

**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: Рекурсия на примере бинарного коромысла.**

Студент гр.9382

\_\_\_\_\_

Балаева М.О

Преподаватель

\_\_\_\_\_

Фирсов М.А.

Санкт-Петербург

2020

## **Цель работы.**

Применить на практике знания о иерархических списках, написать программу на языке C++ с использованием рекурсивных функций для работы с бинарными коромыслами.

## **Задание 5.**

Говорят, что бинарное коромысло сбалансировано, если момент вращения, действующий на его левое плечо, равен моменту вращения, действующему на правое плечо (то есть длина левого стержня, умноженная на вес груза, свисающего с него, равна соответствующему произведению для правой стороны), и если все под коромысла, свисающие с его плеч, также сбалансированы.

Написать рекурсивную функцию или процедуру `Balanced`, которая проверяет заданное коромысло на сбалансированность и выдает (значение `true`, если коромысло сбалансировано, и `false` в противном случае).

## **Основные теоретические положения.**

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.

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

$$\text{БинКор} ::= (\text{Плечо Плечо}),$$

где первое плечо является левым, а второе – правым. В свою очередь Плечо будет представляться списком из двух элементов

$$\text{Плечо} ::= (\text{ДлинаГруз}),$$

где Длина есть натуральное число, а Груз представляется вариантами

$$\text{Груз} ::= \text{Гирька} \mid \text{БинКор},$$

где в свою очередь Гирька есть натуральное число. Таким образом, БинКор есть специального вида иерархический список из натуральных чисел.

## **Ход работы.**

### ***Файл BinKor.h.***

Здесь реализовано три структуры (Weight, BinKor и Edge). Каждая структура хранит информацию о себе.

Weight – является ли он бинарным коромыслом, если является, то хранит ссылку на объект структуры BinKor, если нет, то является грузиком, соответственно хранит вес.

Edge – длину плеча и ссылку на объект класса Weight.

BinKor – ссылку на два объекта класса Edge, являющиеся правым и левым плечом бинарного коромысла.

Также здесь хранится объявление всех функций, которые затрагивают в своей реализации бинарное коромысло (Parser, BinKorParse, EdgeParse, Value, WeightParser, Balanced, WeightFunc).

### **Файл *BinKor.cpp*.**

**Функция *Parser*** – функция, начинающая обработку строки. Вызывает функцию для определения того, является ли строка бинарным коромыслом.

**Функция *BinKorParse*** – функция, которая проверяет подходит ли поданная строка/часть строки под определение бинарного коромысла. Вызывает функции обработки строки для определения правильности введения левого и правого плеча. Возвращает истину, если строка была введена верно и ложь, если нет.

**Функция *EdgeParse*** – функция, которая проверяет, является ли часть строки плечом бинарного коромысла. Так же получает значение длины плеча и вызывает функцию для определения грузика.

**Функция *WeightParse*** – функция, проверяет является ли часть строки еще одним узлом бинарного коромысла или грузиком. Если часть строки – грузик, то получает его вес (натуральное значение). Если часть строки – бинарное коромысло, то рекурсивно вызывает функцию *BinKorParse*.

Таким образом функции *BinKorParse*, *EdgeParse* и *WeightParse* являются рекуррентными и последовательно вызывают друг друга при обработке строки. Также в данных функциях забиваются значения объектов структур, объявленных в заголовочном файле.

**Функция *Value*** – функция, которая проверяет является ли часть строки значением, если да – то возвращает это значение.

**Функция *Balanced*** – функция, которая рекурсивным вызовом самой себя, а также функции *WeightFunc* определяет является ли бинарное коромысло сбалансированным. Содержит переменные, которые хранят информацию о том, сбалансированы ли все вложенные коромысла, а также вес, действующий на плечо коромысла.

**Функция *WeightFunc*** – функция, которая рекурсивным вызовом самой себя определяет вес действующий на текущий узел бинарного коромысла.

### Файл *Source.cpp*.

**Функция Main** –внутри этой функции происходит открытие файла с входной строкой, вызов функции для обработки входной строки и вызова *Balanced* же происходит вывод информации о работе программы (правильная ли строка была подана на вход, является ли заданное коромысло сбалансированным).

### Тестирование:В

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные
1.	((5 ((20 20) (20 ((5 10) (5 10)))))) (5 ((10 20) (10 20))))	Вход в <i>Balanced</i> Глубина: 0 Вход в левого потомка Глубина: 1 Вес левого сына(Глубина:2): 400 Вход в правого потомка Глубина: 2 Вес левого сына(Глубина:3): 50 Вес правого сына(Глубина:3): 50 Полный вес правого потомка(Глубина:2): 400 Полный вес левого потомка(Глубина:1): 200 Вход в правого потомка Глубина: 1 Вес левого сына(Глубина:2): 200 Вес правого сына(Глубина:2): 200 Полный вес правого потомка(Глубина:1): 200 Коромысло сбалансировано.
2.	8-800-555-35-35	Вход в <i>BinKorParse</i> : curPos = 0 Это не бинарное

