ГУАП

КАФЕДРА № 43

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

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

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ |
| Обработка текстовых данных |
| по курсу: ОСНОВЫ ПРОГРАММИРОВАНИЯ |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4135К |  |  |  | А.А.Сенин |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2022

**1.Цель работы:** Целью работы является изучение представления строковых данных в языке С и алгоритмов их обработки.

**2.Задачи работы:** Разработать функцию, которая выполняет ту обработку символьной строки, которая определена в Вашем индивидуальном задании. Все утечки памяти должны быть устранены.

Функция находит последнее вхождение в строку заданной подстроки **3.Описание функций:**  
main-основная функция  
**1)Имя: main**  
**Назначение:** ввод пользователем строки текста и строки со словом, которое нужно найти

**Входные данные:** нет  
**Выходные данные:** нет  
**Побочный эффект:** отсутствует.

**Тестовые данные:**   
**Ввод: hello world world**

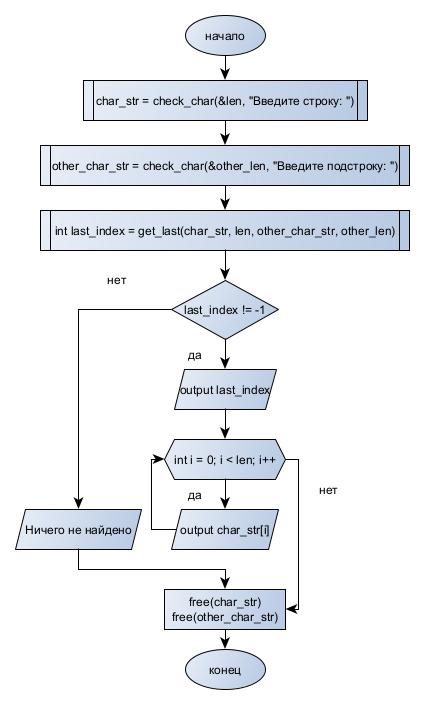
**world**

**Вывод:** **hello world [world]**  
**Прототип:** int main()  
**Псевдокод:**ввод первой строки пользователем

ввод второй строки пользователем

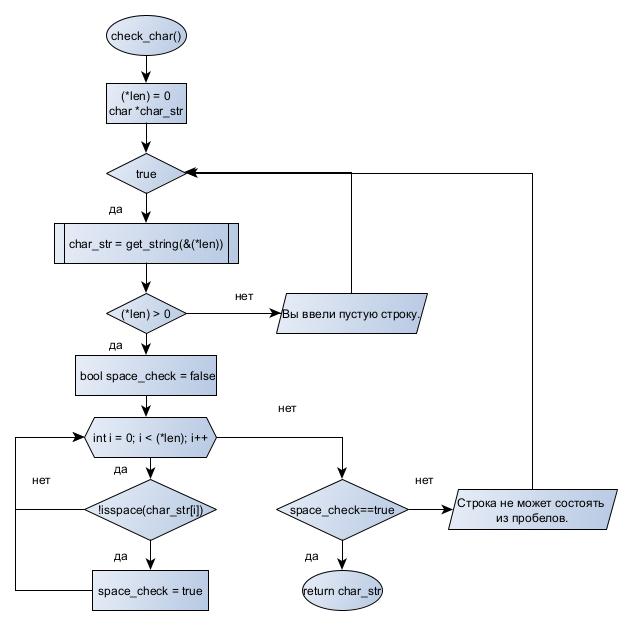
вызов функции для поиска строки

вывод строки (в скобках будет показана найденная строка)

очистка памяти  
**Блок-схема:**  
  
  
**2)Имя: check\_char()**  
**Назначение:** функция для ввода строки и проверки её на заполнение  
**Входные данные:** нет  
**Выходные данные:** char\_str  
**Побочный эффект:** отсутствует.   
**Прототип:** char\* check\_char()  
**Псевдокод:**Функция ввода строки

Если символов больше 0, то проверка на пробелы

Если количество пробелов=количество символов, то ввод строки заново

Если символов 0, то ввод строки заново  
**Блок-схема:**  
****

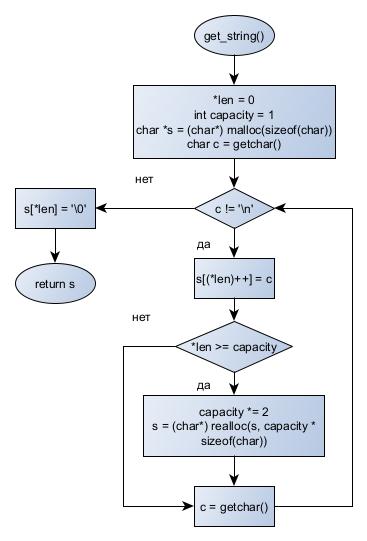
**3)Имя: get\_string()**  
**Назначение:** функция для ввода строки  
**Входные данные:** нет  
**Выходные данные:** s  
**Побочный эффект:** отсутствует.   
**Прототип:** char\* get\_string()  
**Псевдокод:**Создание массива

Получение символа

Если этот символ не равен ‘\n’, то запись в массив

Если равен ‘\n’, то конец функции

**Блок-схема:**

  
**4)Имя: get\_last()**  
**Назначение:** функция нахождения строки  
**Входные данные:** \*str, len, \*other\_char\_str, other\_len  
**Выходные данные:** i  
**Побочный эффект:** отсутствует.   
**Прототип:** int str\_find()  
**Псевдокод:**обратный цикл по всей строке

