Федеральное государственное автономное образовательное учреждение высшего образования
«Пермский национальный исследовательский политехнический университет»

# Лабораторная работа «Бинарное дерево»

Выполнил студент группы ИВТ-23-2Б
Муравьев Дмитрий Александрович
Проверила: доцент кафедры ИТАС
Ольга Андреева Полякова

2023

1. Постановка задачи:

1. Самостоятельно придумать вид Дерева и

Реализовать алгоритмы для этого собственного варианта бинарного дерева поиска, имеющего не менее трёх уровней.

2. Алгоритмы:

  2.1. Необходимо реализовать функции для редактирования дерева:
  - Вставка узла.
  - Удаление узла.
  - Поиск элемента по ключу.

  2.2 Реализовать алгоритмы обхода дерева:

    2.2.1 Прямой

    2.2.2 Симметричный

    2.2.3 Обратный

  2.3 Выполнить задание своего варианта из методички Laby_Chast_3.docx

3. Реализовать алгоритм балансировки дерева.

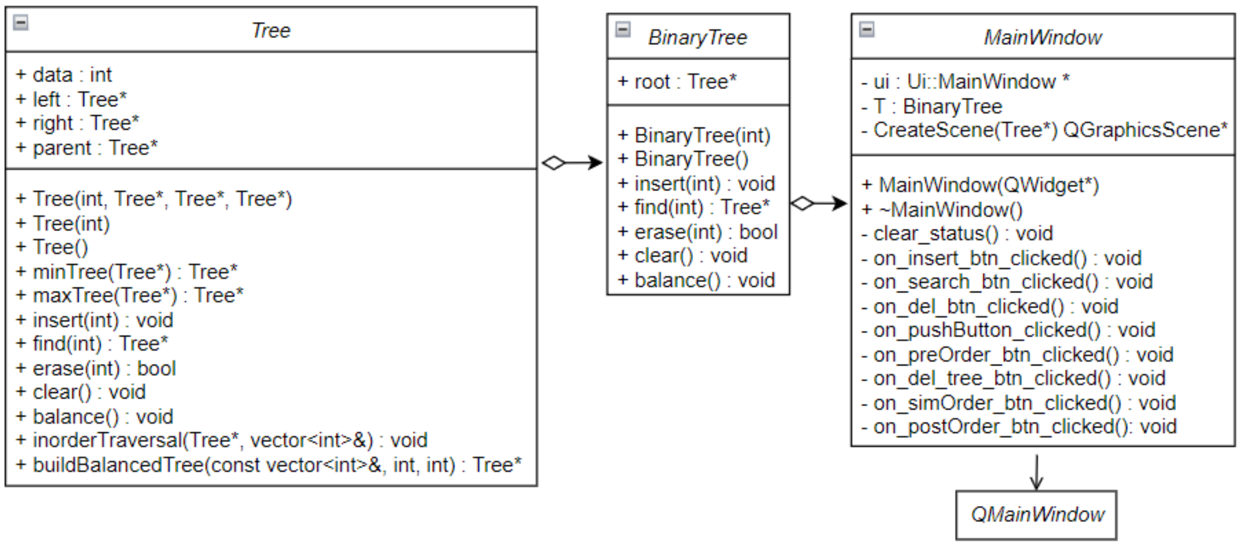
4. Реализовать вертикальную и горизонтальную печать.

5. Визуализацию дерева выполнить с использованием любой доступной графической библиотеки – SFML, SDL, OpenGL…

6. Пользовательский интерфейс – на усмотрение разработчика - с условием кроссплатформенности

7. Выполнить отчет:
- постановка задачи;
- анализ задачи с разбором применения используемых структур данных, функций;
- код программы на C++ с подробными комментариями;
- скриншоты работы программы;
- визуализация решения;
- диаграмма классов.

## 2.UML диаграмма



## 3. Код программы:

### Заголовочные файлы

BinaryTree.h

```
#pragma once
#include "Tree.h"

class BinaryTree {
public:
    Tree* root;
public:
    void setRoot(Tree* root);
    Tree* getRoot();
    BinaryTree();
    BinaryTree(int data);
    void insert(int data);
    Tree* find(int data);
    bool erase(int data);
    void balance();
    void clear();
};
```

Tree.h

```
#pragma once
#include <vector>

class Tree {
public:
    int data;
    Tree* left;
    Tree* right;
    Tree* parent;
public:
    int getData();
    void setData(int data);
    Tree* getLeft();
    void setLeft(Tree* left);
    Tree* getRight();
    void setRight(Tree* right);
    Tree* getParent();
    void setParent(Tree* parent);
    Tree();
    Tree(int data);
    Tree(int data, Tree* left, Tree* right, Tree* parent);
    Tree* minTree(Tree* tree);
    Tree* maxTree(Tree* tree);
    void insert(int data);
    Tree* find(int data);
    bool erase(int data);
    void clear();
    void balance();
    void inorderTraversal(Tree* node, std::vector<int>& values);
    Tree* buildBalancedTree(const std::vector<int>& values, int start, int
end);
};
```

mainwindow.h

```
#include <QMainWindow>
#include <QGraphicsScene>
#include "BinaryTree.h"
#include "Tree.h"
namespace Ui {
    class MainWindow;
}

class MainWindow : public QMainWindow {
    Q_OBJECT
public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();
private slots:
    void on_insert_btn_clicked();
    void on_search_btn_clicked();
    void on_del_btn_clicked();
    void on_pushButton_clicked();
    void on_preOrder_btn_clicked();
    void on_del_tree_btn_clicked();
    void on_simOrder_btn_clicked();
    void on_postOrder_btn_clicked();
private:
    Ui::MainWindow *ui;
    BinaryTree binaryTree;
    QGraphicsScene* CreateScene(Tree*);
    void clear_status();
};
```

## Исходные файлы

BinaryTree.cpp

```
#include "BinaryTree.h"
#include "Tree.h"

void BinaryTree::setRoot(Tree* root) {
    this->root = root;
}

Tree* BinaryTree::getRoot() {
    return root;
}

BinaryTree::BinaryTree() {
    root = nullptr;
}

BinaryTree::BinaryTree(int data) {
    root = new Tree(data, nullptr, nullptr, nullptr);
}

void BinaryTree::insert(int data) {
    this->root->insert(data);
}

Tree* BinaryTree::find(int data) {
    return (this->root->find(data));
}
bool BinaryTree::erase(int data) {
    return (this->root->erase(data));
}

void BinaryTree::balance() {
    if(this->root != nullptr)
        this->root->balance();
}

void BinaryTree::clear() {
    this->root->clear();
    this->root = nullptr;
}
```

