**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра математического обеспечения и применения ЭВМ**

**Лабораторная работа №5**

**по дисциплине «Алгоритмы и структуры данных»**

**Тема: Случайные БДП – вставка и исключение.**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 8381 |  | Звегинцева Е.Н. |
| Преподаватель |  | Жангиров Т.Р. |

Санкт-Петербург

2019

**Цель работы.**

Ознакомиться с основными понятиями и приёмами использования БДП , получить навыки решения задач с их помощью. Построение БДП, вставка и исключение элементов.

## 1.1. Описание интерфейса пользователя

Интерфейс программы разделен на две части: верхняя панель ввода и генерации данных, нижняя панель вывода сгенерированного варианта и его визуальное представление. Основные виджеты и их назначение представлены в табл. 1 и табл. 2

Таблица 1 – Основные виджеты верхней панели программы

|  |  |  |
| --- | --- | --- |
| Класс объекта | Название виджета | Назначение |
| QHBoxLayout | horizontalLayout | Выбор сложности варианта |
| QHBoxLayout | horizontalLayout\_2 | Генерация варианта(ов) |
| QHBoxLayout | horizontalLayout\_3 | Выбор исходного дерева, а также предоставление вставки и удаления элементов |
| QSpinBox | varNum | Выбор количества генерируемых вариантов на экран |
| QLineEdit | elementEdit | Окно ввода элемента для удаления/вставки |

Таблица 2 – Основные виджеты правой панели программы

|  |  |  |
| --- | --- | --- |
| Класс объекта | Название виджета | Назначение |
| QTextBrowser | questionBrowser | Окно вывода задания |
| QGraphicsView | questionGraphicsView | Окно для визуализации действия с деревом |
| QTextBrowser | answerBrowser | Окно вывода ответа |
| QGraphicsView | answerGraphicsView | Окно для визуализации предыдущего дерева |

## 

## 1.2. Описание алгоритма вставки и исключения

Бинарное дерево поиска — это бинарное дерево, обладающее дополнительными свойствами: ключ (значение) левого потомка меньше ключа родителя, а значение правого потомка больше значения родителя для каждого узла дерева. То есть, данные в бинарном дереве поиска хранятся в отсортированном виде. Пример такого дерева (которое является также идеально сбалансированным деревом поиска) представлен на рисунке 1.

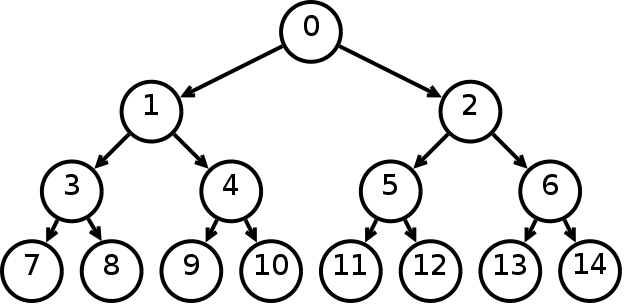


Рисунок 1 - Пример бинарного дерева поиска(перечислены узлы)

При каждой операции вставки нового или удаления существующего узла отсортированный порядок дерева сохраняется.

Случайным бинарным деревом поиска называется такое БДП, в котором последовательность значений, задающая конечное, образовано случайно. Примеры деревьев, которые имеют одинаковые элементы, но с разным порядком этих элементов в последовательности, представлены на рисунка 2 и 3.

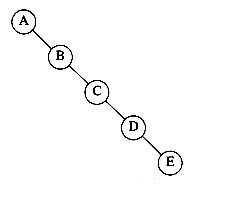
Часто в реализации этого типа деревьев учитывают количество повторяющихся элементов. Для сортировки всех ключей используют обратный порядок обхода дерева (ЛКП-обход).

Рисунок 2 - Случайное дерево поиска для последовательности A,B,C,D,E

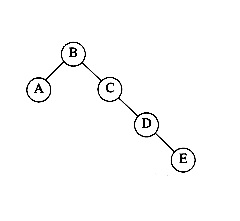


Рисунок 3 - Случайное дерево поиска для последовательности A,B,C,D,E

Вставка в корень происходит следующим образом:

1. Сначала рекурсивно вставляем новый ключ в корень левого или правого поддеревьев (в зависимости от результата сравнения с корневым ключом)
2. Выполняем правый (левый) поворот[2], который поднимает нужный нам узел в корень дерева.

