Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

**Пермский национальный исследовательский**

**политехнический университет**

Кафедра «Информационные технологии и автоматизированные системы»

УДК 004.4

**ОТЧЕТ**

**О НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ**

**по теме: Бинарные деревья**

Выполнил:   
студент группы РИС-23-1б   
Молодых Никита Андреевич

Проверил:   
доцент кафедры ИТАС   
Петренко А.А.

Пермь, 2024 г.

**ОГЛАВЛЕНИЕ**

**РЕФЕРАТ 3**

**ТЕРМЕНЫ И ОПРЕДЕЛЕНИЯ 3**

**1 Анализ задачи 4**

**1.1 Постоновка задачи 4  
1.2 Анализ функций 4-6**

**1.3 Алгоритмы решения 6-7**

**2 Написание кода для решения задачи 8-10**

**2.1 Визуализация решения 8-10  
ЗАКЛЮЧЕНИЕ 11  
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 12**

**РЕФЕРАТ**

**Ключевые слова**

Сбалансированное дерево, функция,

узел, прямой обход, симметричный обход , обратный обход

Исследование бинарных деревьев является актуальной задачей в области компьютерных наук. Бинарные деревья широко применяются для хранения и обработки данных, а также для решения различных задач в программировании.

Целью данного исследования является познание методики анализа и оптимизации бинарных деревьев в среде программирования C++. В процессе работы проводились экспериментальные исследования различных структур бинарных деревьев, а также анализ их производительности и эффективности.

В результате исследования были созданы оптимизированные бинарные деревья. Были разработаны алгоритмы для быстрого доступа к данным в деревьях, а также для более эффективной вставки и удаления элементов.

Основные конструктивные и технические показатели оптимизированных бинарных деревьев включают в себя увеличение скорости доступа к данным.

Задача данного исследования: реализация программы на языке программирования С++ для решения бинарных деревьев, выполнение визуализации решения.

**ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ**

**Бинарное дерево** — это конечное множество элементов, связанных с двумя разными бинарными деревьями — правым и левым поддеревьями.

OpenGL (Open Graphics Library) - это спецификация API для работы с компьютерной графикой.

**Узел (вершина)** — это каждый элемент бинарного дерева.

**Ветви** — связи между узлами.

**Глубина (высота)** — наибольший уровень какого-нибудь элемента.

**Лист (терминальный узел)** — термин применяется, если элемент не имеет потомков.

**Внутренние узлы** — узлы ветвления.

**Степень внутреннего узла** — число его потомков.

**Длина пути к X** — количество ветвей, которые необходимо пройти от корня до текущего узла X.

**1 Анализ задачи**

**1.1 Постановка задачи**

1. Сформировать идеально сбалансированное бинарное дерево, тип

информационного поля указан в варианте.

2. Распечатать полученное дерево.

3. Выполнить обработку дерева в соответствии с заданием, вывести полученный результат.

4. Преобразовать идеально сбалансированное дерево в дерево поиска.

5. Распечатать полученное дерево.

**Вариант 16:**

Тип информационного поля int. Найти максимальный элемент в дереве.

**1.2** **Анализ функций**

insert(T data): Эта функция вставляет новый узел со значением data в нужное место в дереве, с учетом порядка значений. Она выполняет поиск места для вставки нового узла и вызывает функции insert\_right и insert\_left для добавления нового узла в правое или левое поддерево соответственно.

insert\_right(T data): Эта функция добавляет новый узел со значением data в правое поддерево текущего узла.

insert\_left(T data): Эта функция добавляет новый узел со значением data в левое поддерево текущего узла.

delete\_tree(): Эта функция удаляет полностью дерево, но имеет ошибку в реализации. Удаление должно быть выполнено правильно для всех узлов дерева.

get\_data(), get\_right(), get\_left(), get\_parent(): Эти функции возвращают значение в узле, указатель на правое поддерево, указатель на левое поддерево и указатель на родительский узел соответственно.

erase(T data): Эта функция удаляет узел с заданным значением, ищет узел по значению, а затем выполняет операцию удаления в зависимости от типа удаляемого узла (узел без потомков, узел с одним потомком, узел с двумя потомками).

delete\_left(), delete\_right(): Эти функции удаляют левое и правое поддерево текущего узла соответственно.

