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: Рекурсивная обработка иерархических списков

Студент гр. 9382 _____ Рыжих Р.В.

Преподаватель _____ Фирсов М.А.

Санкт-Петербург

2020

Цель работы.

Изучить рекурсивный способ работы с иерархическими списками в C++.

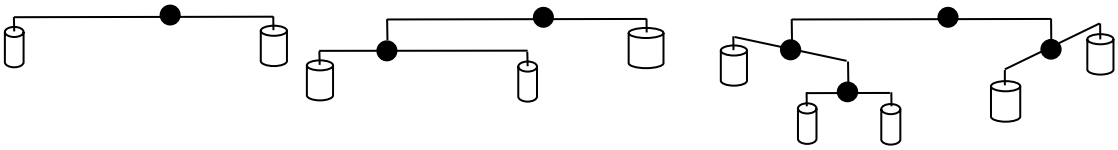
Основные теоретические положения.

Иерархический список – последовательность элементов, которые могут быть или элементами базового типа El, то есть атомами (атомарным S-выражением), или линейными списками из S-выражений. Списки заключаются в скобки, а элементы списка разделяются пробелами.

Задание.

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.

Можно (но не обязательно) представлять себе *бинарное коромысло*, чем-то похожим на конструкции, изображенные на рисунке.



а *б* *в*

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,

где в свою очередь Гирька есть натуральное число. Таким образом, БинКор есть специального вида иерархический список из натуральных чисел. Например, следующие списки представляют бинарные коромысла, изображенные выше на рисунке (вместо натуральных чисел здесь для общности и удобства восприятия представлены их обозначения с учетом места появления в списке):

а) $((l_1\ m_1)\ (l_2\ m_2))$;
б) $((l_1\ ((l_{11}\ m_{11})\ (l_{12}\ m_{12})))\ (l_2\ m_2))$;
в) $((l_1\ ((l_{11}\ m_{11})\ (l_{12}((l_{121}\ m_{121})\ (l_{122}\ m_{122}))))\ (l_2\ ((l_{21}\ m_{21})\ (l_{22}\ m_{22}))))$.

Для работы с бинарными коромыслами в таком представлении следует использовать базовые функции для работы с иерархическими списками.

1) Подсчитать общий вес заданного бинарного коромысла bk, т. е. суммарный вес его гирек. Для этого ввести рекурсивную функцию

unsigned int W (const БинКор bk).

Описание структур данных для реализации иерархических списков.

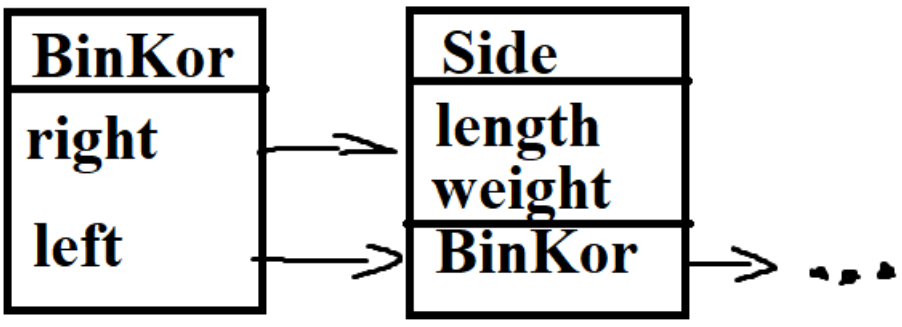
```
struct Side {
   int length;// Длинна плеча
   bool isWeight;// true:Груз false:БинКор
   union {
      int weight;//Масса груза
      BinKor* bin_kor;//БинКор
   } data;
};
struct BinKor {
   Side* left;// Левое плечо
   Side* right;//Правое плечо
};
```

Создано 2 структуры для реализации иерархических списков: struct Side и struct BinKor.

struct Side – структура, которая содержит в себе информацию о плече бинарного коромысла: длину плеча, булевую переменную isWeight, для проверки на то, является ли элемент, находящийся на плече, атомом или бинарным коромыслом, и union, который содержит в себе массу груза (если элемент – атом), или указатель на структуру BinKor (если элемент – бинарное коромысло).

struct BinKor – структура, которая содержит в себе 2 указателя: Side* left и Side* right. Это указатели на левое и правое плечо соответственно.

Графическая схема примера иерархического списка.



Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 — Результаты тестирования

No	Входные данные	Выходные данные	Комментарии
1			
2	1.) (a) 2.) (abc)	1.)You entered:(a) Wrong input ! 2.)You entered:(abc) Wrong input !	Если ввести букву или слово
3	(a b c)	You entered:(a b c) Wrong input !	Несколько букв
4	(2 2)	You entered:(2 2) Wrong input !	При вводе не бинарного коромысла
5	()	You entered:() Wrong input !	При вводе пустых скобок
6	((1 1) (1 1))	You entered:((1 1) (1 1)) right weight weights: 1 left weight weights: 1 right + left weights weight: 2 Weight of all weights: 2	Простейшее бинарное коромысло
7	((1 ((11 11) (12 12))) (2 2))	You entered:((1 ((11 11) (12 12))) (2 2)) Left side: (11 11) Right side: (12 12) Left side: (1 (...)) Right side: (2 2) right weight weights: 2 right weight weights: 12 left weight weights: 11 right + left weights weight: 23	

		right + left weights weight: 25 Weight of all weights: 25	
8	((1 ((11 11) (12 ((121 121) (122 122))))) (2 ((21 21) (22 22))))	You entered:((1 ((11 11) (12 ((121 121) (122 122))))) (2 ((21 21) (22 22)))) Left side: (11 11) Left side: (121 121) Right side: (122 122) Right side: (12 (...)) Left side: (1 (...)) Left side: (21 21) Right side: (22 22) Right side: (2 (...)) right weight weights: 22 left weight weights: 21 right + left weights weight: 43 right weight weights: 122 left weight weights: 121 right + left weights weight: 243 left weight weights: 11 right + left weights weight: 254 right + left weights weight: 297 Weight of all weights: 297	
9	((1 ((23 45) (67 ((891 234) (567 890))))) (1 ((23 4) (5 ((1 3) (9 19))))))	You entered:((1 ((23 45) (67 ((891 234) (567 890))))) (1 ((23 4) (5 ((1 3) (9 19)))))) Left side: (23 45) Left side: (891 234) Right side: (567 890) Right side: (67 (...)) Left side: (1 (...))	

		Left side: (23 4) Left side: (1 3) Right side: (9 19) Right side: (5 (...)) Right side: (1 (...)) right weight weights: 19 left weight weights: 3 right + left weights weight: 22 left weight weights: 4 right + left weights weight: 26 right weight weights: 890 left weight weights: 234 right + left weights weight: 1124 left weight weights: 45 right + left weights weight: 1169 right + left weights weight: 1195 Weight of all weights: 1195	

Выводы.

Были изучены и опробованы методы рекурсивной работы с иерархическими списками на языке C++. Была создана программа для реализации и работы с бинарным коромыслом как с иерархическим списком.

ПРИЛОЖЕНИЕ С КОДОМ

main.cpp :

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

struct BinKor;

//Плечо ::= (Длина Груз(Гирька | БинКор))
struct Side {
    int length;// Длинна плеча
    bool isWeight;// true:Груз false:БинКор
    union {
        int weight;//Масса груза
        BinKor* bin_kor;//БинКор
    } data;
};

//БинКор ::= (Плечо Плечо)
struct BinKor {
    Side* left;// Левое плечо
    Side* right;//Правое плечо
};

void rightPrint(int indent, int result) {
    for (int i = 0; i < indent; i++) {//выводим нужное количество отступов
        std::cout << ("\t");
    }
    std::cout << "right weight weights: " << result;//Выводим результат рекурсии
    std::cout << '\n' << '\n';
}

void leftPrint(int indent, int result) {
    for (int i = 0; i < indent; i++) {//выводим нужное количество отступов
        std::cout << ("\t");
    }
    std::cout << "left weight weights: " << result;//Выводим результат рекурсии
    std::cout << '\n' << '\n';
}

void rightLeftPrint(int indent, int result) {
    for (int i = 0; i < indent; i++) {//выводим нужное количество отступов
        std::cout << ("\t");
    }
    std::cout << "right + left weights weight: " << result;;//Выводим результат рекурсии
    std::cout << '\n' << '\n';
}

unsigned int W(BinKor* bin_kor, int indent) {//Функция для поиска длины
    unsigned int result = 0;

    if (bin_kor->right) {//Если есть правое плечо
        if (bin_kor->right->isWeight) {
            rightPrint(indent, bin_kor->right->data.weight);// Выводим результат
            result += bin_kor->right->data.weight;
        }
        if (!bin_kor->right->isWeight)//Если груз
```

