_type  
{  
    file = 0,  
    console  
};****using******namespace****std;  
   
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\*              П Р О Т О Т И П Ы   Ф У Н К Ц И Й                \*  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
   
   
// функция ищет индекс наиболее часто встречаемого элемента****int****ind\_often\_meeting(****int****&size\_kv,   // размер массивов ключей и значений****char****\*&keys,    // массив ключей****int****\*&values    // массив значений  
);  
   
// шаблонная функция для увеличения массива на add\_cell ячеек  
template<typename T>****void****inc\_size(****int****&size,          // размер массива arr  
    T\*&arr,            // массив arr****int****add\_cell = 1    // количество новых ячеек; изначально оно равно 1  
);  
   
// расчет часто встречаемых символов****void****often\_met(****int****&size\_kv,   // размер массивов keys и values****char****\*&keys,    // массив keys****int****\*&values,   // массив values****char****\*&str,     // массив str****int****&size\_str   // размер массив str  
);  
   
// шаблонная функция печати массивов  
template<typename T>****void****print\_arr(  
    T\*&arr,            // массив введенных элементов****int****length,         // длина массива arr****int****space = 5,      // размер отступов между словами****bool****wrap =****true****// флаг для оборачивания элементов  
);  
   
// функция, считывающая данные из введенного потока ввода****void****stream\_read(  
    istream&stream,    // поток ввода данных с консоли/файла****int****&size,          // размер массива str****char****\*&str          // массив str  
);  
   
// функция печатает сообщение об ошибке  
// в зависимости от кода самой ошибки****void****print\_error\_msg(****int****code    // код ошибки  
);  
   
// функция проверки информации из файла****int****check\_file(  
    ifstream&stream    // имя потока ввода информации  
);  
   
// функция ввода с файла****void****file\_read(****int****&size,              // размер массива str****char****\*&str              // массив str  
);  
   
// проверка строки на наличие ошибок****int****check\_str(****int****&size,  // размер строки****char****\*&str  // строки  
);  
   
// функция ввода с консоли****void****console\_read(****int****&size\_str,      // размер массива str****char****\*&str          // массив str  
);  
   
// выбор места считывания информации****void****read\_str(  
    input\_typeinput,   // считывание из файла/консоли****int****&size\_str,      // размер массива str****char****\*&str          // массив str  
);  
   
// функция, решающая первый пункт задачи****int****first\_task(****int****&size\_kv,       // размер массива keys****char****\*&keys,        // массив keys****int****\*&values,       // массив values****int****&size\_str,      // размер массива str****char****\*&str,         // массив str  
    input\_typeinput    // чтение из файла/консоли  
);  
   
// функция, решающая втрой пункт задачи****void****second\_task(****int****&size\_str,      // размер массива str****char****\*&str,         // массив str****char****&symb          // массив symb  
);  
   
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\*                Г Л А В Н А Я   Ф У Н К Ц И Я                  \*  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/****int****main()  
{  
    // смена кодовой страницы консоли  
    system("chcp 1251&cls");  
   
    // размер массива строки****int****size\_str = 0;  
   
    // массив строки****char****\* str =****new******char****[size\_str];  
   
    // индекс часто встречаемого символа****int****index\_om = 0;****int****size\_kv = 0;                // размеры массивов ключей и значений****char****\* keys =****new******char****[size\_kv]; // массив ключей****int****\* values =****new******int****[size\_kv]; // массив значений  
   
    // переменная, устанавливающая место считывания  
    // из файла или из консоли  
    input\_type****in****= input\_type::file;  
   
    // выбор места считывания информации  
    read\_str(****in****, size\_str, str);  
   
    // функция, решающая первую задачу  
    index\_om = first\_task(size\_kv, keys, values, size\_str, str,****in****);  
   
    // функция, решающая вторую задачу  
    second\_task(size\_str, str, keys[index\_om]);****return****0;  
}  
   
// функция ищет индекс наиболее часто встречаемого элемента****int****ind\_often\_meeting(****int****&size\_kv,   // размер массивов ключей и значений****char****\*&keys,    // массив ключей****int****\*&values    // массив значений  
)  
{  
    // индекс наиболее часто встречаемого элемента****int****id = 0;  
   
    // максимальное значение элемента  
    // с индексом id****int****\_max = values[id];  
   
    // цикл ищет индекс элемента с  
    // максимальным значением в массиве values****for****(****int****i = 0; i <size\_kv; i++)  
    {****if****(\_max<values[i])  
        {  
            // изменение значения максимума  
            \_max = values[i];  
            id = i;  
        }// endif  
    }// endfor i  
   
    // проверка на наличие такого же  
    // количества повторений у другого символа****for****(****int****i = 0; i <size\_kv; i++)  
    {****if****(values[i] == values[id] && i != id)****return****-1;  
    }// endfor i****return****id;  
}  
   
