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;

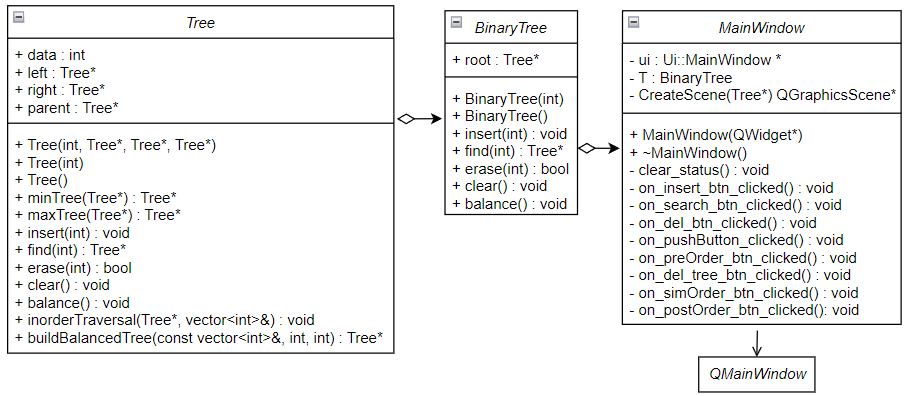
- код программы на C++ с подробными комментариями;

- 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*;  } |

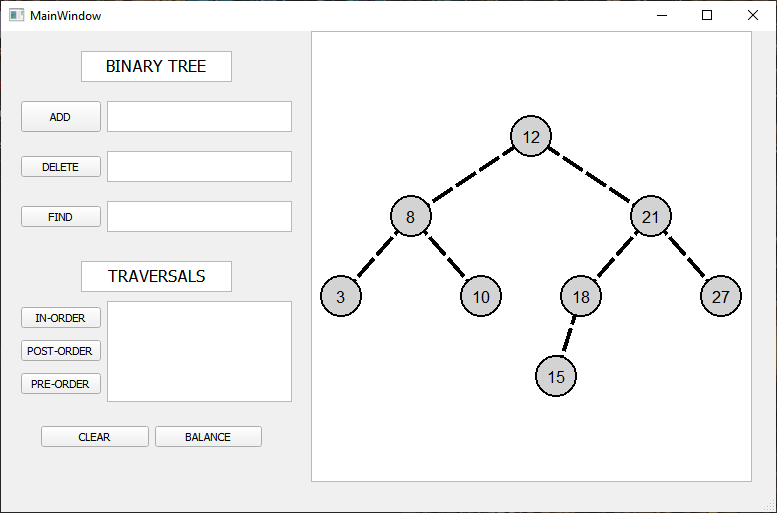
|  |
| --- |
| Tree.cpp |
| #include "Tree.h"  int Tree::**getData**() {  *return* data;  }  void Tree::**setData**(int data) {  *this*->data = data;  }  Tree\* Tree::**getLeft**() {  *return* left;  }  void Tree::**setLeft**(Tree\* left) {  *this*->left = left;  }  Tree\* Tree::**getRight**() {  *return* 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;  };  } |

