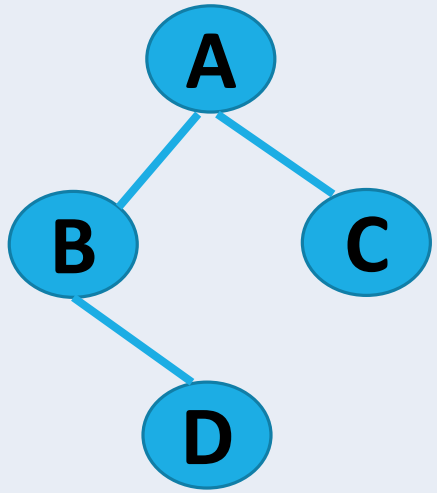
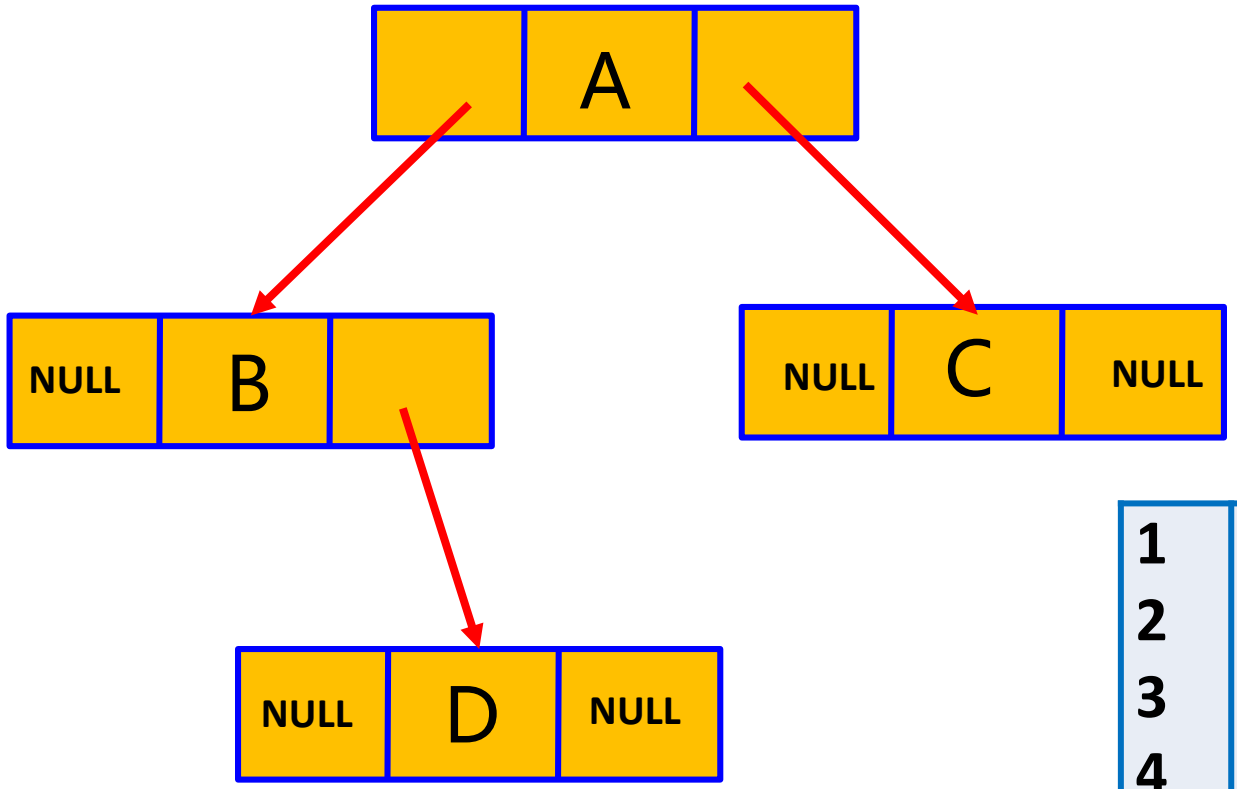
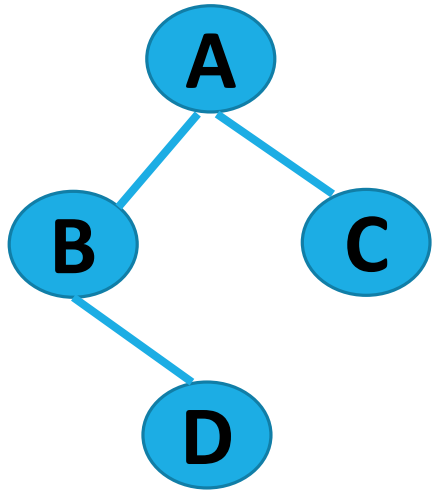
二叉树的递归建立算法

算法4-10

```
1   BinTree CreateBinTree_Recursion()
2   {
3       char ch;  BinTree bt;
4       scanf_s("%c",&ch);
5       if(ch=='@')    bt=NULL;
6       else{
7               bt = (BinTreeNode *)malloc(sizeof(BinTreeNode));
8               bt->data = ch;
9               bt->leftchild = CreateBinTree_Recursion ();
10              //递归构造左子树
11             bt->rightchild = CreateBinTree_Recursion ();
12             //递归构造右子树
13          }
14      return bt;
15  }
```







输入序列：AB@D@@C@@

```
1   BinTree CreateBinTree_Recursion()
2   {
3      char ch;  BinTree bt;
4      scanf_s("%c",&ch);
5      if(ch=='@')    bt=NULL;
6      else{
7          bt = (BinTreeNode *)malloc(sizeof(BinTreeNode));
8                  bt->data = ch;
9                  bt->leftchild = CreateBinTree_Recursion ();
10                 bt->rightchild = CreateBinTree_Recursion ();
11       }
12     return bt;
13  }
```

4.9 二叉树的建立和遍历—递归算法　　算法4-6

```
1  void PreOrder_Recursion(BinTree bt)     //递归先序遍历
2  {
3          if (bt == NULL) return;                    //如果是空则返回
4          printf("%c", bt->data);                    //访问数据域
5          PreOrder_Recursion(bt->leftchild);  //递归遍历左子树
6          PreOrder_Recursion(bt->rightchild);//递归遍历右子树
7  }
```

4.9 二叉树的建立和遍历—递归算法

算法4-7

```
1  void InOrder_Recursion(BinTree bt) //递归中序遍历
2  {
3          if(bt==NULL) return;                    //如果是空则返回
4          InOrder_Recursion(bt->leftchild);    //递归遍历左子树
5          printf("%c",bt->data);                  //访问数据域
6          InOrder_Recursion(bt->rightchild);    //递归遍历右子树
7  }
```

4.9 二叉树的建立和遍历—递归算法

算法4-8

```
1  void PostOrder_Recursion(BinTree bt)//递归后序遍历
2  {
3          if(bt==NULL) return;                          //如果是空则返回
4          PostOrder_Recursion(bt->leftchild); //递归遍历左子树
5          PostOrder_Recursion(bt->rightchild);//递归遍历右子树
6          printf("%c",bt->data);                        //访问数据域
7  }
```