**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: Динамические структуры данных**

|  |  |  |
| --- | --- | --- |
| Студент гр. 6303 |  | Архипцев Евгений |
| Преподаватель |  | Чайка Константин |

**Цель работы:**  
 Требуется 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**.

**Ход выполнения:**

**Flag** переменная, показывающая полноту стэка (Количество элементов)

**1.** Опишем функцию **push**  
Функция push() ставит значение на вершину стека.

Функция получает указатель на стэк, и элемент, который нужно поместить на вершину стэка.

void push(char \*\* stack, char \* element,int\* flag){

(\*flag)++;

strcpy(stack[\*flag],element);

}  
  
**2.** Опишем функцию **pop**  
Функция pop() удаляет значение с вершины, то есть удаляет последний элемент.

void pop(char \*\* stack,int\* flag){ // Удаляем последний элемент СТЭКа

(\*flag)--;

}

**3.** Опишем функцию **size\_s**  
Функция **size\_s**() возвращает количество элементов в стеке.

int size\_s(char \*\* stack,int\* flag){

return \*flag;

}

**4.** Опишем функцию **empty\_s**  
Функция **empty\_s()** дает понять пуст стек или нет .

Возвращает 0, если стэк пуст

Возвращает 1, если стэк не пуст

int empty\_s(char \*\* stack,int\* flag){ // Возвращает 0 ,если стэк пуст

if ((\*flag) == -1){//Возвращает 1 ,если стэк не пуст

return 0;

}

else{

return 1;

}

}

**5**. Считываем Тэги  
  
 **е**сли он открывающийся – помещаем в стэк  
 **е**сли он закрывающийся – проверяем, что он совпадает с предыдущим

**6**. Если стэк пуст по окончанию работы- выводим “**correct**”, иначе “**wrong**”.

**Исходный код программы:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

void push(char \*\* stack, char \* element,int\* flag){ // Помещаем в СТЭК

(\*flag)++;

strcpy(stack[\*flag],element);

}

char \*top(char \*\* stack,int\* flag){

return stack[\*flag];

}

void pop(char \*\* stack,int\* flag){ // Удаляем последний элемент СТЭКа

(\*flag)--;

}

int size\_s(char \*\* stack,int\* flag){ // Возвращает размер СТЭКа

return \*flag;

}

int empty\_s(char \*\* stack,int\* flag){ // Возвращает 0 ,если стэк пуст

if ((\*flag) == -1){//Возвращает 1 ,если стэк не пуст

return 0;

}

else{

return 1;

}

}

void FM(char\*\* tag){

int q;

for (q = 0; q<100; q++){

free(tag[q]);}

free(tag);

}

int main()

{

int flag = -1;

/\*ввод из файл\*/

//freopen("input.txt","r",stdin);

char c,cup[80];

int i,k;

/\*Выделение памяти под стэк\*/

char \*\*tag;

tag = (char\*\*)malloc(100 \* sizeof(char\*));

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

tag[i] = (char\*)malloc(80 \* sizeof(char));

}

/\*\*/

char \*string = (char\*)malloc(3000 \* sizeof(char));

gets(string);

for( i= 0;i<strlen(string);i++){

c = string[i];

if (c == '<') {

k=0;

while((cup[k] = string[i+k+1]) != '>')k++;

cup[k] = '\0';

if (strcmp(cup,"hr") && strcmp(cup,"br") ){

if (cup[0] != '/'){ // Если тэг - открывающийся : заносим в стэк

push(tag,cup,&flag);

}

else{

if (empty\_s(tag,&flag) == 0){ // Если стэк пуст, то есть ЗАКРВАЮЩИЙСЯ тэг не был открыт, то выход

printf("wrong");

return 0;

}

else{

if (strcmp(top(tag,&flag),cup+1) != 0){ // Если закрывающийся, не равен перед ним стоящему открывающемуся

free(string);

FM(tag);

printf("wrong");

return 0;

}

else{ // Если тэги совпадают

pop(tag,&flag);

}

}

}

}

}

}

if (empty\_s(tag,&flag) == 0){ // Если стэк пуст, то страницы ВАЛИДНА

printf("correct");

}

else{

printf("wrong");

}

FM(tag);

free(string);

}

**Вывод:** Были усовершенствуемы знания в использовании динамических массивов. Ознакомились с таким абстрактным типом данным как Стек.

Изучены функции Pop,Push,Size,Empty для работы со Стеком.