信息技术

XINXI JISHU

程序设计入门

《信息技术》编写组 编

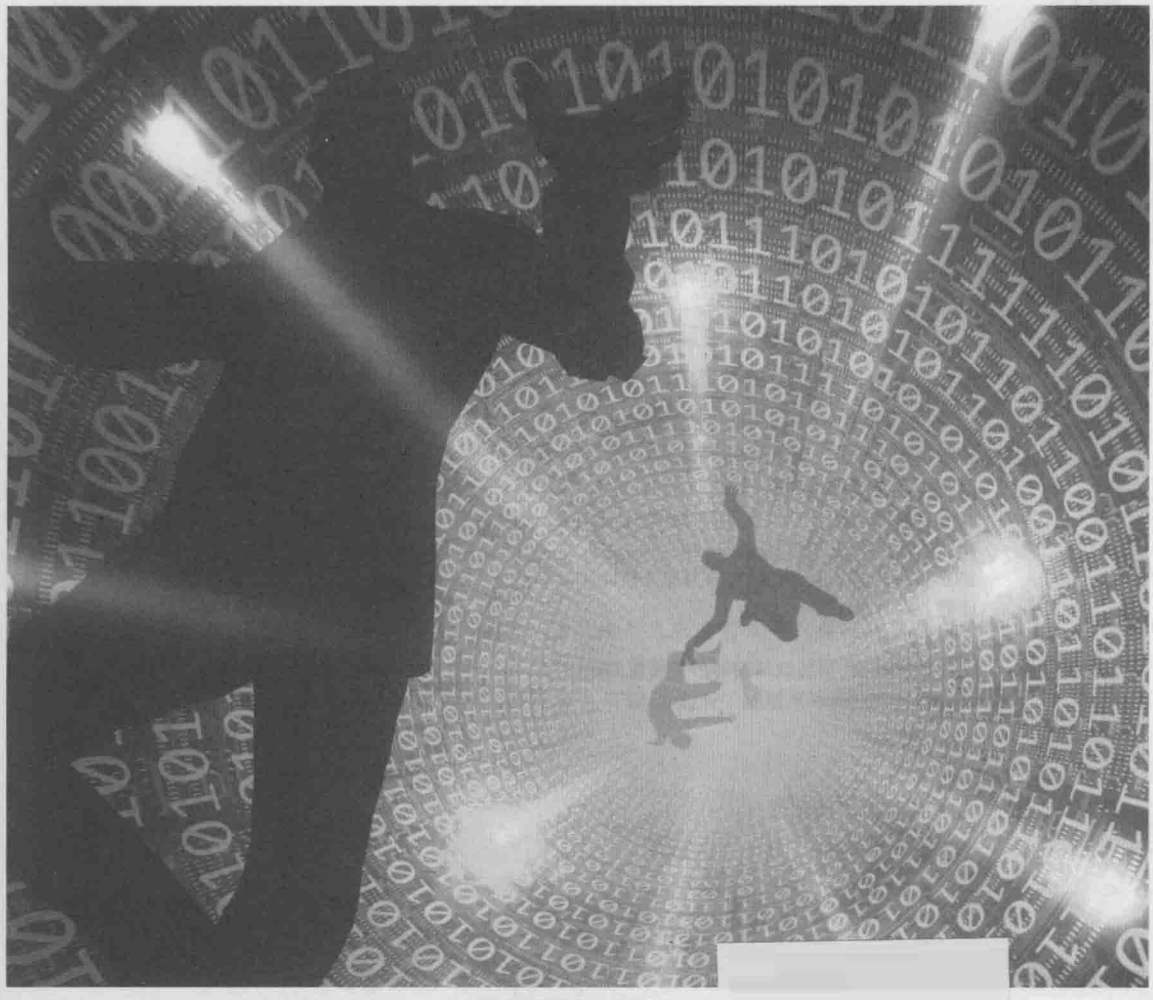
上海科学技术出版社

程序设计入门

信　息　技　术

程序设计入门

《信息技术》编写组　编



上海科学技术出版社

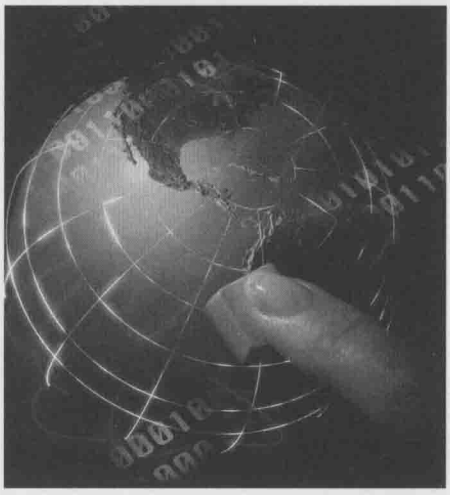
图书在版编目(CIP)数据

信息技术.程序设计入门/《信息技术》编写组编.
上海:上海科学技术出版社,2002.3
ISBN 7-5323-6163-2

Ⅰ.信… Ⅱ.信… Ⅲ.程序设计-基本知识
Ⅳ.TP311.1

中国版本图书馆 CIP 数据核字(2001)第 067929 号

上海科学技术出版社出版、发行
(上海瑞金二路 450 号 邮政编码 200020)
新华书店上海发行所经销
苏州市望电印刷厂印刷
开本 787×1092 1/16
印张 5.5 字数 120 000
2002 年 3 月第 1 版 2002 年 3 月第 1 次印刷
印数:1-3 500
定价:7.00 元
本书如有缺页、错装或坏损等严重质量问题,
请向本社出版科联系调换



出版说明

当今社会，现代信息技术已不仅在社会各个领域中得到广泛应用，而且正日益改变着人们的生产与生活方式、工作与学习方式。信息素养已成为现代社会成员必备的基本素质。信息技术教育已成为社会发展的必然要求。为此，我社组织编写这套《信息技术》教程。

