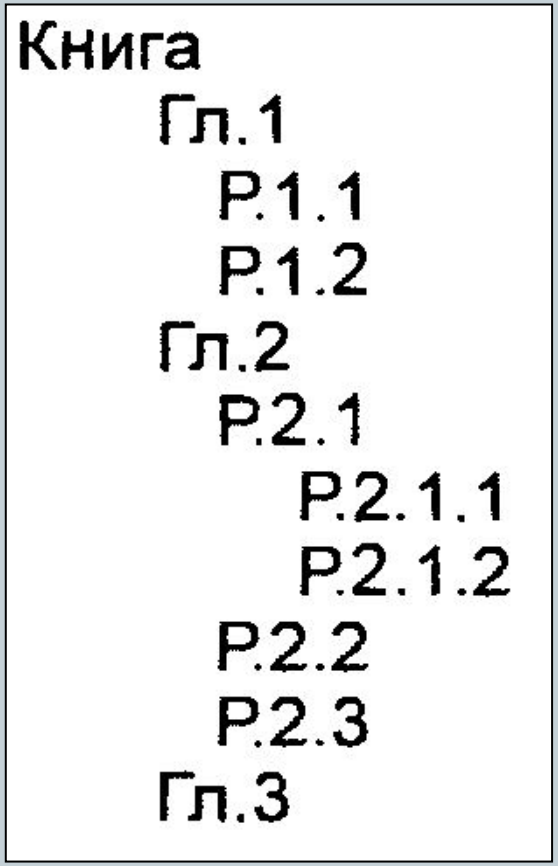
Лекция 5

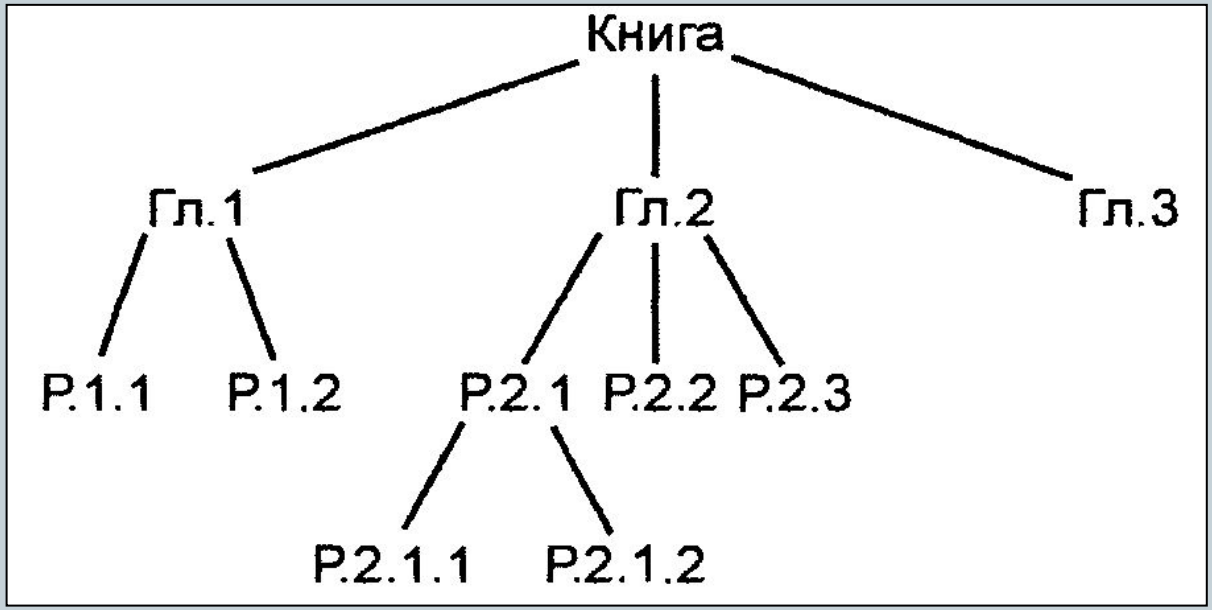
А.Ф. ЗУБАИРОВ

Дерево

- Дерево – совокупность элементов, называемых *узлами,* и отношений, образующих иерархическую структуру узлов.
- Дерево (tree) – конечное множество T одного или более узлов со следующими свойствами:
  - существует один выделенный узел – корень (root) данного дерева T;
  - остальные узлы (за исключением корня) распределены среди $m \geq 0$ непересекающихся множеств $T_1 \ldots T_m$, и каждое из этих множеств в свою очередь является деревом; деревья $T_1 \ldots T_m$ являются поддеревьями (subtree) данного корня.