// шаблонная функция для увеличения массива на add\_cell ячеек  
template<typename T>****void****inc\_size(****int****&size,          // размер массива arr  
    T\*&arr,            // массив arr****int****add\_cell    // количество новых ячеек; изначально оно равно 1  
)  
{  
    // увеличение длины массива  
    size += add\_cell;  
   
    // создание нового массива  
    T\* new\_arr =****new****T[size];  
   
    // копирование данных в новый массив из старого****for****(****int****i = 0; i <size; i++)  
    {  
        new\_arr[i] = arr[i];  
    }// endfor i  
   
    // удаление старого массива  
    delete[] arr;  
   
    // присваивание старому массиву адресс памяти нового  
    arr = new\_arr;  
}  
   
// расчет часто встречаемых символов****void****often\_met(****int****&size\_kv,   // размер массивов keys и values****char****\*&keys,    // массив keys****int****\*&values,   // массив values****char****\*&str,     // массив str****int****&size\_str   // размер массив str  
)  
{****for****(****int****i = 0; i <size\_str; i++)  
    {  
        // флаг, определяющий,  
        // был ли добавлен символ в массив ключей****bool****exist = 0;  
   
        // цикл, проходящий по массиву ключей****for****(****int****j = 0; j <size\_kv; j++)  
        {  
            // если символ есть в массиве ключей  
            // увеличиваем соответствующее  
            // ему значение в массиве значений на один****if****(keys[j] == str[i])  
            {  
                values[j]++;  
                exist = 1;  
            }// endif  
        }// endfor j  
   
        // если текущий символ еще не был учтен****if****(!exist)  
        {  
            // увеличение массива ключей на один элемент  
            inc\_size(size\_kv, keys);  
   
            // из-за предыдущего шага изменилась  
            // длина массива keys,  
            // а values - нет => уменьшаем size\_kv на 1  
            size\_kv--;  
   
            // увеличение массива значений на один элемент  
            inc\_size(size\_kv, values);  
   
            // добавление нового символа  
            keys[size\_kv - 1] = str[i];  
            values[size\_kv - 1] = 1;  
        }// endif  
    }// endfor i  
}  
   
// шаблонная функция печати массивов  
template<typename T>****void****print\_arr(  
    T\*&arr,    // массив введенных элементов****int****length, // длина массива arr****int****space,  // размер отступов между словами****bool****wrap   // флаг для оборачивания элементов  
)  
{  
    // оборачивание массива в квадратные скобки, если wrap = true****if****(wrap)  
        cout<< "[";  
    // endif  
   
    // вывод самого массива****for****(****int****i = 0; i <length; i++)  
    {  
        // печатаем элементы массива с пробелами равными space.  
        // пробелы печатаются, если это не первый элемент****if****(i != 0)  
            cout<<setw(space) <<setfill(' ');  
   
        // печать кавычек, если wrap = true****if****(wrap)  
            cout<< "'";  
   
        // печать элемента  
        cout<<arr[i];  
   
        // печать кавычек, если wrap = true****if****(wrap)  
            cout<< "'";  
    }// endfor i  
   
    // оборачивание массива в квадратные скобки, если wrap == true****if****(wrap)  
        cout<< "]";  
    // endif  
   
    // перенос строки  
    cout<< "\n";  
}  
   
// функция, считывающая данные из введенного потока ввода****void****stream\_read(  
    istream&stream,    // поток ввода данных с консоли/файла****int****&size,          // размер массива str****char****\*&str          // массив str  
)  
{  
    // символ для считвания данных****char****temp;  
   
    // считывание информации из консоли****do*** *{  
        // считывание символа  
        temp = stream.****get****();  
   
        // не конец строки ли файла или строки****if****(temp != EOF &&temp != '\n')  
        {  
            // увеличение длины массива  
            inc\_size(size, str);  
   
            // добавление нового элемента в массив  
            str[size - 1] = temp;  
        }// endif  
   
    }****while****(!stream.eof() &&temp != '\n' &&temp != EOF);  
    // endwhile  
}  
   
// функция печатает сообщение об ошибке  
// в зависимости от кода самой ошибки****void****print\_error\_msg(****int****code    // код ошибки  
)  
{****switch****(code)  
    {  
        // отсутствие файла****case****1:  
        cout<< "[ОШИБКА] Файл не существует.\  
 Возможно, неверно указан путь или имя файла.\n";  
        exit(1);****break****;  
   
        // отсутствие данных в файле****case****2:  
        cout<< "[ОШИБКА] Файл пуст.\n";  
        exit(2);  
   
        // не ввели строку****case****3:  
        cout<< "[ОШИБКА] Строка не введена.\n";  
        exit(3);****break****;  
    }// endswitchcode  
}  
   
