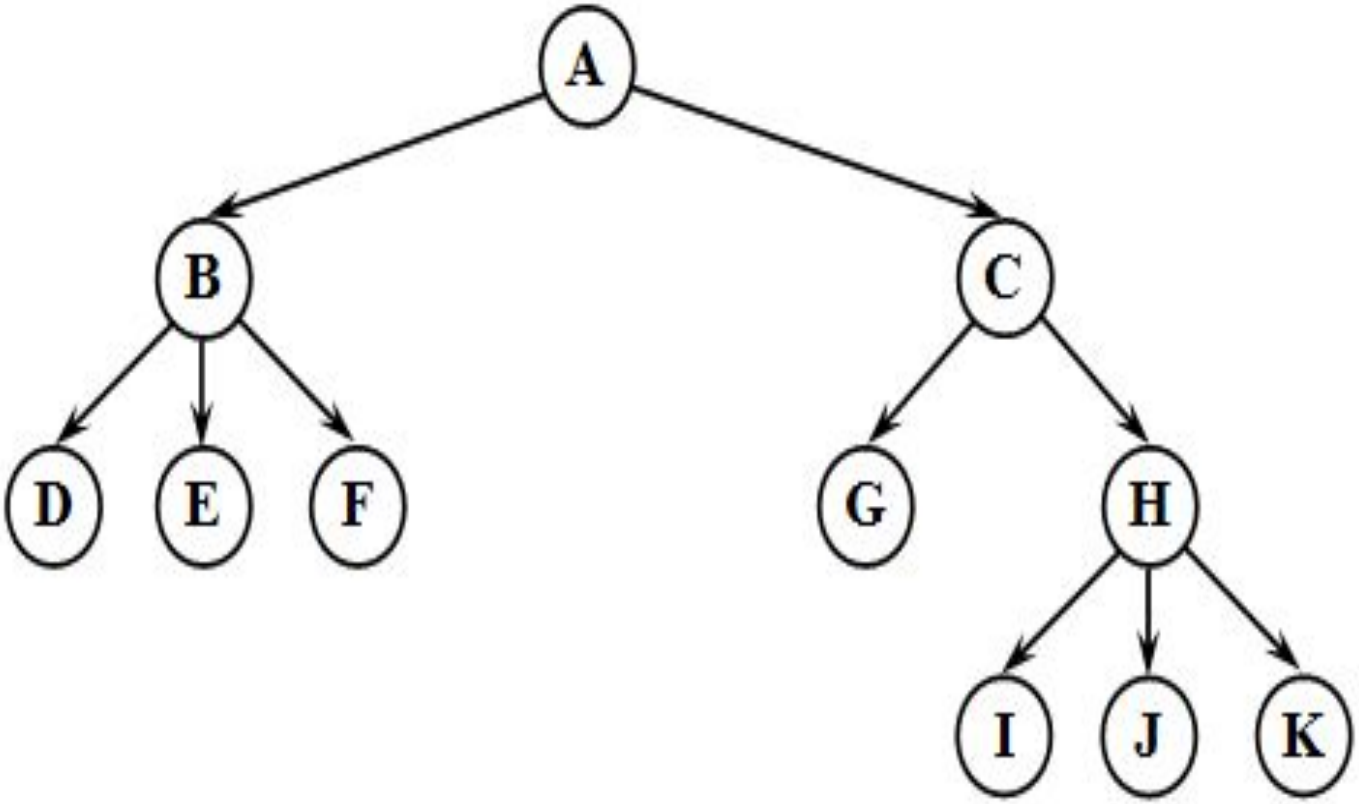
Деревья

Разработала Бубнова Надежда Дмитриевна

Пример дерева



- **Дерево** – это совокупность элементов и отношений, образующих иерархическую структуру этих элементов.
- Каждый элемент дерева называется *вершиной (узлом) дерева*.
- Вершины дерева соединены направленными дугами, которые называют *ветвями дерева*. Начальный узел дерева называют *корнем дерева*, ему соответствует нулевой уровень. *Листьями дерева* называют вершины, в которые входит одна ветвь и не выходит ни одной ветви

Формальное определение дерева

- Один узел является деревом. Этот же узел также является корнем этого дерева.
- Пусть n – это узел, а T 1 , T 2 , ... T m – деревья с корнями n 1 , n 2 , … n m соответственно. Можно построить новое дерево, сделав n родителем узлов n 1 , n 2 , … n m . В этом дереве n будет корнем, а T 1 , T 2 , ... T m – поддеревьями этого корня. Узлы n 1 , n 2 , … n m называются **сыновьями** узла n .

Термины

- Все вершины, в которые входят ветви, исходящие из одной общей вершины, называются *потомками*, а сама вершина – *предком*. Для каждого предка может быть выделено несколько потомков(отец, сыновья)
- *Определение уровня*
  1. *Уровень корня равен 1*
  2. *Уровень* потомка на единицу превосходит уровень его предка.

