计算机实时控制软件设计导论

严隽薇 严隽永

清华大学出版社

计算机实时控制软件设计导论

严隽薇　严隽永　编著

清华大学出版社

内 容 简 介

本书介绍计算机实时控制软件设计的有关问题。内容包括：实时软件系统的设计方法，实时控制和数据采集软件系统实例，实时操作系统的体系结构和功能，实时软件工程方法论简介等。本书既通俗实用，又有利于向软件设计纵深领域提高。

可供计算机应用类专业师生、工程技术人员阅读、参考。

计算机实时控制软件设计导论

严蔚薇　严续永　编著

☆

清华大学出版社出版

北京　清华园

北京海淀昊海印刷厂印刷

新华书店总店科技发行所发行

☆

开本：787×1092 1/16　印张：19.5　字数：482 千字

1990 年 2 月第 1 版　1990 年 2 月第 1 次印刷

印数：0001—8000

ISBN 7-302-00416-1/TP · 139

定价：3.90 元

前　言

为了适应计算机技术在实时控制领域中迅速发展的形势，为了向计算机应用专业的学生以及有关的工程技术人员提供系统学习的参考书，清华大学计算机系人工智能及计算机控制教研组正在陆续编写有关计算机控制系统理论基础、系统硬件设计和工程以及实时软件设计内容的专业书籍。关于计算机控制的理论基础，包括数字信号分析、离散系统的数学描述、稳定性和性能准则以及计算机控制系统的综合与设计方法等两方面的内容，有刘植桢、郭木河、何克忠编著的《计算机控制》一书可供学习；另外，还有孙增圻编著的《计算机控制理论及应用》和何克忠、郝忠恕编著的《计算机控制系统分析与设计》两本书即将出版。前者系统、全面、深入地介绍了离散系统、采样系统和数字系统理论及其在设计中的典型方法；后者阐述计算机控制系统的基本理论，重点介绍了系统的组成、综合设计及实现方法。关于以微型计算机为核心，配置各种外围接口和模拟通道形成计算机实时控制系统方面的内容，有刘植桢、王秀玲、岳震伍编著的《微型机控制系统工程与设计》一书可作参考。前三本书是为计算机控制与应用专业高年级学生与研究生开设“计算机控制”专业课的教材，最后一本则是编著者近几年来在科研、教学中深入研究和应用微型机技术的成果总结。这些书同时也可作为自动控制工程技术人员在利用微型计算机技术完成各种实时控制时的学习参考书。在掌握了一定基础理论和系统硬件设计调试等技术的基础上，为了进行计算机实时系统的软件开发，还必须进一步了解实时控制系统的软件设计方法，当然这首先应包括了解一些软件的基础知识。本书是一本既具有综合性，又有利于向实时软件设计领域开拓的导论性读物。

根据几年来科研和教学工作中的体会，我们在本书中主要编入了以下各项内容：

在第二、三、五章中，以微型机为主体，从常用的数据结构和基本的程序设计方法开始，介绍实时软件系统的基本设计方法；第七章介绍如何为应用系统配置一种专用语言；第八章中给出了一批实时控制和数据采集方面的软件系统实例。以上章节的内容力求通俗实用。同时，为兼顾学科系统性和科学性，也编入了一部分介绍新理论、新方法的内容：第四章对实时操作系统的讨论，从对其体系结构和功能描述与划分来进行，这种讨论方法目前尚少见；第六章初步介绍了实时软件工程方法论的主要内容，并辅以利用该方法设计复杂实时软件的实例。

软件基础较差的读者可先按第一、二、三、五章的顺序阅读本书，然后再读第四、七章。若读者已具有一定的编程能力，则可跳过第三章的内容，直接阅读后续章节。对软件工程方法论有兴趣的读者，可阅读第六章。总之，本书可当作一本综合性的参考读物，而书中的一些软件实例也许对您手头从事的任务会有所帮助。

本书的第一章由严隽永和严隽薇共同编写，第二、三、五、七、八章由严隽薇编写，第四、六章由严隽永编写。全书由刘植桢同志审阅。在本书编写过程中，刘植桢同志

提出了许多宝贵的意见，并进行了许多具体指导和帮助；本书的出版还得到王秀玲同志的热心帮助；杨网、屈玉俊同志分别为第八章和第 5.5 节提供了有用的素材，在此一并致以衷心的感谢。

