

# 数据结构

杨正宏 编著

- ★ 内容采用循序渐进的方式，包括数组、链接、递归、栈、队列、树、图、排序和查找等
- ★ 精辟深入的概念分析与丰富的实例解说
- ★ 习题突出本章内容的重点和难点
- ★ 各种应用实例探讨，以C语言实际操作

## [目录]

|       |              |
|-------|--------------|
| ..... | [第1章] 数据结构概论 |
| ..... | [第2章] 数组结构   |
| ..... | [第3章] 链表     |
| ..... | [第4章] 递归     |
| ..... | [第5章] 栈      |
| ..... | [第6章] 队列     |
| ..... | [第7章] 树      |
| ..... | [第8章] 图      |
| ..... | [第9章] 排序     |
| ..... | [第10章] 查找    |

数据结构



# 数 据 结 构

杨正宏 编著

中 国 铁 道 出 版 社

2 0 0 1 年 • 北 京

## (京)新登字 063 号

北京市版权局著作权合同登记号: 01-2001-1329 号

### 版 权 声 明

本书中文繁体字版由台湾松岗电脑图书资料股份有限公司出版(2000)。本书中文简体字版经台湾松岗电脑图书资料股份有限公司授权由中国铁道出版社出版(2001)。任何单位或个人未经出版者书面允许不得以任何手段复制或抄袭本书内容。

### 图书在版编目(CIP)数据

数据结构/杨正宏编著. —北京: 中国铁道出版社, 2001.6

ISBN 7-113-04187-6

I. 数… II. 杨… III. 数据结构 IV. TP311.12

中国版本图书馆 CIP 数据核字 (2001) 第 24089 号

书 名: 数据结构

作 者: 杨正宏

出版发行: 中国铁道出版社 (100054, 北京市宣武区右安门西街 8 号)

策划编辑: 苏 茜

特邀编辑: 王占清

封面设计: 孙天昭

印 刷: 北京兴顺印刷厂

开 本: 787×1092 1/16 印张: 16.75 字数: 400 千

版 本: 2001 年 6 月第 1 版 2001 年 6 月第 1 次印刷

印 数: 1~5000 册

书 号: ISBN 7-113-04187-6/TP·553

定 价: 25.00 元

版权所有 盗印必究

凡购买铁道版的图书, 如有缺页、倒页、脱页者, 请与本社计算机图书批销部调换。

# 出版说明

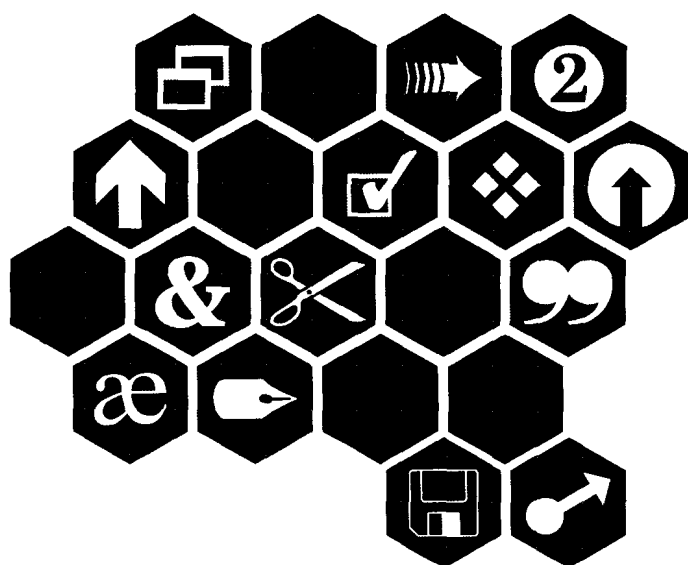
数据结构在计算机科学领域中有着不可缺少的重要性，是学习计算机专业的必修课程。数据结构主要探讨的是计算机内部各种数据存储的方式，并对如何有效地维护、处理和应用数据，提供评估与分析方法。本书的课程纲要共分十个章节，分别是概论、数组、链接、递归、栈、队列、树、图、排序和查找等，详细介绍了数据结构中每个重要的领域，以表达完整的数据结构概念，增进学习效果。

