**МИНОБРНАУКИ РОССИИ**
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**
**Кафедра математического обеспечения и применения ЭВМ**

# ОТЧЕТ

**по лабораторная работа №2**

**по дисциплине «Алгоритмы и структуры данных»**

**ТЕМА: ИЕРАРХИЧЕСКИЕ СПИСКИ**

Студент гр. 9303 _____ Павлов Д.Р.

Преподаватель _____ Филатов А.Ю.

Санкт-Петербург

2020

**Цель работы.**

Ознакомиться с понятием «иерархический список», реализовать программу, решающая данную задачу на языке С++.

**Задание.**

Вариант 19.

Арифметическое, проверка синтаксической корректности и деления на 0 (простая), постфиксная форма

**Основные теоретические положения.**

Пусть выражение (логическое, арифметическое, алгебраическое*) представлено иерархическим списком. В выражение входят константы и переменные, которые являются атомами списка. Операции представляются в префиксной форме ( (<операция> <аргументы>) ), либо в постфиксной форме (<аргументы> <операция>) ). Аргументов может быть 1, 2 и более. Например (в префиксной форме): (+ a (* b (- c))) или (OR a (AND b (NOT c))).
В задании даётся один из следующих вариантов требуемого действия с выражением: *проверка синтаксической корректности*, *упрощение* (преобразование), *вычисление*.
Пример *упрощения*: (+ 0 (* 1 (+ a b))) преобразуется в (+ a b).
В задаче *вычисления* на входе дополнительно задаётся список значений переменных

$$( (x_1\ c_1)\ (x_2\ c_2)\ \dots\ (x_k\ c_k) ),$$

где $x_i$ – переменная, а $c_i$ – её значение (константа).
В индивидуальном задании указывается: тип выражения (возможно дополнительно - состав операций), вариант действия и форма записи. Всего 9 заданий.

---

* - здесь примем такую терминологию: в *арифметическое* выражение входят операции +, -, *, /, а в *алгебраическое* – +, -, * и дополнительно некоторые функции.

**Выполнение работы.**

Программа получает строку записанную в файл input.txt затем создаем указатель на head и запускаем функцию foundPostfix, которая проверяет строку, чтобы она была записана в постфиксной форме. В функции foundPostfix начинает считывать список со знака ‘(’, затем проверяет чтобы следующие два символа были операндами, а третий знак операции.

**Вывод.**

В ходе выполнения лабораторной работы была создана программа, принимающая на вход список переменных и их значений, логическое выражение и считает значение данного логического выражения.

Также было изучено понятие «иерархический список» и принципы работы с ним.

# ПРИЛОЖЕНИЕ А

# ИСХОДНЫЙ КОД ПРОГРАММЫ

## Файл main.cpp

```
#include <iostream>
#include <fstream>
#include <variant>

class Node {
    using NodePtr = std::shared_ptr<Node>;

public:
    NodePtr next{ nullptr };
    std::variant<char, NodePtr> value;
    Node() = default;
    ~Node() = default;
};

std::shared_ptr<Node> foundPostfix(std::string& string, int& iter){
    using NodePtr = std::shared_ptr<Node>;
    int counter = 0;
    short int arg = 3;
    NodePtr headLocal = std::make_shared<Node>();
    NodePtr curLocal = headLocal;
    headLocal->value = string[iter];
    while ((iter != string.length()) && (counter < arg)) {
        iter++;
            if (!string.compare(iter, 1, "(")) {
                curLocal->next = std::make_shared<Node>();
                curLocal = curLocal->next;
                curLocal->value = foundPostfix(string, iter);
                counter++;
            } else {
                if(((string[iter]>='a' && string[iter]<= 'z') ||
(string[iter]>='0' && string[iter]<='9')) && counter != 2){
                    if(counter == 1 && string[iter] == '0' && string[iter+1]
== '/'){
                        return headLocal;
                    } else {
                        curLocal->next = std::make_shared<Node>();
                        curLocal = curLocal->next;
                        curLocal->value = string[iter];
                        counter++;
                    }
                }else if(counter == 2 && (string[iter] == '+' || string[iter]
== '-' || string[iter] == '*' || string[iter] == '/')){
                    curLocal->next = std::make_shared<Node>();
                    curLocal = curLocal->next;
                    curLocal->value = string[iter];
                    counter++;
                }else {
                    return headLocal;
                }
            }

    }
    if (string[iter] != ')' && (string[iter-1] != '+' && string[iter-1] != '-
' && string[iter-1] != '*' && string[iter-1] != '/'))
        iter++;
    return headLocal;
```

```
}
int main(){
    std::shared_ptr<Node> head;
    int iter = 0;
    std::ifstream fin("./Tests/input.txt");
    std::string str;
    fin>>std::noskipws;
    if(!fin){
        std::cout<<"Can't open this file!"<<'\n';
        return 1;
    }
    fin>>str;
    //std::cout<<str.length()-1<<'\n';
    head = foundPostfix(str, iter);
    //std::cout<<iter<<'\n';
    if (iter == str.length() - 1)
        std::cout << "Success: Correct syntax!\n";
    else
        std::cout << "Error: Incorrect syntax!\n";
    return 0;
}
```