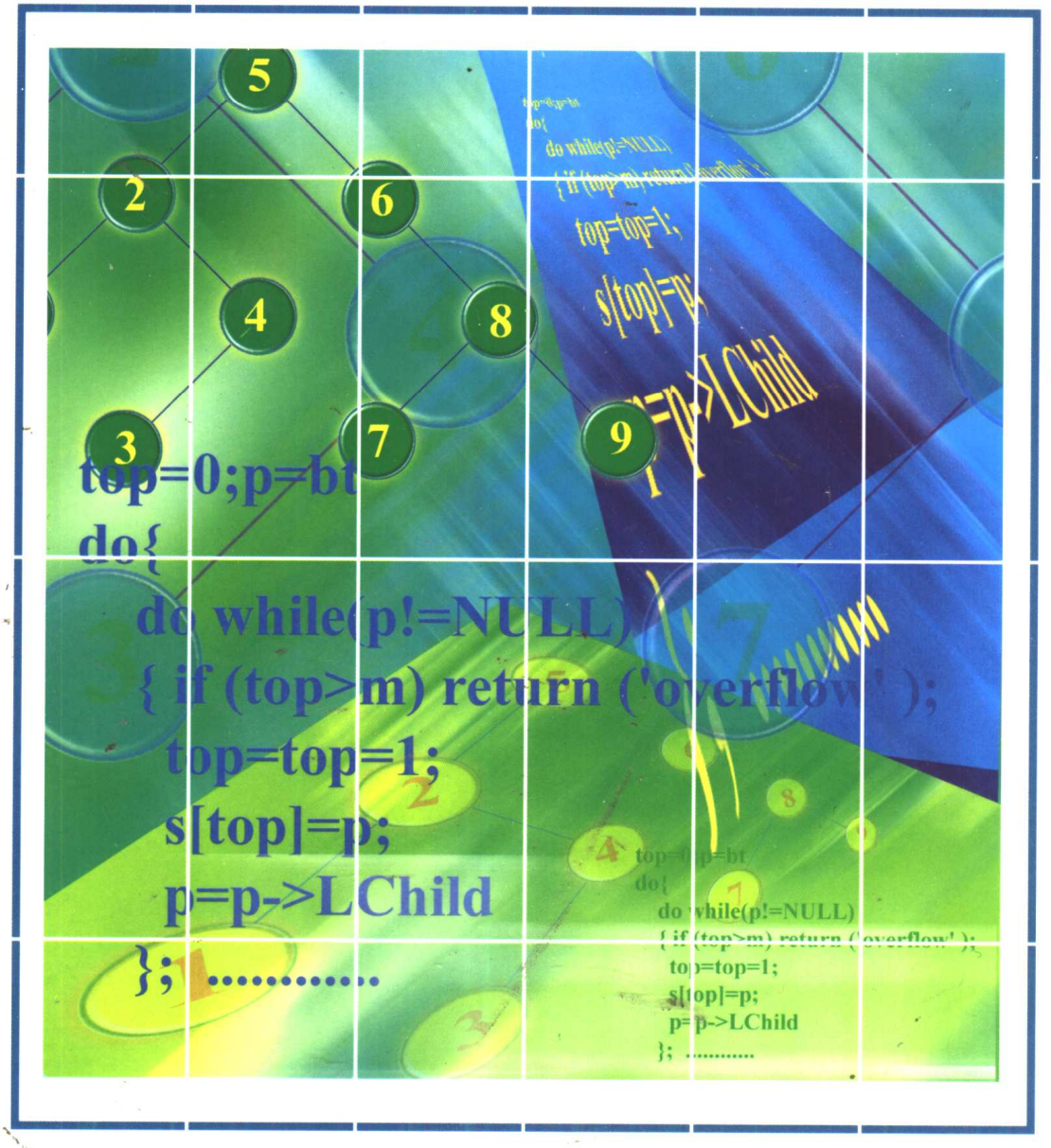
新世纪计算机类本科系列教材




# 数据结构(C语言)实践教程

胡元义　邓亚玲　罗作民　胡明星　编著

西安电子科技大学出版社
http://www.xduph.com

新世纪计算机类本科系列教材

# 数据结构(C语言)实践教程

胡元义　邓亚玲
罗作民　胡明星　编著

西安电子科技大学出版社
2002

## 内 容 简 介

本书是作者积多年讲授数据结构课程及指导学生上机实践的经验编写而成的。作者力求通过实践的角度，帮助学生深入学习、掌握并灵活应用数据结构的知识。

全书共分为两篇。第一篇为数据结构实践篇，共由七章组成，其内容函盖了数据结构课程中的全部实验；第二篇为数据结构的应用与提高，共有两章，给出了数据结构的典型应用及相应的研究，可帮助读者开拓学习和应用的视野。

