# Задание 4. Двоичные деревья.

Требуется реализовать формирование узлов двоичного дерева поиска следующими способами:

1. Классическая вставка (процедура Tree\_Insert) – новый узел становится листом дерева.
2. Вставка в корень (процедура Tree\_Root\_Insert, см. пояснения) – новый узел становится корнем дерева.
3. Вставка и удаление узлов алгоритмом «Красно-черного дерева» (одна из разновидностей сбалансированного двоичного дерева). Проверить, что при формировании дерева как из случайной последовательности, так и из упорядоченной последовательности высоты всех ветвей приблизительно равны.

При завершении программа должна корректно уничтожать дерево, освобождая занятую динамическую память.

**Пояснения к п.2**. Вставка узлов в корень характерна тем, что последние узлы сосредоточены вблизи вершины дерева. Зачастую это свойство оказывается полезным, т.к. в приложениях нередко встречается ситуация, когда необходимо осуществлять поиск в основном последних узлов. Алгоритм вставки в корень основан на рекурсии и использует операции поворота:

Tree\_Root\_Insert (y, z)

1. if y = NIL then
2. root[T] ← z
3. else if key[z] < key[y] then
4. if left[y] = NIL then
5. left[y] ← z
6. p[z] ← y
7. else Tree\_Root\_Insert(left[y], z)
8. Right\_Rotate(T, y)
9. else if right[y] = NIL then
10. right[y] ← z
11. p[z] ← y
12. else Tree\_Root\_Insert(right[y], z)
13. Left\_Rotate(T, y)

Вставка узла x осуществляется вызовом Tree\_Root\_Insert(root[T], x).

Результат вставки в дерево ключей 22, 4, 16, 19, 15, 8, 23, 8, 22, 23:

Tree\_Insert

Tree\_Root\_Insert

Замечание: для контроля структуры дерева удобно организовать вывод дерева при помощи прямого обхода (Preorder\_Tree\_Work), печатая для каждого узла строку вида , , . Например, для изображенного выше дерева (Tree\_Insert) вывод будет выглядеть так:

22 4 23

4 NIL 16

16 15 19

15 8 NIL

8 NIL 8

8 NIL NIL

19 NIL NIL

23 22 23

22 NIL NIL

23 NIL NIL