Санкт-Петербургский Национальный Исследовательский Университет Информационных Технологий, Механики и Оптики

Факультет Прикладной Информатики

**Лабораторная работа №3**

Выполнил:

Зенин Д.Д.

Проверил Иванов С. Е.

Санкт-Петербург,

2025

Оглавление

[ВВЕДЕНИЕ 3](#_Toc192504922)

[1. Задание 1 4](#_Toc192504923)

[ЗАКЛЮЧЕНИЕ 9](#_Toc192504930)

# ВВЕДЕНИЕ

Цель лабораторной работы №3 состояла в изучении алгоритма работы с деревьями. Необходимо было реализовать средствами ООП дерево и рекурсивные функции, выполняющие обход дерева в прямом, обратном, концевом порядке. Также вычислить значение выражения, заданного деревом.

**ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ**

1. Реализовать средствами ООП дерево и рекурсивные функции, выполняющие обход дерева в прямом, обратном, концевом порядке. Выполнить расчет примера. Обход дерева в прямом порядке a b d e c f. Обход дерева в обратном порядке d b e a c f. Обход дерева в концевом порядке d e b f c a
2. Вычислить значение выражения, заданного деревом. Вычисление выполнять в порядке концевого обхода.

# Задание 1

# В данном задании необходимо реализовать средствами ООП дерево и рекурсивные функции, выполняющие обход дерева в прямом, обратном, концевом порядке.

# Реализуется класс TreeNode – узел дерева. Его полями являются указатель на левое поддерево, на правое поддерево и значение узла. Конструктор инициализирует узел с заданным значением, а указатели Left и Right устанавливаются в null.

# 

Рисунок 1 – Класс TreeNode

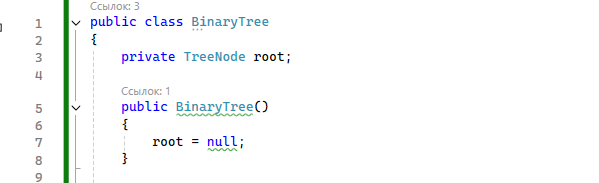
Далее создадим класс BinaryTree, который реализует бинарное дерево. Его поле – корень дерева. 

Рисунок 2 – Класс BinaryTree

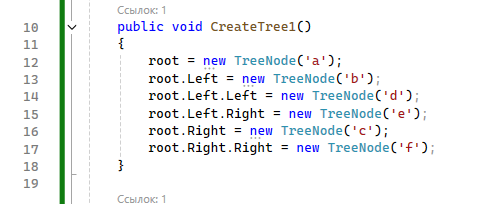
Создадим дерево для первого задания

Рисунок 3 – Метод CreateTree1()

Перейдем к прямому обходу. Логика прямого обхода:

* Если узел null, метод завершается.
* Выводится значение текущего узла.
* Рекурсивно вызывается метод для левого поддерева.
* Рекурсивно вызывается метод для правого поддерева.

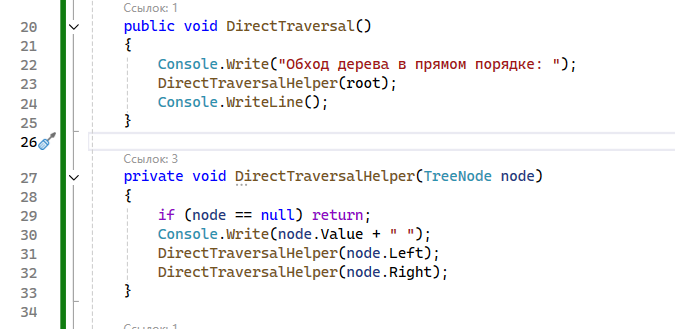


Рисунок 4 – Листинг прямого обхода

Перейдем к обратному обходу. Логика обратного обхода:

* Если узел null, метод завершается.
* Рекурсивно вызывается метод для левого поддерева.
* Выводится значение текущего узла.
* Рекурсивно вызывается метод для правого поддерева.

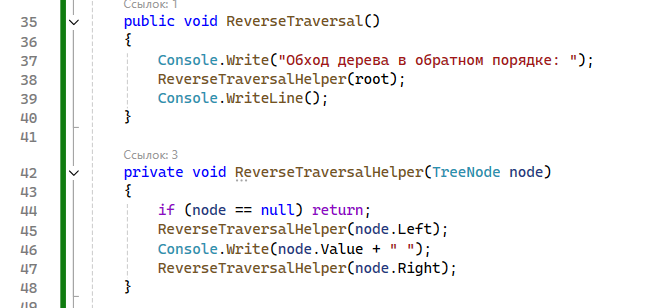


Рисунок 5 – Листинг обратного обхода

Перейдем к концевому обходу. Логика концевого обхода:

* Если узел null, метод завершается.
* Рекурсивно вызывается метод для левого поддерева.
* Рекурсивно вызывается метод для правого поддерева.
* Выводится значение текущего узла.