本书可以配合目前各类数据结构(C 语言)教材使用，起到衔接教学与实践的作用。本书还可作为计算机应用人员的参考书。

**图书在版编目(CIP)数据**

**数据结构(C 语言)实践教程** / 胡元义等编著.

—西安：西安电子科技大学出版社，2002.12

新世纪计算机类本科系列教材

ISBN 7-5606-1181-8

Ⅰ. 数… Ⅱ. 胡… Ⅲ. ① 数据结构—高等学校—教材 ② C 语言—程序设计—高等学校—教材

Ⅳ. TP311.12

**中国版本图书馆 CIP 数据核字(2002)第 079197 号**

策　　划　陈宇光

责任编辑　王素娟

出版发行　西安电子科技大学出版社(西安市太白南路 2 号)

电　　话　(029)8227828　　邮　　编　710071

http://www.xduph.com　　E-mail: xdupfxb@pub.xaonline.com

经　　销　新华书店

印　　刷　陕西光大印务有限责任公司

版　　次　2002 年 12 月第 1 版　2002 年 12 月第 1 次印刷

开　　本　787 毫米×1092 毫米　1/16　印张 14

字　　数　327 千字

印　　数　1～4 000 册

定　　价　15.00 元

ISBN 7 - 5606 - 1181 - 8 / TP・0613

**XDUP 1452001-1**

# 前　言

数据结构是计算机专业的主干课程之一，其目的是让读者学习、分析和研究数据对象的特性及数据的组织方法，以便选择合适的数据逻辑结构和存储结构，设计相应的运算操作，把现实世界中的问题转化为计算机内部的表示与处理的方法。在计算机科学领域，尤其是在系统软件和应用软件的设计和应用中要用到各种数据结构，因此，掌握数据结构对提高软件设计和程序编制水平有很大的帮助。

“数据结构”课对理论与实践的要求都相当高，并且内容多难度大。虽然多数数据结构教材都强调了实践的重要性，但比较缺乏供实践练习的材料，很多教材对算法的描述也只是扼要的和概述性的，很多算法都采用类 C 或类 PASCAL 语言描述，无法直接上机实现。针对这种情况，我们编写了这本《数据结构(C 语言)实践教程》。本书可作为“数据结构”课程的辅助教材，供计算机专业的学生在“数据结构”课程实习时使用，以帮助学生在尽可能短的时间内对数据结构知识的实践与应用有一个比较全面、深入和系统的认识，达到理论与实践相结合的目的。

本书分为两篇。

第一篇为数据结构的实践，共七章。该篇从实践角度论述了数据结构的有关知识，并给出了全部实验。每一个实验均给出了本次实验的目的、实验内容，对实验中的要点和难点进行了说明，并给出了相应的参考程序以供学习使用；此外，每个实验后还附有思考题，它使读者能够在此实验的基础上做进一步的思考与提高，从而达到举一反三、触类旁通的目的。第一章：线性表，重点介绍了两种存储结构——顺序表和单链表的操作，特别是单链表，它是贯穿全书的其它逻辑结构——树、图的基础；第二章：栈和队列，对常用的栈和队列进行了介绍；第三章：串与数组，对串的运算特别是串的模式匹配以及数组中的稀疏矩阵做了重点介绍；第四章：树和二叉树，它是本书的重点内容之一，重点介绍了递归和非递归遍历二叉树算法，此外，还介绍了如何由遍历序列恢复二叉树的实现方法；第五章：图，图也是本书的重点，包括图的搜索算法、最小生成树构造方法和最短路径；第六章，排序，排序也是本书的重点，本章介绍了各种排序方法的实现；第七章：查找，介绍了各种查找算法。

