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

ОТЧЕТ

по лабораторной работе №4

по дисциплине «Алгоритмы и структуры данных»

Тема: Деревья

Студент гр. 8381 _____ Облизанов А.Д.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2019

Цель работы.

Ознакомиться с основными характеристиками и особенностями такой структуры данных, как бинарное дерево, изучить особенности ее реализации на языке программирования C++. Разработать программу, использующую бинарное дерево для обработки формулы.

Задание.

Формулу можно представить в виде бинарного дерева («дерева-формулы») согласно следующим правилам:

- формула из одного терминала представляется деревом из одной вершины с этим терминалом;
- формула вида (f_1 *s* f_2) представляется деревом, в котором корень – это знак *s*, а левое и правое поддеревья – соответствующие представления формул f_1 и f_2.

Вид формулы в постфиксной форме:

< формула > ::= < терминал > | (< формула > < формула > < знак >)

< знак > ::= + | – | *

< терминал > ::= 0 | 1 | ... | 9 | *a* | *b* | ... | *z*

Требуется:

- преобразовать дерево-формулу *t*, заменяя в нем все поддеревья, соответствующие формулам ((f_1 * f_2) + (f_1 * f_3)) и ((f_1 * f_3) + (f_2 * f_3)), на поддеревья, соответствующие формулам (f_1 * (f_2 + f_3)) и ((f_1 + f_2) * f_3);
- с помощью построения дерева-формулы *t* преобразовать заданную формулу *f* из постфиксной формы (перечисление узлов в порядке ЛПК) в инфиксную.

Основные теоретические положения.

Арифметическое выражение с бинарными операциями можно представить в виде бинарного дерева. Пусть, например, дано арифметическое выражение в инфиксной записи: $(a + b) * c - d / (e + f * g)$. На рис. 1 представлено соответствующее ему бинарное дерево.

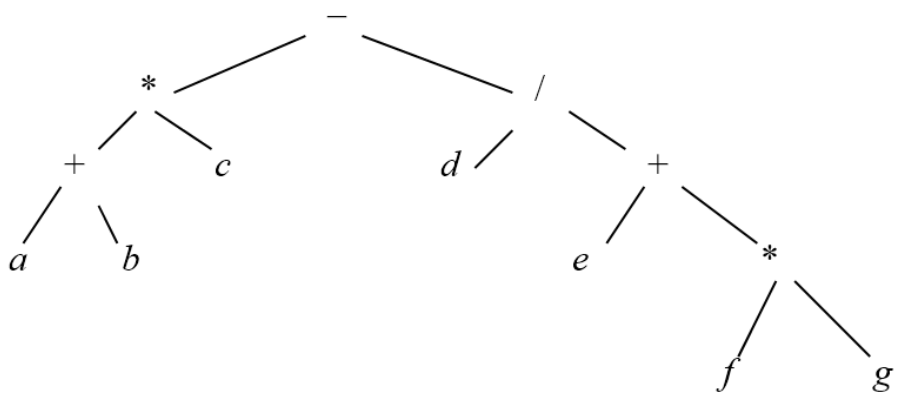

Рисунок 1 - Бинарное дерево, представляющее выражение

Тогда три варианта обхода этого дерева порождают три формы записи арифметического выражения:

1) КЛП – префиксную запись

$– * + a\ b\ c\ /\ d + e * f\ g$;

2) ЛКП – инфиксную запись (без скобок, необходимых для задания последовательности выполнения операций)

$a + b * c – d / e + f * g$;

3) ЛПК – постфиксную запись

$a\ b + c * d\ e\ f\ g * + / –$.

Выполнение работы.

Написание работы производилось на базе операционной системы Windows 10 в среде разработки QtCreator с использованием фреймворка Qt. Сборка, отладка производились в QtCreator, запуск программы осуществлялся через

командную строку. Исходные коды файлов программы представлены в приложениях А-М.

Для реализации программы был разработан графический интерфейс с помощью встроенного в QtCreator UI-редактора. Он представляет из себя поле ввода, кнопку считывания и переноса в поле ввода информации из файла, поле вывода с возможностью графического отображения результата. Основные слоты для работы графического интерфейса приведены в табл. 1.