Алгоритм поворотов представлен на рисунке 4. Случайная вставка в дерево включает шанс того, что следующий элемент дерева будет помещен с помощью обычной вставки в дерево или вставкой в корень.

При рекурсивном удаления узла из бинарного дерева нужно рассмотреть три случая:

1. удаляемый элемент находится в левом поддереве текущего поддерева
2. удаляемый элемент находится в правом поддереве
3. удаляемый элемент находится в корне.

В двух первых случаях нужно рекурсивно удалить элемент из нужного поддерева. Если удаляемый элемент находится в корне текущего поддерева и имеет два дочерних узла, то нужно заменить его минимальным элементом из правого поддерева и рекурсивно удалить минимальный элемент из правого поддерева. Иначе, если удаляемый элемент имеет один дочерний узел, нужно заменить его потомком

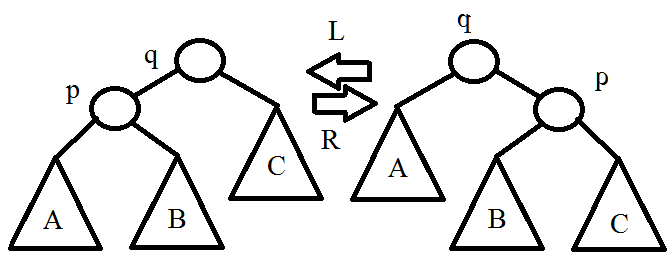


Рисунок 4 -Алгоритм левых и правых поворотов

## 2. Генерация дерева с выводом из файла, вставка и исключение элементов

После ввода дерева, нажимаем кнопку «Генерировать дерево», затем кнопку «Вставить» (предварительно введя число 49 в окно) и происходит визуализация, как представлено на рис. 5.

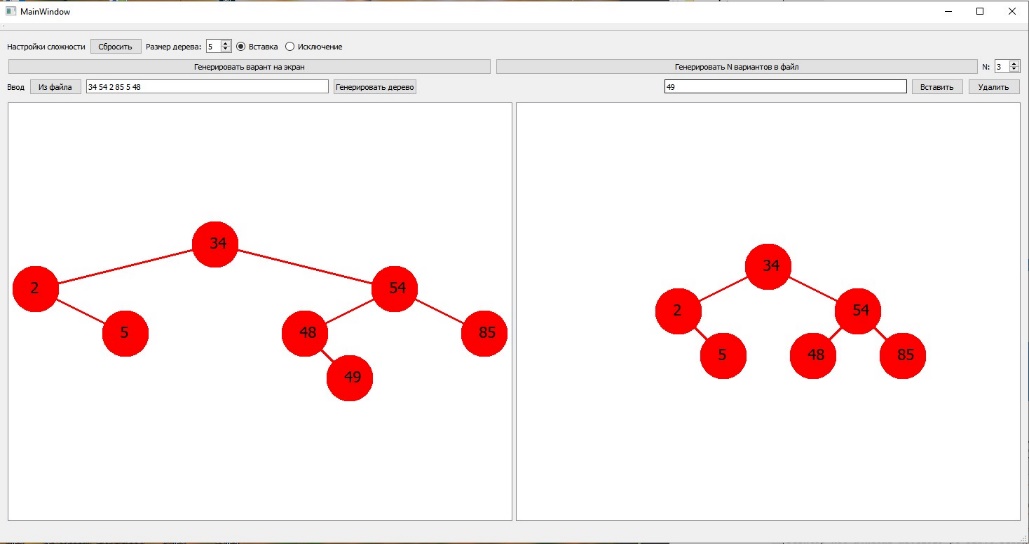


Рисунок 5 – Вид программы после генерации задания на вставку элемента в заданное дерево