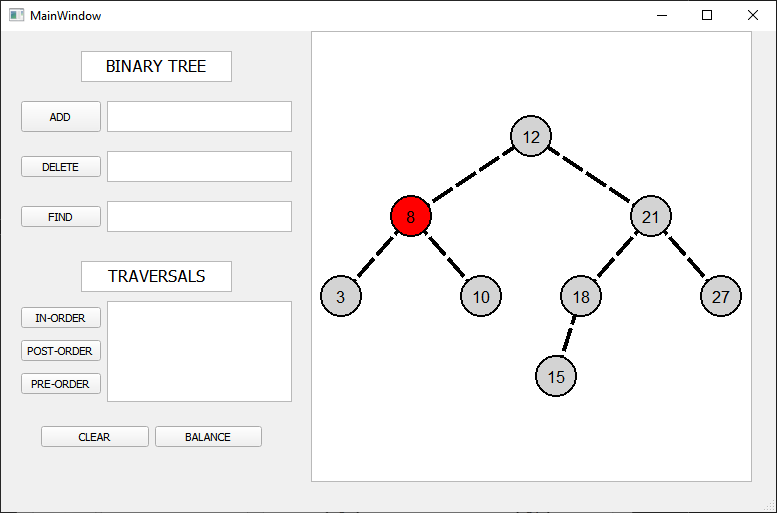
|  |
| --- |
| main.cpp |
| #include "mainwindow.h"  #include <QApplication>  int main(int argc, char \*argv[]) {  QApplication a(*argc*, *argv*);  MainWindow w;  w.show();  *return* a.exec();  } |

|  |
| --- |
| mainwindow.ui |
| <?**xml** 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*> |

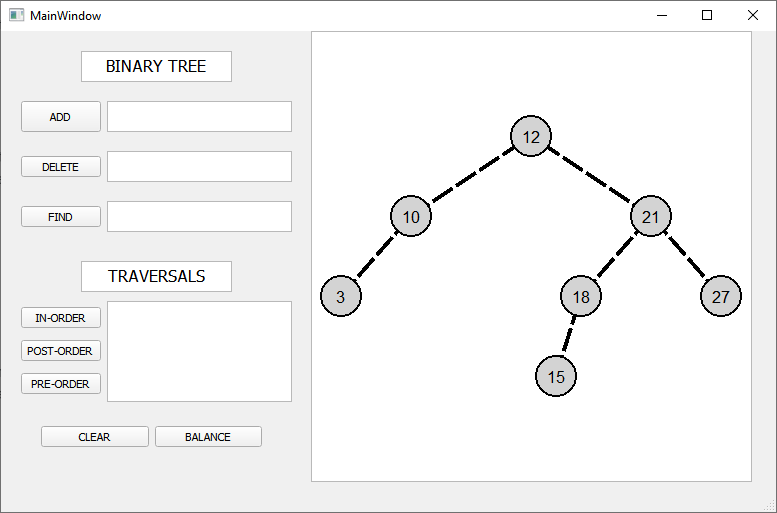
4. Работа программы:



Поиск

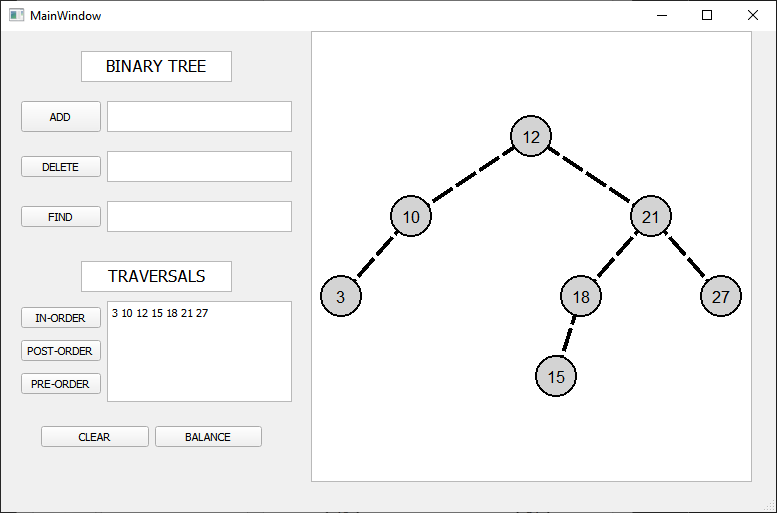


Удаление

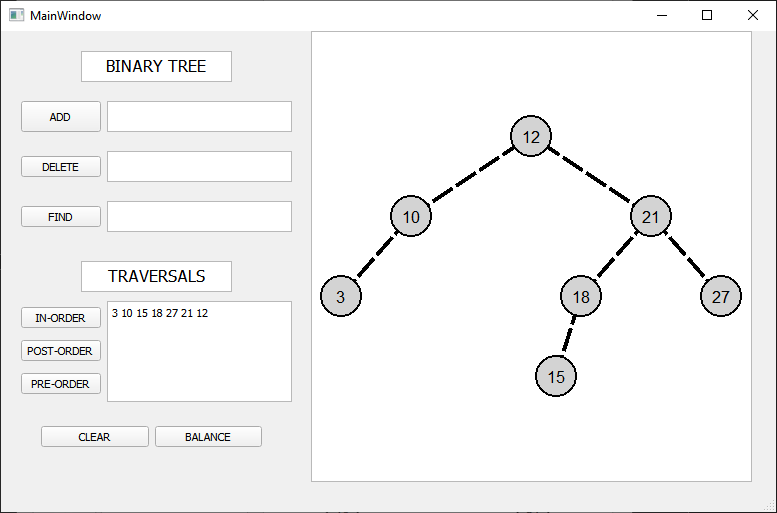


Обходы

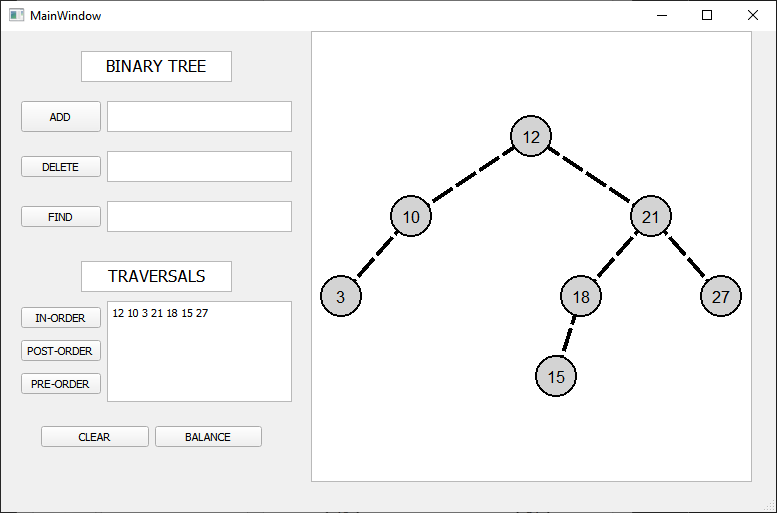
Прямой



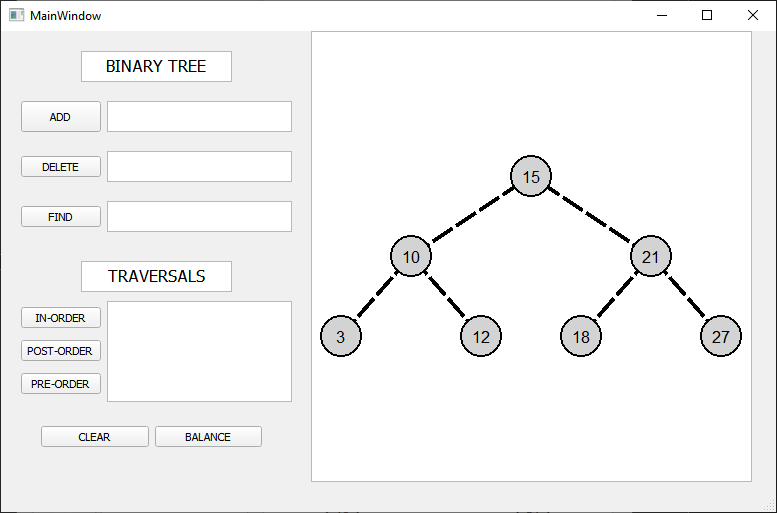
Обратный



Симметричный



Балансировка



5. Github: