**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: Бинарные деревья

|  |  |  |
| --- | --- | --- |
| Студент гр. 6304 |  | Ковынев М.В. |
| Преподаватель |  | Балтрашевич В.Э. |

Санкт-Петербург

2017

**Цель работы.**

Знакомство с моделью бинарного дерева. Построение алгоритмов по его реализации и обработке

**Постановка задачи.**

Задано бинарное дерево b типа ВТ с типом элементов Elem. Для введенной пользователем величины Е (var E: Elem) определить число вхождений элемента Е в дерево b.

**Ход работы**

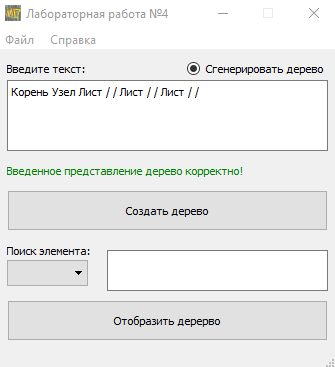
1. Создадим программу, реализующую заданную функции по поиску числа вхождений элемента в бинарное дерево.

Рисунок 1. Вид программы

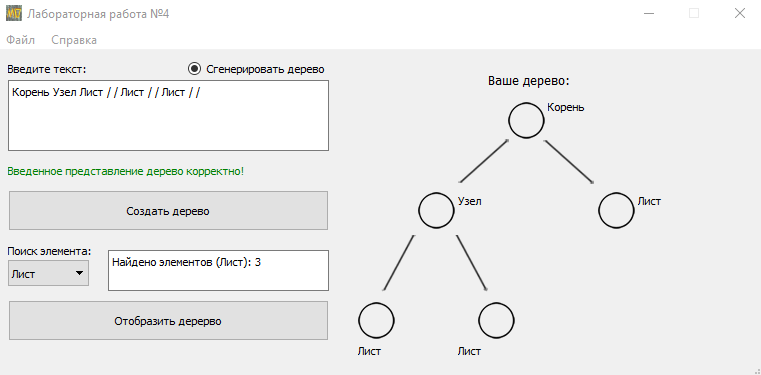
1. Для наглядности и представления бинарного дерева реализуем вывод полученного бинарного дерева в виде графа.

Рисунок 2. Вид программы с графом

1. Проведем тесты программы

|  |  |
| --- | --- |
| **Входные данные** | **Выходные данные** |
| ОдинЭлемент / / | Найдено элементов (ОдинЭлемент): 1 |
| Корень Узел Лист / / Лист / / Лист / / | Найдено элементов (Лист): 3 |
| Qt / is best / / program / / | Найдено элементов (best): 1 |
| Я Мама Бабушка / / Дедушка / / Папа Бабушка / / Дедушка / / | Найдено элементов (Бабушка): 2 |

# **Вывод**

В результате выполнения данной лабораторной работы были построены функции для работы с бинарным деревом, и создана программа, использующая реализованные функции. Также был создан графический интерфейс для более удобного её использования.

# **ПРИЛОЖЕНИЕ А.**

# **ИХОДНЫЙ КОД MAIN.CPP**

#include "mainwindow.h"

#include <QApplication>

#include <QGraphicsScene>

#include <QLabel>

int main(int argc, char \*argv[])

{

QApplication a(argc, argv);

MainWindow w;

w.setWindowTitle("Лабораторная работа №4");

w.setWindowIcon(QIcon("D:\\2.png"));

w.setFixedSize(340, 340);

w.show();

return a.exec();

}

# **ПРИЛОЖЕНИЕ Б.**

# **ИХОДНЫЙ КОД MAINWINDOW.CPP**

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <QPixmap>

#include <QWidget>

#include <QFile>

#include <QMessageBox>

#include <QFileDialog>

#include <QTextStream>

using namespace binTree\_modul;

binTree **enterBT**(QStringList string);

void **displayBT**(binTree b, int n);

void **levelBT**(binTree b);

void MainWindow::**clear\_tree**()

