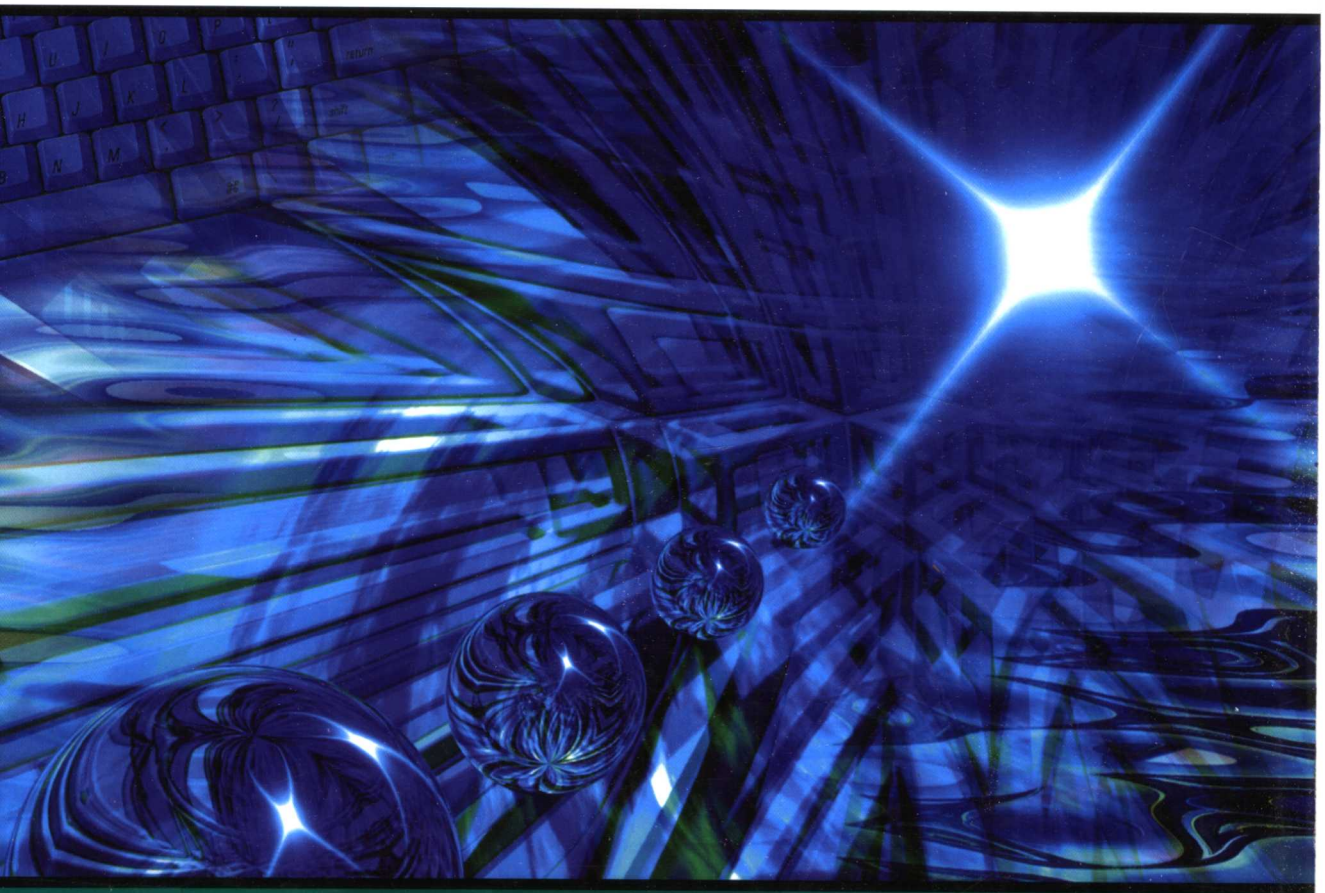
JISUANJI CHENGXU SHEJI JICHU JIAOCHENG 刘 莹 车光宏 等 编著

计算机程序设计基础教程



上海交通大学出版社

计算机程序设计基础教程

刘 莹 车光宏 等编著

上海交通大学出版社

图书在版编目(CIP)数据

计算机程序设计基础教程/刘莹，车光宏等编著. —上海：上海交通大学出版社，2005(2006 重印)

ISBN 7-313-04137-3

Ⅰ.计… Ⅱ.① 刘… ② 车… Ⅲ.程序设计-高等学校-教材 Ⅳ.TP311

中国版本图书馆 CIP 数据核字(2005)第 097947 号

计算机程序设计基础教程

刘 莹 车光宏 等编著

上海交通大学出版社出版发行

(上海市番禺路 877 号 邮政编码 200030)

电话：64071208 出版人：张天蔚

立信会计出版社常熟市印刷联营厂印刷 全国新华书店经销

开本：787mm×1092mm 1/16 印张：16.75 字数：414 千字

2005 年 8 月第 1 版 2006 年 2 月第 2 次印刷

印数：4 251～7 750

ISBN 7-313-04137-3/TP・632 定价：25.00 元

前　言

本教材的主要教学对象是非计算机专业的大、专院校学生。在学习计算机程序设计课程前，假定这些同学已经对计算机基础知识有了一定程度的了解和掌握。对于需要利用计算机进行信息处理的同学来说，应该进一步学习和掌握计算机程序设计的知识和技能，以便继续向更高的管理和应用人才层次发展。

