（C语言实现）

杨 峰 编著

5.5小时教学视频、86个趣味算法题、61个算法面试题，一学就会

帮您开阔眼界，培养编程兴趣，提高编程能力，增强求职竞争力

清华大学出版社

（C语言实现）

杨 峰 编著

清华大学出版社
北 京

内 容 简 介

本书理论与实践相结合，旨在帮助读者理解算法，并提高 C 语言编程能力，培养读者的编程兴趣，并巩固已有的 C 语言知识。全书分为 2 个部分共 10 章，内容涵盖了编程必备的基础知识（如数据结构、常用算法等），编程实例介绍，常见算法和数据结构面试题等。本书最大的特色在于实例丰富，题材新颖有趣，实用性强，理论寓于实践之中。通过本书的学习，可以使读者开阔眼界，提高编程的兴趣，提高读者的编程能力和应试能力。

本书附带 1 张光盘，内容为本书源代码和作者为本书录制的 5.5 小时多媒体教学视频。

本书可作为算法入门人员的教程，也可以作为学习过 C 语言程序设计的人士继续深造的理想读物，也可作为具有一定经验的程序设计人员巩固和提高编程水平，查阅相关算法实现和数据结构知识的参考资料，同时也为那些准备参加与算法和数据结构相关的面试的读者提供一些有益的帮助。

图书在版编目（CIP）数据

妙趣横生的算法（C 语言实现）/ 杨峰编著. —北京：清华大学出版社，2010.4
ISBN 978-7-302-21601-8

Ⅰ. ①妙…　Ⅱ. ①杨…　Ⅲ. ①C 语言－程序设计　Ⅳ. ①TP312

中国版本图书馆 CIP 数据核字（2009）第 228162 号

责任编辑：夏兆彦
责任校对：徐俊伟
责任印制：何　芊
出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**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
印 刷 者：北京密云胶印厂
装 订 者：北京市密云县京文制本装订厂
经　　销：全国新华书店
开　　本：185×260　**印　张**：24.5　**字　数**：606 千字
（附光盘 1 张）
版　　次：2010 年 4 月第 1 版　　**印　　次**：2010 年 4 月第1 次印刷
印　　数：1～5000
定　　价：49.00 元

前　言

程序 = 数据结构 + 算法

——著名的计算机科学家沃斯（Nikiklaus Wirth）

自从著名的计算机科学家沃斯将程序设计形象地用上面的公式表示出来后，这条“黄金定律”便成为了人们学习程序设计，进行程序开发的准则。要想成为一名真正专业的程序设计人员，基本的数据结构基础和常用的算法知识是必须掌握的。脱离了这两点，编写出来的程序一定不是健壮的好程序。

然而单纯地掌握了一些数据结构基础和常用的算法知识也是远远不够的。空洞地掌握所谓的数据结构和算法等理论知识只是纸上谈兵，这些知识必须要依托于一门程序设计语言才具有真正的生命力，才能够转化为真实的程序代码，才能真正地解决实际问题。

本书就是将数据结构基础和常用的算法知识与目前广泛应用、最具群众基础的C语言相结合而产生的。本书的写作思想是理论与实践相结合，以实践为核心，以实例为主要内容。

首先，本书总结归纳了数据结构基础、常用的排序查找算法和经典的算法思想，提纲挈领地阐述了核心的理论知识。这样可以使没有系统学习过或者不熟悉数据结构和算法等知识的读者对这部分知识有一个基本的了解，并掌握基本的数据结构知识和常用而经典的算法思想，以便更加深入地学习本书的其他内容。

其次，本书列举了大量的编程实例，这些题目都按照知识体系进行了内容上的划分。本书列举的这些编程实例都是一些比较灵活有趣的题目，有些题目渗透了巧妙的算法思想，有些题目则必须借助特殊的数据结构才能更加容易解答。通过这些题目的训练，可以使读者开阔眼界，启迪思维，提高编程的兴趣。最重要的是能够提高读者算法设计的本领，提高读者灵活应用各种数据结构的本领，提高读者编写程序解决实际问题能力。

本书有何特点

1. 结构清晰，知识全面

本书分为两部分。第1部分是基础知识介绍，主要介绍数据结构的基础知识和一些常用的算法思想。这部分内容为核心的理论知识，可以帮助读者学习和回顾数据结构和算法的知识，使读者在理论水平上有所提高，从而能够更加顺利地深入学习后续内容。第2部分主要是编程实例的介绍，通过一些非常有趣的编程实例使读者开阔眼界，发散思维，提高算法设计本领，提高灵活应用各种数据结构的本领，提高读者编写程序解决实际问题能力。