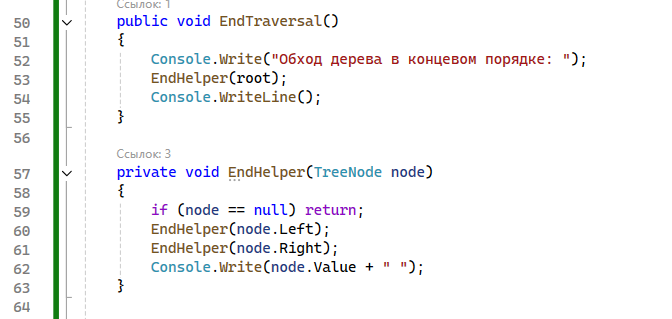


Рисунок 6 – Листинг для концевого обхода

В методе Main создадим дерево и запустим все виды обходов

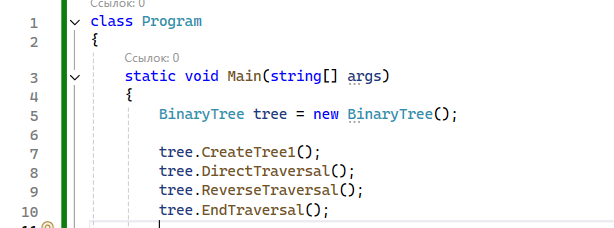


Рисунок 7 – Метод Main для задания 1

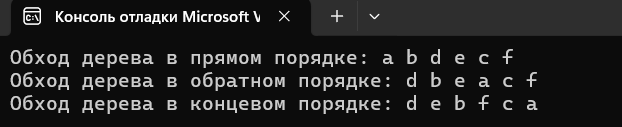
Ниже представлен вывод программы: 

Рисунок 8 – Вывод программы

# Задание 2

В классе BinaryTree создадим методы для построения дерева. Вызывается вспомогательный метод BuildTreeHelper(values, ref index), который рекурсивно строит дерево. Также проверяется, что после построения дерева не осталось необработанных символов (кроме '.').

В методе BuildTreeHelper:

Если index выходит за пределы списка или текущий символ — '.', увеличиваем index и возвращаем null. Создается новый узел с текущим символом (values[index]), index увеличивается, после рекурсивно строится левое и правое поддеревья. Возвращается построенный узел.

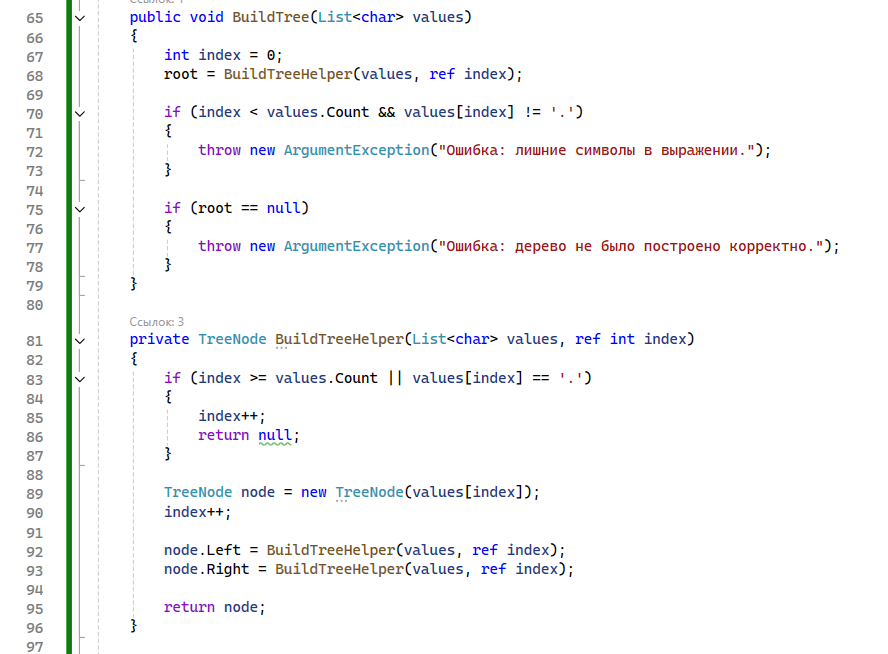


Рисунок 9 – Листинг создания дерева

Внутри метода CalculateExpression вызывается вспомогательный метод CalculateExpressionHelper(TreeNode node) с корнем дерева. В методе CalculateExpressionHelper если узел null, то выбрасывается исключение. Если значение узла — цифра, то возвращается её числовое значение. Иначе рекурсивно вычисляется значение левого поддерева, после для правого, проверяется деление на ноль и выполняется соответствующая операция.



Рисунок 10 – Листинг вычисления выражения

Дальше представлю методы, созданные для сборки дерева в постфиксную нотацию(для ответа).Внутри метода GetPostfixExpression вызывается вспомогательный метод GetPostfixExpressionHelper(TreeNode node). Логика метода GetPostfixExpressionHelper такова: если узел null, возвращается пустая строка. Если узел — число, возвращается его значение с пробелом. Иначе рекурсивно собирается строка для левого поддерева, после для правого. В конце добавляется значение текущего узла (оператор) с пробелом.



Рисунок 11 – Листинг для сборки выражения в постфиксную нотацию

В методе Main создадим дерево и выполним нужные методы.

 Рисунок 12 – Метод Main для задания 2

Вывод программы:

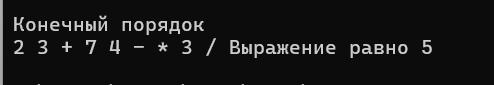


Рисунок 13 – Вывод программы

# ЗАКЛЮЧЕНИЕ

В ходе выполнения работы были изучены алгоритмы работы с деревьями.