На рис. 6 представлено исключение элемента 54 из введенного дерева

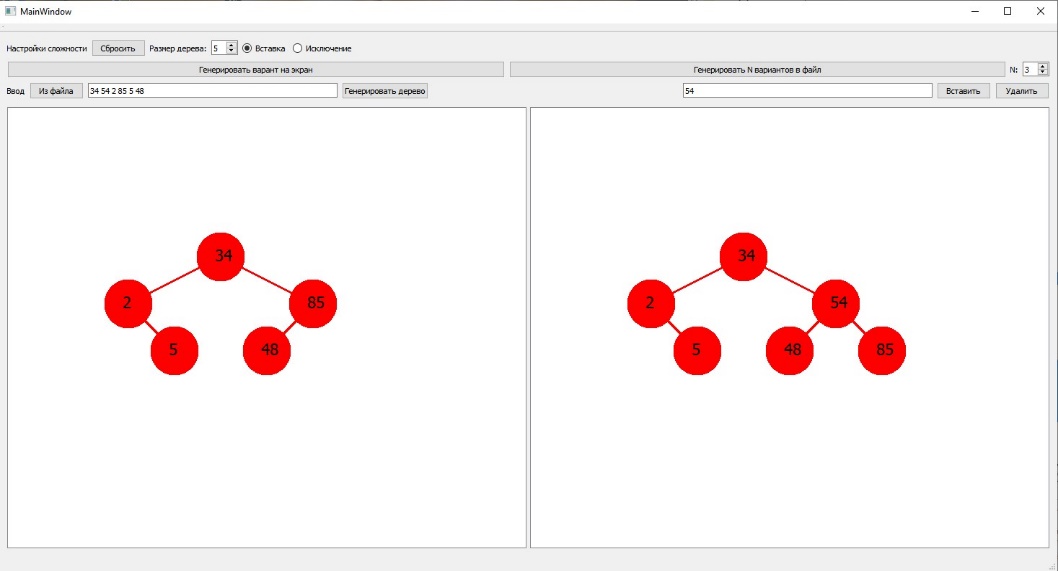


Рисунок 6 – Вид программы после генерации задания на исключение элемента из заданного дерева

При нажатии кнопки «Сбросить» программа приходит в начальный вид, представленный на рис.1

# Заключение

В ходе выполнения лабораторной работы была разработана программа, которая генерирует случайное БДП(или записывает из файла) и имеет возможность вставить и удалить элемент, наглядно показывая как это визуально выглядит.

# Приложение а

# Исходный код программы. main.cpp

#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 <QMainWindow>

#include <QGraphicsScene>

#include <QGraphicsTextItem>

#include <QFileDialog>

#include <QMessageBox>

#include <QTextStream>

#include <cmath>

#include <ctime>

#include <cstdlib>

#include <node.h>

#include "bintree.h"

#include "testing.h"

namespace Ui {

class MainWindow;

}

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit MainWindow(QWidget \*parent = nullptr);

~MainWindow();

void reset();

void visualize();

void draw(QGraphicsScene\* scene, Node \*n, int maxdepth, int depth = 0, int x = 0, int y = 0);

int read(bool& ok);

binTree\* questionTree;

binTree\* answerTree;

QGraphicsScene\* questionGraphicsScene;

QGraphicsScene\* answerGraphicsScene;

int mode; // 0-insert 1-delete

int difficulty;

int var\_count;

private slots:

void on\_resetButton\_clicked();

void on\_varButton\_clicked();

void on\_genFileButton\_clicked();

void on\_varNum\_valueChanged(int arg1);

void on\_difficultyBox\_valueChanged(int arg1);

void on\_radioInsertButton\_toggled(bool checked);

void on\_radioRemoveButton\_toggled(bool checked);

void on\_inFIleButton\_clicked();

void on\_treeGenButton\_clicked();

void on\_insertButton\_clicked();

void on\_removeButton\_clicked();

private:

Ui::MainWindow \*ui;

};

#endif // MAINWINDOW\_H

# Приложение в исходный код программы. testing.h

#ifndef TESTING\_H

#define TESTING\_H

#include <bintree.h>

#include <string>

#include <iostream>

#include <fstream>

using namespace std;

void generate\_var\_insert(string& question, string& answer, binTree& qtree, binTree& aTree, int difficulty);

void generate\_vars\_insert(string& question, string& answer, int count, int difficulty);

void generate\_var\_remove(string& question, string& answer, binTree& qtree, binTree& aTree, int difficulty);

void generate\_vars\_remove(string& question, string& answer, int count, int difficulty);

void file\_write(string filename, string& data);