本套教程充分体现教育部《中小学信息技术课程指导纲要(试行)》的精神和各项要求，强调培养学生获取信息、传递信息、处理信息和应用信息的方法和能力。本套教程按照《中小学信息技术课程指导纲要(试行)》小学、初中、高中不同阶段的要求，整合模块内容，设置为11个分册:《信息技术——基础篇(高中年级用)》、《信息技术——基础篇(初中年级用)》、《信息技术——基础篇(小学年级用)》、《信息技术——板报园地》、《信息技术——文字处理》、《信息技术——网络世界》、《信息技术——网页制作》、《信息技术——数据处理》、《信息技术——程序设计入门》、《信息技术——多媒体信息处理》、《信息技术——多媒体作品制作入门》。

本套教程内容新颖，反映了当前信息技术发展的基本内容和趋势，以“任务”驱动为主线，将信息技术教育融于任务教学之中。在任务中将现代信息技术知识与其他学科内容相结合，将现代信息技术知识与学校教学中的实际应用相结合，力求不仅使学生掌握信息技术知识，更重要的是使学生能够把握现代信息技术知识的共性和趋势，为学生终身学习打好基础。

参加本套教程编写的作者都长期从事信息技术教育的教学和科研工作，对信息技术教育有着丰富的实践经验并具有一定的理论研究水平。本书由叶彬蔚负责编写。

如今我们已跨入21世纪，信息技术发展日新月异，信息技术已成为最活跃、发展最迅速、影响最广泛的科学技术领域之一。编写这套教程对我们来说是一个新的尝试，因此难免有疏漏之处，敬请广大专家学者和师生指正。