Таблица 1 – Слоты класса MainWindow и их назначение

Метод	Назначение
`void on_outputBtn_toggled (bool checked)`	Слот, отвечающий за переключение графического и текстового выводов
`void on_openFile_clicked ()`	Слот, отвечающий за считывание из файла
`void on_saveFile_clicked ()`	Слот, отвечающий за запись из поля вывода в файл
`void on_start_clicked ()`	Слот, отвечающий за запуск алгоритма нахождения пар, за вывод информации об ошибках и результате

Для реализации бинарного дерева были созданы структуры узла BinNode и самого дерева BinTree, представленные на рис. 2.

```
struct BinNode
{
    bool isLeaf;
    char info;
    BinNode *left;
    BinNode *right;
};

struct BinTree
{
    BinNode *root;
    int deep;
};
```

Рисунок 2 – Структуры бинарного дерева и узла

Также были реализованы функции, создающие и изменяющие бинарное дерево, некоторые из них приведены в табл. 2.

Таблица 2 – Основные функции работы с бинарным деревом

Функция	Назначение
`BinTree *createBinTree()`	Создает пустое бинарное дерево
`BinNode *createBinNode (char info, bool isNum)`	Создает узел с заданными полями
`BinNode *appendRight(BinNode *node, char info, bool isNum)`	Создает узел с заданными полями, устанавливая его как правый к заданному узлу
`BinNode *appendLeft(BinNode *node, char info, bool isNum)`	Создает узел с заданными полями, устанавливая его как левый к заданному узлу
`BinNode *setInfo(BinNode *node, char info, bool isNum)`	Изменяет поля в заданном узле
`int countDeep(BinNode *&node)`	Возвращает количество уровней дерева
`BinTree *getTreeFromArray (QStringList in, int &err)`	Создает бинарное дерево из массива строк-элементов, полученного из входной строки
`QString getQStrFromTree (BinNode *root)`	Возвращает строку, представляющую дерево в инфиксной форме
`QString getListFromTree (BinNode *root, int indent)`	Возвращает строку, представляющую дерево в виде уступчатого списка
`QString modBinTree(BinNode *root)`	Преобразовывает дерево, заменяя в нем все поддеревья, соответствующие формулам $((f_1 * f_2) + (f_1 * f_3))$ и $((f_1 * f_3) + (f_2 * f_3))$, на поддеревья, соответствующие формулам $(f_1 * (f_2 + f_3))$ и $((f_1 + f_2) * f_3)$;

Программа имеет возможность графического отображения полученного бинарного дерева с помощью виджета QGraphicsView. Функции, необходимые для связи графического интерфейса и алгоритмов обработки дерева, а также для графического представления дерева, представлены в табл. 3.

Таблица 3 – Функции, связующие графический интерфейс и алгоритмы

Функция	Назначение
`QString startProcess(QString input, BinTree *&tree)`	Выполняет считывание и обработку данных, возвращая результат в виде строки
`QGraphicsScene *graphic(BinTree *tree, QGraphicsScene *&scene)`	По заданному бинарному дереву выполняет рисование в объекте QGraphicsScene
`BinNode *appendRight(BinNode *node, char info, bool isNum)`	Создает узел с заданными полями, устанавливая его как правый к заданному узлу
`int treePainter(QGraphicsScene *&scene, BinNode *node, int w, int h, int wDelta, int hDelta, QPen &pen, QBrush &brush, QFont &font, int depth)`	Рекурсивный алгоритм обхода дерева и рисования узлов в заданном объекте QGraphicsScene
`int connectUI(QTextEdit *&uiInput, QTextEdit *&uiOutput, QGraphicsScene *&scene)`	Главная связующая функция, выполняющая перенос данных из графического интерфейса в функции обработки для обработки и вывода

Оценка сложности алгоритма

Алгоритм создания бинарного дерева по строке является итеративным, каждый элемент строки обрабатывается один раз, а значит сложность алгоритма можно оценить как $O(N)$.

Алгоритм вывода дерева в инфиксной форме, либо же в виде уступчатого списка является рекурсивным, каждый узел дерева обрабатывается один раз, следовательно, сложность алгоритма также $O(N)$.