Tree.cpp

```
#include "Tree.h"

int Tree::getData() {
    return data;
}

void Tree::setData(int data) {
    this->data = data;
}

Tree* Tree::getLeft() {
    return left;
}

void Tree::setLeft(Tree* left) {
    this->left = left;
}

Tree* Tree::getRight() {
    return right;
}

void Tree::setRight(Tree* right) {
    this->right = right;
}

Tree* Tree::getParent() {
    return parent;
}

void Tree::setParent(Tree* parent) {
    this->parent =parent;
}

Tree::Tree() {
    data = NULL;
    left = nullptr;
    right = nullptr;
    parent = nullptr;
}

Tree::Tree(int data) {
    this->data = data;
    this->left = nullptr;
    this->right = nullptr;
    this->parent = nullptr;
}

Tree::Tree(int data, Tree* left, Tree* right, Tree* parent) {
    this->data = data;
    this->left = left;
    this->right = right;
    this->parent = parent;
}

Tree* Tree::minTree(Tree* tree) {
    if (tree->left == nullptr) return this;
    return tree->left->minTree(tree->left);
}

Tree* Tree::maxTree(Tree* tree) {
    if (tree->right == nullptr) return this;
    return tree->right->minTree(tree->right);
}

void Tree::insert(int data) {
    Tree* temp_tree = this;
    while (temp_tree != nullptr) {
        if (data > temp_tree->data) {
            if (temp_tree->right != nullptr) {
                temp_tree = temp_tree->right;
            } else {
                Tree* tmp = new Tree(data);
                tmp->parent = temp_tree;
                temp_tree->right = tmp;
                break;
            }
        } else if (data < temp_tree->data) {
            if (temp_tree->left != nullptr) {
                temp_tree = temp_tree->left;
            } else {
                Tree* tmp = new Tree(data);
                tmp->parent = temp_tree;
                temp_tree->left = tmp;
                break;
            }
        } else {
            break;
        }
    }
}

Tree* Tree::find(int data) {
    if (this == nullptr) {
        return nullptr;
    }
    if (this->data == data) {
        return this;
    } else if (data < this->data) {
        return this->left->find(data);
    } else if (data > this->data) {
        return this->right->find(data);
    }
}

bool Tree::erase(int data){
    Tree* node = this->find(data);
    if (node == nullptr) {
        return false;
    }
    if ((node->left == nullptr) && (node->right == nullptr)) {
        Tree* node_par = node->parent;
        if (node_par->left == node) {
            node->parent->left = nullptr;
        } else {
            node->parent->right = nullptr;
        }
        delete node;
    } else if ((node->left == nullptr && node->right != nullptr) || (node-
>left != nullptr && node->right == nullptr)) {
        Tree* node_par = node->parent;
        if (node->left == nullptr) {
            if (node_par->left == node) {
                node->parent->left = node->right;
            } else {
                node->parent->right = node->right;
            }
            node->right->parent = node->parent;
        }
        else {
            if (node_par->left == node) {
                node->parent->left = node->left;
            } else {
                node->parent->right = node->left;
            }
            node->left->parent = node->parent;
        }
        delete node;
    }
    else {
        Tree* r_tree_min = node->right->minTree(node->right);
        if (r_tree_min->left == nullptr && r_tree_min->right == nullptr) {
            int tmp = r_tree_min->data;
            this->erase(r_tree_min->data);
            node->data = tmp;
        } else {
            int tmp = r_tree_min->data;
            this->erase(r_tree_min->data);
            node->data = tmp;
        }
    }
    return true;
}
void Tree::clear() {
    if (this == nullptr) {
        return;
    }
    this->left->clear();
    this->right->clear();
    delete this;
}

void Tree::balance() {
    std::vector<int> values;
    inorderTraversal(this, values);
    Tree* balancedTree = buildBalancedTree(values, 0, values.size() - 1);
    *this = *balancedTree;
}

void Tree::inorderTraversal(Tree* node, std::vector<int>& values){
    if (node == nullptr) {
        return;
    }
    inorderTraversal(node->left, values);
    values.push_back(node->data);
    inorderTraversal(node->right, values);
}

Tree* Tree::buildBalancedTree(const std::vector<int>& values, int start, int
end) {
    if (start > end) {
        return nullptr;
    }
    int mid = (start + end) / 2;
    Tree* newNode = new Tree(values[mid]);
    newNode->left = buildBalancedTree(values, start, mid - 1);
    if (newNode->left != nullptr) {
        newNode->left->parent = newNode;
    }
    newNode->right = buildBalancedTree(values, mid + 1, end);
    if (newNode->right != nullptr) {
        newNode->right->parent = newNode;
    }
    return newNode;
}
```

mainwindow.cpp

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

#include <QApplication>
#include <QGraphicsScene>
#include <QGraphicsView>
#include <QGraphicsEllipseItem>

#include "BinaryTree.h"
#include "Tree.h"

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

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

void preOrderTreeSceneCreate(Tree* tree, QGraphicsScene* Scene, int ell_r,
int lvlH, int lvlW, int lvlH_delt, int lvlW_delt, bool left, Tree* to_paint)
{
    if (tree == nullptr) {
        return;
    }
    int cur_lvlW_delt;
    if (left) {
        if (lvlW_delt < 25) {
            cur_lvlW_delt = lvlW_delt + 50;
            lvlW_delt = 25;
        } else {
            cur_lvlW_delt = lvlW_delt + 50;
        }
    } else {
        if(lvlW_delt < 25){
            cur_lvlW_delt = lvlW_delt * -1 - 50;
            lvlW_delt = 25;
        } else {
            cur_lvlW_delt = (lvlW_delt + 50) * -1;
        }
    }
    if (tree->parent != nullptr) {
        QGraphicsLineItem *edge = Scene->addLine(lvlW, lvlH, lvlW +
cur_lvlW_delt,lvlH - lvlH_delt);
        QPen PenEdge(Qt::red);  //edges color
        PenEdge.setWidth(4);
        PenEdge.setDashPattern({5, 2});
        edge->setPen(PenEdge);
    }
    preOrderTreeSceneCreate(tree->left, Scene, ell_r, lvlH + lvlH_delt, lvlW
- lvlW_delt, lvlH_delt, lvlW_delt - 50, true, to_paint);
    preOrderTreeSceneCreate(tree->right, Scene, ell_r, lvlH + lvlH_delt,
lvlW + lvlW_delt, lvlH_delt, lvlW_delt - 50, false, to_paint);
    QGraphicsEllipseItem *ELL = Scene->addEllipse(lvlW - ell_r/2, lvlH -
ell_r/2, ell_r, ell_r);
    QPen penELL(Qt::red);
    penELL.setWidth(2);
    QBrush brushELL(QColor(211, 211, 211));
    if (to_paint != nullptr) {
        if (to_paint->data == tree->data) {
            brushELL = QBrush(QColor(255, 0, 0));
        }
    }
    ELL->setPen(penELL);
    ELL->setBrush(brushELL);
    int text_delt = std::to_string(tree->data).size();
    QGraphicsTextItem *text = Scene->addText(QString::number(tree->data));
    text->setDefaultTextColor(Qt::black);
    text->setFont(QFont("Arial", 12));
    text->setPos(lvlW - 5 - 4 * text_delt, lvlH - 12);
}