{

ui->label1\_1->*setVisible*(0);

ui->label1\_2->*setVisible*(0);

ui->label\_0->*setVisible*(0);

ui->label\_2->*setVisible*(0);

ui->label2\_1->*setVisible*(0);

ui->label2\_2->*setVisible*(0);

ui->label\_3->*setVisible*(0);

ui->label3\_1->*setVisible*(0);

ui->label3\_2->*setVisible*(0);

ui->label4\_1->*setVisible*(0);

ui->label4\_2->*setVisible*(0);

ui->label4\_3->*setVisible*(0);

ui->label4\_4->*setVisible*(0);

ui->label\_name\_1->*setVisible*(0);

ui->label\_name\_2->*setVisible*(0);

ui->label\_name\_3->*setVisible*(0);

ui->label\_name\_4->*setVisible*(0);

ui->label\_name\_5->*setVisible*(0);

ui->label\_name\_6->*setVisible*(0);

ui->label\_name\_7->*setVisible*(0);

ui->pushButton->setEnabled(0);

ui->pushButton\_2->setEnabled(0);

ui->textEdit\_2->setEnabled(0);

ui->comboBox->setEnabled(0);

ui->comboBox->clear();

window()->setFixedSize(340, 340);

}

MainWindow::**MainWindow**(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::MainWindow)

{

ui->setupUi(this);

ui->textEdit->setPlaceholderText("Пример: Я Мама Бабушка / / Дедушка / / Папа Бабушка / / Дедушка / / ");

QPixmap picture\_root("D:\\tree\_round.png");

QPixmap picture\_line\_1("D:\\tree\_line.png");

QPixmap picture\_line\_2("D:\\tree\_line\_2.png");

QPixmap picture\_line\_rr\_2("D:\\line\_rr\_1.png");

QPixmap picture\_line\_rr\_1("D:\\line\_rr\_2.png");

ui->label\_0->setPixmap(picture\_root);

ui->label1\_1->setPixmap(picture\_line\_1);

ui->label\_2->setPixmap(picture\_root);

ui->label1\_2->setPixmap(picture\_line\_2);

ui->label\_3->setPixmap(picture\_root);

ui->label2\_1->setPixmap(picture\_line\_rr\_1);

ui->label2\_2->setPixmap(picture\_line\_rr\_2);

ui->label3\_1->setPixmap(picture\_line\_rr\_1);

ui->label3\_2->setPixmap(picture\_line\_rr\_2);

ui->label4\_1->setPixmap(picture\_root);

ui->label4\_2->setPixmap(picture\_root);

ui->label4\_3->setPixmap(picture\_root);

ui->label4\_4->setPixmap(picture\_root);

clear\_tree();

}

MainWindow::~***MainWindow***()

{

delete ui;

}

binTree b;

int index=0;

binTree **enterBT**(QStringList string)

{

QString ch;

binTree p, q;

ch = string[index];

index++;

if (ch == "/")

return NULL;

else

{

p = enterBT(string);

q = enterBT(string);

return ConsBT(ch, p, q);

}

}

void MainWindow::**on\_pushButton\_clicked**()

{

window()->setFixedSize(765, 350);

int val=0;

if (!isNull(b))

{

val++;

ui->label\_0->*setVisible*(1);

ui->label\_name\_1->*setVisible*(1);

ui->label\_name\_1->setText(RootBT(b));

}

if (!isNull(Left(b)))

{

val++;

ui->label1\_1->*setVisible*(1);

ui->label\_2->*setVisible*(1);

ui->label\_name\_2->*setVisible*(1);

ui->label\_name\_2->setText(RootBT(Left(b)));

}

if (!isNull(Right(b)))

{

val++;

ui->label1\_2->*setVisible*(1);

ui->label\_3->*setVisible*(1);

ui->label\_name\_3->*setVisible*(1);

ui->label\_name\_3->setText(RootBT(Right(b)));

}

if (!isNull(Left(Left(b))))

{

val++;

ui->label4\_1->*setVisible*(1);

ui->label2\_1->*setVisible*(1);

ui->label\_name\_4->*setVisible*(1);

ui->label\_name\_4->setText(RootBT(Left(Left(b))));

}

if (!isNull(Right(Left(b))))