Дерево

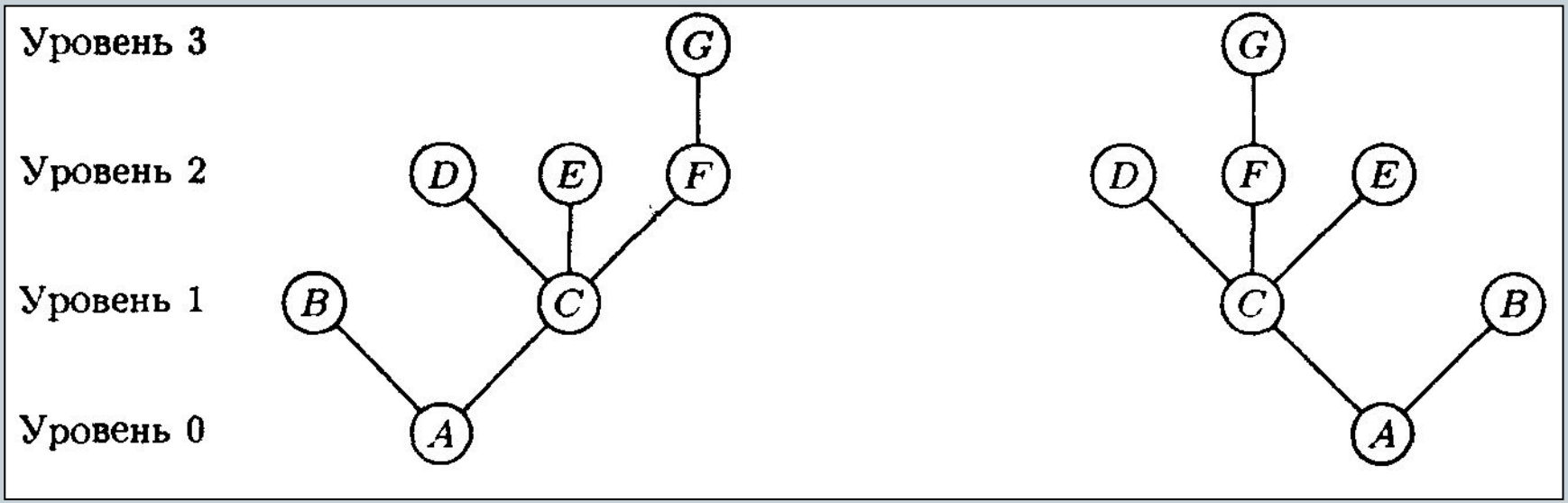

Книга
Гл.1
Р.1.1
Р.1.2
Гл.2
Р.2.1
Р.2.1.1
Р.2.1.2
Р.2.2
Р.2.3
Гл.3


Книга
Гл.1
Гл.2
Гл.3
Р.1.1
Р.1.2
Р.2.1
Р.2.2
Р.2.3
Р.2.1.1
Р.2.1.2

Дерево

- Каждый узел дерева является корнем некоторого поддерева.
- *Степень* (degree) узла – количество поддеревьев узла.
- *Концевой узел* (terminal node) или *лист* (leaf) – узел со степенью 0.
- *Узел ветвления* (branch node) – неконцевой узел.
- *Уровень* (level) узла по отношению к дереву T: уровень корня дерева T равен 0, а уровень любого другого узла на 1 выше, чем уровень корня ближайшего поддерева T, содержащего данный узел.