由于我们水平有限，加上时间又比较仓促，难免存在缺点和错误，敬请有关计算机工作者和广大读者批评指正。

目　录

第一章 绪 论

计算机在社会的各个领域中正在起着日益重要的作用；相应地，计算机技术与作为它的理论基础的计算机科学也得到了极其迅速的发展。几个世纪以前开始的工业革命，人们围绕对力和能的研究与利用，发明了从蒸汽机到电机的一系列力能类型的机器；在这一阶段，理论科学也主要是针对力能过程进行研究。直到第二次世界大战以后，涉及"信息过程"的第二种发展过程逐步占据主导地位，已成为当今社会的一大特征。有人把包括这种过程在内的各种新的技术变革称之为新的技术革命。虽然，在一般的技术过程中，力能过程与信息过程往往相伴而存，然而，计算机的发明，毕竟标志着与力能机不同的新的一类机器的正式问世。这种机器，称为信息机。随着计算机的出现，信息机才作为一大类别而列入人类生产工具的行列。

计算机的应用领域日益广泛。最初，它主要用于科学计算、数据处理，随着应用技术的发展，计算机已大量应用于控制领域。追根溯源，控制乃是力能过程与信息过程相结合的产物。从最简单的继电器、接触器控制到复杂的计算机控制，都是利用信息过程去操纵、支配（或称控制）力能过程。

1.1 计算机与控制

计算机与控制、控制理论（Control Theory）以及控制论（Cybernetics）有着密切的关系。计算机促进了控制学科的巨大发展，并使之发生了质的飞跃。同时，反过来控制学科和技术的每一进步也为计算机的应用开拓了广阔的天地。控制论的奠基者维纳（N.Wiener）也是最早提出现代计算机设计构想的科学家之一。从现代科学的观点来看，两者在理论基础方面有许多共同之处。控制理论，从经典现论到现代控制现论，以及近十年来它与人工智能学科的结合，都无不与计算机息息相关。计算机对控制学科的冲击以及它们的交叉覆盖推动了近十多年来信息、控制等学科领域的巨大进展。计算机能够提供诸如人脑或人的行为这样一些复杂系统的模型，为仿真和综合这类系统创造了条件。计算机使我们能够对复杂庞大的信息进行实时、迅速的加工处理、解算并实施控制，这种对复杂信息处理的快速性，是除计算机以外的任何其他手段无能为力的。

计算机与控制学科的相互结合，体现在各个技术门类和学科分支之中。比如工业控制、系统仿真、环境仿真、信息处理、经济管理、神经与生物控制论、医学控制论、教育系统、指挥控制通信系统、社会系统以及人工智施与机器人等学科及分支中都体理了这种结合。

现代计算机的本质属性之一是内部存放着程序，称为"存入程序的"。这是所谓冯·诺依曼型计算机的基本特点。也就是说，计算机的构成不仅包含着作为物理实体的"硬件"，而且还包含着作为非物理实体的"软件"。软件的雏型（最原始的程序）伴

随着计算机的产生而产生；但是直到五十年代末以后，才真正成为一种系统，成为计算机学科中确定的概念。简言之，计算机的程序系统及其数据系统构成了它的软件。软件的基础是算法，对于计算机所有的程序设计来说，算法的概念总是基本的。一个算法就是一个有穷规则的集合，这些规则规定了解决某一特定类型问题的运算序列。对于某一类问题的解决，只要找到适当的算法，那么总可以设法在计算机上用软件方法来实现。算法的形式化（即数学化）描述，为思维的机械化创造了条件，即为计算机模拟人的思维奠定了基础。对各种工程问题也是如此，只要人们能够用形式化的算法把它们的解法描述出来，那么就可能设法在计算机上实现这些问题的解法。正是因为存在着这种可能性和潜在能力，计算机才能在解决各种复杂问题的过程中大显身手，因而能如此迅速、广泛深入地渗透到各个学科领域中，并取得丰硕的科技成果。如果说各个需要用计算机来解决各种问题的学科的主要任务是寻找这些问题解法的数学模型的话，那么，计算机科学的基本问题便是研究解决各类数学模型的算法。模型不同，算法也不同，甚至差别很大。一切数学分支，无论属于连续数学还是离散数学，只要在实际学科中找到自身的应用，均可用来建立数学模型。

