



（C语言实现）

第2版

杨 峰 编著

算法入门经典畅销书全新升级

5.5小时教学视频、114个经典算法题、61个算法面试题和4个ACM竞赛题
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

生横趣妙的算法

（C语言实现） 第2版

杨 峰 编著

清华大学出版社
北 京

内 容 简 介

本书是深受广大读者好评的《妙趣横生的算法（C 语言实现）》一书的全新升级版。本书在第 1 版的基础上对原书内容做了大量的调整和补充，并将书中的实例代码在 Visual Studio 2010 环境下重新编译通过，以适应当前技术的发展和阅读需求。本书内容涵盖了算法入门的必备基础知识和大量的趣味算法题、面试题和 ACM 竞赛题等。通过学习本书内容，可以开阅读者的视野，帮助读者理解算法，提高编程兴趣和能力，并提高 C 语言编程能力，还可以让读者了解 IT 面试中的常见算法题及编程竞赛中的相关知识。另外，本书提供了 5.5 小时配套教学视频和实例源代码，以提高读者的学习效率。

本书共 11 章，分 2 篇。第 1 篇为算法基础，包括数据结构基础知识、常用的查找与排序方法、常用的算法思想等。第 2 篇为常用算法实例解析，包括 20 个编程基本功能题、39 个数学题、14 个数据结构题、7 个数值计算题、6 个综合题、61 个算法设计与数据结构面试题、4 个 ACM 程序设计竞赛题目。

本书适合学习算法的人员作为入门教程，也适合有 C 语言基础的人作为编程实践读物，还可作为有编程经验的程序员查阅相关算法实现和数据结构知识的参考书。另外，本书也可以为那些准备参加 IT 面试和信息学竞赛的读者提供有益的帮助。

本书封面贴有清华大学出版社防伪标签，无标签者不得销售。

版权所有，侵权必究。侵权举报电话：010-62782989 13701121933

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. — 2 版. — 北京：清华大学出版社，2015
ISBN 978-7-302-38947-7

I. ①妙… II. ①杨… III. ①计算机算法②C 语言—程序设计 IV. ①TP301.6②TP312

中国版本图书馆 CIP 数据核字（2015）第 005653 号

责任编辑：杨如林

封面设计：欧振旭

责任校对：胡伟民

责任印制：王静怡

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

印 刷 者：清华大学印刷厂

装 订 者：三河市漂源装订厂

经 销：全国新华书店

开 本：185mm×260mm 印 张：26.75 字 数：670 千字

版 次：2010 年 4 月第 1 版 2015 年 3 月第 2 版 印 次：2015 年 3 月第 1 次印刷

印 数：1~4000

定 价：59.80 元

产品编号：061184-01

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的能力。

关于本书第 2 版

本书第 1 版出版后广受读者好评，并且多次加印，被大量的算法入门人员和爱好者及一些参加算法竞赛的读者作为参考读物。但是随着技术的发展，第 1 版图书已经不能完全适应当前的阅读需求。例如，很多开发人员已经将 C 语言的编程环境由以前流行的 Turbo C 迁移到了当前流行的 Visual Studio 平台上，加之本书第 1 版图书在解析上还有改进空间。基于这些原因，我们对本书第 1 版进行了系统的改版，以方便读者更好地学习。相比第 1 版图书，本书第 2 版内容上的主要变化体现在以下几个方面：

- ❑ 书中涉及的实例代码在 Visual Studio 2010 环境下重新编译通过；
- ❑ 对一些叙述不清或不够通顺的语句进行了修改；

- ❑ 对第1版图书中存在的一些疏漏进行了修订;
- ❑ 给实例源代码增加了更加详细的注释;
- ❑ 给第1~3章中各增加了一节新内容,作为课后练习,并给出了参考答案;
- ❑ 对第2章中的快速排序和希尔排序等内容进行了较大幅度的修改,同时还新增加了堆排序的介绍及各种排序性能的比较;
- ❑ 新增了一章关于ACM程序设计竞赛的相关内容。