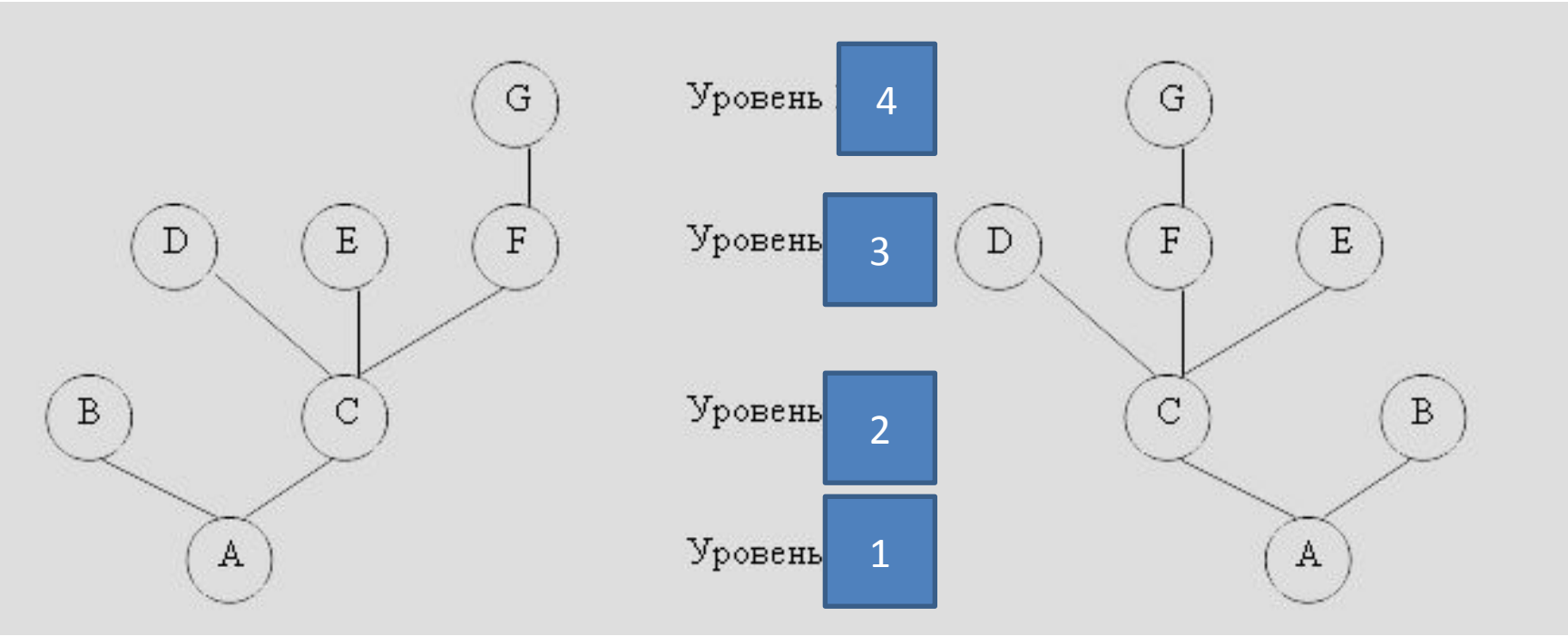
Высота дерева

- *Определение* *Высота (глубина)* дерева равна максимальному уровню вершины в дереве.

Примечание

Высота пустого дерева равна нулю, высота дерева из одного корня – единице.

Деревья высоты 4



Определения

- **Поддерево** – часть дерева, которая может быть представлена в виде отдельного дерева.

- *Степенью вершины* в дереве называется количество дуг, которое из нее выходит.
- *Степень дерева* равна максимальной степени вершин, входящих в дерево. (При этом листьями в дереве являются вершины, имеющие нулевую степень)

Типы деревьев

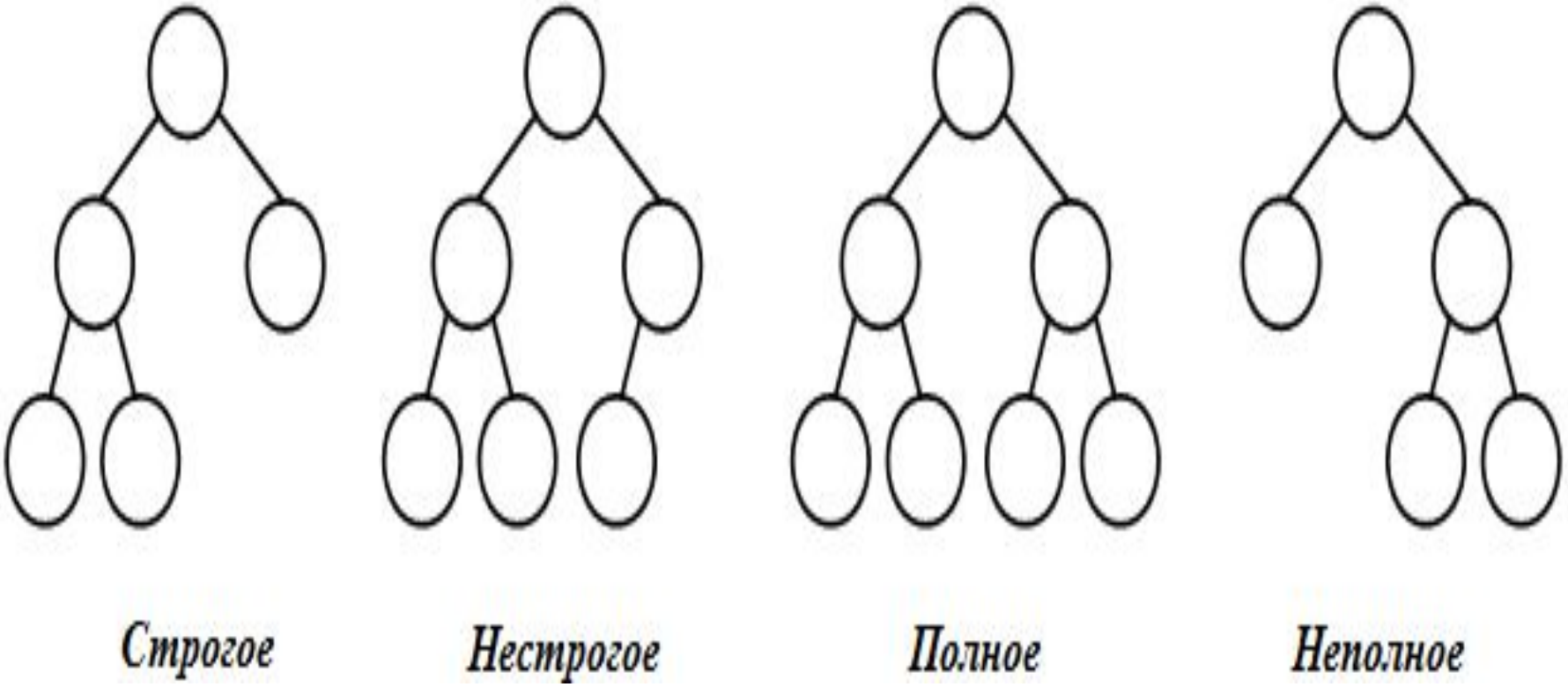
По величине степени выделены два типа деревьев:

- двоичные – степень дерева не более двух;
- сильноветвящиеся – степень дерева произвольна

Типы деревьев

- **Упорядоченное дерево** – это дерево, у которого ветви, исходящие из каждой вершины, упорядочены по определенному критерию.