本书对于理论的介绍力求深入浅出，以便读者能够彻底了解各主题的理论根据。此外，为了让读者能够知道如何应用，再辅以日常生活的应用实例说明，以加深对其内容概念的了解，有助于读者的学习。本书采用 SPARKS 语言（是一种伪语言，类似 Pascal——编者著）表达各种算法，并用目前使用最广泛的 C 程序语言来实际操作。C 程序语言具有极强的控制结构，提高程序的可读性，并可使用如栈、链接和树等数据结构，以不同的方法来实际操作运用各种算法，有助于读者学习程序设计要领并深入理解其原理。

本书由台湾松岗电脑图书资料股份有限公司提供版权，经中国铁道出版社计算机图书项目中心审选，由乔林、王秀平、邓雄、王涛、李谨、陈贤淑、刘涛、何伟、肖志军、孟丽花等同志完成本书的整稿及编排工作。

中国铁道出版社

2001 年 6 月



# 目 录

|                                 |    |
|---------------------------------|----|
| 第 1 章 数据结构概论 .....              | 1  |
| 1.1 数据与信息 .....                 | 2  |
| 1.2 数据处理(Data Processing) ..... | 3  |
| 1.3 计算机任务处理的方式 .....            | 5  |
| 1.4 程序的产生 .....                 | 5  |
| 1.5 程序的分析 .....                 | 6  |
| 1.6 算法 .....                    | 9  |
| 1.7 复杂度(Complexity) .....       | 13 |
| 1.8 NP-COMPLETE 问题 .....        | 15 |
| 1.9 参数的传递 .....                 | 15 |
| 1.10 数据结构(Data Structure) ..... | 17 |
| 习题 .....                        | 18 |
| 第 2 章 数组结构 .....                | 19 |
| 2.1 数组的定义 .....                 | 20 |
| 2.2 数组表示法 .....                 | 20 |
| 2.3 稀疏矩阵(Sparse Matrix) .....   | 26 |
| 2.4 数组的应用 .....                 | 28 |
| 2.4.1 多项式的数据结构 .....            | 28 |
| 2.4.2 多项式相加 .....               | 30 |
| 2.4.3 上三角形和下三角形存储方式 .....       | 31 |



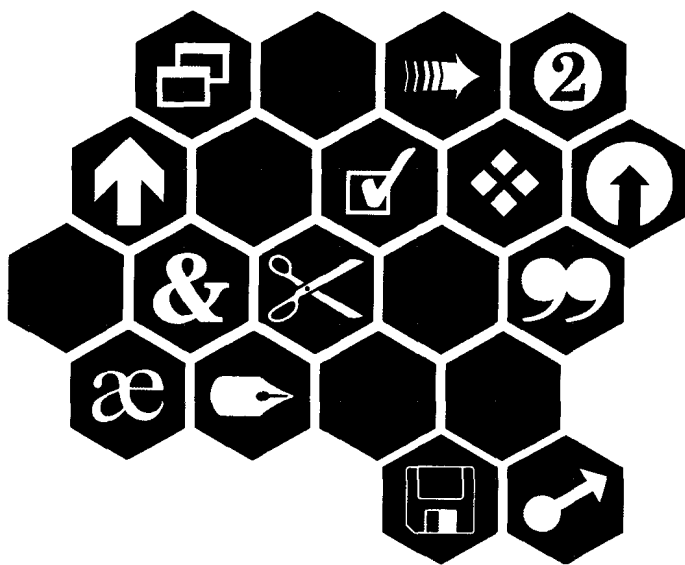
|                                                |           |
|------------------------------------------------|-----------|
| 2.4.4 矩阵乘积.....                                | 34        |
| 习题.....                                        | 35        |
| <b>第3章 链 表.....</b>                            | <b>37</b> |
| 3.1 链表的定义.....                                 | 38        |
| 3.2 动态内存分配.....                                | 38        |
| 3.3 链表的建立.....                                 | 38        |
| 3.4 链表的遍历.....                                 | 41        |
| 3.5 链表的连接.....                                 | 42        |
| 3.6 链表内结点的删除.....                              | 42        |
| 3.7 释放链表的内存空间.....                             | 43        |
| 3.8 链表内结点的插入.....                              | 44        |
| 3.9 链表结构的反转.....                               | 45        |
| 3.10 循环链表结构.....                               | 47        |
| 3.11 使用循环链表结构表示稀疏数组.....                       | 51        |
| 3.12 双向链表结构.....                               | 55        |
| 3.13 循环双向链表结构.....                             | 60        |
| 习题.....                                        | 62        |
| <b>第4章 递 归.....</b>                            | <b>63</b> |
| 4.1 何谓递归.....                                  | 64        |
| 4.2 递归工作原则.....                                | 65        |
| 4.3 递归的执行过程.....                               | 66        |
| 4.4 递归的应用.....                                 | 69        |
| 4.4.1 汉诺塔问题(Towers of Hanoi) .....             | 69        |
| 4.4.2 迷宫问题(Mazing Problem) .....               | 71        |
| 4.4.3 八皇后问题(Eight Queen Problem) .....         | 74        |
| 4.4.4 骑士问题.....                                | 78        |
| 4.5 递归程序与非递归程序的差异.....                         | 80        |
| 习题.....                                        | 81        |
| <b>第5章 栈.....</b>                              | <b>83</b> |
| 5.1 栈的定义.....                                  | 84        |
| 5.2 栈的表示及操作方式.....                             | 84        |
| 5.3 栈的应用.....                                  | 86        |
| 5.3.1 算术运算式的转换(Expression Conversion) .....    | 86        |
| 5.3.2 子程序调用(Subroutine call) .....             | 90        |
| 5.3.3 中断处理 (Interrupt Processing) .....        | 90        |
| 5.3.4 编译错误处理(Compiler Syntax Processing) ..... | 91        |

