Министерство высшего образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«Пермский национальный исследовательский политехнический университет» (ПНИПУ)**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

ОТЧЕТ

по лабораторной работе

Тема: «Деревья»

Выполнила

Студентка группы РИС-24-1б

Конькова С. С.

Проверил доц. Кафедры ИТАС

Полякова О.А.

Пермь 2025

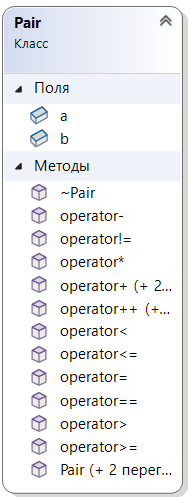
# Постановка задачи

Требуется реализовать алгоритмы для собственного варианта бинарного дерева поиска, имеющего не менее трёх уровней .  
Алгоритмы:  
1. Необходимо реализовать функции для редактирования дерева:  
- Вставка узла.  
- Удаление узла.  
- Поиск элемента по ключу.  
2. Реализовать алгоритмы обхода дерева:  
2.1 Прямой  
2.2 Симметричный  
2.3 Обратный  
3. Реализовать алгоритм балансировки дерева.  
4. Реализовать вертикальную и горизонтальную печать.  
5. Визуализация дерева должна быть выполнена с использованием любой доступной графической библиотеки – SFML, SDL, OpenGL и подобных.  
6. Пользовательский интерфейс по усмотрению разработчика – с условием кроссплатформенности (поощряется использование Qt или иных фреймворков).  
7. Выполнить отчёт

**Анализ**

1. Инициализация: создается модель данных с тестовым деревом (7 узлов). Настраивается графическая сцена с элементами дерева
2. Визуализация дерева: Создание графических элементов и позиционирование элементов в иерархической структуре
3. Интерактивные возможности: Добавление новых узлов к выбранному родителю, удаление выбранных узлов, изменение текста узлов, выполнение различных обходов дерева
4. Элементы управления: Кнопки для вызова операций, текстовое поле для вывода результатов обходов, поле ввода для изменения текста узлов
5. Узлы отображаются как круги с текстом. Связи между узлами - линии
6. Поддержка выбора и перетаскивания узлов
7. Используется QStandardItemModel для хранения иерархии. Каждый узел содержит текстовые данные

UML – диаграмма



#include <QApplication>

#include <QGraphicsView>

#include <QGraphicsScene>

#include <QStandardItemModel>

#include <QStandardItem>

#include <QGraphicsItem>

#include <QGraphicsEllipseItem>

#include <QGraphicsTextItem>

#include <QGraphicsLineItem>

#include <QPushButton>

#include <QVBoxLayout>

#include <QLineEdit>

#include <QMessageBox>

#include <QDebug>

#include <QTextEdit>

*//* *Класс* *для* *создания* *перетаскиваемых* *элементов*

*class* **DraggableItem** : *public* QGraphicsEllipseItem

{

*public*:

**DraggableItem**(QStandardItem\* item, QGraphicsItem\* parent = *nullptr*)

: QGraphicsEllipseItem(*parent*), item(item)

{

setRect(-25, -25, 50, 50);

setFlag(QGraphicsItem::*ItemIsMovable*);

setFlag(QGraphicsItem::*ItemIsSelectable*);

}

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

{

emit itemClicked(*item*);

QGraphicsEllipseItem::mousePressEvent(*event*);

}

QStandardItem\* item;

signals:

void **itemClicked**(QStandardItem\* item);

};

*//* *Обход* *дерева* *в* *прямом* *порядке*

void **preOrderTraversal**(QStandardItem\* item, QTextEdit\* output)

{

*if* (item)

{

output->append(item->text());

*for* (int i = 0; i < item->rowCount(); ++i)

{

preOrderTraversal(*item->child(i)*, *output*);

}

}

}

*//* *Обход* *дерева* *в* *симметричном* *порядке*

void **inOrderTraversal**(QStandardItem\* item, QTextEdit\* output)

{

*if* (item)

{

int middle = item->rowCount() / 2;

*for* (int i = 0; i < middle; ++i)

{

inOrderTraversal(*item->child(i)*, *output*);

}

output->append(item->text());

*for* (int i = middle; i < item->rowCount(); ++i)

{

inOrderTraversal(*item->child(i)*, *output*);

}

}

}

*//* *Обход* *дерева* *в* *обратном* *порядке*

void **postOrderTraversal**(QStandardItem\* item, QTextEdit\* output)

