ДИНАМИЧЕСКИЕ ДАННЫЕ РАЗВЕТВЛЕННОЙ СТРУКТУРЫ

ДЕРЕВЬЯ, ДВОИЧНЫЕ ДЕРЕВЬЯ

Основные определения

Дерево - это частный случай графа, между любыми двумя вершинами которого существует ровно один путь.

Ориентированное дерево - граф, в котором между любыми двумя вершинами существует не более одного пути.

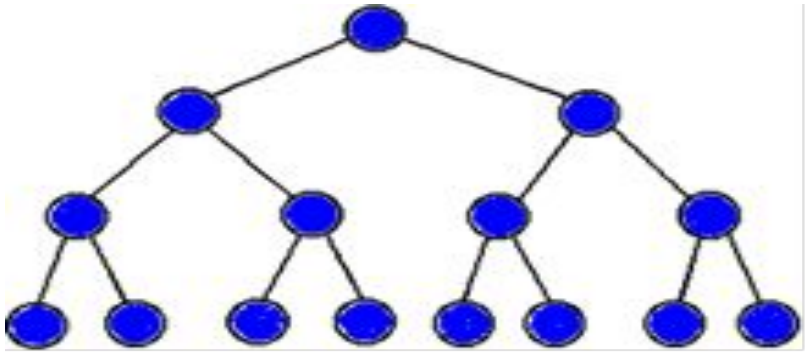
Будем изучать только один вид ориентированных деревьев – корневые.

Корневое дерево

- это ориентированное дерево, в котором можно выделить вершины трех видов: *корень*, *листья* и остальные вершины, причем должны выполняться два обязательных условия:

- из листьев не выходит ни одна дуга; из других вершин может выходить сколько угодно дуг;
- в корень не заходит ни одна дуга; во все остальные вершины заходит ровно по одной дуге.

Традиционно в математике и в родственных ей науках (в том числе и в теоретическом программировании) деревья "растут" вниз головой: это делается просто для удобства наращивания листьев в случае необходимости. Таким образом, на рисунках корень дерева оказывается самой верхней вершиной, а листья - самыми нижними.



Дерево высоты 3

Определения

Предок вершины v - это вершина, из которой исходит дуга, заходящая в вершину v.

Потомок вершины *v* - это вершина, в которую заходит дуга, исходящая из вершин*ы* v.

В этих терминах можно дать другие определения понятиям корень и лист: у корня нет предков, у *листа* нет потомков.

Бинарное дерево - это корневое дерево, каждая вершина которого имеет не более двух потомков. В таком случае иногда говорят о **левом потомке** и **правом потомке** для текущей вершины.

Высота корневого дерева - это максимальное количество дуг, отделяющих листья от корня. (Будем считать, что корень дерева расположен на уровне 0.)

