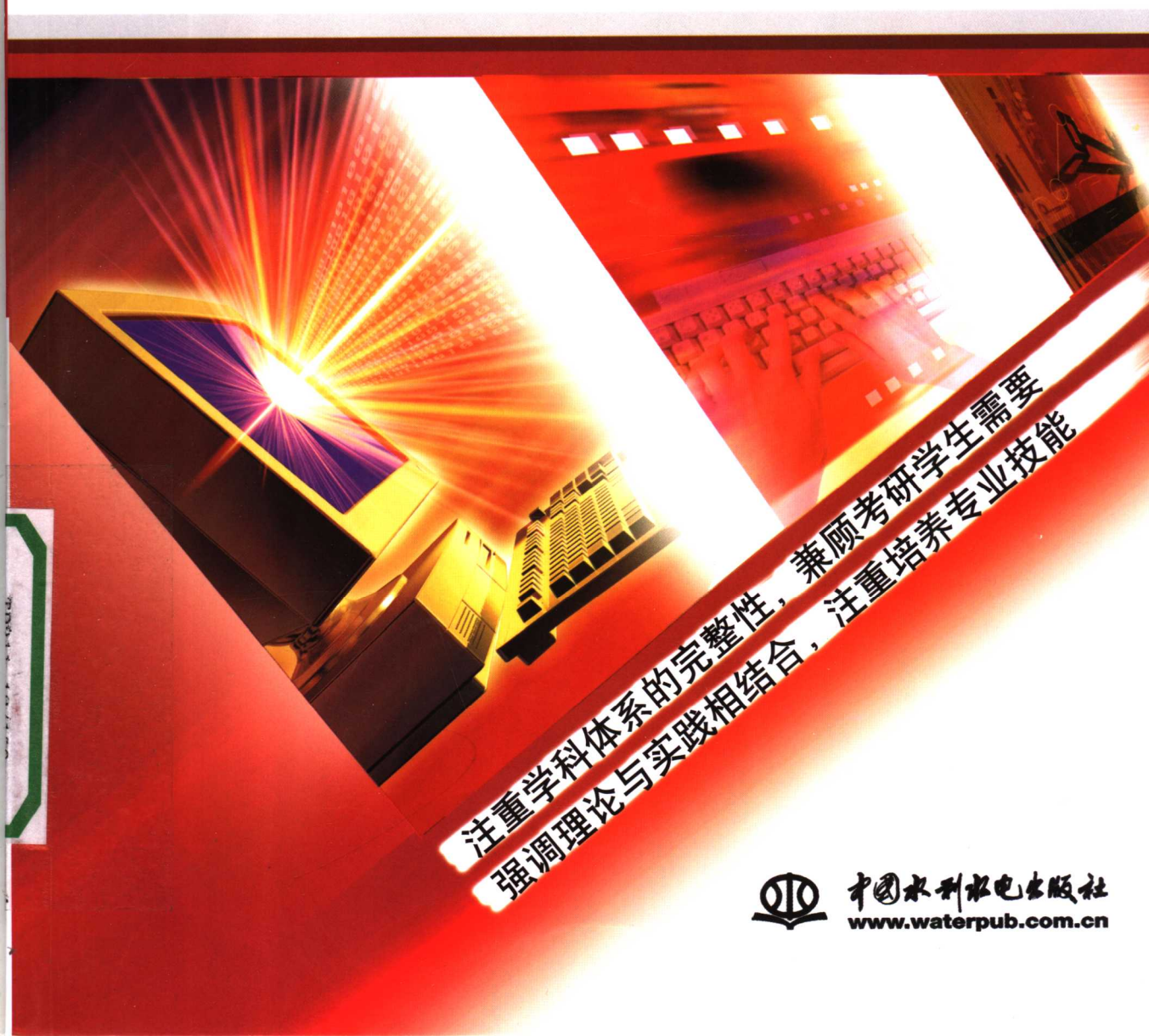
21世纪

高等院校规划教材

智东杰 主 编

数据结构实验程序



中国水利水电出版社
www.waterpub.com.cn

21世纪高等院校规划教材

数据结构实验程序

智东杰　主　编

中国水利水电出版社

内 容 提 要

本书不同于《数据结构》只讲理论，《数据结构习题与解析》是概念与理论的重复，有关程序是片段的，正确与否也不以实例验证，而本书中有丰富的数据结构程序实例，主要内容包括：数组、链表、栈和队列、二叉树、集合与搜索、图、排序等，尤其链表、二叉树、集合与搜索、图的内容更为详细。全书条理清晰、通俗易懂、图文并茂。

