**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра математического обеспечения и применения ЭВМ**

отчет

**по лабораторной работе №2**

**по дисциплине «Программирование»**

ТЕМА: РАССТАНОВКА ТЕГОВ.

|  |  |  |
| --- | --- | --- |
| Студент гр. 6303 |  | Ильяшук Д.И. |
| Преподаватель |  | Берленко Т. А. |

Санкт-Петербург

2017

**Цель работы:**

Научиться создавать и использовать функции работы со стеком, а также проверять код html страницы на правильность.

**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. 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. Были объявлены константы INPUTSIZE и TAGSIZE, которая отвечает за размер входной строки и размер тега.

#define INPUTSIZE 3000

#define TAGSIZE 20

1. Затем была объявлена структура, на базе которой построен стек.

typedef struct stack

{

char \* tag;

struct stack\* prevUnit;

}stack;

1. Была написала функция ***push,*** которая добавляет новую ячейку в стек.

void push(stack\* headUnit, char \*tag)

{

stack\* temp = (stack\*)malloc(sizeof(stack));

temp->tag = tag;

temp->prevUnit = headUnit->prevUnit;

headUnit->prevUnit = temp;

}

1. Была написала функция ***pop,*** которая удаляет последний элемент стека.

void pop(stack\* headUnit)

{

stack\* temp = (stack\*)malloc(sizeof(stack));

temp = headUnit->prevUnit;

headUnit->prevUnit = headUnit->prevUnit->prevUnit;

free(temp);

}

1. Была написала функция ***peek,*** которая возвращает последний элемент списка, не удаляя его.

char\* peek(stack\* headUnit)

{

return headUnit->prevUnit->tag;

}

1. Была написала функция ***freeStack,*** которая очищает стек.

void freeStack(stack\* headUnit)

{

stack\* temp;

while (headUnit)

{

temp = headUnit;

headUnit = headUnit->prevUnit;

free(temp);

}

}

1. Была написала функция ***getTag,*** которая принимает указатель на начало тега, а возвращает тег целиком.

char \* getTag(char\* tagPtr)

{

char \* tag = (char\*)malloc(TAGSIZE \* sizeof(char));

int i = 0;

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

{

tag[i] = tagPtr[i];

i++;

}

tag[i] = '\0';

return tag;

}

1. Далее приведен код основной части программы:

int main()

{

char array[INPUTSIZE];

char \* tag;

char \* pretag;

fgets(array, INPUTSIZE, stdin);

stack\* headUnit = (stack\*)malloc(sizeof(stack));

headUnit->prevUnit = NULL;

pretag = strtok(array, "<");

if (array[0] != '<')

pretag = strtok(NULL, "<");

while(pretag)

{

tag = getTag(pretag);

if (tag[0] != '/')

{

if (strcmp(tag, "br") && strcmp(tag, "hr"))

{

push(headUnit, tag);

pretag = strtok(NULL, "<");

continue;

}

else

{

free(tag);

}

}

if (tag[0] == '/')

{

if (headUnit->prevUnit && !strcmp(&tag[1], peek(headUnit)))

{

pop(headUnit);

}

else

{

printf("wrong");

freeStack(headUnit);

return 0;

}

}

pretag = strtok(NULL, "<");

}

if (!headUnit->prevUnit)

printf("correct");

else

printf("wrong");

freeStack(headUnit);

return 0;

}

1. Затем код работы вместе с отчетом были загружены на github:

* Создана новая ветка с помощью команды *git checkout –b ilyashuk\_sem2\_lab2*
* Добавлены файлы для загрузки с помощью команды *git add lab2.c otchet\_lab2.docx*
* Добавлен комментарий коммита командой *git commit –m “lab2”*
* Файлы были загружены командой *git push origin ilyashuk\_sem2\_lab2*

**Вывод:** В ходе лабораторной работы получены навыки работы со стеком и проверки правильности html страницы.