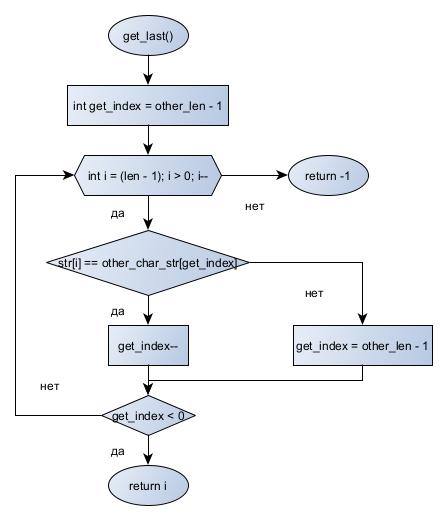
Если находит одинаковый символ, то запись

Если находит различные символы то сброс записи

Если нашли полностью содержимое второй строки, то возврат номера символа

если не нашли, возрвщаем -1

**Блок-схема:**

****

**5.ЛИСТИНГ КОДА:**/\*

Разработать функцию, которая выполняет ту обработку символьной строки, которая

определена в Вашем индивидуальном задании

Вариант 11

Функция находит последнее вхождение в строку заданной подстроки.

\*/

#include <iostream>

using namespace std;

#include "libs/lib.h"

#include <cmath>

#include <time.h>

// проверка ввода

#include "libs/simple\_char.h"

#include "more\_char.h"

int get\_last(char \*str, int len, char \*other\_char\_str, int other\_len) {

int get\_index = other\_len - 1;

// обратный цикл по всей строке

for (int i = (len - 1); i > 0; i--) {

if (str[i] == other\_char\_str[get\_index]) {

get\_index--;

} else {

get\_index = other\_len - 1;

}

// если нашли строку, то возвращаем её индекс

if (get\_index < 0) return i;

}

// если не нашли, возрвщаем -1

return -1;

}

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

\*len = 0;

int capacity = 1;

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

char c = getchar();

while (c != '\n') {

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

if (\*len >= capacity) {

capacity \*= 2;

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

}

c = getchar();

}

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

return s;

}

// функция для ввода строки и проверки её

char \*check\_char(int \*len, const char \*promt = "") {

(\*len) = 0;

char \*char\_str;

// ввод строки

while (true) {

cout << promt;

char\_str = get\_string(&(\*len));

if ((\*len) > 0) {

// проверка на пробелы (строка не должна состоять из пробелов)

bool space\_check = false;

for (int i = 0; i < (\*len); i++)

if (!isspace(char\_str[i])) {

space\_check = true;

break;

}

if (space\_check)

break;

else

cout << "Строка не может состоять из пробелов." << endl;

} else {

cout << "Вы ввели пустую строку." << endl;

}

}

return char\_str;

}

int main() {

// смена кодировки

system("chcp 65001");

// очистка терминала

//clear\_scr();

int len, other\_len;

char \*char\_str, \*other\_char\_str;

// вводим строки

char\_str = check\_char(&len, "Введите строку: ");

other\_char\_str = check\_char(&other\_len, "Введите подстроку: ");

draw\_line(20);

// вызов функции для поиска строки

int last\_index = get\_last(char\_str, len, other\_char\_str, other\_len);

if (last\_index != -1) {

cout << "Начало найденной строки (индекс): " << last\_index << endl;

// вывод строки (в скобках будет показана найденная строка)

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

cout << ((i == last\_index)? "[": "") << char\_str[i] << ((i == last\_index + (other\_len - 1))? "]": "");

}

cout << endl;

} else {

cout << "Ничего не найдено" << endl;

}

// очищаем память

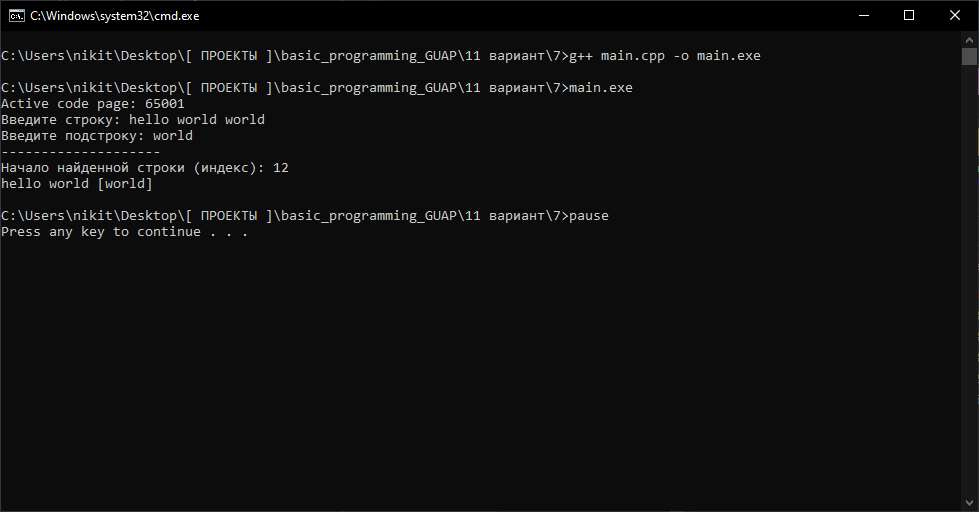
free(char\_str);

free(other\_char\_str);

return 0;

}

**6.Пример выполнения программы:**



**7.Анализ результатов и выводы:**

В ходе этой лабораторной работы мы изучили структуру данных одномерных массивов и перезаписывать их.

Из плюсов можно отметить, что поиск строки происходит корректно и строка выделяется в выводе.

Существенных минусов в программе не обнаружено.