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. 3341 _____ Моисеева А. Е.

Преподаватель _____ Глазунов С.А.

Санкт-Петербург

2024

Цель работы

Цель лабораторной работы – освоить основы языка Си++, ознакомиться с понятиями таких динамических структур, как стек и очередь, научиться реализовывать их самостоятельно. Для этого необходимо:

Изучить синтаксис языка Си++, его базовые механизмы и принципы работы

Изучить понятия стек и очередь

Научиться самостоятельно реализовывать такие динамические структуры на основе массива и связного списка

Написать программу на Си++, которая будет проверять валидность html-страницы на базе стека.

Задание

Вариант 4

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

Примечания:

Указатель на массив должен быть protected.

Подключать какие-то заголовочные файлы не требуется, всё необходимое подключено(<cstring> и <iostream>).

Предполагается, что пространство имен std уже доступно.

Использование ключевого слова using также не требуется.

Пример:

Входная строка:

<html><head><title>HTML Document</title></head><body> This text is bold,
this is bold and italics </body></html>

Результат:

correct

Выполнение работы

1. Создание класса CustomStack, реализующий работу стека на базе массива. Он работает с данными типа char*. Методы public:

конструктор - инициализирует стек с начальной емкостью 10 элементов

деструктор – очищает память, выделенную для стека

void push(const char *) – добавляет элемент в стек, также при необходимости увеличивает его ёмкость с помощью void extend(int)

void pop() – удаляет верхний элемент стека

char *top() – возвращает значение верхнего элемента стека

size_t size() – возвращает количество элементов стека

bool empty() – показывает, пуст ли стек

void extend(int) – увеличивает ёмкость стека на определённое количество единиц

2. Методы private:

size_t mCapacity – ёмкость стека

size_t mSize – размер стека

3. Функции программы:

Функция readData() – считывает пользовательскую строку, подающуюся на вход программе.

Функция bool checkTags(const string, CustomStack&) – проверка корректности кода html-страницы. Производится проход по символам входной строки, при обнаружении открывающего тега (<tag>) содержимое извлекается и проверяется, является ли он одиночным тегом (br или hr). Если он не является таковым – добавляется в стек open_tags. При обнаружении закрывающего тега проводится проверка на наличие парного открывающего тега в стеке, если он найден, то удаляется из стека, иначе функция возвращает false. После прохождения по всей строке если стек пуст, функция возвращает true, иначе false.

Функция void printResult(bool) – выводит результат проверки валидности расстановки тегов.

Функция int main(): происходит считывание входной строки (кода html-страницы), создаётся стек для хранения тегов, затем происходит проверка с помощью фнукции checkTags и вывод результата через printResult.

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.	<html><body> Hello, world! </body></html>	correct	Программа показывает корректность кода html-страницы.
2.	<html><body> Hello, world!</body></html>	wrong	Программа показала, что расстановка тегов некорректна – закрывающий тег не соответствует открывающему .
3.	<html><body> </ body></html>	correct	Программа показывает корректность кода html-страницы (присутствует одиночный тег).
4.	<html><body> </ body></html>	wrong	Программа показала, что расстановка тегов некорректна – отсутствует парный открывающий тег для .

Выводы

При выполнении работы были освоены основы языка программирования Си++, что помогло приобрести навыки работы с динамическими структурами данных. В том числе были изучены и самостоятельно реализованы на базе массива и связного списка такие структуры, как стек и очередь.

В результате выполненной работы была написана программа, проверяющая корректность расстановки тегов в коде html-страницы. Для этого была реализована работа стека на базе массива, что позволило эффективно обрабатывать входные данные и определять их валидность.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

```
class CustomStack{
public:
    CustomStack(){
        mCapacity = 10;
        mSize = 0;
        mData = new char *[mCapacity];
    }

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

    void push(const char *val){
    if (mSize >= mCapacity){
        extend(mCapacity);
    }
    mData[mSize] = new char[strlen(val) + 1];
    strcpy(mData[mSize], val);
    ++mSize;
}

    void pop(){
        if (!empty()){
            delete[] mData[mSize - 1];
            --mSize;
        }
    }

    char *top(){
        if (!empty()){
            return mData[mSize - 1];
        }
        return nullptr;
    }

    size_t size(){
        return mSize;
    }

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

    void extend(int n){
        mCapacity += n;
        char **newData = new char *[mCapacity];
        for (int i = 0; i < mSize; ++i){
            newData[i] = mData[i];
        }
```

```
            delete[] mData;
            mData = newData;
        }

    private:
        size_t mCapacity;
        size_t mSize;
    protected:
        char **mData;
    };

    string readData(){
        string data;
        getline(cin, data);
        return data;
    }

    bool checkTags(const string data, CustomStack& open_tags){
        for (int i = 0; i < data.size(); i++) {
            char res[10];
            res[0] = '\0';
            if (data[i] == '<') {
                int j = i + 1, n = 0;
                while (data[j] != '>'){
                    res[n] = data[j];
                    n++; j++;
                }
                res[n] = '\0';
                if (res[0] == '/'){
                    char* check = open_tags.top();
                    for (int k = 1; res[k]; k++)
                        if (check[k - 1] != res[k]) return false;
                    open_tags.pop();
                }
                 else if (strcmp(res, "br") != 0 && strcmp(res, "hr") !=
0) open_tags.push(res);
                i = j;
            }
        }
        return true;
    }

    void printResult(bool value){
        if (value) cout << "correct";
        else cout << "wrong";
    }

    int main(){
        string data = readData();
        CustomStack open_tags;
        printResult(checkTags(data, open_tags));
        return 0;
    }
```