#endif // TESTING\_H

# приложение г исходный код программы. bintree.h

#ifndef LAB3\_BINTREE\_H

#define LAB3\_BINTREE\_H

#include <iostream>

#include <string>

#include <vector>

#include <QStringList>

#include <QColor>

#include <string>

#include "array\_list.h"

#include "node.h"

using namespace std;

class binTree {

public:

Node\* root = new Node;

int treeInit(QStringList lst, unsigned int& index);

Node\* enterBT();

int checkTwoEqualElem();

void lkp();

void printLKP();

//

int max\_depth(Node \*n, int i);

bool parse\_tree(Node\*& n, std::string &s, int &i);

void get\_elems(array\_list& vec, Node\* n);

bool get\_duplicates(array\_list& vec, Node\*& first, Node\*& second);

void insert(Node\*& n, int data);

void remove(Node\*& n, int data);

Node\* copy(Node\* n);

string into\_string(Node\* n);

};

#endif //LAB3\_BINTREE\_H

# приложение д 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);

srand(time(nullptr));

type = coding;

method = haffman;

variant\_count = 1;

length = 5;

}

MainWindow::~MainWindow()

{

delete ui;

}

void MainWindow::on\_type\_currentIndexChanged(int index)

{

type = (types)index;

}

void MainWindow::on\_method\_currentIndexChanged(int index)

{

method = (methods)index;

}

void MainWindow::on\_generate\_clicked()

{

string issue;

string answer;

generate\_var(type, method, length, issue, answer);

ui->issue->setText(QString::fromStdString(issue));

ui->answer->setText(QString::fromStdString(answer));

}

void MainWindow::on\_variants\_valueChanged(int arg1)

{

variant\_count = arg1;

}

void MainWindow::on\_length\_valueChanged(int arg1)

{

length = arg1;

}

void MainWindow::on\_file\_clicked()

{

string issue = "";

string answer = "";

generate\_pack(type, method, length, variant\_count, issue, answer);

write\_to\_file("issue.txt", issue);

write\_to\_file("answer.txt", answer);

}

void MainWindow::on\_hand\_textChanged(const QString &arg1)

{

}

void MainWindow::on\_lr5\_clicked()

{

}

# приложение е исходный код программы. node.h

#ifndef NODE\_H

#define NODE\_H

#include <iostream>

#include <string>

#include <vector>

#include <QStringList>

#include <QColor>

struct Node {

int data;

Node \*left = nullptr;

Node \*right = nullptr;

QColor color = QColor::fromRgb(255, 0, 0);

Node() {

left = nullptr;

right = nullptr;

}

};

#endif // NODE\_H

# Приложение ж Исходный код программы. bintree.cpp

#include "bintree.h"

void binTree::lkp() {

Node\* temp = root;

if (root){

//elems.push\_back(root->info);

root = temp->left;

lkp();

root = temp->right;

lkp();

root = temp;

}

}

int binTree::treeInit(QStringList lst, unsigned int& index) {

QString x = lst[index++];

if (x == "/"){

delete root;

root = nullptr;

return 1;

} else {

root->data = x.toInt();

Node\* temp = root;

root = temp->left;

root = new Node;

temp->left = root;

if (treeInit(lst, index)){

temp->left = nullptr;

}

root = temp->right;

root = new Node;

temp->right = root;

if (treeInit(lst, index)){

temp->right = nullptr;

}

root = temp;

}

return 0;

}

int binTree::checkTwoEqualElem() {

//elems.clear();

lkp();

// for (unsigned int i = 0; i < elems.size(); i++){

// for (unsigned int j = i + 1; j < elems.size(); j++){

// //if (elems[i] == elems[j]){

// // return 1;

// //}

// }

// }

return 0;

}

void binTree::printLKP() {

Node\* temp = root;

if (root){

std::cout << root->data << std::endl;

root = temp->left;

printLKP();

root = temp->right;

printLKP();

root = temp;

}

}

int binTree::max\_depth(Node \*n, int i)

{

if (!n) return i;

int l = max\_depth(n->left, i + 1);

int r = max\_depth(n->right, i + 1);

if (l > r) return l;

else return r;

}

bool binTree::parse\_tree(Node\*& n, std::string &s, int &i) {

if (s.size() < 1) return true;

std::size\_t current, previous = 0;

current = s.find(' ');

int num;

while (current != std::string::npos) {

try

{

num = stoi(s.substr(previous, current - previous));

}

catch (...)