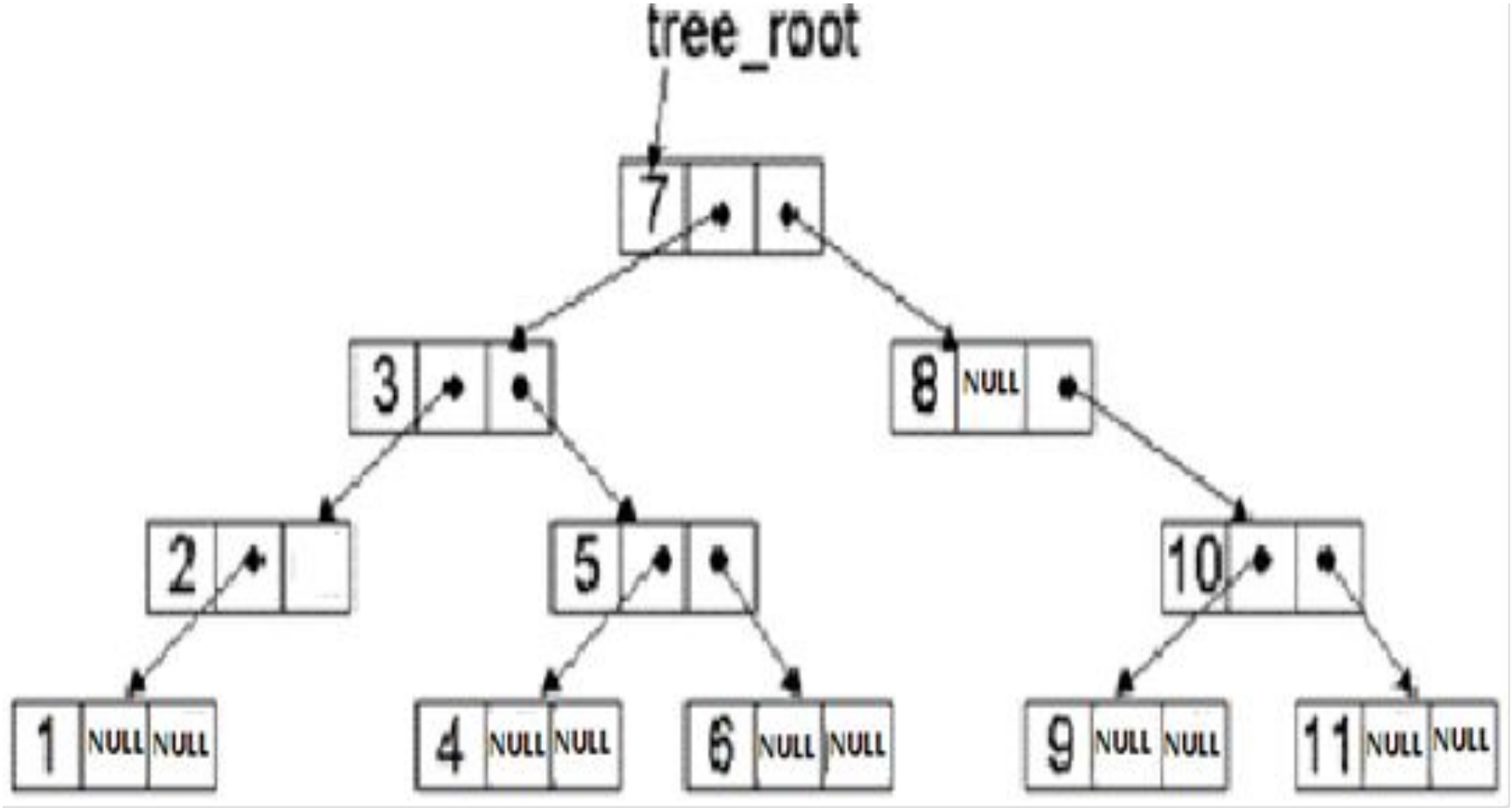
Дерево двоичного поиска

Дерево двоичного поиска для множества чисел S - это бинарное дерево, каждой вершине которого сопоставлено число из множества S, причем:

- существует ровно одна *вершина*, содержащая любое число из множества S;
- все значения вершин левого поддерева строго меньше, чем значение текущей вершины;
- все значения вершин правого поддерева строго больше, чем значение текущей вершины.

Т.е. структура дерева двоичного поиска подчиняется простому правилу: "если больше - направо, если меньше - налево".

Пример двоичного дерева поискадля набора чисел 7, 3, 5, 2, 8, 1, 6, 10, 9, 4, 11



Описание структуры «Дерево»

```
struct Elem
{
    int data;
    Elem * left, * right;
};
typedef Elem * PElem;
```

Создание новой вершины дерева

```
PElem Create ()
{
 PElem b = new Elem;
 b->left   = NULL ;
 b->right = NULL ;
 return b;
}
```

Создание новой вершины дерева с занесением в вершину значения

```
PElem Create (int x)
{
 PElem b = new Elem;
 b->left   = NULL ;
 b->right = NULL ;
 b->data  = x;
 return b;
 }
```

Задача 1.
Построение дерева поиска

I. Создать переменную-указатель на дерево

II. **Пока** не достигли конца ввода:

1. Взять из входного выражения очередной элемент, установить указатель на корень дерева.
2. Вызвать алгоритм занесения элемента в дерево

Алгоритм занесения

Если дерево пусто,

- то создать корневую вершину дерева, записать в нее этот символ, оформить все ссылки.
- иначе если элемент меньше значения узла дерева,
  - то вызвать алгоритм для левого поддерева
  - иначе вызвать алгоритм для правого поддерева

Печать дерева

I. Встать в корень дерева
II. Вызвать алгоритм печати дерева
    1. Распечатать содержимое узла
    2. Вызвать алгоритм печати левого поддерева
    3. Вызвать алгоритм печать правого поддерева

Задание 1

1. Создайте дерево поиска
2. Меняя местами пункты 1, 2 и 3 можно получить принципиально разные выводы содержимого дерева. В чем они заключаются? Распечатайте дерево различными способами и сделайте выводы.
3. Распечатайте дерево в виде «ярусов», т.е. сделайте его похожим на дерево.