本书适合高等院校计算机专业、软件专业和信息管理专业的学生和教师使用，也可供计算机软件开发人员和计算机用户阅读。

本书程序源代码可以从中国水利水电出版社网站上免费下载，网址为：http://www.waterpub.com.cn/softdown/。

图书在版编目（CIP）数据

数据结构实验程序 / 智东杰主编. —北京：中国水利水电出版社，2008

21 世纪高等院校规划教材

ISBN 978-7-5084-5092-6

Ⅰ. 数… Ⅱ. 智… Ⅲ. 数据结构－高等学校－教学参考资料 Ⅳ. TP311.12

中国版本图书馆 CIP 数据核字（2007）第 172620 号

书　　名	**数据结构实验程序**
作　　者	智东杰　主　编
出版　发行	中国水利水电出版社（北京市三里河路 6 号　100044） 网址：www.waterpub.com.cn E-mail：mchannel@263.net（万水） sales@waterpub.com.cn 电话：（010）63202266（总机）、68331835（营销中心）、82562819（万水）
经　　售	全国各地新华书店和相关出版物销售网点
排　　版	北京万水电子信息有限公司
印　　刷	北京蓝空印刷厂
规　　格	787mm×1092mm　16 开本　10 印张　225 千字
版　　次	2008 年 1 月第 1 版　2008 年 1 月第 1 次印刷
印　　数	0001—4000 册
定　　价	15.00 元

序

随着计算机科学与技术的飞速发展，计算机的应用已经渗透到国民经济与人们生活的各个角落，正在日益改变着传统的人类工作方式和生活方式。在我国高等教育逐步实现大众化后，越来越多的高等院校会面向国民经济发展的第一线，为行业、企业培养各级各类高级应用型专门人才。为了大力推广计算机应用技术，更好地适应当前我国高等教育的跨跃式发展，满足我国高等院校从精英教育向大众化教育的转变，符合社会对高等院校应用型人才培养的各类要求，我们成立了“21 世纪高等院校规划教材编委会”，在明确了高等院校应用型人才培养模式、培养目标、教学内容和课程体系的框架下，组织编写了本套“21 世纪高等院校规划教材”。

众所周知，教材建设作为保证和提高教学质量的重要支柱及基础，作为体现教学内容和教学方法的知识载体，在当前培养应用型人才中的作用是显而易见的。探索和建设适应新世纪我国高等院校应用型人才培养体系需要的配套教材已经成为当前我国高等院校教学改革和教材建设工作面临的紧迫任务。因此，编委会经过大量的前期调研和策划，在广泛了解各高等院校的教学现状、市场需求，探讨课程设置、研究课程体系的基础上，组织一批具备较高的学术水平、丰富的教学经验、较强的工程实践能力的学术带头人、科研人员和主要从事该课程教学的骨干教师编写出一批有特色、适用性强的计算机类公共基础课、技术基础课、专业及应用技术课的教材以及相应的教学辅导书，以满足目前高等院校应用型人才培养的需要。本套教材消化和吸收了多年来已有的应用型人才培养的探索与实践成果，紧密结合经济全球化时代高等院校应用型人才培养工作的实际需要，努力实践，大胆创新。教材编写采用整体规划、分步实施、滚动立项的方式，分期分批地启动编写计划，编写大纲的确定以及教材风格的定位均经过编委会多次认真讨论，以确保该套教材的高质量和实用性。

教材编委会分析研究了应用型人才与研究型人才在培养目标、课程体系和内容编排上的区别，分别提出了 3 个层面上的要求：在专业基础类课程层面上，既要保持学科体系的完整性，使学生打下较为扎实的专业基础，为后续课程的学习做好铺垫，更要突出应用特色，理论联系实际，并与工程实践相结合，适当压缩过多过深的公式推导与原理性分析，兼顾考研学生的需要，以原理和公式结论的应用为突破口，注重它们的应用环境和方法；在程序设计类课程层面上，把握程序设计方法和思路，注重程序设计实践训练，引入典型的程序设计案例，将程序设计类课程的学习融入案例的研究和解决过程中，以学生实际编程解决问题的能力为突破口，注重程序设计算法的实现；在专业技术应用层面上，积极引入工程案例，以培养学生解决工程实际问题的能力为突破口，加大实践教学内容的比重，增加新技术、新知识、新工艺的内容。

