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

ОТЧЕТ

по лабораторной работе №3
по дисциплине «АиСД»
Тема: Деревья

Студент гр. 9384 _____ Звега А.Р.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2020

Цель работы.

Получить практические навыки работы с бинарным деревом и его созданием.

Задание. (Вариант 6 Д)

Задано бинарное дерево b типа BT с произвольным типом элементов. Используя очередь, напечатать все элементы дерева b по уровням: сначала − из корня дерева, затем (слева направо) − из узлов, сыновних по отношению к корню, затем (также слева направо) − из узлов, сыновних по отношению к этим узлам, и т. д.

Выполнение работы.

Был реализован шаблонный класс бинарного дерева BT через связный список. Он имеет 2 метода: getNode - возвращает значение узла, height - выводит высоту дерева.

Программа написана при помощи Qt. Был написан класс MainWindow который считывает и отрисовывает дерево. Объект startButton при клике на активирует слот **on_startButtom_clicked**(), в котором происходит считывания с lineEdit, затем запускаются функции отрисовки и считывания дерева.

PrintTree() функция вывода дерева на graphicsView (находится слева), PrintLine() функция вывода дерева печатая узлы по уровням в линию на Task (находится справа), так же выводит результат в виде текста (находится снизу).

Тестирования.

Результаты тестирования представлены в таблице 1.

Табл. 1

№	Входные данные	Вывод	Комментарий
1	(a(b)(c))	abc	Все верно.

2	(a(b)d)	Invalid tree	Дерево неправильно введено
3	(1(2(4)(5))(3(6)(7)))	1234567	Все верно
4	(a(b (a(b)(c)) (a(b)(c)))(c (a(b)(c))))	Invalid tree	Пропусченны скобки после (a(b (a(b)(c)) (a(b)(c)))(c (a(b)(c))

Выводы.

В ходе работы были получены навыки создания деревьев и работы с ними.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ.

