**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 |  | Иванов Д. М. |
| Преподаватель |  | Глазунов С. А. |

Санкт-Петербург

2024

## Цель работы

Изучить динамические структуры данных, основы ООП на языке С++. Применить эти знания для решения поставленной задачи.

## Задание

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: <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:

1) void push(const char\* val) - добавляет новый элемент в стек

2) void pop() - удаляет из стека последний элемент

3) char\* top() - доступ к верхнему элементу

4) size\_t size() - возвращает количество элементов в стеке

5) bool empty() - проверяет отсутствие элементов в стеке

6) extend(int n) - расширяет исходный массив на n ячеек

## Выполнение работы

Рассмотрим структуру класса CustomStack. Она имеет 2 поля: массив данных и индекс верхнего значения.

1) CustomStack() - конструктор класса, присваиваются начальные значения для полей

2) push — выделяется память для следующего элемента стека и копирование в эту область передаваемого значения

3) ~CustomStack — очищение полей при уничтожении экземпляра

4) pop — если массив не пустой, то идет очищение памяти и уменьшение значения поля индекса

5) empty — проверка на пустой стек по значению поля индекса

6) size — вывод размера через значения индекса

7) extend — выделение памяти для n новых значений

Также были реализованы в программе функции для работы алгоритма стека:

main: прописан алгоритм стека и проверка на правильность введенной строки

equal: сравнение двух тегов на их равность(закрытый соответствует открытому)

memory\_error: вывод ошибки в случае неуспешного выделения памяти

## Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Комментарий |
|  | <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 | Верный вывод |

## Выводы

Был написан класс для реализации алгоритма стека на языке С++. С его помощью была реализована программа для проверки корректности введенной html-строки.

# Приложение А Исходный код программы

Название файла: main.cpp

void memory\_error(){

cout << "Memory error!" << endl;

exit(1);

}

class CustomStack {

public:

CustomStack() {

mData = nullptr;

topIndex = -1;

};

void push(const char\* val) {

extend(1);

memcpy(mData[++topIndex], val, strlen(val));

mData[topIndex][strlen(val)] = '\0';

}

void pop(){

if (topIndex >= 0){

delete[] mData[topIndex--];

}

}

~CustomStack() {

for (int i = 0; i <= topIndex; i++){

delete mData[i];

}

delete[] mData;

};

char\* top(){

if (topIndex >= 0){

return mData[topIndex];

}

return NULL;

}

bool empty(){

return topIndex == -1;

}

size\_t size(){

return topIndex + 1;

}

void extend(int n){

char\*\* NewData = new char\* [topIndex + 1];

if (NewData == nullptr)

memory\_error;

for (int i = 0; i <= topIndex; i++){

NewData[i] = new char [300];

if (NewData[i] == nullptr)

memory\_error;

memcpy(NewData[i], mData[i], strlen(mData[i]));

NewData[i][strlen(mData[i])] = '\0';

}

for (int i = 0; i <= topIndex; i++){

delete [] mData[i];

}

delete [] mData;

mData = new char\* [topIndex + 1 + n];

if (mData == nullptr)

memory\_error;

for (int i = 0; i <= topIndex; i++){

mData[i] = new char [300];

if (mData[i] == nullptr)

memory\_error;

memcpy(mData[i], NewData[i], strlen(NewData[i]));

mData[i][strlen(NewData[i])] = '\0';

}

for (int i = 1; i <= n; i++){

mData[topIndex + i] = new char [300];

}

for (int i = 0; i <= topIndex; i++){

delete[] NewData[i];

}

delete[] NewData;

}

private:

int topIndex;

protected:

char\*\* mData;

};

int equal(char\* str\_start, char\* str\_end){

if (strlen(str\_end) == strlen(str\_start) + 1){

int flag = 1;

for (int i = 0; i < strlen(str\_start); i++){

if (str\_start[i] != str\_end[i + 1]){

flag = 0;

return 0;

}

}

if (flag == 1)

return 1;

}

return 0;

}

int main(){

CustomStack obj;

char\* text = new char [3000];

if (text == nullptr)

memory\_error();

int result = 0;

int flag = 0;

int index = 0;

char\* str = new char [3000];

if (str == nullptr)

memory\_error;

fgets(text, 3000, stdin);

for (int i = 0; i < strlen(text); i++){

if (text[i] == '<'){

flag = 1;

continue;

}

if (text[i] == '>'){

str[index] = '\0';

if (!(strlen(str) == 2 && ((str[0] == 'b' && str[1] == 'r') || (str[0] == 'h' && str[1] == 'r')))){

if (str[0] != '/'){

obj.push(str);

}

if (str[0] == '/' && (equal(obj.top(), str) == 1)){

obj.pop();

result = 0;

}

else if (str[0] == '/' && (equal(obj.top(), str) == 0)){

obj.push(str);

result = 1;

}

}

flag = 0;

delete [] str;

str = new char [3000];

if (str == nullptr)

memory\_error;

index = 0;

}

if (flag == 1){

str[index++] = text[i];

}

}

if (obj.empty() && result == 0){

cout << "correct" << endl;

}

else{

cout << "wrong" << endl;

}

delete[] text;

delete[] str;

return 0;

}