|                                                       |            |
|-------------------------------------------------------|------------|
| 5.3.5 汉诺塔问题(Towers of Hanoi)                          | 91         |
| 5.3.6 迷宫问题(Mazing Problem)                            | 94         |
| 5.3.7 八皇后问题(Eight Queen Problem)                      | 96         |
| 习题                                                    | 98         |
| <b>第6章 队 列</b>                                        | <b>101</b> |
| 6.1 队列的定义                                             | 102        |
| 6.2 线性队列的表示及操作方式                                      | 102        |
| 6.2.1 以数组表示线性队列                                       | 102        |
| 6.2.2 以链表表示线性队列                                       | 106        |
| 6.3 循环队列的表示及操作方式                                      | 107        |
| 6.3.1 以数组表示循环队列                                       | 107        |
| 6.3.2 以链表表示循环队列                                       | 110        |
| 习题                                                    | 112        |
| <b>第7章 树</b>                                          | <b>113</b> |
| 7.1 基本术语                                              | 114        |
| 7.2 树的表示法                                             | 115        |
| 7.3 二叉树                                               | 118        |
| 7.3.1 二叉树的建立                                          | 121        |
| 7.3.2 二叉树的遍历                                          | 122        |
| 7.3.3 二叉树的排序                                          | 125        |
| 7.3.4 二叉树的查找                                          | 126        |
| 7.3.5 二叉树的删除                                          | 126        |
| 7.3.6 一般树转换至二叉树                                       | 128        |
| 7.3.7 二叉表示树(Binary Expression Tree)                   | 130        |
| 7.3.8 相关二叉树                                           | 133        |
| 7.3.8.1 完全平衡树(Perfectly Balanced Tree)                | 133        |
| 7.3.8.2 满二叉树(Full Binary Tree)                        | 134        |
| 7.3.8.3 完全二叉树(Complete Binary Tree)                   | 134        |
| 7.3.8.4 线索二叉树(Threaded Binary Tree)                   | 134        |
| 7.3.8.5 扩充二叉树(Extended Binary Tree)                   | 135        |
| 7.3.8.6 哈夫曼树(Huffman Tree)                            | 137        |
| 7.4 树的应用                                              | 140        |
| 7.4.1 皇后问题                                            | 140        |
| 7.4.2 井字游戏                                            | 141        |
| 7.4.3 决策树                                             | 143        |
| 7.4.4 高度平衡二叉树 (Height Balanced Binary Tree, AVL Tree) | 144        |
| 7.4.5 2-3 树与 2-3-4 树                                  | 148        |
| 7.4.6 红-黑树                                            | 152        |