本套规划教材的编写原则是：

在编写中重视基础，循序渐进，内容精炼，重点突出，融入学科方法论内容和科学理念，反映计算机技术发展要求，倡导理论联系实际和科学的思想方法，体现一级学科知识组织的层次结构。主要表现在：以计算机学科的科学体系为依托，明确目标定位，分类组织实施，兼容互补；理论与实践并重，强调理论与实践相结合，突出学科发展特点，体现

学科发展的内在规律；教材内容循序渐进，保证学术深度，减少知识重复，前后相互呼应，内容编排合理，整体结构完整；采取自顶向下设计方法，内涵发展优先，突出学科方法论，强调知识体系可扩展的原则。

本套规划教材的主要特点是：

（1）面向应用型高等院校，在保证学科体系完整的基础上不过度强调理论的深度和难度，注重应用型人才的专业技能和工程实用技术的培养。在课程体系方面打破传统的研究型人才培养体系，根据社会经济发展对行业、企业的工程技术需要，建立新的课程体系，并在教材中反映出来。

（2）教材的理论知识包括了高等院校学生必须具备的科学、工程、技术等方面的要求，知识点不要求大而全，但一定要讲透，使学生真正掌握。同时注重理论知识与实践相结合，使学生通过实践深化对理论的理解，学会并掌握理论方法的实际运用。

（3）在教材中加大能力训练部分的比重，使学生比较熟练地应用计算机知识和技术解决实际问题，既注重培养学生分析问题的能力，也注重培养学生思考问题、解决问题的能力。

（4）教材采用“任务驱动”的编写方式，以实际问题引出相关原理和概念，在讲述实例的过程中将本章的知识点融入，通过分析归纳，介绍解决工程实际问题的思想和方法，然后进行概括总结，使教材内容层次清晰，脉络分明，可读性、可操作性强。同时，引入案例教学和启发式教学方法，便于激发学习兴趣。

（5）教材在内容编排上，力求由浅入深，循序渐进，举一反三，突出重点，通俗易懂。采用模块化结构，兼顾不同层次的需求，在具体授课时可根据各校的教学计划在内容上适当加以取舍。此外还注重了配套教材的编写，如课程学习辅导、实验指导、综合实训、课程设计指导等，注重多媒体的教学方式以及配套课件的制作。

（6）大部分教材配有电子教案，以使教材向多元化、多媒体化发展，满足广大教师进行多媒体教学的需要。电子教案用 PowerPoint 制作，教师可根据授课情况任意修改。相关教案的具体情况请到中国水利水电出版社网站 www.waterpub.com.cn 下载。此外还提供相关教材中所有程序的源代码，方便教师直接切换到系统环境中教学，提高教学效果。

总之，本套规划教材凝聚了众多长期在教学、科研一线工作的教师及科研人员的教学科研经验和智慧，内容新颖，结构完整，概念清晰，深入浅出，通俗易懂，可读性、可操作性和实用性强。本套规划教材适用于应用型高等院校各专业，也可作为本科院校举办的应用技术专业的课程教材，此外还可作为职业技术学院和民办高校、成人教育的教材以及从事工程应用的技术人员的自学参考资料。

我们感谢该套规划教材的各位作者为教材的出版所做出的贡献，也感谢中国水利水电出版社为选题、立项、编审所做出的努力。我们相信，随着我国高等教育的不断发展和高校教学改革的不断深入，具有示范性并适应应用型人才培养的精品课程教材必将进一步促进我国高等院校教学质量的提高。

