МИНОБРНАУКИ РОССИИ

Санкт-Петербургский государственный

электротехнический университет

«ЛЭТИ» им. В.И. Ульянова (Ленина)

Кафедра МО ЭВМ

отчет

по лабораторной работе №1

по дисциплине «Программирование»

Тема: Моделирование стека

|  |  |  |
| --- | --- | --- |
| Студентка гр. 6304 |  | Прозорова А.Д. |
| Преподаватель |  | Берленко Т.А. |

Санкт-Петербург

2017

Оглавление

Цель работы 3

[Задание 3](#_Toc482737541)

[Содержание 4](#_Toc482737542)

[Вывод 6](#_Toc482737543)

**Цель работы**

Научиться реализовывать стек и использовать его для определенных задач.

**Задание**

 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. Область действия данного тега распространяется до соответствующего закрывающего тега </**tag**> который отличается символом /. Теги могут иметь вложенный характер, но не могут пересекаться.

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 требуется реализовать самостоятельно на базе**массива.**

**Содержание**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int j = -1;

void push(char \*\*stack, char \*element){ *//добавляет элемент в массив со стеком*

j++;

strcpy(stack[j],element);

}

void pop(char \*\*stack){ *//"удаляет последний элемент стека"*

j--;

}

int vacuum(){ *//определяет, пустой ли стек по индексу последнего элемента*

if(j==-1) return 0;

else return 1;

}

int main()

{

char \*\*stack = (char\*\*)malloc(100 \*sizeof(char\*)); /*/выделяем память под массив указателей, каждый из которых будет указывать на строку тега, который мы запишем в сте*к

for (int i = 0; i<100; i++)

stack[i] = (char\*)malloc(20 \*sizeof(char)); *//выделяем память для каждого указателя из стэка*

char c;

char \*tag=(char\*)malloc(20\*sizeof(char)); *//указатель на текущий тэг*

while (c!='\n'){ *//пока не закончилась текущая строка*

scanf("%c",&c);

if (c=='<'){ /*/если этот символ равен символу, открывающему тег*

scanf("%20[^>]с",tag); /*/записываем следующие символы, пока не встретим закрывающий символ, в строку tag*

if (strcmp(tag,"br") && strcmp(tag,"hr")){ *//если тег не равен тегу <br> или <hr>*

if (tag[0] != '/') push(stack,tag);  *//если тег не закрывающий, записываем его в стек, прибавляем к j единицу*

else{

if (vacuum() == 0){ *//если стек пустой, след-но последний записанный тег закрывающий, то строка неверная, выходим из программы*

printf("wrong");

return 0;

}

else{ /*/если тег закрывающий, но стек не пустой*

if (strcmp(stack[j],tag+1) != 0){ /*/если последний элемент стека и тег, не считая одного элемента "/", не идентичны, закрываем программу*

printf("wrong");

return 0;

}

else pop(stack); *//если идентичны, "удаляем" последний элемент*

}

}

}

}

}

if (vacuum() == 0) *//если стек пустой (не осталось открывающих тегов)*

printf("correct");

else *//если стек не пустой*

printf("wrong");

*//Освобождение памяти*

free(tag);

int i=0;

while(stack[i++]!=NULL)

free(stack[i]);

free(stack);

}

**Вывод**

При выполнении данной работы было закреплено на практике написание программы, моделирующей работу стека, была применена реализация стека на базе массива.