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. 9303 _____ Хафаева Н.Л.

Преподаватель _____ Филатов Ар.Ю.

Санкт-Петербург

2020

Цель работы.

Ознакомиться с понятием «рекурсия», её приёмах и методах.

Задание.

Вариант 22.

Построить синтаксический анализатор для определённого далее понятия *логическое_выражение*.

логическое_выражение ::= TRUE | FALSE |*идентификатор* |
NOT (*операнд*) |*операция* (*операнды*)

идентификатор::=буква

операция::= AND | OR

операнды::= операнд |*операнд, операнды*

операнд::= логическое_выражение

Основные теоретические положения.

В программировании рекурсия — вызов функции (процедуры) из неё же самой, непосредственно (простая рекурсия) или через другие функции (сложная или косвенная рекурсия). Количество вложенных вызовов функции или процедуры называется глубиной рекурсии. Рекурсивная программа позволяет описать повторяющееся или даже потенциально бесконечное вычисление, причём без явных повторений частей программы и использования циклов.

Рекурсивные функции используют так называемый «Стек вызовов». Когда программа вызывает функцию, функция отправляется на верх стека вызовов. Адрес возврата и локальные переменные функции записываются в стек, благодаря чему каждый следующий рекурсивный вызов этой функции пользуется своим набором локальных переменных и за счёт этого работает корректно. Оборотной стороной этого механизма является то, что при чрезмерно большой глубине рекурсии может наступить переполнение стека вызовов.

Выполнение работы.

Для выполнения задания были реализованы следующие функции: *bool check (string str, int depth)* и *string trim (string str).*

int main() – в данной функции считывается введенное с консоли пользователем имя файла, если файл с таким именем существует, то будет происходить считывание строки из файла и к ней будет применяться функция *check*, также будет выводиться информация о том, является ли данная строка логическим выражением или нет.

bool check (string str, int depth) – функция принимает 2 аргумента: текущую строку и глубину рекурсии. Возвращает true, если строка является логическим выражением, иначе возвращает false. В функции проверяется не является ли строка значениям TRUE или FALSE, если это верно, то функция возвращает true. Если первое слово NOT, AND или OR, и если элемент после этого слова и последний элемент строки равны скобкам, то вызывается рекурсия, иначе функция возвращает false.

string trim (string str) – функция принимает строку и возвращает ее без лишних пробелов в начале и в конце строки.

Выводы.

В процессе выполнения лабораторной работы было изучено понятие рекурсия, её приёмы и особенности.

Была разработана программа, являющаяся синтаксическим анализатором для понятия логического выражения.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл main.cpp

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

using namespace std;

bool check (string str, int depth);
string trim (string str);

int main(){
    string input_file;
    cout << "Введите название файла " << endl;
    getline(cin, input_file);
    ifstream input(input_file);

    if (!input) {
        cout << "Входной файл неверный." << endl;
    } else {
        string str;
        getline(input, str);

        if (check(str, 1)){
            cout << "Это логическое выражение: " << str << endl;
        } else {
            cout << "Это не логическое выражение ";
            cout << str << endl;
        }
    }
    return 0;
}

bool check(string str, int depth) {
    cout << depth << endl << str << endl;

    str = trim(str);

    if (str.length() == 0) {
     cout << "Строка пустая. " << endl;
        return false;
    }

    if (str == "TRUE" || str == "FALSE") {
        return true;
    } else if (str.length() == 1 && isalpha(str[0])) {
        return true;
```

```
    } else if (str.length() >= 3 && str.substr(0, 3) == "NOT") {
        str = trim(str.substr(3, str.length() - 3));

        if (str.length() < 2 || str[0] != '(' || str[str.length() - 1] !=
')') {
            return false;
        }

        return check(str.substr(1, str.length() - 2), depth + 1);
    } else if ((str.length() >= 3 && str.substr(0, 3) == "AND") ||
(str.length() >= 2 && str.substr(0, 2) == "OR")) {
        if (str.substr(0, 2) == "OR") {
            str = trim(str.substr(2, str.length() - 2));
        } else {
            str = trim(str.substr(3, str.length() - 3));
        }

        if (str.length() < 2 || str[0] != '(' || str[str.length() - 1] !=
')') {
            return false;
        }

        str = str.substr(1, str.length() - 2);

        string tmp = "";
        int balance = 0;

        for (int i = 0; i < str.length(); i++) {
            if (str[i] != ',' || (str[i] == ',' && balance != 0)) {
                tmp += str[i];

                if (str[i] == '(') {
                    balance++;
                } else if (str[i] == ')') {
                    balance--;
                }
            } else if (balance == 0) {
                if (!check(tmp, depth + 1)) {
                    return false;
                }
                tmp = "";
            }
        }

        return check(tmp, depth + 1);
    }

    return false;
}
```

```
string trim(string str) {
    int l = 0;
    while (l < str.length() && str[l] == ' ') {
        l++;
    }

    int r = str.length() - 1;
    while (r >= 0 && str[r] == ' ') {
        r--;
    }

    string result = "";
    for (int i = l; i <= r; i++) {
        result += str[i];
    }
    return result;
}
```

ПРИЛОЖЕНИЕ Б

ТЕСТИРОВАНИЕ

Таблица Б.1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1	AND (TRUE, FALSE, NOT (TRUE), S)	2 TRUE 2 FALSE 2 NOT (TRUE) 3 TRUE 2 S Это логическое выражение: AND (TRUE, FALSE, NOT (TRUE), S)	Программа работает правильно.
2	FALSE	1 FALSE Это логическое выражение: FALSE	Программа работает правильно.
3	NOT(NOT(AND(FALSE,TRUE)))	1 NOT(NOT(AND(FALSE,TRUE))) 2 NOT(AND(FALSE,TRUE)) 3 AND(FALSE,TRUE)	Программа работает правильно.

		4 FALSE 4 TRUE Это логическое выражение: NOT(NOT(AND(FALSE,TRUE)))	
4	NOT ()	1 NOT () 2 Это не логическое выражение: NOT ()	Программа работает правильно.
5	a	1 a Это логическое выражение: a	Программа работает правильно.
6	AND (OR (TRUE, FALSE, b))	1 AND (OR (TRUE, FALSE, b)) 2 OR (TRUE, FALSE, b) 3 TRUE 3 FALSE 3 b	Программа работает правильно.

		Это логическое выражение: AND (OR (TRUE, FALSE, b))	
7	AND (TRUE))	1 AND (TRUE)) 2 TRUE) Это не логическое выражение: AND (TRUE))	Программа работает правильно.
8	()	1 () Это не логическое выражение: ()	Программа работает правильно.
9	not (FALSE)	1 not (FALSE) Это не логическое выражение: not (FALSE)	Программа работает правильно.
10	sefdsgsgs	1 sefdsgsgs Это не логическое выражение: sefdsgsgs	Программа работает правильно.