**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра Математического обеспечения и применения ЭВМ**

отчет

**по курсовой работе**

**по дисциплине «Алгоритмы и Структуры Данных»**

Тема: **Случайные бинарные деревья поиска**

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

Санкт-Петербург

2017

Содержание

[Цель работы. 3](#_Toc498468141)

[Постановка задачи. 3](#_Toc498468142)

[Ход работы. 3](#_Toc498468143)

[1. Анализ. 3](#_Toc498468144)

[2. Формальная постановка задачи. 4](#_Toc498468145)

[2.1. Исходные данные. 4](#_Toc498468146)

[2.2. Результирующие данные. 4](#_Toc498468147)

[3. Алгоритм работы программы. 4](#_Toc498468148)

[4. Спецификация программы. 4](#_Toc498468149)

[4.1. Входные данные. 4](#_Toc498468150)

[4.2. Выходные данные. 4](#_Toc498468151)

[5. Реализация бинарного дерева. 5](#_Toc498468152)

[6. Описание функций для работы бинарным деревом. 5](#_Toc498468153)

[7. Примеры работы программы. 7](#_Toc498468154)

[Вывод. 8](#_Toc498468155)

[ПРИЛОЖЕНИЕ А 9](#_Toc498468156)

[ПРИЛОЖЕНИЕ Б 10](#_Toc498468157)

[ПРИЛОЖЕНИЕ В 12](#_Toc498468158)

[ПРИЛОЖЕНИЕ Г 13](#_Toc498468159)

[ПРИЛОЖЕНИЕ Д 15](#_Toc498468160)

[ПРИЛОЖЕНИЕ Е 16](#_Toc498468161)

[ПРИЛОЖЕНИЕ Ж 18](#_Toc498468162)

[ПРИЛОЖЕНИЕ И 19](#_Toc498468163)

[ПРИЛОЖЕНИЕ К 21](#_Toc498468164)

[ПРИЛОЖЕНИЕ Л 22](#_Toc498468165)

[ПРИЛОЖЕНИЕ М 23](#_Toc498468166)

# Цель работы.

Знакомство с моделью случайных бинарных деревьев поиска. Построение алгоритмов по его реализации и обработке

# Постановка задачи.

Необходимо задать случайное бинарное дерево поиска путем вставки и исключения случайных чисел, генерируемых в программе или вводимых пользователем с клавиатуры и продемонстрировать поиск в полученном бинарном дереве заданного числа.

# Ход работы.

## **Анализ.**

В исходных данных имеем последовательность ключей узлов. Мы должны, беря последовательно эти ключи добавлять их в наше дерево специальным образом. Случайным образом мы должны для каждого ключа определять способ вставки. Всего их два:

* Обычная (рекурсивная) вставка элемента в дерево
* Вставка в корень – эта вставка использует вспомогательные элементы, сначала рекурсивно вставляем этот узел в корень правого или левого поддерева, в зависимости от сравнения с текущим корнем дерева, а после используем левый/правый поворот, чтобы вставляемый узел оказался корнем нового дерева.

Чередуя, таким образом, различные вставки мы можем получить случайное БДП, при этом метод рандомизации позволяет уменьшить шанс получить худшее случайное БДП – самое высокое из возможных.

## **Формальная постановка задачи.**

### **Исходные данные.**

Последовательность ключей узлов случайного бинарного дерева поиска.

### **Результирующие данные.**

Построенное случайное бинарного дерева поиска.

## **Алгоритм работы программы.**

Программа выполняет следующую последовательность действий:

* Запрашивает входную последовательность
* Обрабатывает входную последовательность и строит результирующее дерево
* Выводит конечное дерево на экран
* Для заданного элемента строит путь его нахождения бинарном дереве

## **Спецификация программы.**

### **Входные данные.**

1. Ограничения на входные данные:

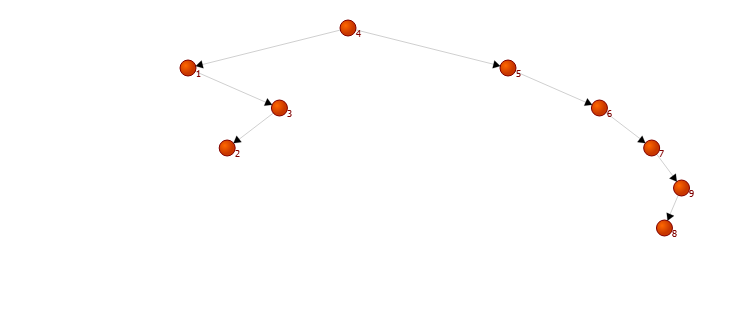
Каждый из ключей должен быть представлен числом, входящим в пределы переменной *int*.