同样道理，控制问题的解决，归结起来，也是要根据计算机控制的理论寻找它们的数学模型，然后是设计控制和数字校正算法，开发软件（当然，在控制工程项目中，还必须做硬件方面即系统的工程设计工作，这里不作讨论）。在“开发软件”这一问题上，常常存在着严重的问题，这就是由于实时系统有其特殊的开发背景（见本章 1.3 节），于是针对具体的实时系统的课题，往往要在裸机上建立本实时系统特有的虚拟机。虚拟机是由设置在用户和硬件之间的软件层形成的[8]。这种软件层每层之间层层相继，每增加一个软件层都使原系统的复杂性增加一层，最终使其属性更紧密地适合设计者的要求。每一层虚拟机组成的软件，自动地将定义在本层的数据和动作翻译成定义在下一层机器上的较简单的数据结构和动作。图 1.1 给出了实时系统课题与软件的关系。从硬件出发，相继的软件层为设计者提供了一个越来越方便的虚拟机（在这里，往往是一个实时操作系统）。这时，用户要做的软件开发工作，就是如何建立课题与虚拟机所要求的数据和动作之间的映象。这个工作也不是很容易的，但是肯定比试

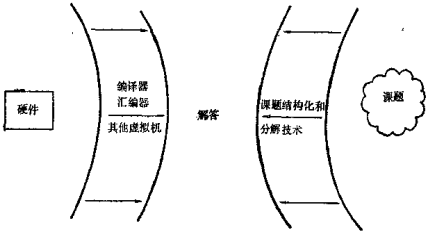

图 1.1 课题对硬件的关系

图把问题直接映射到物理硬件上要简单得多。至此，我们很容易理解：软件的设计与实现，在计算机控制工程中，是极其重要的一环。作为一个控制工程师，不仅需要熟悉有关的硬件，还应掌握一定的软件知识和技能。

1.2 过程控制与实时控制

生产过程的计算机控制，即对生产过程中各种动作流程的控制，一般称为过程控制。这种控制是在对被控制对象和环境进行不断观测的基础上作出的及时而恰当的反应。在控制过程中，计算机扮演着中心的角色。它通过传感器从外部控收有关过程的信息，对这些信息进行加工处理，然后对执行机构发出控制指令。当然，在具体的控制系统中，不一定只有一个计算机，可以有各种各样的配置方案。微处理机的出现及其发展，改变了控制系统的结构模式，使得能建立层次式和分布式控制系统。这也是近年来计算机技术的发展方向之一。

基于计算机的快速性，可以利用它来实现对迅变过程的实时控制。关于实时性，人们往往有不尽相同的理解和解释。一般将联机系统视作实时系统，也有人把人-机交互性的系统称为实时系统。当然，它们都是计算机发展到一定阶段的产物。本书讨论的实时系统是指前者，即它能在某个持续的过程（包括连续的或离散的）中，对应于系统的一组特定的输入数值，在它们尚未发生有意义的变化时，就作出恰当的反应。也就是说，实时系统是处理具有严格时间响应限制的联机外部过程的系统。由外部过程发来的中断信号引起系统动作，系统对此在规定的时间内作出反应。

在分布式和层次式实时控制系统中，各个计算机成员可以按一定的规则相互替代、执行不同的任务。当某些成员发生故障时，系统的配置可以重新调整，这称为系统重构。系统重构与容错是建立可靠的实时系统的两个重要问题。容错是指系统某些成分发生故障时系统仍然具有正常工作的能力。如果系统能够重构，那么就提高了整个系统的容错能力。容错是一个专门问题，它可以有各种不同的实现方法。从元器件配置到模块部件组成，以致到单机系统或多机系统均可以实现一定的容错能力，并不仅限于分布式或层次式结构。

自动化系统按照取得信息和发出控制指令之间的关系可以分为“反馈”控制与“顺馈”控制两类。前者是系统从过程的输出获取信息；后者则是从过程的输入获取信息。另有一些过程控制无需从过程接收信息，仅由控制指令组成的命令程序来控制，这是一种开环控制。

自动化可以理解为“可编程的计算机”与人的合理的结合。人与计算机的相互作用称为交互作用。使用计算机的人应当知道什么时候对计算进行干预，以保证过程的正常运转。在研究与设计计算机实时控制系统时，人-机交互性是必需考虑的重要问题之一。

计算机在自动控制系统中所起的作用，还表现为使系统能适应变化着的外界环境，也就是说，可以实现自适应的控制。在控制学科的文献中，对此已有很多的阐述。从数学上对自适应系统进行研究，并建立它们的模型，这是当前自适应控制系统发展的一个重要途径。数学模型的建立，有助于用计算机精确地实现这种系统。实际上，也只有用