本书有何特点

1. 结构清晰, 知识全面

本书分为2篇。第1篇是基础知识介绍,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篇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的能力。

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篇中,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. 视频教学, 高效直观

本书中的重点内容和实例提供了配套教学视频辅助讲解。读者可以先阅读书中的内容讲解,然后再结合教学视频进行学习,可以获得更加高效而直观的学习效果。

本书内容及知识体系

本书共 11 章，分为 2 篇，主要内容介绍如下。

第1篇 算法基础（第1~3章）

本篇主要介绍一些必备的算法基础知识，包括数据结构基础知识、常用的查找和排序方法、常用的算法思想。

第2篇 常用算法实例解析（第4~11章）

本篇主要介绍一些基于 C 语言的算法编程实例。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、ACM 程序设计竞赛试题等。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

- ☐ 对算法设计有兴趣的入门人员；
- ☐ 有一定编程基础的算法爱好者；
- ☐ 需要提高 C 语言编程水平的人员；
- ☐ 参加 IT 企业面试的人员；
- ☐ 信息学竞赛的参赛人员；
- ☐ 各种程序设计选拔赛的参赛人员；
- ☐ 大中专院校的学生。

本书配套资源获取方式

本书提供了配套的教学视频、源代码和教学 PPT，这些资源需要读者自行下载。读者可以在清华大学出版社的网站上（www.tup.com.cn）搜索到本书页面，然后按照提示下载，也可以通过本书服务网站上（www.wanjuanchina.net）的光盘下载版块下载。

本书作者

本书由杨峰主笔编写。其他参与编写的人员有韩先锋、何艳芬、李荣亮、刘德环、孙姗姗、王晓燕、杨平、杨艳艳、袁玉健、张锐、张翔、陈明、邓睿、巩民顺、吉燕、水淼、宗志勇、安静、曹方、曾苗苗、陈超。

如果您在阅读本书的过程中有任何疑问，请发 E-mail 到 bookservice2008@163.com 以获得帮助。

编者

目 录

第 1 篇 算法基础

第 1 章 数据结构基础	2
1.1 什么是数据结构	2
1.2 顺序表	2
1.2.1 顺序表的定义	3
1.2.2 向顺序表中插入元素	4
1.2.3 从顺序表中删除元素	6
1.2.4 实例与分析	7
1.3 链表	11
1.3.1 创建一个链表	12
1.3.2 向链表中插入结点	13
1.3.3 从链表中删除结点	14
1.3.4 销毁一个链表	15
1.3.5 实例与分析	16
1.4 栈	18
1.4.1 栈的定义	18
1.4.2 创建一个栈	19
1.4.3 入栈操作	20
1.4.4 出栈操作	21
1.4.5 栈的其他操作	22
1.4.6 实例与分析	23
1.5 队列	25
1.5.1 队列的定义	25
1.5.2 创建一个队列	26
1.5.3 入队列操作	27
1.5.4 出队列操作	28
1.5.5 销毁一个队列	29
1.5.6 循环队列的概念	29
1.5.7 循环队列的实现	30
1.5.8 实例与分析	32
1.6 树结构	34
1.6.1 树的概念	34
1.6.2 树结构的计算机存储形式	35