Алгоритм изменения дерева по формулам является рекурсивным и на каждом шаге в случае подозрения на соответствие формулы требует обхода поддеревьев текущего листа. Обход поддеревьев имеет сложность $O(N)$, глубина рекурсии также имеет сложность $O(N)$, что приводит к сложности алгоритма $O(N^2)$.

Тестирование программы.

Вид программы после выполнения представлен на рис. 3.

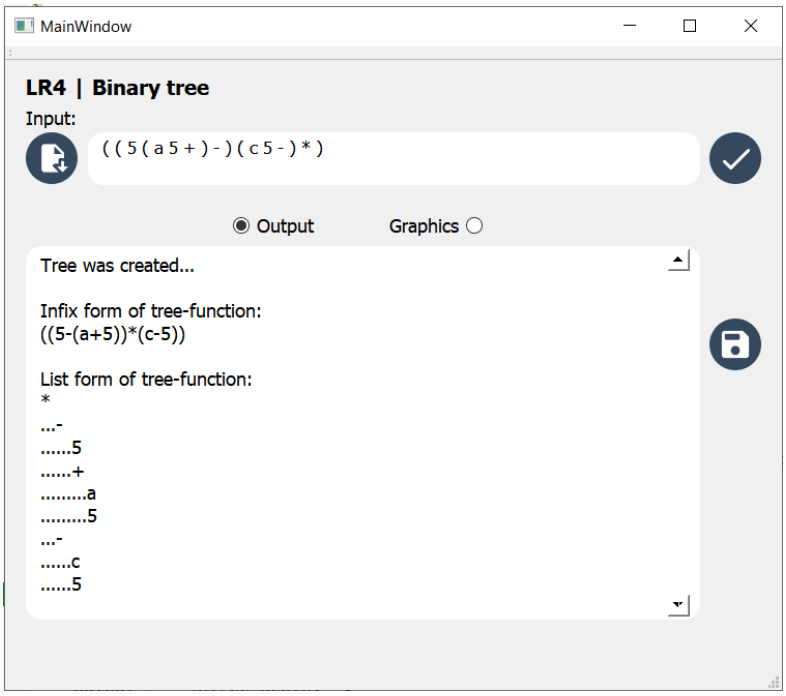

Рисунок 3 – Графический интерфейс программы

Графическое отображение бинарного дерева в программе представлено на рис. 4.

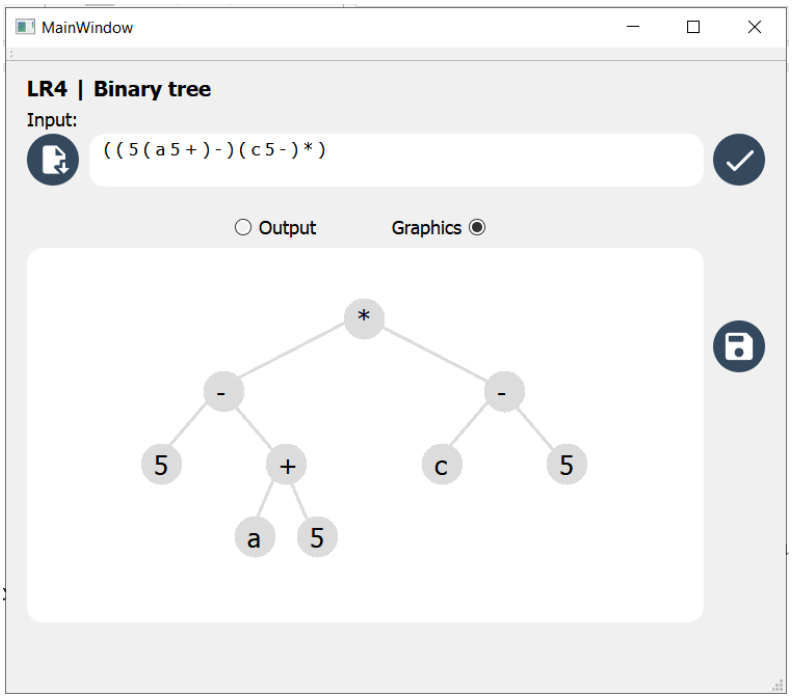

Рисунок 4 – Графическое отображение дерева

Был проведен ряд тестов, проверяющих корректность работы программы. Результаты тестирование приведены в табл. 4.

