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

ОТЧЕТ

по лабораторной работе №2
по дисциплине «Алгоритмы и структуры данных»
Тема: Иерархические списки

Студент гр. 9381 _____ Аухадиев А.А.

Преподаватель _____ Фирсов М.А.

Санкт-Петербург
2020

Цель работы.

Изучить основные понятия иерархического списка и получить опыт работы с ним на языке программирования C++, разработать программу, использующую иерархический список.

Задание.

Вариант 3.

Подсчитать общую длину всех плеч заданного бинарного коромысла bk. Для этого ввести рекурсивную функцию

short Length (**const** БинКор bk).

Основные теоретические сведения.

Бинарное коромысло устроено так, что у него есть два плеча: левое и правое. Каждое плечо представляет собой (невесомый) стержень определенной длины, с которого свисает либо гирька, либо еще одно бинарное коромысло, устроенное таким же образом.

В соответствии с данным выше рекурсивным определением бинарного коромысла представим бинарное коромысло (БинКор) списком из двух элементов

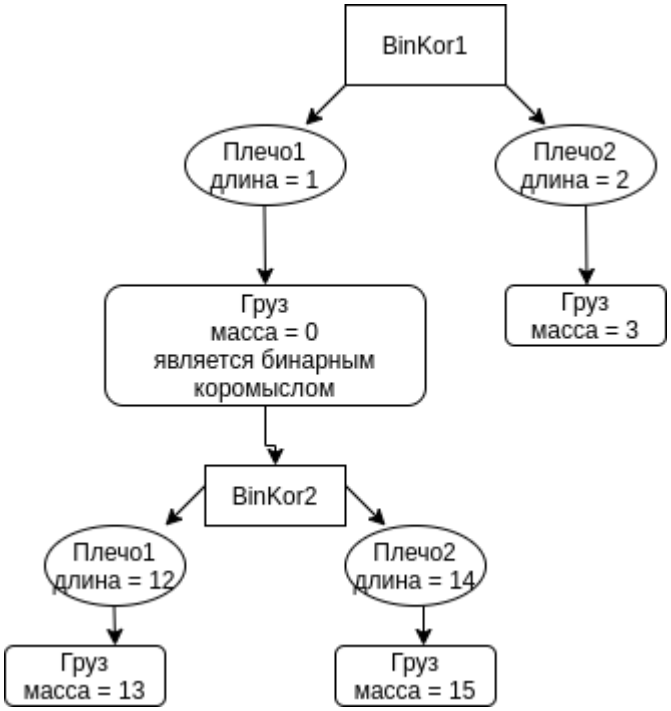
БинКор ::= (Плечо & Плечо), где первое плечо является левым, а второе – правым.

В свою очередь Плечо будет представляться списком из двух элементов

Плечо ::= (Длина & Груз), где Длина есть натуральное число, а Груз представляется вариантами

Груз ::= (Гирька | БинКор).

Коромысло вида ((1((12 13) (14 15))) (2 3)) можно представить следующим образом:



Функции и структуры.

Для выполнения задачи были определены три структуры:

1. binKor (бинарное коромысло): два поля - shoulder* pl1 и shoulder* pl2 (указатели на два плеча);

2. shoulder (плечо): два поля - int l (длина) и cargo* mass (указатель на груз);

3. cargo (груз): три поля - bool tag (= true, если груз является бинарным коромыслом, =false, если груз является гирькой), int m - масса (=0, если tag = true) и binKor* bk - указатель на бинарное коромысло (=nullptr, если tag = false).

Функции, реализованные в программе, можно разделить на три части: сохранение введённых данных в иерархический список, обработка данных (подсчёт общей длины всех плеч) и очистка памяти.

Обработка данных включает в себя работу с тремя рекурсивными функциями:

1. short Length(const binKor* bk) - возвращает сумму результатов функций Length(bk->pl1) + Length(bk->pl2), то есть сумму длин первого и второго плеча со всеми прилегающими к ним другими бинарными коромыслами;

2. short Length(const shoulder* pl) - возвращает сумму поля l (длины плеча) и результата функции Length(pl->mass), который представляет суммарную длину

плеч других бинарных коромысел, которые могут бить связаны с этим плечом.

3. short Length(const cargo* mass) - возвращает результат функции Length(mass->bk), если mass->tag = true, то есть является бинарным коромыслом, а не гирькой.

Таким образом совершается рекурсивный проход по всем плечам соединённых бинарных коромысел.

Описание алгоритма.

При считывании данных образуется иерархический список, состоящий из бинарных коромысел, плеч и грузов. Каждое коромысло имеет два указателя на левое и правое плечи.