{

return true;

}

previous = current + 1;

current = s.find(' ', previous);

insert(n, num);

}

try

{

num = stoi(s.substr(previous, current - previous));

}

catch (...)

{

return true;

}

insert(n, num);

return false;

}

void binTree::get\_elems(array\_list& vec, Node \*n)

{

if (!n) return;

vec.push\_back(n);

get\_elems(vec, n->left);

get\_elems(vec, n->right);

}

bool binTree::get\_duplicates(array\_list& vec, Node\*& first, Node\*& second)

{

int count = vec.size();

for (int i = 0; i < count; i++) {

for (int k = i + 1; k < count; k++) {

if (vec[i]->data == vec[k]->data) {

first = vec[i];

second = vec[k];

return true;

}

}

}

return false;

}

void binTree::insert(Node\*& n, int data)

{

if (!n)

{

n = new Node();

n->data = data;

}

else if (n->data > data)

{

insert(n->left, data);

}

else if (n->data < data)

{

insert(n->right, data);

}

}

void binTree::remove(Node \*&n, int data)

{

if (!n) return;

if (data < n->data)

{

remove(n->left, data);

}

else if (data > n->data)

{

remove(n->right, data);

}

else

{

if (!n->left && n->right)

{

Node\* temp = n->right;

delete n;

n = temp;

}

else if (!n->right && n->left)

{

Node\* temp = n->left;

delete n;

n = temp;

}

else if (!n->right && !n->left)

{

delete n;

n = nullptr;

}

else

{

Node\* min = n->right;

if (!min->left)

{

n->right = nullptr;

}

else

{

Node\* t = min;

while(t->left->left)

{

t = n->left;

}

min = t->left;

t->left = nullptr;

}

n->data = min->data;

delete min;

}

}

}

Node \*binTree::copy(Node \*n)

{

if (!n) return nullptr;

Node\* co = new Node();

co->data = n->data;

co->left = copy(n->left);

co->right = copy(n->right);

co->color = n->color;

return co;

}

string binTree::into\_string(Node \*n)

{

if (!n) return "";

else return "(" + to\_string(n->data) + into\_string(n->left) + into\_string(n->right) + ")";

}

# Приложение з исходный код программы. testing.cpp

#include "testing.h"

void generate\_var\_insert(string& question, string& answer, binTree &qtree, binTree &aTree, int difficulty)

{

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

{

int n = rand() % 100;

question += to\_string(n) + " ";

qtree.insert(qtree.root, n);

}

question += "\n";

aTree.root = qtree.copy(qtree.root);

int v = rand() % 100;

question += "Вставьте в данное бинарное дерево элемент " + to\_string(v) + ", выпишите полученное дерево в скобочном представлении БДП\n";

aTree.insert(aTree.root, v);

answer += aTree.into\_string(aTree.root) + "\n";

}

void generate\_vars\_insert(string& question, string& answer, int count, int difficulty)

{

for (int i = 1; i <= count; i++) {

binTree\* qt = new binTree();

binTree\* at = new binTree();

qt->root = nullptr;

at->root = nullptr;

string current = "Вариант " + to\_string(i) + ". Сложность варианта(количество элементов дерева): " + to\_string(difficulty) + "\n" + "Дерево:\n";

string one\_issue = current;

string one\_answer = current;

generate\_var\_insert(one\_issue, one\_answer, \*qt, \*at, difficulty);

question += one\_issue + "\n--------------\n";

answer += one\_answer + "\n--------------\n";