1. Место и форма предоставления входных данных

Входные данные задаются в окне поля ввода *TextEdit* программы, каждый из ключей разделяется пробелами, данные вводятся по нажатию кнопки «Отобразить дерево»

* 1. **Выходные данные.**

Результат работы выводиться в программе в виде графа с подвижными вершинами.

Рисунок 1. Пример построенного случайного бинарного дерева поиска для последовательности «1 2 3 4 5 6 7 8 9»

## **Реализация бинарного дерева.**

Для хранения данных используются следующая структура *Node*:

|  |  |  |
| --- | --- | --- |
| **Имя переменной** | **Тип переменной** | **Назначение** |
| *value* | *int* | Ключ узла |
| *Size* | *int* | Количество узлов в поддеревьях узла |
| *Left* | *Node\** | Указатель на левое поддерево |
| *Right* | *Node\** | Указатель на правое поддерево |

Таблица 1 — Описание переменных структуры *Node*

## **Описание функций для работы бинарным деревом.**

1. *int getsize(node\* p);*

Возвращает *size* узла *node\* p*

1. *void fixsize(node\* p);*

Пересчитывает size узла *node\* p*

1. *node\* rotateright(node\* p);*

Совершает правый поворот бинарного дерева *node\* p*

1. *node\* rotateleft(node\* p);*

Совершает левый поворот бинарного дерева *node\* p*

1. *node\* insertroot(node\* p, int k);*

Совершает вставку k узла в корень дерева *node\* p*

1. *node\* insertrandom (node\* p, int k, int &t);*

Строит случайное бинарное дерево *node\* p* поиска с рандомизацией

1. *bool isNull(binTree);*

Проверяет узел *binTree* на пустоту

1. *int RootBT (binTree);*

Возвращает ключ корня, передаваемого дерева *binTree*

1. *binTree Left (binTree);*

Возвращает левое поддерево текущего корня *binTree*

1. *binTree Right (binTree);*

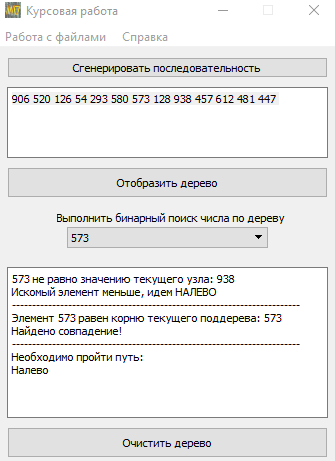
Возвращает правое поддерево текущего корня *binTree*

1. int CountTreeEl(binTree b);

Возвращает количество ненулевых элементов текущего корня *binTree*

## **Примеры работы программы.**

1. Пример №1

 Входные данные: «906 520 126 54 293 580 573 128 938 457 612 481 447»

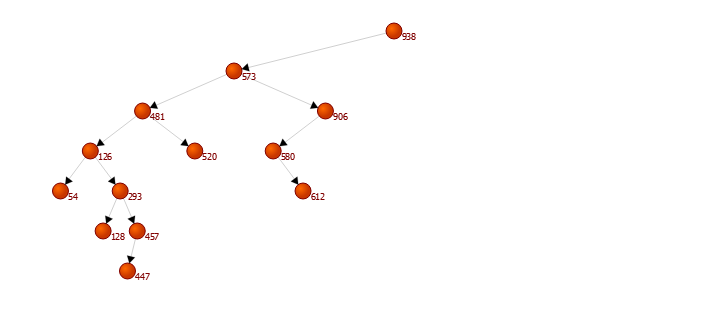
Рисунок 2. Вид окна программы для примера №1