```
            result += W(bin_kor->right->data.bin_kor, indent + 1);
    }
    if (bin_kor->left) {//Если есть левое плечо
        if (bin_kor->left->isWeight) {
            leftPrint(indent, bin_kor->left->data.weight);// Выводим результат
            result += bin_kor->left->data.weight;
        }
        if (!bin_kor->left->isWeight)//Если груз
        {
            result += W(bin_kor->left->data.bin_kor, indent + 1);
        }
        rightLeftPrint(indent, result);// Выводим результат
    }
    return result;//Возвращает результат
}


void drop(std::string& str, int n) {
    if (str.length() >= n) {//Если длинна больше или равно
        str = str.substr(n);//Отрезаем лишнии символы
    }
}

short readNum(string& str) {
    string number = "";

    while (isdigit(str[0])) {//Пока цифры сохраняем в строчку
        number += str[0];
        drop(str, 1);// Отрезаем не нужный символ
    }
    try {
        string i;
        i = stoi(number);
    }
    catch (const std::invalid_argument&) {
        std::cout << "Wrong input !" << '\n';
        exit(0);
    }
    return stoi(number);//Возвращаем число
}


// (Side Side)
Side* createNewSide(string& input, int n);
BinKor* createNewBinKor(string& input, int n) {
    BinKor* bin_kor = new BinKor;
    drop(input, 1);// Отрезаем ненужный символ
    bin_kor->left = createNewSide(input, n);
    for (int i = 0; i < n; i++)
        std::cout << '\t';
    std::cout << "Left side: ";
    if (bin_kor->left->isWeight)
        std::cout << '(' << bin_kor->left->length << ' ' << bin_kor->left->data.weight <<
')' << std::endl;
    else
        std::cout << '(' << bin_kor->left->length << " (...))" << std::endl;
    drop(input, 1);// Отрезаем ненужный символ
    bin_kor->right = createNewSide(input, n);
    for (int i = 0; i < n; i++)
        std::cout << '\t';
    std::cout << "Right side: ";
    if (bin_kor->right->isWeight)
        std::cout << '(' << bin_kor->right->length << ' ' << bin_kor->right->data.weight <<
')' << std::endl;
```

```
    else
        std::cout << '(' << bin_kor->right->length << " (...))" << std::endl;
    drop(input, 1);// Отрезаем ненужный символ
    return bin_kor;
}

Side* createNewSide(string& input, int n) {// Создаем сторону
    Side* side = new Side;
    drop(input, 1);// Отрезаем ненужный символ
    side->length = readNum(input);
    drop(input, 1);// Отрезаем ненужный символ
    side->isWeight = (input[0] != '(');
    if (side->isWeight) {
        side->data.weight = readNum(input);
    }
    else {
        side->data.bin_kor = createNewBinKor(input, n+1);
    }
    drop(input, 1);// Отрезаем не нужный символ
    return side;
}


void freeBinKor(BinKor* bin_kor) {
    if (bin_kor != nullptr) {// Если указывает на  BinKor
        if (bin_kor->right->isWeight == 0) {// Если не груз
            freeBinKor(bin_kor->right->data.bin_kor);// Вызываем рекурсивную функцию
            delete bin_kor->right;// Освобождаем сторону
        }
        else {
            delete bin_kor->right;// Освобождаем сторону
        }
        if (bin_kor->left->isWeight == 0) {// Если не груз
            freeBinKor(bin_kor->left->data.bin_kor);// Вызываем рекурсивную функцию
            delete bin_kor->left;// Освобождаем сторону
        }
        else {
            delete bin_kor->left;// Освобождаем сторону
        }
        delete bin_kor;// Освобождаем BinKor
    }
    else {
        std::cout << "delete" << '\n';
    }
}

int main(int argc, char** argv) {
    std::string input;
    if (argc == 2)
    {
        std::string file = argv[1];
        std::ifstream fin(file);// окрываем файл для чтения
        if (fin.is_open())
            getline(fin, input);
        else
        {
            std::cout << "Can`t open file" << std::endl;
            return 0;
        }
        fin.close();
    }
    else
    {
        getline(cin, input);
```

```
    }
    int indent = 0;// отступ
    if (input.length() == 0) {// Если файл  пуст
        std::cout << "Input is empty" << '\n';
        exit(0);
    }
    std::cout << '\n' << "You entered:" << input << '\n' << '\n';
    BinKor* bin_kor = new BinKor;// Создаем указатель на BinKor
    int n = 0;// отступы
    bin_kor = createNewBinKor(input, n);
    std::cout << std::endl;
    int res = W(bin_kor, indent);
    std::cout << '\n' << "Weight of all weights: " << res << '\n';
    freeBinKor(bin_kor);// Особождаем память
    return 0;
}
```