

# Программирование



*Деревья*



*Морис Эшер*

Три мира

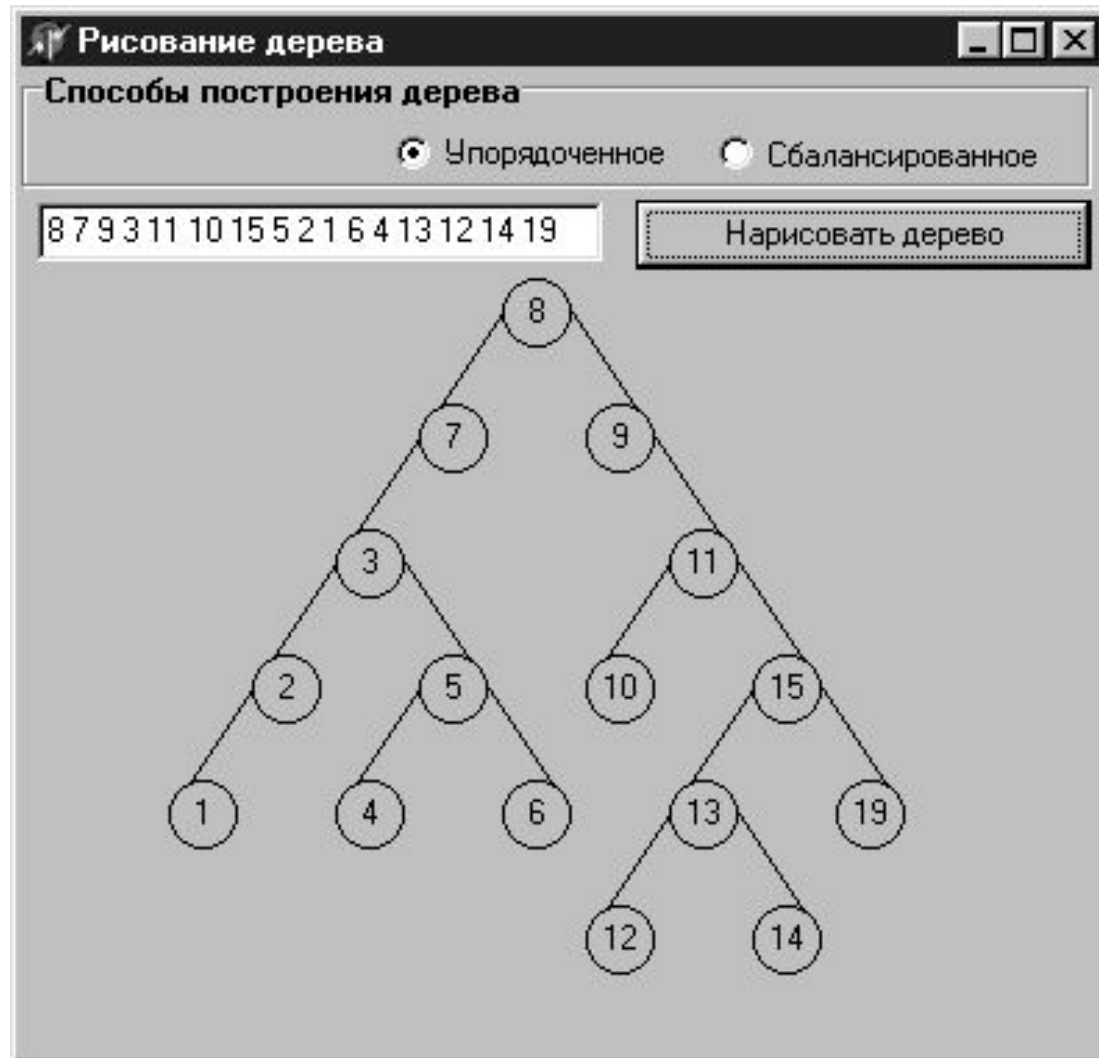
# Деревья --

структуры данных, определяемые с помощью рекурсии.

Древовидная структура (дерево) определяется следующим образом:  
*дерево (tree)* с базовым типом  $T$  — это:

- либо пустая структура;
- либо узел типа  $T$ , с которым связано конечное число древовидных структур, называемых *поддеревьями (subtree)*.