Рисунок 3. Пример построенного случайного бинарного дерева поиска для примера №1

1. Пример №2

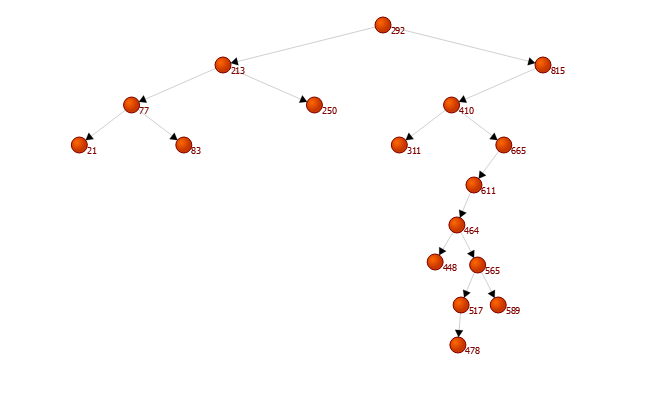
Входные данные: «292 517 815 589 565 665 213 464 250 77 410 83 478 21 311 448 611»

Рисунок 3. Пример построенного случайного бинарного дерева поиска для примера №2

# Вывод.

В ходе выполнения данной курсовой работы была изучена модель случайного бинарного дерева поиска. Также были построены алгоритмы по его реализации и обработке и создана графическая интерпретация данного бинарного дерева как графа с подвижными вершинами в среде разработки Qt.

# ПРИЛОЖЕНИЕ А

**ИСХОДНЫЙ КОД BINTREE.H**

#include <iostream>

namespace binTree\_modul

{

struct node

{

int value;

int size;

node\* left;

node\* right;

node(int k) { value = k; left = right = 0; size = 1; }

};

typedef node \*binTree;

int getsize(node\* p);

void fixsize(node\* p);

node\* **rotateright**(node\* p);

node\* **rotateleft**(node\* q);

node\* **insertroot**(node\* p, int k);

node\* **insertrandom**(node\* p, int k, int &t);

bool isNull(binTree);

int RootBT(binTree);

binTree Left(binTree);

binTree Right(binTree);

int **CountTreeEl**(binTree b);

}

# ПРИЛОЖЕНИЕ Б

**ИСХОДНЫЙ КОД BINTREE.CPP**

#include "bintree.h"

#include <iostream>

using namespace std;

namespace binTree\_modul

{

int **getsize**(node\* p)

{

if (!p) return 0;

return p->size;

}

void **fixsize**(node\* p)

{

p->size = getsize(p->left) + getsize(p->right) + 1;

}

node\* **rotateright**(node\* p)

{

node\* q = p->left;

if (!q) return p;

p->left = q->right;

q->right = p;

q->size = p->size;

fixsize(p);

return q;

}

node\* **rotateleft**(node\* q)

{

node\* p = q->right;

if (!p) return q;

q->right = p->left;

p->left = q;

p->size = q->size;

fixsize(q);

return p;

}

node\* **insertroot**(node\* p, int k)

{

if (!p)

return new node(k);

if (k<p->value)

{

p->left = insertroot(p->left, k);

return rotateright(p);

}

else

{

p->right = insertroot(p->right, k);

return rotateleft(p);

}

}

node\* **insertrandom**(node\* p, int k, int &t)

{

if (!p)

{

return new node(k);

t = 0;

}

int random= rand()%10;

if (!random)

{

t = 1;

return insertroot(p, k);

}

if (p->value>k)

{

p->left = insertrandom(p->left, k, t);

t = 0;

}

else

{

p->right = insertrandom(p->right, k, t);

t = 0;

}

fixsize(p);

return p;

}

bool **isNull**(binTree b)

{

return (b == NULL);

}

int **RootBT**(binTree b)

{

if (b == NULL) { exit(1); }

else return b->value;

}

binTree **Left**(binTree b)

{

if (b == NULL) { exit(1); }

else return b->left;

}

binTree **Right**(binTree b)

{

if (b == NULL) { exit(1); }

else return b->right;

}

int **CountTreeEl**(binTree b)

{

int c = 1;

if (b ==NULL)

return 0;

else

{

c += CountTreeEl(Left(b));

c += CountTreeEl(Right(b));

return c;

}

}

}

# ПРИЛОЖЕНИЕ В

**ИСХОДНЫЙ КОД EDGE.H**

#ifndef EDGE\_H

#define EDGE\_H

#include <QGraphicsItem>

class Node;

class Edge : public QGraphicsItem

{

public:

Edge(Node \*sourceNode, Node \*destNode);

Node \***sourceNode**() const;

Node \***destNode**() const;

void **adjust**();

enum { Type = UserType + 2 };

int ***type***() const override { return Type; }

protected:

QRectF ***boundingRect***() const override;

void ***paint***(QPainter \*painter, const QStyleOptionGraphicsItem \*option, QWidget \*widget) override;

private:

Node \*source, \*dest;

QPointF sourcePoint;

QPointF destPoint;

qreal arrowSize;

};

#endif // EDGE\_H

# ПРИЛОЖЕНИЕ Г

**ИСХОДНЫЙ КОД EDGE.CPP**

#include "edge.h"

#include "node.h"

#include <math.h>

#include <QPainter>

static const double Pi = 3.14159265358979323846264338327950288419717;

static double TwoPi = 2.0 \* Pi;

Edge::**Edge**(Node \*sourceNode, Node \*destNode)

: arrowSize(10)

{

setAcceptedMouseButtons(0);

source = sourceNode;

dest = destNode;

source->addEdge(this);

dest->addEdge(this);

adjust();

}

Node \*Edge::**sourceNode**() const

{

return source;

}

Node \*Edge::**destNode**() const

{

return dest;

}

void Edge::**adjust**()

{

if (!source || !dest)

return;

QLineF line(mapFromItem(source, 0, 0), mapFromItem(dest, 0, 0));

qreal length = line.length();

prepareGeometryChange();

if (length > qreal(20.)) {

QPointF edgeOffset((line.dx() \* 10) / length, (line.dy() \* 10) / length);

sourcePoint = line.p1() + edgeOffset;

destPoint = line.p2() - edgeOffset;

} else {

sourcePoint = destPoint = line.p1();

}

}

QRectF Edge::***boundingRect***() const

{

if (!source || !dest)

return QRectF();

qreal penWidth = 1;

qreal extra = (penWidth + arrowSize) / 2.0;

return QRectF(sourcePoint, QSizeF(destPoint.x() - sourcePoint.x(),

destPoint.y() - sourcePoint.y()))

.normalized()

.adjusted(-extra, -extra, extra, extra);

}

void Edge::***paint***(QPainter \*painter, const QStyleOptionGraphicsItem \*, QWidget \*)

{

if (!source || !dest)

return;

QLineF line(sourcePoint, destPoint);

if (qFuzzyCompare(line.length(), qreal(0.)))

return;

// Draw the line itself

painter->setPen(QPen(QColor(0,0,0,60), 1, Qt::SolidLine, Qt::RoundCap, Qt::RoundJoin));

painter->drawLine(line);

// Draw the arrows

double angle = ::acos(line.dx() / line.length());

if (line.dy() >= 0)

angle = TwoPi - angle;

QPointF destArrowP1 = destPoint + QPointF(sin(angle - Pi / 3) \* arrowSize,

cos(angle - Pi / 3) \* arrowSize);

QPointF destArrowP2 = destPoint + QPointF(sin(angle - Pi + Pi / 3) \* arrowSize,

cos(angle - Pi + Pi / 3) \* arrowSize);

painter->setBrush(Qt::black);

painter->drawPolygon(QPolygonF() << line.p2() << destArrowP1 << destArrowP2);

}

# ПРИЛОЖЕНИЕ Д

**ИСХОДНЫЙ КОД NODE.H**

#ifndef NODE\_H

#define NODE\_H

#include <QGraphicsItem>

#include <QList>

class Edge;

class GraphWidget;

class QGraphicsSceneMouseEvent;

class Node : public QGraphicsItem

{

public:

Node(GraphWidget \*graphWidget);

void **addEdge**(Edge \*edge);

QList<Edge \*> **edges**() const;

enum { Type = UserType + 1 };

int ***type***() const override { return Type; }

void **calculateForces**();

QRectF ***boundingRect***() const override;

QPainterPath ***shape***() const override;

void ***paint***(QPainter \*painter, const QStyleOptionGraphicsItem \*option, QWidget \*) override;

protected:

QVariant ***itemChange***(GraphicsItemChange change, const QVariant &value) override;

void ***mousePressEvent***(QGraphicsSceneMouseEvent \*event) override;

void ***mouseReleaseEvent***(QGraphicsSceneMouseEvent \*event) override;

private:

QList<Edge \*> edgeList;

QPointF newPos;

GraphWidget \*graph;

};

#endif // NODE\_H

# ПРИЛОЖЕНИЕ Е

**ИСХОДНЫЙ КОД NODE.CPP**

#include "edge.h"

#include "node.h"

#include "graphwidget.h"

