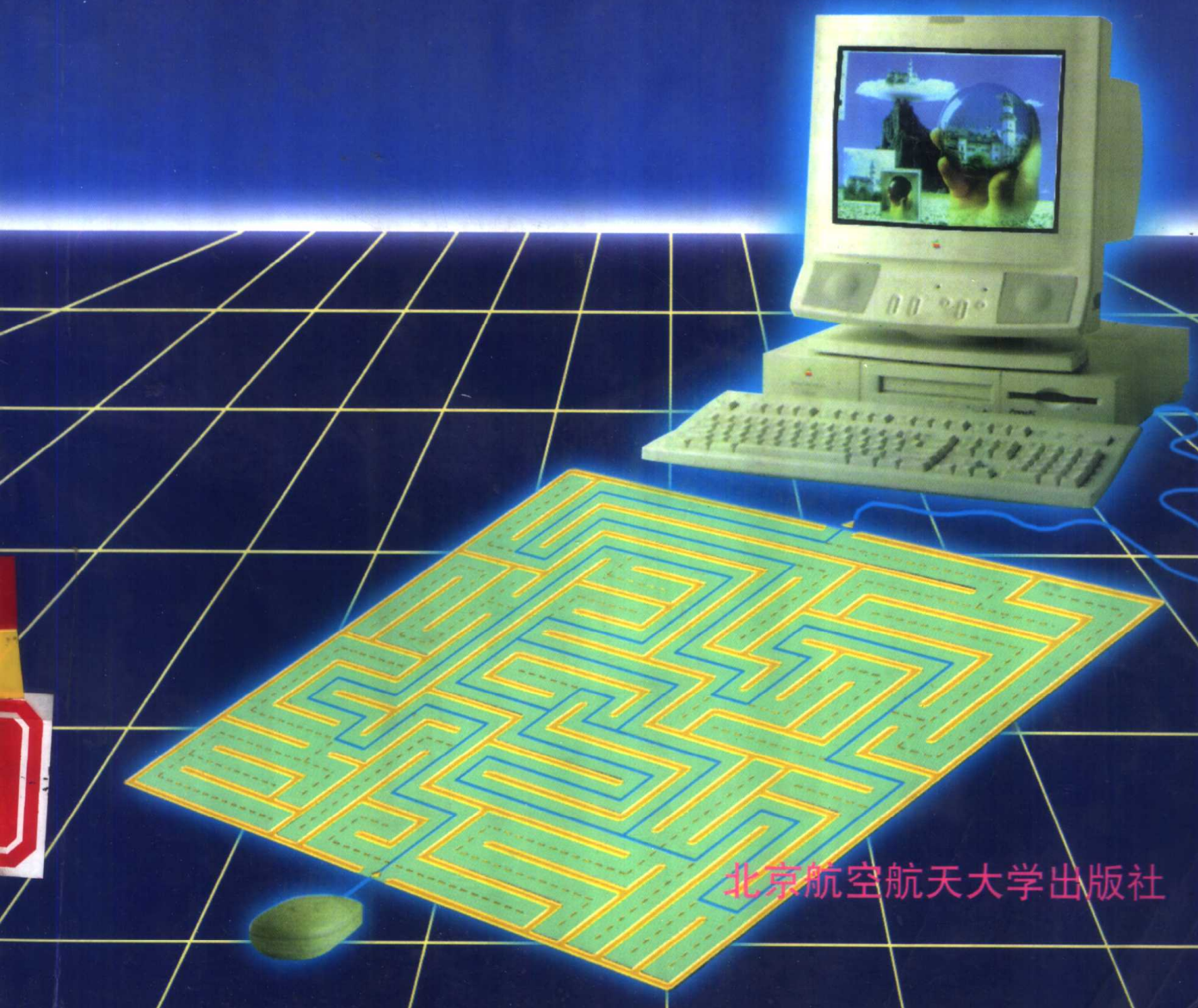
单片机程序设计基础

周航慈　饶运涛



北京航空航天大学出版社

单片机程序设计基础

周航慈　　饶运涛

北京航空航天大学出版社

图书在版编目(CIP)数据

单片机程序设计基础/周航慈等编著. —北京:北京航空航天大学出版社,1997.4
ISBN 7-81012-692-X

Ⅰ.单… Ⅱ.周… Ⅲ.单片微型计算机-程序设计 Ⅳ.TP311

中国版本图书馆 CIP 数据核字(96)第 25111 号

内 容 简 介

本书是《单片机应用程序设计技术》一书的姐妹篇,《单片机应用程序设计技术》以程序设计的经验和技巧为主,而本书则以程序设计的基本算法为主,目的在于全面提高单片机程序设计者的软件素质。