// функция проверки информации из файла****int****check\_file(  
    ifstream&stream    // имя потока ввода информации  
)  
{  
    // файл не был открыт****if****(!stream.is\_open())  
    {****return****1;  
    }// endif  
   
    // проверка следующего элемента,  
    // не извлекая его****char****temp = stream.peek();  
   
    // файл пуст****if****(temp == EOF)  
    {****return****2;  
    }// endif  
   
    // ошибок не найдено****return****0;  
}  
   
// функция ввода с файла****void****file\_read(****int****&size,              // размер массива str****char****\*&str              // массив str  
)  
{  
    // файл  
    ifstreamfin(file\_name);  
   
    // проверка файла на его существование  
    // и на существование данных в нем  
    print\_error\_msg(check\_file(fin));  
   
    // считывание строки  
    stream\_read(fin, size, str);  
}  
   
// проверка строки на наличие ошибок****int****check\_str(****int****&size, // размер строки****char****\*&str // строки  
)  
{  
    // если строка отсутствует****if****(size == 0)  
    {****return****3;  
    }// endif  
   
    // ошибок не найдено****return****0;  
}  
   
// функция ввода с консоли****void****console\_read(****int****&size\_str,      // размер массива str****char****\*&str          // массив str  
)  
{  
    // считывание строки  
    stream\_read(cin, size\_str, str);  
   
    // проверка введенной строки  
    print\_error\_msg(check\_str(size\_str, str));  
}  
   
// выбор места считывания информации****void****read\_str(  
    input\_typeinput,   // считывание из файла/консоли****int****&size\_str,      // размер массива str****char****\*&str          // массив str  
)  
{  
    // выбор места считывания строки****switch****(input)  
    {  
        // считывание информации из файла****case****input\_type::file:  
        // вывод диалогового сообщения  
        cout<< "Ввод данных производится из файла.\n";  
   
        // само считывание из файла  
        file\_read(size\_str, str);****break****;  
   
        // считывание информации из консоли****case****input\_type::console:  
        // вывод диалогового сообщения  
        cout<< "Ввод данных производится из консоли.\n Введите строку: ";  
   
        // само считывание из консоли  
        console\_read(size\_str, str);****break****;  
    }  
}  
   
// функция, решающая первый пункт задачи****int****first\_task(****int****&size\_kv,       // размер массива keys****char****\*&keys,        // массив keys****int****\*&values,       // массив values****int****&size\_str,      // размер массива str****char****\*&str,         // массив str  
    input\_typeinput    // чтение из файла/консоли  
)  
{  
    cout<< "\n\t\t\t\tПЕРВАЯ ЗАДАЧА\n";  
    // определение часто встречаемого символа в строке  
    often\_met(size\_kv, keys, values, str, size\_str);  
   
    // эхопечать строки  
    cout<< "\nВведенная строка: ";  
    print\_arr(str, size\_str, 0, 0);  
   
    //// эхопечать двух массивов  
    //cout<< "\nkeys:   ";  
    //print\_arr(keys, size\_kv);  
    //cout<< "values: ";  
    //print\_arr(values, size\_kv);  
    //cout<< '\n';  
   
    // вывод часто встречаемого элемента****int****index = ind\_often\_meeting(size\_kv, keys, values);****if****(index == -1)  
    {  
        cout<< "Часто встречающийся элемент не был найден.\n";  
    }// endif****else*** *{  
        cout<< "Часто встречающийся элемент: [" <<keys[index] << "] \n"  
            << "Столько раз он был в строке: " <<values[index] <<endl;  
    }// endelse****return****index;  
}  
   
// функция, решающая втрой пункт задачи****void****second\_task(****int****&size\_str,      // размер массива str****char****\*&str,         // массив str****char****&symb          // элемент symb  
)  
{  
    cout<< "\n\n\t\t\t\tВТОРАЯ ЗАДАЧА\n";****int****f\_cnt = 0;          // кол-во слов удов.условию****int****symb\_cnt = 0;       // кол-во часто встречаемого  
                            // символа в текущем слове****int****beg = 0;            // начало/конец слова****int****i = 0;              // итератор цикла  
   
    // вывод слов удов. условию  
    cout<< "Слова: ";  
   
    // цикл вывода слов удов.условию****for****(i; i <= size\_str; i++)  
    {  
        // проверка, что самый встречаемый сивол находится в слове****if****(str[i] == symb)  
        {  
            ++symb\_cnt;  
        }// endif  
   
        // проверка, что часто встречаемый символ встречается в слове  
        // ровно 2 раза и слово кончилось****if****(symb\_cnt == 2 && (str[i] == ' ' || i == size\_str ))  
        {  
            ++f\_cnt;  
            symb\_cnt = 0;  
   