了计算机，自适应控制才能成为现实，并得以发展。

1.3 实时系统的开发背景

在实时环境中运用计算机，其目的是为了实现对外部对象的实时处理与控制。实时系统中软件的设计、研制和实现，直到最后投入运行以前的各个阶段，我们统称为开发（有时把运行阶段直至系统生命期终结的整个过程中的软件维护工作也归属于开发）。实时系统中运行的软件与一般通用计算机中运行的软件在开发背景方面是不相同的。通用计算机的外围设备是通用的、标准的；各种高级语言已发展得相当成熟和标准化，并且与操作系统已有较完备的接口。对于同一系列的计算机，有着各类通用的操作系统；对于同一系列内的各个具体计算机装置，它们基本上是相同的。通用计算机系统上的系统软件是由计算机厂家或软件生产者提供的，它强调通用性和标准化，以适用于广泛的用户，用户不需要自己去开发一个操作系统。然而，对于实时系统，即使计算机硬件系统是相同的，如果具体的应用对象不同，那么，其上的实时操作系统也可能是很不相同的，因此，实时系统上整个运行软件，包括通常意义下的操作系统功能部分与实时应用功能部分，必须一体化地针对特定的应用要求进行设计裁制和研制。另外，实时系统中，计算机是用来代替人和其他机械功能的中心环节；被替代的功能越多，计算机对各种功能的协调作用就越强，因此，实时系统中运行软件的开发涉及到解决各种时间和空间上的配合问题，既要保证实时性，又要求尽可能压缩存储空间的开销。目前有些计算机原始设备制造厂家提供了某种实时操作系统，作为 OEM 产品的组成部分，但用户仍必须根据自己的需要进行选取和裁制，作二次性开发。人们已经认识到对实时操作系统也需要实现一定程度的标准化，但是由于实时系统的千差万别，很难设计一种统一的操作系统而又保证系统高性能和低开销，因而这种努力到目前为止尚未达到令人满意的程度。随着计算机速度的提高、成本的下降，设计一种在一定范围内通用的标准实时操作系统还是可能的。

如上所述，实时系统开发背景的这种针对性和应用对象的这种专用性，是实时系统的软件与通用系统的软件的基本差别。一切工程系统的设计，都必须从应用需求出发，并且考虑到各种可行性条件（包括实现可能性与制约因素），才能制定出设计任务的规格要求。因此，我们在设计一个实时系统时，必须将开发的背景搞得十分清楚。

1.4 实时系统的基本特点

1. 实时性

顾名思义，实时系统的本质属性在于它的实时性。关于实时性的概念，我们在 1.2 节中已概略地说明过，现在进一步作一些讨论。用一个一般概图表示一个系统，如图 1.2 所示。系统具有若干个输入 U（至少一个）和若干个输出 V（至少一个）。对于一组特定的输入值 U_i，系统就产生出一组对应的输出值 V_i。若把前者称为激励（或驱动），后者称为响应（或效果），则对于一组特定的激励，在满足对于系统提出的一

图 1.2 一般实时系统描述

定时间要求和准则下，系统就得出了相应的响应，这种操作就称为实时操作，这种系统便称之为实时系统。例如一个化学工业过程，将测得的某些过程参数输入系统，去激励系统产生相应的响应，后者又去控制执行机构动作，校正过程参数，达到控制与调整的目的，这就是实时操作。如果当响应出现的时候，输入已经远离初始对应的激励状态，以致该响应已失去对系统参数实施控制与调整的意义，那么就不是实时操作。响应对激励的时间滞后，称为响应时间或反应时间。实时性的基本指标是响应时间。对于不同的过程，有不同的响应时间要求。对于有些慢变化过程，具有几分钟甚至更长的响应时间都可以认为是实时的。对于快速过程，其响应时间可能要求达到毫秒、微秒、毫微秒级甚至更短。因此，实时性不能单纯从绝对的响应时间长短上来衡量，应当根据不同的对象，在相对意义上进行评价。

在研究和设计实时系统时，必须紧紧把握住实时性这一基本特点。一般通用操作系统设计时强调达到尽可能大的信息吞吐量，这些系统将能适应各种各样的数据处理要求。而实时操作系统是根据具体应用系统的要求而设计、裁制和开发的，它具有确定的数据处理要求。因此，设计时强调达到尽可能小的任务切换时间，以实现较高的实时响应特性。

