

**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: Проверка на валидность html-страницы**

Студент гр. 6303

\_\_\_\_\_

Зубов К.А

Преподаватель

\_\_\_\_\_

Берленко Т.А.

Санкт-Петербург

2017

**Цель.** Реализация стека на базе списка.

**Задание.**

Требуется написать программу, получающую на вход строку, (без кириллических символов и не более 3000 символов) представляющую собой код "простой" [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. атрибутов 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. Объявление структуры, в которой будет лежать стек и количество элементов, которые в нем содержатся.

```
typedef struct Stack {
    char tag[100];
    struct Stack * next;
}Stack_t;
```

2. Описание функции, которая добавляет элемент в стек.

```
void push(Stack_t **head,char *tag)
{
    Stack_t *tmp = (Stack_t*)malloc(sizeof(Stack_t));
    tmp->next = *head;
    strcpy(tmp->tag, tag);
    *head=tmp;
}
```

3. Описание функции, которая определяет пуст ли стек.

```
int headnull(Stack_t ** head) {  
    return *head == NULL;  
}
```

4. Описание функции, которая возвращает значение верхнего элемента стека.

```
char* head(Stack_t **head){  
    if (!headnull(head))  
        return (*head)->tag;  
    else  
        return NULL;  
}
```

5. Описание функции для удаления последнего элемента в стеке.

```
void pop(Stack_t **head) {  
    Stack_t *node;  
    node = *head;  
    *head=(*head)->next;  
    free(node);  
}
```

6. Функция main – основная функция программы. Она ищет открывающиеся теги, добавляет перед ними “ / ” и кладет в стек. Происходит поиск закрывающих тегов путем сравнения тега, добавленного в буффер с последним элементом стека. Если закрывающий тег не совпадает с тем, что лежит на вершине стека, то html-код является не валидным. Если после считывания всей строки в стеке есть элементы, то существует тег, не имеющий пары, следовательно, html-страница не валидна. Также необходимо исключить теги br и hr, не являющиеся парными. Функция их пропускает.

```
int main() {  
    Stack_t *tags = NULL;  
    char *buf, c;  
    int i;  
    buf = (char *)malloc(sizeof(char) * 100);  
    c = getchar();  
    while (c != '\n') {  
        while (c != '<')  
        {  
            if (c == '\n')  
            {  
                if (head(&tags) == NULL)  
                {  
                    printf("correct\n");  
                    return 0;  
                }  
            }  
        }  
    }  
}
```

```

        else
        {
            printf("wrong\n");
            return 0;
        }

    }
    c = getchar();
}

i = 0;

c = getchar();
while (c != '>')
{
    buf[i] = c;
    i++;
    c = getchar();
}
buf[i] = '\0'; //
if ((strcmp(buf, "br") != 0) &&
    (strcmp(buf, "hr") != 0) &&
    (buf[0] != '/'))
{
    char sl[25];
    strcpy(sl, "/");
    char *str;
    str = strcat(sl, buf);
    push(&tags, str);
}
else if (buf[0] == '/' &&
    (head(&tags) != NULL) &&
    (strcmp(head(&tags), buf) == 0))
    pop(&tags);
else if (buf[0] == '/' && (head(&tags) == NULL))
{
    printf("wrong\n");
    return 0;
}

c = getchar();
}
if (!headnull(&tags))
printf("wrong\n");
else printf("correct\n");
return 0;

```

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