            // вывод слова при удов.условию  
            cout<< " [";****for****(beg; beg< i; beg++)  
            {  
                cout<<str[beg];  
            }  
            cout<<  "] ";  
   
            beg++;  
        }// endif  
   
        // если в слове не 2 самых часто встречаемых символов****else******if****(symb\_cnt != 2 && (str[i] == ' ' || i == size\_str))  
        {  
            symb\_cnt = 0;  
            beg = i + 1;  
        }// endelseif  
    }// endfor i  
   
    // Проверка были ли найдены удов.слова****if****(f\_cnt == 0)  
    {  
        cout<< "Таких слов нет" <<endl<<endl;  
    }// endif****else*** *{  
        cout<< "\nКоличество слов: " <<f\_cnt<<endl<<endl;  
    }// endelse  
}// endmain  
   
/\*\*\*\*\*\*\*\*\*\*    EndofSimvolnuye danniye.cpp file   \*\*\*\*\*\*\*\*\*\*/*

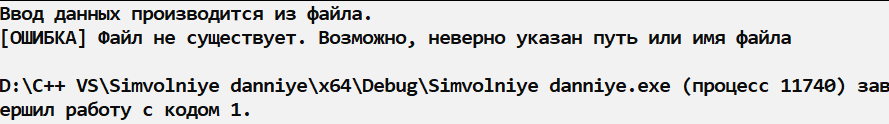
# Тестирование программы

## Тестирование некорректных режимов

### Тест 1

**Цель:** Проверить работу программы при несуществующем файле.

**Исходные данные:** файл:test/incorrect/1.txt

**Ожидаемый Результат:** Вывод сообщения об ошибке: Файл не существует. Возможно, неверно указан путь или имя файла.

**Полученный результат:**

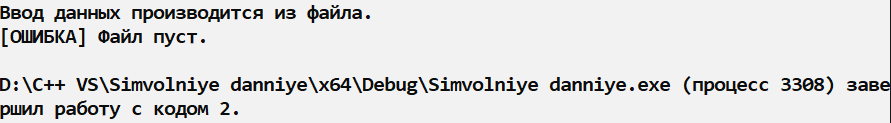
**Вывод по тесту:** Ожидаемый результат совпал с полученным результатом. Тест ошибок не обнаружил.

### Тест 2

**Цель:** Проверить работу программы при отсутствии данных в файле.

**Исходные данные:** файл: test/incorrect/2.txt

**Ожидаемый Результат:** Вывод сообщения об ошибке: Файл пуст.

**Полученный результат:**

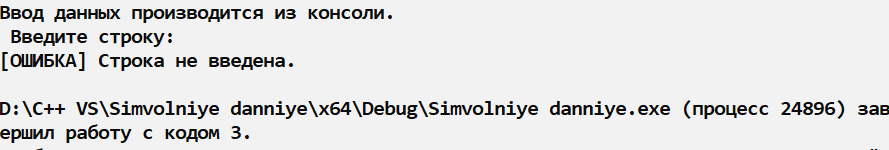
**Вывод по тесту:** Ожидаемый результат совпал с полученным результатом. Тест ошибок не обнаружил.

### Тест 3

**Цель:** Проверить работу программы при отсутствии введенных данных в консоли.

**Исходные данные:**

**Ожидаемый Результат:** Вывод сообщения об ошибке: Строка не введена.

**Полученный результат:**

**Вывод по тесту:** Ожидаемый результат совпал с полученным результатом. Тест ошибок не обнаружил.

## Тестирование корректных режимов

### Тест 1

**Цель:** Проверить работу программы при выполнении всех условий задачи.

**Исходные данные:** файл: test/correct/2.txt

**Ожидаемый Результат:** Вывод сообщения:

ПЕРВАЯ ЗАДАЧА

Введенная строка: abcdddaskldfd;slf;sd d ddddssdddd

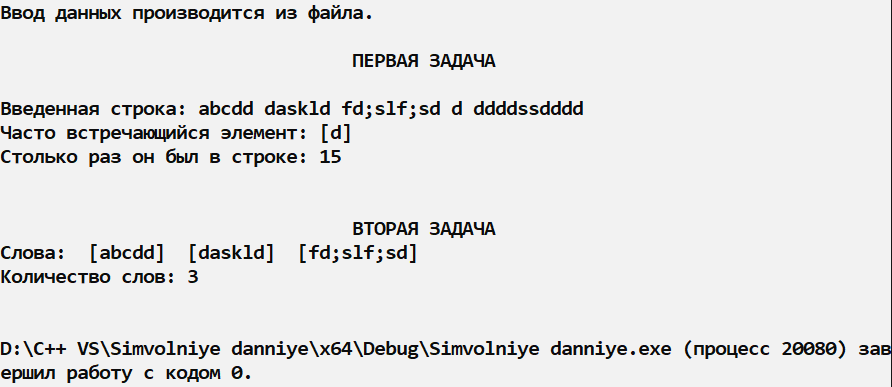
Часто встречающийся элемент: [d]

