“十二五”普通高等教育本科国家级规划教材　计算机系列教材

宁波市特色教材

数据结构与算法实验指导书

汪沁　邓芳　奚李峰　主编

清华大学出版社

“十二五”普通高等教育本科国家级规划教材　计算机系列教材

汪 沁　邓 芳　奚李峰　主编

数据结构与算法实验指导书

清华大学出版社
北京

内 容 简 介

本书是数据结构课程的辅助教材，采用C和C++两种语言来描述数据结构，让学生在实验与习题中体会与掌握数据结构，同时培养编程能力和分析能力。主要内容包括实验与习题两大部分，用于巩固数据结构的理论知识，提高实践应用能力。

本书内容立足于高校教学的要求，适用于本科院校的课程和学生群体，可作为数据结构与算法课程的辅助教材，也可作为初学数据结构读者的自学读物。

图书在版编目(CIP)数据

数据结构与算法实验指导书/汪沁，邓芳，奚李峰主编. —北京：清华大学出版社，2018
(计算机系列教材)
ISBN 978-7-302-49944-2

Ⅰ. ①数… Ⅱ. ①汪… ②邓… ③奚… Ⅲ. ①数据结构－高等学校－教材 ②算法分析－高等学校－教材 Ⅳ. ①TP311.12

中国版本图书馆CIP数据核字(2018)第066133号

责任编辑：张　民　李　晔
封面设计：常雪影
责任校对：梁　毅
责任印制：董　瑾

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦A座　　**邮　　编**：100084
社 总 机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课件下载：http://www.tup.com.cn，010-62795954
印 装 者：北京密云胶印厂
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：7.5　　**字　　数**：178千字
版　　次：2018年9月第1版　　**印　　次**：2018年9月第1次印刷
定　　价：19.00元

产品编号：076395-01

前　言

数据结构是计算机专业的核心课程，它从长期的程序设计实践中提炼而成，运用于程序设计；更是操作系统、编译原理等计算机核心课程的基础，在计算机专业课程中起着承上启下的作用。

数据结构与算法的原理比较抽象，概念性强，难度大，不易掌握，但同时也具有较强的可应用性和实践性。实验是一个重要的教学环节，通过实现原理与算法，并将实验结果反馈到原理与算法中去，有助于理解。

各种数据结构以及相应算法的描述总是要选用一种语言工具。本书兼顾面向对象及面向过程两种编程思想，采用了C语言和C++两种语言来实现数据结构以及相应算法。

本书分为四个部分：第一部分绪论，要求学习者养成良好的实验习惯；第二部分和第三部分分别采用C语言和C++语言来描述线性结构、栈、队列、串与数组、树与二叉树、图、查找和排序等数据结构及算法；第四部分为数据结构每章的习题练习。

本书中所有算法的实现均通过实验验证可行。希望读者通过本书的学习，能更加深入地理解和掌握数据结构的相关知识。

由于编者水平有限，书中难免存在差错，敬请广大读者批评指正。作者的电子邮箱地址：qinwang@126.com。

作　者

2018年3月

目　录

第 1 部分　实验要求及规范

数据结构课程具有比较强的理论性，同时也具有较强的应用性和实践性。上机实验是一个重要的教学环节。一般情况下学生能够重视实验环节，对于编写程序上机练习具有一定的积极性；但是容易忽略实验的总结，忽略实验报告的撰写。对于一名大学生来说，必须严格训练分析总结能力和书面表达能力。需要逐步培养编写科学实验报告以及科技论文的能力。拿到一个题目，一般不要急于编程。按照程序设计思路，正确的方法是：首先理解问题，明确给定的条件和要求解决的问题，然后按照自顶向下、逐步求精、分而治之的策略，逐一解决子问题。具体实践步骤如下。