Типы бинарных деревьев



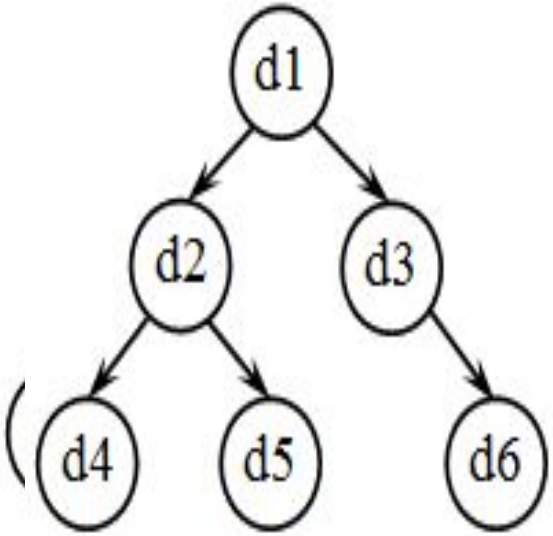
Типы бинарных деревьев

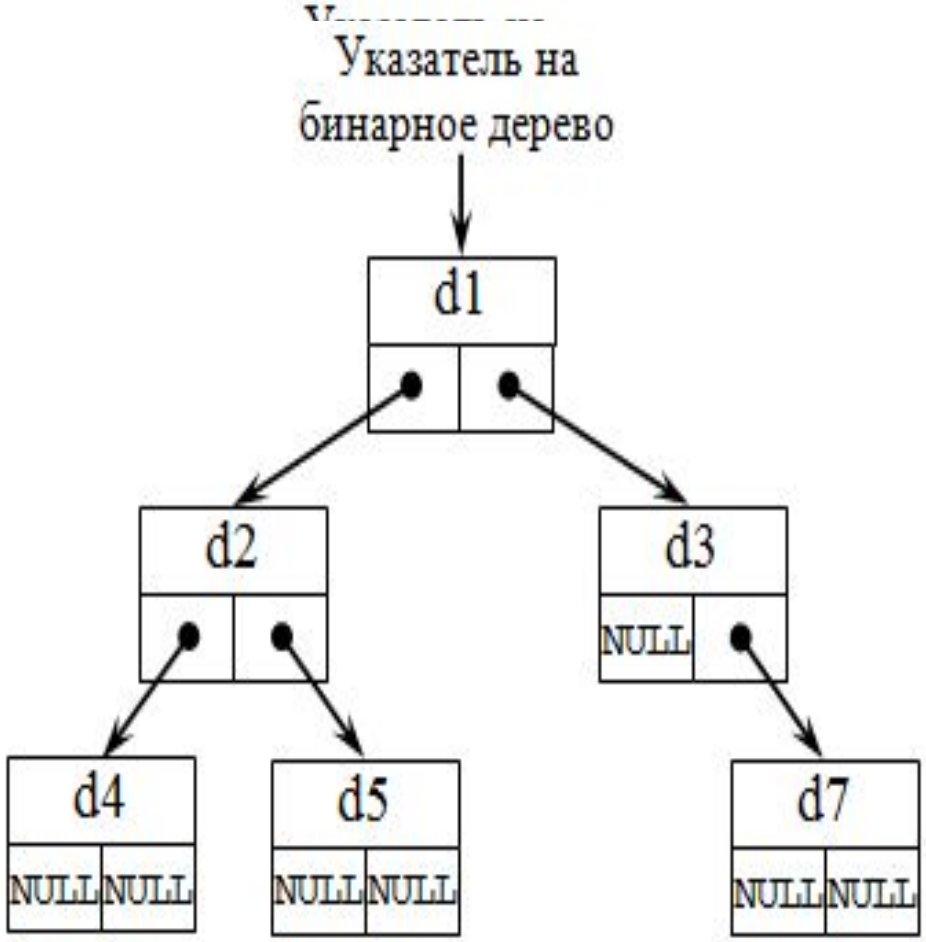
- *строгие* – вершины дерева имеют степень ноль (у листьев) или два (у узлов);
- *нестрогие* – вершины дерева имеют степень ноль (у листьев), один или два (у узлов);
- полные – на всех уровнях, кроме последнего, узлы степени 2, а на последнем – степени 0.

Представление бинарных деревьев в памяти

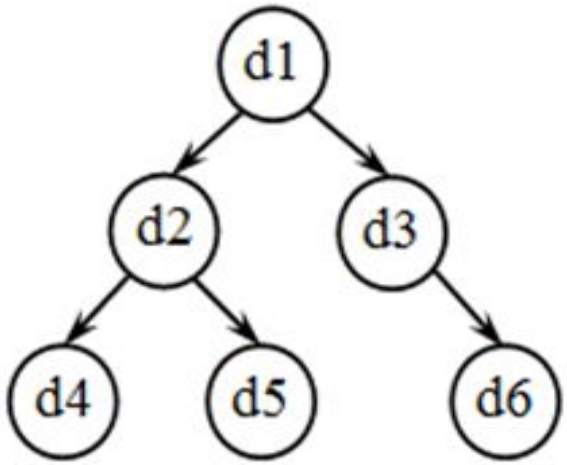
- Списковая структура
- Векторное представление

Представление бинарных деревьев списковой структурой





Представление бинарных деревьев в векторной памяти



Для хранения дерева выделяется массив. Например, массив A. Корень хранится в элементе массива A[1]. Для остальных узлов справедливо утверждение: если отец записан в элементе A[I], то его левый сын в A[2*I], а правый сын – в A[2*I+1].

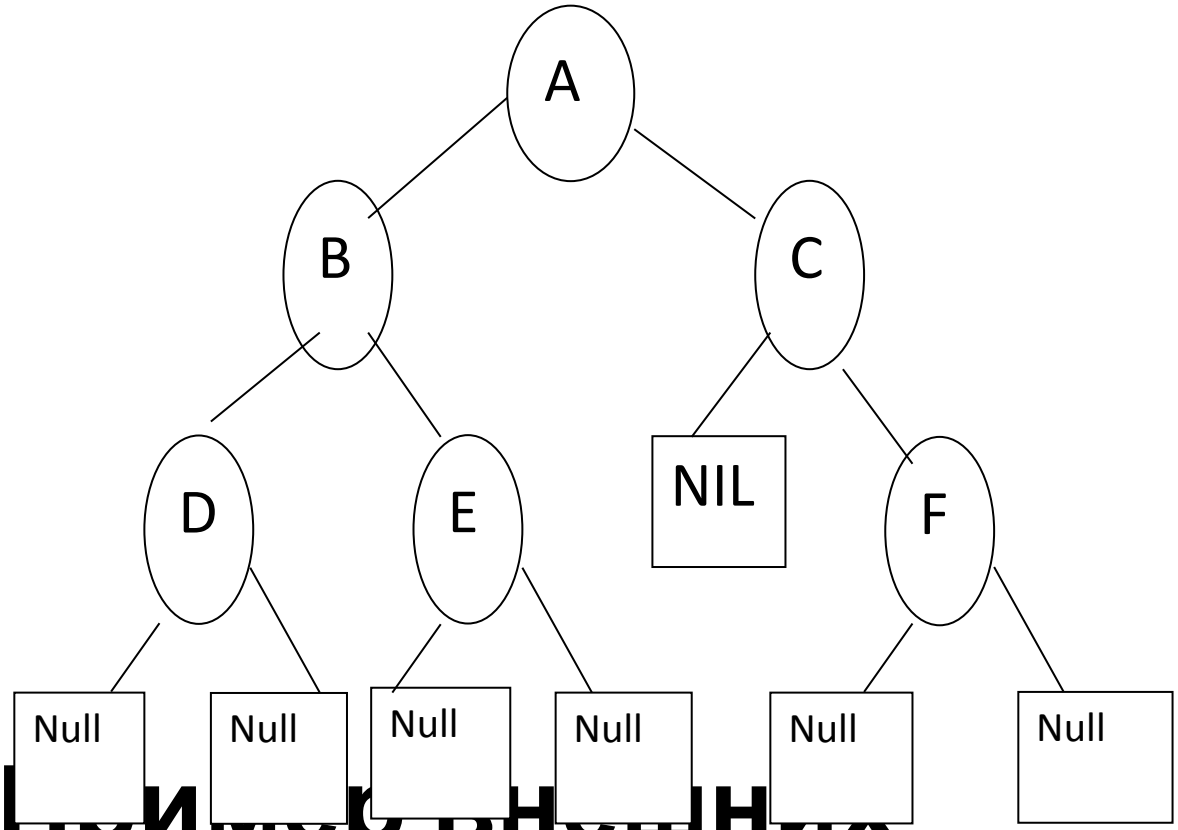
d1	*d2*	*d3*	*d4*	*d5*		*d6*
1	2	3	4	5	6	7

Информация в узле бинарного дерева

- информационное поле (ключ вершины);
- служебное поле (их может быть несколько или ни одного);
- указатель на левое поддерево;
- указатель на правое поддерево.

