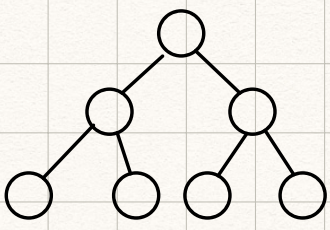
樹狀結構 Tree

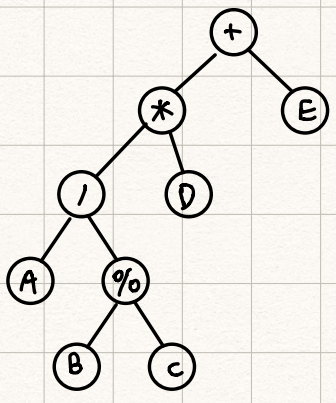
分支度、節點、Link 等專有名詞



二元樹 < 以一維陣列儲存
Linked List

※二元樹的走訪是種遞迴走訪，走訪過程是持續決定向左 or 向右走

中序追蹤：左子節點 → 根節點 → 右子節點
前序追蹤：根節點 → 左子節點 → 右子節點
後序追蹤：左子節點 → 右子節點 → 根節點



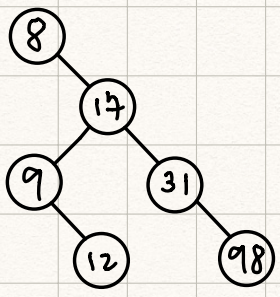
中序：A / B % C * D + E
中序順序就是由左至右每個元素跑一次

前序：+ * / A % B C D E
根節點在最前面

後序：A B C % / D * E +
根節點在最後面

利用中序走訪進行排序

小往左、大往右（可用該原則判斷中序順序）



8 . 17 . 31 . 9 . 98 . 12

二個Class：Node, BTree

```
class Node
{
    public:
        Node *nextR, *nextL;
        int Data;

        Node(int i)
        {
            nextR = nextL = NULL;
            Data = i;
        }
};

class BTree
```

```
{
  public:
      Node *root;
      void addTreeNode(int i);
      void preVisit(Node *);
      void inVisit(Node *);
      void postVisit(Node *);
      BTree(void)
      { root = NULL; }
};

void BTree :: addTreeNode(int i)
{
  Node *newNode = new Node(i);
  if(root == NULL)
  {
      root = newNode;
      return;
  }
  Node * tmpNode = root;

  while(1)
  {
      if(newNode->Data < tmpNode->Data)
      {
                  = i
```

```
            if ( tmpNode -> nextL != NULL)
                tmpNode = tmpNode -> nextL;
            else
            {
                tmpNode -> nextL = newNode;
                return;
            }
        }
        else
        {
            if ( tmpNode -> nextR != NULL)
                tmpNode = tmpNode -> nextR;
            else
            {
                tmpNode -> nextR = newNode;
                return;
            }
        }
    }
}
```

可用 Code 來想 Order

```
void BTree :: preVisit( Node * Tree )
{
    if ( Tree != NULL ) // 遞迴停止條件
    {
```

```
        cout << Tree -> Data << endl;
        preVisit( Tree -> nextL );
        preVisit( Tree -> nextR );
    }
}

void BTree :: inVisit( Node * Tree)
{
    if( Tree != NULL)
    {
        inVisit( Tree -> nextL );
        cout << Tree -> Data << endl;
        inVisit( Tree -> nextR);
    }
}

void BTree :: postVisit( Node * Tree)
{
    if( Tree != NULL)
    {
        postVisit( Tree -> nextL);
        postVisit( Tree -> nextR);
        cout << Tree -> Data << endl;
    }
}
```