QGraphicsScene* MainWindow::CreateScene(Tree* to_paint) {
    int ell_r = 40;
    int lvlH = 0;
    int lvlW = 0;
    int lvlH_delt = 80;
    int lvlW_delt = 120;
    QGraphicsScene* new_Scene = new QGraphicsScene;
    preOrderTreeSceneCreate(binaryTree.root, new_Scene, ell_r, lvlH, lvlW,
lvlH_delt, lvlW_delt, false, to_paint);
    return new_Scene;
}

void preOrderQStringCreate(Tree* tree, QString* qString) {
    if (tree == nullptr) {
        return;
    }
    *qString += QString::number(tree->data);
    *qString += " ";
    preOrderQStringCreate(tree->left, qString);
    preOrderQStringCreate(tree->right, qString);
}

void simOrderQStringCreate(Tree* tree, QString* qString) {
    if (tree == nullptr) {
        return;
    }
    simOrderQStringCreate(tree->left, qString);
    *qString += QString::number(tree->data);
    *qString += " ";
    simOrderQStringCreate(tree->right, qString);
}

void postOrderQStringCreate(Tree* tree, QString* qString) {
    if (tree == nullptr) {
        return;
    }
    postOrderQStringCreate(tree->left, qString);
    postOrderQStringCreate(tree->right, qString);
    *qString += QString::number(tree->data);
    *qString += " ";
}

void MainWindow::clear_status() {
    ui->search_status_label->setText("");
    ui->Order_result_textBrowser->setText("");
}

void MainWindow::on_insert_btn_clicked() {
    QGraphicsScene *prev_scene = ui->graphicsView->scene();
    clear_status();
    int to_add = ui->inser_textEdit->toPlainText().toInt();
    if (binaryTree.root == nullptr) {
        binaryTree.root = new Tree(to_add);
    } else {
        binaryTree.insert(to_add);
    }
    ui->inser_textEdit->setText("");
    QGraphicsScene *new_Scene = CreateScene(nullptr);
    ui->graphicsView->setScene(new_Scene);
    if (prev_scene) {
        delete prev_scene;
    };
}

void MainWindow::on_search_btn_clicked() {
    QGraphicsScene *prev_scene = ui->graphicsView->scene();
    clear_status();
    Tree* found = binaryTree.find(ui->search_textEdit-
>toPlainText().toInt());
    QGraphicsScene *new_Scene;
    if (found != nullptr) {
        //ui->search_status_label->setText("Элемент найден");
        new_Scene = CreateScene(found);
    } else {
        //ui->search_status_label->setText("Элемент не найден");
        new_Scene = CreateScene(nullptr);
    }
    ui->search_textEdit->setText("");
    ui->graphicsView->setScene(new_Scene);
    if (prev_scene) {
        delete prev_scene;
    };
}


void MainWindow::on_del_btn_clicked() {
    QGraphicsScene *prev_scene = ui->graphicsView->scene();
    clear_status();
    int to_del = ui->del_textEdit->toPlainText().toInt();
    bool isErase = binaryTree.erase(to_del);
    ui->del_textEdit->setText("");
    QGraphicsScene *new_Scene = CreateScene(nullptr);
    ui->graphicsView->setScene(new_Scene);
    if (prev_scene) {
        delete prev_scene;
    };
}

void MainWindow::on_pushButton_clicked() {
    QGraphicsScene *prev_scene = ui->graphicsView->scene();
    clear_status();
    binaryTree.balance();
    QGraphicsScene *new_Scene = CreateScene(nullptr);
    ui->graphicsView->setScene(new_Scene);
    if (prev_scene) {
        delete prev_scene;
    };
}

void MainWindow::on_preOrder_btn_clicked() {
    QGraphicsScene *prev_scene = ui->graphicsView->scene();
    clear_status();
    QString restult;
    preOrderQStringCreate(binaryTree.root, &restult);
    ui->Order_result_textBrowser->setText(restult);
    QGraphicsScene *new_Scene = CreateScene(nullptr);
    ui->graphicsView->setScene(new_Scene);
    if (prev_scene) {
        delete prev_scene;
    };
}

void MainWindow::on_del_tree_btn_clicked() {
    QGraphicsScene *prev_scene = ui->graphicsView->scene();
    clear_status();
    binaryTree.clear();
    QGraphicsScene *new_Scene = CreateScene(nullptr);
    ui->graphicsView->setScene(new_Scene);
    if (prev_scene) {
        delete prev_scene;
    };
}

void MainWindow::on_simOrder_btn_clicked() {
    QGraphicsScene *prev_scene = ui->graphicsView->scene();
    clear_status();
    QString result;
    simOrderQStringCreate(binaryTree.root, &result);
    ui->Order_result_textBrowser->setText(result);
    QGraphicsScene *new_Scene = CreateScene(nullptr);
    ui->graphicsView->setScene(new_Scene);
    if (prev_scene) {
        delete prev_scene;
    };
}