Термины

- **Длина пути** от корня до узла соответствует уровню узла.
- **Длина внутреннего пути** – сумма длин путей до всех узлов дерева(их иногда называют внутренними).
- **Внешний узел** – обозначение позиции вставки нового узла в бинарном дереве.
- **Длина внешнего пути** – сумма длин путей до всех внешних узлов дерева.



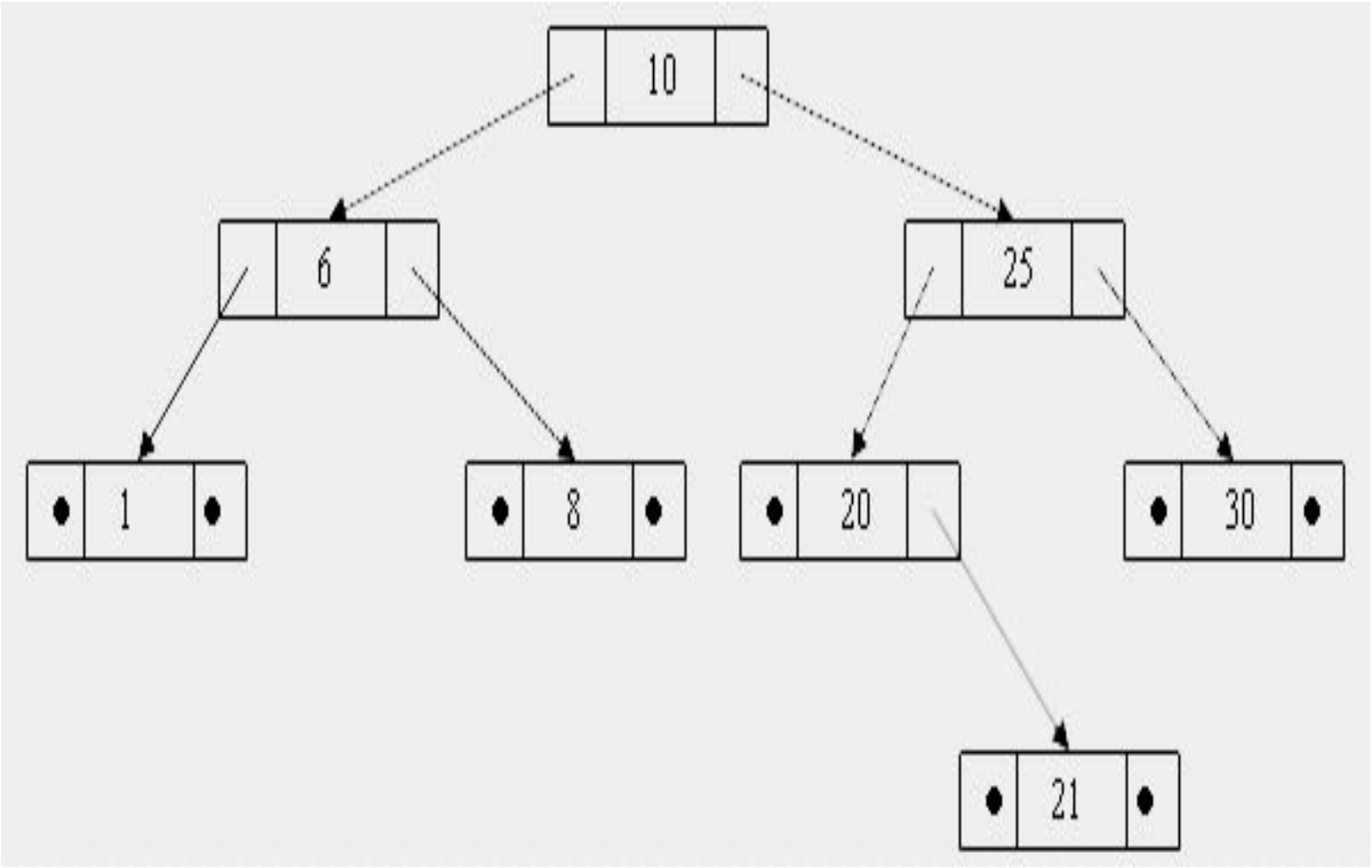
Пример внешних узлов

Внешние узлы помечены NULL

Дерево поиска

Дерево, для каждого узла которого все ключи узлов его левого поддерева меньше ключа этого узла, а все ключи его правого поддерева больше, называется **деревом поиска** или двоичным упорядоченным деревом.

Пример дерева поиска



Операции с двоичными деревьями(деревья поиска)

- Поиск в дереве;
- Обход дерева;
- Включение узла в дерево;
- Удаление узла из дерева.

Поиск в дереве. Алгоритм

- Если дерево не пусто, то нужно сравнить искомый ключ с ключом в корне дерева:
- если ключи совпадают, поиск завершен;
- если ключ в корне больше искомого, выполнить поиск в левом поддереве;
- если ключ в корне меньше искомого, выполнить поиск в правом поддереве.
- если дерево пусто, то искомый элемент не найден.

