МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ  
федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА № 43

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ассистент |  |  |  | М.А. Мурашова |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ № 7 |
| «Обработка текстовых данных» |
| по курсу: Основы программирования |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4134К |  |  |  | Опарин С.Н. |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2022

1. **Цель работы:**

Целью работы является изучение представления строковых данных в языке С и алгоритмов их обработки

1. **Задание**

Разработать функцию, которая выполняет ту обработку символьной строки, которая определена в Вашем индивидуальном задании.



1. **Описание созданных функций**

Для реализации задания нам потребуется следующие функции:

**Имя:** get\_string

**Назначение:** формирование строки

**Входные данные:** текст

**Выходные данные:** сформированная строка

**Побочный эффект:** отсутствует

**Псевдокод**:

Ввод переменной для запоминания

Объявление символьного массива

Получение первого элемента строки ввода

Цикл с условием

Присваивание взятого символа из ввода в элемент массива

Если текущая длинна взятых элементов ввода длиннее переменной для запоминания расширение массива символов

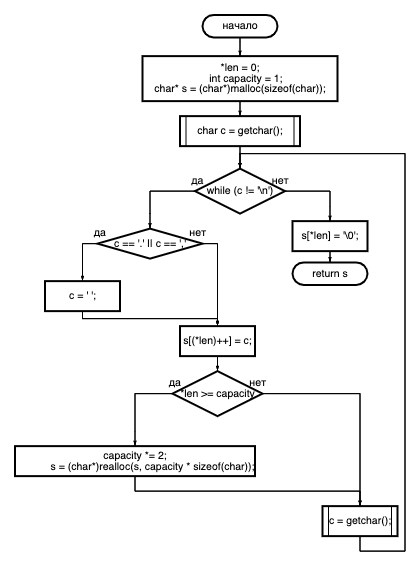
Взятие следующего элемента из строки ввода

Возврат массива символов

**Тестовые данные:**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| Любые данные | Любые данные |

**Блок-схема:**

****

**Имя:** SubWord

**Назначение:** Подсчёт необходимых слов и вывод их на экран

**Входные данные:** const char\* str, int n, int l, int strLength

**Выходные данные:** отсутствуют

**Побочный эффект:** отсутствует

**Псевдокод:**

Объявление переменных счётчиков

Цикл с посимвольным перебором строки и нахождением окончания слов

Цикл до длинны строки

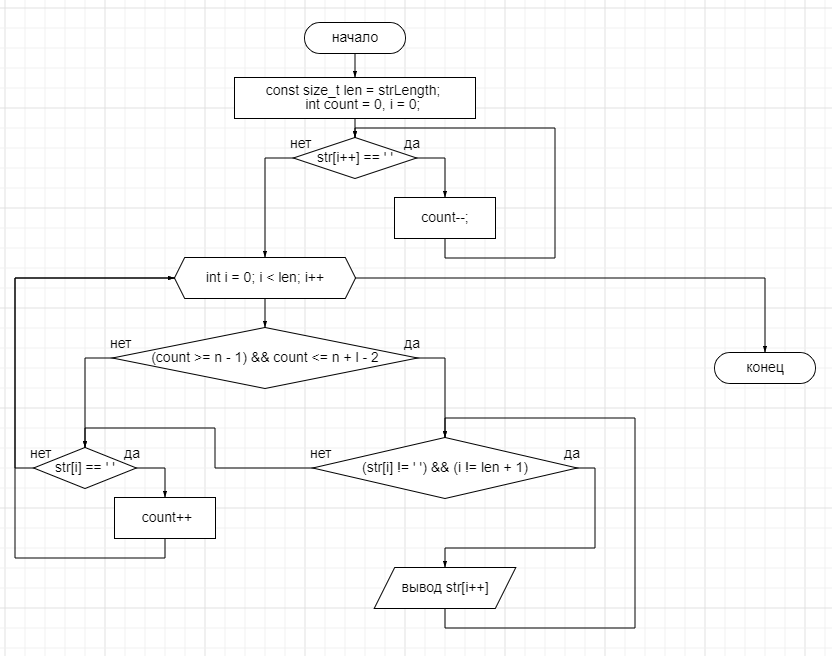
Условие если слово попадает в выбранный промежуток вывести его посимвольно

**Прототип:** void SubWord(const char\* str, int n, int l, int strLength)

**Тестовые данные:**

|  |  |
| --- | --- |
| **Входные данные** | **Выходные данные** |
| **Отсутствуют** | **Отсутствуют** |