2. 实例丰富，讲解到位

本书的写作思想就是以实践为核心，以编程实例为主要内容。因此本书中包含了大量的编程实例，并都附有详细的分析和解答。作者认为讲解到位是本书与同类书籍相比的一大特点。本书尽量做到深入浅出，多用简单的语句配以图示来讲解比较复杂的问题。而且尽量做到讲解透彻明白，不敷衍读者。

3. 题材新型，趣味性强

兴趣是最好的老师。本书在编写过程中始终贯穿这一思想。因此本书中的题目设置尽量做到既有练习意义，又富有趣味性。特别是在本书的第 2 部分中，列举了大量的兼顾难度和趣味性的经典题目，例如魔幻方阵、汉诺塔、魔王语言翻译、约瑟夫环、马踏棋盘、巧算 24、八皇后问题等。这样使读者对所谓的难题也不再那么畏惧，而是更加愿意面对它。

4. 重点突出，实用性强

本书的写作意图是通过讲解大量生动有趣的实例，培养读者的编程能力、算法设计思想和对数据结构的灵活运用。归根到底就是通过程序设计解决实际问题的能力。因此本书中的所有题目都不只是给出答案而已，而是从算法思想的层面来剖析。涉及到复杂数据结构的内容，还通过图示的方法形象地加以说明。特别值得一提的是，本书的最后一章为算法设计与数据结构面试题精粹，这部分内容从实战和应试的角度出发，旨在巩固读者的知识水平和提高读者的应试能力，同时使得本书更具实用价值。

5. 配多媒体语音视频讲解

为了方便读者理解本书中的一些重点内容，作者专门为本书录制了 5.5 小时媒体教学视频，收录于本书配书光盘中。另外，本书涉及的源代码也收录于本书配书光盘中。

本书内容及知识体系

全书包含两个部分，共 10 章。第 1 部分为基础篇；第 2 部分为编程实例解析。

第 1 部分包括第 1～3 章，主要介绍一些必备的基础知识，包括数据结构基础知识、常用的查找和排序算法、常用的算法思想。

第 2 部分包括第 4～10 章，主要介绍一些基于 C 语言的编程实例。包括编程的基本功、数学趣题、数据结构题、数值计算题、综合题、算法设计与数据结构面试题等。这些题目内涵丰富，兼顾趣味性，从不同侧面体现出对数据结构知识和算法设计思想的灵活运用，相信会对读者有一定帮助。

本书读者对象

本书适合对算法设计有兴趣的入门人员阅读，也可作为学习过 C 语言程序设计的人士继续深造的理想读物。另外，本书可以作为具有一定经验的程序设计人员巩固和提高编程水平，查阅相关算法实现和数据结构知识的参考资料，同时也为那些准备参加 C 语言面试

的读者提供一些帮助。

本书作者及编委会成员

本书主要由杨峰编写。其他参与编写和资料整理的人员有陈世琼、陈欣、陈智敏、董加强、范礼、郭秋滟、郝红英、蒋春蕾、黎华、刘建准、刘霄、刘亚军、刘仲义、柳刚、罗永峰、马奎林、马味、欧阳昉、蒲军、齐凤莲、王海涛、魏来科、伍生全、谢平、徐学英、杨艳、余月、岳富军、张健和张娜。在此一并表示感谢。

本书编委会成员有欧振旭、陈杰、陈冠军、项宇峰、张帆、陈刚、程彩红、毛红娟、聂庆亮、王志娟、武文娟、颜盟盟、姚志娟、尹继平、张昆、张薛。

编　者

目　录

第 1 部分　基础篇

第 2 部分 编程实例解析

第 1 部分　基础篇

第 1 章　数据结构基础

要想成为一名真正的程序员，数据结构是必备的基础知识。只有学过数据结构，才能真正有效规范地组织程序中的数据。而在实际编程中，有些问题必须通过特定的数据结构才能更方便地解决。因此数据结构是每一个搞计算机的人都应当必须掌握的知识。

要想全面而系统地学习数据结构的知识，这里的介绍显然是不充分的，建议寻找专门介绍数据结构的书籍学习。如果你只想掌握一般层次的知识，或是已经学过数据结构，只是为了深入学习本书后续的内容而进行回顾和复习，那么本章的介绍是足够的。