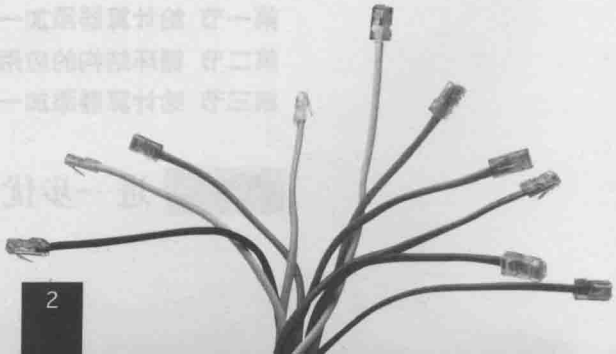
上海科学技术出版社

2002年1月

目录

XINXIJISHUCHENGXUSHEJIRUMEN

目录



第1章

用程序处理信息

- 程序和程序设计
- 程序设计的思想和方法
- 程序设计的语言和环境
- 熟悉一个可视化集成开发环境——Visual Basic
- 使用“帮助”学习 VB

第1章 用程序处理信息

本章将介绍程序和程序设计的概念，以及用程序处理问题的一般方法和完整的过程。无论一个程序如何复杂，它都是由一些基本算法构成的，因此还将介绍算法的概念，怎样用自然语言、流程图和程序来描述算法，以及算法的三种最基本的结构——顺序结构、分支结构和循环结构。

程序设计的思想和方法与计算机硬件一样，都在不断地发展。在其发展道路上，经历了两个里程碑——20世纪60年代的结构化程序设计和80年代的面向对象的程序设计思想，这两种程序设计都各有特点。

随着程序设计思想和方法的发展，编写程序的计算机语言和开发环境也在不断地发展——从最初的命令行编译，到如今“所见即所得”的可视化集成开发环境。

最后，向大家介绍本教材所采用的编程载体——Visual Basic（以下简称VB），并介绍如何使用VB自身所带的“帮助”来学习VB。

第一节 程序和程序设计

一、软件和程序

人们常说，硬件是计算机的基础，软件是计算机的灵魂。没有硬件，一切都无从谈起；但没有软件，计算机同样无法工作：若没有文字处理系统，就很难用计算机打印出一篇版面漂亮的文章；没有数据处理软件，就会被一大堆杂乱无序的数据弄得头昏脑胀。

所谓计算机软件，指的就是程序及其有关的文档资料。因此可以说，程序是软件的灵魂！

迄今为止，计算机的一切能力，都是人赋给它的。计算机所做的每一件事的每一步，都是由人事先替它设计好的。软件设计人员为使计算机能够完成某项任务，为它精心设计每一条“命令”——指令，并按照任务的要求把这些指令有序地排列起来，这就是程序(Program)。如果把程序和要操作的对象——数据一起存放到计算机的存储器中，到要使用时，再向计算机发出运行的命令，计算机便能按照事先编排好的“序”一条一条地执行这些命令，最终完成人们要求的任务。从表面上看，似乎是计算机在没有人指挥的情况下“自动”完成了某项任务，实际上还是人在控制一切。

二、试试用程序解决问题

计算机作为一种处理信息的工具，对于绝大多数人来说，只要能使用它就行了。

而编制程序是一项非常复杂的工作，需要深厚的计算机方面的专业知识，不可能人人都进行程序设计，因此有专门的软件公司和专业的程序设计人员来开发各种各样的软件。就好像人们都爱读小说、看电影，却并不是人人都必须成为小说家、电影导演。但有时，我们常会遇到用现成的软件解决不了或很难解决的问题，那该怎么办呢？下面举一个例子。

例1　有一篇英文文章，需要对其进行加密处理：每个字符都变成它的后续字母，即“A”变成“B”，“B”变成“C”……

解决办法：

对于文字处理，Word或WPS等软件有极为强大的功能，但如果要通过使用Word或WPS现成的查找、替换功能来完成这项特殊任务，显然是比较困难的。因此，我们不妨自己动手编写一段程序来解决这个问题。程序的设计思路是这样的：

(1) 把这篇文章看作一个字符串，并设计一个指针用来指向第一个字符；

(2) 把指针所指的字符的ASCII码增1，指针指向下一个字符；

(3) 反复执行(2)，直到文章结束。

这样就可以完成这个任务了，用程序解决问题是不是显得简单了？

由此看来，如果只会使用现成的软件，那还只是处在用计算机处理信息的“必然王国”之中，因为你不得不受到这些软件的功能的限制，而当你找不到任何软件来处理某些信息时，你对这些信息就束手无策了。只有自己设计程序，才可随心所欲地对各种各样的信息作不同的处理，真正进入处理信息的“自由王国”。

