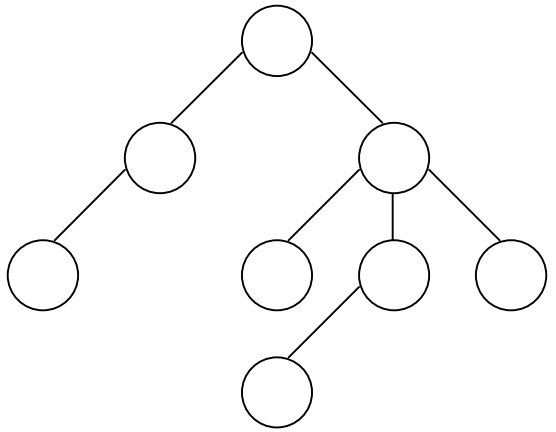
Деревья

Основные понятия

дерево (tree) с базовым типом T — это
– либо пустая структура;
– либо узел типа T, с которым связано конечное число древовидных структур, называемых поддеревьями (subtree).

Пустое дерево – не содержит узлов.



Основные понятия

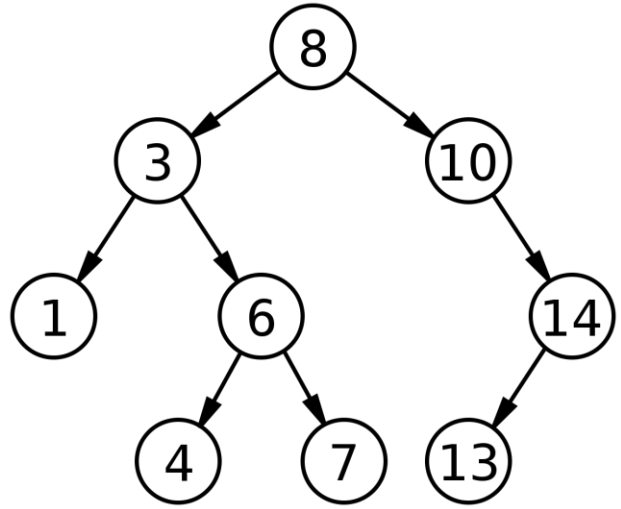
корень (root) – самый «верхний» узел дерева;
ветвь (brunch) – цепочка связанных между собой узлов;
Терминальный узел / лист (leaf) – узел, не имеющий поддеревьев.

родительским (parent) называется узел, который находится непосредственно над другим узлом;
дочерним (child) называется узел, который находится непосредственно под другим узлом;

предки данного узла — все узлы на пути от корня до данного узла;
потомки — все узлы, расположенные ниже данного

Основные понятия

узел (node) — это точка, где может возникнуть ветвь;
внутренний узел (internal node) — узел, не являющийся ни листом, ни корнем;
Порядок/степень узла (node degree) — количество его дочерних узлов;
Глубина / уровень (depth) узла — количество его предков плюс единица;



длина пути к узлу — количество ветвей, которые нужно пройти, чтобы продвинуться от корня к данному узлу;
длина пути дерева — сумма длин путей всех его узлов, называемая также длиной внутреннего пути.
сестринские (братские) — узлы, у которых один и тот же родитель.

Основные понятия

Степень/порядок дерева (d) – максимальная степень(порядок) всех узлов дерева

Высота / глубина дерева (h) – максимальный уровень всех узлов дерева

Полное – дерево, содержащее максимальное количество узлов
В полном дереве у всех узлов, за исключением терминальных, число непосредственных потомков равно степени дерева.

Количество узлов в полном дереве общего вида

$$\sum_{i=0}^{h-1} d^i$$

Где

i – номер уровня без единицы

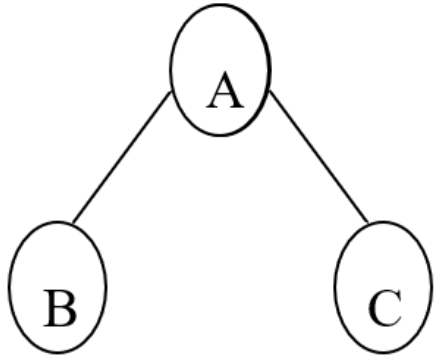
d – степень

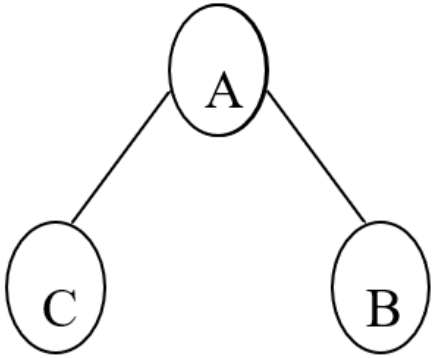
h - высота дерева

Основные свойства деревьев общего вида

- корень не имеет предков;
- каждый узел, за исключением корня, имеет только одного предка;
- каждый узел связан с корнем единственным путем, т.е. в деревьях отсутствуют замкнутые контуры (циклы).

Упорядоченное дерево – подразумевает важность относительного порядка поддеревьев



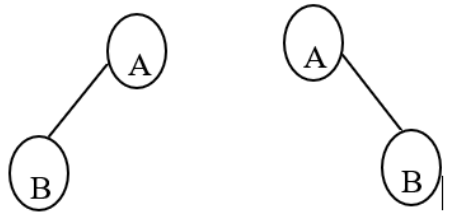


Бинарные деревья

- это конечное множество узлов, которое или пусто, или состоит из корня и двух непересекающихся бинарных деревьев, называемых левым и правым поддеревьями данного корня

Максимальное число узлов в бинарном дереве высотой h $(d=2)$:

$$N_2(h) = \sum_{i=0}^{h-1} 2^i = 2^h - 1$$

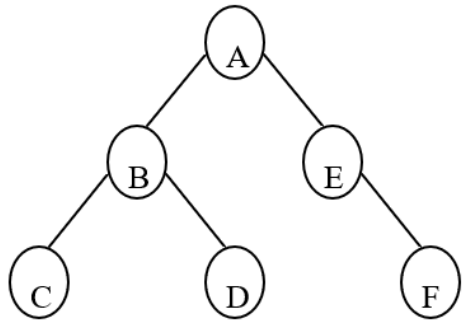


Максимальное число узлов на уровне i в бинарном дереве:

$$n_i = 2^{i-1}$$

Высота полного бинарного дерева, содержащего N узлов:

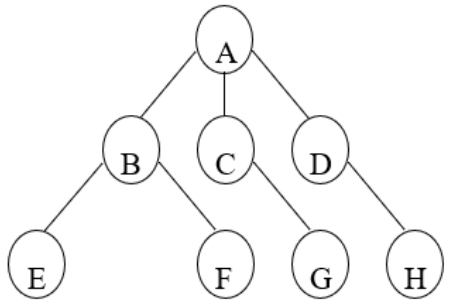
$$h = \lfloor log_2 N \rfloor + 1$$



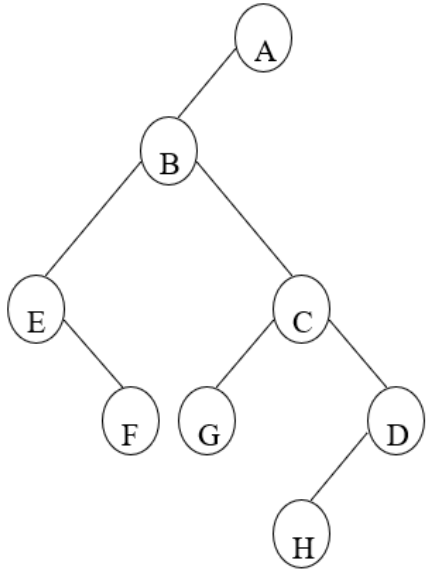
Преобразование дерева произвольной степени к бинарному

Левосторонний алгоритм:

- у каждого узла дерева произвольной степени необходимо сохранить самую левую связь,
- узлы – потомки одного и того же узла следует соединить правой связью



Дерево произвольной степени



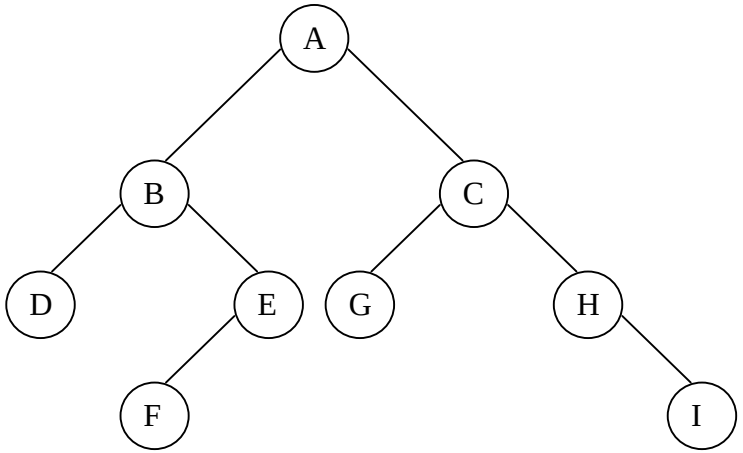
Бинарное дерево

Представление бинарных деревьев

1. Последовательное представление

Если известен максимальный размер дерева, то структуру дерева можно хранить в виде массива

Для каждого узла с номером *k* его левый потомок будет храниться в элементе с индексом [2 * k], а правый – в элементе с индексом [2 * k + 1]



A	B	C	D	E	G	H			F					I
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Представление бинарных деревьев

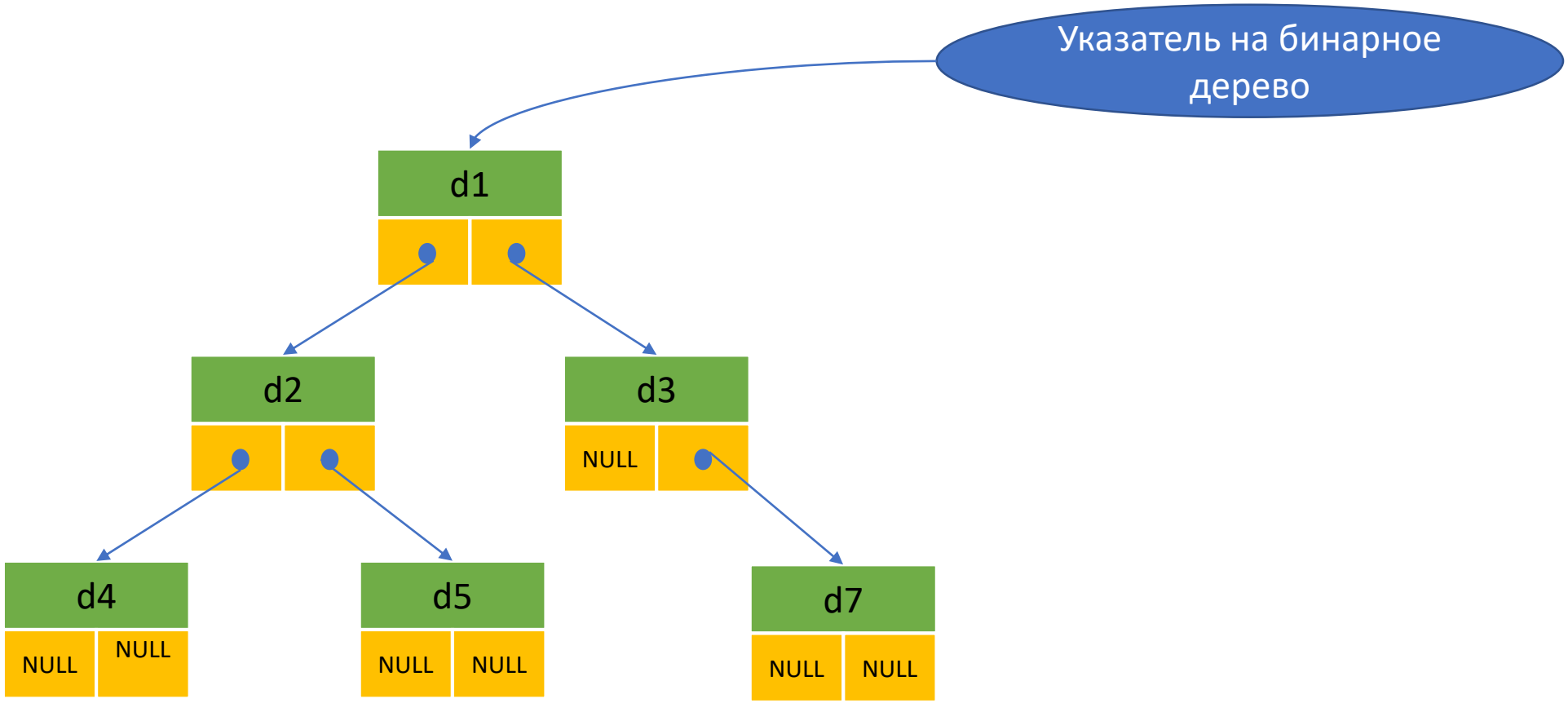
2. Связанное представление

```
public class TreeNode// Класс «Узел бинарного дерева»
    {
        private char info;       // информационное поле
        private TreeNode left;   // ссылка на левое поддерево
        private TreeNode right; // ссылка на правое поддерево

        public char Info { get; set; }    // свойства
        public TreeNode Left { get; set; }
        public TreeNode Right { get; set; }

        public TreeNode() { }   // конструкторы
        public TreeNode(char info)
        {
            Info = info;
        }
        public TreeNode(char info, TreeNode left, TreeNode right)
        {
            Info = info; Left = left; Right = right;
        }
    }
```

```
public class BinaryTree // Класс «Бинарное дерево произвольного вида»
    {
        private TreeNode root;  // ссылка на корень дерева
        public TreeNode Root    // свойство, открывающее доступ к корню дерева
        {
            get { return root; }
            set { root = value; }
        }
        public BinaryTree() // создание пустого дерева
        {
            root = null;
        }
        //...
    }
```

