****

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«МИРЭА – Российский технологический университет»**

# РТУ МИРЭА

Отчет по выполнению практического задания №7

**Тема: «НЕЛИНЕЙНЫЕ СТРУКТУРЫ»**

Дисциплина: Структуры и алгоритмы обработки данных

Выполнил студент Кузнецов Л. А.

группа ИКБО-20-23

**Москва 2024СОДЕРЖАНИЕ**

[ЧАСТЬ 7.1…………………………………………………………………………3](#a7)

[Условие………………………………………………………………………….....3](#a1)

[Вариант………………………………………………………………………….....3](#a2)

[Метод решения……………………………………………………………………3](#a3)

[Тестирование……………………………………………………………………..11](#a4)

[ВОВОДЫ…………………………………………………………………………13](#a5)

[СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ….……………………….13](#a6)

**ЧАСТЬ 7.1**

**Условие**

Составить программу создания двоичного дерева поиска и реализовать процедуры для работы с деревом согласно варианту.

**Вариант**

В данной работе представлен вариант 18, в котором требуется работать с АВЛ-деревом. По мере выполнения работы мне необходимо написать следующие функции: вставка элемента и балансировка дерева, симметричный обход, среднее арифметическое всех узлов.

**Метод решения**

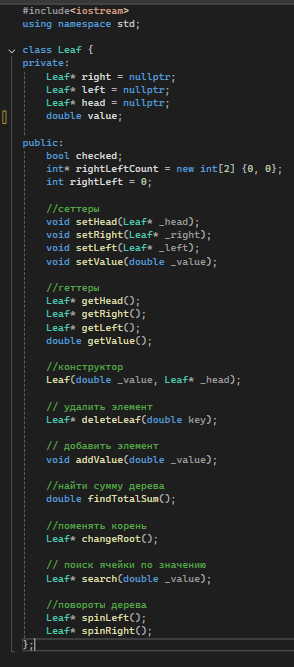
Для начала стоит реализовать само АВЛ-дерево, главное правило которого гласит – все элементы справа от узла больше значения текущего листа, а слева – меньше. На рис.1 представлена реализация данного дерева ср всеми необходимыми методами и полями.

Теперь же стоит заняться описанием действия каждого метода, использованного в текущей работе. Начнём, пожалуй, с сеттеров (рис. 2). На данном рисунке всё очевидно: сеттеры возвращают значения, прописанные в их названии.

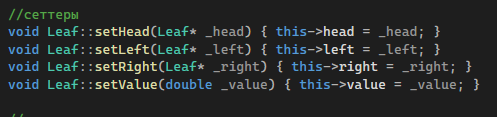
Далее затронем геттеры (рис.3). С ними всё аналогично – возвращают искомое, а зачем они нужны? Геттеры, как и сеттеры, нужны для безопасности элементов, чьё непосредственное изменение может быть губительно для структуры программы: ошибка в этих полях способна нарушить работу всей программы, поэтому данные меры предосторожности чрезмерно желательны к исполнению.

Стоит обратиться и к ядру нашего класса, а именно к конструктору (рис. 4). Он также не примечателен, ведь для создания объекта класса нашего дерева не нужно мудрить и делать лишних действий.

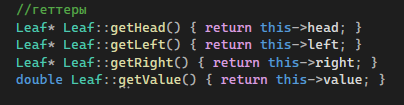
Рассмотрим метод поиска ячейки по значению (рис. 5). Подаём искомое значение и тривиально проходимся по элементам соответственно следующей методике: искомое значение больше текущего? Нет? Тогда идём налево. Меньше или равно? Нет? Тогда идём направо. При не нахождении элемента будем оповещать об этом пользователя, но это уже позже.