1. 问题分析与系统结构设计

充分地分析和理解问题本身，弄清要求做什么（而不是怎么做），限制条件是什么。按照以数据结构为中心的原则划分模块，搞清数据的逻辑结构（是线性表，还是树、图），确定数据的存储结构（是顺序结构，还是链表结构）。然后设计有关操作的函数。在每个函数模块中，要综合考虑系统功能，使系统结构清晰、合理、简单和易于调试。最后写出每个模块的算法头和规格说明，列出模块之间的调用关系（可以用图表示），便完成了系统结构设计。

2. 详细设计和编码

详细设计是对函数（模块）的进一步求精，用伪高级语言（如类 C 语言）或自然语言写出算法框架，这时不必确定很多结构和变量。

编码，即程序设计，是对详细设计结果的进一步求精，即用某种高级语言（如 C/C++ 语言）表达出来。尽量多加一些注释语句，以使程序清晰易懂。尽量临时增加一些输出语句，以便于差错纠正，在程序编译成功后再删去它们。

3. 上机准备

熟悉高级语言用法，如 C 语言。熟悉操作系统的常用命令。静态检查主要有两条路径：一是用一组测试数据手工执行程序（或分模块进行）；二是通过阅读或给别人讲解自己的程序而深入全面地理解程序逻辑，在这个过程中再加入一些注释和断言。如果程序中逻辑概念清楚，后者将比前者有效。

4. 上机调试程序

调试最好分块、自底向上进行，即先调试底层函数，必要时可以另写一个调用驱动程序，表面上麻烦的工作可以大大降低调试时所面临的复杂性，从而提高工作效率。

5. 整理实习报告

在上机实践开始之前要充分准备实验数据，在上机实践过程中要及时记录实验数据，在上机实践完成之后必须及时总结分析，并写出实验报告。

1）实验报告的基本要求

一般性、较小规模的上机实验题，必须遵循下列要求，养成良好的习惯。

（1）姓名、班级、学号、日期。

（2）题目：内容叙述。

（3）程序清单（带有必要的注释）。

（4）调试报告。

实验者必须重视这一环节，否则等同于没有完成实验任务。这里可以体现个人特色或创造性思维。具体内容包括：测试数据与运行记录；调试中遇到的主要问题，自己是如何解决的；经验和体会等。

2）实验报告的提高要求

阶段性、较大规模的上机实验题，应该遵循下列要求，养成科学的习惯。

（1）需求和规格说明。

描述问题，简述题目要解决的问题是什么。规定软件做什么。原题条件不足时应补全。

（2）设计。

- 设计思想：存储结构（题目中限定的要描述）；主要算法基本思想。
- 设计表示：每个函数的头和规格说明；列出每个函数所调用和被调用的函数，也可以通过调用关系图表达。
- 实现注释：说明各项功能的实现程度，在完成基本要求的基础上还有什么功能。

（3）用户手册：即使用说明。

（4）调试报告：调试过程中遇到的主要问题是如何解决的；设计的回顾、讨论和分析；时间复杂度、空间复杂度分析；改进设想；经验和体会等。

第 2 部分　面向过程语言实现数据结构

实验 0　复数 ADT 及其实现

1. 实验目的

(1) 了解抽象数据类型(ADT)的基本概念及描述方法。

(2) 通过对复数抽象数据类型 ADT 的实现，熟悉 C 语言语法及程序设计，为以后章节的学习打下基础。

2. 实例

复数抽象数据类型 ADT 的描述及实现。

[复数 ADT 的描述]

```
ADT complex{
    数据对象:D={ c1,c2   c1,c2∈FloatSet }
    数据关系:R={ <c1,c2>  c1   c2    }
    基本操作:创建一个复数          creat(a);
             输出一个复数          outputc(a);
             求两个复数相加之和    add(a,b);
             求两个复数相减之差    sub(a,b);
             求两个复数相乘之积    chengji(a,b);
             等等;
         } ADT complex;
```

