**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: ДИНАМИЧЕСКИЕ СТРУКТУРЫ ДАННЫХ

|  |  |  |
| --- | --- | --- |
| Студент гр. 6303 |  | Эвергрин П. С. |
| Преподаватель |  | Берленко Т.А. |

Санкт-Петербург

2017

Цель работы: 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

**Ход работы.**

1. **Описание стуктуры.**

struct Node

{

char \*data;

struct Node \*next;

};

1. **Функции для работы со стеком.**

* **Добавление элемента в стек.**

Функция принимает 2 аргумента:

void push(struct Node\*\*top,char\*sign)

{

struct Node\* tmp = (struct Node\*)malloc(sizeof(struct Node));

tmp->data=sign;

tmp->next=\*top;

\*top=tmp;

}

* **Удаление вершины стека.**

Функция принимает в качестве аргумента указатель на указатель вершины стека.

void pop(struct Node \*\*head)

{

struct Node\*tmp = \*head;

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

free(tmp);

}

* **Возвращение элемента, находящегося на вершине стека.**

char\*topdata(struct Node\*top)

{

return top->data;

}

* **Пуст ли список**

int isempty(struct Node\*top) {

if(top == NULL)

return 1;

else

return 0;

}

* **Освобождение данн**

void freedata(struct Node \*\*head)

{

while(\*head)

{

free((\*head)->data);

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

}

}

**3.Функция main()**

int main()

{

struct Node\* top=NULL;

int i,y,x;

char\*arr=(char\*)malloc(3001\*sizeof(char));

fgets (arr, 3001, stdin);

arr[strlen(arr) - 1] = '\0';

for(i=0;arr[i];i++)

{

if(arr[i]=='<'&&arr[i+1]!='/')

{

char \*wordin=(char\*)malloc(20\*sizeof(char));

y=i+1;

for(x=0;arr[y]!='>';x++,y++)

{

wordin[x]=arr[y];

}

wordin[x]='\0';

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

push(&top,wordin);

i=y;

}

if(arr[i]=='<'&&arr[i+1]=='/')

{

if(isempty(top)== 1)

{

printf("wrong");

freedata(&top);

free(top);

return 0;

}

else

{

char\*wordout=(char\*)malloc(20\*sizeof(char));

y=i+2;

for(x=0;arr[y]!='>';x++,y++)

{

wordout[x]=arr[y];

}

wordout[x]='\0';

i=y;

if(strcmp(wordout,topdata(top))!=0)

{

printf("wrong");

free(wordout);

free(arr);

freedata(&top);

free(top);

return 0;

}

else

pop(&top);

free(wordout);

}

}

}

if(isempty(top)== 1)

printf("correct");

else

printf("wrong");

freedata(&top);

free(arr);

free(top);

return 0;

}

**Полный код программы.**

#include <stdio.h>

#include <stdlib.h>

struct Node

{

char \*data;

struct Node \*next;

};

char\*topdata(struct Node\*top)

{

return top->data;

}

void push(struct Node\*\*top,char\*sign)

{

struct Node\* tmp = (struct Node\*)malloc(sizeof(struct Node));

tmp->data=sign;

tmp->next=\*top;

\*top=tmp;

}

void freedata(struct Node \*\*head)

{

while(\*head)

{

free((\*head)->data);

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

}

}

void pop(struct Node \*\*head)

{

struct Node\*tmp = \*head;

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

free(tmp->data);

free(tmp);

}

int isempty(struct Node\*top) {

if(top == NULL)

return 1;

else

return 0;

}

int main()

{

struct Node\* top=NULL;

int i,y,x;

char\*arr=(char\*)malloc(3001\*sizeof(char));

fgets (arr, 3001, stdin);

arr[strlen(arr) - 1] = '\0';

for(i=0;arr[i];i++)

{

if(arr[i]=='<'&&arr[i+1]!='/')

{

char \*wordin=(char\*)malloc(20\*sizeof(char));

y=i+1;

for(x=0;arr[y]!='>';x++,y++)

{

wordin[x]=arr[y];

}

wordin[x]='\0';

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

push(&top,wordin);

i=y;

}

if(arr[i]=='<'&&arr[i+1]=='/')

{

if(isempty(top)== 1)

{

printf("wrong");

freedata(&top);

free(top);

return 0;

}

else

{

char\*wordout=(char\*)malloc(20\*sizeof(char));

y=i+2;

for(x=0;arr[y]!='>';x++,y++)

{

wordout[x]=arr[y];

}

wordout[x]='\0';

i=y;

if(strcmp(wordout,topdata(top))!=0)

{

printf("wrong");

free(wordout);

free(arr);

freedata(&top);

free(top);

return 0;

}

else

pop(&top);

free(wordout);

}

}

}

if(isempty(top)== 1)

printf("correct");

else

printf("wrong");

freedata(&top);

free(arr);

free(top);

return 0;

}

1. **Загрузка файла на GitHub.**

* В локальном репозитории была создана новая ветка

evergreen\_sem2\_lr2

git checkout -b evergreen\_sem2\_lr2

* Создана директория evergreen\_sem2\_lr2

mkdir evergreen\_sem2\_lr2

* В вышеуказанную папку был перемещен файл LAB2.c
* Исходный файл добавлен в локальный репозиторий

git add LAB2.c

* Командой <git commit –m> были зафиксированы изменения в проекте

git commit –m « evergreen\_sem2\_lr2»

* Командой <git push> последний commit был отправлен на удаленный репозиторий

git push origin evergreen\_sem2\_lr2

**Вывод:** в ходе работы был реализована стековая машина на базе списка, написана программа, 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**.Код и отчет по работе были загружены в репозиторий.