**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №1**

**по дисциплине «Программирование»**

Тема: «Обзор стандартной библиотеки»

|  |  |  |
| --- | --- | --- |
| Студент гр. 6304 |  | Попов С.С. |
| Преподаватель |  | Берленко Т.А. |

Санкт-Петербург

2017

# Цель работы

Написать программу на языке Си, которая проверяет корректность расстановки тегов в html-странице.

# 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

**lr2.c**

#include <stdio.h>

#include <string.h>

typedef struct tag

{

char value[300];

} tag;

typedef struct tag\_list

{

tag list[3000];

int count;

} tag\_list;

int parse\_html(char html[])

{

// initialize stack for list of tags

tag\_list t\_list;

t\_list.count = 0;

int len = strlen(html);

int i,j;

tag temp\_tag;

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

{

// close brackets are not allowed without open bracket

if (html[i] == '>')

return 1; // error

if (html[i] == '<')

{

j = 0;

// add "<" to tag

temp\_tag.value[j] = html[i];

j++;

i++;

// missing close bracket

if (i >= len)

return 1; // error

// read tag body

while (html[i] != '>')

{

// open brackets are not allowed in tag body

if (html[i] == '<')

return 1; // error

temp\_tag.value[j] = html[i];

j++;

i++;

}

// add ">" and trailing "\0" to tag

temp\_tag.value[j] = html[i];

temp\_tag.value[j+1] = '\0';

// tag found, if it's closing, check if it matches last open tag

if (is\_closing(&temp\_tag))

{

if (are\_tags\_matched(&(t\_list.list[t\_list.count-1]), &temp\_tag))

{

t\_list.count--; // pop tag from a list

}

else

{

return 1; // error, closing tag doesn't match open one

}

}

else

{

// skip void tags like <br>, <hr>

if (is\_tag\_void(&temp\_tag) == 0)

{

// push new open tag to the list

strcpy(t\_list.list[t\_list.count].value, temp\_tag.value);

t\_list.count++;

}

}

}

}

if (t\_list.count != 0)

return 1; // error, tags mismatch

return 0; // success

}

int is\_tag\_void(tag\* test\_tag)

{

if (strcmp(test\_tag->value,"<br>") == 0)

return 1;

if (strcmp(test\_tag->value,"<hr>") == 0)

return 1;

return 0;

}

int is\_closing(tag\* test\_tag)

{

if (test\_tag->value[1] == '/')

{

return 1; // closing

}

else

{

return 0; // not closing

}

}

int are\_tags\_matched(tag\* open\_tag, tag\* close\_tag)

{

if (strcmp(&(open\_tag->value[1]), &(close\_tag->value[2])) == 0)

return 1;

return 0;

}

int main()

{

// read html

int i = 0;

char html[3000];

char temp;

while((temp = getchar()) != '\n')

{

html[i] = temp;

i++;

}

html[i] = '\0';

// parse html

int result = parse\_html(html);

if (result != 0)

{

printf("wrong\n");

return 0;

}

printf("correct\n");

return 0;

}

# Вывод

В ходе работы была написана программа, отвечающая условиям задачи. Был создан стек для хранения и обработки информации. Также были закреплены навыки работы со строками и структурами.