**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. 3344 |  | Пачев Д.К. |
| Преподаватель |  | Глазунов С.А. |

Санкт-Петербург

2024

**Цель работы**

Написать программу на языке С++, в которой реализуется структура данных stack на базе массива, с помощью которой проверяется валидность html кода.

**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](https://ru.wikipedia.org/wiki/HTML" \t "_blank)-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**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

## **Выполнение работы**

Структура данных реализуется с помощью класса CustomStack

- CustomStack() – конструктор, в котором выделяется память для mData и значение top\_elem устанавливается -1.

- ~CustomStack() – деструктор, в котором происходит очищение памяти, выделенной под mData.

- push() – метод для добавления элемента в стек, выделяет память под новый элемент, и копирует туда значение.

- pop() – метод для удаления последнего элемента стека, внутри очищается память и уменьшается индекс top\_elem.

- top() – метод, который возвращает верхний элемент стека, обращаясь к элементу массива mData по индексу top\_elem.

- size() – метод, возвращающий размер стека.

- emty() – метод, возвращающий true - если стек пуст, false - если стек не пуст.

- extend() – метод, расширяющий массив mData на n ячеек

Функция is\_valid\_html() принимает на вход html код, с помощью цикла while() проходится по тексту, заполняет стек. Если теги расставлены верно, то стек должен быть пустой, соответственно возвращает значение true или false.

В функции main() реализуется считывание данных и вызов функции is\_valid\_html(), и в зависимости от возвращенного значения выводит «correct» или «wrong».

**Тестирование**

Результаты тестирования представлены в Таблице 1

Таблица 1 - Результаты тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Комментарии |
|  | <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> | correct | Верно |
|  | <html>hello<body></html></body> | wrong | верно |

**Выводы**

В ходе лабораторной работы была написана программа на языке С++, в которой был реализован класс для представления структуры данных stack, а также написана функция для проверки валидности html кода при помощи этой структуры.

# ПРИЛОЖЕНИЕ А ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

#include <iostream>

#include <cstring>

class CustomStack {

public:

CustomStack() {

this->mData = new char \*;

top\_elem = -1;

}

~CustomStack() {

delete[] mData;

}

void push(const char \*val) {

this->mData[++top\_elem] = new char[strlen(val) + 1];

strcpy(mData[top\_elem], val);

}

void pop() {

if (top\_elem >= 0) {

delete[] mData[top\_elem];

top\_elem--;

}

}

char \*top() {

if (top\_elem >= 0) {

return mData[top\_elem];

}

return nullptr;

}

size\_t size() {

return top\_elem + 1;

}

bool empty() {

return top\_elem == -1;

}

void extend(int n) {

char \*\*temp = new char \*[size() + n];

for (size\_t i = 0; i < size(); i++) {

temp[i] = mData[i];

}

delete[] mData;

mData = temp;

}

protected:

char \*\*mData;

private:

int top\_elem;

};

bool is\_valid\_html(const char \*code) {

CustomStack stack;

const char \*p = code;

while (\*p) {

if (\*p == '<') {

if (\*(p + 1) == '/') {

if (stack.empty()) {

return false; // нет открывающего тега для закрывающего

}

const char \*top = stack.top();

char \*tag\_name = strdup(p + 2);

const char \*end\_tag\_name = strchr(tag\_name, '>');

char \*value = strndup(tag\_name, end\_tag\_name - tag\_name + 1);

if (strcmp(top + 1, value) == 0) {

stack.pop();

} else {

return false; // несоответствие открывающего и закрывающего тегов

}

} else {

if (strncmp(p, "<br>", 4) == 0 || strncmp(p, "<hr>", 4) == 0) {

} else {

const char \*tag\_name = p;

const char \*end\_tag = strchr(tag\_name, '>');

stack.push(strndup(tag\_name, end\_tag - tag\_name + 1));

}

}

}

p++;

}

return stack.empty(); // если стек пуст, то все теги закрыты верно

}

int main() {

char str[3001];

std::cin.getline(str, 3001);

str[strlen(str)] = '\0';

bool a = is\_valid\_html(str);

if (a) {

std::cout << "correct\n";

} else {

std::cout << "wrong\n";

}

return 0;

}