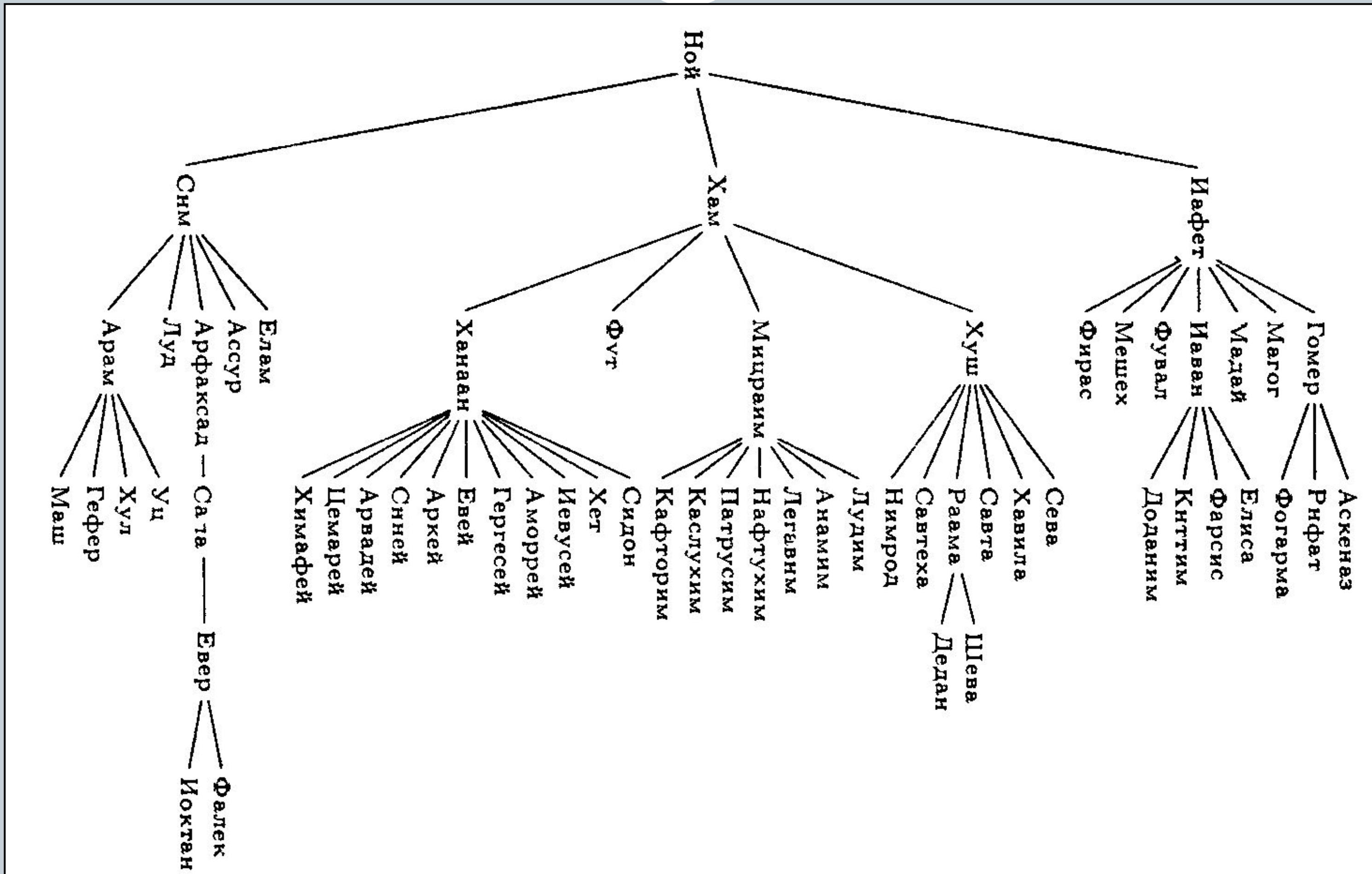
Дерево



Деревья

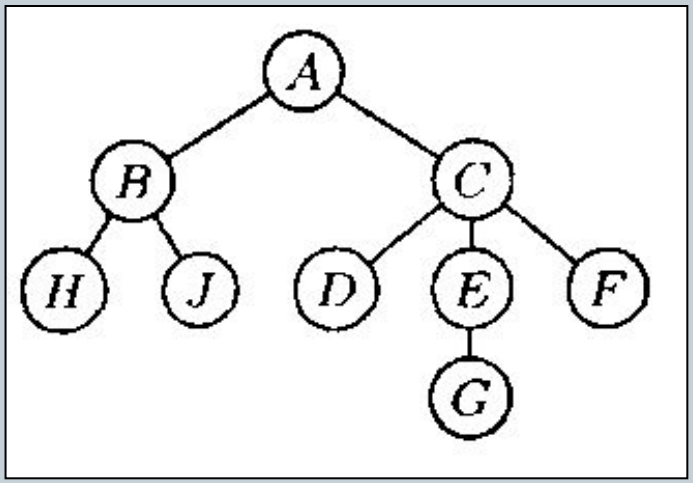
- Путь из узла n_1 в узел n_k – последовательность узлов n_1, n_2, ..., n_k, где для всех i, $1 \leq i \leq k$ узел n_i является родителем узла n_{i+1}.
- Длина пути – число, на единицу меньшее числа узлов, составляющих этот путь.

