МИНОБРНАУКИ 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: Рекурсия

Студент гр. 8381 _____ Сергеев А.Д.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2019

Цель работы.

Ознакомиться с основными понятиями и приёмами рекурсивного программирования, получить навыки программирования рекурсивных процедур и функций на языке программирования C++. Разработать программу, использующую рекурсию, и сопоставить рекурсивное решение с итеративным решением задачи.

Задание.

Построить синтаксический анализатор для понятия *текст_со_скобками.*

текст_со_скобками::=элемент | *элемент текст_со_скобками*

*элемент::=*A |B | (*текст_со_скобками*) | [*текст_со_скобками*]|

{ текст_со_скобками }

Основные теоретические положения.

Синтаксическим анализатором назовём программу, которая определяет, является ли заданная (входная) последовательность символов *текстом со скобками* или нет.

Синтаксический анализатор для рекурсивно заданных скобок удобно реализовать с помощью рекурсии. Функция, проверяющая выражение на скобки, может запускать внутри себя саму себя для проверки вложенных скобок.

Выполнение работы.

Написание работы производилось на базе операционной системы Ubuntu 18.04 в среде разработки CLion. Сборка, отладка и тестирование также производились в CLion.

На вход программе подаётся последовательность скобок и букв A и B.

После завершения работы функции в консоль выводится:

- Проанализированная часть строки
- Сообщение о том, является ли введенная строка скобками

Оценка эффективности алгоритма.

Алгоритм имеет сложность O(n). Поскольку алгоритм рекурсивный, он может занимать большое количество памяти, если на вход будет предложено много вложенных скобок.

Тестирование алгоритма.

Ввод	Вывод
A(B{A}B)	INPUT IS: A(B{A}B) TEXT WITH BRACKETS!
({)}	INPUT IS: ({)} ERROR!
((A[]B{})A[])B{ABA}	INPUT IS: ((A[]B{})A[])B{ABA} TEXT WITH BRACKETS!
(A{B[ABA]B}A)	INPUT IS: (A{B[ABA]B}A) TEXT WITH BRACKETS!
A[B}{A)B{}	INPUT IS: A[B}{A)B{} ERROR!
(AB{}[]A))A[](AB}	INPUT IS: (AB{}[]A)) ERROR!
{}[SAMPLE]()	INPUT IS: {}[SAM ERROR!

Выводы.

В ходе выполнения лабораторной работы был изучен такой вид программ, как синтаксические анализаторы. Была реализована программа, которая анализирует строку рекурсивным методом, определяя соответствие определению. Было выявлено, что для рекурсивных определений сложно найти оптимальное итеративное решение, при этом рекурсивный метод дает высокую эффективность.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

```
#include <iostream>
#include <fstream>
#include <iomanip>

using namespace std;

bool isBracket(char t);
bool isChar(char t);
bool skipChar(ifstream &text, char breakChar);

int main() {
    setlocale (0,"Rus");

    ifstream infile("input.txt");
    if (!infile) cout << "NO FILE!" << endl;

    cout << "INPUT IS:" << endl;
    bool b = skipChar(infile, 0);

    cout << endl;
    if (b) {
        cout << "TEXT WITH BRACKETS!" << endl;
    } else {
        cout << "ERROR!" << endl;
    }

    return 0;
}

bool isChar(char t) {
    return (t == ' ') || (t == 'A') || (t == 'B');
}

bool isBracket(char t) {
     return (t == '(') || (t == ')') || (t == '[') || (t == ']') || (t == '{') || (t == '}');
}
```

```
bool skipChar(ifstream &text, char breakChar) {
    char t;
    bool b = true;
    while (text >> t) {
        cout << t;

        if (isBracket(t) || !isChar(t)) {
            if (t == breakChar) {
                return true;
            }
            if (t == '(') {
                b &= skipChar(text, ')');
            } else if (t == '[') {
                b &= skipChar(text, ']');
            } else if (t == '{') {
                b &= skipChar(text, '}');
            } else {
                return false;
            }
        }
    }
    return b & (0 == breakChar);
}
```