第二节　程序设计的思想和方法

一、用程序处理信息的全过程

用电子计算机来解决问题（即编写程序）的过程一般分为四个阶段。

1. 需求分析

所谓“需求分析”，就是对“要解决什么问题”毫不含糊地、尽可能详细地进行剖析：用户需要输入些什么，又需要输出些什么等。这是确定要求计算机去“做什么？”的问题。

2. 设计算法、数据结构和界面

所谓算法（Algorithm），就是解决某一问题的、确定的、有限的步骤。算法是使用计算机解决问题的基础，对一个具体的问题，只有事先设计好完整的、严密的、正确的算法，才能使计算机按照人们的意愿去迅速地完成预定的任务。

在日常生活中，人们处理一个问题，总是先考虑好处理这个问题的各个步骤，然后一步一步地去实现。如果遇到意外情况或与原先考虑不符的情况，则人脑会根据实际情况对考虑好的步骤进行修正或改变，以保证要处理的问题得以顺利解决。

要计算机处理一个问题，也必须如人处理问题一样，事先考虑好能为计算机接受和操作的处理步骤。但就目前来说，计算机还不具备像人脑那样能对变化了的情况作自动调整或修改的能力，只会按部就班地按人们事先告诉它的步骤去执行。当遇到与设想不符的新情况时，就会不知所措，或进行错误的操作，或干脆停机，从而达不到解决问题的目的。因此，使用计算机处理或解决问题，事先考虑好解决问题的各个具体步骤就显得十分重要。

解决同一问题，往往可以设计出不同的求解步骤。这就是说，一个问题的算法往往不止一种。应该通过分析比较，挑选最“合适”的一种算法，即寻找“优解”。

例1中所说的“程序的设计思路”其实就是算法，它表述了解决文本加密这个问题的步骤。当然，也可以有其他的办法来解决同样的问题。

由于计算机的操作主要是对各种类型的数据进行处理，因此，程序中还必须对操作的对象——数据进行合理的安排，相互间存在一种或多种特定关系的数据元素的集合就是数据结构（Data Structure）。

程序设计出来以后是被大家使用的，因此程序设计者必须考虑“友好的”用户界面，以方便程序的使用。

3. 编写程序

计算机只能接受并执行用计算机程序设计语言编写的程序。当我们为解决某个实际问题安排好数据结构、确定了算法并精心设计了用户界面之后，还必须用计算机程序设计语言，把求解问题的每一步骤依次“翻译”成计算机程序，这个过程称为“编码”或“编程”。顾名思义，“编码”就是把设计好的数据结构与算法转换成计算机能够识别的代码。

4. 上机调试与维护

刚编写好的程序，并不一定完全符合客观实际，还必须在计算机上运行这个程序，排除编程中的技术错误，测试其能否达到预期的结果。这个过程就是上机调试。

经过调试的程序，在使用一段时间后，仍会被发现尚有错误或不足之处有待改进，或需要增、删某些功能，使之更趋完善。这项工作就是维护程序，也称为软件维护。

二、算法和算法的描述

为解决每一个具体问题而设计的算法，必须用适当的方法把它描述出来。算法的描述必须具有清晰、明了、易懂、易改，以及易为计算机接受、理解和实现的特点。描述算法的方法很多，下面介绍常用的几种。

1. 用自然语言描述算法

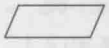
用自然语言描述算法，就是把算法的各个步骤，依次用人们所熟悉的日常会话的语言表示出来。例1中有关文字“加密”的算法，就是用自然语言描述的。

