**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 |  | Вердин К.К |
| 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](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

## Выполнение работы

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

При помощи цикла и функкции getline считывается входная строка до символа “>”. Затем в цикле проверяется наличие символа “<” и строка добавляется в стэк.

Далее была реализована функция bool isValidHtml(CustomStack &stack).

Объявляется новая переменная CustomStack newstack в который в дальнейшем будут добавляться закрывающие теги. Далее при помощи цикла проверяются условия:

Если последний элемент стэка является закрывающим тэгом, то он добавляется в новый стэк и удаляется из старого;

Иначе, если последний элемент нового стека явлчется закрывающим тэгом последнего элемента старого, то оба элемента удаляются;

Иначе, если последний элемент старого стэка является тэгом, который не требует закрывающего тэга, то он удаляется

В противном случае функция возвращает false.

В зависимости от значения функции будет выведена строка. Если функция вернула false, то выведется “wrong”, если true, то выведется “correct”

## Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Комментарии |
|  | <html><head><title>HTML Document</title></head><body><p><b>This text is bold,<br><i>this is bold and italics</i></b></p></body></html> | correct | - |

## Выводы

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

# Приложение А Исходный код программы

#include <string>

#include <cstdlib>

using std::cin;

using std::cout;

using std::string;

using std::realloc;

class CustomStack

{

public:

CustomStack()

{

mDataSize = 0;

mDataCapacity = 100;

mData = new char \*[mDataCapacity];

}

~CustomStack()

{

for (size\_t i = 0; i < mDataSize; i++)

delete[] mData[i];

delete[] mData;

}

void push(const char \*val)

{

if (mDataCapacity < mDataSize + 1)

extend(10);

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

strcpy(mData[mDataSize++], val);

}

void pop()

{

delete[] mData[mDataSize - 1];

mDataSize--;

}

char \*top()

{

return mData[mDataSize - 1];

}

size\_t size()

{

return mDataSize;

}

bool empty()

{

return mDataSize == 0;

}

void extend(int n)

{

mDataCapacity += n;

mData = (char\*\*)realloc(mData, sizeof(char\*) \* mDataCapacity);

}

void print()

{

for (int i = mDataSize - 1; i >= 0; i--)

{

cout << mData[i] << '\n';

}

}

protected:

size\_t mDataSize;

size\_t mDataCapacity;

char \*\*mData;

};

bool isVoidElement(char \*el)

{

if (strcmp(el, "br") == 0 || strcmp(el, "hr") == 0)

return true;

return false;

}

bool isValidHtml(CustomStack &stack)

{

CustomStack newstack;

while (!stack.empty())

{

char \*top = stack.top();

if (strchr(stack.top(), '/'))

{

newstack.push(top);

stack.pop();

}

else if (!newstack.empty() && strstr(newstack.top(), top))

{

stack.pop();

newstack.pop();

}

else if (isVoidElement(top))

stack.pop();

else

return false;

}

return true;

}

int main()

{

CustomStack stack;

string line;

for (string line; getline(cin, line, '>');)

{

if (line.find("<") != std::string::npos)

{

stack.push(line.substr(line.find("<") + 1).data());

}

}

if (isValidHtml(stack))

cout << "correct\n";

else

cout << "wrong\n";

return 0;