void MainWindow::on_postOrder_btn_clicked() {
    QGraphicsScene *prev_scene = ui->graphicsView->scene();
    clear_status();
    QString result;
    postOrderQStringCreate(binaryTree.root, &result);
    ui->Order_result_textBrowser->setText(result);
    QGraphicsScene *new_Scene = CreateScene(nullptr);
    ui->graphicsView->setScene(new_Scene);
    if (prev_scene) {
        delete prev_scene;
    };
}
```

main.cpp

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

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

mainwindow.ui

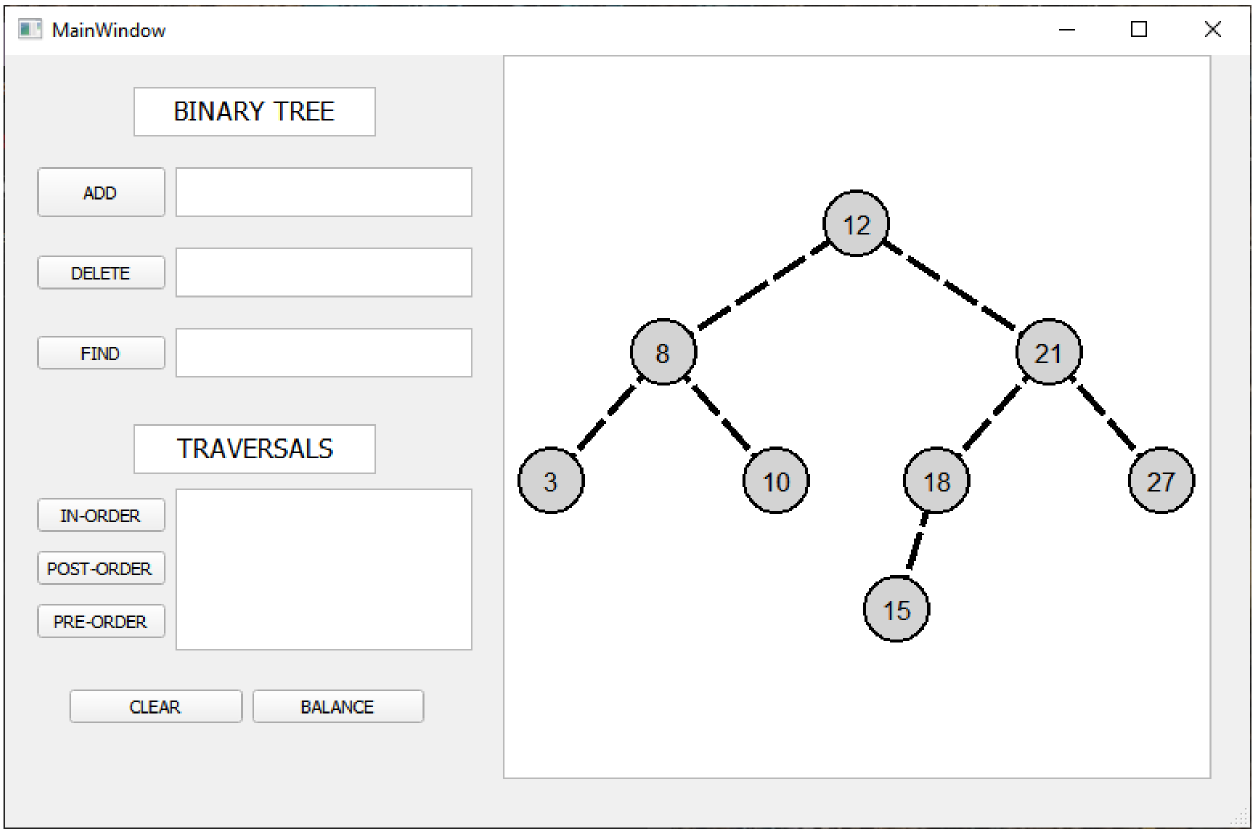
```xml
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>MainWindow</class>
 <widget class="QMainWindow" name="MainWindow">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>775</width>
    <height>481</height>
   </rect>
  </property>
  <property name="sizePolicy">
   <sizepolicy hsizetype="Minimum" vsizetype="Minimum">
    <horstretch>0</horstretch>
    <verstretch>0</verstretch>
   </sizepolicy>
  </property>
  <property name="windowTitle">
   <string>MainWindow</string>
  </property>
  <widget class="QWidget" name="centralwidget">
   <widget class="QGraphicsView" name="graphicsView">
    <property name="geometry">
     <rect>
      <x>310</x>
      <y>0</y>
      <width>441</width>
      <height>451</height>
     </rect>
    </property>
    <property name="sizePolicy">
     <sizepolicy hsizetype="Expanding" vsizetype="Expanding">
      <horstretch>0</horstretch>
      <verstretch>0</verstretch>
     </sizepolicy>
    </property>
   </widget>
   <widget class="QWidget" name="verticalLayoutWidget">
    <property name="geometry">
     <rect>
      <x>20</x>
      <y>70</y>
      <width>271</width>
      <height>194</height>
     </rect>
    </property>
    <property name="sizePolicy">
```

