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. 8381 Гречко В.Д.

Преподаватель Жангиров Т.Р.

Санкт-Петербург

Цель работы.

Ознакомиться с основными понятиями и приёмами рекурсивной обработки списков, изучить особенности реализации бинарного коромысла на языке программирования C++. Разработать программу, использующую бинарное коромысло и его рекурсивную обработку.

Задание.

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 подкоромысла, свисающие с его плеч, также сбалансированы.

Написать рекурсивную функцию или процедуру Balanced, которая проверяет заданное коромысло на сбалансированность (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.

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.

Выполнение работы.

Написание работы производилось на базе операционной системы Linux

Для реализации программы был разработан графический интерфейс с помощью встроенного в QtCreator UI-редактора. При запуске программы пользователю предоставляется две кнопки для выбора метода считывания данных: из вышестоящей консоли или выбранного им файла. Вывод результата работы программы происходит в отдельное информационное окно.

Для реализации бинарного коромысла были реализованы следующие структуры: BinRock — в ней хранятся указатели на правое и левое плечо — Side — хранит в себе длину стержня и указатель на груз — Plum — хранит в себе состояние: груз или еще одно коромысло и объединение: либо вес груза, либо ссылка на последующее коромысло.

Изначально происходит проверка на минимальную длину строки, после чего запускается функция parseString, которая принимает на вход строку и корень коромысла и создает бинарное коромысло на основе полученных данных. Если встречается случай создания бинарного коромысла, то вызывается функция push, которая выделяет под него память и заносит в него необходимые данные. Функция является рекурсивной, она делит исходную строку на две подстроки и затем запускается в них.

Далее происходит вызов функции Balanced, которая проверяет, является ли коромысло сбалансированным. Для этого высчитываются моменты для каждого из плеч, и в случае, когда к плечу крепится коромысло, она запускает рекурсивно саму себя и функцию для подсчета веса Weight.

Функция Weight является также рекурсивной и подсчитывает вес заданного коромысла. Если плечо хранит груз, то он просто прибавляется к общему, в противном случае, повторяется вызов Weight.

Оценка эффективности алгоритма.

Алгоритм, реализованный в программе, имеет линейную зависимость от длины строки, то есть сложность оценивается как O(n). Все функции осуществляют единственный проход в глубину для выполнения задачи.

Примеры работы программы.

Окно запуска

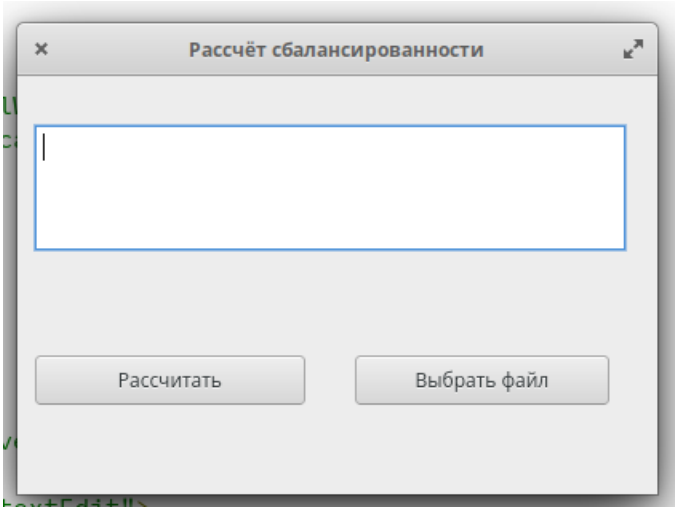

Таблица 1 – тестирование программы

Входные данные	Результат
((2 3) (3 2))	Balanced
((7 ((3 5) (4 6))) (1 7))	Not balanced
((1 ((2 7) (7 2))) (1 9))	Balanced
(13 45)	Incorrect input
((1 ((4 13) (13 4))) (1 ((10 7) (7 10))))	Balanced

Выводы.