用自然语言描述算法，比较容易理解，但书写繁琐，而且往往会由于语言含义的不够确切而引起歧义，即对同一句话可以有不止一种理解；而对比较复杂的问题，又难以表述准确。世界上人类的语言并不统一，不同语言文字描述的算法各不相同，因此，用自然语言描述算法一般较少被采用。

2. 用图形描述算法

用图形描述算法，就是用含义确切的图形符号描述算法。用来描述算法的图形有多种，这里只介绍其中的一种——流程图。流程图是一种用规定的符号、连线和文字说明来表示算法的图形。用流程图描述算法比较直观、简便、明了，并且容易变换成计算机易于接受的形式。因此，在用计算机解决问题时，一般多用流程图来描述算法。

为了大家都能明确地读懂和理解流程图，必须对流程图中所使用的图形符号作出规定。我国对流程图的各种基本图形符号作了明确的统一规定，符号形状及其含义如下。

图形符号	名　称	说　　明
	起止框	表示一个算法的开始或结束
	输入、输出框	框内必须标明输入、输出的内容
	处理框	框内必须标明所进行的处理
	判别框	框内标明判别条件，并要在框外标明条件成立和条件不成立时的两种不同流向
	流程线	表示从某一框到另一框的流向
	连接框	表示算法流向的出口连接点或入口连接点

下面，就用流程图来分别表示一些算法。

例2　根据三角形的边长，计算三角形的面积。流程图如图1-1所示。

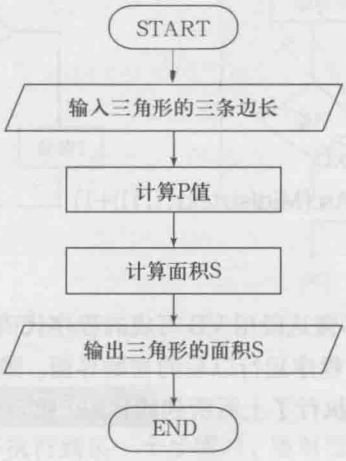


图　1-1

例3　打印出两个数中较大的数。流程图如图1-2所示。

例4　求出10个数的和。流程图如图1-3所示。

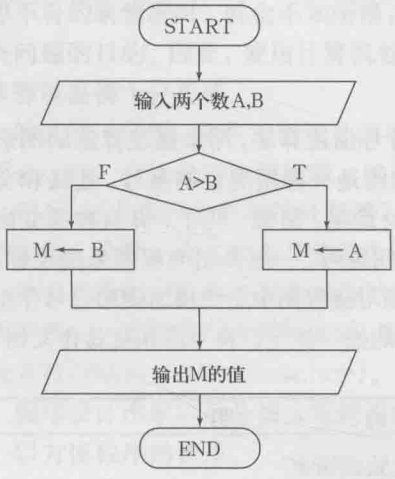


图　1-2

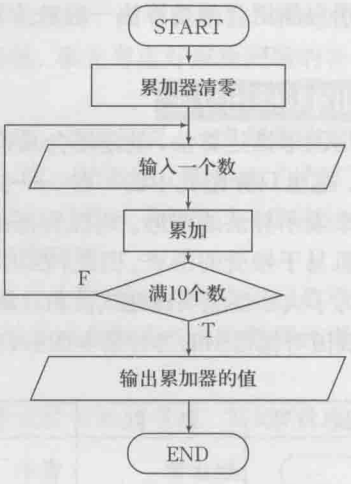


图　1-3

3. 用程序描述算法

任何一个用计算机解决问题的算法，最终都必须以计算机能够接受、理解和执行的程序描述出来。把程序输入计算机并运行，计算机才能按照预定的算法解决问题，因此程序必须以计算机能够接受的指令或语言来编写。不同的计算机能够识别的指令和语言不尽相同，即使对同一种计算机语言，不同机型对该语言的翻译程序也有差异。因此，用程序对算法的描述也有不同。用程序描述算法必须按照所使用的计算机的具体规定来编写。在本书中，选用的是目前较流行的编程语言Visual Basic。现在把有关文字“加密”的算法用VB描述如下：

```
Dim strtext As String
Dim i As Integer

strtext=Text1.Text
For i=1 To Len(strtext)
Mid(strtext,i,1)=Chr(Asc(Mid(strtext,i,1))+1)
Next
Text1.Text=strtext
```

