**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра Вычислительной техники**

отчет

**по лабораторной работе № 5**

**по дисциплине «Программирование»**

Тема: **«Обработка строк»**

|  |  |  |
| --- | --- | --- |
| Студент гр. 3312 |  | Мохно Даниил. |
| Преподаватель |  | Аббас Саддам |

Санкт-Петербург

2023

**Цель работы.**

Целью работы является изучение обработки строк в языке Си и получение практических навыков в программировании на этом языке.

**Задание (вариант 5)**

Ввести строку текста, которая состоит из слов и произвольного количества символов-разделителей до и после слов. Массив символов-разделителей вводится после ввода строки. Количество символов-разделителей задается. Определить номер и вывести слово максимальной длины (если таких слов несколько, вывести данные по последнему слову).

**Постановка задачи и описание решения**

Для начала создадим два массива символов, заведомо большой длинны. Первый массив - строка, второй - символы разделители. Через функцию fgets получим строку. Затем получаем количество символов разделителей, и передаём его и массив символов разделителей, в функцию получения символов разделителей. В ней перебираем массив пока итератор не станет равен количеству разделителей, получая с каждой итерацией символ разделитель и записывая его в массив. Теперь вызываем функцию нахождения индекса первого символа максимального слова, его длинны и порядкового номера слова (для удобства дальше в тексте называется find\_max**)**. Ей параметрами передаём кол-во символов разделителей, массив символов разделителей, строку, полученную от пользователя, указатель на переменную, содержащую индекс первого символа максимально большого слова в строке и указатель на переменную, содержащую длину максимально большого слова в строке. В функции зададим длину текущего слова равной 0, индекс первого символа текущего слова равным 0 а порядковый номер текущего слова равным 1. Далее будем перебирать строку пока не дойдём до символа перехода на следующую строку (\n). В каждой итерации проверяем чтобы функция проверки символа вернула 1. Если да, увеличиваем текущую длину на 1, если при этом указатель на длину максимального слова будет ссылаться на значение меньшее или равное длине текущего слова, то заменяем значение этого указателя длинной текущего слова, значение указателя ссылающегося на индекс первого символа максимально длинного слова в строке заменяем значением индекса первого символа текущего слова, в порядковый номер искомого слова записываем порядковый номер текущего слова. Если же функция проверки символа вернула 0, то обнуляем значение длинны текущего слова, а значение индекса первого символа слова делаем равным текущему значению итератора + 1 (+1 нужно потому, что символ текущего индекса — это символ разделитель) и инкриминируем значение порядкового номера текущего слова. Функция проверки символа принимает в себя параметрами i-тый символ строки, кол-во символов разделителей, и сам массив символов разделителей. Функция перебирает символы из массива символов разделителей и сравнивает их с переданным символом. На каждой итерации проверяем, если переданный символ из строки равен символу из массива, значит это символ разделитель, и мы возвращаем 0. Если символ не совпадёт не с одним из символов массива, то мы возвращаем 1. Функция find\_max возвращает порядковый номер слова максимальной длинны. Теперь, когда мы имеем индекс первого символа слова максимальной длинны, его длину и порядковый номер, мы выводим их на экран с помощью функции вывода. Передадим в функцию индекс первого символа слова максимальной длинны, длину слова, и строку. Функция перебирает строку начиная с переданного индекса до переданного индекса + длинна слова, и выводит каждый элемент, таким образом мы выведем нужное нам слово. После чего выводим порядковый номер слова.