我们期待广大读者对本套规划教材提出宝贵意见，以便进一步修订，使该套规划教材不断完善。

21 世纪高等院校规划教材编委会

2004 年 8 月

前　言

计算机加工处理的对象是数据，而数据具有一定的结构，所以编写计算机程序仅仅掌握计算机语言还不够，还必须掌握数据组织、存储和运算的方法，这便是“数据结构”课程所学习和研究的内容，它为编写计算机程序提供良好的基础，因此，“数据结构”被列为计算机等相关专业最重要的专业基础课程，特别对计算机学科来说，起到承前启后的作用。由于数据结构的原理和算法较为抽象，使很多同学难以掌握，特别是该课程一般在低年级开设，对于仅仅具有一些计算机语言程序设计知识的初学者，理解和掌握其中的原理就更困难了，在解答数据结构的习题时，往往感到无从下手。为了给读者一些启发，我编写了本书，目的是：通过对本书的阅读、补充、修改和参照数据结构习题的练习，使学生充分掌握数据结构的原理以及求解数据结构问题的思路与方法，深化对基本概念的理解，提高分析与解决问题的能力。

本书遵循“数据结构”课程的教学习惯，内容分 7 章：第 1 章给出了数组 C++程序、顺序表 C++程序和字符串 C++程序；第 2 章给出了带头结点的单链表 C 程序、不带头结点的单链表 C 程序、循环链表的 C 程序、不带表头结点的单链表 C++程序、用模板定义的带头结点的单链表 C++程序和单链表的游标（Iterator）类的 C++程序；第 3 章给出了栈的 C 程序、链式栈的 C++程序和实现链队列的 C 程序；第 4 章给出了二叉树的 Turbo Pascal（5.5）语言程序、二叉树的 C++程序及实例、线索二叉树 Turbo Pascal（5.5）语言程序和哈夫曼树及应用 C 程序；第 5 章给出了用位向量实现集合运算的 C++程序、用有序链表实现集合运算的 C++程序、顺序搜索 C++程序、基于有序顺序表的折半搜索的 C++程序、二叉搜索树的 C++程序和平衡二叉搜索树（AVL）的 C++程序；第 6 章给出了邻接表的 C 程序、十字链表的 C 程序、图的连通性的 C 程序、拓扑排序的 C 程序、关键路径的 C 程序、邻接表/图的深度优先搜索的 C++程序、用顶点表示活动的网络（AOV 网络）的 C++程序和用边表示活动的网络（AOE 网络）的 C++程序；第 7 章给出了直接插入排序的 Turbo Pascal 程序、折半插入排序（Binary Insertion Sort）程序、希尔排序的 Turbo Pascal 语言程序、选择排序的 Turbo Pascal 程序和堆排序的 Turbo Pascal 程序。

本书谈到的 C 程序是指 Turbo C 2.0 版本，Turbo Pascal 程序是指 Turbo Pascal 5.5 版本，C++程序是指 Visual C++ 6.0 版本。