|                                                              |            |
|--------------------------------------------------------------|------------|
| 7.4.7 最小-最大堆集树.....                                          | 154        |
| 7.4.8 双堆集树 .....                                             | 156        |
| 7.4.9 B 树 .....                                              | 157        |
| 习题 .....                                                     | 159        |
| <b>第 8 章 图</b> .....                                         | <b>163</b> |
| 8.1 前言.....                                                  | 164        |
| 8.2 图的基本概念.....                                              | 164        |
| 8.3 图的存储结构.....                                              | 168        |
| 8.3.1 邻接矩阵(Adjacency matrix) .....                           | 168        |
| 8.3.2 邻接表(adjacency list) .....                              | 170        |
| 8.3.3 邻接多重表(Adjacency multilist) .....                       | 171        |
| 8.3.4 索引表(Indexed Table) .....                               | 172        |
| 8.4 图的遍历(Graph Traversal) .....                              | 173        |
| 8.5 生成树(Spanning Tree) .....                                 | 174        |
| 8.6 拓扑排序(Topological Sorting) .....                          | 179        |
| 8.7 最短路径.....                                                | 182        |
| 习题 .....                                                     | 188        |
| <b>第 9 章 排 序</b> .....                                       | <b>191</b> |
| 9.1 前言.....                                                  | 192        |
| 9.2 内部排序法.....                                               | 193        |
| 9.2.1 冒泡排序法(Bubble Sort) .....                               | 193        |
| 9.2.2 线性选择排序法(Linear Selection Sort) .....                   | 195        |
| 9.2.3 交换-线性选择排序法 (Linear Selection With Exchange Sort) ..... | 197        |
| 9.2.4 二次选择排序法 (Quadratic Selection Sort) .....               | 198        |
| 9.2.5 中心插入排序法 (Centered Insertion Sort) .....                | 200        |
| 9.2.6 折半插入排序法 (Binary Insertion Sort) .....                  | 203        |
| 9.2.7 快速排序法 (Quick Sort) .....                               | 205        |
| 9.2.8 希尔排序法 (Shell Sort) .....                               | 208        |
| 9.2.9 归并排序法(Merge Sort) .....                                | 211        |
| 9.2.10 堆排序法(Heap sort) .....                                 | 216        |
| 9.2.11 二叉树排序法 (Binary Tree Sort) .....                       | 221        |
| 9.2.12 计数排序法(Counting Sort) .....                            | 225        |
| 9.2.13 基数排序法(Radix Sort) .....                               | 227        |
| 9.3 外部排序法.....                                               | 230        |
| 9.3.1 直接归并排序法 (Direct Merge Sort) .....                      | 230        |
| 9.3.2 自然归并排序法 (Natural Merge Sort) .....                     | 230        |
| 9.3.3 k 路归并法(k-Way Merge Sort) .....                         | 232        |
| 9.3.4 多段归并法(Polyphase Merge) .....                           | 234        |

|                                          |            |
|------------------------------------------|------------|
| 9.4 排序法的效益评估.....                        | 236        |
| 习题.....                                  | 237        |
| <b>第 10 章 查 找.....</b>                   | <b>239</b> |
| 10.1 前言.....                             | 240        |
| 10.2 顺序查找法 (Sequential Search) .....     | 240        |
| 10.3 折半查找法 (Binary Search) .....         | 242        |
| 10.4 杂凑查找法(Hashing) .....                | 243        |
| 10.4.1 直接定址法(direct addressing) .....    | 244        |
| 10.4.2 抽取法(extraction) .....             | 244        |
| 10.4.3 除法(division method) .....         | 245        |
| 10.4.4 乘法( multiplicative method ) ..... | 245        |
| 10.4.5 中段平方法( midsquare method ) .....   | 246        |
| 10.4.6 折叠法(folding method) .....         | 246        |
| 10.4.7 解决杂凑冲突的方法.....                    | 246        |
| 10.4.7.1 开放地址法(open addressing) .....    | 246        |
| 10.4.7.2 双重杂凑法(double hashing) .....     | 248        |
| 10.4.7.3 分开链接法(separate chaining) .....  | 249        |
| 10.4.8 从杂凑表删除项目.....                     | 249        |
| 10.4.9 杂凑法的评估.....                       | 250        |
| 10.5 树状查找法.....                          | 250        |
| 10.5.1 折半查找树(Binary Search Tree) .....   | 250        |
| 10.5.2 B-Tree 查找法 (B-Tree Search) .....  | 251        |
| 10.6 斐波那齐查找法 (Fibonacci Search) .....    | 252        |
| 习题.....                                  | 256        |