В ходе выполнения лабораторной работы была изучена такая структура данных как бинарное коромысло, а также рекурсивные методы его обработки. Была реализована программа на C++, основанная на полученных знаниях.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

```
Название файла: main.cpp
#include "mainwindow.h"
#include <QApplication>
int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.setWindowTitle("Рассчёт сбалансированности");
    w.show();
    return a.exec();
}

Название файла: mainwindow.h
#include "mainwindow.h"
#include "ui_mainwindow.h"
MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    binrock = new (binary_rock);
}
MainWindow::~MainWindow()
{
    delete ui;
}
void MainWindow::on_choose_file_clicked()
{
    QString file_name = QFileDialog().getOpenFileName();
    if(!file_name.isNull()){
        QFile file(file_name);
        if(file.open(file.ReadOnly)){
            QString data  = file.readAll();
            if(data.toStdString().length() < 13 ){
                QMessageBox::warning(this,"Error",
"Incorrect input");
            }
            else{
                Side* side = new (Side);
                binrock->parseString(data.toStdString(),
*side);
                if
(binrock->Balanced(side->plum->uni.child))
                    QMessageBox::information(this, "Info",
"Balanced");
                else
```

```
                        QMessageBox::information(this, "Info",
"Not balanced");
                    delete side;
                }
            }
        }
        else QMessageBox::warning(this,"Error", "Not Found");
    }
    void MainWindow::on_onconsole_clicked()
    {
        QString data = ui->textEdit->toPlainText();
        if(data.toStdString().length() < 13 ){
            QMessageBox::warning(this,"Error",      "Incorrect
input");
        }
        else{
            Side* side = new (Side);
            binrock->parseString(data.toStdString(), *side);
            if (binrock->Balanced(side->plum->uni.child))
                QMessageBox::information(this,         "Info",
"Balanced");
            else
                QMessageBox::information(this,   "Info",   "Not
balanced");
            delete side;
        }
    }

    Названиe файла: binary_rock.h
    #ifndef BINARY_ROCK_H
    #define BINARY_ROCK_H
    #include <string>
    struct BinRock;
    struct Plum{
        bool isPlum;
        union{
          int weight;
          BinRock *child;
        } uni;
    };
    struct Side{
        int lenght;
        Plum *plum;
    };
    struct BinRock {
        Side *right;
        Side *left;
    };
    class binary_rock
    {
```

```
    private:
    public:
        binary_rock();
        int Weight(BinRock *);
        bool Balanced(BinRock *);
        void parseString (std::string, Side&);
        void push(Side &);
    };
    #endif // BINARY_ROCK_H

    Названиe файла: binary_rock.cpp
    #include "binary_rock.h"
    #include <QFileDialog>
    #include <QMessageBox>
    binary_rock::binary_rock()
    {
    }
    int binary_rock::Weight(BinRock *rock){
        int weig = 0;
        if(rock->right->plum->isPlum)          weig          +=
rock->right->plum->uni.weight;
        else Weight(rock->right->plum->uni.child);
        if(rock->left->plum->isPlum)          weig          +=
rock->left->plum->uni.weight;
        else Weight(rock->left->plum->uni.child);
        return weig;
    }
    bool binary_rock::Balanced(BinRock *rock){
        bool stateRight =true , stateLeft = true;
        int momentR = 0;
        int momentL = 0;
        if(rock->right->plum->isPlum){
            momentR       =       rock->right->lenght       *
rock->right->plum->uni.weight;
        }
        else {
            stateRight                                      =
Balanced(rock->right->plum->uni.child);
            momentR       =       rock->right->lenght       *
Weight(rock->right->plum->uni.child);
        }
        if(rock->left->plum->isPlum){
            momentL       =       rock->left->lenght        *
rock->left->plum->uni.weight;
        }
        else {
            stateLeft = Balanced(rock->left->plum->uni.child);
            momentL       =       rock->left->lenght        *
Weight(rock->left->plum->uni.child);
        }
```

```
        return momentL == momentR && stateRight && stateLeft;
    }
    void binary_rock::push(Side &side){
        side.plum = new Plum;
        side.plum->isPlum = false;
        side.plum->uni.child = new BinRock;
        side.plum->uni.child->left = new Side;