本书中可能存在不准确或不完整的地方，内容编排上可能存在不合理之处，敬请广大读者批评指正。作者 E-mail：zhidongjie@126.com。

作　者

2007 年 10 月

目　　录

第 1 章　数组

定义　一个一维数组为具有相同数据类型的 n（n≥0）个元素的有限序列，其中的 n 叫做数组长度或数组的大小，若 n=0 则是空数组。一维数组也可以称为向量。

1.1　数组的 C++程序

```
const int defaultsize=10;
# include <iostream.h>
# include <stdlib.h>

template <class type> class array {
    public:
        array(int size = defaultsize);                    //构造函数
        ~array() {delete [ ] elements;}                   //析构函数
        type & operator[ ] (int i);                       //下标
        array <type> & operator = (const array <type> & x);        //数组复制
        type * operator * () const { return elements;}             //指针转换
        int length() const { return arraysize;}           //取数组长度
        void resize (int sz);       //修改数组长度
    private:
        type * elements;            //数组
        int arraysize;              //元素个数
        void getarray();            //动态分配数组存储空间
};

template <class type> void array<type> :: getarray() {
    elements= new type[arraysize];
    if (elements == 0) cerr << "Memory Allocation Error."<<endl;
}

template <class type> array<type> :: array(int sz) {
    if (sz <=0 ) cerr << "Invalid Array Size."<<endl;
    arraysize = sz;
```

```
    getarray();
}

template <class type> array<type> & array<type> :: operator = (const array<type> & x){
    int n;
    int arraysize = n = x.arraysize;
    elements = new type[n];
    if (elements == 0 ) cerr <<"Memory Allocation Error."<<endl;
    type * srcptr = x.elements;
    type * destptr = elements;
    while (n--) * destptr ++ = * srcptr ++;
    return   * this;
}

template <class type> type & array<type> :: operator [ ](int i) {
    if (i< 0 || i>arraysize - 1) cerr <<"Index out of Range." << endl;
    return elements[i];
}

template <class type> void   array <type> ::resize(int sz) {
    if (sz <= 0 ) cerr << "Invalid Array Size."<< endl;
    if (sz != arraysize) {
        type * newarray = new type [sz];
        if (newarray == 0) cerr << "Memory Allocation Error."<< endl;
        int n= (sz <= arraysize) ? sz : arraysize;
        type * srcptr = elements;
        type * destptr = newarray;
        while (n--) * destptr ++ = * srcptr ++;
        delete [ ] elements;
        elements = newarray ; arraysize = sz;
    }
}

void main()
{
    array<int> a,b(5),c(12); //定义数组为整型，a 缺省，b 有 5 个元素，c 有 12 个元素
    int len,i;
```

```
    len=a.length();
    for (i=0;i<len;i++) {
        cout<<"a["<<i<<"]="; cin >>a[i];
    }
    cout <<"Array a: ";
    for (i=0; i<len; i++) cout << a[i]<<"   "; cout<<endl;
    len=b.length();
    for (i=0;i<len;i++) {
        cout<<"b["<<i<<"]="; cin >> b[i];
    }
    cout <<"Array b: ";
    for (i=0; i<len; i++) cout << b[i]<<"   "; cout<<endl;
    len=c.length();
    for (i=0;i<len;i++) {
        cout<<"c["<<i<<"]="; cin >> c[i];
    }
    cout <<"Array c: ";
    for (i=0; i<len; i++) cout << c[i]<<"   "; cout<<endl;
    array <int> d(12);          //第 12 个元素
    d=c;                        //拷贝
    len=d.length();
    cout <<"Array d: ";
    for (i=0; i<len; i++) cout << d[i]<<"   "; cout<<endl;
    d.resize(6);                //修改数组长度
    len=d.length();
    cout <<"Array d: ";
    for (i=0; i<len; i++) cout << d[i]<<"   "; cout<<endl;
}
```

运行如下：

```
a[0]=1,…,a[9]=10
Array a: 1  2  3  4  5  6  7  8  9  10
b[0]=11,…,b[4]=15
Array b: 11  12  13  14  15
c[0]=21,c[1]=22,…,c[11]=32
Array c: 21 22 23 24 25 26 27 28 29 30 31 32
Array d: 21 22 23 24 25 26 27 28 29 30 31 32
Array d: 21 22 23 24 25 26
```

