



单片机程序设计基础

(修订版)

周航慈 著



北京航空航天大学出版社
<http://www.buaapress.com.cn>

单片机程序设计基础

(修订版)

周航慈 著

北京航空航天大学出版社

<http://www.buaapress.com.cn>

内 容 简 介

本书是《单片机应用程序设计技术》(修订版)一书的姐妹篇。《单片机应用程序设计技术》的内容以程序设计的经验和技巧为主,而本书内容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。基本内容包括:单片机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及常用编码方法等。从单片机系统的实际硬件环境出发,语言通俗易懂,使读者轻松地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,本书配套了上机学习光盘,在光盘中还提供了最新修订的子程序库和相关调试工具软件,以及众多单片机资料。

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和研究生参考。

图书在版编目(CIP)数据

单片机程序设计基础/周航慈著. —2版(修订版).

—北京:北京航空航天大学出版社,2003.7

ISBN 7-81077-314-3

I. 单… II. 周… III. 单片微型计算机—程序设计
IV. TP368.1

中国版本图书馆 CIP 数据核字(2003)第 056318 号

单片机程序设计基础 (修订版)

周航慈 著

责任编辑 孔祥燮

*

北京航空航天大学出版社出版发行

北京市海淀区学院路 37 号(100083) 发行部电话:(010)82317024 传真:(010)82328026

<http://www.buaapress.com.cn>

E-mail: bhpess@263.net

河北省涿州市新华印刷厂印装 各地书店经销

*

开本:787×1092 1/16 印张:18.5 字数:474 千字

2003 年 7 月第 1 版 2003 年 7 月第 1 次印刷 印数:5 000 册

ISBN 7-81077-314-3 定价:32.00 元(附光盘)

前 言

嵌入式系统的应用越来越广,我国从事嵌入式系统开发的人员也越来越多,而单片机是嵌入式系统的核心单元。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。

在开发一种嵌入式系统产品(含单片机产品)时,有关单片机部分的工作主要是做两方面的设计: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,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对软件设计水平无疑起决定作用。软件设计水平在嵌入式系统产品开发的过程中占有重要地位,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、科研院所和大中型企业中,有一批专家,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电子产品开发的技术人员身在基层,基本上不是计算机专业毕业的,有的没有上过大学,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被掌握,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。

作者已经出版了《单片机应用程序设计技术》(修订版)一书。该书内容以程序设计的经验和技巧为主,而本书内容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。对于广大电子技术人员来说,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大都是专为大学生编写的,如《数据结构》、《计算方法》、《线性代数》及《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,对于广大电子产品开发人员来说不是十分适合,学起来会感到比较抽象和吃力。

出于提高我国广大电子产品开发人员软件素质的愿望,我决定编写一本适合自学软件理论基础和基本算法的书。该书起点不高,只要有中学数学基础,并已从事了一段时间电子产品的开发,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、线性代数及代数与编码,但不可能介绍全部内容,只是选取与单片机应用系统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。

在单片机中应用最多的是线性数据结构,故本书第 1 章首先介绍常用的线性数据结构及

部分基础算法;第2章和第3章集中介绍线性数据结构的排序算法和查找算法;第4章介绍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5,6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7章中介绍常用线性方程组的解法;第8章介绍插值算法;第9章介绍数理统计中常用的计算方法;第10章介绍常用编码方法,这对优化设计很有帮助;第11章介绍两个实例,作为全书的一个总结;附录A介绍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为本书上机学习光盘文件说明。

为了更好地掌握本书内容,建议上机实际运行一下书中的程序,效果一定要好得多。为配合学习,我们将书中的程序添加上主程序和各种支撑子程序以及数据样品,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上机学习光盘,附在本书中。在该光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。

本书1997年出版了第一版,受到读者好评;但也存在一些缺陷,本次修订工作就显得十分必要。修订工作如下:

(1) 对过时或不合适的字句进行修改,对插图中少量的瑕疵进行了弥补。

(2) 对汇编语言程序中的常量和变量在定义时分别用 EQU 和 DATA 进行区分,使程序更准确、更规范。

(3) 对上机学习光盘中的程序进行了全面整理,更方便上机学习。

(4) 为配合 C51 的推广,增加了大量与汇编语言程序对应的 C 算法程序。

(5) 增加了少量新内容。

在本书修订过程中,得到北京航空航天大学出版社的大力支持,在此表示衷心感谢!

由于水平有限,书中错误及不足之处敬请广大读者予以指正,不胜感谢!