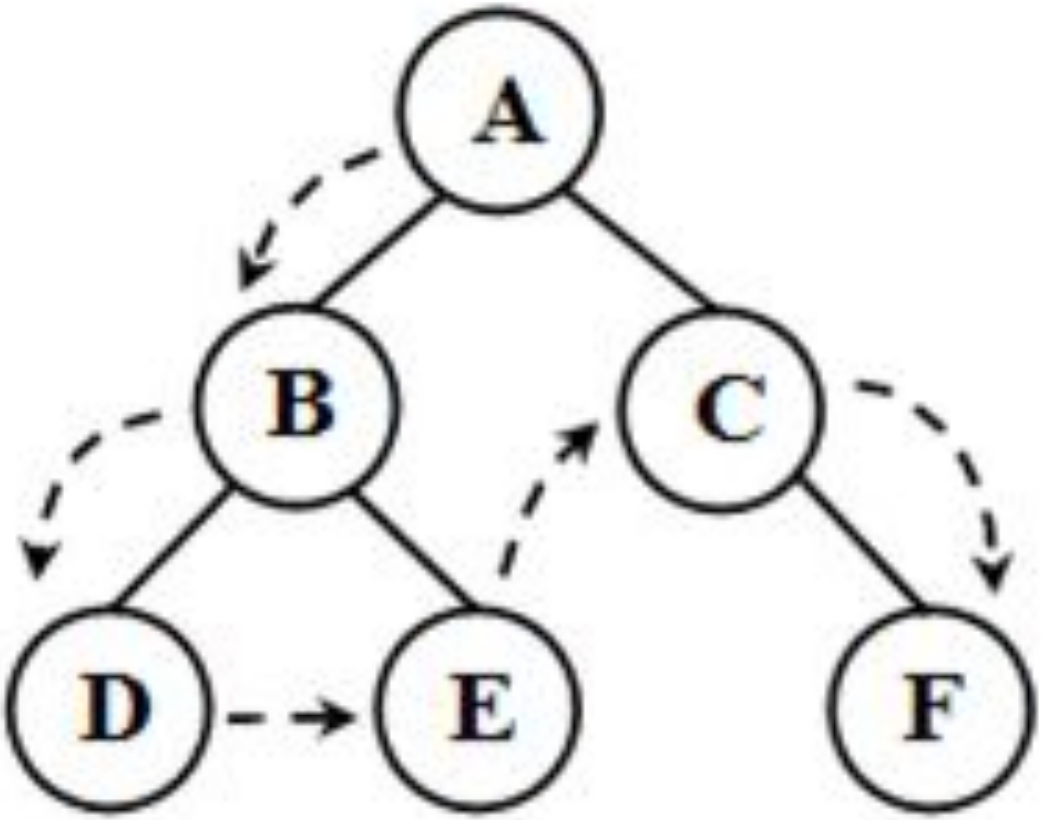
Обходы дерева

- прямой,
- обратный ,
- симметричный обходы.

Прямой обход

- Если дерево T является нулевым деревом, то в список обхода записывается пустая строка;
- Если дерево T состоит из одного узла, то в список обхода записывается этот узел;
- Сначала посещается корень n ,
- в прямом порядке посещаются узлы левого поддерева,
- в прямом порядке посещаются узлы правого поддерева

Прямой

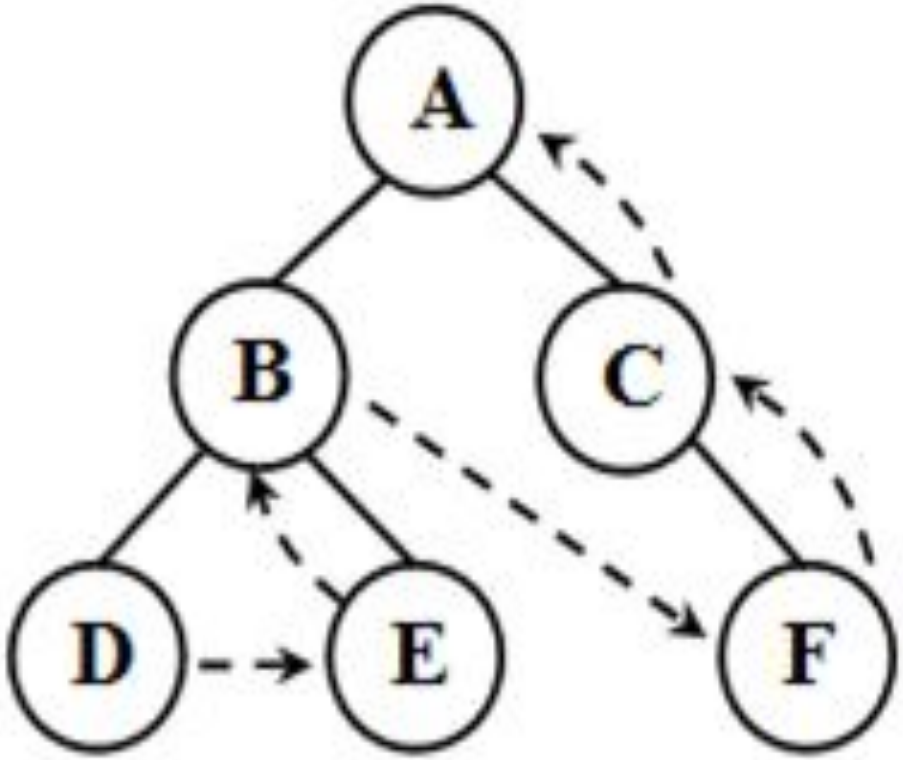


A B D E C F

Обратный обход .

- Если дерево T является нулевым деревом, то в список обхода записывается пустая строка;
- Если дерево T состоит из одного узла, то в список обхода записывается этот узел;
- Посещаются в обратном порядке все узлы левого поддерева,
- Посещаются в обратном порядке все узлы правого поддерева,
- посещается корень n .

Обратный

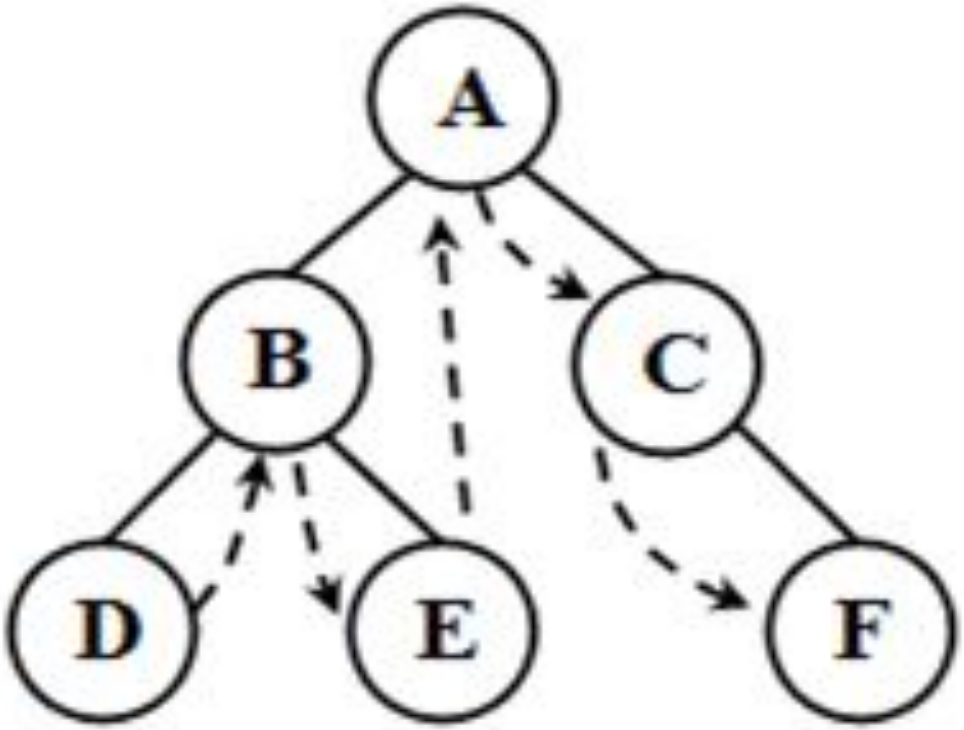


D E B F C A

Симметричный обход .

- Если дерево T является нулевым деревом, то в список обхода записывается пустая строка;
- Если дерево T состоит из одного узла, то в список обхода записывается этот узел;
- В симметричном порядке посещаются все узлы левого поддерева,
- Посещается корень,
- В симметричном порядке посещаются все узлы правого поддерева.

Симметричный



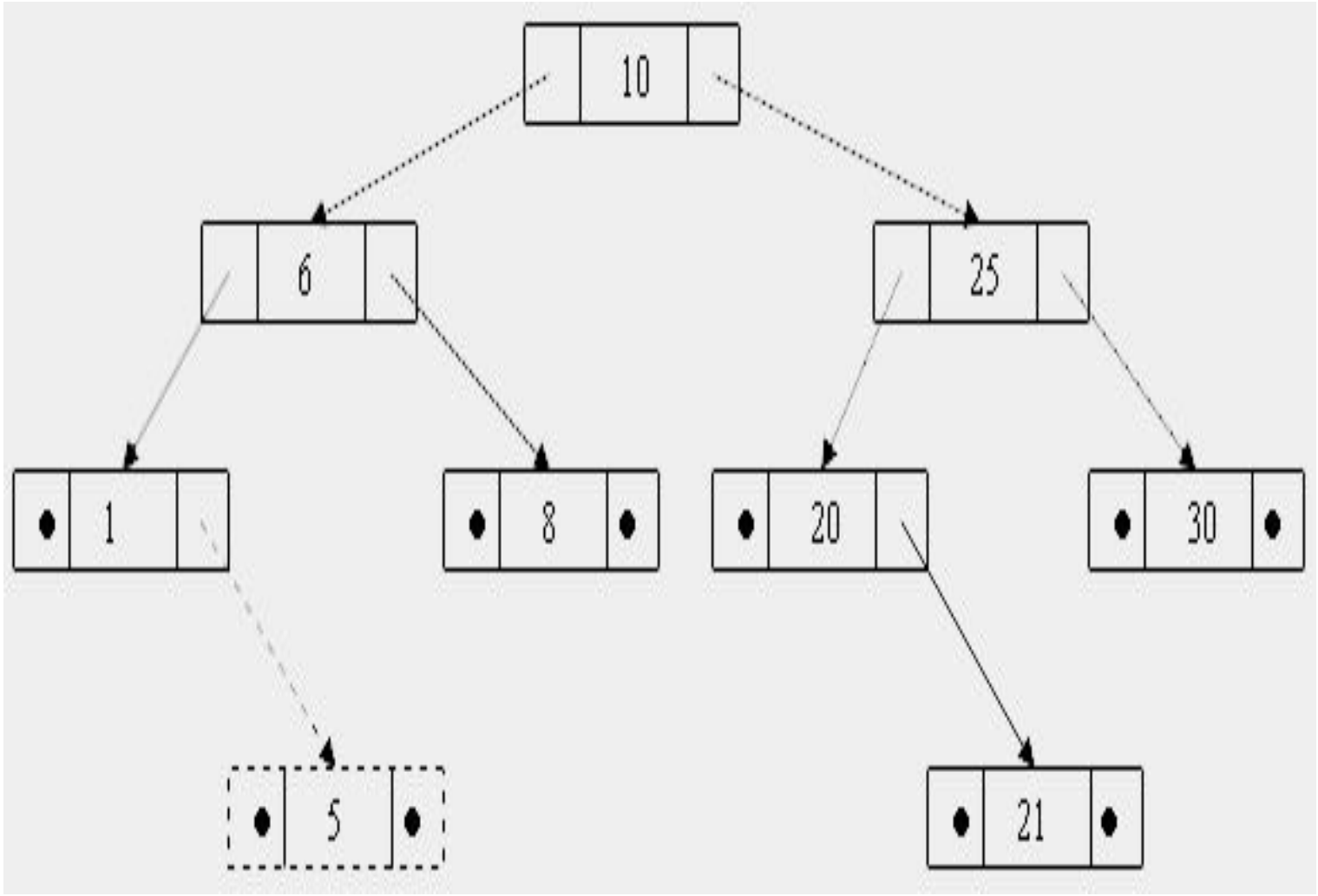
D B E A C F

Вставка узла в дерево поиска (алгоритм)

Для того чтобы вставить узел, необходимо найти его место.

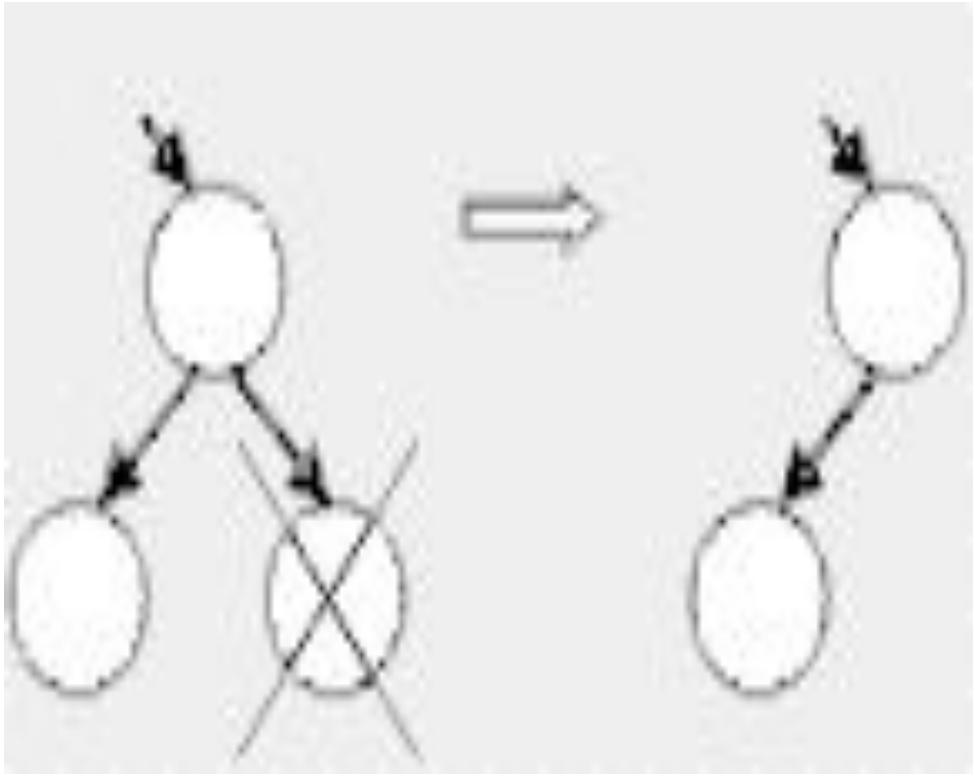
1. Сравним вставляемый ключ с корнем, если ключ больше, чем ключ корня, и правый потомок есть, уходим в правое поддерево, а иначе, при условии существования левого потомка -в левое. Если потомков нет – дошли до позиции вставки.
2. Выполняем пункт 1, пока не дойдем до позиции вставки.
3. Сравниваем вставляемый ключ с ключом найденного узла. Если ключ меньше ключа найденного узла, то добавляем листу левого сына, а иначе – правого сына. Например, необходимо вставить в дерево, изображенное на рисунке, узел с ключом 5.