本书的基本内容有:单片机中常用的线性数据结构和相关算法;排序和查找算法;树和图在单片机中的实现;常用的数据处理算法;常用编码方法等。

本书从单片机系统的实际硬件环境出发,用通俗易懂的语言代替枯燥难懂的理论说教,使读者在比较轻松的条件下将单片机程序设计基本算法学到手。为加强学习效果和增加实用价值,本书配有上机学习软盘,在软盘中还提供了最新修订的子程序库,供读者选购。

本书可作为电子技术人员自学单片机程序设计基本算法的教材,也可供高等院校电子技术类专业本科生、专科生作为教学参考书。

●书　　名:单片机程序设计基[illegible]
DANPIANJI CHENGX[illegible]SHUJIICHU
●编 著 者:周航慈　　饶运涛
●责任编辑:韦秋虎
●责任校对:陈　坤
●出 版 者:北京航空航天大学出版社出版发行([illegible]:100083)
北京航空航天大学出版社发行科电话(010)62015720
●印 刷 者:北京市宏文印刷厂
●发　　行:新华书店总店北京发行所
●经　　售:全国各地新华书店
●开　　本:787×1092　1/16
●印　　张:14.75
●字　　数:376 千字
●印　　数:6000 册
●版　　次:1997 年 5 月第 1 版
●印　　次:1997 年 5 月第 1 次印刷
●书　　号:ISBN 7-81012-692-X/TP・237
●定　　价:23.50 元

前　言

单片机在各行各业的应用越来越广，我国从事单片机开发的人员也越来越多。从我国主要的几种电子杂志上可以看出，有关单片机应用的文章也越来越多。

在开发一种含单片机的产品时，有关单片机部分的工作主要是作两方面的设计：硬件设计和软件设计。在硬件设计方面，世界上几家主要的半导体公司（如 INTEL、PHILIPS、MOTOROLA 等）竞相推出各种高性能、低功耗、低成本的单片机和外围芯片，这使我们在进行硬件设计时可以很快地得到最先进的芯片。在这种情况下，硬件设计的外部条件越来越好，集成度越来越高，在实现相同功能的情况下线路越来越简化。在软件设计方面，虽然开发工具和程序设计语言也在不断提高，但技术人员本身的软件素质无疑起决定作用。因此，软件设计水平在单片机产品开发的过程中占有重要的地位，直接影响到产品的水平和竞争能力。不容置疑，在我国各大专院校、科研院所、大中型企业中，有一批单片机专家，他们的软硬件水平均非常高。但另一方面，我国目前绝大多数从事单片机开发的技术人员身在基层，大多数是非计算机专业毕业的，有的可能没有上过大学，他们未接受过系统的软件基础理论教育，软件设计水平仍不太高。在软件开发过程中，他们只是不自觉地采用了一些规律性的设计方法，或者模仿别人的程序设计方法，而有更多成熟的基本方法没有掌握，致使开发出来的软件水平不高，使产品的功能和可靠性受到一定的制约。

笔者曾经写过《单片机应用程序设计技术》一书，该书以程序设计的经验和技巧为主，而本书则以程序设计的基本算法为主，目的在于全面提高单片机程序设计者的软件素质。

软件设计是一门科学，有其自身的规律，也有很多成熟的理论和算法。对于广大的单片机技术人员，不大可能再脱产进大学系统学习基础理论，只能一边干一边学。要学习就要选教材，而目前能选到的教材都是专为大学生编写的教材，如《数据结构》、《计算方法》、《线性代数》、《代数与编码》等。这些教材起点较高，偏重理论证明，不考虑单片机的特点，对于广大单片机开发人员不是十分适合，学起来会感到比较抽象和吃力。

出于提高我国广大单片机开发人员软件素质的愿望，我们决定编写一本适合自学软件理论基础和基本算法的书。该书起点要求不高，只要有中学数学基础并已从事了一段时间单片机开发工作的人员就可以看懂。学完本书后，对单片机程序设计的主要基础理论和常用成熟算法也能初步掌握，在进行软件设计时，可以减少很多盲目性，并为更系统、更深入地学习计算机理论打下良好基础。