```
      <sizepolicy hsizetype="Preferred" vsizetype="Minimum">
       <horstretch>0</horstretch>
       <verstretch>0</verstretch>
      </sizepolicy>
     </property>
     <layout class="QHBoxLayout" name="horizontalLayout_2">
      <item>
       <widget class="QPushButton" name="insert_btn">
        <property name="sizePolicy">
         <sizepolicy hsizetype="Minimum" vsizetype="Minimum">
          <horstretch>0</horstretch>
          <verstretch>0</verstretch>
         </sizepolicy>
        </property>
        <property name="text">
         <string>ADD</string>
        </property>
       </widget>
      </item>
      <item>
       <widget class="QTextEdit" name="inser_textEdit">
        <property name="sizePolicy">
         <sizepolicy hsizetype="Expanding" vsizetype="Minimum">
          <horstretch>0</horstretch>
          <verstretch>0</verstretch>
         </sizepolicy>
        </property>
       </widget>
      </item>
     </layout>
    </widget>
    <widget class="QWidget" name="verticalLayoutWidget_2">
     <property name="geometry">
      <rect>
       <x>20</x>
       <y>120</y>
       <width>271</width>
       <height>72</height>
      </rect>
     </property>
     <layout class="QHBoxLayout" name="horizontalLayout_4">
      <item>
       <widget class="QPushButton" name="del_btn">
        <property name="text">
         <string>DELETE</string>
        </property>
       </widget>
      </item>
      <item>
       <widget class="QTextEdit" name="del_textEdit"/>
      </item>
     </layout>
    </widget>
    <widget class="QWidget" name="verticalLayoutWidget_3">
     <property name="geometry">
      <rect>
       <x>20</x>
       <y>170</y>
       <width>271</width>