Файл: main.cpp

```
#include "mainwindow.h"

#include <QApplication>

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

Файл: mainwindow.cpp

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




MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);
}

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


void MainWindow::on_startButtom_clicked()
{
    QString text = ui->lineEdit->text().replace(" ", "");
    if(text[0] != '('){
        QMessageBox::warning(this, "Input error", "Start tree '('");
        return;
    }
    int index = 0;
    if(readTree(text, Tree, index) && index == text.size()) {
        h = Tree->height();
```

```
        printTree(Tree, 0);
        printLine(Tree, 0);
        ui->textTask->setText(taskText.replace(" ", ""));
    }else{
        QMessageBox::warning(this, "Input error", "Invalid tree");
        return;
    }
    if(Tree) {
        delete Tree;
        Tree = nullptr;
    }
}

bool MainWindow::readTree(QString text, BT<char>*& node, int& index) {
    if(isList(text, index)) {
        if(isRoot(text, index))
            node = new BT<char>(text.toStdString()[index-1]);
        else if(isEmptyList(text, index))
            return true;
        else
            return false;

        if(isEmptyList(text,index))
            return true;

        return readTree(text, node->left, index) &&
               readTree(text, node->right, index) &&
               isEmptyList(text,index);

    }
    else return false;
}

bool MainWindow::isList(QString text, int& index) {
    if(text[index] == '(') {
        index++;
        return true;
    }
    return false;
}

bool MainWindow::isRoot(QString text, int& index) {
    if(index == 0) return false;
    if(text[index].isLetterOrNumber()) {
        index++;
        return true;
    }
    return false;
}
```

```
bool MainWindow::isEmptyList(QString text, int& index) {
   if(text[index] == ')') {
      if(index==1) return false;
      index++;
      return true;
   }
   return false;
}

void MainWindow::printTree(BT<char>*& Tree, int index){
   int x = 300*h;
   if(Tree){
      if(index == 0){
         scene = new QGraphicsScene(0,0,300*h,100*h);
         ui->graphicsView->setScene(scene);
         printTreeL(Tree->left, index+1, x/2);
         printTreeR(Tree->right, index+1, x/2);
         scene->addEllipse(x/2,50*(index+1),25,25,QColor(0,0,0),QColor(255,150,255));
         scene->addText(QString(Tree->returnData()))->setPos(x/2+5,50*(index+1));
      }
   }
}

void MainWindow::printTreeL(BT<char>*& Tree, int index, int offset){
   if(Tree){
      int x = offset - h*50/index;
      scene->addLine(x + 12.5,50*(index+1)+12.5,offset+12.5,50*index
+12.5,QPen(Qt::black,3));
      printTreeL(Tree->left, index+1, x);
      printTreeR(Tree->right, index+1, x);
      scene->addEllipse(x,50*(index+1),25,25,QColor(0,0,0),QColor(255,150,255));
      scene->addText(QString(Tree->returnData()))->setPos(x+5,50*(index+1));
   }
}

void MainWindow::printTreeR(BT<char>*& Tree, int index, int offset){
   if(Tree){
      int x = offset + h*50/index;
      scene->addLine(x + 12.5,50*(index+1)+12.5,offset+12.5,50*index
+12.5,QPen(Qt::black,3));
      printTreeL(Tree->left, index+1, x);
      printTreeR(Tree->right, index+1, x);
      scene->addEllipse(x,50*(index+1),25,25,QColor(0,0,0),QColor(255,150,255));
      scene->addText(QString(Tree->returnData()))->setPos(x+5,50*(index+1));
   }
}

void MainWindow::printLine(BT<char>*& Tree, int index){
   int x = ui->Task->width()-25;
   if(Tree){
```

```
        if(index == 0){
            sceneTask = new QGraphicsScene(0,0,ui->Task->width()-2,100*h);
            ui->Task->setScene(sceneTask);
            sceneTask->addEllipse(x/2,25*(index+1),25,25,QColor(0,0,0),QColor(255,150,255));
            sceneTask->addText(QString(Tree->returnData()))->setPos(x/2+5,25*(index+1));
            taskText[0] = Tree->returnData();
            printLineL(Tree->left,index+1,2);
            printLineR(Tree->right,index+1,3);
        }
    }
}
void MainWindow::printLineL(BT<char>*& Tree, int index, int offset){
    int x = ui->Task->width()-25;
    if(Tree){
            sceneTask->addEllipse(x/2,25*offset,25,25,QColor(0,0,0),QColor(255,150,255));
            sceneTask->addText(QString(Tree->returnData()))->setPos(x/2+5,25*offset);
            taskText[offset-1] = Tree->returnData();
            printLineL(Tree->left,index+1,offset*2);
            printLineR(Tree->right,index+1,offset*2+1);

    }
}

void MainWindow::printLineR(BT<char>*& Tree, int index, int offset){
    int x = ui->Task->width()-25;
    if(Tree){
            sceneTask->addEllipse(x/2,25*offset,25,25,QColor(0,0,0),QColor(255,150,255));
            sceneTask->addText(QString(Tree->returnData()))->setPos(x/2+5,25*offset);
            taskText[offset-1] = Tree->returnData();
            printLineL(Tree->left,index+1,offset*2);
            printLineR(Tree->right,index+1,offset*2+1);

    }
}
```

Файл: mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QGraphicsScene>
#include <QGraphicsItem>
#include <QMessageBox>
#include <QColor>
#include <QString>
#include "BT.h"

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
```

```
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
   Q_OBJECT

public:
   MainWindow(QWidget *parent = nullptr);
   ~MainWindow();
   BT<char> *Tree;
   QString taskText;
   int h;
   void printLine(BT<char>*&, int);
   void printLineL(BT<char>*&, int, int);
   void printLineR(BT<char>*&, int, int);
   void printTree(BT<char>*&, int);
   void printTreeL(BT<char>*&, int, int);
   void printTreeR(BT<char>*&, int, int);
   bool isList(QString, int&);
   bool isRoot(QString, int&);
   bool isEmptyList(QString, int&);
   bool readTree(QString, BT<char>*&, int&);

   QGraphicsScene *scene;
   QGraphicsScene *sceneTask;
private slots:
   void on_startButtom_clicked();

private:
   Ui::MainWindow *ui;
};
#endif // MAINWINDOW_H
```

Файл: BT.h

```
#ifndef BT_H
#define BT_H

template <class T>
class BT{
public:
   BT() {}

   BT(T data) {
      this->data = data;
   }

   ~BT() {
      if(left)
         delete left;
```

```
        if(right)
            delete right;
    }

    T returnData() {
        return data;
    }

    int height() {
        int left_h = 0;
        int right_h = 0;
        if(this->left)
            left_h = left->height();
        if(this->right)
            right_h = right->height();
        return left_h>right_h? left_h+1 : right_h+1;
    }
    BT* left = nullptr;
    BT* right = nullptr;
private:
    T data;
};

#endif // BT_H
```