Если с узлом связаны только два поддерева, то дерево называется *бинарным*.



# Терминология (1)

Из ботаники взяты такие определения:

- **узел** (*node*) — это точка, где может возникнуть ветвь.
- **корень** (*root*) — "верхний" узел дерева.;
- **ветвь** (*branch*) — отрезок, описывающий связь между двумя узлами;
- **лист** (*leaf*) — узел, из которого не выходят ветви, т. е. не имеющий поддеревьев.

# Терминология (2)

*Термины, взятые из генеалогии, описывают отношения:*

- **родительским** (*parent*) называется узел, который находится непосредственно над другим узлом;
- **дочерним** (*child*) называется узел, который находится непосредственно под другим узлом; дочерний узел также называют сыном (что не меняет "родственности" отношений);
- **предки** данного узла — это все узлы на пути вверх от данного узла до корня;
- **потомки** — все узлы, расположенные ниже данного;
- **сестринские (братские)** — узлы, у которых один и тот же родитель.

# Терминология (3)

*Список терминов, возникших в программировании:*

- **внутренний узел** (*internal node*) — узел, не являющийся листом;
- **порядок узла** (*node degree*) — количество его дочерних узлов;
- **глубина** (*depth*) **узла** — количество его предков плюс единица;
- **глубина** (*высота*) **дерева** — максимальная глубина всех узлов;
- **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. Она также называется длиной внутреннего пути.

# Основные операции с бинарными деревьями

При работе программы дерево может модифицироваться: добавляются или удаляются узлы, меняются информационные части узлов. То есть дерево является динамической структурой.

Поэтому узел дерева определяется как переменная с фиксированной структурой, содержащей информационную часть и две ссылки, указывающие на левое и правое поддеревья данного узла.

Ссылка на пустое дерево равна *null*.



# Описание узла дерева

см. *TreeLevel*

# Алгоритмы обхода дерева

Такой алгоритм — это метод, позволяющий получить ***доступ к каждому узлу дерева один и только один раз.***

Для каждого узла выполняются некоторые виды обработки (проверка, суммирование и т. п.), однако **способ обхода не зависит от конкретных действий и является общим для всех алгоритмов обработки узлов.**

# Существуют три способа посещения всех узлов, использующие **обход в глубину:**

*Обход\_сверху\_вниз (PreOrder):*

- обработать корень;
- *Обход\_сверху\_вниз* левого поддерева;
- *Обход\_сверху\_вниз* правого поддерева.

*Обход\_слева\_направо (InOrder):*

- *Обход\_слева\_направо* левого поддерева;
- обработать корень;
- *Обход\_слева\_направо* правого поддерева.

*Обход\_снизу\_вверх (PostOrder):*

- *Обход\_снизу\_вверх* левого поддерева;
- *Обход\_снизу\_вверх* правого поддерева;
- обработать корень.

# Упорядоченное дерево

*Упорядоченным* называется дерево, в котором для каждого узла  $N$  значение левого дочернего узла *меньше*, чем значение в  $N$ , а значение правого дочернего узла *больше* значения в  $N$ .

Если в дереве могут содержаться одинаковые значения, то программист должен сам определить, влево или вправо помещать значение, равное значению в родительском узле, т. е. соответствующее строгое неравенство заменить на нестрогое.

# Сбалансированное дерево

Дерево называется *идеально сбалансированным*, если для каждого его узла количества узлов в левом и в правом его поддеревьях различаются не более чем на 1.

Разумеется, идеально сбалансированное дерево имеет минимальную высоту по сравнению со всеми другими деревьями с тем же числом узлов..

Чтобы построить такое дерево, нужно располагать максимально возможное число узлов на всех уровнях, кроме самого нижнего. Это сделать просто, если распределять узлы поровну слева и справа от каждого узла.

# Построение сбалансированного дерева

Алгоритм равномерного распределения при известном числе узлов  $N$  лучше всего выбрать рекурсивным:

Взять один узел в качестве корня.

Построить левое поддерево с числом узлов  $nLeft = N / 2$  тем же способом.

Построить правое поддерево с числом узлов  $nRight = N - nLeft - 1$  тем же способом.

см. *TreeLevel*