[复数 ADT 实现的源程序]

```
#include <stdio.h>
#include <stdlib.h>
/* 存储表示,结构体类型的定义 */
typedef  struct
  { float x;                          /* 实部子域 */
    float y;                          /* 虚部的实系数子域 */
  }comp;
/* 全局变量的说明 */
comp a,b,a1,b1;
int z;
/* 子函数的原型声明 */
void creat(comp *c);
void outputc(comp a);
```

```
comp add(comp k,comp h);
/* 主函数*/
main( )
{ creat(&a);   outputc(a);
  creat(&b);   outputc(b);
  a1=add(a,b); outputc(a1);
}  /* main*/
/*创建一个复数*/
void creat(comp *c)
{ float c1,c2;
  printf("输入实部 real x=?");scanf("%f",&c1);
  printf("输入虚部 xvpu y=?");scanf("%f",&c2);
  (*c).x=c1;  c->y=c2;
}  /* creat*/
/*输出一个复数*/
void outputc(comp a)
  {if(a.y<0) printf("\n  %f%f i \n\n",a.x,a.y);
else  printf("\n  %f+%f i \n\n",a.x,a.y);
   }
/*求两个复数相加之和*/
comp add(comp k,comp h)
{ comp l;
  l.x=k.x+h.x;  l.y=k.y+h.y;
  return(l);
  }  /* add*/
```

3. 实习题

首先将上面的源程序输入计算机，进行调试。运行程序，输入下列两个复数的实部和虚部，记录两个复数相加的输出结果。原始数据如下：

2.0+3.5i,3.0-6.3i

然后在上面程序的基础上，增加自行设计的复数减、复数乘的两个子函数，适当补充必需的语句(例如函数原型声明、主函数中的调用等)。提示：

```
// 求两个复数相减之差的函数
comp sub(comp k,comp h) { … }
// 求两个复数相乘之积的函数
comp chengji(comp k,comp h){ … }
```

再次调试运行程序。输入数据，记录结果，最后完成实验报告。

实验 1　线性表(顺序表)

1. 实验目的

(1) 了解线性表的逻辑结构特性，以及这种特性在计算机内的顺序存储结构。

(2) 重点是线性表的基本操作在顺序存储结构上的实现。

2. 实例

阅读下列程序请注意以下几个问题：

(1) 关于线性表的顺序存储结构的本质是：逻辑上相邻的两个数据元素 a_{i-1} 和 a_i 在存储地址中也是相邻的，即地址连续。不同的教材对此有不同的表示，有的直接采用一维数组，但这种方法有些过时。有的采用含“动态分配”一维数组的结构体，这种方法过于灵活抽象(对读者要求过高)。我们采用的是含“静态”一维数组和线性表长的结构体：

```
typedef  struct
  { ElemType  a[MAXSIZE];          /* 一维数组子域 */
    int  length;                   /* 表长度子域 */
  }SqList;                         /* 顺序存储的结构体类型 */
```

(2) 本程序是一个完整的、子函数较多的源程序。目的是为学生提供一个示范，提供顺序存储表示的资料，供学生参考。例如，主函数中的“菜单设计”(do-while 循环内嵌套一个 switch 结构)技术。在学习数据结构的初级阶段，并不强求学生一定使用“菜单设计”技术，可以在 main( )函数中直接写几个简单的调用语句，就像前面的复数处理程序中的 main( )一样。但是随着学习的深入，尽早学会使用“菜单设计”技术，会明显提高编程和运行效率。