{

*if* (item)

{

*for* (int i = 0; i < item->rowCount(); ++i)

{

postOrderTraversal(*item->child(i)*, *output*);

}

output->append(item->text());

}

}

*//* *Функция* *для* *балансировки* *дерева*

void **balanceTree**(QStandardItem\* rootItem)

{

*//* *Здесь* *должен* *быть* *код* *для* *балансировки* *дерева,* *но* *сейчас* *уже* *поздно,* *поэтому* *я* *пойду* *спать*

}

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

{

QApplication app(*argc*, *argv*);

*//* *Создаем* *графическую* *сцену*

QGraphicsScene scene;

*//* *Создаем* *модель* *данных*

QStandardItemModel model;

*//* *Создаем* *корневой* *элемент*

QStandardItem\* rootItem = model.invisibleRootItem();

*//* *Создаем* *элементы* *дерева*

QStandardItem\* item1 = *new* QStandardItem("1");

QStandardItem\* item2 = *new* QStandardItem("2");

QStandardItem\* item3 = *new* QStandardItem("3");

QStandardItem\* item4 = *new* QStandardItem("4");

QStandardItem\* item5 = *new* QStandardItem("5");

QStandardItem\* item6 = *new* QStandardItem("6");

QStandardItem\* item7 = *new* QStandardItem("7");

*//* *Добавляем* *элементы* *в* *модель*

rootItem->appendRow(*item1*);

item1->appendRow(*item2*);

item1->appendRow(*item3*);

item2->appendRow(*item4*);

item2->appendRow(*item5*);

item3->appendRow(*item6*);

item3->appendRow(*item7*);

*//* *Создаем* *графические* *элементы* *и* *добавляем* *их* *на* *сцену*

QGraphicsEllipseItem\* rootEllipseItem = scene.addEllipse(-25, -25, 50, 50);

QGraphicsTextItem\* rootTextItem = scene.addText(item1->text());

rootTextItem->setPos(-5, -10);

QGraphicsEllipseItem\* item2EllipseItem = scene.addEllipse(-25, -25, 50, 50);

QGraphicsTextItem\* item2TextItem = scene.addText(item2->text());

item2TextItem->setPos(-105, 90);

QGraphicsEllipseItem\* item3EllipseItem = scene.addEllipse(-25, -25, 50, 50);

QGraphicsTextItem\* item3TextItem = scene.addText(item3->text());

item3TextItem->setPos(95, 90);

QGraphicsEllipseItem\* item4EllipseItem = scene.addEllipse(-25, -25, 50, 50);

QGraphicsTextItem\* item4TextItem = scene.addText(item4->text());

item4TextItem->setPos(-160, 190);

QGraphicsEllipseItem\* item5EllipseItem = scene.addEllipse(-25, -25, 50, 50);

QGraphicsTextItem\* item5TextItem = scene.addText(item5->text());

item5TextItem->setPos(-55, 190);

QGraphicsEllipseItem\* item6EllipseItem = scene.addEllipse(-25, -25, 50, 50);

QGraphicsTextItem\* item6TextItem = scene.addText(item6->text());

item6TextItem->setPos(45, 190);

QGraphicsEllipseItem\* item7EllipseItem = scene.addEllipse(-25, -25, 50, 50);

QGraphicsTextItem\* item7TextItem = scene.addText(item7->text());

item7TextItem->setPos(145, 190);

*//* *Создаем* *линии* *и* *добавляем* *их* *на* *сцену*

QGraphicsLineItem\* line1 = *new* QGraphicsLineItem();

line1->setLine(0, 0, -100, 100);

scene.addItem(*line1*);

QGraphicsLineItem\* line2 = *new* QGraphicsLineItem();

line2->setLine(0, 0, 100, 100);

scene.addItem(*line2*);

QGraphicsLineItem\* line3 = *new* QGraphicsLineItem();

line3->setLine(-100, 100, -150, 200);

scene.addItem(*line3*);

QGraphicsLineItem\* line4 = *new* QGraphicsLineItem();

line4->setLine(-100, 100, -50, 200);

scene.addItem(*line4*);

QGraphicsLineItem\* line5 = *new* QGraphicsLineItem();

line5->setLine(100, 100, 50, 200);

scene.addItem(*line5*);

QGraphicsLineItem\* line6 = *new* QGraphicsLineItem();

line6->setLine(100, 100, 150, 200);

scene.addItem(*line6*);

