Деревья

(деревья поиска / деревья выражений)

Основные понятия дихотомического дерева

Ключ – уникальное значение атрибута (в контексте всего дерева), характерное для узла.
Дихотомия – определенное отношение порядка в текущем дереве, характеризующееся разбиением множества узлов на два непересекающихся подмножества.

Дихотомическое дерево (дерево поиска) – это бинарное дерево, организованное так, что для каждого узла, имеющего ключ К, справедливо утверждение о том, что в его левом поддереве содержатся узлы с ключами, меньшими К, а в его правом поддереве содержатся узлы с ключами, большими К.

Программное представление узла дерева поиска

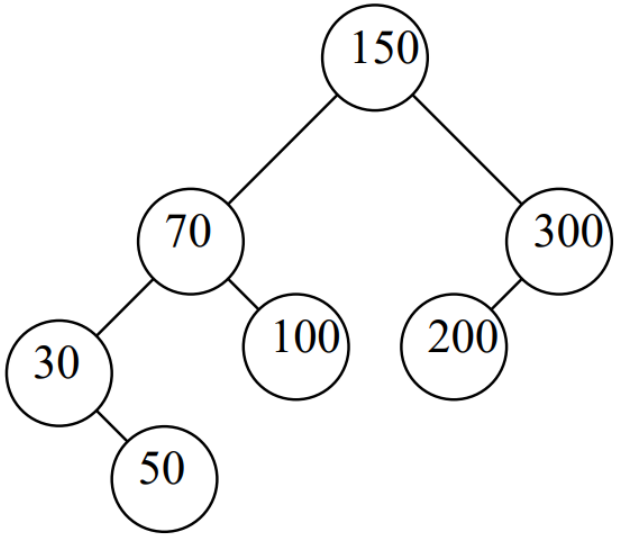
```
public class DTreeNode // Класс «Узел дихотомического дерева»
  {
     private char info; // информационное поле
     private int key; // поле ключа
     private DTreeNode left; // ссылка на левое поддерево
     private DTreeNode right; // ссылка на правое поддерево
     public char Info { get; set; } // свойства
     public int Key { get; set; }
     public DTreeNode Left { get; set; }
     public DTreeNode Right { get; set; }
     public DTreeNode() { } // конструкторы
     public DTreeNode(char info, int key)
     {
        Info = info; Key = key;
     }
     public DTreeNode(char info, int key, DTreeNode left, DTreeNode right)
     {
        Info = info; Key = key; Left = left; Right = right;
     }
  }
```

Основа класса дерева поиска

```
public class DixotomyTree // класс «Дихотомическое дерево»
  {
     private DTreeNode root; // ссылка на корень дихотомического дерева
     public DTreeNode Root // свойство, открывающее доступ к корню дерева
     {
        get { return root; }
        set { root = value; }
     }
     public DixotomyTree() // создание пустого дерева
     {
        root = null;
     }
                          //…
  }
```

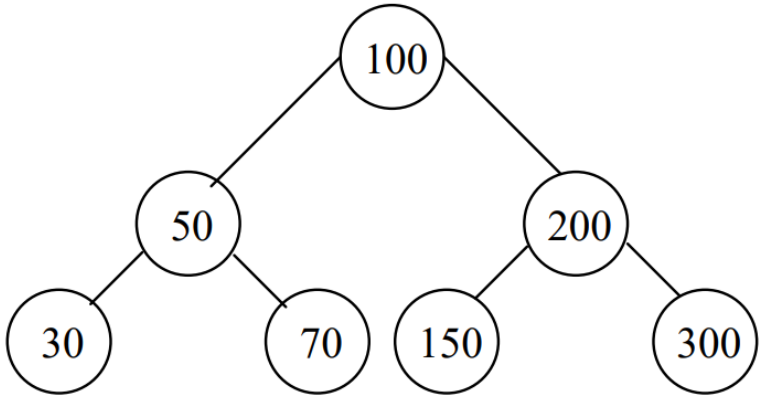
Пример задания дихотомического дерева

150, 70, 300, 100, 30, 200, 50;