Деревья


Ной
Сим
Хам
Иафет
Арам
Луд
Арфаксад
Ассур
Елам
Маш
Гефер
Хул
Уц
Сала
Евер
Иоктан
Фалек
Ханаан
Фут
Мицраим
Хуш
Химафей
Цемарей
Арвадей
Синей
Аркей
Евей
Гергесей
Аморрей
Иевусей
Хет
Сидон
Кафторим
Каслухим
Патрусим
Нафтухим
Легавим
Анамим
Лудим
Нимрод
Савтеха
Раама
Савта
Хавила
Сева
Дедан
Шева
Фирас
Мешех
Фувал
Иаван
Мадай
Магог
Гомер
Доданим
Киттим
Фарсис
Елиса
Фогарма
Рифат
Аскеназ

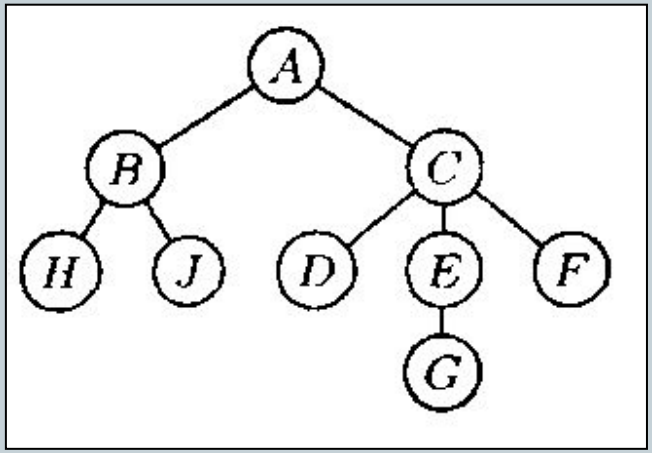
Дерево

- Если существует путь из узла *a* в узел *b*, тогда *a* называется *предком (родителем)* узла *b*, а узел *b* – потомком (ребенком) узла *a*.

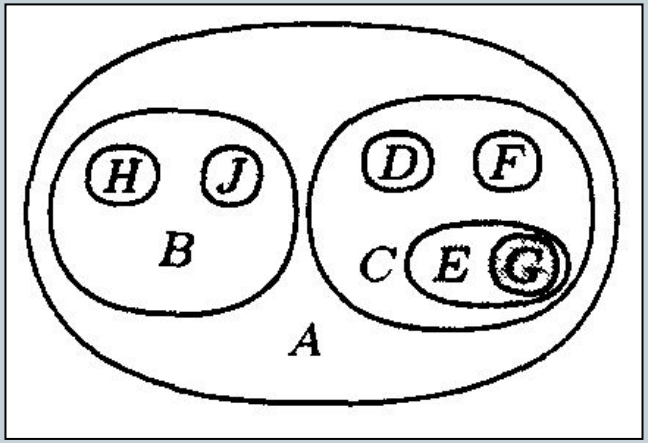


- Высота узла дерева – длина самого длинного пути из этого узла до какого-либо листа.
- Высота дерева совпадает с высотой корня.
- Глубина узла – длина пути от корня до узла.

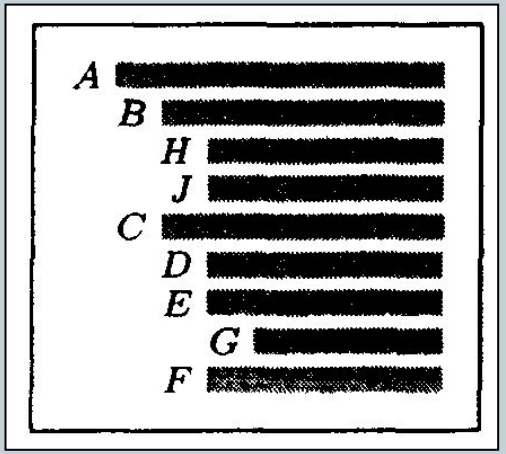
Способы представления деревьев



Ориентированное дерево
(oriented tree)