当然，现在我们还读不懂这段用VB写成的程序代码，但可以看出它并不复杂，只有短短的几行。图1-4为程序运行以后的初始界面。图1-5为单击了“加密”按钮以后的情况，这时，程序已执行了上面所列的代码，窗口中的文字已被加密，变得谁也认不出来了。

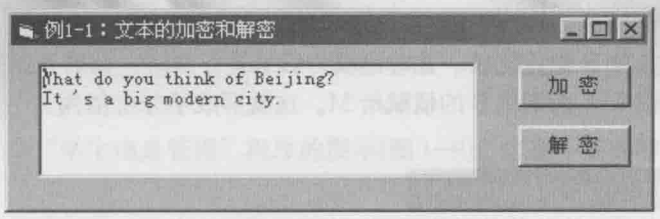


图 1-4

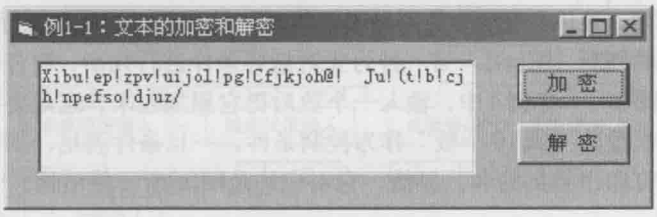


图 1-5

三、算法的基本结构

人们需要解决的问题各种各样，为解决问题而设计的算法也就千变万化。但是，不论怎样复杂的算法，它一般都由三种基本结构组成：顺序结构、分支结构和循环结构。

1. 顺序结构 (Sequence Structure)

在这种结构中，算法的各个步骤是按规定的先后次序顺序执行的，每个步骤都有一个确定的前趋步骤和一个确定的后继步骤(图 1-6)。例 2 所描述的就是一个典型的顺序结构。

图 1-6

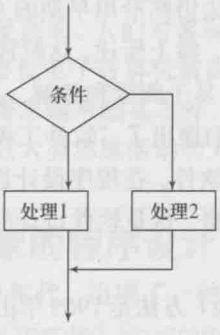


图 1-7

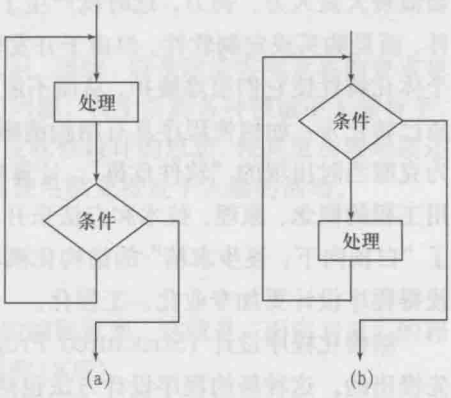


图 1-8

2. 分支结构 (Selection Structure)

在分支结构中，当程序执行到某一个步骤时，要对当时的处理结果进行判断，并

根据判断的不同结论去执行不同的后继步骤(图1-7)。在例3中，当输入两个数A和B以后，要对A和B进行比较，并对比较的结果进行判断：如果A＞B为真，就把A的值赋给变量M；否则把B的值赋给M。这就形成了分支结构。

3. 循环结构(Loop Structure)

在循环结构中，算法的某些步骤需要循环、反复地执行，反复执行的次数由某些条件控制，根据控制条件在执行过程中的具体结果，决定是继续循环执行还是结束循环执行后继步骤。图1-8所示的是两种不同的循环结构：一种是先执行循环体，然后进行循环条件的判断［图(a)］；另一种为先对循环条件进行判断，符合循环的条件才进入循环体［图(b)］。在例4中，输入一个数后把它累加起来，这是要循环执行的部分，而执行的次数由“满10个数”作为控制条件，一旦条件满足，则结束循环执行后继步骤，即打印出累加的和。显然，它对应的是图(a)所示的结构。