*//* *Устанавливаем* *позиции* *графических* *элементов*

rootEllipseItem->setPos(0, 0);

item2EllipseItem->setPos(-100, 100);

item3EllipseItem->setPos(100, 100);

item4EllipseItem->setPos(-150, 200);

item5EllipseItem->setPos(-50, 200);

item6EllipseItem->setPos(50, 200);

item7EllipseItem->setPos(150, 200);

*//* *Создаем* *представление* *и* *устанавливаем* *сцену*

QGraphicsView view(*&scene*);

*//* *Создаем* *виджеты* *для* *вывода* *результатов*

QTextEdit output;

output.setReadOnly(*true*);

*//* *Создаем* *кнопки* *и* *поле* *ввода*

QPushButton preOrderButton("Прямой обход");

QPushButton inOrderButton("Симметричный обход");

QPushButton postOrderButton("Обратный обход");

QPushButton balanceButton("Balance Tree");

QLineEdit inputField;

*//* *Создаем* *кнопку* *"Добавить* *пункт"*

QPushButton addButton("Добавить пункт");

QObject::connect(&addButton, &QPushButton::clicked, [&]() {

QStandardItem\* newItem = *new* QStandardItem("New Item");

*if* (view.scene() == &scene) {

rootItem->appendRow(*newItem*);

} *else* *if* (view.scene() && view.scene()->selectedItems().size() > 0) {

QGraphicsItem\* selectedItem = view.scene()->selectedItems().first();

*if* (selectedItem) {

QStandardItem\* parentItem = *static\_cast*<DraggableItem\*>(selectedItem)->item;

parentItem->appendRow(*newItem*);

}

}

});

*//* *Создаем* *кнопку* *"Удалить* *пункт"*

QPushButton deleteButton("Удалить пункт");

QObject::connect(&deleteButton, &QPushButton::clicked, [&]() {

*if* (view.scene() && view.scene()->selectedItems().size() > 0) {

QGraphicsItem\* selectedItem = view.scene()->selectedItems().first();

*if* (selectedItem) {

QStandardItem\* itemToRemove = *static\_cast*<DraggableItem\*>(selectedItem)->item;

QStandardItem\* parentItem = itemToRemove->parent();

*if* (parentItem) {

parentItem->removeRow(itemToRemove->row());

} *else* {

model.removeRow(itemToRemove->row());

}

}

}

});

*//* *Создаем* *строку* *ввода* *для* *изменения* *текста* *элемента*

QLineEdit textInput;

QObject::connect(&textInput, &QLineEdit::returnPressed, [&]() {

*if* (view.scene() && view.scene()->selectedItems().size() > 0) {

QGraphicsItem\* selectedItem = view.scene()->selectedItems().first();

*if* (selectedItem) {

QStandardItem\* itemToEdit = *static\_cast*<DraggableItem\*>(selectedItem)->item;

itemToEdit->setText(textInput.text());

QGraphicsTextItem\* textItem = *static\_cast*<QGraphicsTextItem\*>(selectedItem->childItems().first());

textItem->setPlainText(textInput.text());

textInput.clear();

}

}

});

*//* *Обработчик* *события* *нажатия* *на* *кнопку* *прямого* *обхода*

QObject::connect(&preOrderButton, &QPushButton::clicked, [&]() {

output.clear();

preOrderTraversal(*rootItem*, *&output*);

});

*//* *Обработчик* *события* *нажатия* *на* *кнопку* *симметричного* *обхода*

QObject::connect(&inOrderButton, &QPushButton::clicked, [&]() {

output.clear();

inOrderTraversal(*rootItem*, *&output*);

});

*//* *Обработчик* *события* *нажатия* *на* *кнопку* *обратного* *обхода*

QObject::connect(&postOrderButton, &QPushButton::clicked, [&]() {

output.clear();

postOrderTraversal(*rootItem*, *&output*);

});

*//* *Создаем* *вертикальный* *лейаут* *и* *добавляем* *в* *него* *элементы* *управления*

QVBoxLayout layout;

layout.addWidget(*&output*);

layout.addWidget(*&addButton*);

layout.addWidget(*&deleteButton*);

layout.addWidget(*&view*);

layout.addWidget(*&preOrderButton*);

layout.addWidget(*&inOrderButton*);

layout.addWidget(*&postOrderButton*);

layout.addWidget(*&balanceButton*);

*//* *Создаем* *виджет,* *устанавливаем* *лейаут* *и* *отображаем* *его*

QWidget widget;

widget.setLayout(*&layout*);

widget.show();

*return* app.exec();

}

**Результат работы**

