МИНОБРНАУКИ РОССИИ

Санкт-Петербургский государственный

электротехнический университет

«ЛЭТИ» им. В.И. Ульянова (Ленина)

Кафедра МОЭВМ

отчет

по лабораторной работе №2

по дисциплине «Программирование»

тема: ДИНАМИЧЕСКИЕ СТРУКТУРЫ ДАННЫХ

|  |  |  |
| --- | --- | --- |
| Студентка гр. 6303 |  | Горбунова А.П. |
| Преподаватель |  | Берленко Т.А. |

Санкт-Петербург

2017

**Цель**: 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.

 Существуют теги, не требующие закрывающего тега: <br>, <hr>

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.

Стек (который потребуется для алгоритма проверки парности тегов) требуется реализовать самостоятельно на базе **списка**.

**Ход работы:**

Алгоритм проверки валидности html-страницы изображен на Рис1.

**ПОИСК ТЕГА В СТРОКЕ**

ОТКРЫВАЮЩИЙ ТЕГ ЗАКРЫВАЮЩИЙ ТЕГ

**ПРОВЕРКА НА «br» «hr»** **ПРОВЕРКА СТЕКА**

«br» «hr» не «br» «hr» стек пуст стек не пуст

*переход добавление wrong* **ПРОВЕРКА**

*к в* **ПОСЛЕДНЕГО**

*следующему стек* **ТЕГА**

*тегу* **В СТЕКЕ**

*переход*

*к*  теги совпали теги не совпали

*следующему*

*тегу* *удаление* *wrong*

*последнего*

*элемента*

*из стека*

*переход*

*к*

*следующему*

*тегу*

**Описание кода**

* Для выполнения данной работы необходимо подключить три библиотеки: «*stdio.h*», «string.h» (в которой содержатся функции strcmp, strncmp, strtok) и «stdlib.h» (для выделения динамической памяти).

|  |
| --- |
| #include <stdio.h>  #include <string.h>  #include<stdlib.h> |

* Односвязный список состоит из узлов, каждый из которых содержит полезную информацию и ссылку на следующий узел. Последний узел ссылается на NULL.

Определим структуру узел:

typedef struct Stack\_Node

{

char\* value;

struct Stack\_Node \*next;

} Stack\_N;

* Далее, до функции *main*, объявляются функции, необходимые для работы со стеком

*Функция добавления тега в стек*

void push(Stack\_N \*\*head, char\* value)

{

Stack\_N \*tmp = malloc(sizeof(Stack\_N));

tmp->next = \*head;

tmp->value = value;

\*head = tmp;

}

Функция создает новый узел, перекидывает указатель next на старый узел, а указатель на вершину - на новый узел.

Вершиной теперь является новый узел.

*Функция удаления последнего добавленного тега из стека*

void pop (Stack\_N \*\*head)

{

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

return;

}

*Функция, возвращающая последний добавленный в стек тег*

char\* peek( Stack\_N\* head)

{

return head->value;

}

*Функция вывода элементов стека*

void printStack(Stack\_N\* head)

{

printf("\n stack's elements:\n ");

while (head)

{

printf("%s ", head->value);

head = head->next;

}

}

*Функция подсчета элементов стека*

size\_t getSize(Stack\_N \*head) {

size\_t size = 0;

while (head) {

size++;

head = head->next;

}

return size;

}

*Функция main*

int main()

{

char str[N+2];

*создан массив типа char, размер которого выбираем N+2, т.к. при считывании в массив запишется нулевой символ и символ перевода строки*

int control=1;

*переменная control необходима чтобы фиксировать ошибки в процессе работы программы*

Stack\_N\* head = (Stack\_N\*)malloc(sizeof(head));

*создан стек*

fgets(str, (N + 2), stdin);

*считана строка*

char\* tag;

char\* op = strchr(str, '<');

char\* cl = strchr(str, '>');

*функция strchr находит первое вхождение символа в строку и возвращает указатель на него или NULL*

while (cl)

{

*открывающий тег*

if (\*(op + 1) != '/')

{

tag = strtok((op + 1), ">");

*функция strtok делит строку на лексемы, разделенные символом-разделителем, записывает в переменную тег*

if (strcmp(tag, "br")!=0 && strcmp(tag, "hr")!=0)

*функция strcmp сравнивает посимвольно две строки, если они равны возвращает 0*

*условие: если это не теги, не требующие закрывающих тегов, то помещаем значение из переменной в стек*

push(&head, tag);

}

*закрывающий тег*

else

{

*в стеке содержится значение NULL. Если кроме него нет элементов, то стек считается пустым. Если стек пуст – ошибка.*

if (getSize==1)

{

control=0;

break;

}

*функция сравнивает n символов двух строк, если они совпали возвращает 0*

if (!strncmp((op + 2), peek(head), (cl - op - 2)))

*условие: если последний элемент из стека совпал с текущим, удаляем элемент из стека, если нет - ошибка*

pop(&head);

else

{

control=0;

break;

}

}

*Переход к следующему тегу*

op = strchr((cl + 1), '<');

cl = strchr((cl + 1), '>');

}

*Если стек не пуст - ошибка*

if (getSize(head)!=1)

control=0;

*Если ошибок нет – correct, иначе - wrong*

if (control==1)

printf("correct");

else

printf("wrong");

*Очищение памяти*

free(head);

return 0;

}

ИСХОДНЫЙ КОД

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#define N 500

//определение структуры узел

typedef struct Stack\_Node

{

char\* value;

struct Stack\_Node \*next;

} Stack\_N;

//функция добавления тега в стек

//(создает новый узел -> указатель next на старый узел,

//указатель на вершину - на новый узел -> вершина - новый узел )

void push(Stack\_N \*\*head, char\* value)

{

Stack\_N \*tmp = malloc(sizeof(Stack\_N));

tmp->next = \*head;

tmp->value = value;

\*head = tmp;

}

//функция удаления последнего добавленного тега из стека

void pop (Stack\_N \*\*head)

{

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

return;

}

//функция возвращает последний добавленный в стек тег

char\* peek( Stack\_N\* head)

{

return head->value;

}

//функция вывда элементов стека

void printStack(Stack\_N\* head)

{

printf("\n stack's elements:\n ");

while (head)

{

printf("%s ", head->value);

head = head->next;

}

}

//функция подсчета элементов стека

size\_t getSize(Stack\_N \*head) {

size\_t size = 0;

while (head) {

size++;

head = head->next;

}

return size;

}

int main()

{

char str[N+2];

int control=1;

Stack\_N\* head = (Stack\_N\*)malloc(sizeof(head));

fgets(str, (N + 2), stdin);

char\* tag;

char\* op = strchr(str, '<');

char\* cl = strchr(str, '>');

//функция strchr находит первое вхождение символа в строку и возвращает указатель на него или NULL

while (cl)

{

//открывающий тег

if (\*(op + 1) != '/')

{

tag = strtok((op + 1), ">");

//функция strtok делит строку на лексемы, разделенные символом-разделителем

if (strcmp(tag, "br")!=0 && strcmp(tag, "hr")!=0)

//функция strcmp сравнивает повимвольно две строки, если они равны возвращает 0

push(&head, tag);

}

//закрывающий тег

else

{

if (getSize==1)

{

control=0;

break;

}

//функция сравнивает n символов двух строк, если они совпали возвращает 0

if (!strncmp((op + 2), peek(head), (cl - op - 2)))

pop(&head);

else

{

control=0;

break;

}

}

op = strchr((cl + 1), '<');

cl = strchr((cl + 1), '>');

}

if (getSize(head)!=1)

control=0;

if (control==1)

printf("correct");

else

printf("wrong");

free(head);

return 0;

}

**Пример работы программы**

**1)**

**Input:**

<html><head><title>HTML Document</title></head><body><p><b>bold,<br><i>italic</i></b></p></body></html>

**Output:**

Успешно time: 0 memory: 9424 signal:0

correct

**2)**

**Input:**

<html><head><title>HTML Page</title></head><body><p><b>page ok<br><i>but not</b></p></body></html>

**Output:**

Успешно time: 0 memory: 9424 signal:0

wrong

**Добавление файлов на Github**

### Добавление файлов на Github было произведено с помощью консоли

### Переход на ветку master и создания + переход в новую ветку Gorbunova\_Alina\_lr2 :

git checkout master && git pull origin master

git checkout -b Gorbunova\_Alina\_2sem\_lr2

* Cозданиe папки :

mkdir Gorbunova\_Alina\_2sem\_lr2

* Добавлениe файлов, создание коммита и отправка коммитов на GitHub :

git add Gorbunova\_Alina\_2sem\_lr2/main.c

git commit -m "Gorbunova LR №2 done"

git push origin Gorbunova\_Alina\_2sem\_lr2

**Вывод:** в ходе лабораторной работы получены навыки работы с динамическими структурами данных, отработаны навыки использования различных функций стандартных библиотек, разработан алгоритм проверки html-страницы на валидность, основывающийся на работе со стеком, реализованным на односвязном списке.

В результате лабораторной работы на базе разработанного алгоритма была написана программа, проверяющая html-страницу на валидность.