**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

отчет

**по лабораторной работе №4**

**по дисциплине «Программирование»**

Тема: Динамические структуры данных

|  |  |  |
| --- | --- | --- |
| Студент гр. 2383 |  | Мордасов Е.Д. |
| Преподаватель |  | Азаревич А.Д. |

Санкт-Петербург

2023

**Цель работы.**

Написать программу для выполнения лабораторной работы, а также изучить и научится применять на практике двусвязные списки.

**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

**Примечания:**

1. Указатель на массив должен быть protected.
2. Подключать какие-то заголовочные файлы не требуется, всё необходимое подключено(<cstring> и <iostream>).
3. Предполагается, что пространство имен std уже доступно.
4. Использование ключевого слова using также не требуется.

**Выполнение работы.**

Для написания программы были подключены и использованы библиотеки <iostream>, <cstring>. В начале кода определен класс CustomStack, который представляет стек для хранения тегов. Стек реализован на основе динамического массива указателей на char\*\*. Класс содержит методы для работы со стеком: push, pop, top, size, empty, а также приватные поля для хранения данных стека. Далее определена функция isValidhtml(), которая принимает исходную строку и возвращается True, если она верна. Внутри функции посимвольно проверяется строка и ищутся символы «< >», если они найдены, то запоминается строка между ними и кладется в стек. Далее если найдены символы «< >» вместе с «/» и строки совпадают, то она удаляется из стека. Если по итогу оказалось, что стек не пуст или в какой-то момент был не найден закрывающий тег, то функция возвращается false. Далее в функции main() в зависимости от возвращенного результата выводится информация о поданной строке, «correct» или «wrong».C результатами тестирования можно ознакомиться в таблице 1.

Таблица 1 – Результаты

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Комментарии |
| 1 | <h1>text!</h2> | wrong | Проверяем разные имена одинаковых тегов. |
| 2 | <h1>Text</h1> | correct | Проверяем одинаковые имена одинаковых тегов. |
| 3 | <br><i>Text</i> | correct | Проверяем тег, который не требует закрытия. |

**Выводы.**

Была разработана программа на языке программирования «C++» для выполнения представленного задания. Также изучены классы в С++ и работа со строками. Исходный код можно посмотреть в приложении А.

**ПРИЛОЖЕНИЕ А**

**ИСХОДНЫЙ КОД ПРОГРАММЫ**

Название файла: main.cpp

#include <iostream>

#include <cstring>

class CustomStack {

public:

CustomStack(int capacity = 10) : mCapacity(capacity), mSize(0) {

mData = new char\*[mCapacity];

}

~CustomStack() {

clear();

delete[] mData;

}

void push(const char\* val) {

if (mSize == mCapacity) {

extend(10);

}

mData[mSize] = new char[strlen(val) + 1];

strcpy(mData[mSize], val);

mSize++;

}

void pop() {

if (mSize > 0) {

delete[] mData[mSize - 1];

mSize--;

}

}

char\* top() const {

if (mSize > 0) {

return mData[mSize - 1];

}

return nullptr;

}

size\_t size() const {

return mSize;

}

bool empty() const {

return (mSize == 0);

}

private:

void clear() {

while (mSize > 0) {

pop();

}

}

void extend(int n) {

char\*\* newData = new char\*[mCapacity + n];

memcpy(newData, mData, mSize \* sizeof(char\*));

delete[] mData;

mData = newData;

mCapacity += n;

}

protected:

char\*\* mData;

size\_t mCapacity;

size\_t mSize;

};

bool isValidHTML(const std::string& html) {

CustomStack stack;

size\_t pos = 0;

while (pos < html.length()) {

if (html[pos] == '<') {

size\_t tagStart = pos + 1;

size\_t tagEnd = html.find('>', tagStart);

if (tagEnd != std::string::npos) {

std::string tag = html.substr(tagStart, tagEnd - tagStart);

if (tag[0] == '/') {

if (!stack.empty()) {

std::string topTag = stack.top();

if (topTag == tag.substr(1)) {

stack.pop();

} else {

return false;

}

} else {

return false;

}

} else if (tag != "br" && tag != "hr") {

stack.push(tag.c\_str());

}

pos = tagEnd + 1;

} else {

return false;

}

} else {

pos++;

}

}

return stack.empty();

}

int main() {

std::string html;

std::getline(std::cin, html);

bool isValid = isValidHTML(html);

if (isValid) {

std::cout << "correct" << std::endl;

} else {

std::cout << "wrong" << std::endl;

}

return 0;

}