1.6.3	二叉树的定义	35
1.6.4	二叉树的遍历	36
1.6.5	创建二叉树	37
1.6.6	实例与分析	38
1.7	图结构	40
1.7.1	图的概念	40
1.7.2	图的存储形式	41
1.7.3	邻接表的定义	42
1.7.4	图的创建	43
1.7.5	图的遍历 (1) —— 深度优先搜索	45
1.7.6	图的遍历 (2) —— 广度优先搜索	47
1.7.7	实例与分析	49
1.8	章后习题	52
第 2 章	常用的查找与排序方法	55
2.1	顺序查找	55
2.2	折半查找	58
2.3	排序的概述	61
2.4	直接插入排序	62
2.5	选择排序	64
2.6	冒泡排序	67
2.7	希尔排序	69
2.8	快速排序	72
2.9	堆排序	76
2.10	各种排序算法性能比较	82
2.11	章后习题	83
第 3 章	常用的算法思想	86
3.1	什么是算法	86
3.2	算法的分类表示及测评	87
3.2.1	算法的分类	87
3.2.2	算法的表示	87
3.2.3	算法性能的测评	89
3.3	穷举法思想	89
3.3.1	基本概念	89
3.3.2	寻找给定区间的素数	90
3.3.3	TOM 的借书方案	91
3.4	递归与分治思想	92
3.4.1	基本概念	92
3.4.2	计算整数的划分数	93
3.4.3	递归的折半查找算法	96
3.5	贪心算法思想	98
3.5.1	基本概念	98
3.5.2	最优装船问题	99
3.6	回溯法	102

3.6.1 基本概念	102
3.6.2 四皇后问题求解	104
3.7 数值概率算法	108
3.7.1 基本概念	108
3.7.2 计算定积分	108
3.8 章后习题	110

第 2 篇 常用算法实例解析

第 4 章 编程基本功	114
4.1 字符类型统计器	114
4.2 计算字符的 ASCII 码	115
4.3 嵌套 if-else 语句的妙用	116
4.4 基于 switch 语句的译码器	118
4.5 判断闰年	119
4.6 指针变量作参数	120
4.7 矩阵的转置运算	121
4.8 矩阵的乘法运算	124
4.9 巧用位运算	125
4.10 文件的读写	126
4.11 计算文件的大小	128
4.12 记录程序的运行时间	129
4.13 十进制/二进制转化器	130
4.14 打印特殊图案	131
4.15 打印杨辉三角	133
4.16 复杂级数的前 n 项和	135
4.17 寻找矩阵中的“鞍点”	137
4.18 n 阶勒让德多项式求解	138
4.19 递归反向输出字符串	140
4.20 一年中的第几天	141
第 5 章 数学趣题 (一)	143
5.1 舍罕王的失算	143
5.2 求两个数的最大公约数和最小公倍数	144
5.3 歌德巴赫猜想的近似证明	145
5.4 三色球问题	148
5.5 百钱买百鸡问题	150
5.6 判断回文数字	152
5.7 填数字游戏求解	153
5.8 新郎和新娘	155
5.9 爱因斯坦的阶梯问题	157
5.10 寻找水仙花数	159
5.11 猴子吃桃问题	160

5.12	兔子产仔问题	161
5.13	分解质因数	162
5.14	常胜将军	165
5.15	求 π 的近似值	166
5.16	魔幻方阵	169
5.17	移数字游戏	172
5.18	数字的全排列	174
5.19	完全数	176
5.20	亲密数	178
5.21	数字翻译器	180
5.22	递归实现数制转换	183
5.23	谁在说谎	185
第 6 章	数学趣题（二）	187
6.1	连续整数固定和问题	187
6.2	表示成两个数的平方和	189
6.3	具有特殊性质的数	191
6.4	验证角谷猜想	192
6.5	验证四方定理	194
6.6	递归法寻找最小值	197
6.7	寻找同构数	199
6.8	验证尼科彻斯定理	201
6.9	三重回文数字	203
6.10	马克思手稿中的数学题	205
6.11	渔夫捕鱼问题	206
6.12	寻找假币	208
6.13	计算组合数	212
6.14	递归法求幂	213
6.15	汉诺塔	215
6.16	选美比赛	217
第 7 章	数据结构趣题	222
7.1	顺序表的就地逆置	222
7.2	动态数列排序	224
7.3	在原表空间进行链表的归并	227
7.4	约瑟夫环	232
7.5	二进制/八进制转换器	236
7.6	回文字符串的判定	241
7.7	括号匹配	245
7.8	魔王语言翻译	249
7.9	动态双向链表的应用	253
7.10	判断完全二叉树	259
7.11	动画模拟创建二叉树	263
7.12	打印符号三角形	267
7.13	递归函数的非递归求解	271

