用 C 语言实现二叉树的遍历及其应用

谷 立 东

(牡丹江教育学院，黑龙江 牡丹江 157005)

[摘 要] C 语言有较丰富的数据类型、运算符以及函数，能直接与内存打交道，使修改、编辑其他程序与文档变得简单。树型结构是一类重要的非线性数据结构，其中以树和二叉树最为常用。二叉树的遍历算法是树形结构中其他运算的基础，在二叉树遍历的各种算法中包括了一些精致的、并且在其他应用范围内也有用的技巧，所以本文主要讨论用 C 语言去实现二叉树遍历的几种不同算法。

[关键词] 数据结构；树；二叉树；二叉树的遍历；C 语言

[中图分类号]TP312 [文献标识码]A [文章编号]1009－2323(2006)06－0142－02

《数据结构》在计算机科学中是一门综合性的专业基础课。在《数据结构》中，树型结构是结点之间有分支的、层次的关系的结构。树结构在客观世界中广泛存在，如人类的族谱、动植物的分类、图书情报资料的编目等，都可以按照层次表示成树的形式。在计算机程序设计方面，树也是很重要的，如在编译程序中，可以用树来表示源程序的语法结构。又如在数据库系统中，树形结构也是信息的重要组织形式之一。树型结构是一类重要的非线性数据结构，其中以树和二叉树最为常用。

1. **树** (Tree)

树是 n(n>=0) 个结点的有限集。在一棵非空树中：(1)有且仅有一个特定的称为根(Root)的结点；(2)当 n>1 时，其余结点可分为 m(m>0)个互不相交的有限集 T_1，T_2，……，T_m，其中每一个集合本身又是一棵树，并且称为子树(Subtree)。结点拥有的子树数称为结点的度(degree)。树的度是树内各结点的度的最大值。

2. **二叉树** (Binary tree)

二叉树是另一种树型结构，它的特点是每个结点至多只有二棵子树(即二叉树中不存在度大于 2 的结点)，二叉树的子树有左右之分，其次序不能任意颠倒。二叉树第 i 层上的结点数目最多为 2^{i-1}(i>=1)；深度为 k 的二叉树至多有 $2k-1$ 个结点(k≥1)；对任何一棵二叉树 T，如果其终端结点数 n_0，度为 2 的结点数为 n_2，则 $n_0=n_2+1$。

对于使用 C 语言去实现二叉树，首先需要抽象其二叉树的数据类型，也就是需要构造一个基本二叉树的基础操作的类和一个二叉树结点数据类型。

第一步分析结点数据类型。二叉树的结点包括有本身的数据和左右子树的位置。

第二步是去设计二叉树的基本操作，这里需要通过分析该二叉树基本功能，然后去构造一个类来完成，这步需要使用到上一步中的自定义类型。

二叉树的基本功能包括：

InitBiTree(&T)：构造空二叉树 T.

DestroyBiTree(&T)：销毁二叉树 T.

CreateBiTree(&T, description)根据 description 构造二叉树 T.

ClearBiTree(&T)：清空二叉树.

IsEmptyBiTree(T)：若 T 为空二叉树，则返回 TRUE；否则返回 FALSE.

GetBiTreeDepth(T)：返回 T 的深度.

GetBiTreeRoot(T, &root)：返回二叉树 T 的 root 根结点.

GetBiTNodeValue(e, &value)：返回结点 e 的 data 值字段.

AssignBitNode(&e, value)：把 value 的值赋给结点 e 的 data 字段.

GetBiTNodeParent(T, e, &parent)：若 e 是 T 的非根结点，则返回它的双亲，否则返回 NULL.

GetBiTNodeLeftChild(e, &lChild)：若 e 有左孩子，则返回它的左孩子，否则返回 NULL.

GetBiTNodeRightChild(e, &rChild)：若 e 有右孩子，则返回它的右孩子，否则返回 NULL.

GetBiTNodeLeftSibling(T, e, &lSibling)：返回 e 的左兄弟，若 e 无左兄弟，返回 NULL.

GetBiTNodeRightSibling(T, e, &rSibling)：返回 e 的右兄弟，若 e 无右兄弟，返回 NULL.

InsertBiTNode(T, p, LR, c)：根据枚举 LR 的内容，插入结点 e 到 p 所指向的结点下.

DeleteBiTNode(T, p, LR)：根据枚举 LR 的内容，删除结点 e 的左/右结点.

PreOrderBiTreeTraverse(&T, visit())：先序遍历 T，对每个结点调用 visit 函数.

InOrderBiTreeTraverse(&T, visit())：中序遍历 T，对每个结点调用 visit 函数.

PostOrderBiTreeTraverse(&T, visit())：后序遍历 T，对每个结点调用 visit 函数.