		коромысло
3.	((5 1) (5 2))	<p>Вход в BinKorParse: curPos = 0 Это бинарное коромысло.</p> <p>Вход в Balanced Глубина: 0 Вес левого сына(Глубина:1): 5 Вес правого сына(Глубина:1): 5 Коромысло не сбалансировано.</p>
4.	fghjk	<p>Вход в BinKorParse: curPos = 0 Это не бинарное коромысло</p>
5.	Пустая строка	<p>Вход в BinKorParse: curPos = 0 Это не бинарное коромысло</p>
6.	((6 1) (6 1))	<p>Вход в BinKorParse: curPos = 0 Это бинарное коромысло.</p> <p>Вход в Balanced Глубина: 0 Вес левого сына(Глубина:1): 6 Вес правого сына(Глубина:1): 6 Коромысло сбалансировано.</p>

## Выводы.

Применены на практике знания о иерархических списках, написана программа на языке C++ с использованием рекурсивных функций для работы с бинарными коромыслами.

## ПРИЛОЖЕНИЕ А

### КОД ПРОГРАММЫ

#### Source.cpp

```
#include <iostream>
#include <stdio.h>
#include "src/BinKor.h"

using namespace std;

int main()
{
    setlocale(0, "");
    ifstream infile;
    string inputString;
    while(true){
        system("clear");
        cout << "Выберите опцию:" << endl;
        cout << "1. Консоль" << endl;
        cout << "2. Файл" << endl;
        cout << "3. Выход" << endl;
        string a;
        getline(cin, a);
        system("clear");
        if(a == "1"){
            cout << "Введите бинарное коромысло:" << endl;
            getline(cin, inputString);
        }
        else if(a == "2"){
            cout << "Введите название файла: ";
            string file;
            getline(cin, file);
            infile.open(file);
            if (!infile.is_open()){
                system("clear");
                cout << "Входной поток не открыт.\n";
                cout << "Press enter to continue ...";
                cin.get();
                continue;
            }
            getline(infile, inputString);
        }
        else if(a == "3"){
            cout << "До встречи" << endl;
            break;
        }
        else{
            cout << "Неверный ввод" << endl;
            cout << "Press enter to continue ...";
            cin.get();
            continue;
        }

        int curPosition = 0;
        BinKor* binKor = new BinKor();
```

```

system("clear");
cout << inputString << endl;
if (Parser(inputString, curPosition, binKor)) {
    cout << "Это бинарное коромысло.\n";
    if (Balanced(binKor, 0)) {
        cout << "Коромысло сбалансировано.\n";

        } else {
            cout << "Коромысло не сбалансировано.\n";

        }
    cout << "Press enter to continue ...";
    cin.get();
}
else
{
    cout << "Это не бинарное коромысло\n";
    cout << "Press enter to continue ...";
    cin.get();
}
delete binKor;
}
cout << "Press enter to continue ...";
cin.get();
return 0;
}

```

### **BinKor.cpp**

```

#include "BinKor.h"

using namespace std;
Weight::~Weight()
{
    delete binKor;
}

int WeightFunc(BinKor* binKor)
{
    int sumWeight = 0;
    if (!binKor->leftEdge->weight->isBinKor)
    {
        sumWeight+= binKor->leftEdge->weight->weight;
    }
    else
    {
        sumWeight += WeightFunc(binKor->leftEdge->weight->binKor);
    }

    if (!binKor->rightEdge->weight->isBinKor)
    {
        sumWeight += binKor->rightEdge->weight->weight;
    }
    else
    {
        sumWeight += WeightFunc(binKor->rightEdge->weight->binKor);
    }
}

```



```

    }

    return sumWeight;
}

bool Balanced(const BinKor* binKor, int count)
{
    if(count == 0) {
        cout << endl;
        cout << "Вход в Balanced" << endl;
    }
    cout << "Глубина: " << count << endl;
    bool balancedRight = true;
    bool balancedLeft = true;
    int torqueLeft = 0;
    int torqueRight = 0;
    if (!binKor->leftEdge->weight->isBinKor)
    {
        torqueLeft = binKor->leftEdge->length * binKor->leftEdge->weight-
>weight;
        cout << "Вес левого сына(Глубина:" << count+1 << "): " <<
torqueLeft << endl;
    }
    else
    {
        cout << "Вход в левого потомка" << endl;
        balancedLeft = Balanced(binKor->leftEdge->weight->binKor, 1+count);
        torqueLeft = binKor->leftEdge->length*WeightFunc(binKor->leftEdge-
>weight->binKor);
        cout << "Полный вес левого потомка(Глубина:" << count+1 << "): " <<
torqueLeft << endl;
    }
    if (!binKor->rightEdge->weight->isBinKor)
    {
        torqueRight = binKor->rightEdge->length * binKor->rightEdge->weight-
>weight;
        cout << "Вес правого сына(Глубина:" << count+1 << "): " <<
torqueLeft << endl;
    }
    else
    {
        cout << "Вход в правого потомка" << endl;
        balancedRight = Balanced(binKor->rightEdge->weight->binKor,
1+count);
        torqueRight = binKor->rightEdge->length * WeightFunc(binKor-
>rightEdge->weight->binKor);
        cout << "Полный вес правого потомка(Глубина:" << count+1 << "): "
<< torqueLeft << endl;
    }
    return torqueLeft == torqueRight && balancedLeft && balancedRight;
}

bool Parser(string inputString, int& curPosition, BinKor* binKor)
{
    if (BinKorParse(inputString, curPosition, binKor))
        return true;
    else
        return false;
}