1.2 顺序表

1.2.1 顺序表描述

顺序表是一种线性聚集。它是一个顺序存储的 n（n≥0）个表项的序列，其中 n 是表的长度，可以是任意整数。n=0 时，称为空表。

顺序表的特点是：为了得到顺序表中所要求的表项，必须从表的第一个表项开始逐个访问表项，直到找到满足要求的表项为止，也就是说，对于顺序表只能顺序存取。

1.2.2 顺序表的 C++程序

```
const int defaultsize=8;
template <class type>
class seqlist
{
    private:
        type *data;                          //顺序表的存放数组
        int maxsize;                         //顺序表的最大可容纳项数
        int last;                            //顺序表当前已存表项的最后位置
    public:
        seqlist(int maxsize=defaultsize);    //构造函数
        ~seqlist() {delete [ ] data;}        //析构函数
        int length() const {return last +1;} //计算表长度
        int find (type & x) const;           //定位函数：找 x 在表中的位置
        int isin (type & x);                 //判断 x 是否在表中
        int insert (type & x,int i);         //插入 x 在表中的第 i 个位置处
        int remove (type & x);               //删除 x
        int next (type & x);                 //寻找 x 的后继
        int prior (type & x);                //寻找 x 的前趋
        int isempty () { return last==-1;}   //判断顺序表是否空，空则返回 1；否则返回 0
        int isfull () {return last == maxsize-1;} //判断顺序表是否满，满则返回 1；否则
                                             //返回 0
        type * get(int i) {return i <0 || i>last ? NULL:&data[i];}   //取第 i 个元素的值
        type & operator [ ] (int i) {return data[i];}
};

template <class type> seqlist <type> :: seqlist (int sz)
```

```
{    //构造函数，通过指定参数 sz 定义数组长度
     if (sz>0){
          maxsize = sz; last=-1;
          data=new type[maxsize];
     }
}

template <class type> int seqlist <type>::find (type& x) const
{    //定位：找 x 在表中的位置，若查找成功，函数返回表项的位置；否则函数返回-1
     int i=0;
     while (i<=last && data[i] !=x) i++;
     if (i>last) return -1;
     else return i;
}

template <class type> int seqlist<type>::isin(type & x)
{    //判断 x 是否在表中
     int i=0, found =0;
     while (i<=last && ! found)
          if (data[i] != x) i++;
          else found = 1;
     return found;
}

template <class type> int seqlist <type> ::insert(type & x,int i)
{    //插入 x 在表中的第 i 个位置处，函数返回插入是否成功的信息，若为 0 则插入不成功
     if (i<0||i>last+1 ||last==maxsize-1) return 0;
     else{
          last ++;
          for (int j=last ;j>i;j--) data[j]=data[j-1];
          data[i]=x;
          return 1;
     }
}
template <class type > int seqlist <type> ::remove(type & x)
{    //删除 x
     int i=find(x);              //在表中查找 x
```

```
    if (i>=0){                  //x 在表中存在
        last--;                 //表长度减 1
        for (int j=i;j<=last;j++)   data[j]=data[j+1];      //依此前移
        return 1;
    }
    return 0;                   //x 在表中不存在，不能删除
}

template <class type> int seqlist <type>::next(type & x)
{   //寻找 x 的后继数据
    int i=find(x);
    if (i>=0 && i<last) return i+1;     //x 的后继存在
    else return -1;                     //x 不在表中或 x 的后继不存在
}
template <class type> int seqlist<type>::prior(type & x)
{   //寻找 x 的前趋
    int i=find (x);
    return i-1;                 //x 的前趋位置
}

#include <iostream.h>
void main()
{
    seqlist<int> sl(10);
    int elem;
    int len=sl.length();
    cout <<"length="<< len <<endl;
    //顺序表长
    for (int i=0;i<5;i++)
    {
        cout <<"sl["<<i<<"]=";
        cin >> sl[i];
    }
    for (i=0;i<5;i++)
        cout <<sl[i]<< "   ";
    cout <<"\n";
    //插入 5 个整数
```

```
    for (i=0;i<5;i++)
    {
        cout <<"input integer: ";
        cin >> elem;
        if (sl.insert(elem,i)==1) cout <<"insert success!\n";
        else cout <<"no success!\n";
        cout <<"length="<<sl.length()<<"\n";
    }
    cout <<"seqlist is: \n";
    for (i=0;i<5;i++)
        cout <<sl[i]<< "   ";
    cout <<"\n";
    cout <<"在第 i 个位置上插一个整数: \n";
    cout <<"i=";cin >>i; cout <<"\n";
    cout <<"elem="; cin >>elem; cout<<"\n";
    if (sl.insert(elem,i)==1) cout <<"insert success!\n";
    else cout <<"no success!\n";
    cout <<"length="<<sl.length()<<"\n";
    cout <<"seqlist is: \n";
    for (i=0;i<sl.length();i++)
        cout <<sl[i]<< "   ";
    cout <<"\n";
    cout <<"查找元素: "; cin >>elem;
    int loc=sl.find(elem);
    if(sl.remove(elem))
    {
        cout <<"删除成功!\n";
        cout <<"删除该元素后顺序表是: \n";
        cout <<"seqlist is: \n";
        for (i=0;i<sl.length();i++)
            cout <<sl[i]<< "   ";
        cout <<"\n";
        cout <<"length="<<sl.length()<<"\n";
    }
    else cout <<"没找到!\n";
}
```

运行结果如下：

```
length=0
sl[0]=10
sl[1]=11
sl[2]=12
sl[3]=13
sl[4]=14
10   11   12   13   14
input integer:      24
inset success !
length=1
input integer:      46
inset success !
length=2
input integer:      21
inset success !
length=3
input integer:      76
inset success !
length=4
input integer:      77
inset success !
length=5
sequist is
24    46    21    76    77
在第 i 个位置上插入一个整数
i=3          //位置从 0 开始
elem=50
insert success!
length=6
seqlist is
24   46    21    50    76    77
查找元素:      50
删除成功!
删除该元素后顺序表是:
seqlist is:
24   46   21   76    77
length = 5
```