Столько раз он был в строке: 15

ВТОРАЯ ЗАДАЧА

Слова: [abcdd] [daskld] [fd;slf;sd]

Количество слов: 3

**Полученный результат:**

**Вывод по тесту:** Ожидаемый результат совпал с полученным результатом. Тест ошибок не обнаружил.

### Тест 2

**Цель:** Проверить работу программы, где не выполняется 2-е условие.

**Исходные данные:** файл: test/correct/1.txt

**Ожидаемый Результат:** Вывод сообщения:

ПЕРВАЯ ЗАДАЧА

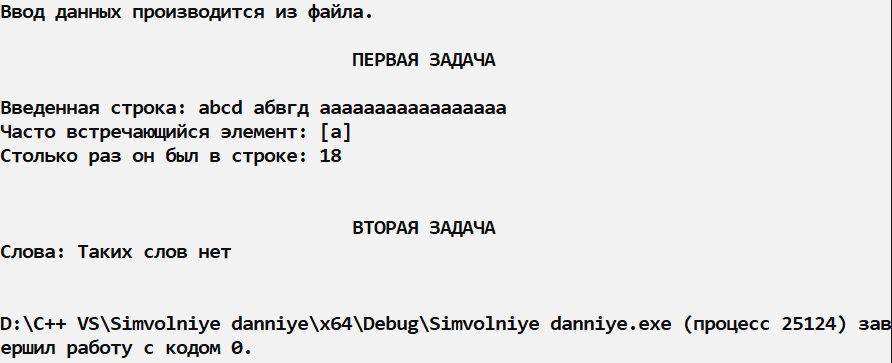
Введенная строка: abcdабвгдааааааааааааааааа

Часто встречающийся элемент: [а]

Столько раз он был в строке: 18

ВТОРАЯ ЗАДАЧА

Слова: Таких слов нет

**Полученный результат:**

**Вывод по тесту:** Ожидаемый результат совпал с полученным результатом. Тест ошибок не обнаружил.

### Тест 3

**Цель:** Проверить работу программы, где нет повторяющихся элементов.

**Исходные данные:** файл: test/correct/4.txt

**Ожидаемый Результат:** Вывод сообщения:

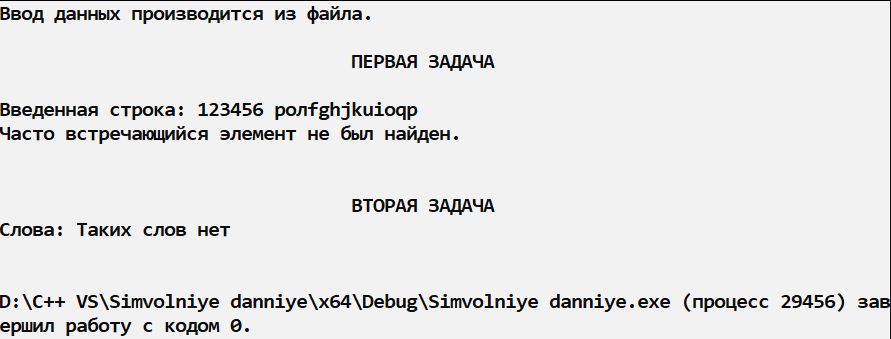
ПЕРВАЯ ЗАДАЧА

Введенная строка: 123456 ролfghjkuioqp

Часто встречающийся элемент не был найден.

ВТОРАЯ ЗАДАЧА

Слова: Таких слов нет

**Полученный результат:**

**Вывод по тесту:** Ожидаемый результат совпал с полученным результатом. Тест ошибок не обнаружил.

### Тест 4

**Цель:** Проверить работу программы, где всего один символ в строке.

**Исходные данные:** файл: test/correct/3.txt

**Ожидаемый Результат:** Вывод сообщения:

Ввод данных производится из файла.