# 1

## 数据结构概论

---

- 1.1 数据与信息
- 1.2 数据处理(Data Processing)
- 1.3 计算机任务处理的方式
- 1.4 程序的产生
- 1.5 程序的分析
- 1.6 算法
- 1.7 复杂度(Complexity)
- 1.8 NP-COMPLETE 问题
- 1.9 参数的传递
- 1.10 数据结构(Data Structure)



## 1.1 数据与信息

人类通过各种活动获取所需的物质与知识，这些活动内容的详细记录就称数据(Data)。数据也可解释为用来说明人类活动的事实观念或事物的一些文字、数字或符号。例如学生的学科成绩，飞机的班次，较复杂的如定单中的客户姓名、地址、物品名称、数量、金额等。数据的类型可分成数值数据及文字数据两大类。数值数据有整数、定点数、浮点数等，文字数据有逻辑数据、内码和交换码等。

### 数据类型(Data Type)

程序语言中变量所表示的数据种类有多种，如 Pascal 的 Integer、Real 和 Pointer，SNOBOL 中数据类型是 character、string，LISP 的(List)，及 C 语言中的 int（整数）、float（浮点数）、char（字符）、\*Pointer（指针）等。

### 数据的级别

数据的级别由低而高依次为位、字节、字、数据项、数据字段、记录、文件、数据库。

1. 位(Bit): Bit 为 Binary Digit 的简称，是构成数据的最小单位，只能存储 0 或 1 的二进制数字，也是计算机的最基本单位。
2. 字节(Byte): 计算机容量的单位，通常 8 位构成一个字节。一个字节可存一个字符。可以有 2256 种组合，即可表示 256 个不同的字符。
3. 字(Word): 汉字由两个字节所构成。
4. 数据项(Item): 单项数据通常由一个或多个字节构成，如学生基本数据中的学号、姓名等。
5. 数据字段(Field): 由一个或多个数据字段组合而成，是一个具有意义的数据单位，如学生基本数据中的成绩字段、学号字段等。
6. 记录(Record): 由一群相关的数据字段组合，如学生的所有相关数据则构成一记录，有时可简称为数据记录。
7. 文件(File): 一群数据类型相同或相关的记录组合而成。
8. 数据库(Data Base): 由相关而且避免重复的数据组成的集合。

信息(Information)是指对某一特定的目的而言，具有意义的事实与知识，是由源数据经过有系统的处理成为决策或参考的依据。信息的定义有多种，这些定义分别从不同角度描述了信息的性质。其中最基本的三种定义如下：

- 信息是经过记录、分类、组织、关联与解释的数据，且就某一论点而言，具有意义。
- 在人类沟通中，信息是人类转换数据后的产物。
- 信息是人类在形成决策时，能导致个人改变其期望或评估的刺激。

综上所述，数据只是事实的记录，没有特定的目的；而信息是针对某一问题来收集数据及进行处理，作为决策或参考的依据。所以两者的差别在于是否由决策问题来鉴定信息。而信息处理的目的，是降低不确定因素，以便方案的选择。

## 1.2 数据处理(Data Processing)

数据处理是将数据通过人力或机器，将收集到的数据加以系统的处理，归纳出有价值的信息（参见图 1-1）。

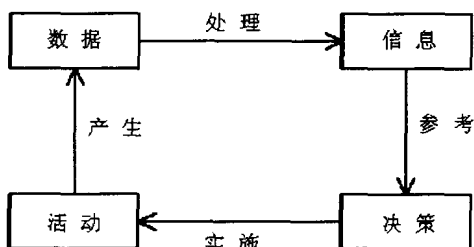


图 1-1

数据与信息两者具有密切关系。目前，人们常用计算机进行数据处理，以便获得信息，再通过信息作为未来活动的依据，而根据决策时所订的准则，进行所有的活动，从活动中又产生新的数据，如此不断的循环。一般来说，数据对人类并无直接的用处，因此，必须经过处理才能产生对人类较有用的信息。常用的处理方式有下列几种任务方式：

1. 编辑(Edit)：将存在某种媒体上的数据经过计算机复制到另一媒体时，对输入数据逐一检查，其目的在于改变数据的存储形式和效率，以便后面的处理，此种程序一般称为编辑程序。