#include <QGraphicsScene>

#include <QGraphicsSceneMouseEvent>

#include <QPainter>

#include <QStyleOption>

Node::**Node**(GraphWidget \*graphWidget)

: graph(graphWidget)

{

setFlag(ItemIsMovable);

setFlag(ItemSendsGeometryChanges);

setCacheMode(DeviceCoordinateCache);

setZValue(-1);

}

void Node::**addEdge**(Edge \*edge)

{

edgeList << edge;

edge->adjust();

}

QList<Edge\*> Node::**edges**() const

{

return edgeList;

}

void Node::**calculateForces**()

{

if (!scene() || scene()->mouseGrabberItem() == this)

{

newPos = pos();

return;

}

QRectF sceneRect = scene()->sceneRect();

newPos = pos();

newPos.setX(qMin(qMax(newPos.x(), sceneRect.left() + 10), sceneRect.right() - 10));

newPos.setY(qMin(qMax(newPos.y(), sceneRect.top() + 10), sceneRect.bottom() - 10));

}

QRectF Node::***boundingRect***() const

{

qreal adjust = 2;

return QRectF( -10 - adjust, -10 - adjust, 23 + adjust+200, 23 + adjust+200);

}

QPainterPath Node::***shape***() const

{

QPainterPath path;

path.addEllipse(-10, -10, 20, 20);

return path;

}

QVariant Node::***itemChange***(GraphicsItemChange change, const QVariant &value)

{

switch (change) {

case ItemPositionHasChanged:

foreach (Edge \*edge, edgeList)

edge->adjust();

break;

default:

break;

};

return QGraphicsItem::*itemChange*(change, value);

}

void Node::***mousePressEvent***(QGraphicsSceneMouseEvent \*event)

{

QGraphicsItem::*mousePressEvent*(event);

}

void Node::***mouseReleaseEvent***(QGraphicsSceneMouseEvent \*event)

{

QGraphicsItem::*mouseReleaseEvent*(event);

}

# ПРИЛОЖЕНИЕ Ж

**ИСХОДНЫЙ КОД GRAPHWIDGET.H**

#ifndef GRAPHWIDGET\_H

#define GRAPHWIDGET\_H

#include <QGraphicsView>

#include <bintree.h>

using namespace binTree\_modul;

class Node;

class GraphWidget : public QGraphicsView

{

Q\_OBJECT

public:

GraphWidget(QWidget \*parent = 0);

public slots:

void **addElementsToScene**(binTree, QGraphicsScene\*, Node\*, bool, double x, double y, double);

void **zoomIn**();

void **zoomOut**();

protected:

void ***keyPressEvent***(QKeyEvent \*event) override;

void ***timerEvent***(QTimerEvent \*event) override;

#if QT\_CONFIG(wheelevent)

void wheelEvent(QWheelEvent \*event) override;

#endif

void ***drawBackground***(QPainter \*painter, const QRectF &rect) override;

void **scaleView**(qreal);

private:

int timerId;

Node \*centerNode;

};

#endif // GRAPHWIDGET\_H

# ПРИЛОЖЕНИЕ И

**ИСХОДНЫЙ КОД GRAPHWIDGET.CPP**

#include "graphwidget.h"

#include "edge.h"

#include "node.h"

#include <math.h>

#include <vector>

#include <iostream>

using namespace std;

#include <QKeyEvent>

void GraphWidget::***keyPressEvent***(QKeyEvent \*event)

{

switch (event->key())

{

case Qt::Key\_Plus:

zoomIn();

break;

case Qt::Key\_Minus:

zoomOut();

break;

}

}

void GraphWidget::***timerEvent***(QTimerEvent \*event)

{

Q\_UNUSED(event);

QList<Node \*> nodes;

foreach (QGraphicsItem \*item, scene()->items())

{

if (Node \*node = qgraphicsitem\_cast<Node \*>(item))

nodes << node;

}

foreach (Node \*node, nodes)

node->calculateForces();

}

#if QT\_CONFIG(wheelevent)

void GraphWidget::wheelEvent(QWheelEvent \*event)

{

scaleView(pow((double)2, -event->delta() / 240.0));

}

#endif

void GraphWidget::***drawBackground***(QPainter \*painter, const QRectF &rect)

{

Q\_UNUSED(rect);

QRectF sceneRect = this->sceneRect();

QLinearGradient gradient(sceneRect.topLeft(), sceneRect.bottomRight());

gradient.setColorAt(0, Qt::white);

painter->drawRect(sceneRect);