       <height>72</height>
      </rect>
     </property>
     <layout class="QHBoxLayout" name="horizontalLayout_5">
```

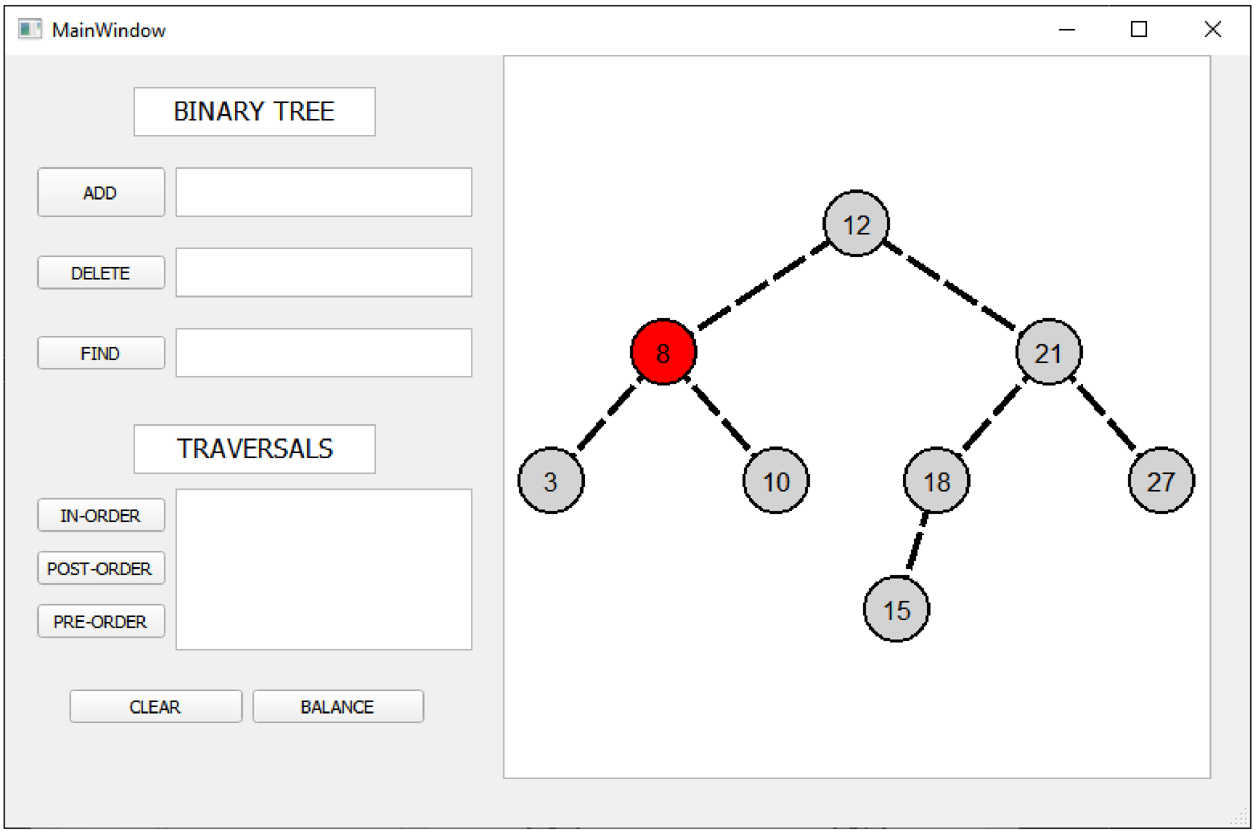
```
      <item>
       <widget class="QPushButton" name="search_btn">
        <property name="text">
         <string>FIND</string>
        </property>
       </widget>
      </item>
      <item>
       <widget class="QTextEdit" name="search_textEdit"/>
      </item>
     </layout>
    </widget>
    <widget class="QWidget" name="horizontalLayoutWidget">
     <property name="geometry">
      <rect>
       <x>260</x>
       <y>230</y>
       <width>21</width>
       <height>21</height>
      </rect>
     </property>
     <layout class="QHBoxLayout" name="horizontalLayout"/>
    </widget>
    <widget class="QWidget" name="verticalLayoutWidget_5">
     <property name="geometry">
      <rect>
       <x>40</x>
       <y>380</y>
       <width>221</width>
       <height>51</height>
      </rect>
     </property>
     <layout class="QHBoxLayout" name="horizontalLayout_3">
      <item>
       <widget class="QPushButton" name="del_tree_btn">
        <property name="text">
         <string>CLEAR</string>
        </property>
       </widget>
      </item>
      <item>
       <widget class="QPushButton" name="pushButton">
        <property name="text">
         <string>BALANCE</string>
        </property>
       </widget>
      </item>
     </layout>
    </widget>
    <widget class="QWidget" name="layoutWidget">
     <property name="geometry">
      <rect>
       <x>20</x>
       <y>270</y>
       <width>271</width>
       <height>101</height>
      </rect>
     </property>
     <layout class="QHBoxLayout" name="horizontalLayout_6">
