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

Студентка гр. 3341 _____ Байрам Э.

Преподаватель _____ Глазунов С.А.

Санкт-Петербург

2024

Цель работы

Цель этой работы заключается в том, чтобы создать программу на C++, которая принимает на вход строку с HTML-кодом, проверяет его на валидность и выводит "correct", если HTML-страница валидна, и "wrong", если нет. Для выполнения этой задачи необходимо реализовать класс стека CustomStack, который будет использоваться для проверки корректной вложенности тегов в HTML-коде.

Задание

Необходимо написать программу для проверки валидности HTML-документа с использованием класса CustomStack.

Описание:

HTML-документ состоит из тегов и их содержимого, заключенного в эти теги. Теги могут быть открытыми (<tag>) и закрытыми (</tag>).

Теги могут быть вложенными, но не должны пересекаться.

Некоторые теги не требуют закрывающего тега, например,
 и <hr>.

HTML-документ считается валидным, если каждому открывающему тегу соответствует закрывающий тег.

Для проверки валидности HTML-документа необходимо использовать стек (CustomStack). Каждый открывающий тег добавляется в стек, и при встрече закрывающего тега проверяется соответствие с верхним тегом в стеке.

Если стек пуст и каждому открывающему тегу соответствует закрывающий тег, то HTML-документ считается валидным.

При подготовке презентации вы можете начать с объяснения структуры HTML-документа и алгоритма проверки его валидности. Затем покажите, как реализован класс CustomStack и как он используется для проверки валидности HTML-документа. Наконец, предоставьте пример работы программы на конкретном HTML-документе для наглядности.

Выполнение работы

1. Изучение задания: Внимательно прочитайте задание и убедитесь, что вы полностью понимаете требования.

2. Проектирование класса CustomStack:

Определите структуру класса CustomStack на основе предоставленной спецификации.

Решите, какие переменные и методы вам понадобятся для реализации стека.

3. Реализация класса CustomStack:

Напишите определения методов класса CustomStack.

Убедитесь, что методы работают корректно и соответствуют заданным требованиям.

4. Реализация функции проверки валидности HTML:

Напишите функцию, которая принимает строку с HTML-кодом и использует класс CustomStack для проверки его валидности.

Разработайте алгоритм, который будет использовать стек для проверки соответствия открывающих и закрывающих тегов.

5. Тестирование:

Протестируйте вашу программу на различных примерах HTML-кода.

Убедитесь, что программа корректно определяет валидные и невалидные HTML-документы.

6. Документация и комментарии:

Добавьте комментарии к вашему коду, чтобы другие разработчики могли легко понять его.

Предоставьте описание того, как использовать вашу программу.

7. Подготовка презентации:

Создайте презентацию, в которой объясняете основные концепции вашей программы, предоставляете примеры работы и демонстрируете ее функциональность.

8. Проверка и улучшение:

Проверьте вашу программу на наличие ошибок и несоответствий требованиям.

Улучшите код и исправьте ошибки при необходимости.

Выводы

Понимание HTML-структуры: Работа над этим проектом помогла глубже понять структуру HTML-документов, включая вложенные теги и их взаимодействие.

Работа со стеком: Реализация класса CustomStack позволила углубиться в понимание работы стека и его применения при проверке вложенности тегов.

Алгоритмы проверки: Разработка алгоритма проверки валидности HTML-документа на основе стека требовала тщательного анализа логики работы тегов и обработки различных сценариев.

Тестирование и отладка: Проведение тестирования на различных примерах HTML-кода позволило выявить ошибки и улучшить работу программы. Отладка помогла исправить выявленные проблемы.

Документация и презентация: Подготовка документации и презентации помогла систематизировать знания о разработанной программе и ее функциональности, а также продемонстрировать результаты работы другим.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

```
#define MAX_STR_SIZE 3000

class CustomStack {
public:
	CustomStack()
	{
		mHead = nullptr;
	}

	~CustomStack()
	{
		while (mHead != nullptr) pop();
	}

	void push(const char *tag)
	{
		auto *NewNode = new ListNode;
		NewNode->mData = new char[strlen(tag) + 1];

		strcpy(NewNode->mData, tag);
		NewNode->mNext = mHead;
		mHead = NewNode;
	}

	void pop()
	{
		if (mHead != nullptr)
		{
			ListNode *tmp = mHead;

			mHead = mHead->mNext;
			delete[] tmp->mData;
			delete tmp;
		}
	}

	char *top()
	{
		return mHead != nullptr ? mHead->mData : nullptr;
	}

	size_t size()
	{
		size_t size = 0;

		for (ListNode *tmp = mHead; tmp != nullptr; tmp =
tmp->mNext, size++);

		return size;
	}

	bool empty()
```

```
	{
		return size() == 0;
	}

protected:
	ListNode *mHead;
};

char *getTag(char *str)
{
	static char *localStr = nullptr;
	if (str != nullptr) localStr = str;
	char *startPos = localStr != nullptr ? strchr(localStr, '<') :
nullptr;
	localStr = startPos != nullptr ? strchr(startPos, '>') : nullptr;

	if (localStr != nullptr)
		*(localStr++) = '\0';
	else
		return nullptr;

	return startPos + 1;
}

bool checkHtmlCode(char *htmlCode)
{
	CustomStack TagStack;

	for (char *tag = getTag(htmlCode); tag != nullptr; tag =
getTag(nullptr))
	{
		if (*tag != '/')
		{
			if (strcmp(tag, "br") != 0 && strcmp(tag, "hr") != 0)
TagStack.push(tag);
		}
		else
		{
			if	(TagStack.top()	!=	nullptr	&&
strcmp(TagStack.top(), tag + 1) == 0) TagStack.pop();
			else return false;
		}
	}

	return TagStack.empty();
}

int main()
{
	char htmlCode[MAX_STR_SIZE];
	cin.getline(htmlCode, MAX_STR_SIZE);

	cout << (checkHtmlCode(htmlCode) ? "correct" : "wrong");

	return 0;
}
```