第二篇为数据结构的应用与提高，共两章。第八章：数据结构应用实例，为开拓学习视野给出了数据结构的具体应用，即(1) 线性表应用——仓库管理；(2) 栈的应用——表达式转换；(3) 队列应用——一个简单事件的规划问题；(4) 二叉树应用——银行财务实时处理系统；(5) 图的应用——工程工期控制问题；(6) 查找应用——学生档案管理。第九章：数据结构典型问题研究，为进一步深入学习、

研究数据结构知识给出了(1) 最短路径输出问题研究；(2) 递归转换为非递归问题研究；(3) 人工智能应用研究。

为了阅读方便，在表述串、顶点、结点等时，语言叙述部分的外文字符的大小写，以及采用的下标符等与程序有所差异。

在本书的出版过程中，得到了西安电子科技大学出版社，尤其是陈宇光老师的热情帮助和大力支持，在此表示衷心的感谢。

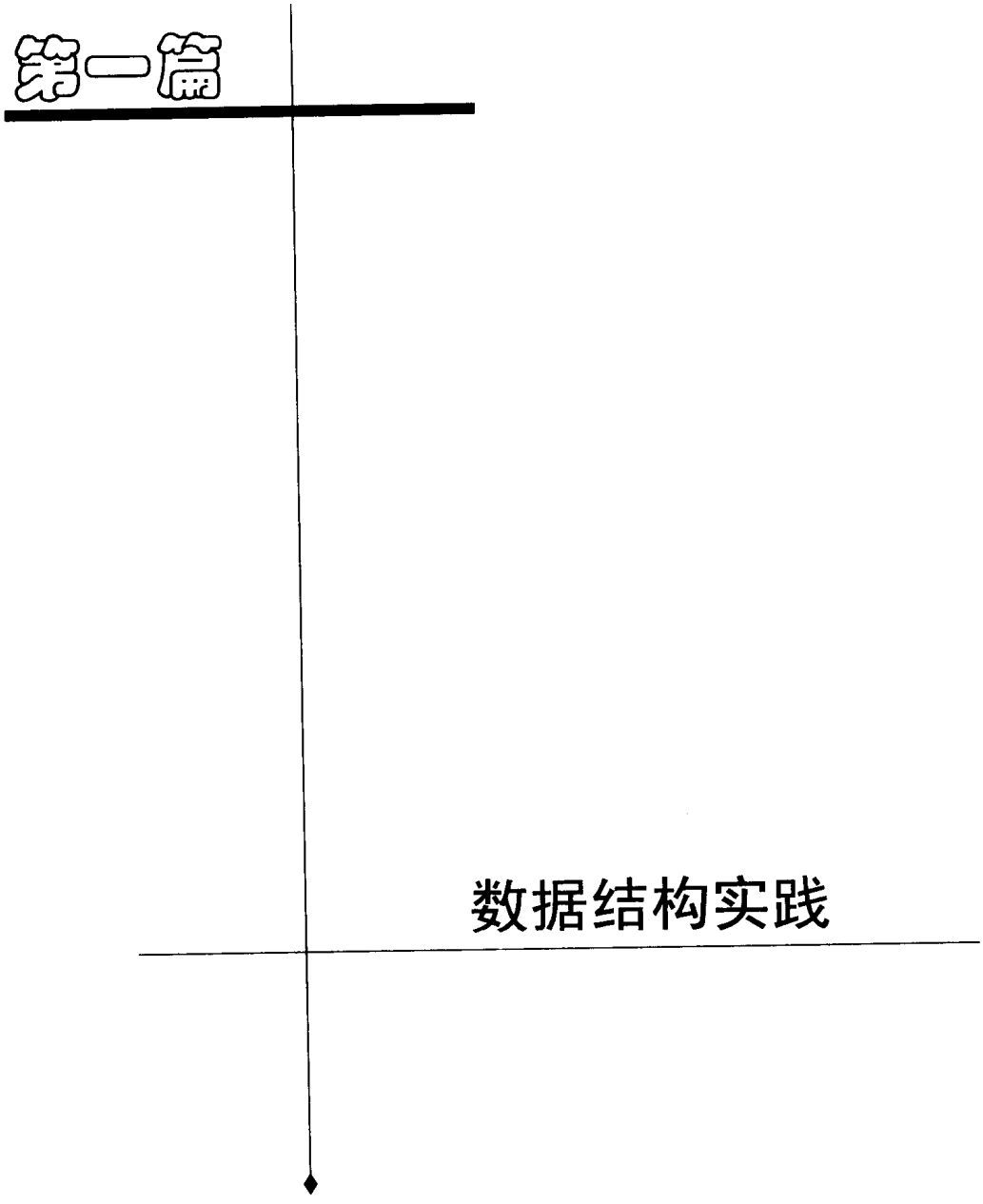
由于作者水平有限，书中难免存在错误和不妥之处，敬请广大读者批评指正。

作　者

2002 年 4 月 2 日

# 目　录

## 第一篇　数据结构实践


第一篇
数据结构实践

# 第一章 线性表

## 1.1 内容与要点

线性表是最基本、最常用的数据结构。简单地说，一个线性表是 n 个元素的有限序列，其特点是在数据元素的非空集合中：

(1) 存在惟一一个称为“第一个”的元素；

(2) 存在惟一一个称为“最后一个”的元素；

(3) 除第一个元素之外，集合中的每个元素均只有一个直接前驱；

(4) 除最后一个元素之外，集合中的每个元素均只有一个直接后继。

### 1.1.1 线性表的顺序存储

线性表顺序存储方式是用一组地址连续的存储单元依次存储线性表的数据元素。在这种顺序存储结构中，逻辑上相邻的两个元素在物理位置上也相邻，即线性表的逻辑关系无须额外增加存储空间来表达。

线性表第 i 个元素 $a_i$ 的存储位置如下：

$$Loc(a_i)= Loc(a_1)+( i-1)* L$$

其中，$Loc(a_1)$为线性表的起始位置(即第一个元素的位置)；L 为每个元素所占空间的大小。由此可知，线性表中的任一个元素都可以随机存取。