作者于江西省抚州市

2003年3月

目 录

第 1 章 单片机中常用的线性数据结构

1.1	数据结构的基本概念	(1)
1.1.1	逻辑结构	(2)
1.1.2	存储结构	(3)
1.1.3	算 法	(4)
1.2	简单变量	(5)
1.2.1	系统变量	(5)
1.2.2	临时变量	(5)
1.2.3	计数器	(5)
1.2.4	指 针	(7)
1.3	表 格	(8)
1.3.1	固定表格	(9)
1.3.2	动态表格	(9)
1.3.3	线性表的插入算法	(10)
1.3.4	线性表的删除算法	(15)
1.4	数组和数据块	(19)
1.4.1	一维数组	(19)
1.4.2	多维数组	(20)
1.4.3	数据块操作	(22)
1.5	数据缓冲区	(24)
1.5.1	输入缓冲区	(24)
1.5.2	输出缓冲区	(25)
1.6	队 列	(26)
1.6.1	队列的基本概念	(26)
1.6.2	队列的存储结构	(27)
1.6.3	入队的算法	(29)
1.6.4	出队的算法	(30)
1.6.5	队列的应用实例	(33)
1.7	用户堆栈	(35)
1.7.1	用户堆栈的基本概念	(36)
1.7.2	用户堆栈的存储结构	(37)
1.7.3	入栈的算法	(38)
1.7.4	出栈的算法	(39)

第2章 线性表的排序算法

2.1 插入排序	(41)
2.2 选择排序	(43)
2.3 冒泡排序	(45)
2.4 归并排序	(49)
2.5 快速排序	(60)
2.6 排序算法比较	(66)

第3章 线性表的查找算法

3.1 顺序查找	(67)
3.2 折半查找	(68)
3.3 分块查找	(70)
3.4 串的匹配	(73)
3.5 查找算法应用实例	(78)

第4章 几种常用的特殊算法

4.1 递归算法	(80)
4.1.1 递归算法的特点	(80)
4.1.2 递归算法举例	(80)
4.2 递推算法	(83)
4.2.1 递推算法的适用性	(83)
4.2.2 递推算法举例	(83)
4.3 回溯算法	(85)
4.3.1 回溯算法的特点	(86)
4.3.2 回溯算法举例	(86)

第5章 树及其在单片机中的算法实现

5.1 树的基本概念	(92)
5.1.1 树的定义	(92)
5.1.2 树的基本概念	(93)
5.2 二叉树	(93)
5.2.1 二叉树的定义及其性质	(93)
5.2.2 完全二叉树及其特点	(94)
5.2.3 二叉树的顺序存储	(95)
5.2.4 二叉树按编号的遍历算法	(97)
5.2.5 二叉树的前根遍历算法	(98)
5.2.6 二叉树的中根遍历算法	(101)
5.2.7 二叉树的后根遍历算法	(103)

5.3 普通树	(105)
5.3.1 普通树的存储方法	(105)
5.3.2 普通树的前根遍历算法	(108)
5.3.3 普通树的后根遍历算法	(113)

第6章 图及其在单片机中的算法实现

6.1 图的基本概念	(118)
6.1.1 图的定义	(118)
6.1.2 图的基本概念	(118)
6.2 图的存储结构	(119)
6.2.1 邻接矩阵	(120)
6.2.2 邻接表	(121)
6.3 图的遍历算法	(123)
6.3.1 图的深度优先搜索遍历算法	(123)
6.3.2 图的广度优先搜索遍历算法	(129)
6.4 图的生成树和最短路径问题	(136)
6.4.1 图的生成树	(136)
6.4.2 图的最短路径问题	(142)
6.5 网络的最小生成树和最短路径问题	(145)
6.5.1 网络的最小生成树	(146)
6.5.2 网络的最短路径问题	(153)

第7章 常用线性方程组求解算法

7.1 引 言	(159)
7.2 主元消去法	(159)
7.2.1 无回代过程的主元消去法	(160)
7.2.2 有回代过程的主元消去法	(168)
7.3 三元线性方程组的行列式法	(173)
7.3.1 行列式法概述	(173)
7.3.2 三元线性方程组的行列式法	(173)

第8章 常用插值算法

8.1 引 言	(179)
8.2 线性插值算法	(179)
8.3 抛物线插值算法	(181)
8.3.1 算法概述	(181)
8.3.2 逐次线性插值算法	(182)
8.4 插值算法应用实例	(184)
8.4.1 线性插值算法应用实例	(184)

