第 21 卷第 3 期　　佛山科学技术学院学报(自然科学版)　　Vol. 21 No. 3
2003 年 9 月　　Journal of Foshan University (Natural Science Edition)　　Sep. 2003

文章编号:1008-0171(2003)03-0035-04

二维及多维链表及其算法实现

李晓东

(佛山科学技术学院 计算机系,广东 佛山 528000)

摘要:通过对线性链表和数组的比较研究,总结出二(多)维链表的结构和算法,指出采用二(多)维链表可以克服数组内存分配的局限性和单链表数据访问的局限性,并通过实例说明二维链表的用法。

关键词:二维链表;多维链表;数据结构;算法

中图分类号:TP391.41　　文献标识码:

数组是任何一种程序设计语言最常用的数据结构之一,数组的使用非常方便、简单。但是在计算机中数组要求内存分配必须是连续的,数组很大时内存分配常得不到满足。并且,大多数编译程序所支持的数组大小是有限的,这也给数组的使用带来了一定的局限。

链表也是一种常用的、重要的数据结构,其内存分配不要求连续。主要特点是结构比较简单、实用,也比较方便,可以用它来实现比较复杂的数据关系。但是简单的单链表也有其固有的缺点,例如,当链表结点数较多,搜索结点数据的速度比较慢,其访问的时间复杂度为 $O(n)$[1]。并且,当用链表表示二维数组时,访问数据缺乏直观性,给数据访问带来了一些麻烦。

如果能够将数组和链表结合起来,克服它们各自的缺点,发扬各自的长处,这肯定是一项比较好的技术。二(多)维链表就能较好地做到这一点。

1　二维链表存储结构及其算法

1.1　一维链表的存储结构

(1)结点结构。包括 2 个域:数据域与指针域,如图 1 所示。用 C++语言定义如下

```
typedef loat Datatype //数据域类型(也可为其他合适类型)
typedef struct Lnode {
    Datatype  *data;//数据指针
    struct Lnode  *pnext;//结点指针,指向下一结点
    }Lnode, *plinklist;
```

(2)链表结构[1,2]。一维链表结构如图 2 所示。用 C++语言定义如下

```
typedef struct Llist {
int nodenum; //结点数
Lnode  *HeadNode ;//头结点指针
} Llist;
```

很显然,一维链表就是普通的单链表。可以为其定义一组相关的操作使得它象操作一维数组那样方便[3]。

收稿日期:2002-09-17

作者简介:李晓东(1968-),男,江西东乡人,佛山科学技术学院助教,硕士,主要从事图像图形处理技术等方面的研究。

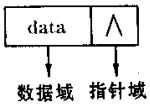

图 1 一维链表结点

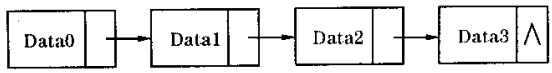

图 2 一维链表结构

1.2 二维链表的存储结构

(1)结点结构。包括 3 个域:数据域,行指针域 px 及列指针域 py,见 3 所示。用 C++语言定义如下

```
typedef float Datatype //数据域类型(也可为其他合适类型)
    typedef struct Two_Dim_Lnode {
      Datatype *data;//数据指针
      struct Two_Dim_Lnode *prow, *pcol;//行、列指针
      } Two_Dim_Lnode, *pTwo_Dim_Lnode;
```

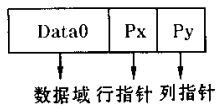

图 3 二维链表结点

(2)链表结构。二维链表结构如图 4 所示,用 C++语言描述如下

```
typedef struct Two_Dim_Link {
  int rownum,colnum;//链表行、列数
  Two_Dim_Lnode *HeadNode ;//头结点指针
  } Two_Dim_Link;
```

显然,对于二维链表来说,它的每一个结点类型是相同的。要得到 1 个结点的数据可以通过多种不同的搜索路径来实现。如可以先进行行搜索,再进行列搜索,或者相反。(如上图搜索路径 1 和搜索路径 2)。可以看出,对于二维链表,它的时间复杂度为 $O(n^{\frac{1}{2}})$。另外,二维链表非常类似于十字链表,但二者之间也存在着不同,十字链表存在两种不同类型的结点,而二维链表只存在 1 种类型的结点,因而它们的实现算法是不同的。

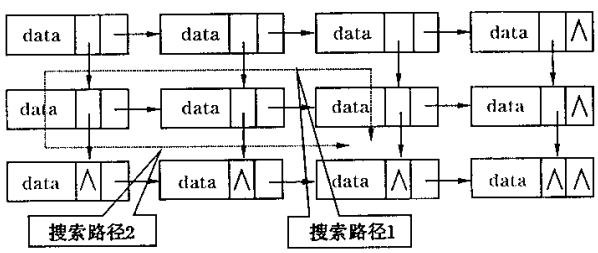

图 4 二维链表结构

1.3 二维链表的有关操作

由于一维链表可以看成是二维链表的特例,在此主要讨论二维链表的算法。可以给二维链表定义如下算法。(1)创建二维链表 bool CreateTwo_Dim_Link(int rownum, int colnum, Two_Dim_Link *Dlink);(2)增加一行(列)结点 bool AddALine(Two_Dim_Link *Dlink);(3)删除一行(列)结点 void DeleteALine(int RowNo,Two_Dim_Link * Dlink);(4)取得第 row 行第 col 列结点 Two_Dim_Lnode GetNode(int row,int col,Two_Dim_Link * Dlink);(5)取得 row 行 col 列结点的数据 Two_Dim_Lnode GetNode(int row, int col,Two_Dim_Link *Dlink)(6)设置 row 行 col 列结点的数据 void SetNodeData(int row,int col,Two_Dim_Link *Dlink,Datatype data);(7)插入一行(列)结点 bool InsertALine(int lineNo,Two_Dim_Link *Dlink);(8)销毁链表 void DeleteTwo_Dim_Link(Two_Dim_Link *Dlink)。