[源程序]

```
#include <stdio.h>
#include <stdlib.h>
#define MAXSIZE 20                 /* 数组最大界限 */
typedef int ElemType;              /* 数据元素类型 */
typedef  struct
   { ElemType  a[MAXSIZE];         /* 一维数组子域 */
     int  length;                  /* 表长度子域 */
   }SqList;                        /* 顺序存储的结构体类型 */
SqList  a,b,c;
/* 函数声明 */
void creat_list(SqList *L);
void out_list(SqList L);
void insert_sq(SqList *L,int i,ElemType e);
ElemType delete_sq(SqList *L,int i);
int locat_sq(SqList L,ElemType e);
/* 主函数 */
main( )
{ int i,k,loc; ElemType e,x; char ch;
  do { printf("\n\n\n");
       printf("\n     1. 建立线性表 " );
       printf("\n     2. 在 i 位置插入元素 e");
```

```
        printf("\n     3. 删除第 i 个元素,返回其值");
        printf("\n     4. 查找值为 e 的元素");
        printf("\n     5. 结束程序运行");
        printf("\n=========================================");
        printf("\n    请输入您的选择(1,2,3,4,6)");
        scanf("%d",&k);
        switch(k)
      { case 1:{ creat_list(&a);  out_list(a);
               } break;
        case 2:{ printf("\n i,e=?"); scanf("%d,%d",&i,&e);
                 insert_sq(&a,i,e);   out_list(a);
               } break;
        case 3:{ printf("\n i=?");  scanf("%d",&i);
                 x=delete_sq(&a,i);  out_list(a);
                 printf("\n x=%d",x);
               } break;
        case 4:{ printf("\n e=?"); scanf("%d",&e);
                 loc=locat_sq(a,e);
                 if (loc==-1) printf("\n 未找到 %d",loc);
                    else printf("\n 已找到,元素位置是 %d",loc);
               } break;
        } /* switch */
      }while(k!=6);
      printf("\n              再见!");
      printf("\n          按 Enter 键,返回。"); ch=getch( );
    } /* main */
/* 建立线性表 */
void creat_list(SqList *L)
{ int i;
   printf("\n n=?"); scanf("%d",&L->length);
   for(i=0;i<L->length;i++){ printf("\n data %d=?",i);
                             scanf("%d",&(L->a[i]));
                           }
} /* creat_list */
/* 输出线性表 */
void out_list(SqList L)
{ int i; char ch;
   printf("\n");
   for(i=0;i<=L.length-1;i++) printf("%10d",L.a[i]);
   printf("\n\n    按 Enter 键,继续。"); ch=getch( );
} /* out_list */
/* 在线性表的第 i 个位置插入元素 e */
void insert_sq(SqList *L,int i,ElemType e)
{ int j;
```

```
  if (L->length==MAXSIZE) printf("\n overflow !");
  else if(i<1||i>L->length+1)  printf("\n erroe i !");
       else { for(j=L->length-1; j>i-1; j--) L->a[j+1]=L->a[j];
                                        /* 向后移动数据元素 */
              L->a[i-1]=e;              /* 插入元素 */
              L->length++;              /* 线性表长加 1 */
              }
  } /* insert_sq */
/* 删除第 i 个元素,返回其值 */
ElemType delete_sq(SqList *L, int i)
{ ElemType x; int j;
  if( L->length==0) printf("\n  是空表。underflow !");
  else if(i<1||i>L->length){ printf("\n error i !");
                             x=-1;}
       else {   x=L->a[i-1];
                for(j=i; j<=L->length-1; j++) L->a[j-1]=L->a[j];
                L->length--;
            }
return(x);
} /* delete_sq */
/* 查找值为 e 的元素,返回它的位置 */
int locat_sq(SqList L, ElemType e)
{ int i=0;
  while(i<=L.length-1 && L.a[i]!=e) i++;
  if(i<=L.length-1)  return(i+1);
             else return(-1);
}/* locat_sq */
```

实验 2　线性表(链表)

1. 实验目的

(1) 了解线性表的逻辑结构特性,以及这种特性在计算机内的链表存储结构。

(2) 重点是线性表的基本操作在链表结构上的实现,其中,以链表的操作为侧重点,进一步学习结构化的程序设计方法。

2. 实例

阅读下列程序时请注意几个问题。

(1) 线性表的链表存储结构的本质是:逻辑上相邻的两个数据元素 a_{i-1} 和 a_i 在存储地址中可以不相邻,即地址不连续。不同教材中的表示基本是一致的。

```
typedef  struct LNode
```

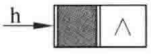
```
  { ElemType data;                    /* 数据子域 */
    struct LNode *next;               /* 指针子域 */
  }LNode;                             /* 结点结构类型 */
```

