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

# ОТЧЕТ

**по лабораторной работе №2**

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

**Тема: Иерархические списки**

Студент гр. 7383 ___ Сычевский Р.А.

Преподаватель ___ Размочаева Н.В.

Санкт-Петербург

2018

# Содержание

# 1. ЦЕЛЬ РАБОТЫ

Цель работы: познакомиться с иерархическими списками и использованием их в практических задачах на языке программирования C++.

Формулировка задачи: при помощи логических операций вычислить значение, которое хранится в бинарном коромысле.

# 2. РЕАЛИЗАЦИЯ ЗАДАЧИ

В данной работе используется главная функция (main()) и дополнительные функции (pr_menu(), find_args(), work_with_console(), work_with_file(), full_elem(), find_ans()).

Сначала функция main() вызывает функцию pr_menu(), которая используется для вывода меню. Пользователь выбирает какой-нибудь пункт в меню и вводит число, которое передается в switch(). В зависимости от выбранного пункта выбирается необходимая опция:

1 – ввод логического выражения с клавиатуры;

2 – считывание логического выражения из файла;

0 – выход из программы;

Если пользователь введет не то, что от него ожидает программа, она сообщит ему об этом и попросит ввести число повторно.

При вводе единицы, программа вызывает функцию work_with_console(), которая считывает логическое выражения из консоли и передает его в функцию full_elem().

При вводе двойки, программа спрашивает у пользователя имя файла, в котором хранится логическое выражение для считывания, считывает его и передает в функцию full_elem().

Функция full_elem() получает на вход указатель на корень дерева или поддерева, строку с логическим выражением и строку со значениями переменных. Сначала при помощи функции find_args() full_elem() определяет логическую операцию для данного корня. После этого функция проверяет: является первый аргумент переменной или логическим выражением. Если первый элемент является логическим выражением, то в левом поддереве создается указатель на новый узел и full_elem() вызывает себя же для нового логического выражения. Если первый аргумент является переменной, то в левое поддерево заносится значение переменной, найденное во второй строке. После этого full_elem() проверяет является ли наше логическое выражение выражением с одним аргументом, если да, то full_elem() завершает работу. В противном случае full_elem() проверяет является второй аргумент переменной

или же выражением и вызывает саму себя или нет, как и для первого аргумента.

После этого work_with_console() или work_with_file() вызывают функцию find_ans() которая берет из левого поддерева операцию и вычисляет её для левого и правого поддерева, если они хранят переменные, иначе вызывает себя же для левого и/или правого поддерева. После чего find_ans() возвращает полученное значение выражения, которое выводится в консоль.

# 3. ТЕСТИРОВАНИЕ

## 3.1 ПРОЦЕСС ТЕСТИРОВАНИЯ

Программа собрана в операционной системе Ubuntu 18.04.1 LTS bionic компилятором g++ (Ubuntu 7.3.0-16ubuntu3) 7.3.0. В других ОС и компиляторах тестирование не проводилось.

## 3.2 РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

Тестовые случаи представлены в Приложении А.

Результаты тестирования показали, что программа работает верно, значит поставленная задача выполнена.

# 4. ВЫВОД

В ходе лабораторной работы были изучены основные понятия об иерархических списках, получены навыки создания иерархических списков и функций работы со списками на языке программирования C++.

# ПРИЛОЖЕНИЕ А: ТЕСТОВЫЕ СЛУЧАИ

| № | Ввод | Вывод |
|---|---|---|
| 1 | AND a OR b c<br>a,1 b,0 c,1 | 1 |
| 2 | OR XOR NOT a b AND NOT c d<br>a,0 b,0 c,0 d,0 | 1 |
| 3 | AND a NOT XOR b c<br>a,0 b,0 c,0 | 0 |
| 4 | OR OR OR a XOR b c d e<br>a,0 b,1 c,0 d,0 e,0 | 1 |
| 5 | AND NOT a XOR OR b c d<br>a,0 b,0 c,0 d,1 | 1 |
| 6 | AND AND AND a b c d<br>a,1 b,1 c,1 d,0 | 0 |
| 7 | AND NOT a XOR AND b c OR d e<br>a,0 b,1 c,1 d,0 e,0 | 1 |