2 三(多)维链表结构

2.1 三维链表的存储结构

(1)结点结构。包括 3 个域:数据域,指针域 px,指针域 py 及指针域 pz,如图 5 所示。用 C++语言定义如下

```
typedef struct Three_Dim_Lnode {
```

```
    Datatype *data;//数据指针
    struct Three_Dim_Lnode *px, *py, *pz;//各域指针
    } Three_Dim_Lnode, *pThree_Dim_Lnode ;
```

(2)**链表结构。三维链表结构用** C++**语言描述如下**

```
    typedef struct Three_Dim_Link {
      int xNum,yNum,zNum;//链表各维深度
      Three_Dim_Lnode *HeadNode ;//头结点指针
      } Three_Dim_Link;
```

2.2　多维链表的存储结构

(1)**结点结构。包括** 3 **个域：数据域，指针域** x，**指针域** y，**指针域** z，**如图** 6 **所示。用** C++**语言描述定义如下**

```
    typedef struct Mul_Dim_Lnode {
      Datatype *data;//数据指针
      struct Mul_Dim_Lnode *px1, *px2, …, *pxn;//各域指针
    } Mul_Dim_LNode, *pMul_Dim_LNode ;
```

(2)**链表结构。多维链表结构用** C++**语言描述如下**

```
    typedef struct Three_Dim_Link {
      int x1Num,x2Num,…,xnNum;//链表各维深度
      Mul_Dim_Lnode *HeadNode ;//头结点指针
    } Mul_Dim_Link;
```

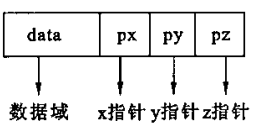

图 5　**三维链表结点**

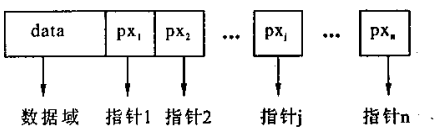

图 6　**三维链表结构**

3　应用举例

以下是一个应用二维链表存储并输出 9×9 **乘法表实现代码(为了节省篇幅，以下没有给出二维链表操作的实现代码，所有代码在** VC++6.0 **调试通过)。**

```
#include "iostream.h"
#include "iomanip.h"
void main()
{   Two_Dim_Link *Dlink1;
    Two_Dim_Lnode pT_D_Lnode;
    Dlink1=(Two_Dim_Link *)malloc(sizeof(Two_Dim_Link));//分配链表内存
    if (Dlink1==NULL) {
      cout<<"内存不足";
      exit;
    }
    if (CreateTwo_Dim_Link(9, 9, Dlink1)>0) //创建二维链表结点
    {   for (int i=0;i<9;i++)
          for(int j=0;j<9;j++)
            SetNodeData(i, j, Dlink1,(i+1)*(j+1));//设置二维链表数据域值
```

```
        for (int m=0;m<9;m++){
         for(int n=0;n<9;n++){
            pT_D_Lnode=GetNode(m, n, Dlink1);//获得二维链表数据域值
            cout<<pT_D_Lnode.data<<setw(5);//输出数据
             }
         cout<<endl;
        }
      }
      else cout<<"链表创建失败"<<endl;
  }
```

4 结论

从以上叙述可以知道,二维及多维链表解决了数组和单链表使用时存在的问题,因此对于一些程序员特别是需要从事科学计算的科研人员来说是具有非常重要的意义。一方面,二(多)维链表可以充分利用链表的优点,克服二(多)维数组的缺点,通过定义一些基本操作,使得二维链表的使用也比较简单。另一方面,使用二(多)维链表可以提高数据的访问速度,也方便了数据的访问。因此,二(多)维链表也可以用于数据库数据的检索,提高数据库的检索效率。

参考文献:

[1] 严蔚敏,吴 伟. 数据结构(C 语言版)[M]. 北京:清华大学出版社,1997. 30-32.

[2] ROBERT Kruse,TONDO C L,BRUCE P Leung. 数据结构与程序设计(C 语言描述)英文影印版[M]. 北京:清华大学出版社,1998. 152-160.

[3] WILLIANT Ford,WILLIAM Topp. 数据结构 C++语言描述[M]. 北京:清华大学出版社,1996. 304-312.

【责任编辑:任小平】

Data structures and algorithm implementation of two-dimension and multi-dimension linked lists

LI Xiao-dong

(Computer Science Department, Foshan University, Foshan 528000, China)

Abstract: By the comparison of general linked lists and arrays, the data structures and algorithm of two-dimension linked lists and multi-dimension linked lists can be gained. It would get over the shortcoming of arrays in memory allocation and the date getting of general linked lists. An example is given to show the usage of two-dimension linked lists.

Key words: two-dimension linked lists; multi-dimension linked lists; data structures; algorithm