

**МИНОБРНАУКИ РОССИИ**  
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**  
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**  
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**  
**Кафедра МО ЭВМ**

**ОТЧЕТ**  
**по лабораторной работе №4**  
**по дисциплине «Программирование»**  
**Тема: Динамические структуры данных**

Студент гр. 0304:

\_\_\_\_\_

Шквиря Е.В.

Преподаватели:

\_\_\_\_\_

Чайка К. В.

Санкт-Петербург

2021

### **Цель работы.**

Изучить устройство работы динамической структуры данных под названием «стек», реализованной на базе массива.

### **Задание.**

Вариант 6.

### **Расстановка тегов.**

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. атрибутов 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 **массива**. Для этого необходимо:

Реализовать **класс** CustomStack, который будет содержать перечисленные ниже методы. Стек должен иметь возможность хранить и работать с типом данных `char*`

Объявление класса стека:

```
class CustomStack {

public:

    // методы push, pop, size, empty, top + конструкторы, деструктор

private:

    // поля класса, к которым не должно быть доступа извне

protected: // в этом блоке должен быть указатель на массив данных

    char** mData;

};
```

Перечень методов класса стека, которые должны быть реализованы:

- **void push(const char\* val)** - добавляет новый элемент в стек
- **void pop()** - удаляет из стека последний элемент
- **char\* top()** - доступ к верхнему элементу
- **size\_t size()** - возвращает количество элементов в стеке
- **bool empty()** - проверяет отсутствие элементов в стеке

- **extend(int n)** — расширяет исходный массив на n ячеек

### Примечания:

1. Указатель на массив должен быть protected.
2. Подключать какие-то заголовочные файлы не требуется, всё необходимое подключено(<cstring> и <iostream>)
3. Предполагается, что пространство имен std уже доступно
4. Использование ключевого слова using также не требуется

### Пример

*Входная строка:*

```
<html><head><title>HTML Document</title></head><body><p><b>This text is  
bold,<br><i>this is bold and italics</i></b></p></body></html>
```

*Результат:*

**correct**

### Выполнение работы.

Порядок выполнения поставленной задачи программой:

- 1) Для работы со словами, которые находятся в тегах, был реализован класс CustomStack, в котором присутствуют стандартные методы для взаимодействия со структурой данных, такие как *push*, *pop*, *empty* и тп.
- 2) В стек добавляются названия открывающихся тегов. Корректность html-строки проверяется посредством перехода к словам, заключённым между `<` и `>`. Для этого используется функция strchr для поиска соответствующих угловых скобок.

- 3) После того, как скобки были найдены, проверяется принадлежность слова к тегам, для которых не нужен соответствующий закрывающийся тег. Если они являются таковыми, то происходит переход к следующим скобкам.
- 4) Если тег начинающийся, то его слово добавляется на вершину стека. Если тег завершающийся и у нас есть хотя бы один элемент в стеке, то имена сравниваются, и если они совпали, то этот тег достаётся из стека, иначе html-строка некорректна и на этом её проверка завершается. Процесс сравнения производится для всех тегов. Если в конце у нас в цикле не возникли разные имена у двух соответствующих тегов и при выходе из цикла у нас стек пуст, то html-строка корректная.

Используемые переменные:

capacity — количество выделенной памяти под ячейки массива для стека,

mData — массив строк, в котором хранятся данные в стеке,

ind — индекс последней строки, хранящейся в стеке,

allocatedData — количество ячеек массива, для которых выделена память под строки,

temp (в extend) — временное хранилище данных массива для расширения памяти в стеке,

stack — стек для хранения названий тегов,

str — html-строка для проверки корректности,

lstr — указатель на начало слова в теге,

rstr — указатель на конец слова в теге,

isRight — хранит состояние правильности html-строки.