例如：

- (1)用户可利用编辑程序将指令逐一编辑成程序。
- (2)将员工的人事数据登录在卡片上，再利用编辑程序将卡片上的人事数据编辑在磁带媒体上。

2. 排序(Sort)：将数据根据某一键值，以某种顺序排列后输出，其目的在于方便其他方面的数据处理。

例如：

- (1)学生成绩档案依照成绩高低列输出成绩单。
- (2)将业务员销售档案依照其销售金额的高低排列，以利于主管部门的参考。

3. 归并(Merge)：将两种以上相同性质的文件数据归并在一起。

例如：

- (1)客户的数据档案分别为北区和南区二个文件，将其归并成为全省的客户数据档案。
- (2)上半年度的销售数据和下半年度的销售数据归并成全年的销售数据。

4. 分配(Distribute)：将一个文件的数据按照某一基准分置在两个以上的存储体，其目的在于方便各个分置的文件能独自处理。

例如：

- (1)销售档案可分配为已收货款档案和未收款档案。
- (2)支票档案可分配为已到期支票档案和未到期支票档案，以利于资金的周转运用。

5. 建档(Generate)：建档是根据某些条件规格，配合某些已存在的文件，再产生一个新



的且有利用价值的文件。

例如：

(1)依据毕业生名单，从在校生文件产生一个毕业生文件。

(2)根据政府颁布的绩优厂商条件，从全国厂商名录档案中产生一个绩优厂商文件，以便于适当节日表扬。

6. 更新(Update)：更新是根据数据的变动来更新主档案，以保持主档案的正确与完整性。

例如：

(1)库存主档案由于常有货物的进出，因此每隔一段时间，必须对原库存主档案做进一步的更新处理，以利于库存的管理。

(2)全国汽车牌照数据档案，由于汽车常会异主或车主更换住址或翻新车身颜色，因此数据档案就须常做更新以保有最正确的数据。

7. 计算(Compute)：将读取的文件数据，依据规定方法计算处理。

例如：

(1)读取学生的答案卡，评阅并计算得分后输出。

(2)读取外务员的业绩档案并依据其金额大小，依照公司订立的佣有金算法及薪水扣缴税方案计算所得佣有金额。

8. 链表(List)：是一种数据的集合，也就是一系列的数据存储于内存，以某种关系来链接这些相关连的数据。

例如：

(1)字符串是一种字符链表。

(2)文件是一种存储在外界媒体（譬如：磁盘、磁带）的链表。

9. 查找(Search)：输入一个键值到数据表中进行对照，找出具有相同键值的数据。其目的在于方便数据表的更新操作，因为数据的插入及删除在操作之前都须先做数据的查找。

例如：

(1)公司查找旧客户的数据以便做客户记录。

(2)学生查找调查某项数据人数。

10. 查询(Inquiry)：根据数据项的键值或条件，到主档案中找出符合该条件或键值相同的数据，依照用户指定的方法输出。

例如：

(1)航空公司查询某旅客是否搭乘某班次的飞机。

(2)学生查询成绩。

11. 其他处理如：分类(Classifying)，摘要(Summarizing)，变换(Transmission)等。

适用于电子数据处理任务的特性如下：

1. 须快速处理的任务。
2. 反复处理的任务。
3. 数据量大的任务。
4. 需精确度高的任务。
5. 需保密性及安全性的任务。

## 1.3 计算机任务处理的方式

以处理任务的时间或周期分类:

1. Batch Processing System (批处理系统)

定期收集数据整批按时处理的任务方式, 如: 日报、周报及月报任务。

2. On-Line Real Time System (联机实时处理系统)

利用连接通讯系统, 将数据由终端机输入, 并由中央处理机将输出结果传送到所需要的地方。

3. Remote Batch Processing System (远程批处理系统)

终端机设备并不与主计算机连接, 亦即数据先存储在辅助内存中, 而不直接传输到主计算机处理。

以任务使用计算机系统 (CPU 和 Memory) 的时间或机器的分配类型分类:

1. Time Sharing System (分时操作系统)

许多用户分别利用终端机, 几乎同时登录到一个分时操作系统, CPU 以时间片轮转的方式轮流处理程序, 即每个程序都能分配一段处理时间, 当分配的该段时间结束, CPU 便转移给下一个程序使用。