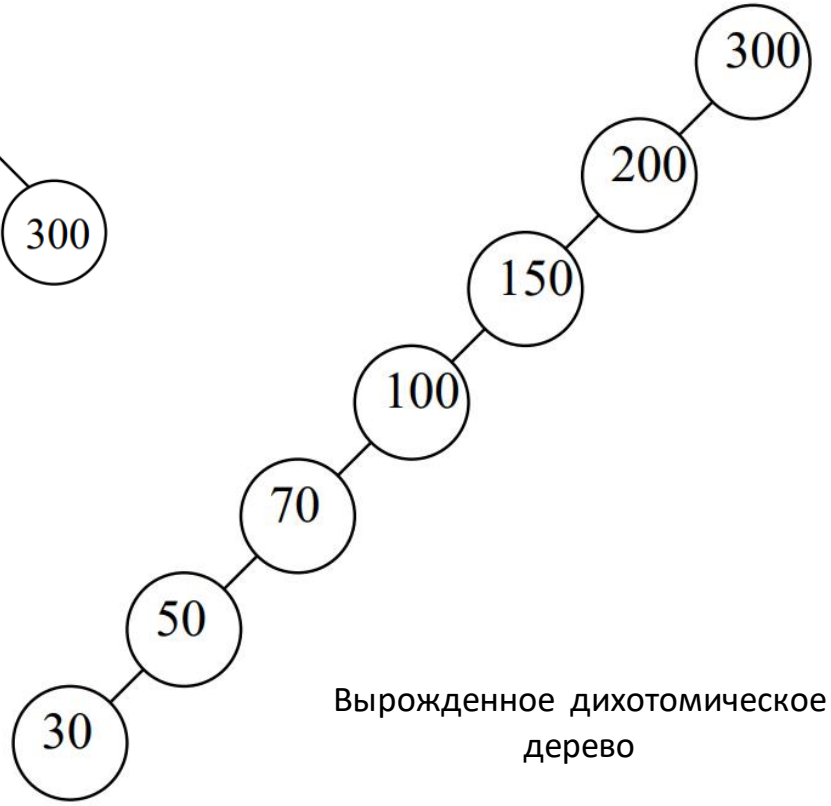
Несбалансированное дихотомическое дерево

100, 50, 30, 70, 200, 150, 300;



Сбалансированное дихотомическое дерево

300, 200, 150, 100, 70, 50, 30.



Вырожденное дихотомическое дерево

Операции с дихотомическими деревьями

Поиск - осуществляется на основе сравнения заданного ключа с ключом корня.

Включение в дерево узла с заданным значением:

- Если поиск завершен на null –> новый узел включается на место пустого поддерева

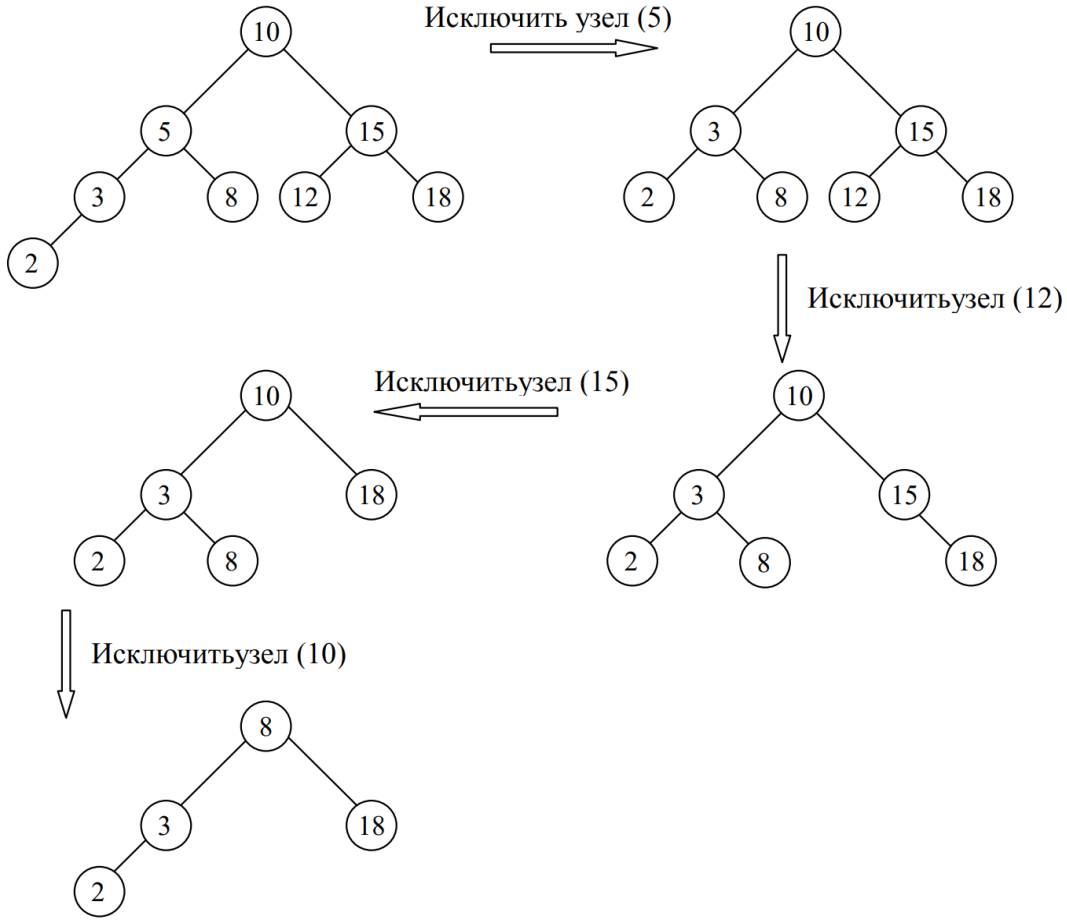
```
public DTreeNode Insert_DNode(DTreeNode root, int k)
    {
      if(root == null)
      {//создание корня дерева
        root = new DTreeNode(' ', k, null, null);
      }
      else //дерево не пусто - продолжить поиск элемента с ключом
      {
        if (k < root.Key) //поиск в левом поддереве
          root.Left = Insert_DNode(root.Left, k);
        else if (k > root.Key)//поиск в правом поддереве
          root.Right = Insert_DNode(root.Right, k);
        else //ключ дб уникальным
          Console.WriteLine($"узел с ключом {k} уже есть в дереве");
      }
      return root;
    }
```