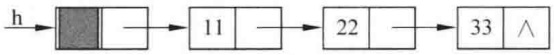
(2) 本程序是一个完整的、子函数较多的源程序。目的是提供一个示范，提供关于链表操作的资料，供学生参考。可以看到本程序的 main( )与前一程序的 main( )函数的结构十分相似。稍加改动还可为其他题目的源程序所用。

[源程序]

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
typedef int ElemType;
typedef  struct LNode
   { ElemType data;                                  /* 数据子域 */
     struct LNode *next;                             /* 指针子域 */
   }LNode;                                           /* 结点结构类型 */
LNode *L;
/* 函数声明 */
LNode *creat_L( );
void  out_L(LNode *L);
void  insert_L(LNode *L,int i, ElemType e);
ElemType delete_L(LNode *L,int i);
int locat_L(LNode *L,ElemType e);
/* 主函数 */
main( )
{ int i,k,loc; ElemType e,x; char ch;
  do { printf("\n\n\n");
       printf("\n\n     1. 建立线性链表 ");
       printf("\n\n     2. 在 i 位置插入元素 e");
       printf("\n\n     3. 删除第 i 个元素,返回其值");
       printf("\n\n     4. 查找值为 e 的元素");
       printf("\n\n     5. 结束程序运行");
       printf("\n=======================================");
       printf("\n     请输入您的选择 (1,2,3,4,5)");  scanf("%d",&k);
       switch(k)
        { case 1:{  L=creat_L( );  out_L(L);
                 } break;
          case 2:{ printf("\n i,e=?"); scanf("%d,%d",&i,&e);
                   insert_L(L,i,e);   out_L(L);
                 } break;
          case 3:{ printf("\n i=?");  scanf("%d",&i);
                   x=delete_L(L,i);  out_L(L);
                   if(x!=-1) printf("\n x=%d\n",x);
```

```
            } break;
        case 4:{ printf("\n e=?"); scanf("%d",&e);
                 loc=locat_L(L,e);
                 if (loc==-1) printf("\n 未找到 %d",loc);
                   else printf("\n 已找到,元素位置是 %d",loc);
               } break;
      } /* switch */
      printf("\n ----------------");
    }while(k>=1 && k<5);
    printf("\n                再见!");
    printf("\n      按 Enter 键,返回。"); ch=getch();
} /* main */
/* 建立线性链表 ( 11,22, 33 ) */
LNode *creat( )
{ LNode *h,*p,*s;  ElemType x;
  h=(LNode *)malloc(sizeof(LNode));          /* 分配头结点 */
  h->next=NULL;
  p=h;
  printf("\n  data=?");  scanf("%d",&x);      /* 输入第一个数据元素 */
  while( x!=-111)                            /* 输入-111,结束循环 */
    { s=(LNode *)malloc(sizeof(LNode));      /* 分配新结点 */
      s->data=x;  s->next=NULL;
      p->next=s;  p=s;
      printf("data=?( -111 end) "); scanf("%d",&x); /* 输入下一个数据 */
    }
   return(h);
} /* creat_L */
```





建成后的链表 h

```
/* 输出单链表中的数据元素 */
void out_L(LNode *L)
{ LNode *p; char ch;
  p=L->next;    printf("\n\n");
  while(p!=NULL) {  printf("%5d",p->data); p=p->next;
                 };
  printf("\n\n 按 Enter 键,继续。"); ch=getch();
  } /* out_link */
/* 在 i 位置插入元素 e */
void insert_L(LNode *L,int i, ElemType e)
{ LNode *s,*p,*q;  int j;
  p=L;                                       /* 找第 i-1 个结点 */
  j=0;
  while(p!=NULL && j<i-1) { p=p->next; j++; }
  if(p==NULL || j>i-1) printf("\n i ERROR !");
  else { s=(LNode *)malloc(sizeof(LNode));
```