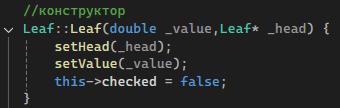
**Рисунок 1 – Вид АВЛ-дерева в коде**

****

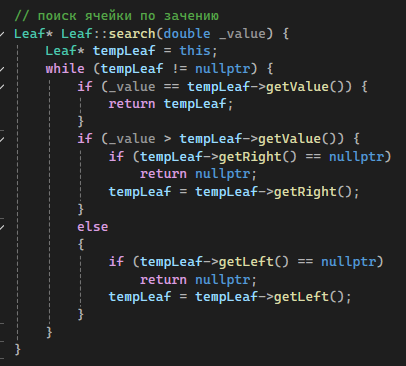
**Рисунок 2 – Сеттеры**

****

**Рисунок 3 – Геттеры**

****

**Рисунок 4 – Конструктор**

****

**Рисунок 5 – Метод поиска ячейки по значению**

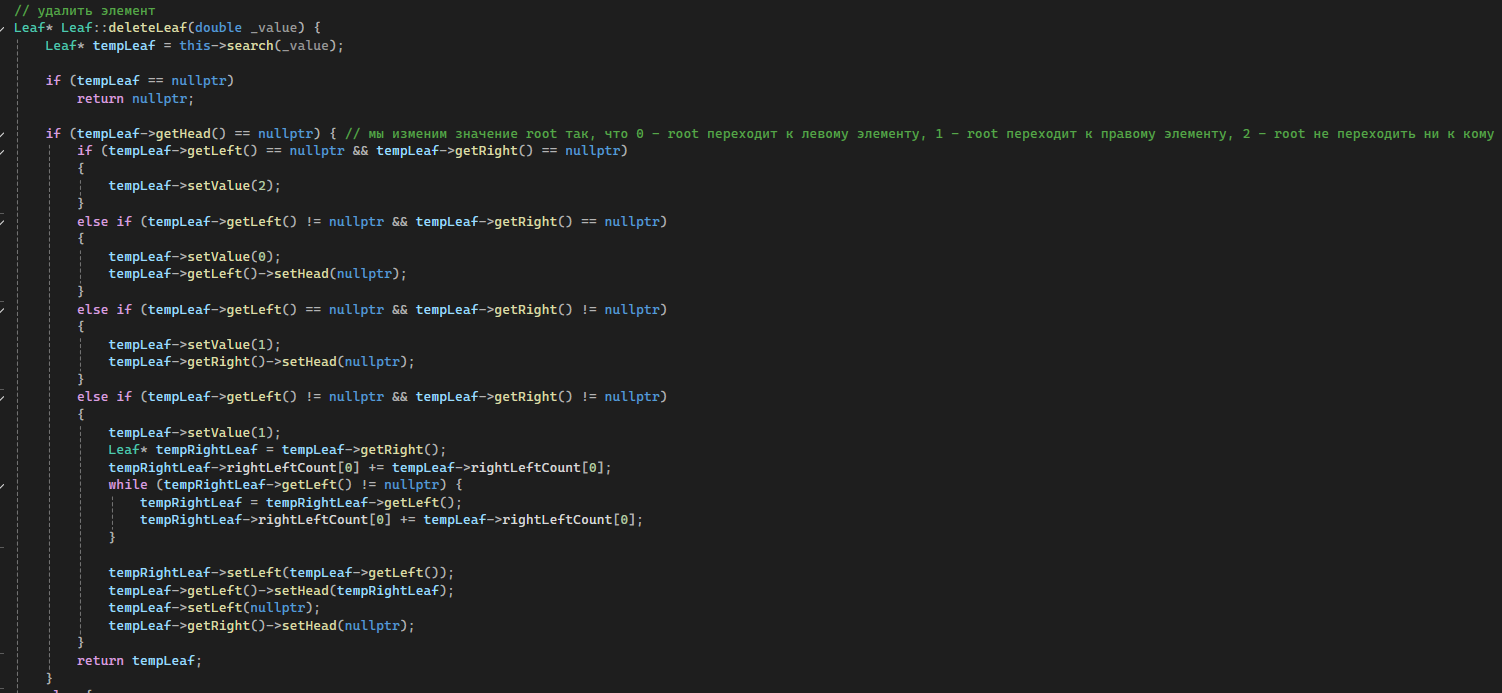
На рис. 6 представлена работа метода по добавлению элемента в наше дерево. Нахождение свободной ячейки аналогично поиску ячейки по значению, однако же в данном случае на пустое место добавляется поданное значение.

Разберём метод удаления элемента (рис. 7-8). Я решил, что мы будем способны удалить даже корень, поэтому данный метод разделён на две части.

Возможны 4 различные ситуации, от которых будет зависеть поведение нашей программы, что и показано в нашем коде за счёт установления значения ячейки при помощи сеттера.