在内容上，我们选择最基本和最成熟的理论和算法予以介绍，主要范围为《数据结构》、《计算方法》、《线性代数》、《代数与编码》，但不可能介绍全部内容，只是从中选取与单片机开发联系最紧密的内容加以重新整理编排，打破大学教科书的传统教材格式，基本取消推导证明过程，尽量结合单片机的特点，使广大读者能够比较顺利地理解和接受。

通常介绍软件算法多是采用 PASCAL 语言，但为了使广大尚未学习过这种高级语言的读

者能看懂本书，我们采用我国当前最流行的MCS－51系列单片机的汇编语言，并配合程序流程图来描述各种算法，以便使大多数读者能够接受，并可直接利用。

在单片机中应用最多的是线性数据结构，故本书第一章首先介绍常用的线性数据结构及部分基础算法；第二章和第三章集中介绍线性数据结构的排序算法和查找算法；第四章介绍了几种常用的特殊算法，为进一步介绍非线性数据结构作准备；对于常用的非线性数据结构及其有关算法分别在第五、六章进行介绍；带单片机的电子仪器设备一般都具有较强的数据处理能力，为此在第七章中介绍常用线性方程组的解法；在第八章中介绍插值算法；在第九章中介绍数理统计中常用的计算方法；第十章介绍了常用编码方法，这对优化设计很有帮助；第十一章介绍了两个实例，作为全书的一个总结。附录A介绍了浮点数据处理的基本原理，这对不熟悉浮点数的读者会有一定帮助；附录B为本书上机操作指南。本书第一、二、三、四、五、六、十、十一章和两个附录由周航慈编写，第七、八、九章由饶运涛编写。全书由周航慈负责策划编排、文稿修改和审定。

为了更好地掌握本书内容，建议大家上机实际运行一下书中的程序，效果一定要好得多。上机的硬件条件要求不高，有一套最普通的IBM PC计算机加一套可以和IBM PC计算机联机调试的MCS—51开发系统即可。为配合读者学习，我们将书中的程序添加上各种支撑子程序，形成可执行可调试的运行程序，并进行了较严格的测试，然后汇总成一张配套的程序软盘[①]。在该盘中还包含了经过长期考验和优化的最新版本的子程序库，实用价值较高。

在本书编写过程中，得到出版社的大力支持，何立民教授和陈粤初教授给予了无私的帮助，高斌参与了本书的文字录入工作，吴允平和卢瑜参与了本书上机程序的测试工作，在此表示衷心感谢！

由于我们的水平有限，书中错误及不足之处在所难免，敬请广大读者予以指正，不胜感谢！

作者于江西省临川市

一九九六年八月

① 本书所附程序软盘邮购方法

- 邮购地址：（邮编344000）江西省临川市华东地质学院352信箱
- 邮购费：45元（含特快专递邮资，有发票）
- 联系人：高斌
- 注意事项：请在汇款单上说明邮购《单片机程序设计基础》上机软盘

目　录

第一章　单片机中常用的线性数据结构

计算机界有一个著名的公式：

数据结构＋算法＝程序

由公式可以看出，如果要进行程序设计，而对数据结构和算法的有关知识懂得不甚明了，那他设计出来的程序的质量必定不高，常见的表现有：

- 可靠性差：有时工作正常，有时又不正常，甚至给出错误的结果。
- 功能低：只能处理简单的任务，对于用户提出的比较复杂的任务不知如何实现。
- 效率低：处理问题时采用的方法较笨拙，比正常的方法费时费资源。
- 可维护性差：对于运行中暴露出来的问题和用户提出的修改意见，很难进行修改和扩充，甚至越改问题越多。

可以说，数据结构和算法是程序设计的基础，必须认真学好。

§1.1　数据结构的基本概念

首先，我们来讨论一下什么是数据。在计算机中，一切由计算机处理的对象都是数据，既包括我们日常生活中所讲的数据，也包括我们日常生活中认为不是数据的东西：文字、图形、声音等等信息。这些信息都必须用某种格式的二进制代码来表示，才能被计算机识别和加工处理，而一旦表示成某种格式的二进制代码，在计算机看来也就都成为数据了。