8.4.2 抛物线拟合算法应用实例	(186)
-------------------------	-------

第9章 常用数理统计分析

9.1 引言	(189)
9.2 均值和标准离差的估算	(190)
9.2.1 计算 $\bar{X}$ (估算均值 μ)	(190)
9.2.2 计算 S (估算标准离差 σ)	(191)
9.3 用数理统计方法消除粗大误差	(192)

第10章 编码方法简介

10.1 信息源及编码	(197)
10.1.1 顺序编码	(197)
10.1.2 特征编码	(198)
10.1.3 哈夫曼编码(变长码)	(199)
10.2 检错码	(202)
10.2.1 检错原理	(202)
10.2.2 奇偶校验	(203)
10.2.3 和校验	(205)
10.2.4 循环冗余校验(CRC)	(210)
10.3 纠错码	(213)
10.3.1 纠错原理	(214)
10.3.2 汉明码	(214)
10.3.3 检二纠一码	(218)
10.3.4 矩形码	(231)
10.4 随机数发生器	(235)
10.4.1 交互式随机数发生器	(235)
10.4.2 用线性移位寄存器构成随机数发生器	(238)
10.4.3 软件随机数发生器	(242)

第11章 算法设计示例

11.1 老鼠过迷宫	(245)
11.1.1 数学模型分析	(245)
11.1.2 算法设计	(247)
11.1.3 数据结构设计	(251)
11.1.4 程序设计	(252)
11.2 仪器系数自动标定	(256)
11.2.1 数学模型分析	(257)
11.2.2 算法设计	(258)
11.2.3 数据结构设计	(260)

11.2.4 程序设计	(260)
-------------------	-------

附录 A 浮点数据处理的基本原理

A.1 浮点数的基本概念	(266)
A.1.1 十进制浮点数	(266)
A.1.2 二进制浮点数	(267)
A.1.3 浮点数的规格化	(268)
A.2 浮点运算基本原理	(269)
A.2.1 浮点加减运算基本原理	(270)
A.2.2 浮点比较运算基本原理	(272)
A.2.3 浮点乘法运算基本原理	(273)
A.2.4 浮点除法运算基本原理	(274)
A.2.5 浮点开平方运算基本原理	(274)
A.2.6 浮点多项式运算基本原理	(277)
A.2.7 浮点超越函数运算基本原理	(279)

附录 B 本书配套上机学习光盘说明

参考文献

第 1 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：

$$\text{数据结构} + \text{算法} = \text{程序}$$

由公式可以看出,如果要进行程序设计,但对数据结构和算法的有关知识不甚了解,那设计出来的程序的质量必定不高,常见的表现有：

- 可靠性差:工作有时正常,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,既包括日常生活中所讲的数据,也包括日常生活中认为不是数据的东西:例如文字、图形或声音等信息。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:例如温度、湿度、风速及气压等,且每种采样对象的数据格式、精度及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。

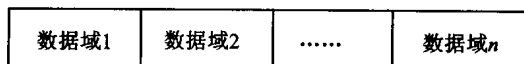


图 1-1 数据元素的内部结构

其次,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。各种数据不能被随意乱存放,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,解决数据存储方式和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;二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,则它们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、数组及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,它们就属于非线性结构。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。

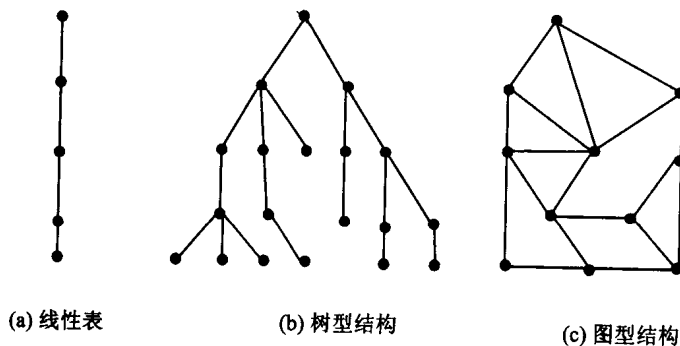


图 1-2 数据结构的逻辑分类

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,另一个叫直接后继。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表示方法,即元素序号。树型结构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。图型结构的关系最为复杂,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;而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、优化算法及具有一定人工智能的数据处理算法等,将在第 5 章和第 6 章中对其进行讨论。