Таблица 4 – Тестирование программы

Входная строка	Вывод (в инфиксной записи)
((4 5 +) 5 5 -)	Error! Invalid postfix record.
((@ 5 +) 5 -)	Error! Invalid argument.
(p (((4 c -) 5 *) ((4 c -) 7 *) +) -)	(p-((4-c)*(5+7)))
((((4 6 +) (2 r -) *) ((2 5 /) (5 7 +) -) +) (((2 d *) (q 2 +) /) ((2 j -) (k l +) /) +) +)	((((4+6)*(2-r))+((2/5)-(5+7)))+(((2*d)/(q+2))+((2-j)/(k+l))))
(p ((((a b *) (a p *) +) 5 *) (((a b *) (a p *) +) 7 *) +) -)	(p-((a*(b+p))*(5+7)))
(p ((((3 3 -) 3 *) 3 /) 3 +) -)	(p-((((3-3)*3)/3)+3))

Выводы.

В ходе выполнения лабораторной работы была написана программа, создающая бинарное дерево согласно выражению в постфиксной форме, восстанавливающая выражение из дерева в инфиксной форме, изменяющая дерево согласно правилу вынесения общего множителя, представляющая дерево в виде уступчатого списка, а также графически.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ. MAIN.C

```
#include "mainwindow.h"
#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();

    return a.exec();
}
```

ПРИЛОЖЕНИЕ Б

ИСХОДНЫЙ КОД ПРОГРАММЫ. MAINWINDOW.H

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include "actions.h"

namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

private slots:
    void on_start_clicked();

    void on_saveFile_clicked();

    void on_openFile_clicked();

    void on_outputBtn_toggled(bool checked);

private:
    Ui::MainWindow *ui;
    QGraphicsScene *scene;
};

#endif // MAINWINDOW_H
```

ПРИЛОЖЕНИЕ В

ИСХОДНЫЙ КОД ПРОГРАММЫ. MAINWINDOW.CPP

```
#include "mainwindow.h"
#include "ui_mainwindow.h"

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    ui->outputBtn->setChecked(true);
    ui->graphicsView->hide();
    scene = new QGraphicsScene;
    ui->graphicsView->setScene(scene);
}

MainWindow::~MainWindow()
{
    delete ui;
}

void MainWindow::on_start_clicked()
{
    connectUI(ui->input, ui->output, scene);
}

void MainWindow::on_saveFile_clicked()
{
    QString filePath = QFileDialog::getSaveFileName(this, tr("Save to TXT file"),
QDir::homePath(), tr("*.txt"));
    if (QString::compare(filePath, QString()) != 0)
    {
        saveStrToFile(ui->output->toPlainText(), filePath);
    }
}

void MainWindow::on_openFile_clicked()
{
    QString fileName = QFileDialog::getOpenFileName(this, tr("Choose input file
(TXT)"), QDir::homePath(), tr("*.txt"));
    if (QString::compare(fileName, QString()) != 0)
    {
        ui->input->setPlainText(getStrFromFile(fileName));
    }
}
```

```
void MainWindow::on_outputBtn_toggled(bool checked)
{
    if (checked)
    {
        ui->graphicsView->hide();
        ui->output->show();
    }
    else
    {
        ui->output->hide();
        ui->graphicsView->show();
    }
}
```

ПРИЛОЖЕНИЕ Г

ИСХОДНЫЙ КОД ПРОГРАММЫ. INOUT.H

```
#include "bintree.h"

QString getStrFromFile(QString fileName);

int saveStrToFile(QString output, QString fileName);

QStringList getArrayFromStr(QString input);
```

ПРИЛОЖЕНИЕ Д

ИСХОДНЫЙ КОД ПРОГРАММЫ. INOUT.CPP

```
#include "inout.h"

QString getStrFromFile(QString fileName)
{
    ifstream fin(qPrintable(fileName), ios::in);
    string out;
    getline(fin, out);
    fin.close();
    return QString::fromStdString(out);
}

int saveStrToFile(QString output, QString fileName)
{
    ofstream fout(qPrintable(fileName));
    fout << qPrintable(output);
    fout.close();
    return 0;
}

QStringList getArrayFromStr(QString input)
{
    return input.split(" ");
}
```

ПРИЛОЖЕНИЕ Е

ИСХОДНЫЙ КОД ПРОГРАММЫ. BINTREE.H