2. 一体性

在通用计算机系统中，操作系统是制造厂家提供的，用户在通用操作系统的控制下，利用通用计算机系统提供的手段，开发自己的应用程序。操作系统则是计算机资源的管理者，比如为用户（应用程序）提供 I/O 操作或其他服务。大多数通用操作系统保留着该系统自己运行时的特权状态，并与运行应用程序时的用户状态相区分。因此，对通用计算机系统来说，系统软件与应用软件的界限分明。应用软件往往用高级语言在该系统上由用户进行开发。而在许多实时系统中，这一界限并不十分明显。I/O 操作（如某种执行机构）可能不是标准的，也许要由应用程序来提供。有的小系统并没有单独用硬件实现的特权状态，因此关键的操作系统功能与应用程序的区分要由软件来实现。

因此，我们把实时系统中运行的操作系统与应用程序一体化为运行软件，也就是把它们作为一个整体。在系统总的设计目标指导下将它们统一起来加以考虑、设计与实现。在现代实时系统研制过程中，为了保证系统的高性能和研制本身的高生产率，运行软件可以不必等待系统硬件研制完成以后再在该目的机自身上开发，而是在另外一些作为开发系统的所谓宿主机上开发。后者往往具有丰富而有效的支持软件和开发手段（有时称为开发支持环境）。宿主机可以是与目的机相同的机型，也可以是不同机型，或者是一种功能更强的机型。目的机是将要实现的实时系统的中心环节，它称为“嵌入式”计算机。由于时间和内存空间开销的限制，它本身一般不配置开发手段；除非是大型实

时系统，计算机功能很强，本身也可能具备某种开发能力。在宿主机上配置了通用操作系统（作为开发支持环境的一个重要组成部分），以支持实时操作系统与应用程序的开发。

3. 多任务与并发性

目前，许多大型或较大型通用操作系统都支持多道程序；多道程序技术支持了多用户作业的并发执行。每个作业可以由若干个顺序执行的任务所组成。在一个作业内支持若干个任务并发地执行的技术称多任务技术。通用操作系统一般较少支持这样的多任务功能，而实时系统为了保证实时性往往要求支持多任务操作。实时操作系统往往只提供两种作业：一个是前台或实时作业，另一个是后台或非实时作业。从体系结构的观点看，实时操作系统由于要求多任务操作，比较复杂。在设计系统时，由于实时性的要求，必须注意降低任务切换的时间开销。

并发性分为两种情形。一种是在单处理机系统中，多个任务在宏观上看是并发的，但在微观上看是顺序执行的；另一种是在多处理机系统或分布式系统中，多个任务可以分别在不同的处理机上执行，宏观上看是并发的，微观上看也是并发的。前者称为伪并发性，后者称为真并发性。操作系统设计的复杂性主要来源于系统并发性要求。实时系统中多任务并发性所引起的任务间的同步、互斥、通信以及资源共享与保护等问题比较复杂。这也是目前在文献中较多讨论的一些问题。为了设计良好的实时系统的软件，人们寻求着各种有效的开发工具与手段，其中包括合适的程序设计语言。

4. 环境行为的随机性

从事自动控制学科研究的人们都知道，一切被控过程从本质上讲都是随机的、不确定的。即其并发的外部事件之间往往具有随机的时间关系。这就是说，实时系统需响应的事件是在难以估计的时间以难以预料的顺序出现，并且各种事件有时间上的紧迫性。即使在最“忙”时，也不允许因处理不当而推迟处理时间。为此，系统各部分的处理能力必须按照“忙”时的负荷来计算。在系统设计阶段，其应答时间是按每种处理的类别，以概率与统计的方法来表示的。对于随机同时到达的信息，就要设置优先级，对优先级高的先处理。为提高系统的效率，对于那些随机到达的信号，大多采用中断的方式来处理。因此，中断技术的使用是实时系统的重要特点之一。

第二章　常用的数据结构

数据结构是研究计算机程序的加工对象——数据的学问。它不仅研究数据本身的特性，而且研究数据之间存在的关系和数据的组织。随着计算机应用范围的不断扩大，在应用中，需要处理大量的非数值数据。这就需要研究比单纯数值数据更为复杂的数据及其逻辑结构和物理结构，这对于提高程序的效率及可靠性是极为重要的。实时系统中大量的问题是实时数据处理，不少是符号、表格及图表等。在进行实时软件的设计中，为了编写出有效、清晰、周密的程序，必须有效地组织数据、选取最合适的数据结构，这就应掌握一些基本的数据结构知识，这些是软件设计的基础知识。