**Блок-схема:**



**Имя:** inputWordIndex

**Назначение:** проверка на корректность данных

**Входные данные:** int col

**Выходные данные:** n

**Побочный эффект:** отсутствует

**Псевдокод:**

Объявление переменных

Пока правда

Ввод слова

Если ввод некорректный, то

Очисти ввод

Скажи, что неправильный ввод

Иначе если номер меньше или равен нулю, то

Скажи, что номер должен быть больше

Иначе если номер выходит за границы, то

Скажи, что слова с таким номером не существует

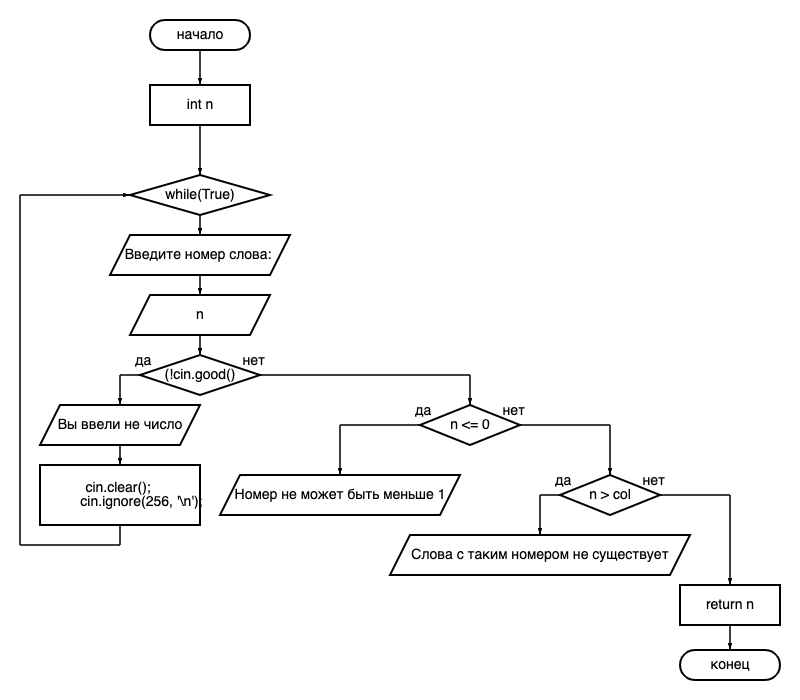
Иначе Выйти

**Прототип:** int inputWordIndex(int col)

**Тестовые данные:**

|  |  |
| --- | --- |
| **Входные данные** | **Выходные данные** |
| **Число** | **Число** |
| **Не число** | **Ошибка** |

**Блок-схема:**

****

**Имя:** inputWordLength

**Назначение:** Пользовательский ввод

**Входные данные:** int col, int n

**Выходные данные:** l

**Побочный эффект:** отсутствует

**Псевдокод:**

Объявление переменной l

Бесконечный цикл

Ввод значения и проверки на корректность

Конец цикла

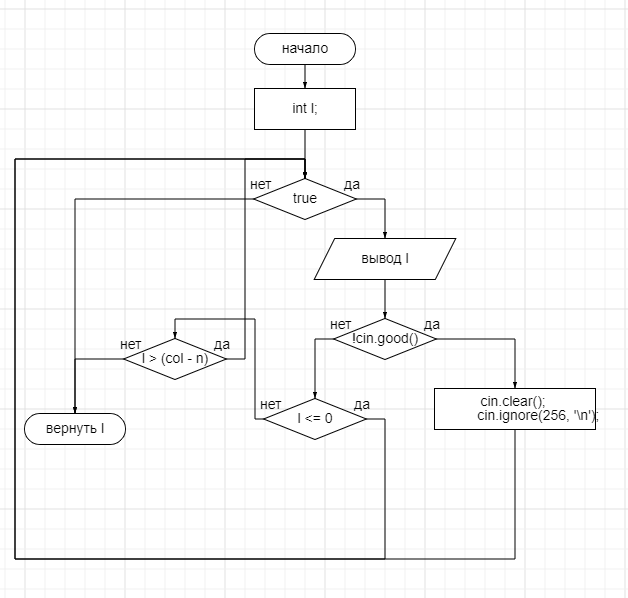
Возврат значения

**Прототип:** double task\_2(double\* arr[], int n)

**Тестовые данные:**

|  |  |
| --- | --- |
| **Входные данные** | **Выходные данные** |
| **Число** | **Число** |
| **Не число** | **Ошибка** |

