**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №2**

**по дисциплине «Программирование»**

Тема: Динамические структуры данных

|  |  |  |
| --- | --- | --- |
| Студент гр. 6304 |  | Зыль С. Е. |
| Преподаватель |  | Берленко Т.А. |

Санкт-Петербург

2017

**Оглавление**

[Цель работы 3](#_Toc478915831)

[Задание 3](#_Toc478915832)

[Содержание 3](#_Toc478915833)

[Вывод 5](#_Toc478915834)

# **Цель работы**

Реализация стека на базе массива. Изучение функций для работы со стеком.

# **Задание**

Требуется написать программу, получающую на вход строку, (без кириллических символов и не более 3000 символов) представляющую собой код "простой" [html](https://ru.wikipedia.org/wiki/HTML)-страницы и проверяющую ее на валидность. Программа должна вывести **correct** если страница валидна или **wrong**.   
  
html-страница, состоит из тегов и их содержимого, заключенного в эти теги. Теги представляют собой некоторые ключевые слова, заданные в треугольных скобках. Например, <**tag**> (где tag - имя тега). Область действия данного тега распространяется до соответствующего закрывающего тега </**tag**> который отличается символом /. Теги могут иметь вложенный характер, но не могут пересекаться   
<tag1><tag2></tag2></tag1> - верно  
<tag1><tag2></tag1></tag2> - не верно  
  
Существуют теги, не требующие закрывающего тега.   
  
Валидной является html-страница, в коде которой всякому открывающему тегу соответствует закрывающий (за исключением тегов, которым закрывающий тег не требуется)  
  
Во входной строке могут встречаться любые парные теги, но гарантируется, что в тексте, кроме обозначения тегов, символы < и > не встречаются. аттрибутов у тегов также нет.  
Теги, которые не требуют закрывающего тега: <br>, <hr>  
  
Стек (который потребуется для алгоритма проверки парности тегов) требуется реализовать самостоятельно на базе**массива**.

# **Содержание**

Для решения задачи была создана имитация «стека» на базе массива, который работает по принципу LIFO ( Last In First Out). Для работы со стеком были созданы функции:

1. void push(char\*\* stack, char\* tag, int number){

strcpy(stack[number], tag);

}

Функция по номеру number записывает в стек тег tag.

1. void removeEl(char\*\* stack, int number){

stack[number] = '\0';

}

Функция удаляет элемент стека stack под номером number.

1. int checker(char\*\* stack, char\* tag, int number){

int i;

for(i = 0; i<number; i++){

if(strstr(tag, stack[i]) != NULL){

break;

}

}

return i;

}

Функция проверяет, совпадает ли последний элемент стека stack и переданный тег tag и возвращает его номер в массиве.

1. Основная работа стека реализована в функции main:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define SIZE\_OF\_STACK 1000

void push(char\*\* stack, char\* tag, int number){

strcpy(stack[number], tag);

}

void removeEl(char\*\* stack, int number){

stack[number] = '\0';

}

int checker(char\*\* stack, char\* tag, int number){

int i;

for(i = 0; i<number; i++){

if(strstr(tag, stack[i]) != NULL){

break;

}

}

return i;

}

int main(){

char\*\* stack = (char\*\*)malloc(SIZE\_OF\_STACK\*sizeof(char\*)); // выделение памяти под стек

char c;

int i = 0;

int key;

char\* array = (char\*)malloc(50\*sizeof(char)); // выделение памяти под строку

scanf("%c", &c);

while (c != '\n' ){ // считывание по символа перевода строки

if(c == '<'){

stack[i] = (char\*)malloc(50\*sizeof(char)); // выделение памяти по тег

int j = 0;

scanf("%c", &c);

if(c != '/'){

array[j] = '/';

j++;

}

while((c!='>') && (c!='\n')){ // записываем в строку тег

array[j] = c;

j++;

scanf("%c", &c);

}

if(c=='\n') // если тег обрывается, то печатаем сообщение об ошибке

{

printf("wrong\n");

return 0;

}

array[j] = '\0';

if((strcmp("/br", array)!= NULL) && (strcmp("/hr", array)!=NULL)){

key = checker(stack, array, i);

if( i == key){

push(stack, array, i); // добавляем в стек, если открывающийся тег

++i;

}

else if(i-1 == key){

removeEl(stack, i); // удаляем, если элемент на вершине стека совпал с текущим

free(stack[i]);

--i;

}

else{

printf("wrong\n");

return 0;

}

free(array);

}

}

scanf("%c", &c);

}

free(stack);

if(i == 0) // cтек пуст

printf("correct\n");

else

printf("wrong");

return 0;

}

Программа считывает данные из входного потока по символам. Когда найден символ открытия тега «<», то в строку arrayсчитывается имя тега. Если тег — открывающий, вызывается функция push, если закрывающий — функция removeEl. После того, как мы получили символ конца строки, программа проверяет стек на наличие элементов, если он не пустой (есть незакрытые теги), программа выводит сообщение «wrong», если стек пустой (все теги закрыты), выводит сообщение «correct».

# **Вывод**

Выполнив данную лабораторную работу, было освоено и закреплено на практике написание программы, моделирующей работу стека на базе массива, были реализованы функции, которые позволяют работать со стеком.