{

val++;

ui->label4\_2->*setVisible*(1);

ui->label2\_2->*setVisible*(1);

ui->label\_name\_5->*setVisible*(1);

ui->label\_name\_5->setText(RootBT(Right(Left(b))));

}

if (!isNull(Left(Right(b))))

{

val++;

ui->label4\_3->*setVisible*(1);

ui->label3\_1->*setVisible*(1);

ui->label\_name\_6->*setVisible*(1);

ui->label\_name\_6->setText(RootBT(Left(Right(b))));

}

if (!isNull(Right(Right(b))))

{

val++;

ui->label4\_4->*setVisible*(1);

ui->label3\_2->*setVisible*(1);

ui->label\_name\_7->*setVisible*(1);

ui->label\_name\_7->setText(RootBT(Right(Right(b))));

}

}

void MainWindow::**on\_action\_triggered**()

{

QString fileName = QFileDialog::getOpenFileName(this, tr("Открыть файл"), QString(),

tr("Text Files (\*.txt)"));

if (!fileName.isEmpty())

{

QFile file(fileName);

if (!file.*open*(QIODevice::ReadOnly))

{

QMessageBox::critical(this, tr("Ошибка"), tr("Не могу открыть данный файл"));

return;

}

QTextStream in(&file);

ui->textEdit->setText(in.readAll());

file.*close*();

}

}

void MainWindow::**on\_action\_2\_triggered**()

{

QString fileName = QFileDialog::getSaveFileName(this, tr("Сохранить файл"), QString(),

tr("Text Files (\*.txt)"));

if (!fileName.isEmpty())

{

QFile file(fileName);

if (!file.*open*(QIODevice::WriteOnly))

{

QMessageBox::critical(this, tr("Ошибка"), tr("Не могу открыть данный файл"));

return;

} else

{

QTextStream stream(&file);

stream << ui->textEdit\_2->toPlainText();

stream.flush();

file.*close*();

}

}

}

void MainWindow::**on\_action\_3\_triggered**()

{

QMessageBox::about(this, tr("О программе"), tr("Задача: Задано бинарное дерево b типа ВТ с типом элементов Elem. Для введенной пользователем величины Е (var E: Elem) определить число вхождений элемента Е в дерево b."));

return;

}

void MainWindow::**on\_action\_4\_triggered**()

{

QMessageBox::about(this, tr("Об авторе"), tr("Лабораторная работа №4\n\nВыполнил: Ковынев М.В.\nГруппа: 6304"));

return;

}

bool **isCorrect**(QStringList list)

{

int br=0;

int val=0;

for (int i=0; i< list.size(); i++)

{

if (list[i]=="/")

br++;

else val++;

}

return (val+1==br);

}

void MainWindow::**on\_textEdit\_textChanged**()

{

clear\_tree();

QString string = ui->textEdit->toPlainText();

QStringList list = string.split(QRegExp("\\s"), QString::SkipEmptyParts);

if (isCorrect(list))

{

ui->warningl->setText("Введенное представление дерево корректно!");

ui->warningl->setStyleSheet("QLabel {color : green; }");

ui->pushButton->setEnabled(1);

ui->pushButton\_2->setEnabled(1);

ui->comboBox->setEnabled(1);

ui->textEdit\_2->setEnabled(1);

}

else

{

ui->warningl->setText("Введенное представление дерево некорректно!");

ui->warningl->setStyleSheet("QLabel {color : red; }");

ui->pushButton->setEnabled(0);

ui->pushButton\_2->setEnabled(0);

ui->comboBox->setEnabled(0);

ui->textEdit\_2->setEnabled(0);

}

}

int **HeightOfTree**(binTree b)

{

if(isNull(b))

return 0;

int left, right;

if (!isNull(Left(b)))

{

left = HeightOfTree(Left(b));

}

else

left = -1;

if (!isNull(Right(b)))

{

right = HeightOfTree(Right(b));

}

else

right = -1;

int max = left > right ? left : right;

return max+1;

}

void MainWindow::**on\_pushButton\_2\_clicked**()

{

ui->comboBox->clear();

index=0;

QString string = ui->textEdit->toPlainText();

QStringList list = string.split(QRegExp("\\s"), QString::SkipEmptyParts);

b = enterBT(list);

if (HeightOfTree(b) >2)