同一类数据可能有很多个，我们把数据的最小单元称为数据元素，一个数据元素内部也许还有更细的组织结构。例如在一个数据采集系统中有 100 个不同位置的信息需要采集，则每个位置的信息就是一个数据元素，总共就有 100 个数据元素。每个位置上要采集的信息可能有若干种：温度、湿度、风速、气压等，且每种采样对象的数据格式、精度、单位都有可能不一样。因此，一个数据元素的内部结构可分为若干个数据域（如图 1-1 所示），每个数据域的数据格式以及数据长度也可以不一样。在多数情况下，一个数据元素只有一个数据域，这时问题也就变得简单一些了。

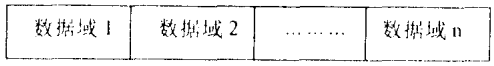


图 1-1　数据元素的内部结构

其次，我们再来讨论一下什么是数据结构。通常计算机处理的数据量是很大的，有时也是很复杂的。各种数据不能随意乱存放，必须根据具体数据的特点按一定的规则来组织和存放，才能有效地加以处理。分析和规划各个数据元素之间的相互关系、对数据进行有效的组织、解决数据存储方式、设计出对应的基本数据处理算法，这些就是“数据结构”所要研究的内容。其中，各数据元素之间的相互关系有两层含义：一种是指各数据元素之间的抽象关系（例如先后关系、层次关系等），即“逻辑结构”；另一种是指它们在计算机中存放位置（地址）之间的关系，即“存储结构”。例如，我们通过 A/D 转换，采集了 100 个信号电压值，则每一个信号电压的采样值就是一个数据元素。这 100 个数据元素相互之间有一个时间上的先后次序，从逻辑上讲是

一种顺序的线性“逻辑结构”。如果我们把这 100 个采样数据依次存放在一片连续的 RAM 地址中，则它的“存储结构”也是线性的，即可以用一个线性方程表达式计算出任何一个采样值在 RAM 中的实际存放地址。

对应于每种特定的“逻辑结构”和“存储结构”，相应都有成熟的数据处理方法，因此，这些基本算法理所当然成为“数据结构”的重要组成部分。

1.1.1 逻辑结构

各数据元素之间的抽象关系称为“逻辑结构”。这种抽象关系只考虑它们之间逻辑上的先后次序、上下层次和相邻情况，而不考虑各数据元素实际存放地址之间的相互关系。如果各数据元素之间在逻辑上只有先后次序，不分上下层次，则这种逻辑结构称为“线性结构”，又称“线性表”。最常见的各种表格、数组、采样数据块等都属于“线性表”。如果各数据元素之间在逻辑上还有上下层次关系或各元素之间互相联系的情况不规则，不能简单地用线性关系来表达时，它们就属“非线性结构”。“树型结构”和“图型结构”是两种最主要的“非线性结构”。

为了直观表达数据结构的逻辑分类，我们用一个节点表示一个数据元素，用线段表示数据元素之间的相互关系，则三种常见的逻辑结构如图 1-2 所示。

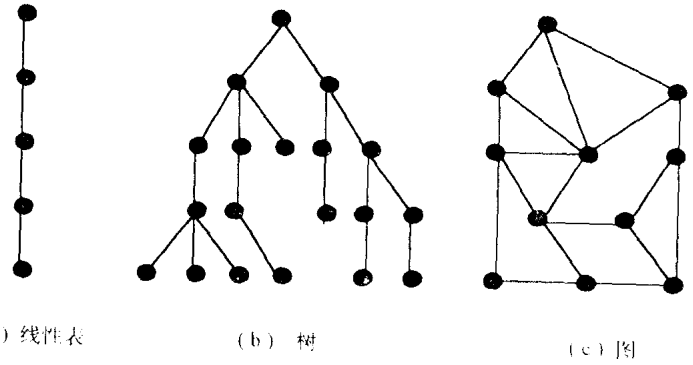


图 1-2 数据结构的逻辑分类

从图 1-2 中可以看出，线性表中的每个数据元素最多有两个相邻的元素，一个叫直接前趋，一个叫直接后继。第一个数据元素没有前趋，最后一个数据元素没有后继。每个数据元素的相对位置可以有一种直观的表示方法，即元素序号。而树的特点是每个数据元素（除树根外）只有一个直接前趋，但直接后继的数目不受限制，对于树叶，则没有后继。而图的关系最为复杂，每个数据元素的直接前趋数和直接后继数（在无序图中，不分前趋和后继，都看成相关联元素）是不受限制的。