2. MultiPrograming System (多道程序系统)

为节省 CPU 时间, 允许多个程序同时存于内存中, 当执行中的程序在处理 I/O 工作时, 便将 CPU 时间给下一个程序使用。

3. MultiProcessing System (多处理器系统)

由两个以上的 CPU 结合而成的处理系统, 任务方式可分为同时任务、单独任务或并行任务。

4. Distributed Data Processing System (分布式数据处理)

利用网络将多部计算机及外设连接起来, 而每部计算机均可存储自己的文件, 开发自己的系统, 自行处理, 各计算机通过网络来完成数据分布及资源共享的目的。

## 1.4 程序的产生

程序的产生分成五个阶段:

1. 需求(Requirements): 充分了解所提供的信息 (输入), 以及将要产生的结果 (输出), 对所有输入及输出进行严密的描述。
2. 设计(Design): 根据需求所得的数据集 (如迷宫, 多项式, 或者是姓名的字符串等), 编写算法解决问题; 算法不必拘泥于形式, 可以用所熟悉的伪语言、图或文字来表示即可。
3. 分析(Analysis): 针对问题提出其他解决方案的; 最后, 在所有的算法中挑选最佳者。
4. 细化与编码(Refinement and Coding): 细化、修改所选择的算法, 配合使用的程序语言特性, 编写出程序的初稿。
5. 验证(Verification): 验证的工作包含程序验证(Program Proving), 测试(Testing)及调试(Debugging)三部分。

## 1.5 程序的分析

程序是由一群指令组合而成，而如何将指令作适当的组合使其解决问题，是编写程序最起码的目的。就结构而言，一个程序是由数据结构(Data Structures)、程序结构(Program Structures)和控制结构(Control Structures)结合而成的。在设计程序时，最先要考虑的是数据结构，其次是程序结构，最后才是控制结构。结构化程序实际就是依据程序的设计技巧和控制方法形成的多级形式。所以，结构化程序设计方法包含下列三个技巧：

### 1. 自顶而下的程序设计方法(Top-Down Design):

- 将问题依据逻辑性质细分为数个单位，再将单位细分成独立的模块，亦即最小单位（参见图 1-2）。

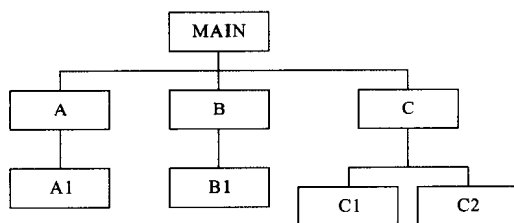


图 1-2

- 结构化程序设计采用自顶而下的程序设计法。

层次式的树状图，可帮助程序设计师在编写程序之前，了解整个程序结构，以及掌握模块与模块间的关联。

### 2. 模块化程序设计(Modular Decomposition Design):

- 将程序分解成若干模块，每个模块具有独立功能，可以独立编写、测试。
- 结构化程序设计采用了模拟化程序设计方法。
- 模块与模块之间是利用参数传递来联系。
- 可以共用某些模块。

#### 模块化设计

- 模块化是结构化系统设计核心，主要的目的是将一个复杂系统划分为一些具有特定功能的模块，使整个软件设计、测试、文档编写或维护任务更加简单。
- 在鉴定模块时必须考虑模块本身的特性，例如输入、输出、功能、程序逻辑即内部数据等，同时给每模块固定编号与名称，作为整个系统整合的依据。

### 3. 控制结构设计(Control Structure Design):

- 结构化程序设计是采用科学化、标准化的规定，所设计、编写、测试生成的程序。
- 结构化程序设计是一种自顶而下的程序设计方法，也是一种模块化的程序设计法。
- 自顶而下程序设计方法将程序细分成多个独立模块。
- 每一模块均可独立编写测试。

结构化程序设计(Structure Programming)的基本结构(Basic Structures):

#### 1. 顺序性结构(Sequence Construct)

顺序性结构中的任一处理方块(Process Box)都可视为单一指令或仅有一入口及一出口  
的指令序列（如子程序）。参见图 1-3。

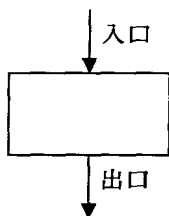


图 1-3

## 2. 选择性结构(Selection Construct)

选择性结构是使用有条件的控制指令执行，又称为 IF-THEN-ELSE 结构，以图 1-4 为例，假如 IF 条件成立则执行甲指令序列，否则执行乙指令序列，最后由一共同的出口离开。其他的选择性结构指令如 SWITCH，CASE 等都类似于 IF-THEN-ELSE 结构。