add\_right(Tree<T> \*temp), add\_left(Tree<T> \*temp): Эти функции устанавливают правое и левое поддерево текущего узла соответственно.

search(T key): Эта функция ищет узел по заданному ключу и возвращает его.

find(T data): Эта функция ищет узел с заданным значением в дереве и возвращает его.

direct\_way(Tree<T> \*current), symmetric\_way(Tree<T> \*tree), reverse\_way(Tree<T> \*tree): Эти функции выполняют прямой, симметричный и обратный обход дерева соответственно.

balanced(intcount): Эта функция создает сбалансированное дерево с заданной высотой, запрашивая данные от пользователя истроит дерево рекурсивно.

getHeight(), getAmountOfNodes(): Эти функции возвращают высоту дерева и количество узлов в дереве соответственно

obh(Tree<T> \*node), printVert(): Эти функции печатают дерево вертикально и используют вспомогательный файл print.txt для хранения данных об узлах.

print\_horizontal(intdepth = 0, charbranch = ' '): Эта функция печатает дерево горизонтально, обозначая ветви узлов.

build\_bst(constvector<T>data, intstart, intend): Эта функция рекурсивно строит сбалансированное дерево по заданным данным вектора.

**1.3 Алгоритмы решения**

Существует несколько способов решения бинарных деревьев, каждый из которых имеет свои плюсы и минусы.

**Обход в глубину** (Depth-First Search, DFS):

Преимущества:

-Прост в реализации

-Эффективен для поиска в глубину, обновления и удаления узлов

-Может быть рекурсивным

Недостатки:

-Может потреблять больше памяти из-за стека вызовов

-Не всегда эффективен для поиска в ширину

**Обход в ширину** (Breadth-First Search, BFS):

Преимущества:

Эффективен для поиска в ширину

-Может быть реализован с использованием очереди

-Меньше вероятности переполнения стека вызовов

Недостатки:

-Сложнее реализовать, чем DFS

-Может быть менее эффективным для обновления и удаления узлов

**Рекурсивный способ обхода**:

Преимущества:

-Прост и понятен

-Может быть эффективным для поиска и некоторых операций обновления узлов

Недостатки:

-Может привести к переполнению стека вызовов при работе с большими деревьями

-Может быть менее эффективным для некоторых типов операций

Каждый из этих способов имеет свои сильные и слабые стороны, и выбор конкретного метода зависит от требований задачи.

**2 Написание кода для решения задачи**

Бинарные деревья - это структуры данных, которые играют важную роль в информатике и программировании. Они широко используются для решения различных задач, таких как хранение и поиск данных, обход дерева, сортировка и др. В данном отчете рассмотрены основные понятия и методы работы с бинарными деревьями. Результаты исследования позволят более глубоко понять принципы построения и использования бинарных деревьев и оптимизировать их применение в практических задачах.

С кодом программы на языке программирования С++ можно ознакомиться в приложении А

**2.1 Визуализация решения**

OpenGL (Open Graphics Library) - это спецификация API для работы с компьютерной графикой. Он предоставляет разработчикам возможность создавать высокопроизводительные и кроссплатформенные приложения, использующие 2D и 3D графику. Важность OpenGL заключается в том, что он является стандартом в области компьютерной графики и позволяет разработчикам быстро создавать высококачественные визуальные приложения для различных платформ. Он обеспечивает доступ к аппаратному ускорению графики и оптимизированным возможностям рендеринга, что делает приложения быстрыми и эффективными.

<https://www.rsdn.org/article/opengl/ogltutor.xml> [4]

С кодом программы на языке программирования С++ можно ознакомиться в приложении А

С кодом программы для визуализации в OpenGL можно ознакомиться в приложении Б

По коду программы (приложение А) была составлена UML таблица

(Рисунок 1 – UML)

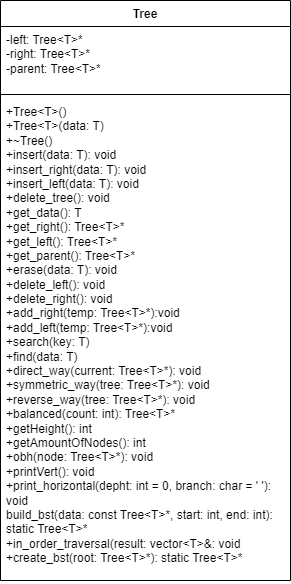


Рисунок 1 - UML

По коду программы была выполнена графическая реализация при помощи Open GL (Рисунок 2 - OpenGL)