Вложенные множества
(nested sets)



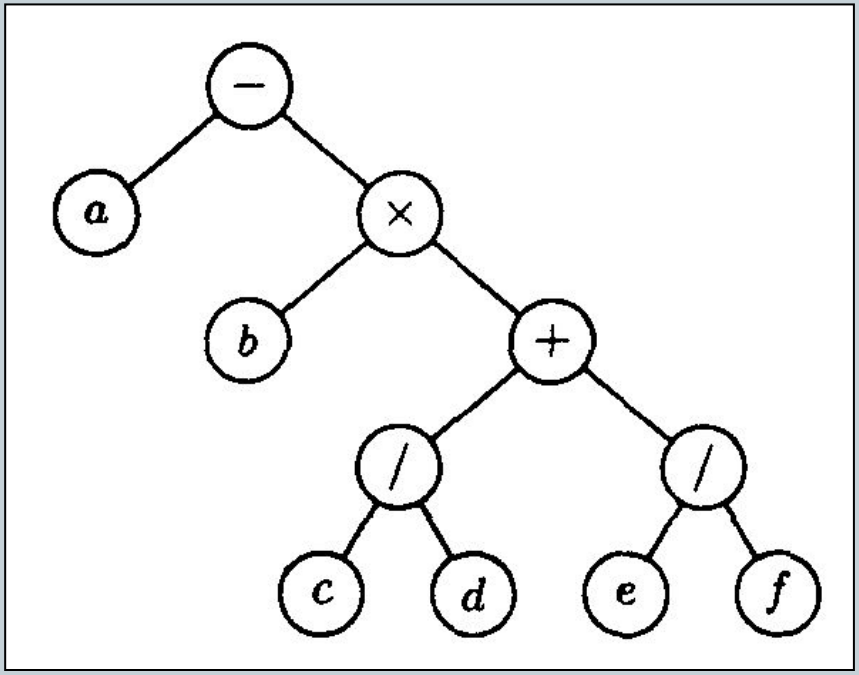
Список с отступами
(indentation)

(A(B(H)(J))(C(D)(E(G))(F)))

Вложенные скобки

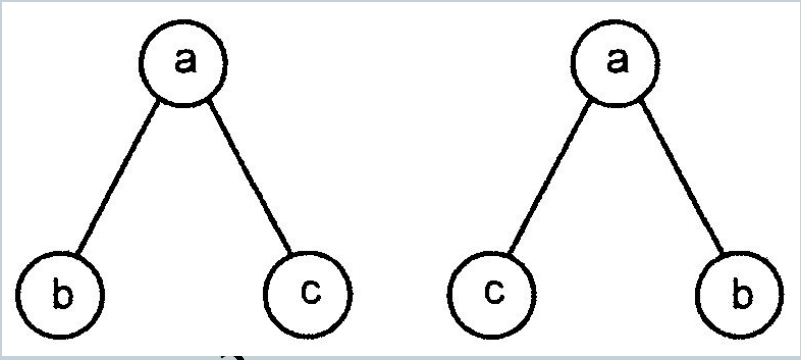
Способы представления деревьев

- *a-b(c/d+e/f)*



Порядок узлов

- Сыновья узла упорядочиваются слева направо. Следующие два дерева различны, так как порядок сыновей узла *a* различен.



- *Неупорядоченное дерево* – дерево, в котором порядок сыновей игнорируется. Иначе дерево называется *упорядоченным*.

Бинарные деревья

- *Бинарное дерево* – конечное множество узлов, которое является пустым или состоит из корня и двух непересекающихся бинарных деревьев, которые называются левым и правым поддеревьями данного дерева.
- Остальные деревья называют *сильноветвящимися*.