Используемые функции:

CustomStack — конструктор для класса CustomStack,  
~CustomStack — деструктор для класса CustomStack,  
push — добавление строки в стек,  
pop — удаление строки из стека,  
top — получение элемента на вершине стека,  
size — получение размера стека,  
empty — проверка стека на пустоту,  
extend — расширение массива стека.

Разработанный программный код см. в приложении А.

## Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	<code>&lt;html&gt;&lt;head&gt;&lt;title&gt;HTML Document&lt;/title&gt;&lt;/head&gt;&lt;body&gt;&lt;p&gt;&lt;b&gt;This text is bold,&lt;br&gt;&lt;i&gt;this is bold and italics&lt;/i&gt;&lt;/b&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</code>	correct	Верный ответ
2.	<code>&lt;abc&gt;&lt;/abc&gt;&lt;/g&gt;</code>	wrong	Верный ответ
3.	<code>&lt;abc&gt;&lt;/abc&gt;&lt;g&gt;</code>	wrong	Верный ответ

## Выводы.

Было изучено устройство работы динамической структуры данных «стек» на базе массива, и написана её реализация на языке C++.

Разработана программа, принимающая на вход строку, представляющую собой html-страницу с набором тегов, и проверяющая эту страницу на валидность тегов.

## ПРИЛОЖЕНИЕ А

### ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab7.cpp

```
//#include <cstdlib>
//#include <cstring>
//#include <iostream>
//using namespace std;

#define STR_SIZE 3001

class CustomStack {

public:
    CustomStack() {
        this->capacity = 5;
        this->mData = new char*[capacity]();
        this->ind = -1;
        this->allocatedData = 0;
    }

    ~CustomStack() {
        for (int i = 0; i < allocatedData; ++i) {
            delete[] mData[i];
        }
        delete[] mData;
    }

    void push(const char *val) {
        if (ind + 1 == capacity) {
            extend(capacity);
        }
        delete[] mData[ind+1];
        mData[++ind] = new char[strlen(val) + 1];
        if (ind + 1 > allocatedData)
            allocatedData++;
    }
};
```



```

        strcpy(mData[ind], val);
    }

    void pop() {
        if (empty()) {
            cerr << "Bad pop\n";
            return;
        }
        ind--;
    }

    char *top() const {
        if (!empty())
            return mData[ind];
        else {
            cerr << "Empty stack\n";
            return nullptr;
        }
    }

    size_t size() const {
        return ind + 1;
    }

    bool empty() const {
        return ind == -1;
    }

private:
    void extend(int n) {
        char **temp = new char*[size() + n]();
        for (int i = 0; i < size(); ++i) {
            temp[i] = mData[i];
        }
        delete[] mData;
        mData = temp;
        capacity += n;
    }

```

```
}
```

protected:

```
char **mData;  
int ind;  
int capacity;  
int allocatedData;
```

```
};
```

```
int main() {
```

```
    CustomStack stack = CustomStack();  
    char str[STR_SIZE];  
    fgets(str,STR_SIZE,stdin);  
    char *lstr = strchr(str,'<');  
    char *rstr = strchr(str,'>');  
    bool isRight = true;  
    while(lstr != nullptr && rstr != nullptr){  
        lstr++;  
        if(strncmp(lstr, "br", 2) == 0 || strncmp(lstr, "hr", 2) == 0){  
            lstr = strchr(rstr + 1,'<'), rstr = strchr(rstr + 1,'>');  
            continue;  
        }  
        if(*lstr == '/'){\br/>            if(!stack.empty() && strncmp(stack.top(), lstr + 1, rstr - lstr - 1) == 0)  
                stack.pop();  
            else{\br/>                isRight = false;  
                break;  
            }  
        } else{\br/>            stack.push(lstr);  
        }  
        lstr = strchr(rstr + 1,'<'), rstr = strchr(rstr + 1,'>');  
    }  
    if(isRight && stack.empty())  
        cout << "correct";  
    else  
        cout << "wrong";  
    return 0;
```

}