{

ui->pushButton->setDisabled(1);

ui->warningl->setText("Высота дерева больше 3. Отображение невозможно.");

ui->warningl->setStyleSheet("QLabel {color : red; }");

}

for(int i = 0; i < list.size(); i++)

{

bool b = true;

for(int j = i + 1; j < list.size(); j++)

{

if(list[i] == list[j])

b = false;

}

if(b)

{

if (list[i] !="/")

{

ui->comboBox->addItem(list[i]);

}

}

}

}

int counter=0;

void **go**(binTree b, QString str)

{

if (!isNull(Left(b))) go(Left(b), str);

if (!isNull(Right(b))) go(Right(b), str);

if (RootBT(b) == str)

counter++;

}

void MainWindow::**on\_comboBox\_currentTextChanged**(const QString &arg1)

{

counter=0;

go(b, ui->comboBox->currentText());

ui->textEdit\_2->setText("Найдено элементов (" +ui->comboBox->currentText()+"): "+QString::number(counter));

}

void MainWindow::**on\_radioButton\_clicked**()

{

srand(time(NULL));

QStringList list = {"Я Мама Бабушка / / Дедушка / / Папа Бабушка / / Дедушка / /", "Qt / is best / / program / /", "Корень Узел Лист / / Лист / / Лист / /", "ОдинЭлемент / /"};

ui->textEdit->setText(list[rand()%list.size()]);

}

# **ПРИЛОЖЕНИЕ В.**

# **ИХОДНЫЙ КОД MAINWINDOW.H**

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <bintree.h>

namespace Ui {

class MainWindow;

}

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit MainWindow(QWidget \*parent = 0);

~***MainWindow***();

private slots:

void **on\_pushButton\_clicked**();

void **on\_action\_triggered**();

void **on\_action\_2\_triggered**();

void **on\_action\_3\_triggered**();

void **on\_action\_4\_triggered**();

void **on\_textEdit\_textChanged**();

void **on\_pushButton\_2\_clicked**();

void **clear\_tree**();

void **on\_comboBox\_currentTextChanged**(const QString &arg1);

void **on\_radioButton\_clicked**();

private:

Ui::MainWindow \*ui;

};

#endif // MAINWINDOW\_H

# **ПРИЛОЖЕНИЕ Г.**

# **ИХОДНЫЙ КОД BINTREE.H**

#include "QString"

namespace binTree\_modul

{

//-------------------------------------

typedef QString base;

struct node

{

base info;

node \*lt;

node \*rt;

// constructor

node() { lt = NULL; rt = NULL; }

};

typedef node \*binTree; // "представитель" бинарного дерева

binTree **Create**(void);

bool isNull(binTree);

bool RooBT(binTree);

base RootBT(binTree); // для непустого бин.дерева

binTree Left(binTree);// для непустого бин.дерева

binTree Right(binTree);// для непустого бин.дерева

binTree **ConsBT**(const base &x, binTree &lst, binTree &rst);

void destroy(binTree&);

}

# **ПРИЛОЖЕНИЕ Д.**

# **ИХОДНЫЙ КОД BINTREE.CPP**

#include <iostream>

#include <cstdlib>

#include "bintree.h"

using namespace std;

namespace binTree\_modul

{

binTree **Create**()

{

return NULL;

}

bool **isNull**(binTree b)

{

return (b == NULL);

}

base **RootBT**(binTree b)

{

if (b != NULL)

return b->info;

}

bool **RooBT**(binTree b)

{

if (b == NULL) return false;

else return true;

}

binTree **Left**(binTree b)

{

if (b != NULL)

return b->lt;

}

binTree **Right**(binTree b)

{

if (b != NULL)

return b->rt;

}

binTree **ConsBT**(const base &x, binTree &lst, binTree &rst)

{

binTree p;

p = new node;

if (p != NULL)

{

p->info = x;

p->lt = lst;

p->rt = rst;

return p;

}

}

void **destroy**(binTree &b)

{

if (b != NULL)

{

destroy(b->lt);

destroy(b->rt);

delete b;

b = NULL;

}

}

}