四、结构化的程序设计

从世界上出现第一台电子计算机到20世纪60年代中期，是计算机系统发展的早期时代。由于当时硬件设备价格昂贵、主机内存小、运算速度低，因此要求程序运行时间尽可能短，占用内存尽可能少。为了达到这一目的，人们施展各种技巧，以致程序难以读懂，也难以维护，程序设计没有统一的规范，基本上没有什么系统化的软件开发方法。

从20世纪60年代到70年代是计算机系统发展的第二代时期。随着CPU速度的迅速提高和存储器的飞速扩容，运行时间短和节省内存已经不是编程者追求的主要目标。而计算机应用的不断普及和深化，所需软件的规模往往相当庞大，以致单个用户无法开发。此外，许多不同的部门和企业往往需要相同或类似的软件，各自开发显然会浪费大量人力、物力，这时就产生了一些“软件作坊”。许多用户不再自己开发软件，而是购买或定制软件。但由于开发软件的方法仍然沿用早期的方法，许多软件的个体化特性使它们很难维护，从而不能正常运行。综上所述，这时程序设计的主要矛盾已转化为：如何使程序具有结构清晰的层次、易于阅读和理解、易于修改和验证。为克服当时出现的“软件危机”，计算机科学家们提出了“软件工程”这一概念，采用工程的概念、原理、技术和方法来开发和维护软件。在程序设计思想上，逐步完善了“自顶向下，逐步求精”的结构化模块设计思想。这是软件设计的一个里程碑，它使得程序设计更加专业化、工程化。

结构化程序设计(Structured Programming)方法是1969年由一些荷兰学者首先提出的。这种新的程序设计方法包括三大内容：

(1) 程序由一些基本结构组成。这些基本结构包括：顺序结构、分支结构和循环结构。

(2) 一个大型程序应按其功能分解成若干个功能模块，并把这些模块按层次关系进行组装。

(3) 在程序设计时，采用“自顶向下、逐步求精”的实施方法。

结构化程序设计不仅解决了当时的“软件危机”，而且使得计算机的发展更为迅速。

下面是一个“学生信息管理”程序的图示(图1-9)，这是一个典型的结构化程序设计的例子。

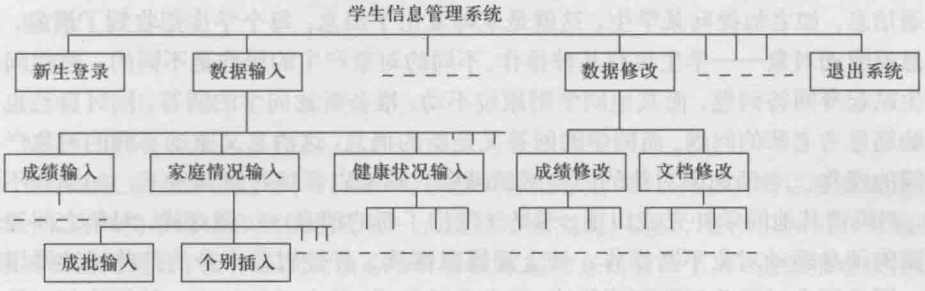


图 1-9

但是，当时程序设计的基本思想都是通过一系列指令序列来处理数据的，数据被动地由程序来处理。对程序设计者来说，他往往围绕过程来考虑数据即操作对象：程序什么时候开始执行，什么时候调用什么过程，在程序执行到什么时候输入数据，什么时候输出数据，什么时候程序结束……这样的程序设计思想被称为是过程控制式的或“面向过程”的。它的一大特点就是由过程来控制数据，过程和数据是分离的。

这种传统的程序设计思想在解决一些比较简单的、诸如DOS环境中的单任务操作问题时，还是非常有效的。迄今为止，在DOS环境下的一些程序设计语言，如Fortran、Pascal、甚至于C及数据库系统Foxbase、Foxpro等，还是采用这种面向过程的设计方法，有效地解决了各种应用问题。它们所提供的语句可以充分满足实现三种基本结构和模块化的需要，它们的函数和过程又为进行“自顶向下、逐步求精”的结构化模块设计提供了方便。

