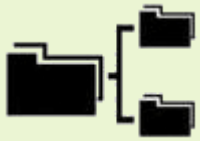
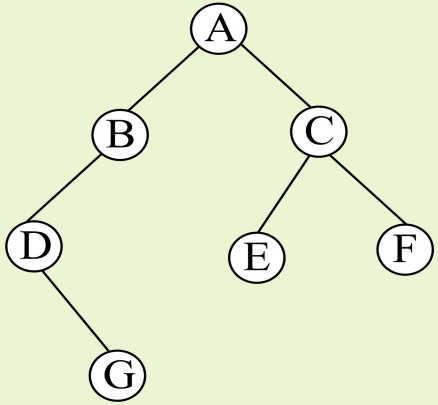


本节主题：

二叉树的遍历

二叉树的遍历操作

- 二叉树的遍历
  - 按照一定次序访问树中所有节点，并且每个节点仅被**访问**一次的过程。
  - 遍历是最基本的运算，是树中所有其他运算的基础。
- 二叉树三种遍历
  - 先序遍历：根节点-->左子树-->右子树。
  - 中序遍历：左子树-->根节点-->右子树。
  - 后序遍历：左子树-->右子树-->根节点



ABDGCEF

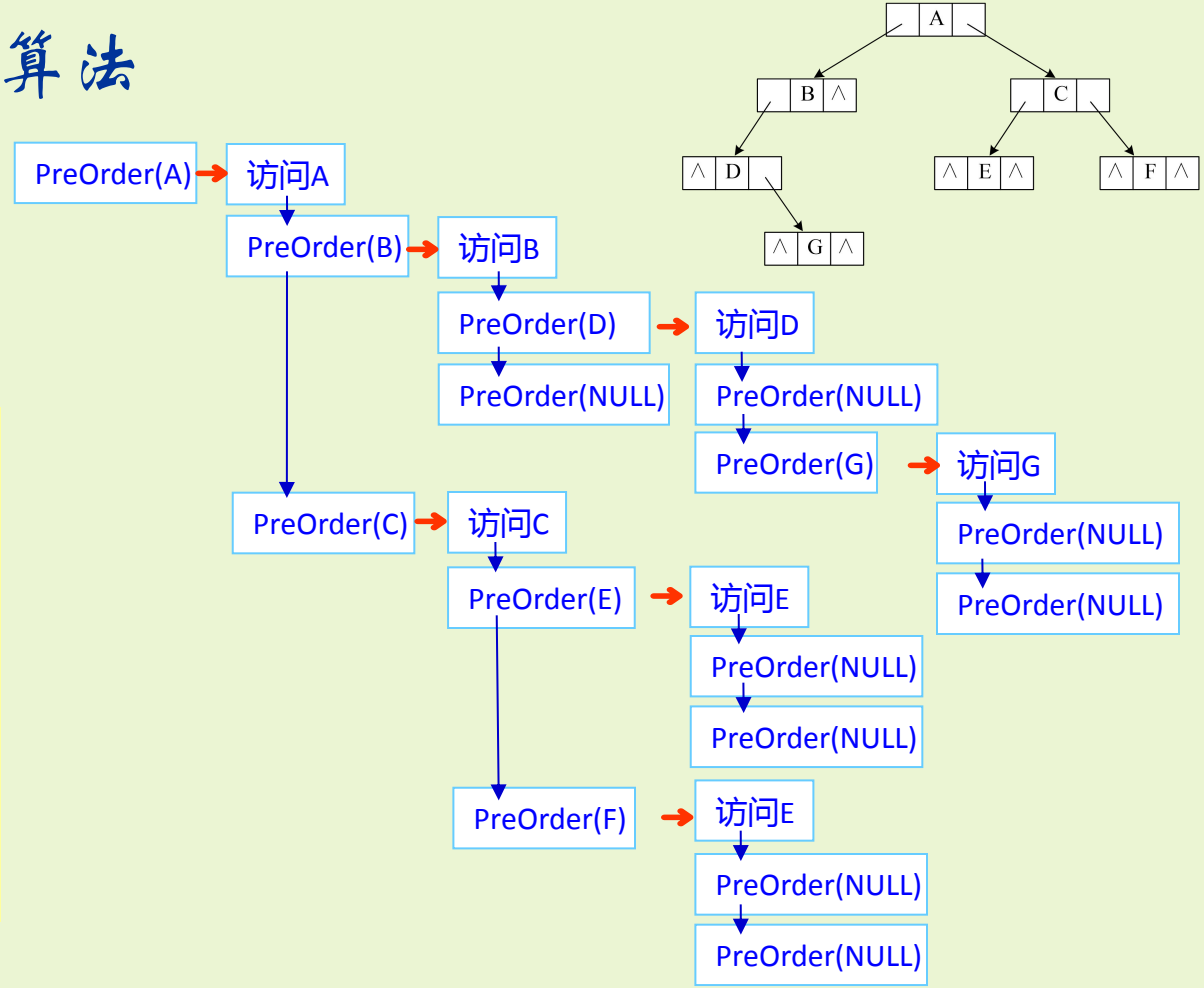
DGBAECF

GDBEFCA

先序遍历的递归算法

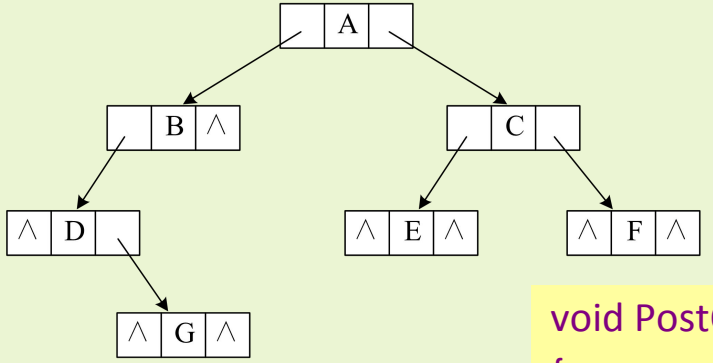
- 先序遍历二叉树的过程
  - 访问根节点
  - 先序遍历左子树
  - 先序遍历右子树

```
void PreOrder(BTNode *b)
{
   if (b!=NULL)
   {
      printf("%c ",b->data);
      PreOrder(b->lchild);
      PreOrder(b->rchild);
   }
}
```



