**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: **Валидатор HTML кода.**

|  |  |  |
| --- | --- | --- |
| Студент гр. 6303 |  | Ваганов Н.А. |
| Преподаватель |  | Берленко Т.А. |

Санкт-Петербург

2017

**Цель.** Реализовать стек на базе списка. Научиться работать 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 **списка**.

**Ход работы.**

1. Объявим структуру, в которой будет лежать стек и количество элементов, которые в нем содержатся.

typedef struct Stack {

char tag[50];

struct Stack \* next;

}stack\_t;

1. Опишем функцию, которая определяет, пуст ли стек.

int is\_empty(stack\_t \*\* top) {

return \*top == NULL;

}

1. Опишем функцию, которая добавляет элемент в стек.

void push(stack\_t \*\*top, char \*tag)

{

stack\_t \*tmp = (stack\_t\*)malloc(sizeof(stack\_t));

tmp->next = \*top;

strcpy(tmp->tag, tag);

\*top = tmp;

}}

1. Опишем функцию, которая возвращает значение последнего элемента в стеке.

char\* top(stack\_t \*\*top) {

if (!is\_empty(top))

return (\*top)->tag;

else

return NULL;

}

1. Опишем функцию для удаления последнего элемента в стеке.

void pop(stack\_t \*\*top) {

stack\_t \*node;

node = \*top;

\*top = (\*top)->next;

free(node);

}

1. Основная функция программы. Функция ищет открывающиеся теги, добавляет перед ними “ / ” и кладет в стек. Это позволяет нам «искать» закрывающие теги путем сравнения с последним элементом стека. В случае, если закрывающий тег не совпадает с тем, что лежит последним в стеке, то html-код не валиден. После прохождения всей строки программа проверяет стек, если в нем есть элементы, то какой из тегов не имеет пары, а, следовательно, html-код не валиден. Исключением является теги br и hr, которые не являются парными, поэтому функция их игнорирует.

int main() {

stack\_t \*tags = NULL;

char \*buffer, checker, addslash[100], br[] = "br", hr[] = "hr";

int i;

buffer = (char \*)malloc(sizeof(char) \* 100);

checker = getchar();

while (checker != '\n')

{

while (checker != '<')

{

if (checker == '\n')

{

if (top(&tags) == NULL)

{

printf("correct\n");

return 0;

}

else

{

printf("wrong\n");

return 0;

}

}

checker = getchar();

}

i = 0;

checker = getchar();

while (checker != '>')

{

buffer[i] = checker;

i++;

checker = getchar();

}

buffer[i] = '\0'; //

if ((strcmp(buffer, br) != 0) && (strcmp(buffer, hr) != 0) && (buffer[0] != '/'))

{

strcpy(addslash, "/");

char \*str;

str = strcat(addslash, buffer);

push(&tags, str); //

}

else if (buffer[0] == '/' && (top(&tags) != NULL) &&(strcmp(top(&tags), buffer)==0))

pop(&tags); //

else if (buffer[0] == '/' && (top(&tags) == NULL))

{

printf("wrong\n");

return 0;

}

checker = getchar();

}

if (!is\_empty(&tags)) printf("wrong\n");

else printf("correct\n");

return 0;

}

1. Файл с лабораторной работой и отчет загружаем на github с помощью консоли:

* Создана новая ветка с помощью команды git checkout –b Vaganov\_sem2\_lab2
* Добавлены файлы загрузки с помощью команды git add Vaganov\_sem2\_lab2
* Добавлен комментарий коммита командой git commit –m “Vaganov\_sem2\_lab2”
* Файлы были загружены командой git push origin Vaganov\_sem2\_lab2

**Вывод:** В ходе работы был реализован валидатор html-кода. Для выполнения этой задачи был использован стек, реализованный на списках.