```

```

}

bool BinKorParse(string inputString, int& curPosition, BinKor* binKor)
{
    cout << "Вход в BinKorParse: curPos = " << curPosition << endl;
    bool isBinkor = false;
    if (inputString[curPosition]== '(' && curPosition< inputString.length() -
2)
    {
        curPosition++;
        binKor->leftEdge = new Edge();
        if(EdgeParse(inputString, curPosition, binKor->leftEdge))
        {
            if (inputString[curPosition] == ' ' && curPosition <
inputString.length() - 2)
            {
                curPosition++;
                binKor->rightEdge = new Edge();
                if (EdgeParse(inputString, curPosition, binKor->rightEdge)
&& inputString[curPosition] == ')') && curPosition < inputString.length())
                {
                    curPosition++;
                    isBinkor = true;
                }
            }
        }
    }
    return isBinkor;
}

bool EdgeParse(string inputString, int& curPosition, Edge* edge)
{
    bool isEdge = false;
    string stringValue;
    if (inputString[curPosition] == '(' && curPosition <
inputString.length() - 2)
    {
        curPosition++;
        while (curPosition < inputString.length() - 1 &&
inputString[curPosition] != ' ')
        {
            stringValue += inputString[curPosition]; curPosition++;
        }
        int val = Value(stringValue);
        if (val <= 0)
            return false;
        else edge->length = val;

        if (inputString[curPosition] == ' ' && curPosition <
inputString.length() - 2)
        {
            curPosition++;
            edge->weight = new Weight();
            if(WeightParser(inputString, curPosition,edge->weight))
            {
                if(inputString[curPosition]== ')')&& curPosition<
inputString.length() -1)

```

```

        {
            curPosition++;
            isEdge = true;
        }
    }
}
return isEdge;
}

bool WeightParser(string inputString, int& curPosition, Weight* weight)
{
    bool isWeight = false;
    if (inputString[curPosition] == '(')
    {
        weight->isBinKor = true;
        weight->binKor = new BinKor();
        isWeight = BinKorParse(inputString, curPosition, weight->binKor);
    }
    else if (curPosition < inputString.length() - 2)
    {
        string stringValue;
        while (curPosition < inputString.length() - 1 &&
inputString[curPosition] != ')')
        {
            stringValue += inputString[curPosition];
            curPosition++;
        }
        int val = Value(stringValue);
        if (val >= 1)
        {
            isWeight = true;
            weight->weight = val;
            weight->isBinKor = false;
        }
    }
    return isWeight;
}

int Value(string stringValue)
{
    int value;
    try
    {
        value = stoi(stringValue);
    }

    catch (const std::invalid_argument& error)
    {
        cerr << "Invalid argument: " << error.what() << '\n';
        return -1;
    }
    if (to_string(value).length() == stringValue.length())
    {
        return value;
    }
    else
    {

```

```

        return-1;
    }
}

```

## BinKor.h

```

#ifndef UNTITLED15_BINKOR_H
#define UNTITLED15_BINKOR_H

#endif //UNTITLED15_BINKOR_H

#ifndef BINKOR_H
#define BINKOR_H
#include "iostream"
#include "fstream"
#include "cctype"
#include "string"
#include "BinKor.h"

using namespace std;
struct BinKor;
struct Weight{
    BinKor* binKor = nullptr;
    bool isBinKor;
    int weight;
    Weight() = default;
    Weight(int weight)
    {
        isBinKor = false;
        this->weight = weight;
    }
    Weight(BinKor* binkor)
    {
        this->weight = 0;
        this->isBinKor = true;
        this->binKor = binkor;
    }
    ~Weight();
};

struct Edge
{
    int length;
    Weight* weight;

    Edge() = default;

    Edge(int length, Weight* weight)
    {
        this->length = length; this->weight = weight;
    }
    ~Edge()
    {
        delete weight;
    }
};

struct BinKor
{

```

```

Edge* leftEdge = nullptr; Edge* rightEdge = nullptr;

BinKor() = default;
BinKor(Edge* leftEdge, Edge* rightEdge)
{
    this->leftEdge = leftEdge; this->rightEdge = rightEdge;
}

~BinKor()
{
    delete leftEdge; delete rightEdge;}
};

bool Parser(string inputString, int& curPosition, BinKor* binkor); bool
BinKorParse(string inputString, int& curPosition, BinKor* binkor);
bool EdgeParse(string inputString, int& curPosition, Edge* edge); int
Value(string stingValue);
bool WeightParser(string inputString, int& curPosition, Weight* weight);
bool Balanced(const BinKor* binkor, int count); int WeightFunc(BinKor*
binKor);

#endif

```