****

**Рисунок 6 – Метод добавления элемента в дерево**

****

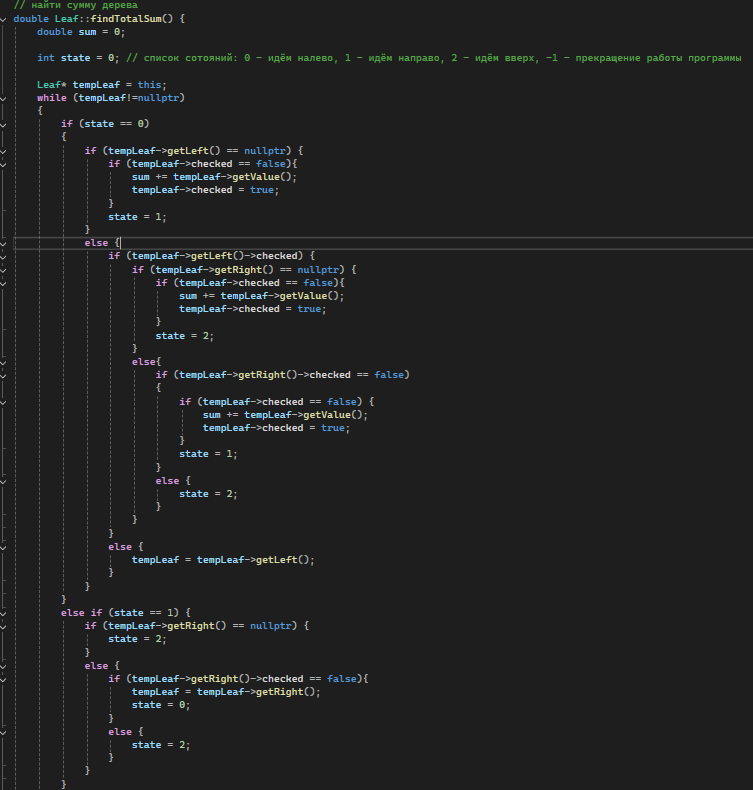
**Рисунок 7 – 1-ая часть метода удаления элемента по значению**

Во второй же части представлен общий случай, когда происходит удаление листа дерева, что провоцирует сортировку дочерних элементов данного четырьмя различными способами.

****

**Рисунок 8 - 2-ая часть метода удаления элемента по значению**

Далее идёт метод нахождения суммы дерева симметричным обходом (рис. 9-10). Да уж, это был самый трудно реализуемый метод, который будет представлен в данной работе, так как пришлось метаться между рекурсией и обходом поэлементно при определённых условиях. Я выбрал второй метод, так как при работе с суммой рекурсия довольно нестабильна.



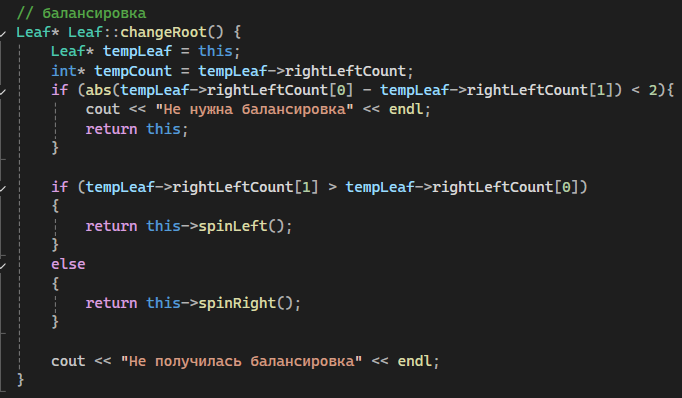
**Рисунок 9 – 1-ая часть метода нахождения суммы дерева**

Пришлось обработать множество различных ситуаций и предотвратить всевозможные ошибки программы, однако, несмотря на все трудности, желаемый результат был достигнут. И я несомненно доволен этим.



**Рисунок 10 - 2-ая часть метода нахождения суммы дерева**

Одним из последних методов является метод балансировки дерева (рис. 11). Он невелик из-за того, что полагается на прочие методы, такие как поворот налево (рис.12) и поворот направо (рис.13).

****

**Рисунок 11 – Метод балансировки дерева**

****

**Рисунок 12 – Метод поворота дерева налево**

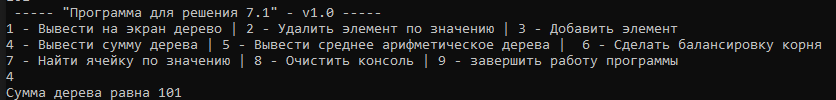
****

**Рисунок 13 – Метод поворота дерева направо**

**Тестирование**

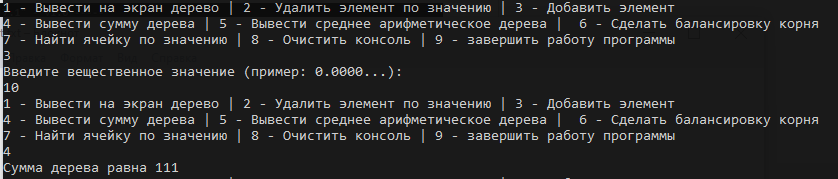
Ниже будут представлены результаты тестирования различных методов на дереве, к которому будут представлены элементы 8.8, 10.2, 7.5, 7.3, 7.8, 9, 9.5, 10.5, 7.6, 7.55, 7.68, 7.57 соответственно.