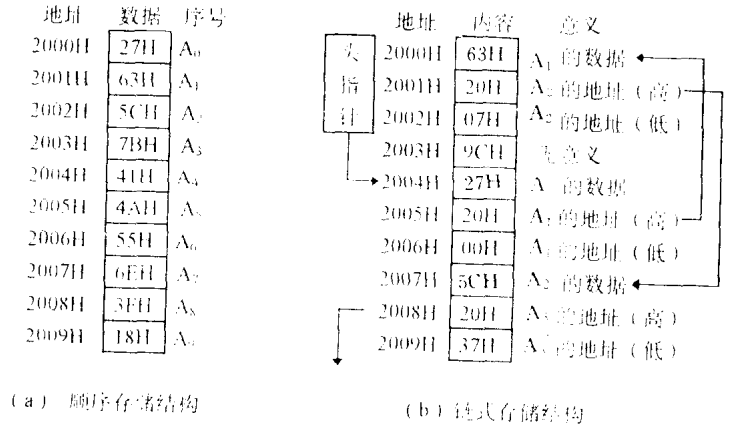
在单片机中，广泛使用的是“线性结构”，故在本章中首先加以介绍。非线性结构的“树型结构”和“图型结构”主要用于一些比较复杂的、智能化程度比较高的系统中，用来进行决策控制、优化算法、具有一定人工智能的数据处理算法等，我们将在第五章和第六章中进行讨论。

1.1.2 存储结构

抽象的数据结构（逻辑结构）最终必须以某种具体的形式来实现，这种具体的形式就是“存储结构”。“存储结构”可用四种基本方法来实现。第一种为“**顺序存储**”，即将各数据元素按逻辑上的顺序存放在一段连续的空间内，并使逻辑上相邻的元素在存储空间上也相邻。第二种为“**链式存储**”，各数据元素存放地址不受约束，可以连成一片，也可以分散在不连续的若干个地

址上，即使连成一片，各数据元素之间的关系也与物理地址先后无关。这种存储结构靠一种特殊的手段来表示各元素的相邻关系：即链接指针。在链式存储结构中，每个数据元素都包含两个以上的数据域，其中至少有一个数据域用于存放本身的数据，剩下的数据域（至少有一个）用于存放链接指针。链接指针用于指明与本数据元素在逻辑上有关的其它数据元素的存放地址，故这种存储结构称为“链式存储结构”。第三种为**“索引存储”**，在这种存储方法中保存有一个索引表，索引表的每一项由两部分组成，其中一项表示数据元素的特征（即关键字），另一项中保存有对应数据元素的实际存放地址。第四种为**“散列存储”**，在这种方法中直接利用数据元素的特征（即关键字）来计算该数据元素的实际存放地址。在单片机应用系统中，绝大多数情况下采用的是顺序存储结构。

逻辑结构为线性结构的线性表常采用顺序存储结构来实现，这时我们把线性表的这种顺序存储结构称为向量，线性表中各元素用序号来指明，如图 1-3(a)所示。当然，并不是线性表一定要用顺序存储结构来实现，用链式存储结构同样可以实现，如图 1-3(b)所示。为了对链式



（a）顺序存储结构　　（b）链式存储结构

图 1-3　线性表的存储结构

存储结构有一个正确的理解，我们可以打个比方：有一个毕业班有 30 个学生，学生的编号从 1 到 30，他们来到一个工厂实习，住进工厂的招待所。最好的方法是将他们住在连续的若干个房间里，并按编号顺序分配床位，这种分配方式相当于“顺序存储结构”，这对找人是很方便的，但有一个前提，招待所必须事先准备好连续在一起的 30 个以上的空床位。如果临时来到工厂，招待所虽然有 30 个以上的空床位，但一般不能保证这些空床位连续在一起，这时只能用见缝插针的方式安排住宿，以充分利用零散的空床位。为了保持学生之间的学号相邻关系，每个学生只要记住学号比自己大一号的学生的住处即可，并且 1 号学生的住处是公开的。这样，如果我们想要找某个学生，就可以通过 1 号学生先打听到 2 号学生的住处并找到 2 号学生，再按同样的方法通过 2 号学生找到 3 号学生……，直到找到想找的学生。这种分配方式相当于“链式存储结构”，保存 1 号学生住处的指针变量叫头指针。从这个例子可以看出，链式存储结构有充分利用零散空间的优点，但查找效率较低。事实上，到招待所找人一般都是先到服务台去查住宿登记簿，在登记簿上按所找对象的姓名（或学生的编号）就可以查到所住床位（或房号）了，这时采用的就是“索引存储”方法，住宿登记簿就是索引表，查找对象的姓名就是关键词。