```
#ifndef BINTREE_H
#define BINTREE_H
#include <iostream>
#include <fstream>
#include <sstream>
#include <QMainWindow>
#include <QGraphicsItem>
#include <QGraphicsView>
#include <QGraphicsEffect>
#include <QFileDialog>
#include <QStandardPaths>
#include <QtGui>
#include <QLabel>
#include <QColorDialog>
#include <QInputDialog>
#include <QMainWindow>
#include <QPushButton>
#include <QMessageBox>
#include <QStringList>
#include <QTextEdit>
#include <stack>
#define EMPTY_STACK 1
#define FULL_STACK 2
#define INVAL_ARG 3
#define INVAL_POST 4

using namespace std;


struct BinNode
{
    bool isLeaf;
    char info;
    BinNode *left;
    BinNode *right;
};

struct BinTree
{
    BinNode *root;
    int deep;
};

BinTree *createBinTree();
```

```
BinNode *createBinNode(char info, bool isNum);

BinNode *appendRight(BinNode *node, char info, bool isNum);

BinNode *appendLeft(BinNode *node, char info, bool isNum);

BinNode *setInfo(BinNode *node, char info, bool isNum);

BinTree *getTreeFromArray(QStringList in, int &err);

QString getQStrFromTree(BinNode *root);

QString getListFromTree(BinNode *root, int indent);

int countDeep(BinNode *&node);

int updateDeep(BinTree *tree);

#endif // BINTREE_H
```

ПРИЛОЖЕНИЕ Ж

ИСХОДНЫЙ КОД ПРОГРАММЫ. BINTREE.CPP

```
#include "bintree.h"

BinTree *createBinTree()
{
    BinTree *tree = new BinTree;
    tree->root = nullptr;
    tree->deep = 0;
    return tree;
}

BinNode *createBinNode(char info, bool isNum)
{
    BinNode *node = new BinNode;
    node->info = info;
    node->isLeaf = isNum;
    node->left = nullptr;
    node->right = nullptr;
    return node;
}

BinNode *appendRight(BinNode *node, char info, bool isNum)
{
    node->right = createBinNode(info, isNum);
    return node->right;
}

BinNode *appendLeft(BinNode *node, char info, bool isNum)
{
    node->left = createBinNode(info, isNum);
    return node->left;
}

BinNode *setInfo(BinNode *node, char info, bool isNum)
{
    if (node == nullptr)
        return nullptr;
    node->info = info;
    node->isLeaf = isNum;
    return node;
}

int countDeep(BinNode *&node)
{
    if (node == nullptr)
```

```
        return 0;
    int cl = countDeep(node->left);
    int cr = countDeep(node->right);
    return 1 + ((cl>cr)?cl:cr);
}

int updateDeep(BinTree *tree)
{
    tree->deep = countDeep(tree->root);
    return tree->deep;
}

BinTree *getTreeFromArray(QStringList in, int &err)
{
    stack <BinNode *> BNStack;
    BinTree *tree = createBinTree();
    tree->root = createBinNode('\0', 1);
    BinNode *temp = tree->root;
    BNStack.push(temp);
    for (int i=0; i<in.length(); i++)
    {
        if (in[i] == "(")
        {
            BNStack.push(temp);
            temp = appendLeft(temp, '\0', 1);
        }
        else if (in[i] == ")")
        {
            if (BNStack.empty() || temp->info == '\0')
            {
                err = INVAL_POST;
                return tree;
            }
            if (temp == BNStack.top()->left)
            {
                temp = BNStack.top();
                temp = appendRight(temp, '\0', 1);
            }
            else
            {
                temp = BNStack.top();
                BNStack.pop();
            }
        }
        else if (in[i] == "*" || in[i] == "/" || in[i] == "-" || in[i] == "+")
        {
            setInfo(temp, qPrintable(in[i])[0], 0);
```