### 1.1.2 线性表的链式存储

在线性表的链式存储方式中，可用连续或不连续的存储单元来存储线性表中的元素，但元素之间的逻辑关系需要用“指针”来指示，这种指针是要额外占用存储空间的。链式存储方式失去了随机存取数据元素的功能，但换来了存储空间操作的方便性，即进行插入或删除时无须移动大量元素。

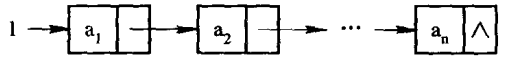
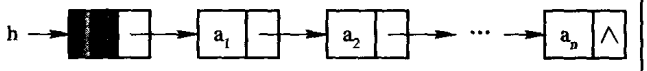
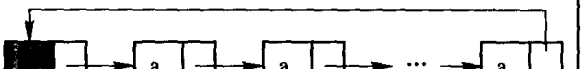
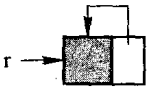
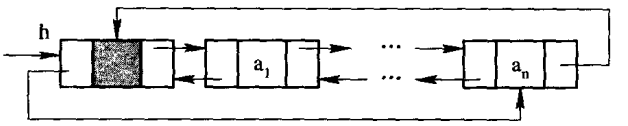
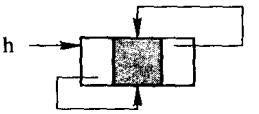
几种常用的链式存储结构见表 1.1。

## 1.2 线性表的实践

### 1.2.1 顺序表实践

用一组地址连续的存储单元依次存储线性表的各个数据元素，称作线性表的顺序存储结构。顺序表的基本操作包括顺序表的建立、在顺序表中插入元素、删除顺序表中的元素及顺序表的复制等。

表 1.1　几种常用的链式存储结构

| 类别 | | 结点结构 | 线性表存储结构图示 |
|---|---|---|---|
| 单链表 | 无头结点 | data \| next<br>next：指示后继结点的地址(或位置) |  <br>l (指示第一个数据结点)　空表：l的值为空(NULL) |
| | 带头结点 | |  <br>h (指示附加头结点)<br>空表：<br>  |
| | 循环链表 | |  <br>空表：<br>  <br>(指示尾结点) |
| | 双向循环链表 | link \| data \| rlink<br>link：指向前驱结点的地址(或位置)<br>rlink：指向后继结点的地址(或位置) |  <br>空表：<br>  |