在大多数数据处理算法中，顺序存储结构比其它非顺序存储结构的处理速度要快得多。

而对于非线性数据结构(树或图),通常是靠非顺序存储结构来实现。不过,如果存储空间很富裕,也可以通过一些技巧,用一种类似顺序存储结构的方式来存储树和图,这时往往要浪费一些存储空间,以此来获得处理数据时的方便(即以空间换时间),在第五、六章中我们将详细讨论这一问题。链式存储结构必须要有一套专门的存储管理系统来支持,在通用计算机中,这一功能由操作系统或高级语言的编译系统来实现。在高级的单片机应用系统中,这一功能由实时多任务操作系统来实现。在普通的单片机应用系统中,为了降低程序设计的难度,尽量避免采用链式存储结构。

1.1.3 算 法

数据结构的逻辑设计和存储设计只解决了数据的组织和存放问题,我们的最终目标是要对数据进行处理。每种数据处理都有一定的目的,习惯上,我们就把实现这个目的的处理方法称为算法。算法并不仅仅是计算方法,还包括逻辑推理等过程。并不是任何一个处理方法都可以称为算法。一个真正的算法必须具有如下特性:

· 有穷性:即一个算法启动后,在任何情况下都能在执行有限的步骤后结束。如果一个运算过程在大多数情况下都能停下来,而在某些情况下就永远不会停下来(除非关机),则这个运算过程不能称为算法。另外,表面上看来是有穷的,但当其执行时间太长,以致不能在人们可以接受的时间内完成,也不能称为有效算法。

· 确定性:在算法的每一步骤中,其操作顺序和操作内容都有确切的定义,不能有任何歧义性。

· 数据输入:一个算法一般需要有若干个输入数据作为原始数据。某些特殊用途的算法可能不需要输入原始数据。

· 信息输出:一个算法一定是用来解决某种问题的,解决的结果是人们所需要的,必须将其输出。

· 可行性:算法的每一步都必须是计算机可以执行的,或是可以分解为若干条具体的计算机指令,而被计算机完成。

在单片机中,我们主要处理的数据结构是线性表,对于线性表,有几种基本的操作,每种操作都有对应的算法。对于同一种操作,由于存储结构不同,算法也不同。

线性表基本操作如下:

· 存取操作:又称"读写操作",这是最基本的操作。"存"操作("写"操作)用来给某数据元素赋值或修改某数据元素的值,"取"操作("读"操作)用来读取某数据元素的值。

· 插入操作:在原线性表的某个希望的位置上插入一个新的数据元素。这种操作将导致数据元素的总数增加一个。

· 删除操作:将某个指定的数据元素删除。这种操作将导致数据元素的总数减少一个。我们特别要注意,删除操作和清零操作是不同的。清零操作属于"写"操作,它将某个指定的数据元素清零后,该数据元素仍然存在,只是数值改变,而数据元素总个数并不减少。

· 合并操作:将两个以上的线性表合并为一个线性表。

· 分解操作:将一个线性表分解为几个线性表。

· 复制操作:在另一个物理空间生成一个同样的线性表。当我们需要保护线性表的原始内容时,事先复制一个线性表作为存根就很有必要。

· 排序操作:将线性表中的数据元素按某种特定的要求进行整理。该操作是一种基本的预

处理操作。主要是为其它操作(如查找、统计分析)创造一个有序环境,从而大大提高后续处理过程的效率。本书第二章将进行较详细的讨论。

·查找操作:在线性表中查找(检索)某特定的数据元素。本书第三章中将进行较详细的讨论。

·求线性表的长度:核准线性表当前有效数据元素的个数。

§1.2 简单变量

数据结构是讨论同一类数据元素之间的组织方式和存储方式的问题。当数据元素的个数只有一个时,就变成了简单数据项。对于简单数据项(简单变量)也就没有逻辑结构等方面的问题可讨论了,一般也不属于数据结构的讨论范围。但为了使以后讨论更方便,我们还是将简单变量的有关问题单独提出来分析一下。