      <item>
       <layout class="QVBoxLayout" name="verticalLayout">
        <item>
         <widget class="QPushButton" name="simOrder_btn">
          <property name="text">
           <string>IN-ORDER</string>
          </property>
         </widget>
        </item>
        <item>
         <widget class="QPushButton" name="postOrder_btn">
          <property name="text">
           <string>POST-ORDER</string>
          </property>
         </widget>
        </item>
        <item>
         <widget class="QPushButton" name="preOrder_btn">
          <property name="text">
           <string>PRE-ORDER</string>
          </property>
         </widget>
        </item>
       </layout>
      </item>
      <item>
       <widget class="QTextBrowser" name="Order_result_textBrowser"/>
      </item>
     </layout>
    </widget>
    <widget class="QTextBrowser" name="textBrowser">
     <property name="geometry">
      <rect>
       <x>80</x>
       <y>20</y>
       <width>151</width>
       <height>31</height>
      </rect>
     </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 align=&quot;center&quot; 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-size:12pt;&quot;&gt;BINARY
 TREE&lt;/span&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>
     </property>
    </widget>
    <widget class="QTextBrowser" name="textBrowser_2">
     <property name="geometry">
      <rect>
       <x>80</x>
       <y>230</y>
       <width>151</width>
       <height>31</height>
      </rect>
     </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 align=&quot;center&quot; 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-
size:12pt;&quot;&gt;TRAVERSALS&lt;/span&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&