1.1　什么是数据结构

数据结构就是指计算机内部数据的组织形式和存储方法。数组就是一种简单而典型的线性数据结构类型。本章中将更加具体地介绍一些常用的数据结构，主要包括：线性结构、树、图。

线性结构是最常用，也是最简单的一种数据结构。所谓线性结构，就是指由 n 个数据元素构成的有限序列。直观地讲，它就是一张有限的一维数表。数组就是一种最为简单的线性结构表示。更加具体地说，线性结构主要包括：顺序表、链表、栈、队列等基本形式。其中顺序表和链表是从存储形式上区分的，而栈和队列是从逻辑功能上区分的。也就是说，顺序表和链表是线性数据结构的基础，队列和栈是基于顺序表和链表的，它们由顺序表或链表构成。

有时仅仅用线性结构存储管理数据是难以胜任的，一些数据之间存在着“一对多”的关系，这就构成了所谓的树形结构，简称树结构。例如人工智能领域常用的“博弈树”，数据挖掘和商业智能中使用的“决策树”，多媒体技术中的“哈夫曼树”等都是应用树结构存储管理数据的实例。

还有一种常用的数据结构叫做图状结构，简称图结构。图结构中数据元素之间存在着“多对多”的关系，因此图结构与树结构相比，其线性结构要复杂得多。在处理一些复杂的问题中，图结构往往能派上用场。例如：人工智能领域中的神经网络系统、贝叶斯网络等都是应用图结构存储管理数据的实例。

1.2　顺　序　表

在计算机内部存储一张线性表（线性结构的数表），最为方便简单的就是用一组连续地址的内存单元来存储整张线性表。这种存储结构称为顺序存储结构，这种存储结构下的

线性表就叫做顺序表。如图 1-1 所示，就是顺序表的示意。

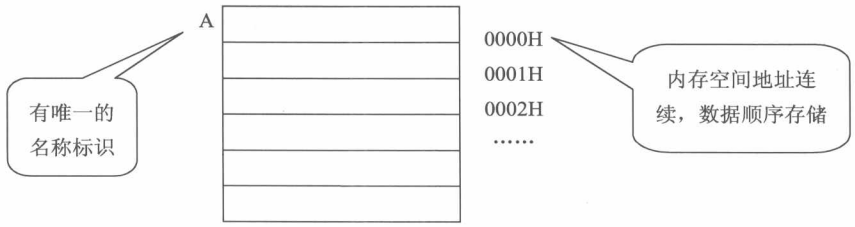

图 1-1　顺序表的示意

从上图中可以看出，一张顺序表包括以下特征。

- 有一个唯一的表名来标识该顺序表。
- 内存单元连续存储，也就是说，一张顺序表要占据一块连续的内存空间。
- 数据顺序存放，元素之间有先后关系。

因为数组满足上述特征，所以一个数组本身就是一张顺序表。

下面介绍如何定义顺序表以及对顺序表的几种操作。

1.2.1　顺序表的定义

定义一张顺序表也就是在内存中开辟一段连续的存储空间，并给它一个名字进行标识。只有定义了一个顺序表，才能利用该顺序表存放数据元素，也才能对该顺序表进行各种操作。

有两种定义顺序表的方法：一是静态地定义一张顺序表；二是动态地生成一张顺序表。

静态地定义一张顺序表的方法与定义一个数组的方法类似。可以描述如下：

```
#define MaxSize 100
ElemType  Sqlist[MaxSize];
int len;
```

为了更完整、更精确地描述一张顺序表，可以把顺序表定义成上面的代码形式。其中 ElemType 是指定的顺序表的类型，这里只是一个抽象的描述，要根据实际情况决定 ElemType 的内容。ElemType 可以是 int、char 等基本类型，也可以是其他构造类型（结构体类型等）。len 为顺序表的长度，定义这个变量目的是更加方便对顺序表进行操作。

这里要注意，上述这种定义方法，包括今后的其他数据类型的定义，都只是一种更完整、更精确的描述。读者不应刻板地机械模仿。例如这样定义：

```
int a[1000];
```

是定义一个数组，但同时也是静态定义一个顺序表。不一定非要预定义 MaxSize 和定义变量 len。因此读者学习时应灵活掌握。

动态地生成一张顺序表的方法可描述如下：

```
#define MaxSize 100
typedef struct{
ElemType *elem;
int length;
```

```
int listsize;
} Sqlist;
void initSqlist(Sqlist *L){ /*初始化一个顺序表*/
    L->elem=(int *)malloc(MaxSize*sizeof(ElemType));
    if(!L->elem) exit(0);
    L->length=0;
    L->listsize= MaxSize;
}
```