// Text

QRectF textRect(sceneRect.left() + 4, sceneRect.top() + 4,

sceneRect.width() - 4, sceneRect.height() - 4);

QString message(tr("Ваше дерево: "));

QFont font = painter->font();

font.setBold(true);

font.setPointSize(10);

painter->setFont(font);

painter->setPen(Qt::black);

painter->drawText(textRect, message);

}

void GraphWidget::**scaleView**(qreal scaleFactor)

{

qreal factor = transform().scale(scaleFactor, scaleFactor).mapRect(QRectF(0, 0, 1, 1)).width();

if (factor < 0.07 || factor > 100)

return;

scale(scaleFactor, scaleFactor);

}

void GraphWidget::**zoomIn**()

{

scaleView(qreal(1.2));

}

void GraphWidget::**zoomOut**()

{

scaleView(1 / qreal(1.2));

}

# ПРИЛОЖЕНИЕ К

**ИСХОДНЫЙ КОД MAIN.CPP**

#include "mainwindow.h"

#include <QApplication>

#include <time.h>

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

{

srand(time(NULL));

QApplication a(argc, argv);

MainWindow w;

w.setFixedSize(340,435);

w.setWindowTitle("Курсовая работа");

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

w.show();

return a.exec();

}

# ПРИЛОЖЕНИЕ Л

**ИСХОДНЫЙ КОД MAINWINDOW.H**

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include "graphwidget.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\_textEdit\_textChanged**();

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

void **on\_comboBox\_currentIndexChanged**(int index);

void **bsearchBinTree**(binTree, int, QString);

void **on\_pushButton\_3\_clicked**();

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

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

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

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

private:

Ui::MainWindow \*ui;

};

#endif // MAINWINDOW\_H

# ПРИЛОЖЕНИЕ М

**ИСХОДНЫЙ КОД MAINWINDOW.CPP**

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <QKeyEvent>

#include "graphwidget.h"

#include "edge.h"

#include "node.h"

#include <math.h>

#include <vector>

#include <iostream>

#include <QProcess>

#include <QFile>

#include <QFileDialog>

#include <QMessageBox>

#include <QTextStream>

using namespace std;

int numberTreeEl;

binTree b;

vector<int> values(numberTreeEl);

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

QMainWindow(parent),

ui(new Ui::MainWindow)

{

ui->setupUi(this);

ui->textEdit->setPlaceholderText

("Введите последовательность чисел для создания бинарного дерева");

ui->pushButton->setEnabled(0);

ui->comboBox->setEnabled(0);

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

}

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

{

delete ui;

}

bool isCreatedBinTreeRoot = false;

void GraphWidget::**addElementsToScene**(binTree b, QGraphicsScene \*scene,

Node\* rootElement, bool isLeftElement, double x, double y, double ratioImmersion)

{

Node\* aa =new Node(this);

if (!isCreatedBinTreeRoot)

{

centerNode =new Node(this);

scene->addItem(centerNode);

centerNode->setPos(x,y);

isCreatedBinTreeRoot = true;;

if (Left(b)!=NULL)

addElementsToScene(Left(b), scene, centerNode, 1, x, y, ratioImmersion);

if (Right(b)!=NULL)

addElementsToScene(Right(b), scene, centerNode, 0, x, y, ratioImmersion);

}

else

{

scene->addItem(aa);

if (isLeftElement)

x -=(50.0\*ratioImmersion);

else

x +=(50.0\*ratioImmersion);

y +=50.0;

aa->setPos(x,y);

scene->addItem(new Edge(rootElement, aa));

if (Left(b)!=NULL)

addElementsToScene(Left(b), scene, aa, 1, x,y, ratioImmersion/1.75);

if (Right(b)!=NULL)

addElementsToScene(Right(b), scene, aa, 0, x,y, ratioImmersion/1.75);

}

}

GraphWidget::**GraphWidget**(QWidget \*parent) : QGraphicsView(parent), timerId(0)

{

QGraphicsScene \*scene = new QGraphicsScene(this);

scene->setItemIndexMethod(QGraphicsScene::NoIndex);

scene->setSceneRect(-900, -400, 1800, 800);

setScene(scene);

setCacheMode(CacheBackground);

setViewportUpdateMode(BoundingRectViewportUpdate);

setRenderHint(QPainter::Antialiasing);

setTransformationAnchor(AnchorUnderMouse);

scale(qreal(0.8), qreal(0.8));

setMinimumSize(400, 400);

setWindowTitle(tr("Курсовая работа"));

addElementsToScene(b,scene, NULL, 0, 0.0, -350.0, 4);