На рис. 14 совершенно верно представлена сумма нашего дерева, пройдёмся же и по прочим методам.



**Рисунок 14 – Результат работы метода поиска суммы**

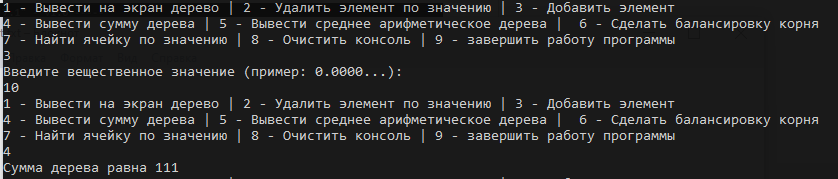
Добавим-ка к нашему дереву новый элемент 10 и проверим результат работы метода методом поиска суммы.



**Рисунок 15 – Результат работы метода добавления элемента**

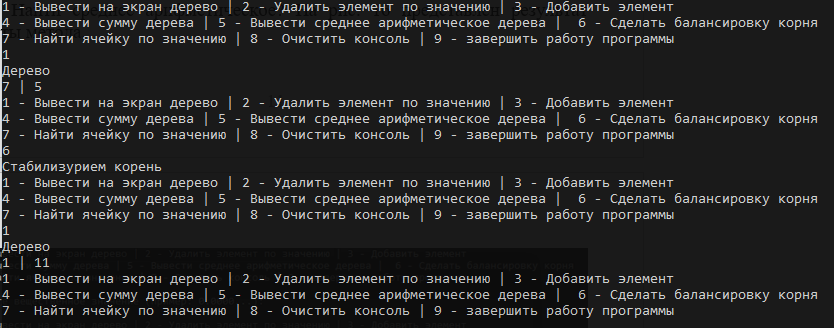
Всё работает. Просто отлично!

Найти среднее арифметическое? На рис. 16 представлен результат работы метода.



**Рисунок 16 – Результат работы метода нахождения среднего арифметического**

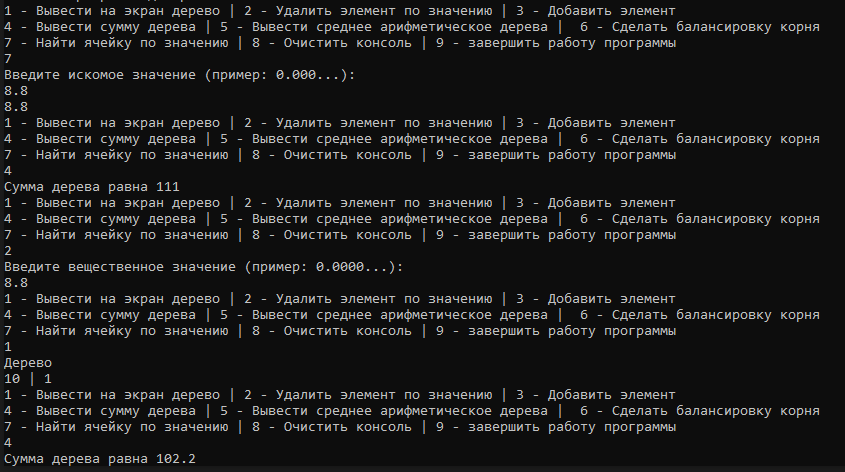
Замечательно. Теперь же сделаем балансировку корня



**Рисунок 16 – Результат работы метода балансировки корня**

Да уж, дерево сильно изменилось, но порой приходится несколько раз балансировать дерево, чтобы прийти к желаемому результату.

Найдём ячейку 8.8, а после удалим!



**Рисунок 17 – Результаты работ методов поиска и удаления элементов**

**ВЫВОДЫ**

Изучили различные виды деревьев и метода работы с ними, а также разработали программу взаимодействия с элементами этих деревьев.

**СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ**

1. Страуструп Б. Программирование. Принципы и практика с использованием C++. 2-е изд., 2016.

2. Документация по языку С++ [Электронный ресурс]. URL: https://docs.microsoft.com/ruru/cpp/cpp/ (дата обращения 01.09.2021).

3. Курс: Структуры и алгоритмы обработки данных. Часть 2 [Электронный ресурс]. URL: https://online-edu.mirea.ru/course/view.php?id=4020 (дата обращения 01.09.2021).