LevelOrderBiTreeTraverse(&T, visit())：层序遍历 T，

[收稿日期]2006－09－28

[作者简介]谷立东(1967－)，女，黑龙江牡丹江人，牡丹江教育学院讲师。

对每个结点调用 visit 函数.

DisplayBiTree(BiTree T):显示二叉树的内容.

3.**二叉树的遍历**(Traversing binary tree)

在二叉树的一些应用中,常常要求在树中查找具有某些特征的结点,或者对树中全部结点逐一进行某种处理。遍历二叉树是二叉树的一种重要的运算。

所谓遍历(Traversal)是指按某条搜索路径巡访树中每个结点,使每个结点均被访问一次,而且仅被访问一次。“访问”的含义很广,可以是对结点作多种处理,如输出结点的信息等。

设访问根结点记作 V,遍历根的左子树记作 L ,遍历根的右子树记作 R,则可能的遍历次序有:

前序:VLR ;中序:LVR ;后序:LRV。

3.1 中序遍历(Inorder Traversal)二叉树算法的定义

若二叉树为空,则空操作;否则,中序遍历左子树 (L);访问根结点 (V);中序遍历右子树 (R)。中序遍历二叉树的递归算法:

```
void InOrder ( BinTreeNode *T )
{
if ( T ! = NULL )
  {
    InOrder ( T->leftChild );
    Visit( T->data);
    InOrder ( T->rightChild );
  }
}
```

3.2 前序遍历(Preorder Traversal)二叉树算法的定义

若二叉树为空,则空操作;否则,访问根结点 (V);前序遍历左子树 (L);前序遍历右子树 (R)。前序遍历二叉树的递归算法:

```
void PreOrder ( BinTreeNode *T )
{
if ( T ! = NULL )
  {
    Visit( T->data);
    PreOrder ( T->leftChild );
    PreOrder ( T->rightChild );
  }
  }
```

3.3 后序遍历(Postorder Traversal)二叉树算法的定义

若二叉树为空,则空操作;否则,后序遍历左子树 (L);后序遍历右子树 (R);访问根结点 (V)。后序遍历二叉树的递归算法:

```
void PostOrder ( BinTreeNode * T )
{
if ( T ! = NULL )
{
    PostOrder ( T->leftChild );
    PostOrder ( T->rightChild );
    Visit( T->data);
  }
}
```

4.**二叉树遍历算法的应用举例**

我们可以在三种遍历算法的基础上改造完成的其它二叉树算法,如:求叶子个数,求二叉树结点总数,求度为 1 或度为 2 的结点总数,复制二叉树,建立二叉树,交换左右子树,查找值为 n 的某个指定结点,删除值为 n 的某个指定结点等等。

4.1 求二叉树中叶子结点的个数

```
int Leaf_Count(Bitree T)
{ //求二叉树中叶子结点的数目
    if(! T) return 0; //空树没有叶子
    elseif (! T->lchild&&! T->rchild) return 1; //叶子结点
    else return Leaf_ Count(T-lchild)+Leaf_ Count(T-rchild); //左子树的叶子数加上右子树的叶子数
}
```

4.2 求二叉树高度(递归算法)

```
int Height ( BinTreeNode * T )
{
    if ( T == NULL ) return 0;
    else
    {
    int m = Height ( T->leftChild );
    int n = Height ( T->rightChild ) );
    return (m > n) ? m+1 : n+1;
    }
}
```

4.3 复制二叉树(递归算法)

```
BiTNode* Copy( BinTreeNode * T )
{
    if ( T == NULL ) return NULL;
    if(! (Temp=(BiTNode *)malloc(sizeof(BiTNode)))) exit(OVERFLOW);
    Temp->data=T->data;
    Temp-> leftChild = Copy( T->leftChild );
    Temp-> rightChild = Copy(T->rightChild ); return Temp;
}
```

4.4 判断二叉树等价(递归算法)

```
int Equal( BinTreeNode *a, BinTreeNode *b)
{
    if ( a == NULL && b == NULL ) return 1;
    if ( a! == NULL && b! == NULL
    && a->data==b->data
    && equal( a-> leftChild, b->leftChild)
    && equal( a->rightChild, b->rightChild) )
    return 1;
    return 0; //如果 a 和 b 的子树不等同,则函数返回 0
}
```

二叉树的遍历也为后面学习的数据结构包括线索二叉树以及线索化后的查找算法,最优二叉树(哈夫曼树)的概念、构成和应用,树与森林的遍历算法及其与二叉树遍历算法的联系,树与森林和二叉树的转换做好铺垫。

[参 考 文 献]

[1]严蔚敏,吴伟民.数据结构[A].清华大学出版社,1997.

[2]谭浩强.C 语言程序设计[M].清华大学出版社,2002.

[责任编辑:丛爱玲]