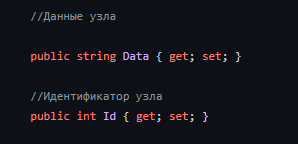
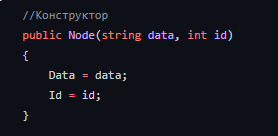
**Класс Node.cs**

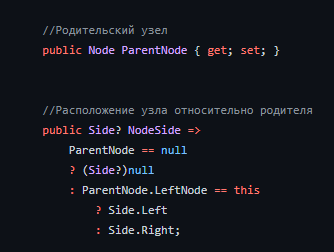
Этот класс является узлом ассоциативного контейнера, реализованного в виде дерева. Он предназначен для хранения пар ключ-значения.



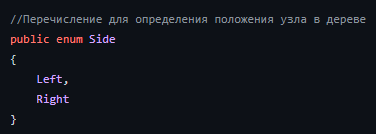
Поле Data предназначено для хранения значений контейнера, а поле Id-ключей, именно с помощью него в классе Tree реализуется поиск.



Данный метод используется в качестве конструктора класса Node, с помощью него происходит вставка данных в узел контейнера, в виде пары ключ-значение.



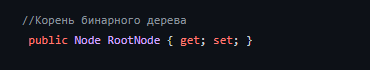
Также этот класс имеет ссылку на родительский Node, который находится наверху ассоциативного контейнера, реализованного в виде двоичного дерева, по другому его называют «корневым элементом». Метод, расположенный ниже позволяет удобно определять расположение узла относительно родительского, чтобы было понятно в какую часть дерева (правую или левую) необходимо добавить данный элемент.



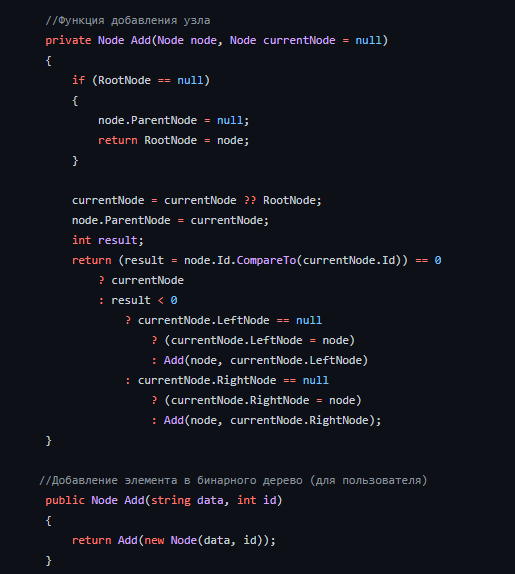
Перечисление используется для удобства обозначения ссылки на правый или левый элемент.

**Класс Tree.cs**

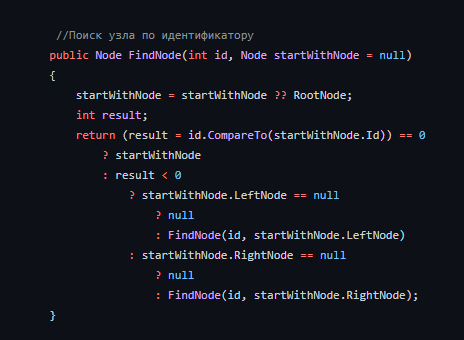
Данный класс представляет собой ассоциативный контейнер, хранящий пары ключ-значение в виде бинарного дерева, в нем реализованы необходимые методы, позволяющие осуществлять добавление, удаление и поиск элементов по ключу.



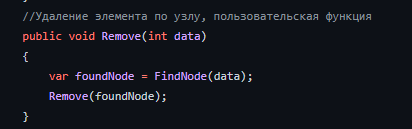
RootNode является корневым узлом, позволяющем в дальнейших методах начинать отсчет и проходиться по веткам дерева.



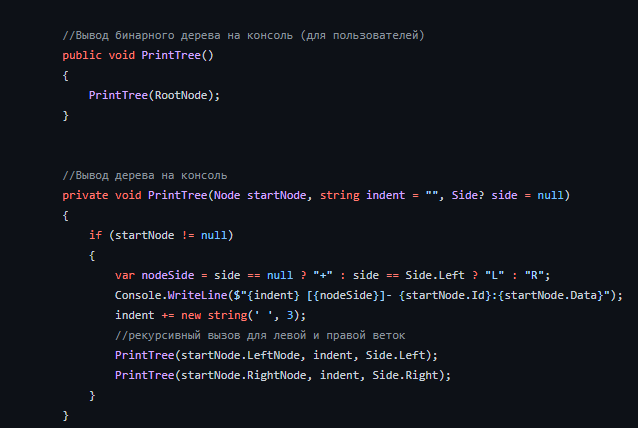
Закрытый метод Add принимает на вход какой-либо узел, и начинает обход для того, чтобы добавить элемент в дерево, публичный Add является оболочкой над закрытым методом и позволяет пользователю удобно добавить нужный элемент в ассоциативный контейнер.