Считывание данных происходит в строку, из которой после образуется иерархический список следующим образом. При нахождении в строке двух подряд идущих скобок, создаётся бинарное коромысло, при нахождении скобки и идущего за ней числа - плечо, длина которого равна этому числу. После этого создаётся груз этого плеча. Если за длиной плеча следует скобка, то в грузе создаётся новое бинарное коромысло, если же далее следует число, то в грузе сохраняется только его масса, а указатель на бинарное коромысло становится равен nullptr.

После сохранения иерархического списка, происходит нахождение длины всех плеч. Сначала рекурсивно находятся длины всех плеч, начиная с левого, путём сложения длины самого плеча и длин плеч других бинарных коромысел, если такие есть. Затем таким же образом находятся длины плеч, начиная с правого плеча первого коромысла.

При каждом вызове рекурсивных функций происходит вывод информации о вызове функции поиска длины всего коромысла или плеча. Глубина рекурсии видна с помощью табуляции, а номера позволяют ориентироваться, какое по счёту бинарное коромысло и какое из двух плеч каждого коромысла обрабатываются программой.

Тестирование.

№	Входные данные	Результат
1.	((1 ((2 3) (4 ((5 6) (7 8))))) (9 ((10 11) (12 13))))	Нахождение длины коромысла № 1 Нахождение длины плеча № 1.1 Нахождение длины коромысла № 2 Нахождение длины плеча № 2.1 Длина плеча № 2.1 найдена Нахождение длины плеча № 2.2 Нахождение длины коромысла № 3 Нахождение длины плеча № 3.1 Длина плеча № 3.1 найдена Нахождение длины плеча № 3.2 Длина плеча № 3.2 найдена Длина коромысла № 3 найдена Длина плеча № 2.2 найдена Длина коромысла № 2 найдена Длина плеча № 1.1 найдена Нахождение длины плеча № 1.2 Нахождение длины коромысла № 4 Нахождение длины плеча № 4.1 Длина плеча № 4.1 найдена Нахождение длины плеча № 4.2 Длина плеча № 4.2 найдена Длина коромысла № 4 найдена Длина плеча № 1.2 найдена Длина коромысла № 1 найдена 50
2.	((1((12 13) (14 15))) (2 3))	Нахождение длины коромысла № 1 Нахождение длины плеча № 1.1 Нахождение длины коромысла № 2 Нахождение длины плеча № 2.1 Длина плеча № 2.1 найдена Нахождение длины плеча № 2.2 Длина плеча № 2.2 найдена Длина коромысла № 2 найдена

		Длина плеча № 1.1 найдена Нахождение длины плеча № 1.2 Длина плеча № 1.2 найдена Длина коромысла № 1 найдена 29
3.	((1 2)(3 4))	Нахождение длины коромысла № 1 Нахождение длины плеча № 1.1 Длина плеча № 1.1 найдена Нахождение длины плеча № 1.2 Длина плеча № 1.2 найдена Длина коромысла № 1 найдена 4
4.	((1 2)(3((4 5)(6 7))))	Нахождение длины коромысла № 1 Нахождение длины плеча № 1.1 Длина плеча № 1.1 найдена Нахождение длины плеча № 1.2 Нахождение длины коромысла № 2 Нахождение длины плеча № 2.1 Длина плеча № 2.1 найдена Нахождение длины плеча № 2.2 Длина плеча № 2.2 найдена Длина коромысла № 2 найдена Длина плеча № 1.2 найдена Длина коромысла № 1 найдена 14
5.	((1((11 12)(13 14))(2((21 22)(23 24))))	Нахождение длины коромысла № 1 Нахождение длины плеча № 1.1 Нахождение длины коромысла № 2 Нахождение длины плеча № 2.1 Длина плеча № 2.1 найдена Нахождение длины плеча № 2.2 Длина плеча № 2.2 найдена Длина коромысла № 2 найдена Длина плеча № 1.1 найдена Нахождение длины плеча № 1.2 Нахождение длины коромысла № 3 Нахождение длины плеча № 3.1 Длина плеча № 3.1 найдена Нахождение длины плеча № 3.2 Длина плеча № 3.2 найдена

		Длина коромысла № 3 найдена Длина плеча № 1.2 найдена Длина коромысла № 1 найдена 71

Вывод.