## 实验一　顺序表的建立

### 1. 实验目的

了解顺序表的结构特点及有关概念，掌握顺序表建立的基本操作算法。

### 2. 实验内容

建立 4 个元素的顺序表 list[]={2，3，4，5}，实现顺序表建立的基本操作。

### 3. 实验要点及说明

顺序表又称为线性表的顺序存储结构，它用一组地址连续的存储单元依次存放线性表的各个元素。

可定义顺序表如下：

```
#define maxnum                  /* 顺序表最大元素个数 11 */
elemtype list[maxnum];          /* 定义顺序表 list */
int num=-1;                     /* 定义当前数据元素下标，并置初值为-1 */
```

参考程序中，通过 while 循环给 list[]表中输入元素。参考程序所建立的顺序表示意如图

1-1 所示。

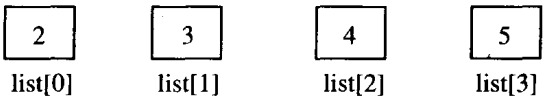



图 1-1　顺序表示意

**4. 参考程序**

```
#include <stdio.h>
#define max   10
main()
 {int i=0,x,*num,ch;
  int list[max];
  printf("Input list:");
  while((ch=getchar())!='\n')                    /* 输入顺序表元素，以换行符结束 */
   {
    list[i]=ch;
    i++;
   }
  *num=i-1;
   for(i=0;i<=*num;i++)                          /* 输出顺序表元素 */
       printf("list[%d]=%c",i,list[i]);
   printf("\n ");
  }
```

**5. 思考题与习题**

如果按由表尾至表头的次序输入数据元素，则顺序表建立的程序应如何设计？

## 实验二　顺序表的插入

**1. 实验目的**

了解顺序表的结构特点及有关概念，掌握顺序表插入的基本操作算法。

**2. 实验内容**

在实验一所建立的 4 个元素的顺序表 list[]={2，3，4，5}的元素 4 和 5 之间插入一个元素 9，实现顺序表插入的基本操作。

**3. 实验要点及说明**

顺序表又称为线性表的顺序存储结构，它用一组地址连续的存储单元依次存放线性表的各个元素。

可定义顺序表如下：

```
#define maxnum                  /* 顺序表最大元素个数 11 */
elemtype list[maxnum];          /* 定义顺序表 list */
  int num= –1;                  /* 定义当前数据元素下标，并置初值为–1 */
```

参考程序中，函数 insertq()实现在顺序表 list[]的指定位置(i=3)插入元素 9，采用前插法。顺序表插入元素 9 示意如图 1-2 所示。

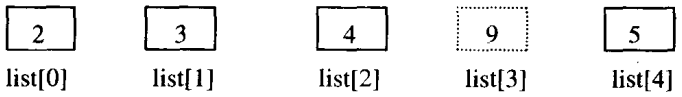



图 1-2　顺序表插入元素 9 示意

**4. 参考程序**

```
#include <stdio.h>
#define max   10
#define true   1
#define false  0
int insertq(int list[],int *num,int i,int x)          /* 在顺序表 i 位置插入 x */
 {int j;
  if((i<0)||(i>*num+1))
  {
    printf("i 值不合法。");
    return(false);
  }
  if(*num>=max-1)
   {
    printf("表已满无法再插入。");
    return(false);
   }
  for(j=*num+1;j>i;j--)
      list[j]=list[j-1];                              /* 元素依次后移 */
  list[i]=x;                                          /* 插入 x */
  (*num)++;                                           /* 当前数据元素位置(下标)加 1 */
  return(true);
 }
main()
 {int i=0,x,*num,ch;
  int list[max];
  printf("Input list:");
  while((ch=getchar())!='\n')                         /* 输入顺序表元素，以换行符结束 */
   {
    list[i]=ch;
    i++;
   }
  *num=i-1;
```

```
        printf("insert No.i:");                    /* 在 i 位置前插入 x */
        scanf("%d",&i);
        getchar();
        printf("insert data:");
        x=getchar();
        getchar();
        insertq(list,num,i,x);
        for(i=0;i<=*num;i++)
            printf("list[%d]=%c ",i,list[i]);
    printf("\n");
   }
```