```
        s->data=e;
        s->next=p->next;
        p->next=s;
      }
} /* insert_L */
/* 删除第 i 个元素,返回其值 */
ElemType delete_L(LNode *L,int i)
{ LNode *p,*q; int j; ElemType x;
  p=L; j=0;
  while(p->next!=NULL && j<i-1){ p=p->next; j++;}
  if(p->next==NULL) {printf("\n i ERROR !"); return(-1);}
    else { q=p->next; x=q->data;
           p->next=q->next; free(q);
           return(x);
         }
  }  /* delete_L */
/* 查找值为 e 的元素, 返回它的位置 */
int locat_L(LNode *L,ElemType e)
{ LNode *p; int j=1;
  p=L->next;
  while(p!=NULL && p->data!=e) {p=p->next; j++;}
  if(p!=NULL)return(j); else return(-1);
} /* locat_L */
```

(3) 约瑟夫问题的一种描述：编号为 1,2,…,n 的 n 个人按顺时针方向围坐一圈，每人持有一个密码(正整数)。一开始任选一个正整数作为报数的上限值 m，从第一个人开始按顺时针方向自 1 开始顺序报数。

方法 1，报 m 的人出列(将其删除)，从他在顺时针方向上的下一个人开始重新从 1 报数……如此下去，直到所有人全部出列为止。试设计一个程序求出出列顺序。要求利用单向循环链表存储结构模拟此过程，按照出列的顺序打印出各人的编号和此人的密码。

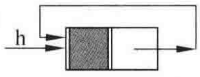
方法 2，报 m 的人出列(将其删除)，将他的密码作为新的 m 值，从他在顺时针方向上的下一个人开始重新从 1 报数……直到所有人全部出列为止。试设计一个程序求出出列顺序。要求利用单向循环链表存储结构模拟此过程，按照出列的顺序打印出各人的编号和此人的密码。

[方法 1 的程序清单]

```
#include <stdio.h>
#include <stdlib.h>
typedef struct node                                  /* 结点的结构定义 */
   {  int  num;                                      /* 编号子域 */
      int  sm;                                       /* 密码子域 */
      struct node *next;                             /* 指针域 */
   }JOS;
```

```
/* 函数声明 */
JOS   *creat( );
void outs(JOS   *h, int m);
/* 主函数 */
main( )
{ int  m;  JOS   *h;
   h=creat();
   printf("\n  enter the begin secret code, m=(m>1)");   scanf("%d",&m);
   outs(h, m);
}  /* main */

/* 生成约瑟夫环 */
JOS   *creat( )
{  int  i=0, mi;
   JOS   *new, *pre, *h;
   h=(JOS *)malloc(sizeof(JOS));                    /* 为头结点申请空间 */
   h->link=h;                                       /* 头结点 h 形成环 */
   pre=h;
   printf("\n 个人密码 =?");  scanf("%d",&mi);
   while( mi !=-111)                                /* 密码为-111,结束循环 */
    { new=(JOS *)malloc(sizeof(JOS));               /* 申请数据元素结点空间 */
     i++;  new->num=i;  new->sm=mi;                 /* 结点送值 */
      new->link=h;
      pre->link=new;
      pre=new;                                      /* 形成循环链表 */
      printf("\n 个人密码 =?");  scanf("%d",&mi);  /* 读入下一个密码 */
} /* while */
   pre->link=h->link; free(h);                      /* 删除头结点 h */
   h=pre->link;  return(h);                         /* 头指针指向第一个数据元素结点 */
}  /* JOS *creat, 该约瑟夫环无附加头结点 */
/* 按 m 被叫出列的顺序,输出结点信息 */
void outs(JOS   *h, int m)
{  int i;   JOS *q=h,  p;
  printf("\n");
  while (q->link!=q)
   { for(i=1;i<m;i++){ p=q; q=q->link;}              /* 报数到第 m 个人 */
     printf("%6d %6d",q->num,q->sm);                /* 输出第 m 个人的编号和密码 */
     p->link=q->link;  free(q);                     /* 第 m 个人出列 */
     q=p->link;
    }
  printf("%6d %6d  \n",q->num,q->sm);               /* 输出最后一个结点的编号值 */
  free(q);
} /* outs */
```



本程序用了不带头结点的循环链表,也可以加上头结点,对于本题来说,有头结点会