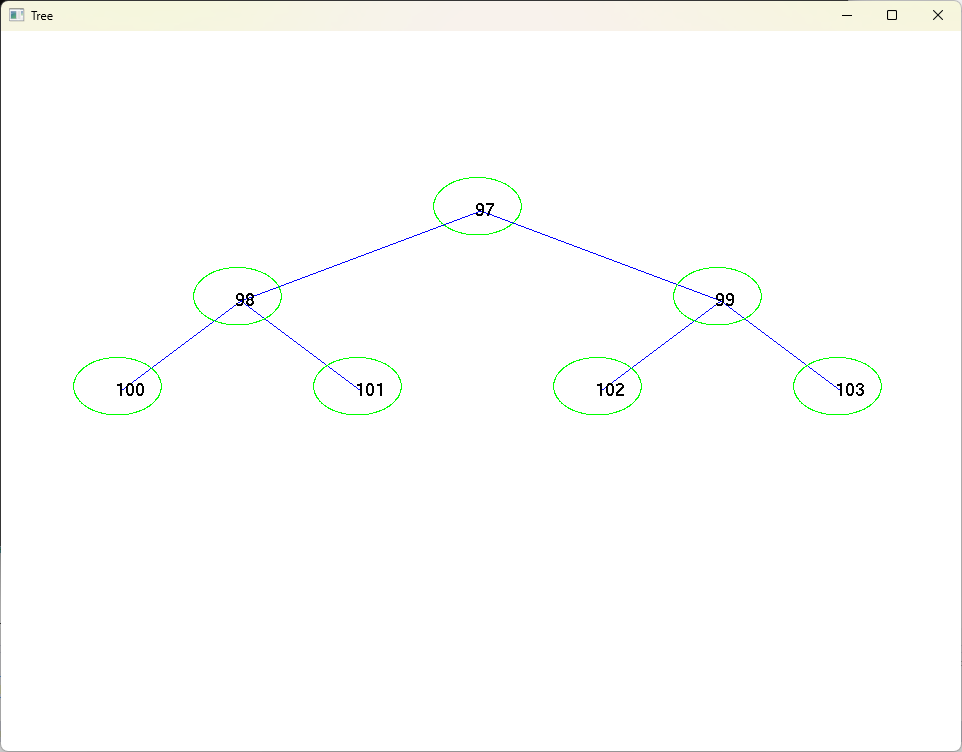


Рисунок 2 – OpenGL

В качестве подтверждения сохранения кода на языке программирования C++, также прилагается скриншот из веб-сервиса GitHub (Рисунок 3 - GIT).

<https://github.com/MolodyhN/Proba/tree/2-Семестр> [2]