gt;</string>
    </property>
   </widget>
   <widget class="QLabel" name="search_status_label">
    <property name="geometry">
     <rect>
      <x>30</x>
      <y>220</y>
      <width>19</width>
      <height>19</height>
     </rect>
    </property>
    <property name="text">
     <string/>
    </property>
   </widget>
  </widget>
  <widget class="QMenuBar" name="menubar">
   <property name="geometry">
    <rect>
     <x>0</x>
     <y>0</y>
     <width>775</width>
     <height>20</height>
    </rect>
   </property>
  </widget>
  <widget class="QStatusBar" name="statusbar"/>
 </widget>
 <resources/>
 <connections/>
</ui>
```

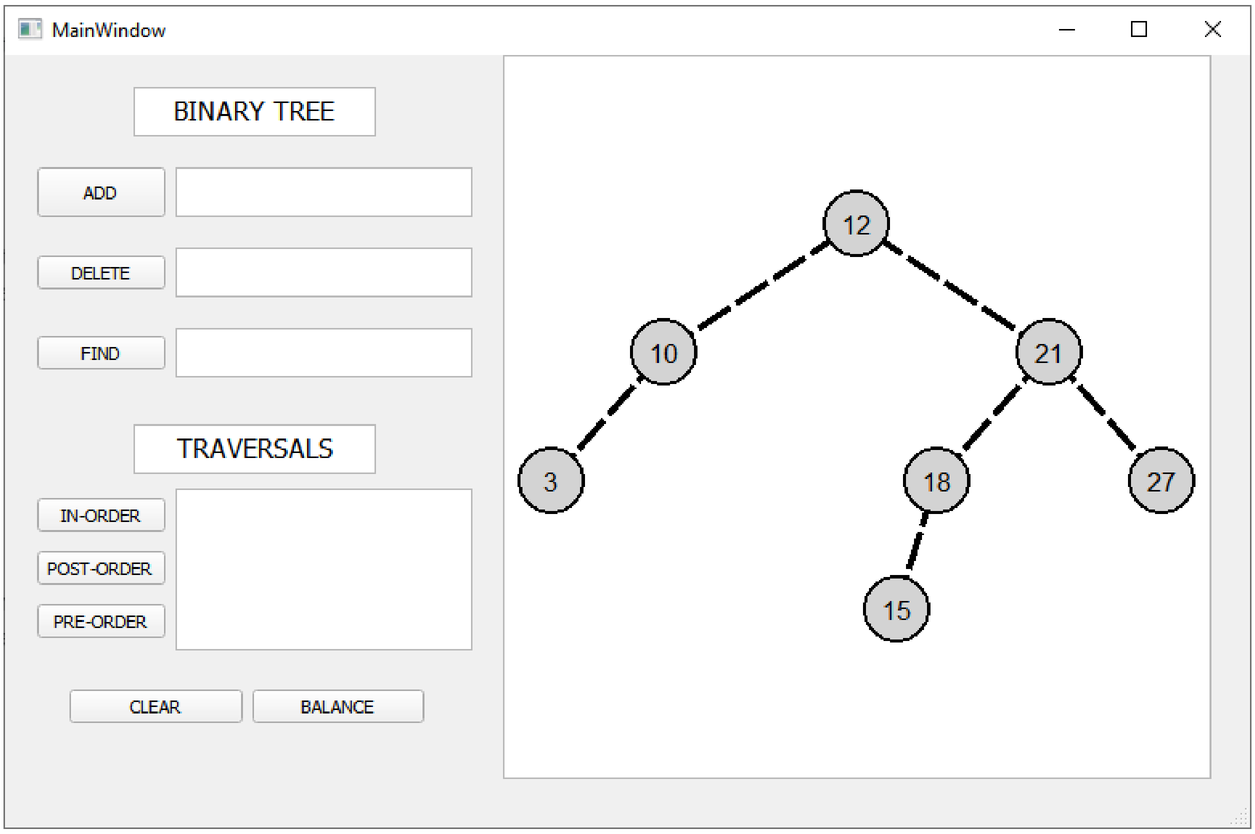
4. Работа программы:



Поиск

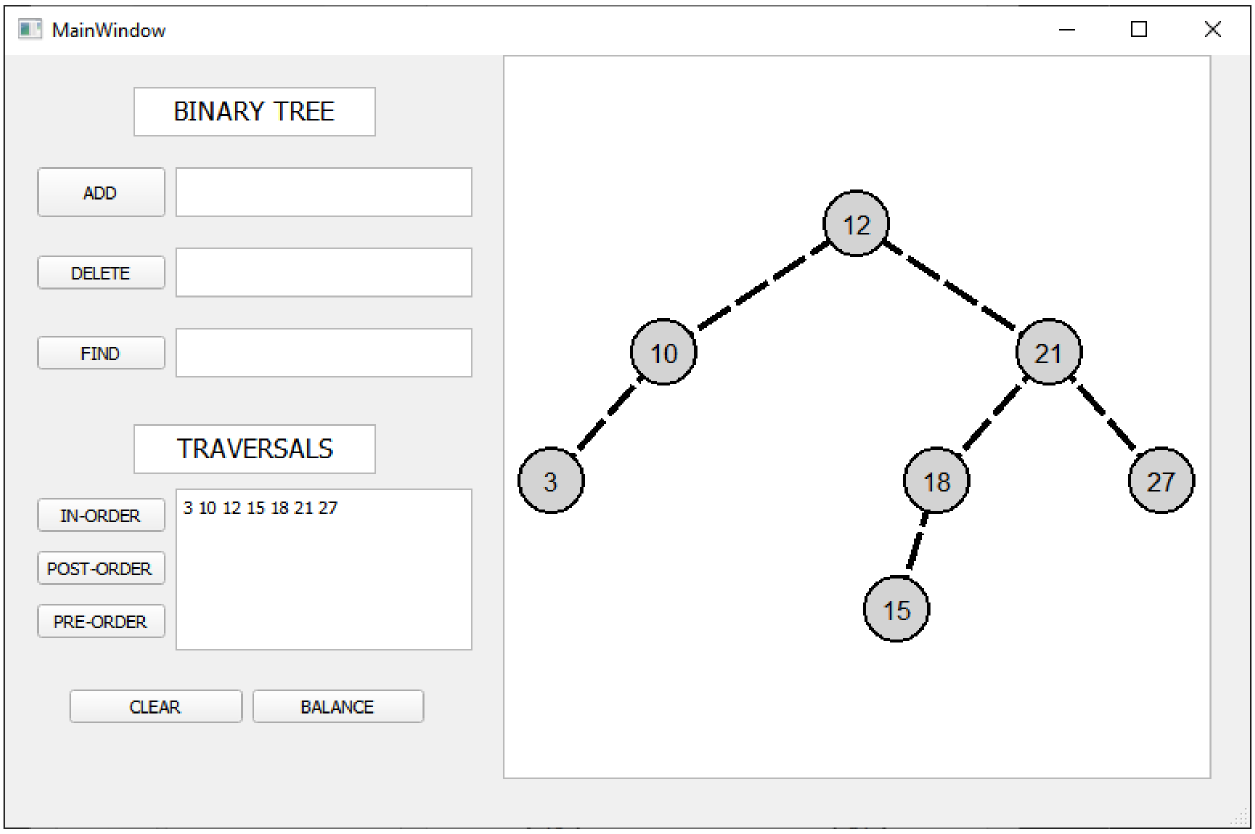


Удаление



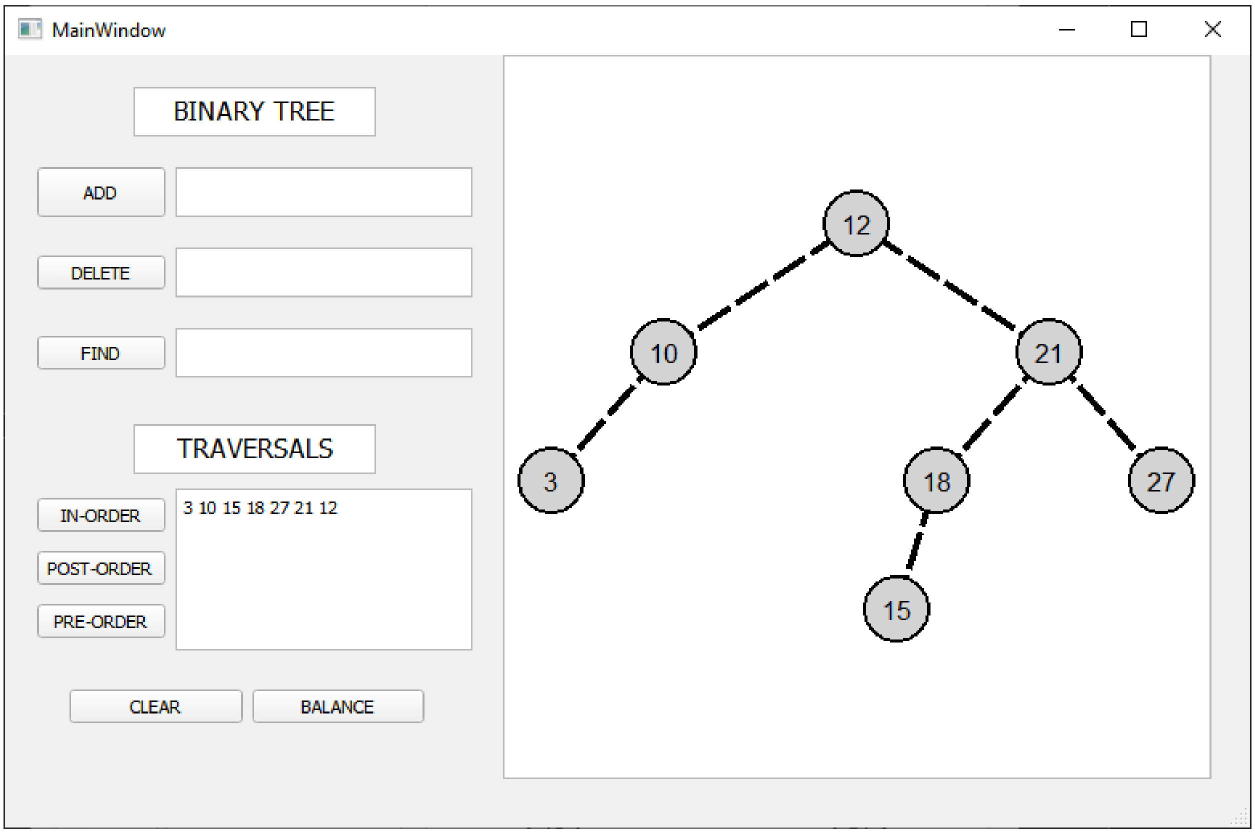
Обходы

Прямой



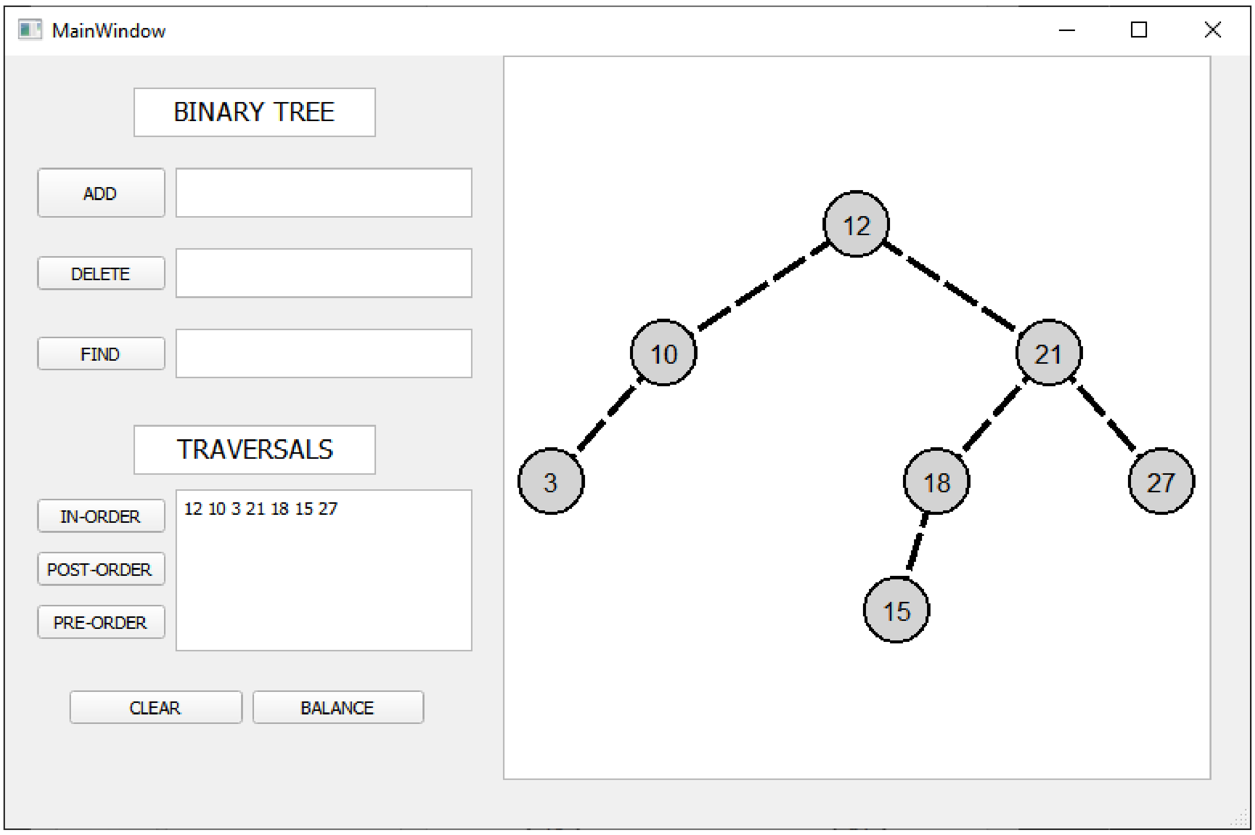
```
3 10 12 15 18 21 27
```

Обратный



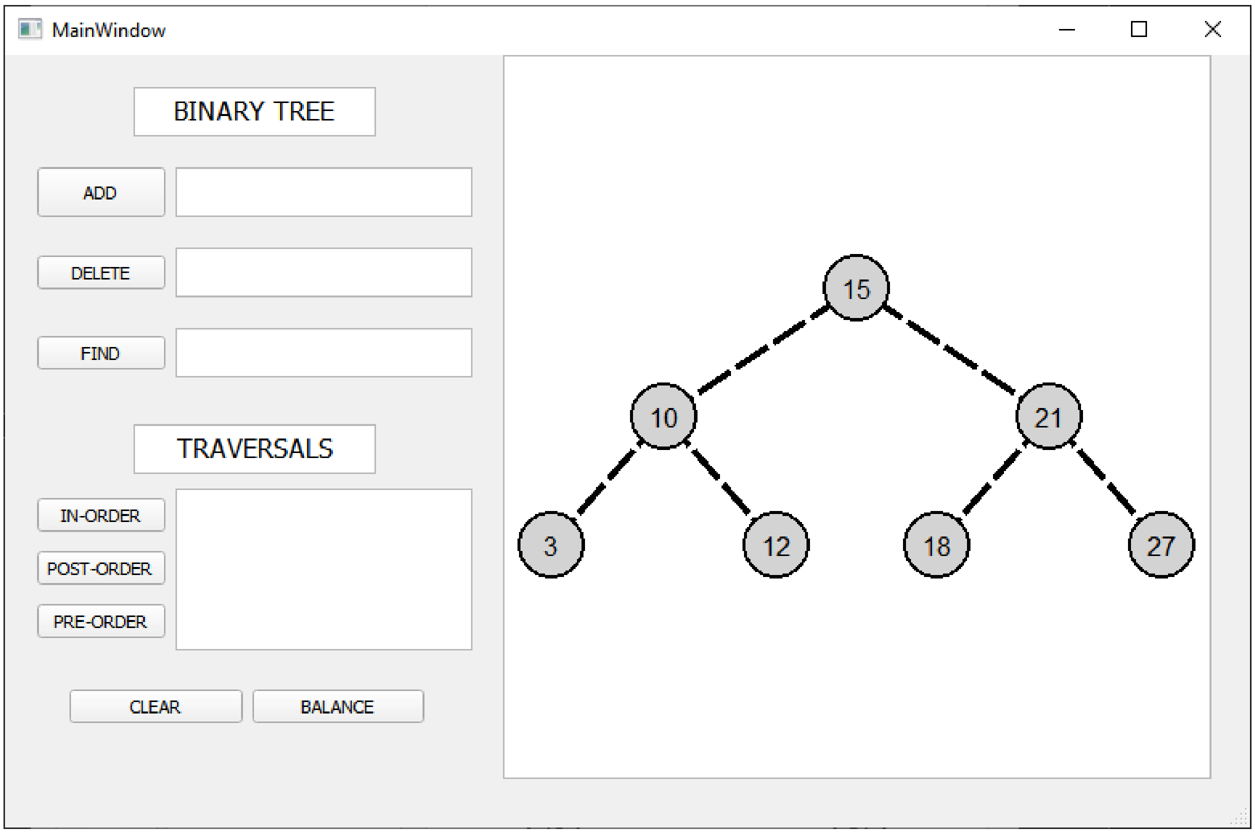
```
3 10 15 18 27 21 12
```

Симметричный



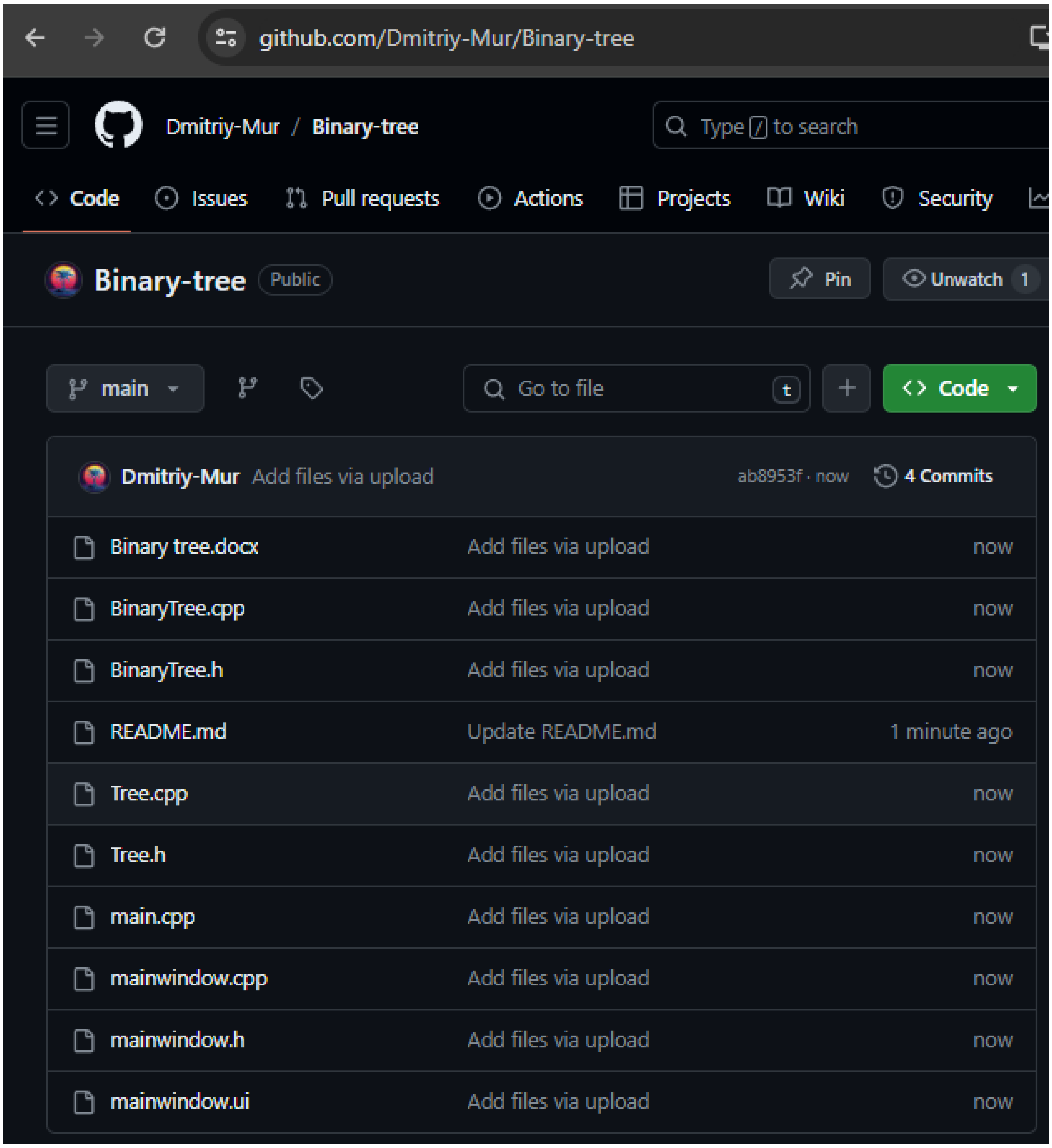
```
12 10 3 21 18 15 27
```

Балансировка



5. Github:

https://github.com/Dmitriy-Mur/Binary-tree



Dmitriy-Mur Add files via upload ab8953f · now 4 Commits

| | | |
|---|---|---|
| Binary tree.docx | Add files via upload | now |
| BinaryTree.cpp | Add files via upload | now |
| BinaryTree.h | Add files via upload | now |
| README.md | Update README.md | 1 minute ago |
| Tree.cpp | Add files via upload | now |
| Tree.h | Add files via upload | now |
| main.cpp | Add files via upload | now |
| mainwindow.cpp | Add files via upload | now |
| mainwindow.h | Add files via upload | now |
| mainwindow.ui | Add files via upload | now |