}

}

void generate\_var\_remove(string& question, string& answer, binTree &qtree, binTree &aTree, int difficulty)

{

int\* va = new int[difficulty];

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

{

int n = rand() % 100;

va[i] = n;

question += to\_string(n) + " ";

qtree.insert(qtree.root, n);

}

question += "\n";

aTree.root = qtree.copy(qtree.root);

int v = va[rand() % difficulty];

question += "Исключите из данного бинарного дерева элемент " + to\_string(v) + ", выпишите полученное дерево в скобочном представлении БДП\n";

aTree.remove(aTree.root, v);

answer += aTree.into\_string(aTree.root) + "\n";

}

void generate\_vars\_remove(string& question, string& answer, int count, int difficulty)

{

for (int i = 1; i <= count; i++) {

binTree\* qt = new binTree();

binTree\* at = new binTree();

qt->root = nullptr;

at->root = nullptr;

string current = "Вариант " + to\_string(i) + ". Сложность варианта(количество элементов дерева): " + to\_string(difficulty) + "\n" + "Дерево:\n";

string one\_issue = current;

string one\_answer = current;

generate\_var\_insert(one\_issue, one\_answer, \*qt, \*at, difficulty);

question += one\_issue + "\n--------------\n";

answer += one\_answer + "\n--------------\n";

}

}

void file\_write(string filename, string &data)

{

cout << data;

cout << &data;

ofstream myfile;

myfile.open(filename);

myfile << data;

myfile.close();

}

# приложение и 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>1050</width>

<height>747</height>

</rect>

</property>

<property name="windowTitle">

<string>MainWindow</string>

</property>

<widget class="QWidget" name="centralWidget">

<layout class="QVBoxLayout" name="verticalLayout" stretch="1,1,1,10">

<item>

<layout class="QHBoxLayout" name="horizontalLayout">

<item>

<widget class="QLabel" name="label">

<property name="text">

<string>Настройки сложности</string>

</property>

</widget>

</item>

<item>

<widget class="QPushButton" name="resetButton">

<property name="text">

<string>Сбросить</string>

</property>

</widget>

</item>

<item>

<widget class="QLabel" name="label\_2">

<property name="text">

<string>Размер дерева:</string>

</property>

</widget>

</item>

<item>

<widget class="QSpinBox" name="difficultyBox">

<property name="sizePolicy">

<sizepolicy hsizetype="Fixed" vsizetype="Fixed">

<horstretch>0</horstretch>

<verstretch>0</verstretch>

</sizepolicy>

</property>

<property name="minimum">

<number>1</number>

</property>

<property name="value">

<number>5</number>

</property>

</widget>

</item>

<item>

<widget class="QRadioButton" name="radioInsertButton">

<property name="text">

<string>Вставка</string>

</property>

<property name="checked">

<bool>true</bool>

</property>

</widget>

</item>

<item>

<widget class="QRadioButton" name="radioRemoveButton">

<property name="text">

<string>Исключение</string>

</property>

</widget>

</item>

<item>

<spacer name="horizontalSpacer">

<property name="orientation">

<enum>Qt::Horizontal</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>40</width>