客观世界是复杂的，人们对客观世界的认识、理解，以及改造客观世界的要求也在不断地加深，要使电子计算机解决更复杂的问题，在生活的各种领域中大显身手，除了对它的硬件要求越来越高之外，还必须改变软件设计的思想，使其更合理地描述客观世界，更接近人类思维活动的方式，从而使电脑更接近于人脑的活动。

五、面向对象的程序设计

在20世纪80年代，出现了一种完全新颖的编程思想，这就是“面向对象”的程序设计（Object-Oriented Programming，简称OOP）。

面向对象的程序设计认为：我们所处的世界是由一组彼此相关并互通信息的实体即对象（Object）组成的。对象具有属性（Properties），也可以进行操作，即具有方法（Methods）。对象之间的通信产生消息（Message）。对象发出消息，消息又驱动其他对象进行操作，而这些操作又使某些对象的属性发生变化，从而完成某一任务。

面向对象的程序设计就是通过对象之间的消息通信，驱动对象执行一系列的操

作，从而完成某一任务的程序设计方法。

例5　用程序模拟学生上课的情景。

这样的程序当然可以用面向过程的方法来描述，但显然比较困难，因为这里的许多过程是交叉进行的，很难说清过程开始和结束的准确时间。但如果用面向对象的方法来描述，就比较容易了。

在教室里上课时，老师、众多学生、黑板、讲台、课桌等都为对象，彼此相关且互通信息。如老师提问某学生，这就是老师发出了消息。每个学生都收到了消息，由消息而驱动对象——学生执行某种操作。不同的对象产生的操作是不同的：被提问的学生站起身回答问题，而其他同学则原位不动，准备听此同学的回答，同时自己也在动脑筋思考老师的问题。而同学的回答又是新的消息，这消息又驱动不同的对象产生不同的操作：老师如果对他的回答感到满意，则加以表扬并请其坐下；如感到不满意，则再请其他同学补充或纠正。于是又发出了新的消息……就这样，对象之间通信得到的消息驱动对象不断操作，使上课得以继续，并使对象本身的某些属性得以改变：同学原来对某些问题不清楚的，现在弄清楚了，整个系统处于一种新的相对稳定的状态，也就完成了某一任务（在这里就是完成了一堂课的教学任务）。

在面向对象的程序设计中，“对象”是用一种将数据和过程或函数合为一体的数据结构——类（Class）来构造的。一个对象是一个软件构造块，它包含了两方面的内容：一是数据，二是相关的操作。数据表征对象的属性或特性，而操作代码（函数或过程）用于响应消息，使对象进行某种操作，操作也称方法。

现在，可以用各种各样的“类”来构造出各种各样的“对象”：如在Windows环境中经常看到的窗口、各种按钮、列表框、各种对话框……而一旦“对象”接受到某种消息，就使它的某些成员函数受到了调用，于是“对象”自身可完成一系列操作，从而改变“对象”。如：用鼠标拖动窗口的边框，窗口接受到“鼠标拖动边框”这个消息，于是它的改变大小的成员函数受到调用，进而完成某些操作，使大小得以改变。

Windows下的程序设计语言，如Borland C++、Visual C++、Visual Basic及高版本的Pascal（Delphi）等都是面向对象的程序设计语言。

第三节　程序设计的语言和环境

一、程序设计语言的发展

随着程序设计思想的不断发展，程序设计的语言和环境也在不断地发展。

最初的数字电子计算机是直接用二进制代码编制程序的——即用“1”和“0”组成机器语言来编写程序。例如，我们要实现00010000和00010001两个单元中的数的相加，最后结果存放在00010001单元中，在个人计算机中常用的指令代码序列按字节排列为：

10011100 00000000 00000001 10001010 00000100 01000110 00010000 00000100 10001000 00000100