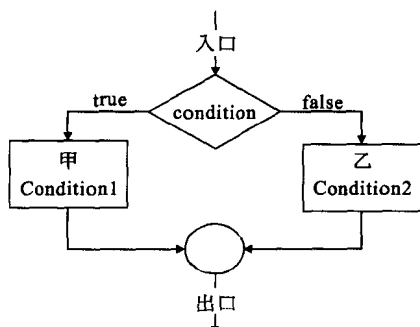


图 1-4

## 3. 迭代结构(Iteration Construct)

迭代结构也称递归结构、循环结构，主要用于程序循环的设计，即用于当指令被重复使用时的情况，迭代结构又称为 DO-WHILE 结构。从理论上说，此种结构有两种格式：WHILE 和 UNTIL。WHILE 执行的情形是先测试再进入循环，而 UNTIL 则先执行循环后再进行测试。其他循环结构指令如 FOR 是 WHILE 指令的应用。

(1)先处理，再判断是否要重复（即后测循环）：先执行循环体程序，再判断是否要继续执行，如符合条件，则重复执行；反之，如不符合条件，则由出口离开，进入下一阶段处理，使用指令如 UNTIL（参见图 1-5）。

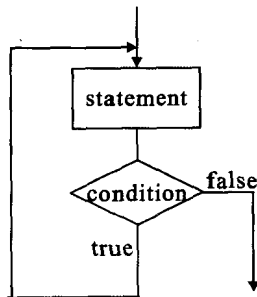


图 1-5

(2)先判断, 符合条件才处理 (即前测循环): 先判断是否符合条件, 如符合条件则执行循环体程序; 反之, 如不符合, 则立即离开, 进入下一阶段处理, 使用指令如 WHILE (参见图 1-6)。

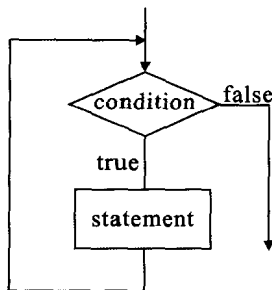


图 1-6

结构化程序设计的重要观念是这三种结构有相同的特性, 亦即是只有一个入口与一个出口。结构化程序设计的关键, 就是结合自顶而下的设计方法, 尽量避免使用 GOTO 指令, 但在实在无法避免时, 不应该使 GOTO 指令跳转过远而用其他较具结构性的控制指令来取代, 总之结构化程序的目的是增加程序的可读性和可维护性、增强程序的执行效率、减少测试问题及时间、增加程序设计师的生产力。

一般常用来评估程序的方法如下:

1. 是否符合所需?
2. 是否能根据原始规格工作?
3. 是否有描述使用方法和工作原理的文件说明?
4. 子程序是否以符合逻辑的方式建立?
5. 程序是否易于阅读?

一个好的程序须满足下列特征:

1. 有效性(Validity): 有效可分为内部有效(Internal Validity)和外部有效(External Validity)两种, 内部有效是指程序要能通过各项测试(Testing)。外部有效是指程序能完成规格书(Specification)所指定的各项功能。如个别程序测试(Program Testing)、相关程序测试(String Testing)与系统测试(System Testing)等。
2. 可读性(Readability): 程序中的变量名称应赋以有意义的名称, 且文件的说明必须详细。
3. 易改性(Modifiability): 在程序的生存周期内, 程序会随着用户或工作要求而不断改变, 因此, 要能满足不断改变的需求, 如输入输出的改变, 添加处理方式等。
4. 效率(Efficiency): 效率的考虑可分成两个部分: 人类和机械。一个程序员有效 (Programmer-efficient) 的程序, 则人类花的时间最少。但是一个机器有效 (Machines-efficient) 的程序, 则 CPU 时间和存储空间最少; 如何取舍因人而异, 原则上以程序员有效优先。一个易读及易修改的程序, 将会使程序易于维护及延长程序的寿命。而评估程序执行的效率须考虑执行程序的计算机、使用的机器语言指令集、执行指令的时间、编译程序的时间等。

由于机器的种类繁多, 每部计算机所使用执行指令的时间也不同, 因此不易评估程序的