阅读：25,316　　作者：解学武

二叉树的链式存储结构及（C语言）实现

上一节　　下一节

上一节讲了二叉树的顺序存储，通过学习你会发现，其实二叉树并不适合用数组存储，因为并不是每个二叉树都是完全二叉树，普通二叉树使用顺序表存储或多或多会存在空间浪费的现象。

本节我们学习二叉树的链式存储结构。

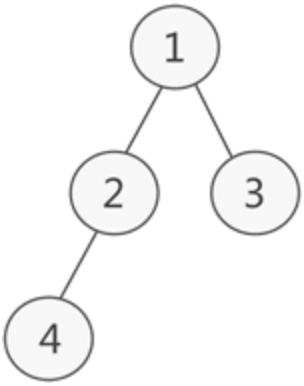

图 1 普通二叉树示意图

如图 1 所示，此为一棵普通的二叉树，若将其采用链式存储，则只需从树的根节点开始，将各个节点及其左右孩子使用链表存储即可。因此，图 1 对应的链式存储结构如图 2 所示：

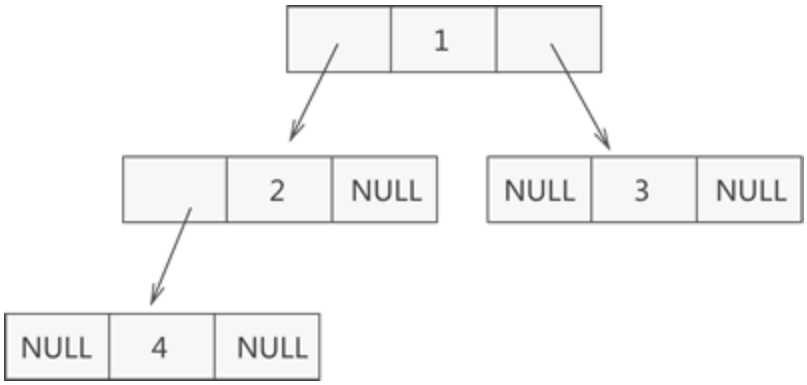

图 2 二叉树链式存储结构示意图

由图 2 可知，采用链式存储二叉树时，其节点结构由 3 部分构成（如图 3 所示）：

- 指向左孩子节点的指针（Lchild）；
- 节点存储的数据（data）；
- 指向右孩子节点的指针（Rchild）；

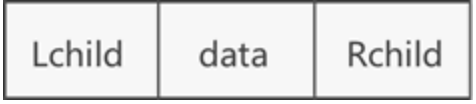

图 3 二叉树节点结构

表示该节点结构的 C 语言代码为：

```
typedef struct BiTNode{
    TElemType data;//数据域
    struct BiTNode *lchild,*rchild;//左右孩子指针
    struct BiTNode *parent;
}BiTNode,*BiTree;
```

图 2 中的链式存储结构对应的 C 语言代码为：

```
#include <stdio.h>
#include <stdlib.h>
#define TElemType int

typedef struct BiTNode{
    TElemType data;//数据域
    struct BiTNode *lchild,*rchild;//左右孩子指针
}BiTNode,*BiTree;

void CreateBiTree(BiTree *T){
    *T=(BiTNode*)malloc(sizeof(BiTNode));
    (*T)->data=1;
    (*T)->lchild=(BiTNode*)malloc(sizeof(BiTNode));
    (*T)->lchild->data=2;
    (*T)->rchild=(BiTNode*)malloc(sizeof(BiTNode));
    (*T)->rchild->data=3;
    (*T)->rchild->lchild=NULL;
    (*T)->rchild->rchild=NULL;
    (*T)->lchild->lchild=(BiTNode*)malloc(sizeof(BiTNode));
    (*T)->lchild->lchild->data=4;
    (*T)->lchild->rchild=NULL;
    (*T)->lchild->lchild->lchild=NULL;
    (*T)->lchild->lchild->rchild=NULL;
}
int main() {
    BiTree Tree;
    CreateBiTree(&Tree);
    printf("%d",Tree->lchild->lchild->data);
    return 0;
}
```

程序输出结果：

4

其实，二叉树的链式存储结构远不止图 2 所示的这一种。例如，在某些实际场景中，可能会做 "查找某节点的父节点" 的操作，这时可以在节点结构中再添加一个指针域，用于各个节点指向其父亲节点，如图 4 所示：

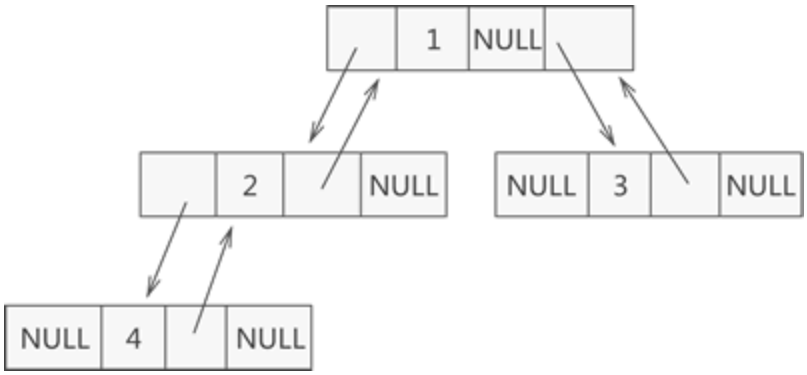

图 4 自定义二叉树的链式存储结构

这样的链表结构，通常称为三叉链表。

利用图 4 所示的三叉链表，我们可以很轻松地找到各节点的父节点。因此，在解决实际问题时，用合适的链表结构存储二叉树，可以起到事半功倍的效果。

上一节 下一节