1.2.1 系统变量

系统变量又称全局变量。程序的任何部分(主程序、子程序、中断子程序)如果需要的话均可以对其进行访问,其生命周期为整个程序的运行期。在单片机应用系统中常见的系统变量有:系统状态字、系统状态标志、系统时钟等。系统变量的存放地址和数据格式是始终固定不变的,为了系统变量的安全,严禁将它们存放在操作频繁的工作寄存器中。

1.2.2 临时变量

临时变量又称局部变量。它们只为特定的程序段服务。其生命期与使用该临时变量的有关程序段相同,使用完后即作废,其占据的存储空间即可用来存放其它临时变量。一个算法所需原始数据的拷贝、控制运算进程的辅助变量、运算的中间结果等一般都需要使用临时变量。在MCS-51系列单片机中,R0~R7被称为工作寄存器,用它们来存放临时变量是最合适的。

1.2.3 计数器

单片机中的计数器有硬件计数器和软件计数器之分。硬件计数器可以独立完成对外部事件的计数,如MCS-51系列单片机中的T0和T1。我们这里只讨论软件计数器,它完全由软件设立,并由软件来操作。软件计数器主要有两个方面的用途:

·统计某种事件的个数:初始化时为零,在程序运行过程中出现某种预定的事件时就加一,运行结束时就可以从计数器中得知某预定事件出现的总次数了。最常见的例子是分类统计程序,它将一批数据按照某种标准分成若干类,每一类设立一个对应的软件计数器,通过比较判断来操作对应的软件计数器就可以完成分类统计任务了。

·控制某种操作的次数:初始化时设定一个控制次数,程序运行过程中,每当执行完一次预定的操作就将计数器减一,当计数器减为零时就结束预定的操作。这一类算法一般设计为一个循环过程,计数器充当了循环控制变量。

软件计数器的设计要考虑到计数值的最大范围。如果不超过255,使用单字节变量即可。如果有可能超过255,但不可能超过65535,则必须用双字节(16位)的计数器。如果有可能超过65535,则必须用三字节以上的计数器。

在8位单片机中,往往不能用一条指令来完成双字节以上的计数器加一或减一操作,这时处理起来就要小心一些。例如在MCS-51单片机中,双字节加一的指令只有一条,即INC DPTR,而DPTR一般用来作地址指针,很少用来作计数器;双字节减一的指令一条也没有。因

此，对于双字节以上的计数器的加一和减一操作，一般要通过几条指令来完成。设R2和R3组成一个双字节计数器，R2为高字节，R3为低字节。当进行加一操作时，可采用如下进位调整算法：

```
        MOV     A,R3
        INC     A
        MOV     R3,A
        JNZ     XXX
        INC     R2
XXX:    .
        .
```

也可再简单一些：

```
        INC     R3
        CJNE    R3,#0,XXX
        INC     R2
XXX:    .
        .
        .
```

当进行双字节减一操作时，可采用下列指令：

```
        CJNE    R3,#0,XXX
        DEC     R2
XXX:    DEC     R3
        .
        .
        .
```

当用双字节计数器来控制一个循环过程时，人们爱使用DJNZ指令，使程序更简洁。例如，我们用R2R3来控制循环的次数，R2为高字节，R3为低字节。循环开始前将循环次数先装到R2R3中，然后用两个DJNZ指令来控制：

```
        MOV     R2,#20H         ;设定循环执行203CH次。
        MOV     R3,#3CH
LOOP:   .
        .                       ;循环体。
        .
        DJNZ    R3,LOOP         ;控制循环次数。
        DJNZ    R2,LOOP
        .
        .
        .
```

以上程序是很常见的，但实际上是有很大风险的。为了说明这一点，我们用DPTR来统计实际循环执行的次数：

```
        MOV     DPTR,#0
        MOV     R2,#20H
        MOV     R3,#3CH
LOOP:   INC     DPTR
```

```
        .
        .
        .
        DJNZ    R3,LOOP
        DJNZ    R2,LOOP
```