ПЕРВАЯ ЗАДАЧА

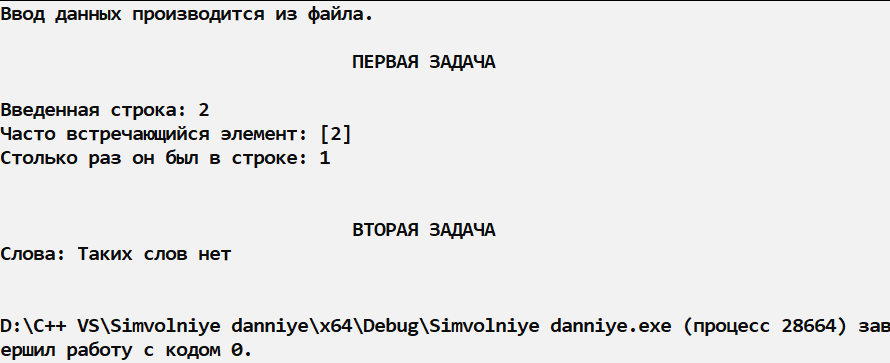
Введенная строка: 2

Часто встречающийся элемент: [2]

Столько раз он был в строке: 1

ВТОРАЯ ЗАДАЧА

Слова: Таких слов нет

**Полученный результат:**

**Вывод по тесту:** Ожидаемый результат совпал с полученным результатом. Тест ошибок не обнаружил.

### Тест 5

**Цель:** Проверить работу программы, где несколько символов с максимальным и одинаковым числом повторений.

**Исходные данные:** файл: test/correct/5.txt

**Ожидаемый Результат:** Вывод сообщения:

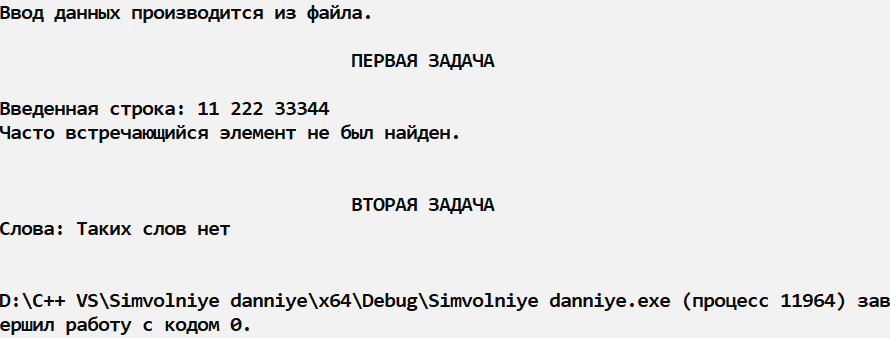
ПЕРВАЯ ЗАДАЧА

Введенная строка: 11 222 33344

Часто встречающийся элемент не был найден.

ВТОРАЯ ЗАДАЧА

Слова: Таких слов нет

**Полученный результат:**

**Вывод по тесту:** Ожидаемый результат совпал с полученным результатом. Тест ошибок не обнаружил.

### Тест 6

**Цель:** Проверить работу программы, где все символы с одинаковым числом повторений.

**Исходные данные:** файл: test/correct/6.txt

**Ожидаемый Результат:** Вывод сообщения:

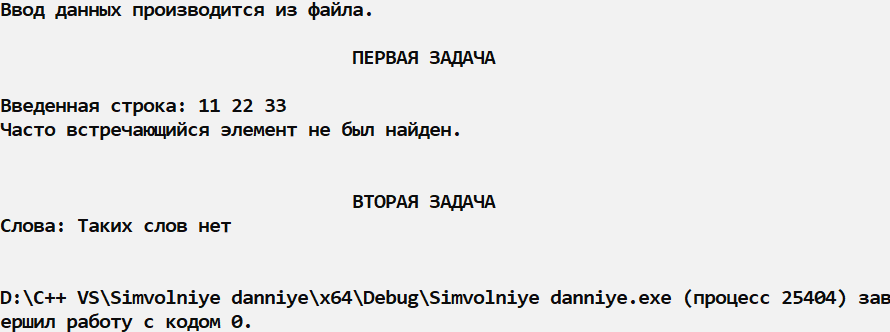
ПЕРВАЯ ЗАДАЧА

Введенная строка: 11 22 33

Часто встречающийся элемент не был найден.