中序和后序遍历的递归算法

- 中序遍历二叉树的过程
  - 中序遍历左子树
  - 访问根节点
  - 中序遍历右子树




```
void InOrder(BTNode *b)
{
   if (b!=NULL)
   {
      InOrder(b->lchild);
      printf("%c ",b->data);
      InOrder(b->rchild);
   }
}
```

- 后序遍历二叉树的过程
  - 后序遍历左子树
  - 后序遍历右子树
  - 访问根节点

```
void PostOrder(BTNode *b)
{
   if (b!=NULL)
   {
      PostOrder(b->lchild);
      PostOrder(b->rchild);
      printf("%c ",b->data);
   }
}
```

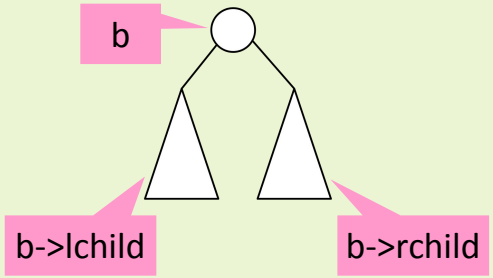
例：计算二叉树节点个数

- 问题
  - 假设二叉树采用二叉链存储结构存储，设计算法，计算一棵给定二叉树的所有节点个数。
- 解：计算一棵二叉树b中所有节点个数的递归模型f(b)如下

$$f(b)=\begin{cases}0 & ,\ b=NULL\\ f(b\text{->}lchild)+f(b\text{->}rchild)+1 & \text{其他情况}\end{cases}$$

  - 算法

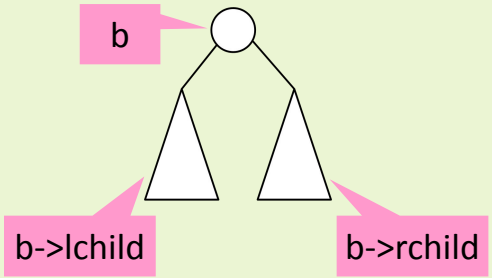
```
int Nodes(BTNode *b)
{
   if (b==NULL)
      return 0;
   else
      return Nodes(b->lchild)+Nodes(b->rchild)+1
   }
}
```



后序遍历思路

例：输出所有叶子节点

- 问题
  - 假设二叉树采用二叉链存储结构存储，试设计一个算法输出一棵给定二叉树的所有叶子节点。
- 解：输出一棵二叉树的所有叶子节点的递归模型f()如下
  - 不做任何事件　　若b=NULL
  - 输出*b节点的data域　　若*b为叶子节点
  - f(b->lchild);f(b->rchild)　　其他情况




```
void DispLeaf(BTNode *b)
{
   if (b!=NULL)
   {
      if (b->lchild==NULL && b->rchild==NULL)
         printf("%c ",b->data);
      else
      {
         DispLeaf(b->lchild);
         DispLeaf(b->rchild);
      }
   }
}
```

先序遍历思路

例：输出节点所在的层数

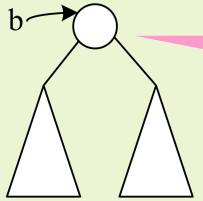
- 问题：二叉树（没有值相同的节点）采用二叉链存储结构，设计一个算法Level(b,x,h)，返回二叉链b中data值为x的节点的层数。

```
int Level(BTNode *b,ElemType x,int h)
{
   if (b==NULL)
      return 0;
   else if (b->data==x)
      return h;
   else
   {
      l=Level(b->lchild,x,h+1);
      if (l==0)
         return Level(b->rchild,x,h+1);
      else
         return l;
   }
}
```

h表示b所指节点的层数

先序遍历思路

类似查找算法



调用Level(b,x,1)
返回x节点的层数

```
int main()
{
   BTNode *b;
   int h;
   ElemType x;
   CreateBTNode(b,"A(B(D(,G)),C(E,F))");
   scanf("%c",&x);
   h=Level(b,x,1);
   ……
}
```

例：相似的二叉树

- 问题
  - 设计一个算法判断两棵二叉树是否相似
  - 所谓二叉树t1和t2相似
    - t1和t2都是空的二叉树，相似；
    - t1和t2之一为空，另一不为空，则不相似；
    - t1的左子树和t2的左子树是相似的，且t1的右子树与t2的右子树是相似的，则t1和t2相似。
- 解：判断两棵二叉树是否相似的递归模型f()如下：

  f(t1,t2)=true　　若t1=t2=NULL

  f(t1,t2)=false　　若t1、t2之一为NULL，另一不为NULL

  f(t1,t2)=f(t1->lchild,t2->lchild) &&f(t1->rchild,t2->rchild)　其他情况
- 对应的算法

```
int Like(BTNode *b1,BTNode *b2)
{
   int like1,like2;
   if (b1==NULL && b2==NULL)
      return 1;
   else if (b1==NULL || b2==NULL)
      return 0;
   else
   {
      like1=Like(b1->lchild,b2->lchild);
      like2=Like(b1->rchild,b2->rchild);
      return (like1 & like2);
   }
}
```