<height>20</height>

</size>

</property>

</spacer>

</item>

</layout>

</item>

<item>

<layout class="QHBoxLayout" name="horizontalLayout\_2">

<item>

<widget class="QPushButton" name="varButton">

<property name="text">

<string>Генерировать варант на экран</string>

</property>

</widget>

</item>

<item>

<widget class="QPushButton" name="genFileButton">

<property name="text">

<string>Генерировать N вариантов в файл</string>

</property>

</widget>

</item>

<item>

<widget class="QLabel" name="label\_3">

<property name="sizePolicy">

<sizepolicy hsizetype="Fixed" vsizetype="Preferred">

<horstretch>0</horstretch>

<verstretch>0</verstretch>

</sizepolicy>

</property>

<property name="text">

<string>N:</string>

</property>

</widget>

</item>

<item>

<widget class="QSpinBox" name="varNum">

<property name="sizePolicy">

<sizepolicy hsizetype="Fixed" vsizetype="Fixed">

<horstretch>0</horstretch>

<verstretch>0</verstretch>

</sizepolicy>

</property>

<property name="minimum">

<number>1</number>

</property>

<property name="value">

<number>3</number>

</property>

<property name="displayIntegerBase">

<number>10</number>

</property>

</widget>

</item>

</layout>

</item>

<item>

<layout class="QHBoxLayout" name="horizontalLayout\_5">

<item>

<widget class="QLabel" name="label\_4">

<property name="text">

<string>Ввод</string>

</property>

</widget>

</item>

<item>

<widget class="QPushButton" name="inFIleButton">

<property name="text">

<string>Из файла</string>

</property>

</widget>

</item>

<item>

<widget class="QLineEdit" name="inputEdit"/>

</item>

<item>

<widget class="QPushButton" name="treeGenButton">

<property name="text">

<string>Генерировать дерево</string>

</property>

</widget>

</item>

<item>

<spacer name="horizontalSpacer\_2">

<property name="orientation">

<enum>Qt::Horizontal</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>40</width>

<height>20</height>

</size>

</property>

</spacer>

</item>

<item>

<widget class="QLineEdit" name="elementEdit">

<property name="text">

<string>10</string>

</property>

</widget>

</item>

<item>

<widget class="QPushButton" name="insertButton">

<property name="text">

<string>Вставить</string>

</property>

</widget>

</item>

<item>

<widget class="QPushButton" name="removeButton">

<property name="text">

<string>Удалить</string>

</property>

</widget>

</item>

</layout>

</item>

<item>

<widget class="QSplitter" name="splitter\_3">

<property name="orientation">

<enum>Qt::Horizontal</enum>

</property>

<widget class="QSplitter" name="splitter\_2">

<property name="orientation">

<enum>Qt::Vertical</enum>

</property>

<widget class="QTextBrowser" name="questionBrowser">

<property name="sizePolicy">

<sizepolicy hsizetype="Expanding" vsizetype="Preferred">

<horstretch>0</horstretch>

<verstretch>0</verstretch>

</sizepolicy>

</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:'MS Shell Dlg 2'; font-size:8.25pt; font-weight:400; font-style:normal;&quot;&gt;

&lt;p style=&quot; margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;&quot;&gt;&lt;span style=&quot; font-family:'MS Shell Dlg 2'; font-size:7.8pt;&quot;&gt;Задание&lt;/span&gt;&lt;/p&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; font-family:'MS Shell Dlg 2'; font-size:7.8pt;&quot;&gt;&lt;br /&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>

</property>

</widget>

<widget class="QGraphicsView" name="questionGraphicsView"/>

</widget>

<widget class="QSplitter" name="splitter">

<property name="orientation">

<enum>Qt::Vertical</enum>

</property>

<widget class="QTextBrowser" name="answerBrowser">

<property name="sizePolicy">

<sizepolicy hsizetype="Expanding" vsizetype="Preferred">

<horstretch>0</horstretch>

<verstretch>0</verstretch>

</sizepolicy>

</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:'MS Shell Dlg 2'; font-size:8.25pt; font-weight:400; font-style:normal;&quot;&gt;

&lt;p style=&quot; margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;&quot;&gt;&lt;span style=&quot; font-family:'MS Shell Dlg 2'; font-size:7.8pt;&quot;&gt;Ответ&lt;/span&gt;&lt;/p&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; font-family:'MS Shell Dlg 2'; font-size:7.8pt;&quot;&gt;&lt;br /&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>

</property>

</widget>

<widget class="QGraphicsView" name="answerGraphicsView"/>

</widget>

</widget>

</item>

</layout>

</widget>

<widget class="QMenuBar" name="menuBar">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>1050</width>

<height>21</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>

# приложение к исходный код программы. zvegintseva\_cw.pro

#-------------------------------------------------

#

# Project created by QtCreator 2019-12-15T20:46:41

#

#-------------------------------------------------

QT += core gui

greaterThan(QT\_MAJOR\_VERSION, 4): QT += widgets

TARGET = zvegintseva\_cw

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 \

bintree.cpp \

array\_list.cpp \

testing.cpp

HEADERS += \

mainwindow.h \

bintree.h \

array\_list.h \

node.h \

testing.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