```
        }
        else
        {
            if (in[i].length() > 1 || ((in[i][0] < 'a' || in[i][0] > 'z') &&
(!in[i][0].isDigit())))
            {
                err = INVAL_ARG;
                return tree;
            }
            if (BNStack.empty())
            {
                err = INVAL_POST;
                return tree;
            }
            setInfo(temp, qPrintable(in[i])[0], 1);
            if (temp == BNStack.top()->left)
            {
                temp = BNStack.top();
                temp = appendRight(temp, '\0', 1);
            }
            else
            {
                temp = BNStack.top();
                BNStack.pop();
            }
        }
    }
    if (!BNStack.empty())
    {
        err = INVAL_POST;
        return tree;
    }
    updateDeep(tree);
    return tree;
}

QString getQStrFromTree(BinNode *root)
{
    if (root == nullptr)
        return "";
    QString output;
    if (root->left || root->right)
        output += "(";
    output += getQStrFromTree(root->left);
    output += root->info;
    output += getQStrFromTree(root->right);
    if (root->left || root->right)
```

```
        output += ")";
    return output;
}

QString getListFromTree(BinNode *root, int indent)
{
    if (root == nullptr)
        return "";
    QString output;
    output += QString(indent, '.');
    output += root->info;
    output += "\n";
    output += getListFromTree(root->left, indent+3);
    output += getListFromTree(root->right, indent+3);
    return output;
}
```

ПРИЛОЖЕНИЕ И

ИСХОДНЫЙ КОД ПРОГРАММЫ. ACTIONS.H

```
#ifndef ACTIONS_H
#define ACTIONS_H
#include "inout.h"
#include "bintree.h"

int connectUI(QTextEdit *&uiInput, QTextEdit *&uiOutput, QGraphicsScene *&scene);

QString startProcess(QString input, BinTree *&tree);

QString modBinTree(BinNode *root);

int treePainter(QGraphicsScene *&scene, BinNode *node, int w, int h, int wDelta,
int hDelta, QPen &pen, QBrush &brush, QFont &font, int depth);

QGraphicsScene *graphic(BinTree *tree, QGraphicsScene *&scene);

#endif // ACTIONS_H
```

ПРИЛОЖЕНИЕ К

ИСХОДНЫЙ КОД ПРОГРАММЫ. ACTIONS.CPP

```
#include "actions.h"

int connectUI(QTextEdit *&uiInput, QTextEdit *&uiOutput, QGraphicsScene *&scene)
{
    BinTree *tree = nullptr;
    QString output = startProcess(uiInput->toPlainText(), tree);
    uiOutput->setPlainText(output);
    if (!output.count("Error"))
        graphic(tree, scene);
    else
        scene->clear();
    return 0;
}

QString startProcess(QString input, BinTree *&tree)
{
    QString output;
    int err = 0;
    QStringList inArr = getArrayFromStr(input);
    tree = getTreeFromArray(inArr, err);
    switch (err)
    {
    case INVAL_POST:
        output += "Error!\nInvalid postfix record.\nMake sure it's: (function
function operation)\n";
        return output;
    case INVAL_ARG:
        output += "Error!\nInvalid argument.\nUse only digits and ()+-*/ divided
by one space\n";
        return output;
    }
    output += "Tree was created...\n\nInfix form of tree-function:\n";
    output += getQStrFromTree(tree->root);
    output += "\n\nList form of tree-function:\n";
    output += getListFromTree(tree->root, 0);
    output += "\nTree depth: ";
    output += QString::number(tree->deep);
    output += "\nModify started...\n";
    output += modBinTree(tree->root);
    output += "\n\nInfix form of tree-function:\n";
    output += getQStrFromTree(tree->root);
    output += "\n\nList form of tree-function:\n";
    output += getListFromTree(tree->root, 0);
    return output;
```

```
}

QString modBinTree(BinNode *root)
{
    QString output;
    if (root == nullptr)
        return "";
    if (root->info == '+' && root->left && root->right)
    {
        if (root->left->info == '*' && root->right->info == '*')
        {
            if             (!QString::compare(getQStrFromTree(root->left->left),
getQStrFromTree(root->right->left)))
            {
                output += "\nBefore: ";
                output += getQStrFromTree(root);
                root->info = '*';
                root->right->left = root->left->right;
                root->left = root->left->left;
                root->right->info = '+';
                output += "\nAfter: ";
                output += getQStrFromTree(root);
                output += "\n";
            }
            else    if    (!QString::compare(getQStrFromTree(root->left->right),
getQStrFromTree(root->right->right)))
            {
                output += "\nBefore: ";
                output += getQStrFromTree(root);
                root->info = '*';
                root->right->right = root->left->left;
                root->left = root->left->right;
                root->right->info = '+';
                output += "\nAfter: ";
                output += getQStrFromTree(root);
                output += "\n";
            }
        }
    }
    output += modBinTree(root->left);
    output += modBinTree(root->right);
    return output;
}