        side.plum->uni.child->right = new Side;
    }
    void  binary_rock::parseString  (std::string  str,  Side
&side){
        int count_br = 0;
        int state = 0;
        for(unsigned int i = 1; i < str.length() - 1; i++){
            if (str[i] == '(') count_br++;
            else if (str[i] == ')') count_br--;
            else if (count_br == 0 && str[i] == ' '){
                std::string leftOne = str.substr(1, i - 1);
                std::string  rightOne  =  str.substr(i  +  1,
str.length() - i - 2);
                if (leftOne[0] == '(' && rightOne[0] == '(') {
                    push(side);
                    parseString(leftOne,
*(side.plum->uni.child->left));
                    parseString(rightOne,
*(side.plum->uni.child->right));
                }
                else if (rightOne[0] == '(') {
                    char* end;
                    double len = strtod(leftOne.c_str(), &end);
                    if(!len) state = 1;
                    side.lenght = (int)len;
                    parseString(rightOne, side);
                }
                else{
                    char* end;
                    double  len_s  =  strtod(leftOne.c_str(),
&end);
                    if(!len_s) state = 1;
                    side.lenght = (int)len_s;
                    double  len_p  =  strtod(rightOne.c_str(),
&end);
                    if(!len_p) state = 1;
                    side.plum = new Plum;
                    side.plum->isPlum = true;
                    side.plum->uni.weight = (int)len_p;
                }
            }
          }
        if(state){
```

```
            QMessageBox::warning(NULL,"Error",       "Incorect
input");
            exit(1);
       }
     }

     Название файла: mainwindow.ui
     <?xml version="1.0" encoding="UTF-8"?>
     <ui version="4.0">
      <class>MainWindow</class>
      <widget class="QMainWindow" name="MainWindow">
       <property name="geometry">
        <rect>
         <x>0</x>
         <y>0</y>
         <width>400</width>
         <height>262</height>
        </rect>
       </property>
       <property name="windowTitle">
        <string>MainWindow</string>
       </property>
       <widget class="QWidget" name="centralWidget">
        <widget class="QWidget" name="verticalLayoutWidget">
         <property name="geometry">
          <rect>
           <x>10</x>
           <y>30</y>
           <width>371</width>
           <height>80</height>
          </rect>
         </property>
         <layout class="QVBoxLayout" name="verticalLayout">
          <item>
           <widget class="QTextEdit" name="textEdit">
            <property name="html">
             <string>&lt;!DOCTYPE HTML PUBLIC &quot;-//W3C//DTD
HTML        4.0//EN&quot;        &quot;http://www.w3.org/TR/REC-
html40/strict.dtd&quot;&gt;
     &lt;html&gt;&lt;head&gt;&lt;meta name=&quot;qrichtext&quot;
content=&quot;1&quot;                                /&gt;&lt;style
type=&quot;text/css&quot;&gt;
     p, li { white-space: pre-wrap; }
     &lt;/style&gt;&lt;/head&gt;&lt;body   style=&quot;   font-
family:'Open   Sans';   font-size:9pt;   font-weight:400;   font-
style:normal;&quot;&gt;
     &lt;p style=&quot;-qt-paragraph-type:empty; margin-top:0px;
margin-bottom:0px;   margin-left:0px;   margin-right:0px;   -qt-
block-indent:0;                text-indent:0px;&quot;&gt;&lt;br
/&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>
```

```
      </property>
     </widget>
    </item>
   </layout>
  </widget>
  <widget class="QWidget" name="gridLayoutWidget">
   <property name="geometry">
    <rect>
     <x>10</x>
     <y>130</y>
     <width>171</width>
     <height>121</height>
    </rect>
   </property>
   <layout class="QGridLayout" name="gridLayout">
    <item row="0" column="0">
     <widget class="QPushButton" name="onconsole">
      <property name="text">
       <string>Рассчитать</string>
      </property>
     </widget>
    </item>
   </layout>
  </widget>
  <widget class="QWidget" name="gridLayoutWidget_2">
   <property name="geometry">
    <rect>
     <x>210</x>
     <y>130</y>
     <width>161</width>
     <height>121</height>
    </rect>
   </property>
   <layout class="QGridLayout" name="gridLayout_2">
    <item row="0" column="0">
     <widget class="QPushButton" name="choose_file">
      <property name="text">
       <string>Выбрать файл</string>
      </property>
     </widget>
    </item>
   </layout>
  </widget>
 </widget>
 <layoutdefault spacing="6" margin="11"/>
 <resources/>
 <connections/>
</ui>
```