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。存储结构可用以下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: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他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(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(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。为了对链式存储结构有一个正确的理解,我们可以打个比方:一个毕业班有30名学生,学生的编号1~30。他们来到一个工厂实习,住进工厂的招待所。最好的方法是让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”,对找人是方便的,但有一个前提,招待所必须事先准备好连续的、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床位即可,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号学生的床位并找到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(或学生编号)即可查到所住床位(或房号),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,顺序存储结构比其他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的。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(以空间换时间),在第5,6章中将详细讨论这一问题。链式存储结构必须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

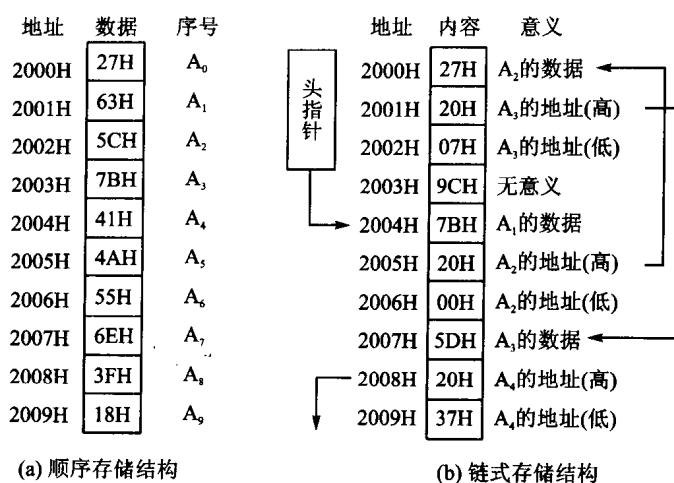


图 1-3 线性表的存储结构

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，而我们的最终目标是对数据进行处理。每种数据处理方法都有一定的目的，习惯上，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。

并不是任何一种处理方法都可以称为算法。一种真正的算法必须具有如下特性：

- 有穷性：即一种算法启动后，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：一种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：一种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，而被计算机完成的。

在单片机中，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，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。这里特别要注意,删除操作和清零操作是不同的。清零操作属于“写”操作,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,且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。当需要保护线性表的原始内容时,事先复制一个线性表作为备份就很有必要。
- 排序操作:将线性表中的数据元素按某种特定的顺序进行整理。该操作是一种基本的预处理操作。主要是为其他操作(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2章将对其进行较详细的讨论。
- 查找操作:在线性表中查找(检索)某特定的数据元素。本书第3章中将对其进行较详细的讨论。
- 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,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、子程序及中断子程序)如果需要的话,均可以对其进行访问(读或写),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、系统状态标志和系统时钟等。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他临时变量。一种算法所需原始数据的拷贝、控制运算进程的辅助变量及运算的中间结果等一般都需要使用临时变量。在80C51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80C51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。软件计数器主要有如下两方面的用途。

- 统计某种事件发生的次数:初始化时为零,在程序运行过程中,发生某种预定的事件时就加 1,运行结束时即可从计数器中得知某预定事件发生的总次数。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,即可完成分类统计任务。
- 控制某种操作的次数:初始化时设定一个控制次数,在程序运行过程中,每当执行完一次预定的操作就将计数器减 1,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,则使用单字节计数器即可;如果有可能超过 255,但不可能超过 65 535,则必须用双字节(16 位)的计数器;如果有可能超过 65 535,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 1 或减 1 操作,这时处理起来就要小心一些。例如在 80C51 系列单片机中,双字节加 1 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 1 的指令一条也没有。因此,对于双字节以上的计数器的加 1 和减 1 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 1 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:    ;
;后续程序

```

也可再简单一些:

```

INC     R3
CJNE    R3,#0,XXX
INC     R2
XXX:    ;
;后续程序

```

当进行双字节减 1 操作时,可采用下列指令:

```

CJNE    R3,#0,XXX
DEC     R2
XXX:    DEC    R3
;
;后续程序

```

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,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:   ;
DJNZ    R3,LOOP    ;控制循环次数

```

```

    DJNZ    R2,LOOP
    ;
    ;后续程序

```

以上程序是很常见的,但实际上是有很风险的。为了说明这一点,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
    ;
    ;后续程序

```

执行完这一段程序后,DPTR 的内容(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 的内容为 2000H,正是我们希望的。分析后可知:当计数器低字节为零时,循环次数正确;当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 1,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)
    ;
    ;后续程序

```

1.2.4 指 针

在单片机中,为了对一个数据集中的元素进行操作,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 80C51 系列单片机进行片内 RAM 寻址的 R0,R1),也可以是 16 位的(如 80C51 系列单片机进行片外 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 时,用 1 B 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的,因此指针的调整操作