ВТОРАЯ ЗАДАЧА

Слова: Таких слов нет

**Полученный результат:**

**Вывод по тесту:** Ожидаемый результат совпал с полученным результатом. Тест ошибок не обнаружил.

### Тест 7

**Цель:** Проверить работу программы при выполнении всех условий задачи.

**Исходные данные:**abcdddaskldfd;slf;sd d ddddssdddd

**Ожидаемый Результат:** Вывод сообщения:

ПЕРВАЯ ЗАДАЧА

Введенная строка: abcdddaskldfd;slf;sd d ddddssdddd

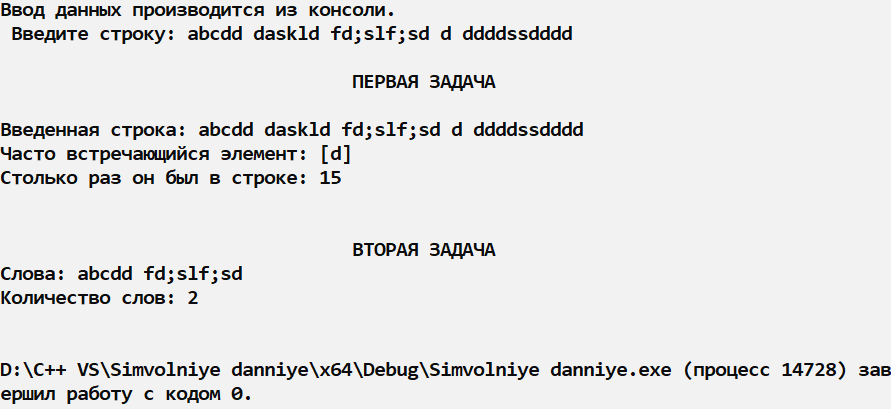
Часто встречающийся элемент: [d]

Столько раз он был в строке: 15

ВТОРАЯ ЗАДАЧА

Слова: abcddfd;slf;sd

Количество слов: 2

**Полученный результат:**

**Вывод по тесту:** Ожидаемый результат совпал с полученным результатом. Тест ошибок не обнаружил.

### Тест 8

**Цель:** Проверить работу программы, где не выполняется 2-е условие.

**Исходные данные:**abcdабвгдааааааааааааааааа

**Ожидаемый Результат:** Вывод сообщения:

ПЕРВАЯ ЗАДАЧА

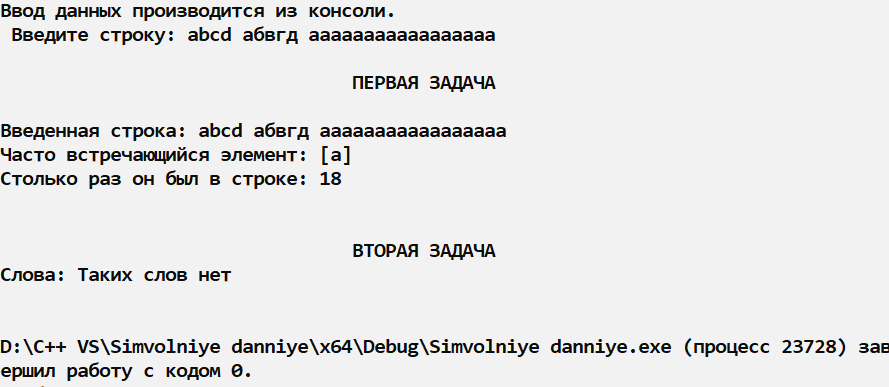
Введенная строка: abcdабвгдааааааааааааааааа

Часто встречающийся элемент: [а]

Столько раз он был в строке: 18

ВТОРАЯ ЗАДАЧА

Слова: Таких слов нет

**Полученный результат:**

**Вывод по тесту:** Ожидаемый результат совпал с полученным результатом. Тест ошибок не обнаружил.

### Тест 9

**Цель:** Проверить работу программы, где нет повторяющтхся элементов.

**Исходные данные:**123456 ролfghjkuioqp

**Ожидаемый Результат:** Вывод сообщения:

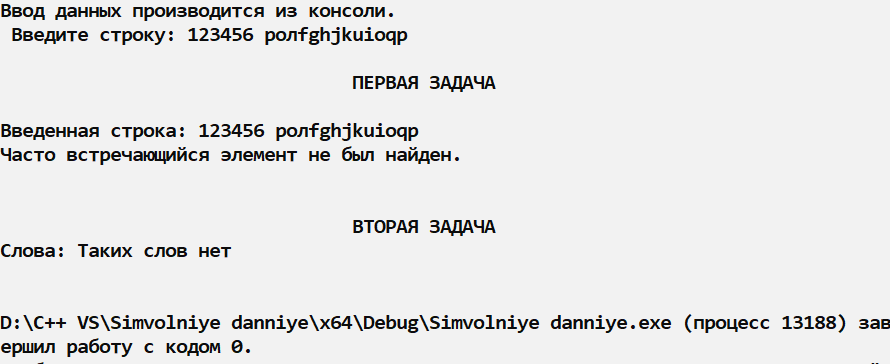
ПЕРВАЯ ЗАДАЧА

Введенная строка: 123456 ролfghjkuioqp

Часто встречающийся элемент не был найден.

ВТОРАЯ ЗАДАЧА

Слова: Таких слов нет

**Полученный результат:**

**Вывод по тесту:** Ожидаемый результат совпал с полученным результатом. Тест ошибок не обнаружил.

### Тест 10

**Цель:** Проверить работу программы, где всего один символ в строке.

**Исходные данные:**2

**Ожидаемый Результат:** Вывод сообщения:

Ввод данных производится из консоли.

