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. 3342 _____ Лапшов К.Н.

Преподаватель _____ Глазунов С.А.

Санкт-Петербург

2024

Цель работы

Реализовать динамическую структуру данных типа стек. Написать программу на языке программирования C++, которая будет определять правильные html-строки. Изучить и применить объектно-ориентированную парадигму программирования.

Задание

Вариант 5.

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>.

Класс стека (который потребуется для алгоритма проверки парности тегов) требуется реализовать самостоятельно на базе списка. Для этого необходимо:

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

Перечень методов класса стека, которые должны быть реализованы:

void push(const char* tag) - 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

Выполнение работы

Был создана класс реализующий стек на базе линейного списка (заранее подготовленной структуры ListNode). Были реализованы следующие методы класса:

Конструктор CustomStack(); - инициализирует новый объект класса.

void push(const char* tag); - добавляет элемент в стек. Создает новый элемент списка, делая его головным, меняя ссылку на следующий элемент.

void pop(); - удаляет головной элемент стека, премещает голову на следующий элемент в списке. Уменьшает значение переменной счетчика класса на 1.

char* top(); - возвращает данные головного элемента стека.

size_t size(); - возвращает значение переменной счетчика длинны элементов класса.

bool is_empty(); - проверяет стек на пустоту.

Деконструктор ~CustomStack(); - удаляет все элементы стека с помощью функции pop().

Также была реализована функция проверки html строки на валидность:

bool checkHtmlString(string htmlString); - с помощью цикла проходится по все переданной строке. Находит содержимое тэгов используя функцию substr, после этого сохраняет содержимое тэга в стек, если первой символ тэга не равен “/”, что означает закрытие тэга. Если у нас закрытый тэг (наличие в начале символа “/”), то этот закрытый тэг проверяется с головным элементом стека. Если проверка прошла успешно, то удаляется головной элемент стека, если нет, то это означает коллизию html тэгов, а значит, строка не валидна. После проверки возвращает переменную типа bool, отвечающую за валидность строки.

Разработанный программный код см. в приложении А.

Выводы

Была создана программа на языке C++, использующая объектно-ориентированный подход, для проверки правильности html-строк с использованием структуры данных типа "стек" и алгоритма проверки правильности тэговой последовательности.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

```
void memoryError(){
    cout << "Memory allocating error!";
    exit(1);
}

class CustomStack{
    private:
        size_t sizeOfStack;
    protected:
        ListNode* mHead;

    public:
        CustomStack(){
            this->mHead = nullptr;
            this->sizeOfStack = 0;
        }

        void push(const char* tag){
            ListNode* newListNode = new ListNode;
            if(newListNode == nullptr) memoryError();

            newListNode->mData = new char[strlen(tag) + 1];
            if(newListNode->mData == nullptr) memoryError();
            strcpy(newListNode->mData, tag);

            newListNode->mNext = this->mHead;
            this->mHead = newListNode;

            this->sizeOfStack++;
        }

        void pop(){
            if(!this->is_empty()){
                ListNode* tmp = this->mHead->mNext;
                delete[] mHead->mData;
                delete mHead;

                this->mHead = tmp;
                this->sizeOfStack--;
            }
        }

        char* top(){
            if(!this->is_empty()){
                return this->mHead->mData;
            }
            return nullptr;
        }

        size_t size(){
```

```
            return this->sizeOfStack;
        }

        bool is_empty(){
            return this->sizeOfStack == 0;
        }

        ~CustomStack(){
            while (!this->is_empty()) {
                this->pop();
            }
        }

};

bool checkHtmlString(string htmlString){
    size_t start_pos = 0, end_pos = 0;
    string currentTag;
    CustomStack tagStack;

    bool is_correct = true;
         while  ((start_pos  =  htmlString.find('<',  end_pos))  !=
string::npos){
        end_pos = htmlString.find('>', start_pos);
           currentTag = htmlString.substr(start_pos + 1, end_pos -
start_pos - 1);

        if(currentTag != "br" && currentTag != "hr"){
            if(currentTag[0] != '/'){
                tagStack.push(currentTag.c_str());
            }else{
                 if(strcmp(tagStack.top(), currentTag.c_str() + 1) !
= 0){
                    is_correct = false;
                    break;
                }else{
                    tagStack.pop();
                }
            }
        }
    }

    return is_correct;
};

int main(){
    string text;
    getline(cin,text);

    bool result = checkHtmlString(text);

    if(result){
        cout << "correct";
    }else{
        cout << "wrong";
    }
    return 0;
}
```