动态生成一张顺序表可分为以下两步。

（1）定义一个类型 Sqlist，它是一个结构体，其成员包括：指向顺序表的首地址，顺序表中表的长度（表中元素个数），顺序表的存储空间容量（占据内存大小，以 sizeof(ElemType)为单位，由 MaxSize 规定）。

（2）通过调用一个函数 initSqlist 实现动态生成一张顺序表。函数 initSqlist 的参数是一个 Sqlist 类型的指针变量，因此可以在函数 initSqlist 中直接对顺序表进行操作。

函数 initSqlist 的作用是动态初始化一个顺序表，其操作步骤如下。

（1）调用了 C 标准库函数 malloc 动态地分配一段长度为 MaxSize*sizeof(ElemType)的内存空间，并将该段空间的首地址赋值给变量 L 的 elem 成员 L->elem。也就是说，L->elem 指向顺序表的首单元。

（2）将 L->length 置为 0，表明此时刚刚生成一张空的顺序表，表内尚无内容，将 L->listsize 置为 MaxSize，表明该顺序表占据内存空间大小为 MaxSize（以 sizeof(ElemType)为单位）。

这样 Sqlist 类型的变量 L 就代表了一张顺序表。其中，L->elem 指向该顺序表的表头地址，L->length 为该顺序表的长度，L->listsize 为该顺序表的容量。通过变量 L 可以灵活地操纵该顺序表，因此 L 就代表了一张顺序表。

至此我们知道了如何静态地定义一张顺序表以及如何动态生成一张顺序表。

1.2.2　向顺序表中插入元素

下面介绍如何在长度为 n 的顺序表中的第 i 个位置插入新元素 item。

所谓在长度为 n 的顺序表中的第 i 个位置插入新元素，是指在顺序表第 i–1 个数据元素和第 i 个数据元素之间插入一个新元素 item。例如顺序表为：

$$A(a_1, a_2, \cdots a_{i-1}, a_i, \cdots a_n)$$

在第 i 个位置插入新元素 item 后，该顺序表变为：

$$A(a_1, a_2, \cdots a_{i-1}, \text{item}, a_i, \cdots a_n)$$

而此时顺序表 A 的长度由 n 变为 n+1。

下面给出向静态顺序表中第 i 个位置插入元素 item 和向动态生成的顺序表中第 i 个位置插入元素 item 的代码描述。

按照 1.2.1 节中介绍的方法创建一个静态的顺序表 Sqlist[MaxSize]，那么向该静态顺序表中第 i 个位置插入元素 item 的代码描述如下：

```
void InserElem(ElemType Sqlist[],int &n,int i, ElemType item){
    /*向顺序表 Sqlist 中第 i 个位置插入元素 item，该顺序表原长度为 n*/
    int t;
```

```
    if(n==MaxSize||i<1||i>n+1)
        exit(0);                          /*非法插入*/
    for(t=n-1;t>=i-1;t--)
        Sqlist[t+1]=Sqlist[t];            /*将 i-1 以后的元素顺序后移一个元素的位置*/
    Sqlist[i-1]=item;                     /*在第 i 个位置上插入元素 item*/
    n=n+1;                                /*表长加 1*/
}
```

函数 InserElem()的作用是在顺序表 Sqlist 中第 i 个位置上插入元素 item，并将顺序表长度加 1 。其实现过程如下。

（1）判断插入元素的位置是否合法。一个长度为 n 的顺序表的可能插入元素的位置是 1～n+1，因此如果 i<1 或者 i>n+1，或者表已满、n==MaxSize（因为表的内存大小固定不变）的插入都是非法的。