一般来说，数据结构包括数据的逻辑结构和数据的物理结构。数据的逻辑结构仅考虑元素之间的逻辑关系，而数据的物理结构，则是指数据元素在计算机存储器中的表示及其配置。后者有时称之为存储结构。我们在本章中将简单扼要地介绍一些线性结构（字符串、栈、队列和数组等），也将涉及一些非线性结构，如树结构等。在讨论这些数据的逻辑结构的同时，也介绍它们相应的存储结构。

2.1　线　性　表

线性表是信息表中最简单的一种形式，但它又是计算机程序中最常用的一种数据结构。线性表是由一组数据元素组成的。数据元素的含义很多，最简单的情况可能是一个字符，也可以是一个数；复杂时可以是一页书，甚至其它更复杂的信息。

例如，英文字母表

（A，B，C，…，Z）

是一个线性表。又如一年十二个月

（一月，二月，…，十二月）

也是一个线性表，表中的数据元素是一个月份的名称。

在稍复杂的线性表中，一个数据元素可以由若干个数据项（item）组成。这时也称数据元素为记录（record）。对于以记录为元素的线性表，也称为文件（file）。例如一个班级学生考试平均成绩的登记表，可以形成一个文件（表 2-1），表中每一个学生的情况就是一个记录，一个记录由姓名、学号、性别、年龄、成绩五个数据项组成。

我们用下列方式来描述线性表：

一个线性表是 $n\geqslant 0$ 个数据元素 a_1，a_2，…，a_n 的有限序列，表中每个数据元素，除了第一个和最后一个外，有且仅有一个直接前趋，有且仅有一个直接后继。线性表写为

$$(a_1,\ a_2,\ \cdots,\ a_i,\ \cdots,a_n) \qquad n\geqslant 0$$

a_i 是属于某个数据对象的元素。从线性表的表示形式，我们可以看到它的结构特性有

表 2-1 以记录为元素的线性表

姓　名	学　号	性　别	年　龄	平均成绩
张建平	830300	男	18	94
王振	830301	男	17	80
曹芙	830302	女	18	92
赵志伟	830303	男	16	73
⋮	⋮	⋮	⋮	⋮

如下两点：（1）线性表是数据元素的一个有限序列。表中数据元素的个数 n 称为表的长度。当 n=0 时为空表。（2）数据元素在线性表中的位置只取决于它们自己的序号，数据元素之间的相对位置是线性的。线性表是一个线性结构。

2.1.1 对线性表的运算

对线性表可能进行如下九种运算：

（1）确定线性表的长度 n；

（2）存取线性表的第 i 个数据元素，检验或改变某个数据项值；

（3）在第 i−1 个和第 i 个数据元素之间，插入一个新的数据元素；

（4）删除第 i 个数据元素；

（5）将两个或两个以上的线性表合并成一个线性表；

（6）重新复制一个线性表；

（8）按某个特定的值查找线性表；

（9）对线性表中的数据元素按某个数据项值递增（或递减）的顺序进行重新排列。

上述各项运算中第（8）、（9）两种我们将在第三章程序设计方法中谈到。第(3)、(4)种是使用较频繁的两种，我们要在本小节介绍。其他运算、或者是较简单，或者使用不很多，我们暂时不作讨论。

2.1.2 线性表的插入和删除

线性表的插入是指在线性表的第 i−1 个元素之后和第 i 个元素之前插入一个新的元素，使长度为 n 的线性表

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

变成长度为 n+1 的线性表

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

其中 a'_i 为新插入的元素。从 a_i 开始分别是原表中的 a_i 至 a_n。即对顺序方式存储的线性表插入时，要将原表中从 a_i 开始到 a_n 的每个元素后移一个元素长度，使表

的长度变为 n+1。

线性表的删除是指删除线性表的第 i 个元素，也就是把长度为 n 的线性表

$$(a_1,a_2,\cdots,a_i,\cdots,a_n)$$

变成长度为 n−1 的线性表

$$(a'_1,a'_2,\cdots,a'_{i-1},a'_i,\cdots,a'_{n-1})$$

原来的 a_i 被删除，所以从 a'_i 开始分别是原来表中 a_{i+1} 至 a_n。即对顺序方式存储的线性表，删除时，要将原表中从 a_{i+1} 开始到 a_n 的每个元素前移一个元素，使表的长度为 n−1。