Пример вставки ключа 5



Удаление узла
Рассмотрим ситуации

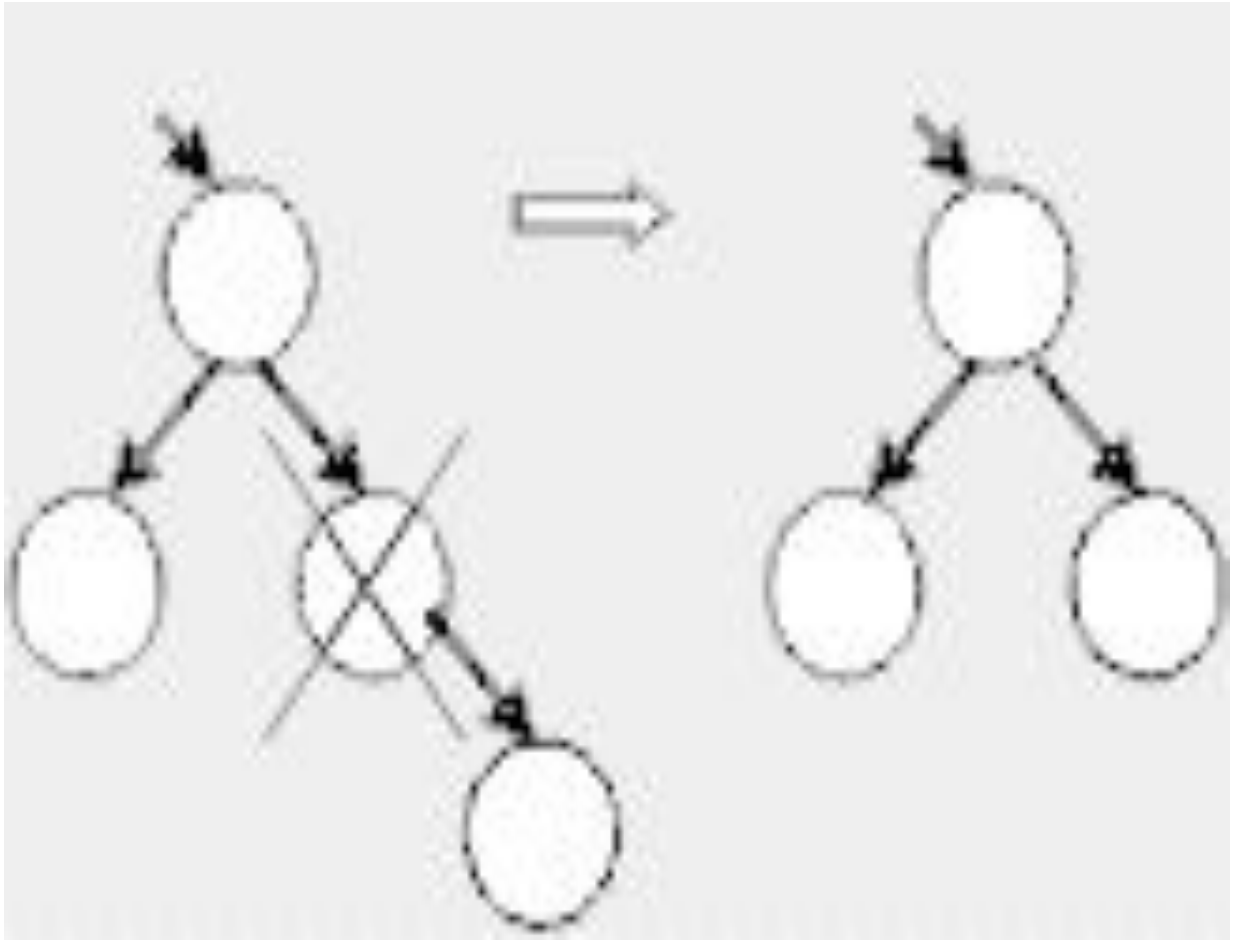
- Удаление листа



Описание ситуации(нет потомков)

- Если узел является конечным (то есть не имеет потомков), то его удаление не вызывает трудностей, достаточно обнулить соответствующий указатель узла-родителя

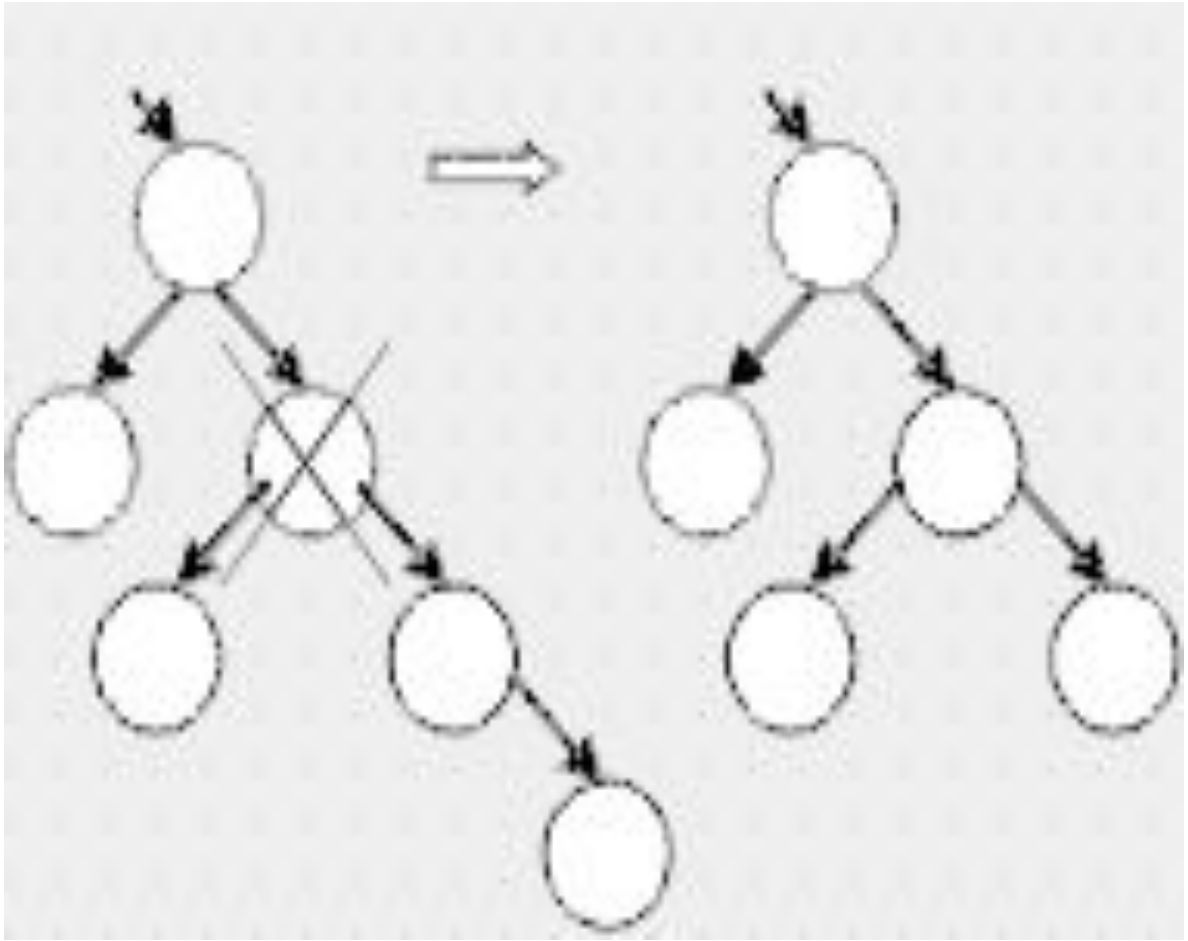
Удаление узла, имеющего одного потомка



Описание ситуации(1 потомок)