}

int indexNameNode=0;

void Node::***paint***(QPainter \*painter, const QStyleOptionGraphicsItem \*option, QWidget \*)

{

QRadialGradient gradient(-3, -3, 10);

QRectF textRect(10,0, 200,200);

QFont font = painter->font();

font.setBold(true);

font.setPointSize(16);

gradient.setColorAt(0, QColor(255, 104, 0));

gradient.setColorAt(1, QColor(200, 55, 0));

painter->setBrush(gradient);

painter->setPen(QPen(Qt::darkRed, 0));

painter->drawEllipse(-10, -10, 20, 20);

painter->drawText(textRect, QString::

number(values[indexNameNode % numberTreeEl]));

indexNameNode++;

}

int indexNameNode2=0;

void **bypassBinTree**(binTree b)

{

values[indexNameNode2]=RootBT(b);

indexNameNode2++;

if (Left(b)!=NULL)

bypassBinTree(Left(b));

if (Right(b)!=NULL)

bypassBinTree(Right(b));

}

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

{

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

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

int t=0;

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

{

QString x;

x=list[i];

b = insertrandom(b, x.toInt(), t);

}

numberTreeEl = CountTreeEl(b);

values.resize(numberTreeEl);

bypassBinTree(b);

GraphWidget \*widget = new GraphWidget;

widget->show();

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

{

QString x;

x=list[i];

ui->comboBox->addItem(x);

}

}

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

{

ui->pushButton->setEnabled(1);

ui->comboBox->setEnabled(1);

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

}

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

{

QProcess::startDetached(QApplication::applicationFilePath(),

QStringList(), QApplication::applicationDirPath());

exit(1);

}

void MainWindow::**bsearchBinTree**(binTree b, int x, QString path)

{

if (x != RootBT(b))

{

ui->textEdit\_2->setText(ui->textEdit\_2->toPlainText()+ QString::number(x) +

" не равно значению текущего узла: " + QString::number(RootBT(b))+"\n");

if (x>RootBT(b))

{

ui->textEdit\_2->setText(ui->textEdit\_2->toPlainText() +

"Искомый элемент больше, идем НАПРАВО"

"\n------------------------------------------------------------------------\n");

if (Right(b) != NULL)

bsearchBinTree(Right(b),x, path+"Направо\n");

}

else

{

if (x<RootBT(b))

{

ui->textEdit\_2->setText(ui->textEdit\_2->toPlainText() +

"Искомый элемент меньше, идем НАЛЕВО"

"\n------------------------------------------------------------------------\n");

if (Left(b) != NULL)

bsearchBinTree(Left(b),x, path+"Налево\n");

}

}

}

else

{

ui->textEdit\_2->setText(ui->textEdit\_2->toPlainText()+

"Элемент " + QString::number(x) + " равен корню текущего поддерева: "

+ QString::number(RootBT(b))+"\n");

ui->textEdit\_2->setText(ui->textEdit\_2->toPlainText() + "Найдено совпадение!"

"\n------------------------------------------------------------------------\n"

+ "Необходимо пройти путь:\n");

ui->textEdit\_2->setText(ui->textEdit\_2->toPlainText() + path);

}

}

void MainWindow::**on\_comboBox\_currentIndexChanged**(int index)

{

ui->textEdit\_2->clear();

bsearchBinTree(b, ui->comboBox->currentText().toInt(), "");

}

void MainWindow::**on\_pushButton\_3\_clicked**()

{

ui->textEdit->clear();

int len = rand()%10 + 12;

for (int i=0; i < len; i++)

{

ui->textEdit->setText(ui->textEdit->toPlainText() + QString::number(rand()%1000) + " ");

}

}

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\_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\_3\_triggered**()

{

QMessageBox::about(this, tr("О программе"), tr("Необходимо задать случайное бинарное дерево поиска путем вставки и исключения случайных чисел, генерируемых в программе или вводимых пользователем и продемонстрировать поиск в полученном бинарном дереве заданного числа."));

return;

}

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

{

QMessageBox::about(this, tr("Об авторе"), tr("Курсовая работа\n\nВыполнил: Ковынев М.В.\nГруппа: 6304"));

return;

}