ММИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Алгоритмы и структуры данных»

Тема: Взаимно-рекурсивные функции и процедуры. Синтаксический анализатор понятия скобки.

Студент гр. 9384 _____ Николаев А.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2020

Цель работы.

Реализовать синтаксический анализатор понятия скобки на основе взаимной рекурсии.

Задание.

ВАРИАНТ 12.

12. Построить синтаксический анализатор для понятия скобки.

скобки::=квадратные | круглые | фигурные
квадратные::=[круглые фигурные] | +
круглые::=(фигурные квадратные) | -
фигурные::={квадратные круглые} | 0

Выполнение работы.

В начале программа запрашивает у пользователя путь до файла из которого требуется считать данные и отправить их на обработку и название файла, в который будет записан результат работы программы. Программа построчно считывает данные из входного файла и обрабатывает их. В самом начале обработки строки вызывается функция:

bool isSquareBrackets (std::string expression, int& index)

Проверяет что бы выражение состояло из квадратных скобок и скобок или знаков, обязанных находиться внутри этих скобок или знака +.

bool isRoundBrackets (std::string expression, int& index)

Проверяет что бы выражение состояло из круглых скобок и скобок или знаков, обязанных находиться внутри этих скобок или знака -.

bool isFiguredBrackets (std::string expression, int& index)

Проверяет что бы выражение состояло из фигурных скобок и скобок или знаков, обязанных находиться внутри этих скобок или знака 0.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

Входные данные	Выходные данные
+	Yes
-	Yes
0	Yes
[]	No
()	No
{}	No
{-}	No
[()]	No
[()0]	No
[(0+)]	No
({[-0]-}+)	Yes

Выводы.

В ходе выполнения данной лабораторной работы были получены навыки по работе с рекурсивными функциями, а также, разработан синтаксический анализатор понятия скобки.

ПРИЛОЖЕНИЕ

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

```
/*
12. Построить синтаксический анализатор для понятия скобки.
скобки::=квадратные | круглые | фигурные
квадратные::=[круглые фигурные] | +
круглые::=(фигурные квадратные) | -
фигурные::={квадратные круглые} | 0
*/

#include <iostream>
#include <fstream>
#include <string>
#include <iomanip>

bool isSquareBrackets(std::string expression, int& index);
bool isRoundBrackets(std::string expression, int& index);
bool isFiguredBrackets(std::string expression, int& index);

bool isSquareBrackets(std::string expression, int& index)
{
	if (expression[index++] == ']')
		return true;
	else
		index--;

	if (expression[index++] == '+')
		return true;
	else
		index--;

	if (expression[index++] == '[')
	{
		if (isRoundBrackets(expression, index) &&
isFiguredBrackets(expression, index) && isSquareBrackets(expression, index))
			return true;
	}

	index--;
	return false;
}
```

```
bool isRoundBrackets(std::string expression, int& index)
{
	if (expression[index++] == ')')
		return true;
	else
		index--;

	if (expression[index++] == '-')
		return true;
	else
		index--;

	if (expression[index++] == '(')
	{
		if (isFiguredBrackets(expression, index) &&
isSquareBrackets(expression, index) && isRoundBrackets(expression, index))
		return true;
	}

	index--;
	return false;
}

bool isFiguredBrackets(std::string expression, int& index)
{
	if (expression[index++] == '}')
		return true;
	else
		index--;

	if (expression[index++] == '0')
		return true;
	else
		index--;

	if (expression[index++] == '{')
	{
		if (isSquareBrackets(expression, index) &&
isRoundBrackets(expression, index) && isFiguredBrackets(expression, index))
			return true;
	}

	index --;
```

```
	return false;
}

int main()
{
	int index = 0;
	std::string expression;
	std::string filePath;

	std::cout << "Enter fileIn path: ";
	std::cin >> filePath;

	std::ifstream fileIn(filePath);

	if (!fileIn)
	{
		std::cout << "File cannot be opened!\n";
		return 1;
	}

	int offset = 0;
	while (getline(fileIn, expression))
	{
		if (offset < expression.length())
			offset = expression.length();
	}
	offset += 5;

	fileIn.clear();
	fileIn.seekg(0);

	std::cout << "Enter fileOut Path: ";
	std::cin >> filePath;

	std::ofstream fileOut(filePath);

	while (getline(fileIn, expression))
	{
		if (isSquareBrackets(expression, index) || isRoundBrackets(expression,
index) || isFiguredBrackets(expression, index) || expression.length() == index)
		{
			fileOut << std::endl << std::setw(offset) << std::left << expression
<< std::setw(offset) << std::left << "Yes";
```

```
                        std::cout << std::endl << std::setw(offset) << std::left <<
expression << std::setw(offset) << std::left << "Yes";
                }
                else
                {
                        fileOut << std::endl << std::setw(offset) << std::left << expression
<< std::setw(offset) << std::left << "No";
                        std::cout << std::endl << std::setw(offset) << std::left <<
expression << std::setw(offset) << std::left << "No";
                }


                expression.clear();
                index = 0;
        }

        fileIn.close();
        fileOut.close();

        return 0;
}
```