**Блок-схема:**



**Имя:** main

**Входные данные:** нет

**Выходные данные:** нет

**Побочный эффект:** отсутствует

**Псевдокод:**

Ввод строки

Ввод номера слова для начала отсчёта

Ввод количества слов

Функция подсчёта слов

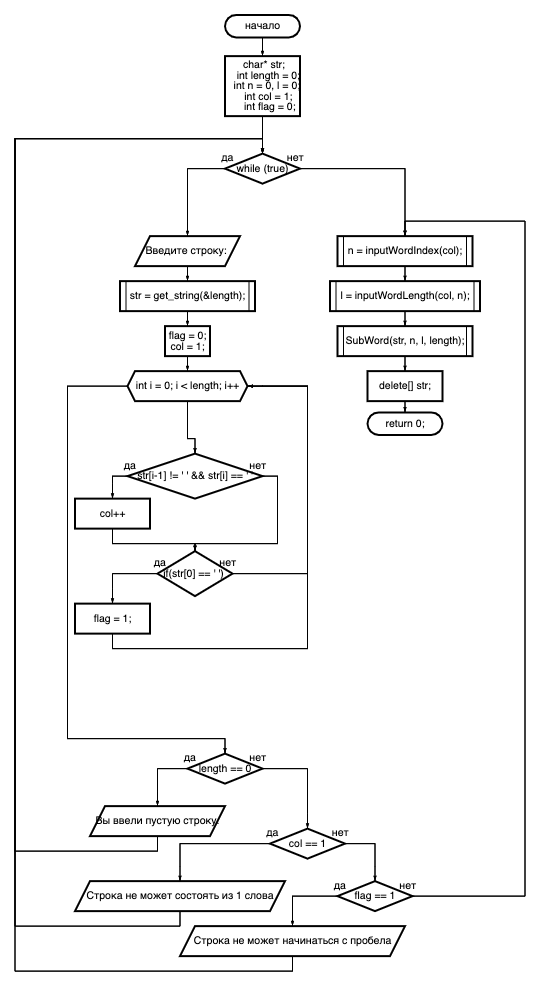
Вывод результата

**Прототип:** int main()

**Тестовые данные:**

|  |  |  |  |
| --- | --- | --- | --- |
| Входные данные | Номер слова | Количество слов к выбору | Ответ |
| Миша Маша Дима | **1** | **2** | Миша Маша |

**Блок-схема:**

****

1. **Текст программы**

#include <iostream>

#include <cstdio>

#include <regex>

#define \_\_CRTDBG\_MAP\_ALLOC

//#include <crtdbg.h>

#define DEBUG\_NEW new(\_NORMAL\_BLOCK, \_\_FILE\_\_, \_\_LINE\_\_)

#define new DEBUG\_NEW

using namespace std;

char\* get\_string(int\* len) {

\*len = 0;

int capacity = 1;

char\* s = (char\*)malloc(sizeof(char));

char c = getchar();

while (c != '\n'){

if(c == '.' || c == ','){

c = ' ';

}

s[(\*len)++] = c;

if (\*len >= capacity) {

capacity \*= 2;

s = (char\*)realloc(s, capacity \* sizeof(char));

}

c = getchar();

}

s[\*len] = '\0';

return s;

}

void SubWord(const char\* str, int n, int l, int strLength) {

int const len = strLength;

int count = 0, i = 0;

while (str[i] != ' ' && str[i++] == ' ')

count--;

cout << "Ваше слово(a): ";

for (int i = 0; i < len; i++) {

if ((count >= n - 1) && count <= n + l - 2) {

while ((str[i] != ' ') && (i != len + 1))

cout << str[i++];

cout << " ";

}

if (str[i-1] != ' ' && str[i] == ' ')

count++;

}

}

int inputWordIndex(int col) {

int n;

while (true) {

cout << "Введите номер слова: ";

cin >> n;

if (!cin.good()) {

cin.clear();

cin.ignore(256, '\n');

cout << "Вы ввели не число" << endl;

}

else if (n <= 0) cout << "Номер не может быть меньше 1" << endl;

else if (n > col) cout << "Слова с таким номером не существует." << endl;

else break;

}

return n;

}

int inputWordLength(int col, int n) {

int l;

while (true) {

cout << "Введите кол-во слов: ";

cin >> l;

if (!cin.good()) {

cin.clear();

cin.ignore(256, '\n');

cout << "Вы ввели не число" << endl;

}

else if (l <= 0) cout << "Вы не можете выбрать меньше 1 слова" << endl;

else if (l > (col - n+1)) cout << "Введенное кол-во слов для вывода больше, чем присутствует в строке." << endl;

else break;

}

return l;

}

int main()

{

setlocale(LC\_ALL, "Russian");

char\* str;

int length = 0;

int n = 0, l = 0; // Начало и количество слов к выбору

int col = 1; //Количество слов в строке

int flag = 0;

while (true) {

cout << "Введите строку: ";

str = get\_string(&length);

flag = 0;

col = 1;

for (int i = 0; i < length; i++)

if (str[i-1] != ' ' && str[i] == ' ') col++;

if(str[0] == ' ') flag = 1;

if (length == 0) cout << "Вы ввели пустую строку." << endl;

else if (col == 1) cout << "Строка не может состоять из 1 слова" << endl;

else if (flag == 1) cout << "Строка не может начинаться с пробела" << endl;

else break;

}

n = inputWordIndex(col);

l = inputWordLength(col, n);

SubWord(str, n, l, length);

delete[] str;

//\_CrtDumpMemoryLeaks();

return 0;

}

1. **Пример выполнения программы**

Ниже показан пример выполнения программы



Рис. 1 - Пример выполнения программы

Видно, что результаты расчетов совпадают с тестовыми данными.

1. **Анализ результатов и выводы**

В ходе этой лабораторной работы мы научились работать со строками (массивами char).

К достоинствам программы можно отнести:

* Решение каждого выражения реализовано в виде отдельной функции
* Есть проверка на корректность данных

Из недостатков можно отметить:

* Недостаточно оптимизирован код