Представление деревьев в компьютере

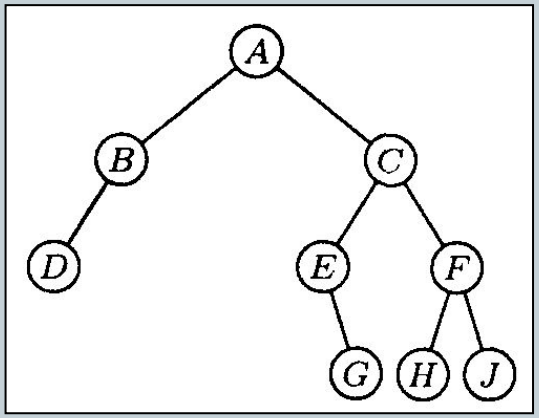
```
typedef struct treeNode {
    int info;
    struct treeNode *llink, *rlink;
} treeNode;
```

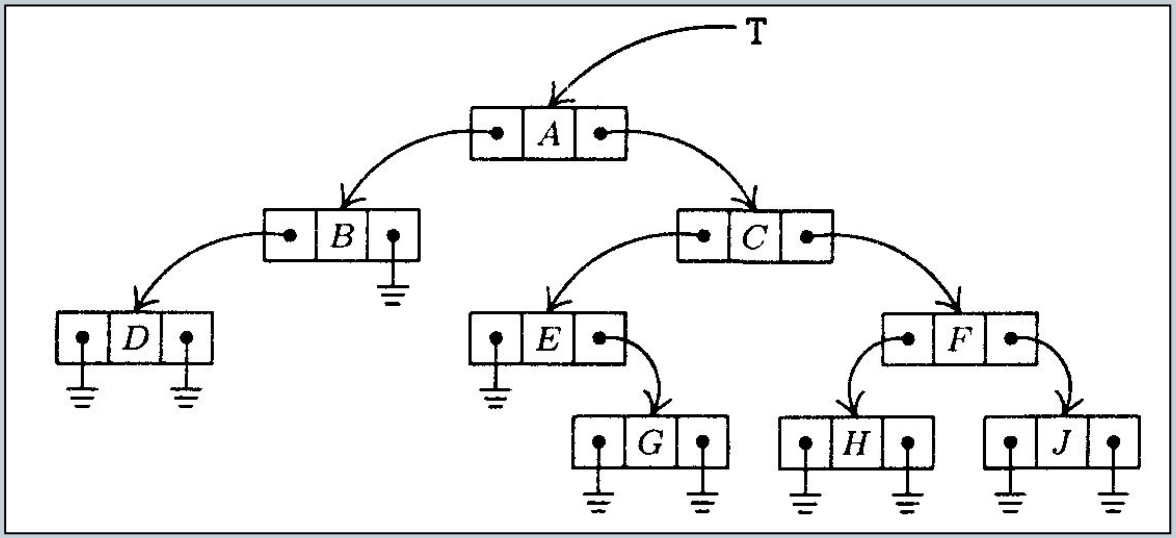
`treeNode *t` – переменная связи, указатель на дерево.

`llink` – указатель на левое поддерево;

`rlink` – указатель на правое поддерево.

Представление деревьев в компьютере



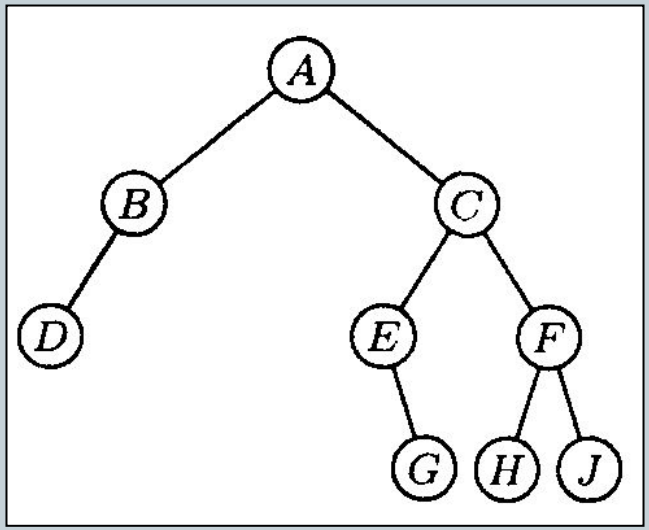


Обход (traversing) дерева

- *Обход дерева (проход по дереву)* – метод исследования дерева, когда каждый узел посещается в точности один раз.
- Три способа обхода:
  - в прямом порядке (preorder);
  - в обратном порядке (postorder);
  - во внутреннем (центрированном) порядке, симметричный обход (inorder);