- Если узел не содержит искомого ключа -> производится рекурсивный поиск в левом и правом поддеревьях
- Если достигнут конец ветви, и не обнаружен ключ -> создается новый узел со значение этого ключа

Операции с дихотомическими деревьями

Удаление:
Подразумевает следующие варианты:

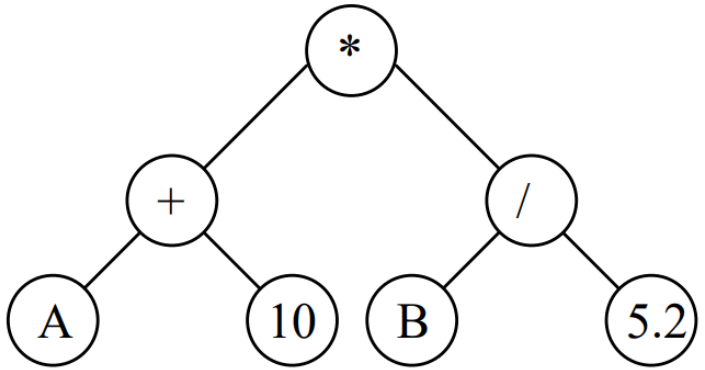
1. узла с заданным значением ключа в дереве нет;
2. узел с заданным значением ключа имеет не более одного потомка.
3. узел с заданным значением ключа имеет двух потомков.



Разрушение бинарного дерева любого вида заключается в присвоении null ссылке на корень дерева. В результате разрушения дерево становится пустым.

Основные понятия деревья выражений

Дерево выражений – бинарное дерево, в корневых узлах которого хранятся признаки операций, а в терминальных узлах – операнды выражения (переменные или константы).



Нисходящий алгоритм обхода (префиксная форма представления)

* + A 10. / B 5.2

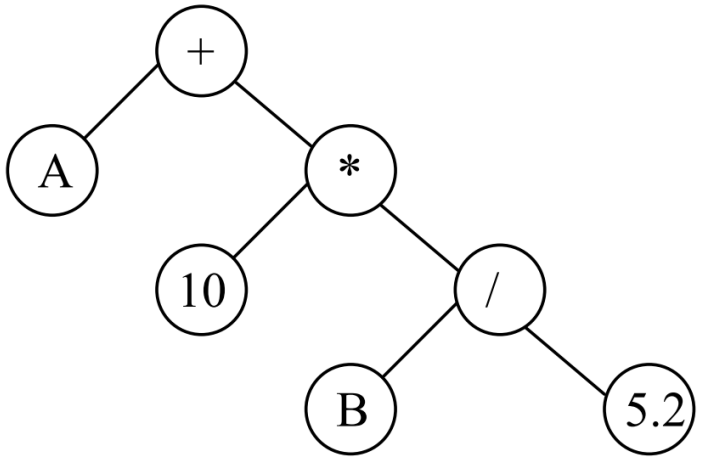
Восходящий алгоритм обхода (постфиксная форма представления)

A 10 + B 5.2 / *

смешанный алгоритм обхода (инфиксная форма представления)

A + 10. * B / 5.2

Дерево выражений



$$(A + (10. * (B / 5.2)))$$