**Лабораторная работа №7**

(Работа с бинарным деревом)

struct tree //узел бинарного дерева  
{

int data, //числовое значение  
count; //кол-во повторений

tree\* left, //указатель на левое поддерево  
\* right; //указатель на правое поддерево

};

**Блок-схема main()**





**Функция delete\_node()**

**1. НАЧАЛО**

2. Входные значения: *root* — указатель на узел(корень) дерева, k — значение удаляемого узла.

Выходные значения: *t1* — указатель на наследника удалённого узла (на *root*, если удаляемое значение не было найдено).

3. Поиск узла. Если *k* **больше** значения текущего узла *root->data*, то рекурсивно запустить функцию *delete\_node()* для правого поддерева *root->right*.

4. Если k **меньше** значения текущего узла *root->data*, то рекурсивно запустить функцию *delete\_node()* для левого поддерева *root->left*. Если узел так и не был найден, перейти к **пункту 8**, иначе — **к пункту 5**.

5. Если *root* не имеет наследников, то очистить память для *root*, вернуть нулевой указатель родителю, перейти к **пункту 8**, иначе — перейти к **пункту 6**.

6. Если *root* имеет только одного наследника, то исключить *root* из дерева, вернуть указатель на наследника, очистить память для *root*, перейти к **пункту 8**, иначе — перейти к **пункту 7**.

7. *Root* имеет двух наследников, тогда найти наименьшее (самое левое) значение в правом поддереве *root->right*, поменять местами с удаляемым узлом, рекурсивно запустить функцию *delete\_node()* для *root->right*.

**8. КОНЕЦ**