执行完这一段程序后，DPTR 的内容（即循环体实际执行次数）应该等于 203CH，但实际上只有 1F3CH。如果(R2)＝20H，(R3)＝00H，运行结果(DPTR)＝2000H，正是我们所希望的。分析可知：当计数器低字节为零时，循环次数正确；计数器低字节不为零时，循环次数少 256 次。当实际控制次数不是立即数，而是一个变量值时，必须在进入循环前先对计数器低字节进行检查和调整。如果低字节不为零，就应该人为地将高字节加一，以保证循环次数正确。设循环次数在双字节变量 30H（高字节）和 31H（低字节）中，要求循环结束后不要破坏 30H 和 31H 的内容，则算法如下：

```
        MOV     R2,30H      ;装入高字节。
        MOV     A,31H       ;取低字节。
        MOV     R3,A        ;装入低字节。
        JZ      LOOP        ;判断。
        INC     R2          ;调整高字节。
LOOP:   .                   ;循环体
        .
        .
        DJNZ    R3,LOOP     ;循环控制（低字节）
        DJNZ    R2,LOOP     ;循环控制（高字节）。
        .
        .
```

1.2.4 指　针

在单片机中，为了对一个数据集合中的元素进行操作，首先要知道该元素的实际存放地址，而用来存放地址信息的变量就是指针变量。指针也是一种简单变量，其中的内容是一种位置信息，按其内容的性质可分为绝对指针和相对指针。绝对指针中存放的内容就是真实的地址，指针可以是 8 位的（如 MCS－51 单片机进行片内 RAM 寻址的 R0，R1），也可以是 16 位的（如 MCS－51 单片机进行片外 RAM 寻址的 DPTR，P2＋R0，P2＋R1）。当数据结构为线性表时，相对指针一般存放的是数据元素的编号（若为数组，则存放的是数组元素的下标），要访问该元素必须先通过一定的计算才能得到物理地址。当元素个数不超过 255 个时，用一个字节即可作为相对指针。

很多算法都是通过修改指针来完成对一批数据元素的依次处理，因此指针的调整操作也是一种很频繁的操作。在 8 位机中，16 位指针的调整同计数器一样，也要小心处理。以 MCS－51 单片机为例：

- DPTR 加一：　INC　　DPTR
- DPTR 减一：

```
        MOV     A,DPL
        JNZ     XXX
        DEC     DPH
XXX:    DEC     DPL
```

```
        .
        .
        .
```

· P2+R0(或P2+R1)加一：

```
        INC     R0
        CJNE    R0,#0,XXX
        INC     P2
XXX:    .
        .
        .
```

· P2+R0(或P2+R1)减一：

```
        CJNE    R0,#0,XXX
        DEC     P2
XXX:    DEC     R0
        .
        .
```

§1.3 表 格

表格是线性表的一种最直观的表现形式，在单片机程序设计中被广泛采用。单片机中的表格几乎都是采用顺序存储结构(向量)。对于一个表格，我们必须知道这样几项参数：

· 表头地址：也就是表格中第一个数据元素存放的地址。在计算机中，第一个数据的序号往往编码为0，这样一来，0号元素的地址即为表头地址。

· 数据元素的大小规格：表格中所有数据元素均属同一类型，具有同样的结构，所需存储空间也一样，即所占字节数一样。知道每个数据元素所占字节数后，我们就可通过下述公式来计算序号为i的数据元素的地址了：

$$X_i = X_0 + i \times L$$

式中 X_i——待求地址(即 i 号元素的地址)；

X_0——表头地址(即0号元素的地址)；

i——数据元素的序号；

L——每个数据元素所占字节数。

如果第一个数据元素的序号编码为1，则计算公式就变为：

$$X_i = X_0 + (i - 1) \times L$$

显然计算起来麻烦了一些，这也就是在计算机中人们喜欢将 n 个元素的编码从0开始，到 $n-1$ 为止的原因。

· 数据元素的总个数：对于固定表格，数据元素的总个数是一个预先确定的常数，往往以立即数的形式直接写入程序中，而不需要占用RAM资源。对于动态表格，数据元素的个数是变化的，必须在RAM资源中设立一个计数器来保存当前数据元素的个数，而且它允许的最大的值必须预先确定好。

1.3.1 固定表格

当表格的规模和内容是固定不变时，我们称为固定表格。显然，固定表格具有只读性质，所