Метод FindNode необходим для поиска узла по значению, он осуществляет обход дерева, начиная с корневого элемента и возвращает найденный узел, в случае его существования в рамках хранимой структуры данных.



RemoveNode осуществляет удаление элемента по индексу, в данном методе сначала реализуется поиск значения по ключу, в случае успешного выполнения вызывается закрытый метод remove, который удаляет найденный элемент из ассоциативного контейнера, меняя в соседних элементах ссылки для сохранения конфигурации структуры данных.



Метод PintTree необходим для вывода элементов ассоциативного контейнера в консоль, он осуществляет обход элементов, начиная с корневого узла и выводит их в соответствии с древовидной структурой.

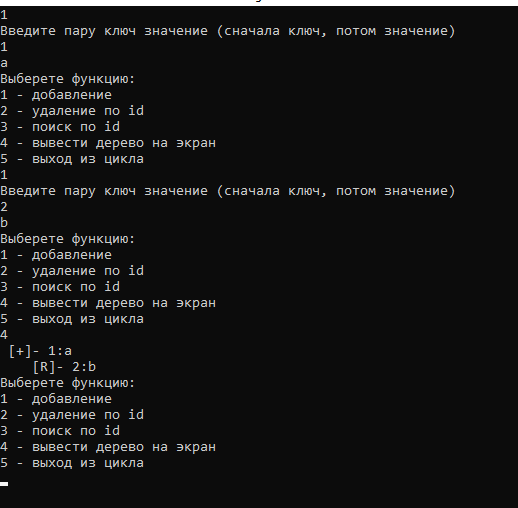
И последней функцией, необходимой для реализации меню и выбора функции из списка Add(), Remove(), FindNode(), PrintTree(), которую пользователь хочет использовать в данный момент является Menu().



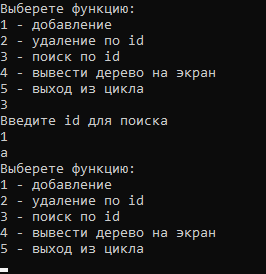
Функция постоянно запрашивает от пользователя ввод данных и на основании него вызывает необходимую функцию. В случае 1 происходит добавление элементов на основе дополнительного ввода данных в форме ключ-значение. Если выбор пал на второй метод, тогда пользователю предлагается ввести id для удаления элемента, для третьего – id для поиска. Последний, четвертый случай, когда появляется возможность вывода древовидной структуры на экран. Так же стоит отметить, что эта функция циклична, то есть может работать множество раз, пока пользователь на введет число «5».

**Класс Program.cs**

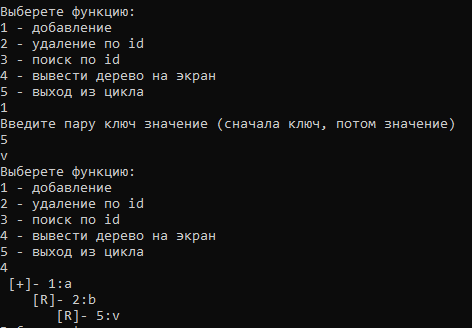
Класс Program.cs необходим для отладки и запуска программы. Он создает новый экземпляр класса Tree и вызывает метод Menu, в результате чего пользователь может воспользоваться всем функционалом программы.



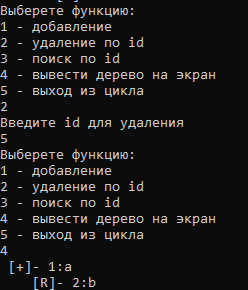
Сначала введем добавим некоторые значения в дерево, а затем протестируем остальные методы в рамках выполняемой программы.



Здесь видна корректность поиска элемента по id. Для подтверждения нормальной работоспособности метода Remove, сначала добавим еще один элемент, выведем дерево на экран, а затем попытаемся его удалить.



Произведем удаление нового элемента. Для этого сначала удалим элемент, а затем вызовем метод printTree().



Таким образом, данные тесты показывают корректность работы программы.