Были изучены основные понятия иерархического списка и получен опыт работы с ним на языке программирования C++. Разработана программа, использующая иерархический список.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

```
#include <iostream>
#include <string>
#include <cmath>
#include <fstream>
#include <cctype>
using namespace std;

struct cargo;
struct shoulder;
struct binKor;
int global_tab = 0;

binKor* new_bk(string& str);
shoulder* new_pl(string& str);
cargo* new_mass(string& str);
void clean_memory(binKor* bk);
void clean_memory(shoulder* pl);
void clean_memory(cargo* mass);
short length(const binKor* bk);
short length(const shoulder* pl, int num, int tab);
short length(const cargo* mass);

struct cargo{
    bool tag; // =false, если груз является гирькой;
=true, если груз является коромыслом
    int m;   //масса гирьки
    binKor *bk; //указатель на бинарное коромысло
(=nullptr, если груз - гирька)
};

struct shoulder{
    int len; //длина плеча
    cargo* mass; //указатель на груз
};

struct binKor{
    shoulder *pl1; //указатели на первое и
    shoulder *pl2; // второе плечи коромысла
};
```

```
//Считывание и сохранение
binKor* new_bk(string& str) {
    binKor* bk = new binKor;
    if(str[0] == '(' && str[1] == '(') { //При нахождении
двух подряд идущих скобок
        str.erase(0, 2);
        bk->pl1 = new_pl(str);         //создаётся первое
плечо
    }else{
        cout << "ERROR\n";
        exit(0);
    }
    if(str[0] == '(' && str[1] != '('){ //При нахождении
одной скобки
        str.erase(0, 1);
        bk->pl2 = new_pl(str);        //создаётся второе
плечо
    }else{
        cout << "ERROR\n";
        exit(0);
    }
    return bk;
}

shoulder* new_pl(string& str){
    shoulder* pl = new shoulder;
    pl->len = atoi(str.c_str()); //сохранение длины плеча
    if(isdigit(str[0]))
        str.erase(0, log10(pl->len) + 1); //удаление из
строки считанной длины
    else{
        cout << "ERROR\n";
        exit(0);
    }
    while(str[0] == ' ')
        str.erase(0, 1); //удаление пробелов
    pl->mass = new_mass(str);
    return pl;
}

cargo* new_mass(string& str){
    cargo* mass = new cargo;
```

```
    if(str[0] == '(') {      //Если встречается скобка,
груз становится бинарным коромыслом
        mass->tag = true;
        mass->bk = new_bk(str);
    }else if(isdigit(str[0])){ //Если находится число, оно
сохраняется в качестве массы гирьки
        mass->tag = false;
        mass->m = atoi(str.c_str());
        str.erase(0, log10(mass->m)+1); //Удаление из
строки считанной массы
    }else{
        cout << "ERROR\n";
        exit(0);
    }
    while(str[0] == ')' || str[0] == ' ')
        str.erase(0,1);
    return mass;
}

//Очистка памяти
void clean_memory(binKor* bk){
    clean_memory(bk->pl1);
    clean_memory(bk->pl2);
    delete bk;
}

void clean_memory(shoulder* pl){
    clean_memory(pl->mass);
    delete pl;
}

void clean_memory(cargo* mass){
    if(mass->tag)
        clean_memory(mass->bk);
    delete mass;
}

//Нахождение суммарной длины всех плеч бинарных коромысел
short length(const binKor* bk){
    int tab = global_tab++;
    for(int i = 0; i<tab; i++)
        cout << "\t";
    tab++;
```

```
    cout << "Нахождение длины коромысла № " << tab << "\
n";
    short len = length(bk->pl1, 1, tab);  //Нахождение
суммы длин первого и
    len+=length(bk->pl2, 2, tab);     //второго плеч
    for(int i = 0; i<tab-1; i++)
        cout << "\t";
    cout << "Длина коромысла № " << tab << " найдена\n";
    return len;
}

short length(const shoulder* pl, int num, int tab){
    for(int i = 0; i<tab; i++)
        cout << "\t";
    cout << "Нахождение длины плеча № " << tab << "." <<
num << "\n";
    short len = length(pl->mass) + pl->len;
    for(int i = 0; i<tab; i++)
        cout << "\t";
    cout << "Длина плеча № " << tab << "." << num << "
найдена\n";
    return len;
}

short length(const cargo* mass){
    short len = 0;
    if(mass->tag) {
        len += length(mass->bk);
    }
    return len;
}

int main(){
    string str;
    int a = 0;
    cout << "Откуда будет производиться ввод? (0 -
консоль, 1 - файл)\n";
    cin >> a;
    if(a != 1 && a != 0){
        return 0;
    }
    if(a == 1){
        string filename;
```

```
        cout << "Введите название файла\n";
        cin >> filename;
        ifstream file(filename);
        if(!file.is_open()){
            cout << "Не удалось открыть файл\n";
            return 0;
        }
        getline(file, str);
        file.close();
    }else{
        cin.ignore();
        getline(cin, str);
    }
    if(str.empty()) {
        cout << "Вы не ввели никаких данных\n";
        return 0;
    }
    binKor* bk = new_bk(str);
    cout << length(bk) << '\n';
    clean_memory(bk);
    return 0;
}
```