由此可见，对顺序方式存储的线性表的插入、删除运算一般都要移动表中的一部分元素。我们假定在表的任何位置插入或删除元素是等概率的，那么可以计算出插入元素和删除元素时所需移动元素的期望值(平均次数)分别为 n/2 和 (n−1)/2。[1]可见对顺序方式存储的线性表进行运算，其效率是很低的。我们把顺序方式的存储结构叫作向量。向量在通常的程序语言中也称为一维数组。

若有向量 V，它是长度为 n 的线性表，因此向量 V 的上界是 n。我们用 V[i] 表示向量的第 i 个分量，那么它也是第 i 个元素 a_i 在计算机存储器中的映象。例如对线性表

$$(a_1,a_2,\cdots,a_n)$$

我们按顺序存放元素，每个元素占 l 个单元。假定第一个元素从地址 b 开始存放，那么第 i 个元素的存放地址 V[i]=b+(i−1)*l。其中 b 和 l 是常数，所以下标 i 也可用来代替 a_i 的存储地址。i 可在 1～n 中随机取定。

现在，我们可以说以向量作为线性表的存储结构，其结构简单，且便于随机访问表中的任一元素。由于插入删除运算效率较低，所以，它较适用于表中元素不常变动的线性表。

2.2 栈和队列

栈和队列是线性表的特例，但用途很广泛，有必要专门在这里介绍。

2.2.1 栈

栈是这样一种线性表，它限定所有的插入和删除只在表的一端进行。在表中，允许插入和删除的一端叫做栈顶（top），而表中不允许插入和删除的另一端则叫做栈底（bottom）。若给定栈 $s=(a_1,a_2,\cdots,a_n)$，则称 a_1 是栈底的元素，a_n 是栈顶的元素。因为栈只允许在表的一端进行插入和删除，所以，删除时必须首先是对 a_n，插入时必须接在 a_n 之上，成为 a_{n+1}。对于栈的插入，我们称为进栈，对于它的删除，我们称为退栈。由此看来，这里的栈 s 中的元素以 $a_1,a_2,\cdots,a_n$ 的顺序进栈。而退栈的次序则是 $a_n,a_{n-1},\cdots,a_1$。也就是说，栈的修改是按后进先出的原则进行的。英文为 Last In First Out，简写为 LIFO 表。栈的示意图见 2.1。它象个堆放货物的盒子，

进出只在栈顶进行。

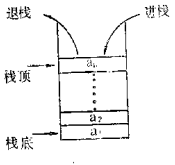

图 2.1 栈的示意图

通常，对栈进行下列几种运算：

（1）PUSH（s，x）：在栈 s 的顶上加入一个新的元素；

（2）POS（s）：在栈 s 中删去栈顶元素；

（3）TOP（s）：读栈 s 中栈顶元素。

由于栈的运算只允许在一端进行，故以向量 s 作为栈的存储结构是合适的。且要设置一个栈顶指针 top，以指示栈顶的位置，另外，栈的容量m决定了向量 s 的上界。在元素尚未进栈时，设 top 为零，s[1] 表示第一个进栈的元素，s[i] 表示第 i 个进栈的元素，s[top] 表示栈顶元素。当top=m 时，表示栈满。栈中元素和栈顶指针之间的关系见图 2.2。很明显，当 top=m 时，作进栈运算，则栈溢出，称之为“上溢”（over-flow）。反之，当 top=0（即栈空）时，作退栈运算，则称之为“下溢”（underflow）。

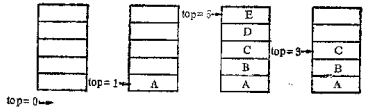

图 2.2 栈顶指针和栈中元素的关系

栈的应用是很广泛的。当然仅把栈看作是一片存储单元的话，起码可以用来存放某些信息。例如，保护现场的一些寄存器、参数等中间信息。然而栈数据结构的特殊性体现为“LIFO”，所以在处理过程的嵌套调用，或者中断发生嵌套时是特别有用的。例如我们的系统有 8 级中断管理（0～7 级），其中 0 级为最高优先级。在程序中 7 级中断首先发生。为了保证中断处理后继续执行中断了的主程序，必须记住断点 r，保存

图 2.3 中断嵌套的过程