МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЁТ

по лабораторной работе №3

по дисциплине «Алгоритмы и структуры данных»

Тема: Стек, очередь, дек

Студент гр. 8304		Кирьянов Д.И.
Преподаватель		Фирсов М.А.

Санкт-Петербург

Задание.

Вариант №5-В

Правильная скобочная конструкция с тремя видами скобок определяется как:

< текст > ::= < пусто > | < элемент > < текст >

< элемент > ::= < символ > | (< текст >) | [< текст >] | { < текст > }

где < символ > - любой символ, кроме (,) , [,] , { , }.

Проверить, является ли текст, содержащийся в заданном файле F, правильной скобочной конструкцией; если нет, то указать номер ошибочной позиции.

Цель работы.

Решить полученную задачу, реализовав стек на основе массива. Получение навыков работы с нелинейными структурами данных.

Описание алгоритма.

Сначала принимаемое значение записываем в строку. После чего при помощи стека проверяем корректность подаваемых значений. Если ввод выполнен неправильно, то программа сообщает об этом и выводит номер ошибочной позиции. Если ввод выполнен верно, то программа сообщает о том, что ошибок нет.

Описание функций программы:

1) `int check(std::string& array)`

Функция предназначена для проверки введенной строки на корректность. В случае неверного ввода возвращает номер ошибочного элемента. При верном вводе возвращает 0.

array – подаваемая строка

Описание структур данных:

1)
```
class Stack {
    private:
    char* mystack;
    unsigned int index;
    int* position;
```

где mystack – массив для хранения элементов стека,
index – позиция текущего элемента стека,
position – массив для хранения номеров элементов стека в строке.

2) `Stack(unsigned int size)`

Конструктор класса Stack. Выделяет память для массивов и инициализирует начальное значение индекса как 0.

size – размер массива.

3) `void push(char element, int pos)`

Метод для помещения нового элемента в стек и хранения его номера в строке.

element – элемент стека,

p
o
s
– его номер в строке.

`char pop()`

Метод возвращает последний элемент стека и уменьшает индекс на 1.

`bool isEmpty()`

Метод, проверяющий стек на пустоту.

`char top()`

Метод возвращает верхний элемент стека.

`int mistake()`

Метод возвращает номер элемента стека в строке.

`~Stack()`

Деструктор класса Stack. Освобождает выделенную память.

Выводы.

В ходе выполнения работы были изучены нелинейные структуры данных, был получен опыт работы со стеком на основе массива. На мой взгляд данную работу было правильно выполнять через стек, но нецелесообразно реализовывать самостоятельно, когда есть уже реализованные аналоги.

Протокол

Тестирование:

Входные данные	Выходные данные
(sdf[dsf[sdf][](000{sdf}123)j89].)	
	No one mistake
	There is a mistake on 9 place
[{]}	There is a mistake on 3 place
812309({[xsdf])}	There is a mistake on 15 place

Исходный код

lab3.cpp

```
#include <iostream>
#include <string>
#include <fstream>

class Stack {
private:
	char* mystack;
	unsigned int index;
	int* position;
public:
	Stack(unsigned int size) {
		mystack = new char[size];
		position = new int[size];
		index = 0;
	}
	void push(char element, int pos) {
		mystack[index] = element;
		position[index] = pos;
		index++;
	}
	char pop() {
		index--;
		return mystack[index];
	}
	bool isEmpty() {
		return (index == 0);
	}
	char top() {
		if (!isEmpty())
			return mystack[index - 1];
		else
			return '0';
	}
	int mistake() {
		return position[index - 1];
```

```
	}
	~Stack() {
		delete[] mystack;
		delete[] position;
	}
};

int check(std::string& array) {
	Stack object(array.length());
	for (unsigned int i = 0; i < array.length(); i++) {
		if ((array[i] == '(')
			|| (array[i] == '[')
			|| (array[i] == '{')) {
			object.push(array[i], i + 1);
		}
		else if (array[i] == ')') {
			if (object.top() == '(')
				object.pop();
			else
				return (i + 1);
		}
		else if (array[i] == ']') {
			if (object.top() == '[')
				object.pop();
			else
				return (i + 1);
		}
		else if (array[i] == '}') {
			if (object.top() == '{')
				object.pop();
			else
				return (i + 1);
		}
	}
	if (object.isEmpty())
		return 0;
	else
		return object.mistake();
}

int main(int argc, char* argv[]) {
	std::string array;
	if (argc == 1) {
		std::getline(std::cin, array);
		int result = check(array);
		if (result == 0)
			std::cout << "No one mistake" << std::endl;
		else
			std::cout << "There is a mistake on " << result << " place" <<
std::endl;
	}
	else {
		std::ifstream in(argv[1]);
		if (!in.is_open()) {
			std::cout << "Can't open file" << std::endl;
			return 0;
		}
		while (std::getline(in, array)) {
			std::cout << array << "\n";
			int result = check(array);
			if (result == 0)
				std::cout << "No one mistake" << std::endl;
			else
				std::cout << "There is a mistake on " << result << " place" <<
std::endl;
```

```
                }
                in.close();
        }
        return 0;
}
```