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 C++, 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-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:
, <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
1.	<html><head><title>HTML Document</title></head><body> This text is bold, this is bold and italics </body></html>	correct	Верно
2.	<html>hello<body></html></body>	wrong	верно

Выводы

В ходе лабораторной работы была написана программа на языке C++, 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;

}
```