Прямой порядок обхода

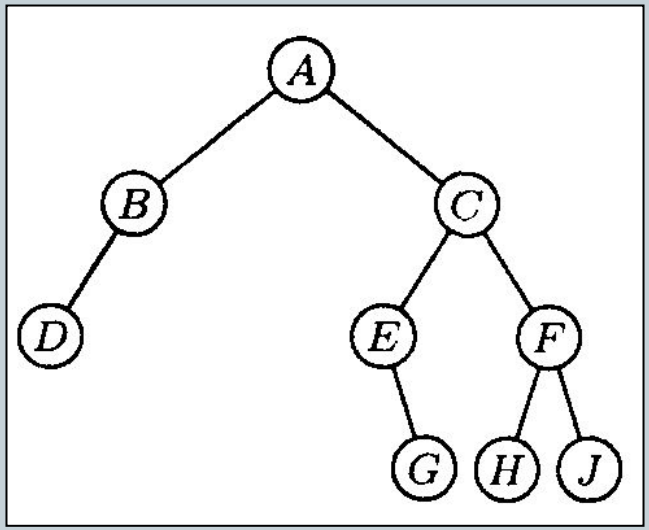
- 1) Попасть в корень;
- 2) Пройти левое поддерево;
- 3) Пройти правое поддерево;



A B D C E G F H J

Внутренний порядок обхода

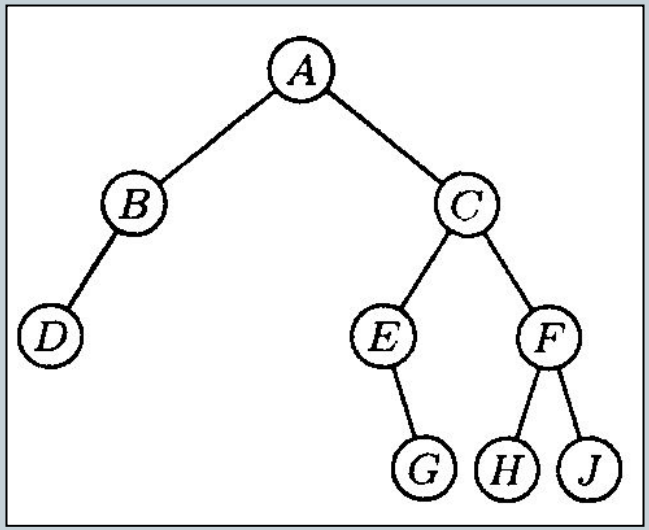
- 1) Пройти левое поддерево;
- 2) Попасть в корень;
- 3) Пройти правое поддерево;



D B A E G C H F J

Обратный порядок обхода

- 1) Пройти левое поддерево;
- 2) Пройти правое дерево;
- 3) Попасть в корень;



D B G E H J F C A

Рекурсивный алгоритм прямого обхода

```
подпрограмма preorder(дерево T) {
    если T != NULL {
        занести в список обхода узел t
        preorder(llink)
        preorder(rlink)
    }
}
```

Рекурсивный алгоритм обратного обхода

```
подпрограмма postorder(дерево T) {
    если T != NULL {
        postorder(llink)
        postorder(rlink)
        занести в список обхода узел t
    }
}
```

Рекурсивный алгоритм внутреннего обхода

```
подпрограмма inorder(дерево T) {
    если T != NULL {
        inorder(llink)
        занести в список обхода узел t
        inorder(rlink)
    }
}
```

Дерево как АТД

- К деревьям как к АТД применяются следующие операторы:
- `parent(n, t)` – эта функция возвращает родителя (parent) узла n в дереве T. Если n является корнем, который не имеет родителя, то в этом случае возвращается NULL.
- `leftmostChild(n, T)`. Данная функция возвращает самого левого сына узла n в дереве T. Если n является листом (и поэтому не имеет сына), то возвращается NULL.
- `rightSibling(n, T)`. Эта функция возвращает правого брата узла n в дереве T и значение NULL, если такового не существует. Для этого находится родитель p узла n и все сыновья узла p, затем среди этих сыновей находится узел, расположенный непосредственно справа от узла n.
- `createi(v, T1, T2, ..., Ti)` — это обширное семейство функций, которые для каждого i = 0, 1, 2, ... создают новый корень r с меткой v и далее для этого корня создает i сыновей, которые становятся корнями поддеревьев T1, T2, ..., Ti. Эти функции возвращают дерево с корнем r. Отметим, что если i = 0, то возвращается один узел r, который одновременно является и корнем, и листом.