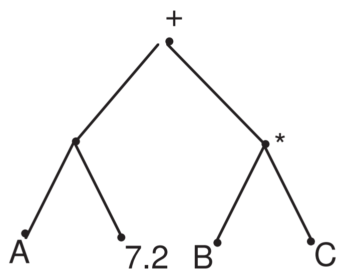
Traversing a tree

Application: given parse tree

A/7.2 + B*C ⇒ 

- evaluate expression, generate machine code

```
Load A
Div "7.2"
STORE temp
Load B
MULT C
ADD temp
```

This is the so-called Post order traversal.

Postorder traversal

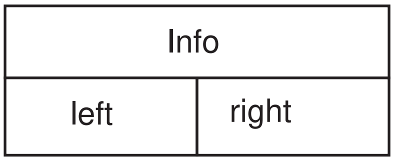
```
postorder(root);
   postorder(root^.left);
   postorder(root^.right);
   process (root);
```

Inorder traversal

```
inorder(root);
   inorder(root^.left);
   process (root);
   inorder(root^.right);
```

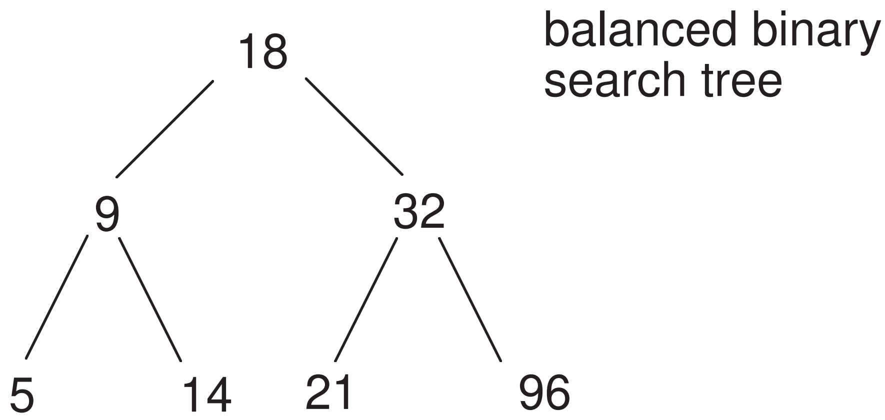
Preorder traversal

```
preorder(root);
   process (root);
   preorder(root^.left);
   preorder(root^.right);
```

a binary tree node: 

Representation of Trees: This depends on (1) operations to perform on the trees and (2) space availability

- **Array**. To represent balanced binary trees.



-

5	9	14	18	21	32	96

1	2	3	4	5	6	7

-

18	9	32	5	14	21	96

root

If the parent is at i then the two children are at $2i$, $2i + 1$ pointers are implicit

• **Use Pointers**: Suppose we have an arbitrary binary tree. We also want to be able to move from any node to successor/predecessor, and move around

We use pointers pointing both ways.

```
o <---------> o
```

Or simple let the pointers go one way (usually top down)

```
        o
       / \
      /   \
     o     o
    /
   /
  o
```

But then there will be alot of unused "nil" pointers. For a tree of n nodes, there are $2n-(n-1)=n+1$ unused pointer.

Threading.

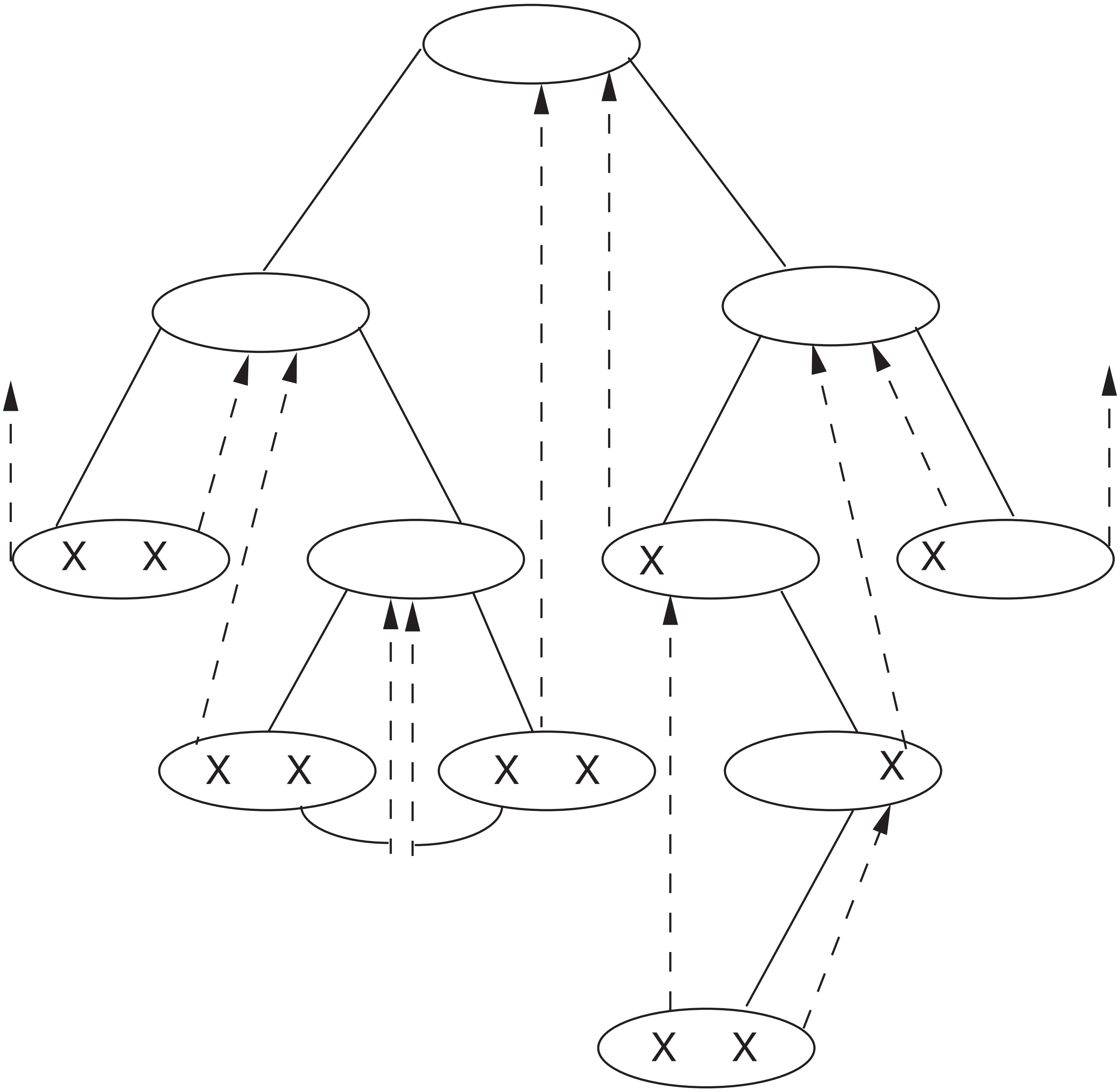
To utilize the unused pointers, we introduce the idea of “threading”

- ”unused” left pointers $\rightarrow$ inorder predecessor

- ”unused” right pointers $\rightarrow$ sucessor

Add flags to tell which pionters are the original tree pointers and which are threads.

See picture next page.



Successor: <u>if</u> right pointer is thread <u>then</u> follow thread <u>else</u> follow right pointer then go left till left pointer is thread.