**Описание переменных**

* **Функция main():**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | string[] | char | Строка слов и произвольного кол-ва символов разделителей |
| 2 | n | int | Кол-во символов разделителей |
| 3 | divisions[] | char | Массив символов разделителей |
| 4 | start\_word\_index | int | Индекс с которого начинается слово максимальной длинны |
| 5 | length | int | Длинна максимального слова |
| 6 | word\_number | int | Порядковый номер слова |

* **Ф-я получения символов разделителей get\_divs(int n, char divs[]):**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | n | int | Кол-во символов разделителей |
| 2 | divs[] | char | Массив символов разделителей |
| 3 | i | int | Итератор |

* **Ф-я проверки, является ли символ разделителем check\_s(char symb, int n, char divs[]):**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | symb | char | Проверяемый символ |
| 2 | n | int | Кол-во символов разделителей |
| 3 | divs[] | char | Массив символов разделителей |
| 4 | j | int | Итератор |

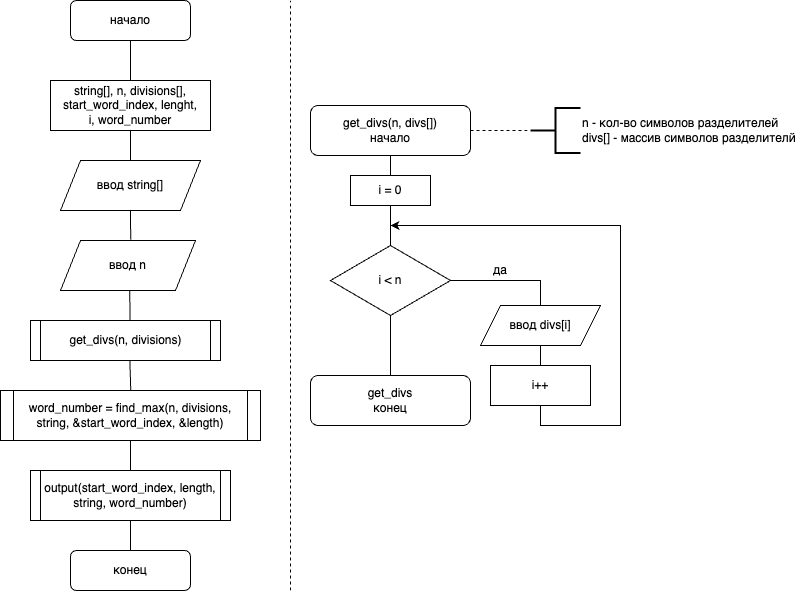
* **Функция нахождения индекса первого символа максимального слова, его длинны и порядкового номера слова find\_max(int n, char divs[], char str[], int \*start\_max\_word\_index, int \*max\_len):**

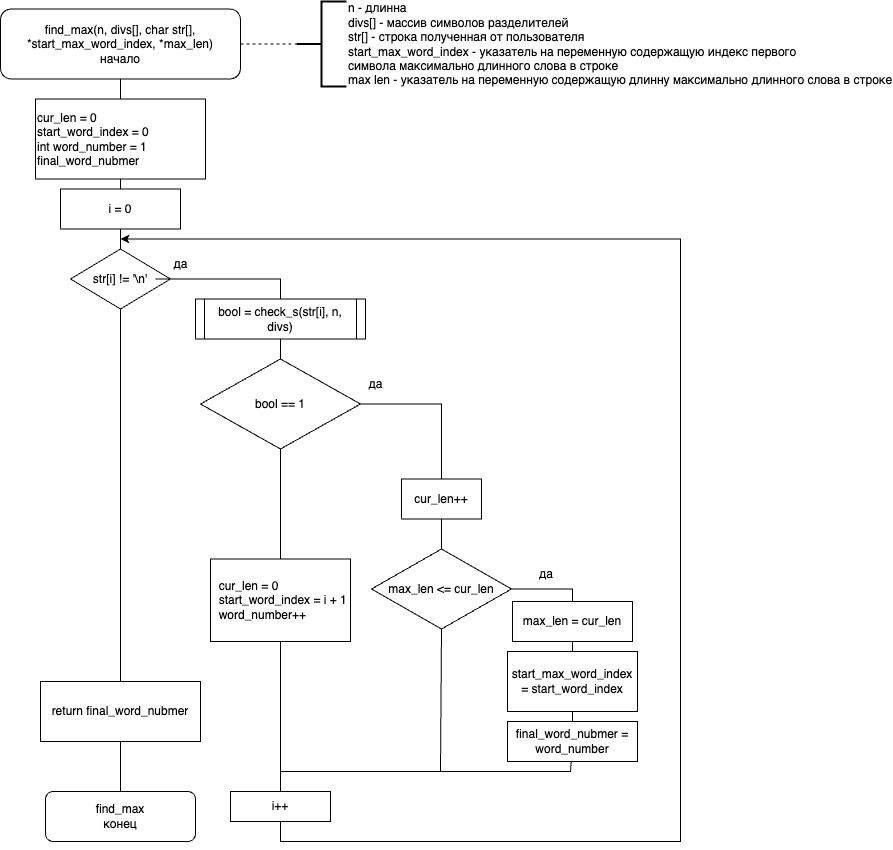
|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | n | int | Кол-во символов разделителей |
| 2 | divs[] | char | Массив символов разделителей |
| 3 | str[] | char | Строка слов и произвольного кол-ва символов разделителей |
| 4 | start\_max\_word\_index | int \* | Указатель на переменную, содержащую индекс первого символа максимально большого слова |
| 5 | max\_len | int \* | Указатель на переменную, содержащую длину максимально большого слова |
| 6 | cur\_len | int | Длинна текущего слова в итерации цикла |
| 7 | i | int | Итератор |
| 8 | start\_word\_index | int | Индекс начального символа слова в итерации цикла |
| 9 | word\_number | int | Порядковый номер текущего слова |
| 10 | final\_word\_nubmer | int | Порядковый номер искомого слова |