（2）将顺序表的 i–1 以后的元素顺序后移一个元素的位置，即：将顺序表从第 i 个元素到第 n 个元素顺序后移一个元素的位置。

（3）在表的第 i 个位置（下标为 i–1）上插入元素 item，并将表长加 1。

按照 1.2.1 节中介绍的方法创建一个动态的顺序表 L，那么向该动态生成的顺序表中第 i 个位置插入元素 item 的代码描述如下：

```
void InsertElem(Sqlist *L,int i,ElemType item){
    /*向顺序表 L 中第 i 个位置上插入元素 item*/
    ElemType  *base,* insertPtr,*p;
    if(i<1||i>L->length+1) exit(0); /*非法插入*/
    if(L->length>=L->listsize)
    {
        base=(ElemType*)realloc(L->elem,(L->listsize+10)*sizeof(ElemType));
        /*重新追加空间*/
        L->elem=base;                     /*更新内存基地址*/
    L->listsize=L->listsize+100;          /*存储空间增大 100 单元*/
    }
    insertPtr=&(L->elem[i-1]);            /*insertPtr 为插入位置*/
    for(p=&(L->elem[L->length-1]);p>= insertPtr;p--)
          *(p+1)=*p;                      /*将 i-1 以后的元素顺序后移一个元素的位置*/
    * insertPtr=item;                     /*在第 i 个位置上插入元素 item */
    L->length++;                          /*表长加 1*/
}
```

上述算法与“向静态顺序表中第 i 个位置插入元素 item”的算法类似，都是利用将 i–1 以后的元素顺序后移一个元素的位置，然后再向第 i 个位置上插入元素 item 的方法实现。不同之处在于向静态顺序表插入元素时，由于表的内存大小固定不变，所以只能在 MaxSize 规定的范围之内顺序插入元素。而向动态生成的顺序表中第 i 个位置插入元素 item 时，由于顺序表是建立在动态存储区的，因此可以随时扩充。故当向表尾插入元素时，如果顺序表已满（L->len>=L->listsize），可以追加一段内存空间，再并入原顺序表中。这就是动态顺序表的优势所在。

1.2.3　从顺序表中删除元素

下面介绍如何删除长度为 n 的顺序表中的第 i 个位置的元素。

所谓删除长度为 n 的顺序表中的第 i 个位置的元素，就是指将顺序表第 i 个位置上的元素去掉。例如顺序表为：

$$A(a_1, a_2, \cdots a_{i-1}, a_i, a_{i+1} \cdots a_n)$$

删除第 i 个位置的元素后，该顺序表变为：

$$A(a_1, a_2, \cdots a_{i-1}, a_{i+1} \cdots a_n)$$

而此时顺序表 A 的长度由 n 变为 $n-1$。

下面给出从静态顺序表中删除第 i 个位置元素和从动态生成的顺序表中删除第 i 个位置元素的代码描述。

按照 1.2.1 节中介绍的方法创建一个静态的顺序表 Sqlist[MaxSize]，那么从该静态顺序表中删除第 i 个位置元素的代码可描述如下：

```
void DelElem(ElemType Sqlist[],int &n,int i){
    int j;
    if(i<1||i>n)
        exit(0)                          /*非法删除*/
    for(j=i;j<n;j++)
        Sqlist[j-1]=Sqlist[j];           /*将第 i 位置以后的元素依次前移*/
    n--;                                 /*表长减 1*/
}
```

函数 DelElem()的作用是从顺序表 Sqlist 中删除第 i 个位置的元素，并将表的长度值减 1。其实现过程如下。

（1）判断要删除的元素是否合法。对于一个长度为 n 的顺序表，删除元素的合法位置是 1～n，因此如果 $i<1$ 或者 $i>n$ 都是不合法的。

（2）将顺序表的第 i 位置以后的元素依次前移，这样会将第 i 个元素覆盖，也就起到删除第 i 个位置元素的作用。

（3）最后将表长减 1。

如果按照 1.2.1 节中介绍的方法创建一个动态的顺序表 L，那么从该动态生成的顺序表中删除第 i 个位置元素的代码描述如下。

```
void DelElem(Sqlist *L,int i) {
    /*从顺序表 L 中删除第 i 个元素*/
    ElemType *delItem, *q;
    if(i<1||i>L->len) exit(0);           /*非法删除*/
    delItem=&(L->elem[i-1]);             /*delItem 指向第 i 个元素*/
    q=L->elem+L->length-1;               /*q 指向表尾*/
    for(++delItem; delItem<=q;++ delItem)*( delItem-1)=* delItem;
                                         /*将第 i 位置以后的元素依次前移*/
    L->length--;                         /*表长减 1*/
}
```

从动态生成的顺序表中删除第 i 个位置元素的方法与从静态顺序表中删除第 i 个位置元素的方法本质上是一样的，都是将第 i 个位置以后的元素依次前移，从而覆盖掉第 i 个元素。

以上介绍了顺序表的定义方法、顺序表的元素插入、删除等操作。顺序表是最简单的一种线性存储结构，它的优点是：构造简单，操作方便，且通过顺序表的首地址（或数组名）可直接对表进行随机存取，从而存取速度快，系统开销小。同时它也存在着缺点：有