# ПРИЛОЖЕНИЕ Б: КОД ПРОГРАММЫ

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <cstring>
#include <cctype>
using namespace std;

struct atom;

struct pairs{
    atom* left;
    atom* right;
};

struct atom{
    bool isleft;
    int action;     // 1-OR, 2-AND, 3-XOR, 4-NOT
    bool flag;
    union{
        int a;
        pairs* next;
    }un;
};

void pr_menu(){
    cout << "Выберите действие:" << endl;
    cout << "1 - ввод с клавиатуры" << endl;
    cout << "2 - ввод из файла" << endl;
    cout << "0 - выход" << endl;
}

int find_arg(string str1){
    if(str1[0] == 'O')
        return 1;
    if(str1[0] == 'A')
        return 2;
    if(str1[0] == 'X')
        return 3;
    if(str1[0] == 'N')
        return 4;
    return 0;
}

int full_elem(pairs* knot, string str1, string str2){ // OR, AND, XOR, NOT
    int FFF = 0;
    int F_LOC = 0;
    knot->left = new atom;
    knot->left->isleft = true;
    knot->left->action = find_arg(str1);
    string tmp;
    switch(knot->left->action){
        case 1:
            F_LOC = 3;
```

```
            tmp = str1.substr(3);
            str1 = tmp;
            break;
        default:
            F_LOC = 4;
            tmp = str1.substr(4);
            str1 = tmp;
    }
    if((str1[0] > 64) && (str1[0] < 91)){
        knot->left->flag = true;
        knot->left->un.next = new pairs;
        FFF += full_elem(knot->left->un.next, str1, str2);
        if(knot->left->action != 4){
            tmp = str1.substr(FFF);
            str1 = tmp;
        }
    } else {
        for(int i = 0; i < str2.length(); i++)
            if(str1[0] == str2[i])
                knot->left->un.a = str2[i+2] - 48;
        if(knot->left->action != 4){
            tmp = str1.substr(2);
            str1 = tmp;
        }
        FFF += 2;
    }
    if(knot->left->action != 4){
        if((str1[0] > 64) && (str1[0] < 91)){
            knot->right = new atom;
            knot->right->isleft = false;
            knot->right->flag = true;
            knot->right->un.next = new pairs;
            FFF += full_elem(knot->right->un.next, str1, str2);
        } else {
            knot->right = new atom;
            for(int i = 0; i < str2.length(); i++)
                if(str1[0] == str2[i]){
                    knot->right->un.a = str2[i+2] - 48;
                    knot->right->flag = false;
                }
            FFF += 2;
        }
    } else {
        knot->right = NULL;
    }
    FFF += F_LOC;
    return FFF;
}

int find_ans(pairs* knot){
    if(knot->left->flag){
        int tmp1 = find_ans(knot->left->un.next);
        delete(knot->left->un.next);
        knot->left->un.next = NULL;
        knot->left->un.a = tmp1;
        knot->left->flag = false;
```

```
    }
    if(knot->left->action != 4)
        if(knot->right->flag){
            int tmp2 = find_ans(knot->right->un.next);
            delete(knot->right->un.next);
            knot->right->un.next = NULL;
            knot->right->un.a = tmp2;
            knot->right->flag = false;
        }
    switch(knot->left->action){
        case 1:
            return (knot->left->un.a | knot->right->un.a);
            break;
        case 2:
            return (knot->left->un.a & knot->right->un.a);
            break;
        case 3:
            return (knot->left->un.a ^ knot->right->un.a);
            break;
        case 4:
            return (!knot->left->un.a);
            break;
    }
}

int work_with_console(){
    cout << "Введите выражение" << endl;
    string str1;
    getline(cin, str1);
    getline(cin, str1);
    cout << "Введите значеия переменных" << endl;
    string str2;
    getline(cin, str2);
    pairs* knot = new pairs;
    full_elem(knot, str1, str2);
    cout << find_ans(knot) << endl;
    delete(knot);
    return 0;
}

int work_with_file(){
    cout << "Введите имя файла, в котором записано выражение и значения
переменных" << endl;
    string file_name;
    cin >> file_name;
    ifstream f;
    f.open(file_name.c_str());
    if (!f){
        cout << "Файл не открыт!" << endl;
        return 0;
    }
    string str1;
    string str2;
    getline(f, str1);
    getline(f, str2);
    cout << str1 << endl << str2 << endl;
```

```
    pairs* knot = new pairs;
    full_elem(knot, str1, str2);
    cout << find_ans(knot) << endl;
    delete(knot);
    return 0;
}

int main(){
    pr_menu();
    int way = 0;
    cin >> way;
    while(way){
        switch (way){
            case 1:
                work_with_console();
                cout << endl;
                pr_menu();
                cin >> way;
                break;
            case 2:
                work_with_file();
                cout << endl;
                pr_menu();
                cin >> way;
                break;
            default:
                cout << "Неверно введены данные!" << endl;
                pr_menu();
                cin >> way;
        }
    }
    return 0;
}
```