5. **思考题与习题**

如何用程序实现顺序表逆置？

## 实验三　顺序表的删除

1. **实验目的**

了解顺序表的结构特点及有关概念，掌握顺序表删除的基本操作算法。

2. **实验内容**

在实验二顺序表 list[]={2，3，4，9，5}中删除指定位置(i=3)上的元素 9，实现顺序表删除的基本操作。

3. **实验要点及说明**

顺序表又称为线性表的顺序存储结构，它用一组地址连续的存储单元依次存放线性表的各个元素。

可定义顺序表如下：

```
#define maxnum                  /* 顺序表最大元素个数 11 */
elemtype list[maxnum];          /* 定义顺序表 list */
int num= -1;                    /* 定义当前数据元素下标，并置初值为-1 */
```

参考程序中，函数 deleteq()实现删除顺序表 list[]的指定位置(i=3)上的元素 9 。参考程序执行时顺序表删除元素 9 示意如图 1-3 所示。

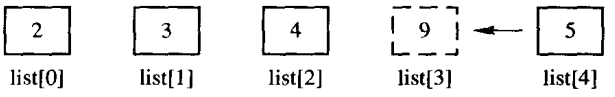



图 1-3　顺序表删除元素 9 示意

4. **参考程序**

```
#include <stdio.h>
#define max     10
#define true    1
#define false   0
```

```
int deleteq(int list[],int *num,int i)          /* 在位置 i 删除元素 */
 {int j;
  if(i<0||i>*num+1)
    {
     printf("删除位置出错。");
     return(false);
    }
 for(j=i+1;j<=*num;j++)
      list[j-1]=list[j];                        /* 数据元素依次前移 */
  (*num) --;                                    /* 当前数据元素位置(下标)减 1 */
  return(true);
 }
main()
 {int i=0,x,*num,ch;
  int list[max];
  printf("Input list:");
  while((ch=getchar())!='\n')                   /* 输入顺序表元素，以换行符结束 */
    {
     list[i]=ch;
     i++;
    }
  *num=i-1;
   printf("insert No.i:");                      /* 删除 i 位置元素 */
   scanf("%d",&i);
   deleteq(list,num,i);
   for(i=0;i<=*num;i++)
      printf("list[%d]=%c ",i,list[i]);
   printf("\n");
}
```

5. 思考题与习题

每次删除操作都会使大量的数据元素移动，删除多个数据元素时，就需多次移动数据元素。能否一次进行删除多个数据元素的操作，使得数据元素的移动只进行一次？

## 实验四　顺序表的复制

1. 实验目的

进一步了解顺序表的结构，掌握顺序表复制的基本操作算法。

2. 实验内容

复制顺序表。先构造一个顺序表 qa={2，3，4，9，5}和一个空表 qb，再将 qa 复制到

qb 中，实现顺序表复制的基本操作。

**3. 实验要点及说明**

顺序表又称为线性表的顺序存储结构，它用一组地址连续的存储单元依次存放线性表的各个元素。

可定义顺序表如下：

```
#define maxnum                  /* 顺序表最大元素个数 11 */
elemtype list[maxnum];          /* 定义顺序表 list */
int num= -1;                    /* 定义当前数据元素下标，并置初值为-1 */
```

参考程序中，函数 insertq(int list[],int *num,int i,int x)实现构造顺序表；函数 copyqlist(int qa[],int numa,int qb[],int *num) 实现将顺序表 qa 复制到 qb 中。顺序表 qa 示意如图 1-4 所示。


图 1-4 顺序表 qa 示意

**4. 参考程序**

```
#include <stdio.h>
#define max   10
#define true   1
#define false  0
int insertq(int list[],int *num,int i,int x)                /* 构造顺序表 qa */
 {int j;
  if((i<0)||(i>*num+1))
    {
      printf("i 值不合法。");
      return(false);
    }
  if(*num>=max-1)
    {
      printf("表已满无法再插入。");
      return(false);
    }
  for(j=*num+1;j>i;j--)
      list[j]=list[j-1];
  list[i]=x;
  (*num)++;
  return(true);
 }
void copyqlist(int qa[],int numa,int qb[],int *num)
```