QGraphicsScene *graphic(BinTree *tree, QGraphicsScene *&scene)
{
    if (tree == nullptr)
```

```
        return scene;
    scene->clear();
    QPen pen;
    QColor color;
    color.setRgb(220, 220, 220);
    pen.setColor(color);
    QBrush brush (color);
    QFont font;
    font.setFamily("Tahoma");
    pen.setWidth(3);
    int wDeep = static_cast<int>(pow(2, tree->deep)+2);
    int hDelta = 70;
    int wDelta = 15;
    font.setPointSize(wDelta);
    int width = (wDelta*wDeep)/2;
    treePainter(scene, tree->root, width/2, hDelta, wDelta, hDelta, pen, brush,
font, wDeep);
    return scene;
}

int treePainter(QGraphicsScene *&scene, BinNode *node, int w, int h, int wDelta,
int hDelta, QPen &pen, QBrush &brush, QFont &font, int depth)
{
    if (node == nullptr)
        return 0;
    QString out;
    out += node->info;
    QGraphicsTextItem *textItem = new QGraphicsTextItem;
    textItem->setPos(w, h);
    textItem->setPlainText(out);
    textItem->setFont(font);
    scene->addEllipse(w-wDelta/2, h, wDelta*5/2, wDelta*5/2, pen, brush);
    if (node->left != nullptr)
        scene->addLine(w+wDelta/2,    h+wDelta,    w-(depth/2)*wDelta+wDelta/2,
h+hDelta+wDelta, pen);
    if (node->right != nullptr)
        scene->addLine(w+wDelta/2,    h+wDelta,    w+(depth/2)*wDelta+wDelta/2,
h+hDelta+wDelta, pen);
    scene->addItem(textItem);
    treePainter(scene, node->left, w-(depth/2)*wDelta, h+hDelta, wDelta, hDelta,
pen, brush, font, depth/2);
    treePainter(scene, node->right, w+(depth/2)*wDelta, h+hDelta, wDelta, hDelta,
pen, brush, font, depth/2);
    return 0;
}
```

ПРИЛОЖЕНИЕ Л

ИСХОДНЫЙ КОД ПРОГРАММЫ. LR4.PRO

```
#-------------------------------------------------
#
# Project created by QtCreator 2019-10-31T13:25:54
#
#-------------------------------------------------

QT       += core gui

greaterThan(QT_MAJOR_VERSION, 4): QT += widgets

TARGET = LR4
TEMPLATE = app

# The following define makes your compiler emit warnings if you use
# any feature of Qt which has been marked as deprecated (the exact warnings
# depend on your compiler). Please consult the documentation of the
# deprecated API in order to know how to port your code away from it.
DEFINES += QT_DEPRECATED_WARNINGS

# You can also make your code fail to compile if you use deprecated APIs.
# In order to do so, uncomment the following line.
# You can also select to disable deprecated APIs only up to a certain version of
Qt.
#DEFINES += QT_DISABLE_DEPRECATED_BEFORE=0x060000     # disables all the APIs
deprecated before Qt 6.0.0

CONFIG += c++11

SOURCES += \
        main.cpp \
        mainwindow.cpp \
    inout.cpp \
    bintree.cpp \
    actions.cpp

HEADERS += \
        mainwindow.h \
    inout.h \
    bintree.h \
    actions.h

FORMS += \
        mainwindow.ui
```

```
# Default rules for deployment.
qnx: target.path = /tmp/$${TARGET}/bin
else: unix:!android: target.path = /opt/$${TARGET}/bin
!isEmpty(target.path): INSTALLS += target