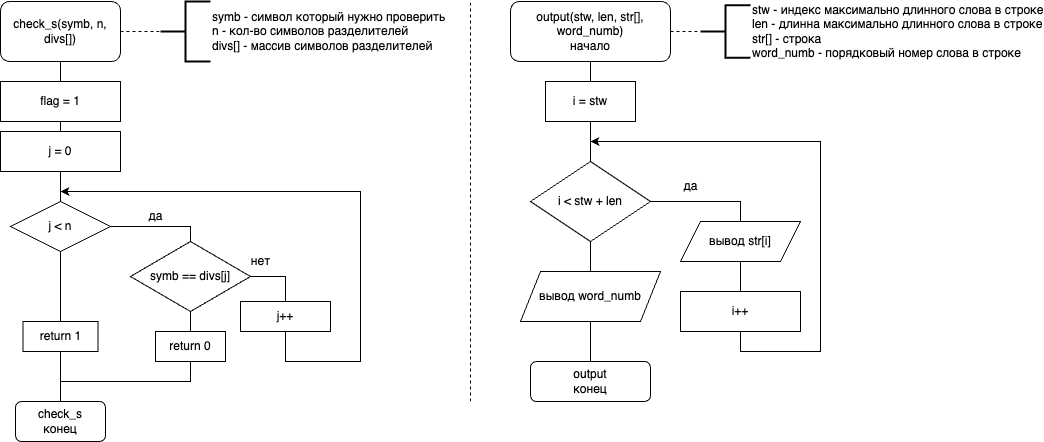
* **Функция вывода слова максимальной длинны и его длинны на экран output(int stw, int len, char str[], int word\_numb)**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | stw | int | Индекс максимально длинного слова в строке |
| 2 | len | int | Длинна максимально длинного слова в строке |
| 3 | str[] | char | Строка слов и произвольного кол-ва символов разделителей |
| 4 | i | int | Итератор |
| 5 | word\_numb | int | Порядковый номер текущего слова |

