Write the Number of Nodes First in the File

How should we write the pyramid tree?

Start by writing the number of nodes.

Why?

When we read the tree,
- we need to **dynamically allocate**
- the **array of nodes**.
- But, to do so, we **need to know the size**.

Write the size first to make that task easier.

Given Tree and File Name, Try to Write ASCII File

Let's start the code:

1 on success, 0 on failure

the file name

the tree

```
int32_t write_pyr_tree_ASCII
    (pyr_tree_t* p, const char* fname)
{
    FILE* out;
    if (NULL == (out =
                 fopen (fname, "w"))) {
        return 0;
    }
    fprintf (out, "%d\n", p->n_nodes);
```


Open the File and Write the Number of Nodes

Let's start the code:

new stream

Open file for writing.

Failed? Give up.

Print number of nodes to stream.

```
int32_t write_pyr_tree_ASCII
    (pyr_tree_t* p, const char* fname)
{
    FILE* out;
    if (NULL == (out =
                 fopen (fname, "w"))) {
        return 0;
    }
    fprintf (out, "%d\n", p->n_nodes);
```


Write Contents of Nodes Distinctly for Internal/Leaf

What about the nodes?

For **internal nodes**, the **`id` field means nothing**.

So we can **write a node's contents as follows**:

```
<x> <y_left> <y_right>
```

For **leaf nodes**,
- **all fields are meaningful**,
- but, if we have the graph,
- we can find x and y position using **`id`**.

So, **for each leaf node**, we can **write**:

```
<id>
```