Связанное дерево

Обход – это операция, при выполнении которой каждый узел обрабатывается ровно один раз одинаковым образом

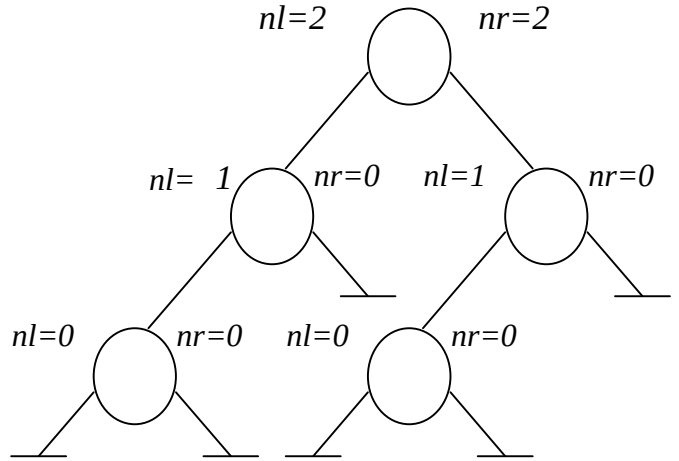


Сбалансированные дерево

Дерево называется ***идеально сбалансированным***, если для каждой вершины число узлов в ее правом и левом поддеревьях отличается не более чем на единицу

Правило равномерного размещения для *N* узлов (*правило 1*) формулируется с помощью рекурсии:

- создать один узел в качестве корня;
- по *правилу 1* построить левое поддерево с числом узлов *nl = N/2;*
- по *правилу 1* построить правое поддерево с числом узлов *nr = N - nl - 1*.



Построение сбалансированного дерева

```
 public TreeNode Create_Balanced(int n)  // n – количество узлов в дереве
        {

            char x;
            TreeNode root;     // ссылка на корень дерева и на корень любого из
поддеревьев
            if (n == 0)
                root = null; // если n == 0, построить пустое дерево
            else
            {   // заполнить информационное поле корня
                Console.WriteLine("введите значение поля узла (символ):");
                x = Char.Parse(Console.ReadLine());
                root = new TreeNode(x);      // создать корень дерева
                root.Left = Create_Balanced(n/2);    // построить левое поддерево
(*1*)
                root.Right = Create_Balanced(n - n/2 - 1);    // построить правое
поддерево  (*2*)
            }
            return root; //(*3*)
        }
```