**Схема алгоритма**







**Контрольные примеры**

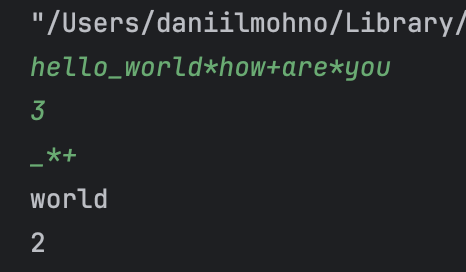
|  |  |  |
| --- | --- | --- |
| Пример № | Входные данные | Выходные данные |
| 1 | hello\_world\*how+are\*you  3  \_\*+ | world  2 |
| 2 | Neque.porro,quisquam:esto quit  4  :. , | quisquam  3 |
| 3 | Neque.porro,quisquam:esto quit  3  :., | esto quit  4 |
| 4 | iii,oooo.ppppp  2  ,. | ppppp  3 |

**Текст программы**

#include <stdio.h>  
  
#define SIZE 100  
  
void get\_divs(int n, char divs[]);  
  
int check\_s(char symb, int n, char divs[]);  
  
int find\_max(int n, char divs[], char str[], int \*start\_max\_word\_index, int \*max\_len);  
  
void output(int stw, int len, char str[], int word\_numb);  
  
int main()  
{  
 char string[SIZE];  
 int n;  
 char divisions[SIZE];  
 int start\_word\_index = 0;  
 int length = 0;  
 int word\_number;  
  
 fgets(string, SIZE, stdin);  
 scanf("%i\n", &n);  
 get\_divs(n, divisions);  
  
 word\_number = find\_max(n, divisions, string, &start\_word\_index,  
 &length);  
  
 output(start\_word\_index, length, string, word\_number);  
  
 return 0;  
}  
  
void get\_divs(int n, char divs[])  
{  
 for (int i = 0; i < n; i++)  
 scanf("%c", &divs[i]);  
}  
  
int check\_s(char symb, int n, char divs[])  
{  
 for (int j = 0; j < n; j++)  
 if (symb == divs[j])  
 return 0;  
 return 1;  
}  
  
int find\_max(int n, char divs[], char str[], int \*start\_max\_word\_index, int \*max\_len)  
{  
 int cur\_len = 0, i;  
 int start\_word\_index = 0;  
 int word\_number = 1, final\_word\_nubmer;  
 for (i = 0; str[i] != '\n'; i++)  
 {  
  
 if (check\_s(str[i], n, divs))  
 {  
 cur\_len++;  
 if (\*max\_len <= cur\_len)  
 {  
 \*max\_len = cur\_len;  
 \*start\_max\_word\_index = start\_word\_index;  
 final\_word\_nubmer = word\_number;  
 }  
 }  
 else  
 {  
 cur\_len = 0;  
 start\_word\_index = i + 1;  
 word\_number++;  
 }  
 }  
 return final\_word\_nubmer;  
}  
  
void output(int stw, int len, char str[], int word\_numb)  
{  
 for (int i = stw; i < stw + len; i++)  
 {  
 printf("%c", str[i]);  
 }  
 printf("\n%i\n", word\_numb);  
}

**Примеры выполнения программы**

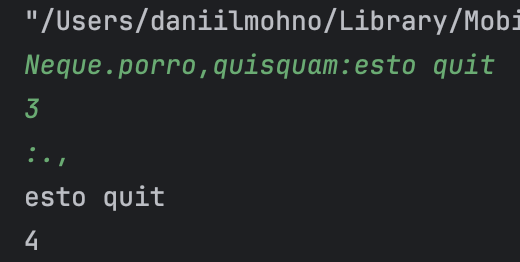
**Пример 1**

****

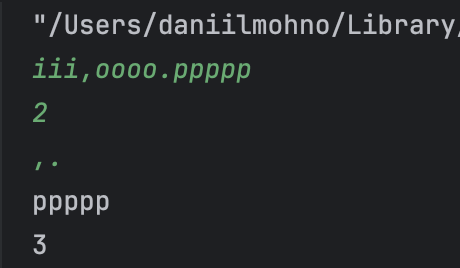
**Пример 2**

****

**Пример 3**

****

**Пример 4**

****

**Выводы.**

В результате выполнения работы была изучена обработка строк в языке Си и получены практические навыки в программировании на этом языке.