- Потомок удаляемого узла становится тем же потомком отца удаляемого узла, каким был удаляемый узел

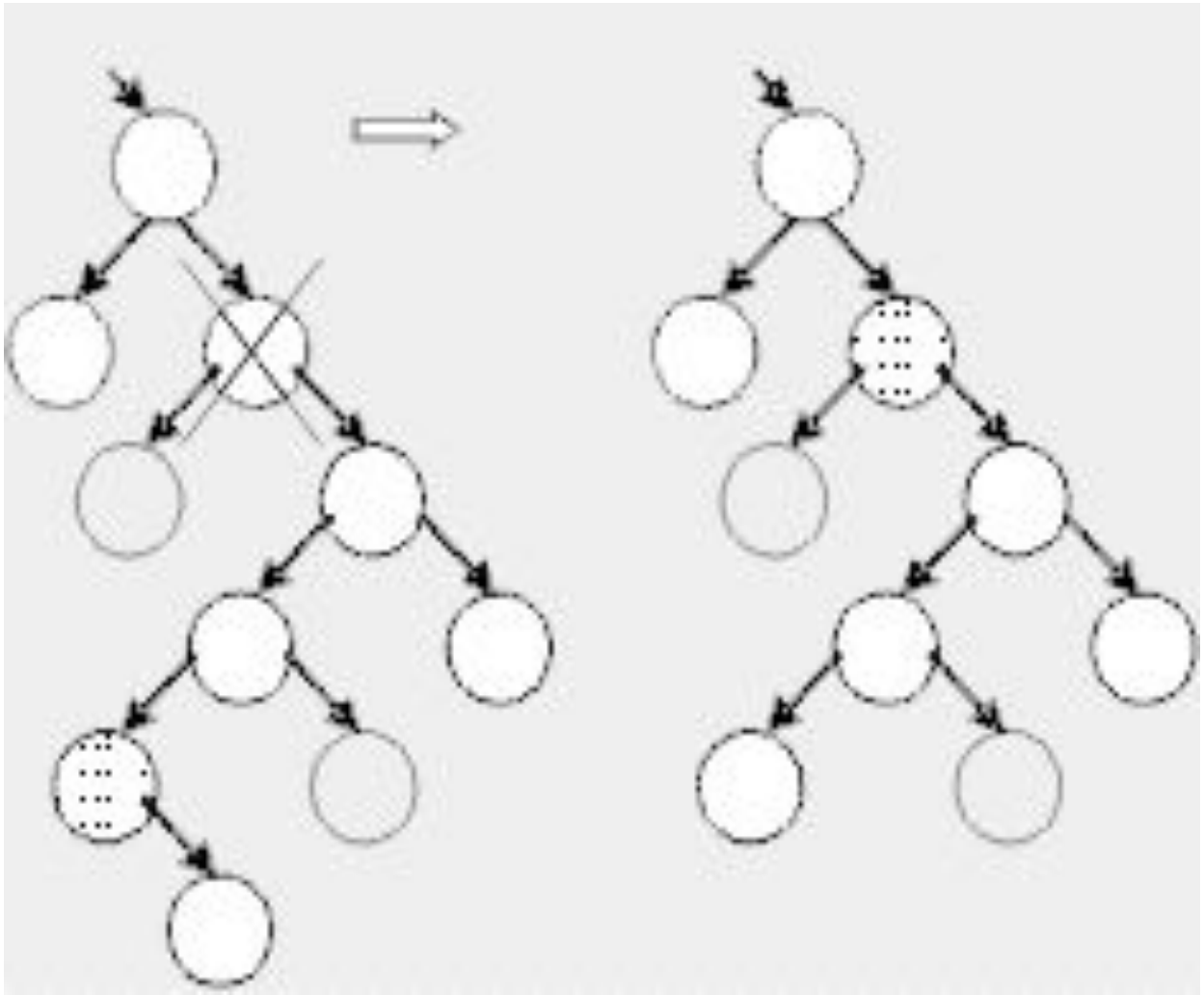
Удаление узла с двумя потомками (1-простая ситуация)



Описание ситуации

- Есть простой особый случай: если у правого потомка удаляемого узла нет левого потомка, удаляемый узел заменяется на своего правого потомка, а его левый потомок подключается вместо отсутствующего левого потомка к замещающему узлу.

Удаление узла с двумя потомками (2)



Описание ситуации

- В общем же случае на место удаляемого узла ставится самый левый лист его правого поддерева (или наоборот – самый правый лист его левого поддерева). Это не нарушает свойств дерева поиска.
- Корень дерева удаляется по общему правилу за исключением того, что заменяющий его узел не требуется присоединять к узлу-родителю.

Комментарии к удалению(1)

- В функцию **del** передаются указатель **root** на корень дерева и ключ **key** удаляемого элемента. С помощью функции **find** определяются указатели на удаляемый элемент **p** и его предка **parent** . Если искомого элемента в дереве нет, то выдается сообщение ({6}) .

Комментарии к удалению(2)

- В операторах определяется указатель на узел **y** , который должен заменить удаляемый. Если у узла **p** нет левого поддерева, на его место будет поставлена вершина (возможно пустая) его правого поддерева.
- Иначе, если у узла **p** нет правого поддерева, на его место будет поставлена вершина его левого поддерева.
- В противном случае, когда оба поддерева существуют, для определения замещающего узла вызывается функция **spusk** , выполняющая спуск по дереву.

Комментарии к удалению(3)

- В функции **spusk** первым делом проверяется особый случай, описанный выше. Если же этот случай (отсутствие левого потомка у правого потомка удаляемого узла) не выполняется, организуется цикл, на каждой итерации которого указатель на текущий элемент запоминается в переменной **pred** , а указатель **y** смещается вниз и влево до того момента, пока не станет ссылаться на узел, не имеющий левого потомка (он-то нам и нужен).

Комментарии к удалению(4)

- В операторе к этой пустующей ссылке присоединяется левое поддерево удаляемого узла. Перед тем как присоединять к этому узлу правое поддерево удаляемого узла, требуется «пристроить» его собственное правое поддерево. Мы присоединяем его к левому поддереву предка узла y , заменяющего удаляемый, поскольку этот узел перейдет на новое место.
- Функция spusk возвращает указатель на узел, заменяющий удаляемый.

Комментарии к удалению(5)

- Если мы удаляем корень дерева, надо обновить указатель на корень, иначе – присоединить этот указатель к соответствующему поддереву предка удаляемого узла.
- После того как узел удален из дерева, освобождается занимаемая им память.