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 _____ Ханнанов А.Ф.

Преподаватель _____ Глазунов С.А.

Санкт-Петербург

2024

Цель работы

Получить представление о работе с ООП на языке C++. Научиться реализовывать стек при помощи класса.

Задание.

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

Был создан класс CustomStack. Поля и методы класса описаны в задании.

Входная строка записывается в переменную text. Цикл пробегает по ней, если найден символ “<” программа начинает запись тэга в переменную tag. Переменная flag хранит определяет, является ли тэг открывающим или закрывающим. После записи тэга проверяется крайнее значение стека. Если записанный тэг является закрывающим, то он сравнивается со значением стека. Если они равны, то этот элемент удаляется из стека, иначе программа возвращает “wrong” и завершается. Если тэг открывающий, то он записывается в стек. Если цикл завершился, то начинается проверка размера стека. Если стек пуст, то возвращается “correct”, иначе “wrong”.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Коммен тарии
1.	<html><head><title>HTML Document</title></head><body> This text is bold, this is bold and italics </body></html>	correct	-

Выводы

Изучены особенности работы с ООП в языке C++. Исследован новый способ создания динамических структур данных.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: Khannanov_Artem_lb4.c

```
class CustomStack {
    public:
        CustomStack() {
            stackSize = 0;
            capacity = 100;
            mData = new char* [capacity];
        }

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

            delete[] mData;
        }

        void
        push(const char* val) {
            if (stackSize >= capacity) extend(capacity);

            mData[stackSize] = new char [strlen(val) + 1];
            strcpy(mData[stackSize++], val);
        }

        void
        pop() {
            delete[] mData[stackSize - 1];
            stackSize--;
        }

        char*
        top() {
            return mData[stackSize - 1];
        }

        size_t
        size() {
            return stackSize;
        }

        bool
        empty() {
            return stackSize == 0;
        }

        void
        extend(int n) {
            capacity += n;
            char** newmData = new char* [capacity];
```

```
            for (size_t i = 0; i < stackSize; i++) {
                newmData[i] = mData[i];
            }
            delete[] mData;
            mData = newmData;
        }

    private:
        size_t stackSize;
        size_t capacity;

    protected:
        char** mData;
};


int
main() {
    char* text, * tag;
    int flag = 0, tagLen = 0, isCorrect;
    text = new char [3001];
    tag = new char [3001];
    fgets(text, 3000, stdin);

    CustomStack stack;

    for (size_t i = 0; i < strlen(text); i++) {
        if (text[i] == '<') {
            flag = 1;
        } else if (flag == 1 && text[i] == '/') {
            flag = 2;
        } else if (text[i] == '>') {
            tag[tagLen] = '\0';
              if (flag == 1 && strcmp("<hr", tag) != 0 && strcmp("<br",
tag)) {
                stack.push(tag);
            } else if (flag == 2) {
                isCorrect = 1;
                for (size_t j = 1; j < strlen(stack.top()); j++) {
                    if (tag[j + 1] != stack.top()[j]) isCorrect = 0;
                }
                if (isCorrect == 1) stack.pop();
            }

            flag = 0;
            tagLen = 0;
        }

        if (flag > 0) tag[tagLen++] = text[i];
    }
    if (stack.empty()) {
        cout << "correct";
    } else {
        cout << "wrong";
    }

    return 0;
```

```
}
```