7.14 任意长度整数加法	274
第 8 章 数值计算问题	282
8.1 递推化梯形法求解定积分	282
8.2 求解低阶定积分	285
8.3 迭代法开平方运算	288
8.4 牛顿法解方程	291
8.5 欧拉方法求解微分方程	293
8.6 改进的欧拉方法求解微分方程	295
8.7 雅可比迭代公式求解线性方程组	298
第 9 章 综合题	302
9.1 破碎的砝码	302
9.2 计算 24 的问题	305
9.3 马踏棋盘	311
9.4 0-1 背包问题	317
9.5 八皇后问题求解	323
9.6 简易文件加密/解密系统	327
第 10 章 算法设计与数据结构面试题精粹	335
10.1 常见的算法设计题	335
10.2 常见的数据结构题	374
第 11 章 ACM 程序设计竞赛题目选讲	398
11.1 奥斯卡奖章	398
11.2 准备就绪?	403
11.3 兔子	406
11.4 阶乘之和	412

第 1 篇 算法基础

- ▶▶ 第 1 章 数据结构基础
- ▶▶ 第 2 章 常用的查找与排序方法
- ▶▶ 第 3 章 常用的算法思想

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必须掌握的知识。

要想全面而系统地学习数据结构的知识，本章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

数据结构是指计算机内部数据的组织形式和存储方法。我们熟悉的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、树结构、图结构。

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（或者说物理结构上）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，因此图结构比线性结构和树结构都复杂得多。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），最方便简单的方法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。

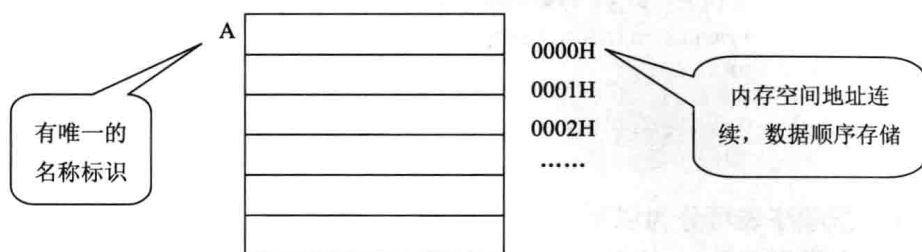


图 1-1 顺序表的示意

从图 1-1 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，例如图 1-1 所示的 A。
- 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类型描述，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，定义这个变量的目的是更加方便地对顺序表进行操作。

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 `elem`，顺序表中表的长度（表中元素个数）`length`，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）`listsize`。

(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。

需要强调一点的是，动态生成顺序表与静态定义一个顺序表的本质区别在于它们所占内存空间的位置不同。对于静态定义一个顺序表，顺序表占用的内存空间开辟在内存的静态区，也就是函数栈上，这个区域的内存空间会随着函数调用的完成而被系统自动回收。而对于动态生成一个顺序表，其内存空间开辟在内存的动态区上，也就是堆内存上，这个区域的内存空间不会被系统自动回收，需要程序主动去释放它。

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：

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

在第 i 个位置插入新元素 `item` 后，该顺序表变为：

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

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 InsertElem(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 *InsertElem()* 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 $1 \sim n+1$ ，因此如果 $i < 1$ 或者 $i > n+1$ ，或者表已满，即 $n == \text{MaxSize}$ （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。因此当向表中插入元素时，如果顺

序表已满 ($L \rightarrow len \geq L \rightarrow listsize$)，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：

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

删除第 i 个位置的元素后，该顺序表变为：

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

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 $1 \sim n$ ，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*/
}
```