**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: Стандартная библеотека Си**

|  |  |  |
| --- | --- | --- |
| Студент гр. 6303 |  | Архипцев Евгений |
| Преподаватель |  | Чайка Константин |

Санкт-Петербург

2017

**Цель работы:**

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**.

**Исходный код программы:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

char \*\* stack;

int last = -1;

void yps(){

printf("wrong");

exit(0);

}

void tag(){

int i,flag = 0;

char s[4] = "20br";

if (stack[last][0] == '2'){

for (i = 0;i<4;i++){

if (stack[last][i] != s[i]){

flag = 1;

}

}

if (flag == 0){

free(stack[last]);

last--;

}

flag = 0;

}

s[4] = "20hr";

if (stack[last][0] == '2'){

for (i = 0;i<4;i++){

if (stack[last][i] != s[i]){

flag = 1;

}

}

if (flag == 0){

free(stack[last]);

last--;

}

}

}

int same(){

int i;

if (stack[last][0] == stack[last-1][0] && stack[last][1] != stack[last-1][1] ){

int size\_s = stack[last][0] - '0';

for ( i = 0;i<size\_s;i++){

if (stack[last][i+2] != stack[last-1][i+2]){

return 1;

}

}

}

return 0;

}

void proverka(){

if ( last - 1 >=0){

if (same() == 0){

free(stack[last]);

last--;

free(stack[last]);

last--;

}

else{

yps();

}

}

else{

yps();

}

}

void vvod\_tag(){

char c;

char \*s;

int i = 0,n=1,j;

last++;

s = (char\*)malloc((15 \* n) \* sizeof(char));

while ((c = getchar()) != '>'){

if (i == 0){

if (c == '/'){

s[i] = '1';

}

else{

s[i] = '0';

i++;

s[i] = c;

}

}

else{

s[i] = c;

}

i++;

if ((i % 15) == 0){ // увеличение места для тега в строке

n++;

s = (char\*)realloc(s,(15 \* n) \* sizeof(char));

}

}

stack[last] = (char\*)malloc((i \* sizeof(char )) + 2); // увеличение места для тега в стеке

stack[last][0] = (i-1) + '0';

if (s[0] == '1'){

stack[last][1] = '1';

}

else{

stack[last][1] = '0';

}

for ( j =1 ;j<i+1;j++){

stack[last][j+1] = s[j];

}

/\*\*/

/\* for (j = 0;j<i+1;j++){

printf("%c",stack[last][j]);

}

printf("\n");\*/

free(s);

tag();

if (stack[last][1] == '1'){

proverka();}

}

int main()

{

int kol=15,n = 1;

stack = (char \*\*)malloc((kol\*n)\*sizeof(char \*));

char c;

//freopen("input.txt","r",stdin);

while((c = getchar()) != EOF){

if (last >= (kol\*n - 1)){ // если количесвто тегов перерасло планку

n++;

stack = (char \*\*)realloc(stack,(kol\*n)\*sizeof(char \*));

}

if (c == '<'){

vvod\_tag();

}

}

if (last == -1){

printf("correct");

}

else{

yps();

}

}