ПЕРВАЯ ЗАДАЧА

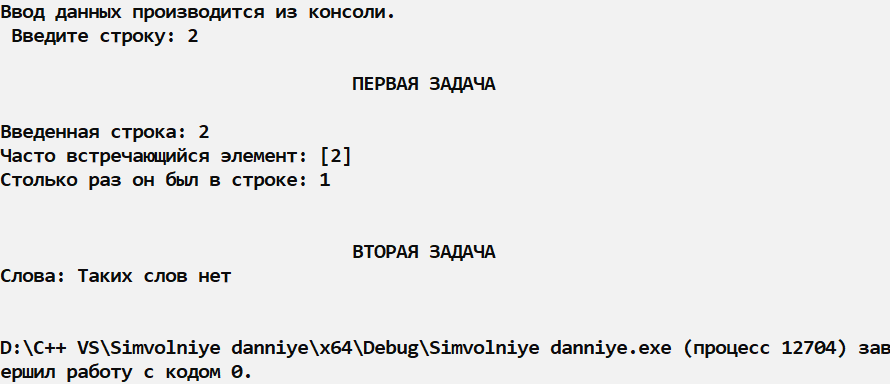
Введенная строка: 2

Часто встречающийся элемент: [2]

Столько раз он был в строке: 1

ВТОРАЯ ЗАДАЧА

Слова: Таких слов нет

**Полученный результат:**

**Вывод по тесту:** Ожидаемый результат совпал с полученным результатом. Тест ошибок не обнаружил.

### Тест 11

**Цель:** Проверить работу программы, где несколько символов с максимальным и одинаковым числом повторений.

**Исходные данные:**11 222 33344

**Ожидаемый Результат:** Вывод сообщения:

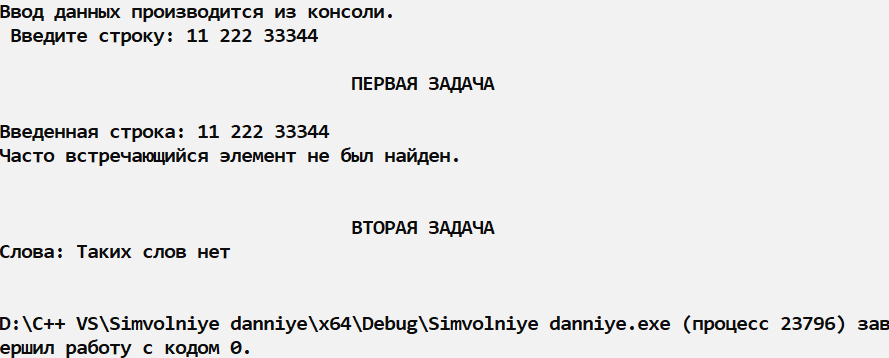
ПЕРВАЯ ЗАДАЧА

Введенная строка: 11 222 33344

Часто встречающийся элемент не был найден.

ВТОРАЯ ЗАДАЧА

Слова: Таких слов нет

**Полученный результат:**

**Вывод по тесту:** Ожидаемый результат совпал с полученным результатом. Тест ошибок не обнаружил.

### Тест 12

**Цель:** Проверить работу программы, где все символы с одинаковым числом повторений.

**Исходные данные:**11 22 33

**Ожидаемый Результат:** Вывод сообщения:

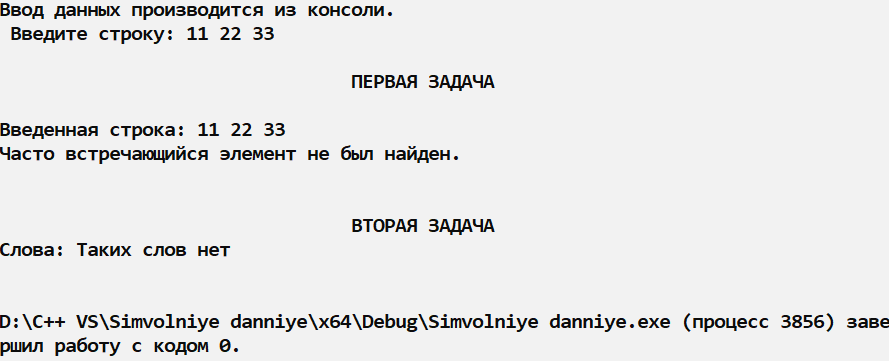
ПЕРВАЯ ЗАДАЧА

Введенная строка: 11 22 33

Часто встречающийся элемент не был найден.

ВТОРАЯ ЗАДАЧА

Слова: Таких слов нет

**Полученный результат:**

**Вывод по тесту:** Ожидаемый результат совпал с полученным результатом. Тест ошибок не обнаружил.

# Вывод

Работа программы завершена на основании:

1) Полученные результаты совпали с ожидаемыми;

2) Считаем набор тестов полным.