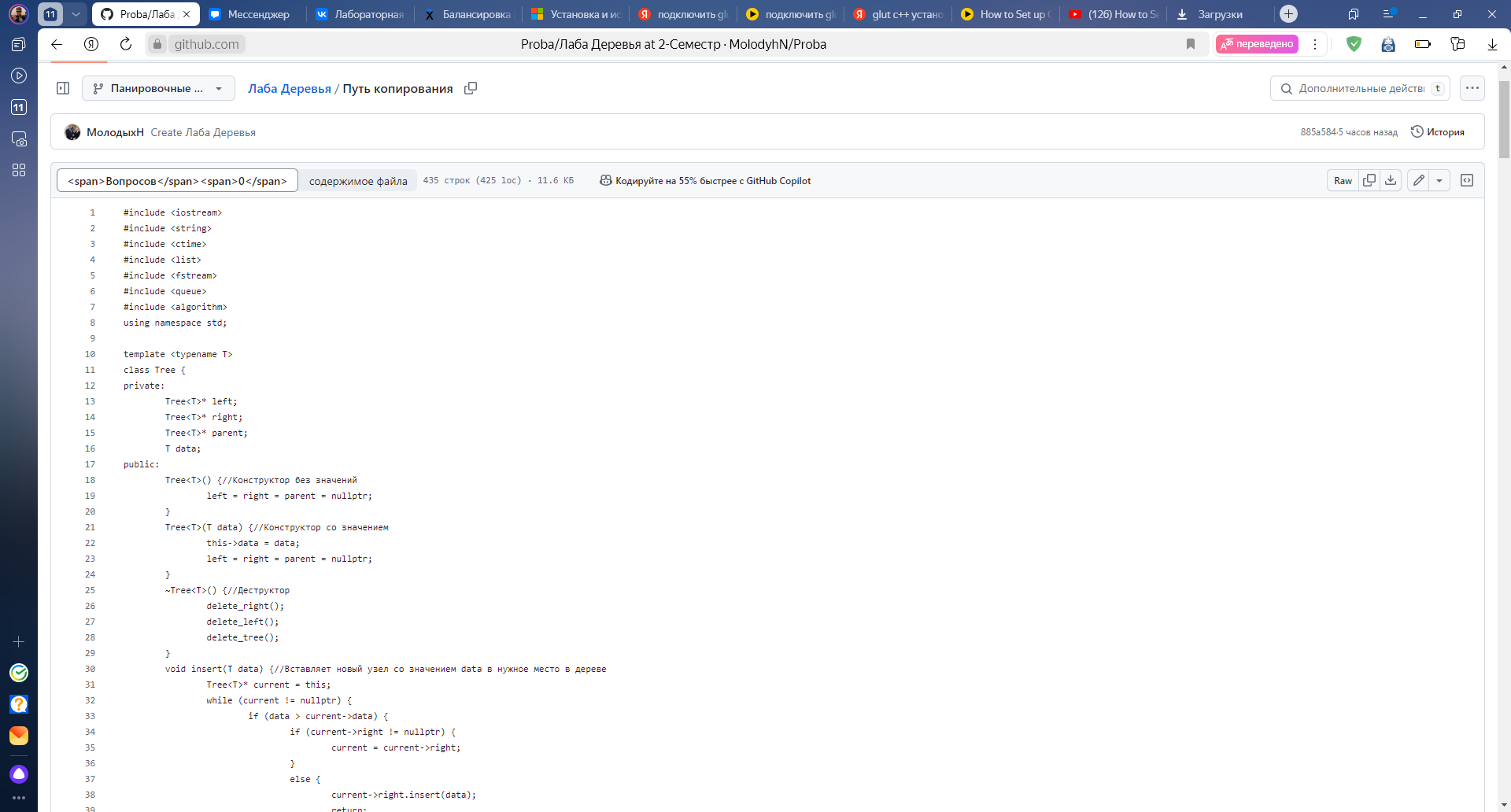


Рисунок 3 – GIT

**ЗАКЛЮЧЕНИЕ**

В ходе выполнения научно-исследовательской работы были изучены основные концепции и принципы бинарных деревьев, их применение в различных областях информатики и вычислительной математики. Было проведено исследование существующих методов объединения и сравнения бинарных деревьев, а также разработаны новые подходы к оптимизации их работы. Полученные результаты могут быть использованы для дальнейших исследований в области алгоритмов обработки данных и оптимизации вычислений. Работа позволила расширить наши знания о бинарных деревьях и их применении, а также предложить новые пути для развития данной темы. В целом, выполнение НИР позволило получить ценные научные результаты и расширить наше понимание этой важной области информатики.

Отчет был оформлен по ГОСТ 7.32.

https://www.hse.ru/data/2020/10/06/1370744192/ГОСТ\_7\_32\_2017\_Отчёт\_по\_НИР\_с\_выделением.pdf. [2]

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. <https://github.com/MolodyhN/Proba/tree/2-Семестр>
2. <https://www.hse.ru/data/2020/10/06/1370744192/ГОСТ_7_32_2017_Отчет_по_НИР_с_выделением.pdf>
3. Бартов Е.И., Ерохин Н.В., Торопицын М.С. Структурное и объектно-ориентированное программирование на С++ // Лабораторный практикум // Часть III // Представление графических объектов и проектирование программ на С++.

4. <https://www.rsdn.org/article/opengl/ogltutor.xml>