RESOURCES += \
    images.qrc
```

ПРИЛОЖЕНИЕ М

ИСХОДНЫЙ КОД ПРОГРАММЫ. 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>750</width>
    <height>623</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>MainWindow</string>
  </property>
  <widget class="QWidget" name="centralWidget">
   <widget class="QTextEdit" name="input">
    <property name="geometry">
     <rect>
      <x>80</x>
      <y>70</y>
      <width>591</width>
      <height>51</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">background-color: white;
border-radius: 15px;
font: 10pt &quot;Tahoma&quot;;
padding: 0px 10px 0px 10px;</string>
    </property>
    <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:'Tahoma'; font-
size:10pt; 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>
```

```
   <widget class="QPushButton" name="openFile">
    <property name="geometry">
     <rect>
      <x>20</x>
      <y>70</y>
      <width>50</width>
      <height>50</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">border-image: url(:/open.png);</string>
    </property>
    <property name="text">
     <string/>
    </property>
   </widget>
   <widget class="QLabel" name="name">
    <property name="geometry">
     <rect>
      <x>20</x>
      <y>10</y>
      <width>251</width>
      <height>31</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">font: 12pt &quot;Tahoma&quot;;
font-weight: bold;</string>
    </property>
    <property name="text">
     <string>LR4 | Binary tree</string>
    </property>
   </widget>
   <widget class="QLabel" name="inputName">
    <property name="geometry">
     <rect>
      <x>20</x>
      <y>45</y>
      <width>161</width>
      <height>21</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">font: 11pt &quot;Tahoma&quot;;</string>
    </property>
    <property name="text">
     <string>Input:</string>
```

```
    </property>
   </widget>
   <widget class="QPushButton" name="start">
    <property name="geometry">
     <rect>
      <x>680</x>
      <y>70</y>
      <width>50</width>
      <height>50</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">border-image: url(:/ok.png);</string>
    </property>
    <property name="text">
     <string/>
    </property>
   </widget>
   <widget class="QPushButton" name="saveFile">
    <property name="geometry">
     <rect>
      <x>680</x>
      <y>250</y>
      <width>50</width>
      <height>50</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">border-image: url(:/savef.png);</string>
    </property>
    <property name="text">
     <string/>
    </property>
   </widget>
   <widget class="QTextEdit" name="output">
    <property name="geometry">
     <rect>
      <x>20</x>
      <y>180</y>
      <width>651</width>
      <height>361</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">background-color: white;
border-radius: 15px;
font: 11pt &amp;quot;Tahoma&amp;quot;;
```

```
padding: 3px 10px 3px 10px;</string>
    </property>
    <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:'Tahoma'; font-
size:11pt; 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>
   <widget class="QGraphicsView" name="graphicsView">
    <property name="geometry">
     <rect>
      <x>20</x>
      <y>180</y>
      <width>651</width>
      <height>361</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">background-color: white;
border-radius: 15px;
padding: 3px 10px 3px 10px;</string>
    </property>
   </widget>
   <widget class="QRadioButton" name="outputBtn">
    <property name="geometry">
     <rect>
      <x>220</x>
      <y>150</y>
      <width>111</width>
      <height>20</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">font: 11pt &quot;Tahoma&quot;;</string>
    </property>
    <property name="text">
     <string>Output</string>
    </property>
   </widget>
   <widget class="QRadioButton" name="graphicsBtn">
```

```
    <property name="geometry">
     <rect>
      <x>340</x>
      <y>150</y>
      <width>121</width>
      <height>20</height>
     </rect>
    </property>
    <property name="layoutDirection">
     <enum>Qt::RightToLeft</enum>
    </property>
    <property name="styleSheet">
     <string notr="true">font: 11pt &quot;Tahoma&quot;;</string>
    </property>
    <property name="text">
     <string>Graphics</string>
    </property>
   </widget>
  </widget>
  <widget class="QMenuBar" name="menuBar">
   <property name="geometry">
    <rect>
     <x>0</x>
     <y>0</y>
     <width>750</width>
     <height>26</height>
    </rect>
   </property>
  </widget>
  <widget class="QToolBar" name="mainToolBar">
   <attribute name="toolBarArea">
    <enum>TopToolBarArea</enum>
   </attribute>
   <attribute name="toolBarBreak">
    <bool>false</bool>
   </attribute>
  </widget>
  <widget class="QStatusBar" name="statusBar"/>
 </widget>
 <layoutdefault spacing="6" margin="11"/>
 <resources/>
 <connections/>
</ui>
```