计算机程序设计类的教材种类繁多，但大多数教材以讲某种高级语言为主题，而程序设计中精髓——算法被淡化了，或者仅作为语言的附属在教材中讲述。尽管语言是用以实现程序设计为目的的不可或缺的工具，但处理问题的方法和操作才是使程序具有活力的关键。在解决各种各样实际问题的过程中，人们已经总结、归纳出了许多经典的、有效的算法，这些算法几乎可以用目前现有的各种计算机语言来实现。计算机语言是不断地发展和变化的，任何人也不可能学完并熟练掌握所有的计算机语言，相反，只要他学习、掌握了一定的算法，就可以利用自己所熟悉的语言来设计解决问题的程序。因此，我们这本教材将以介绍算法、强调程序设计方法为主线，以 VB 为工具，培养学生的程序设计能力。希望能真正地掌握程序设计的方法，而不受缚于某种计算机语言。

本教材的编写过程中，强调并遵循了注重实用、准确严谨、朴实无华的原则，同时，又注意满足各专业培养目标中提到的高级专门人才所需计算机相关知识的要求。参与本教材编写的人员均具有多年计算机专业课程的教学经验，同时又担任非计算机专业相关计算机课程的教学工作，并从事应用软件设计，有着扎实的理论基础、丰富的设计技巧，并对本教材的教学对象应该掌握哪些程序设计知识有充分的了解。因此，本教材在概念描述上力求准确、简洁；在算法介绍上力求清晰、合理；在例题的设计上注意到其典型性和通用性；并且适当地给出了具有启发性的说明和提示。为配合本教材的实际教学，并另外编写了相应的实验指导材料。

本教材为集体编写，车光宏老师编写了第 1、第 2 章以及实验指导；何宗林老师编写了第 3、第 4、第 5、第 6 章；张林老师编写了第 7 章；张海老师编写了第 8、第 9 章；包怀忠老师编写了第 10、第 11 章；王淮生老师对本书的文字进行了审改；刘莹老师负责总撰。在编写过程中李烈敏老师、马季老师为本书的知识结构给出了指导性建议；教务处及计算机系其他教师对教材内容提出了具体的意见和要求，在此表示感谢。

尽管编写中作者为保证内容的合理、正确作了不少努力，错误和纰漏可能难免，欢迎读者批评指正，并将虚心接受读者的意见和建议。（E-mail：acjsjly@aufe.edu.cn）

编者

2005 年 7 月

目　录

第 1 章　程序设计概论 …… 1

1.1　程序设计的基本概念 …… 1
1.1.1　程序与程序设计 …… 1
1.1.2　程序设计的过程 …… 2
1.1.3　程序的运行方式 …… 3
1.1.4　程序设计的特点 …… 3
1.1.5　程序的质量标准 …… 4
1.2　程序设计语言 …… 6
1.2.1　程序设计语言的分类 …… 6
1.2.2　程序设计语言的发展 …… 8
1.2.3　常用程序设计语言简介 …… 9
1.3　程序设计方法 …… 10
1.3.1　结构化程序设计 …… 10
1.3.2　面向对象程序设计 …… 15
1.4　程序设计环境和实现步骤 …… 16
1.4.1　程序设计环境 …… 16
1.4.2　实现步骤 …… 17
习题 1 …… 18

第 2 章　算法及其描述工具 …… 19
2.1　算法的概念 …… 19
2.1.1　算法和算法设计 …… 19
2.1.2　算法的基本特性 …… 21
2.2　算法的描述工具 …… 23
2.2.1　程序流程图 …… 23
2.2.2　PAD 图 …… 24
2.2.3　N-S 图 …… 25
2.2.4　伪码 …… 26
2.3　算法设计的基本思想 …… 26
2.3.1　枚举 …… 26
2.3.2　递推 …… 27
2.3.3　递归 …… 30
2.3.4　迭代 …… 32

2.3.5 回溯 …… 33
2.4 算法设计举例 …… 34
习题 2 …… 41

第 3 章 数据表示及基本运算 …… 43
3.1 数据类型 …… 43
3.1.1 基本数据类型 …… 44
3.1.2 用户自定义类型 …… 45
3.2 常量和变量 …… 46
3.2.1 常量 …… 46
3.2.2 变量 …… 47
3.3 函数 …… 49
3.4 运算符和表达式 …… 51
3.4.1 算术表达式 …… 51
3.4.2 字符串表达式 …… 52
3.4.3 日期表达式 …… 52
3.4.4 关系表达式 …… 53
3.4.5 逻辑表达式 …… 53
3.4.6 混合表达式的执行顺序 …… 54
习题 3 …… 55

第 4 章 程序结构与基本语句 …… 56
4.1 概述 …… 56
4.1.1 程序的构成 …… 56
4.1.2 程序的控制结构 …… 57
4.1.3 顺序结构 …… 58
4.2 赋值语句 …… 58
4.3 数据输出语句 …… 59
4.3.1 Print 方法 …… 59
4.3.2 使用标签 …… 61
4.3.3 使用文本框 …… 62
4.4 数据输入语句 …… 62
4.4.1 使用文本框 …… 62
4.4.2 使用对话框 …… 64
4.5 其他语句 …… 67
4.5.1 注释语句 …… 67
4.5.2 结束语句(End) …… 67
4.6 程序设计举例 …… 68
习题 4 …… 71

第 5 章　分支与循环 …… 72
5.1　分支结构 …… 72
5.1.1　条件语句 …… 73
5.1.2　情况语句 …… 79
5.2　循环结构 …… 81
5.2.1　条件循环语句 …… 82
5.2.2　计数循环语句 …… 85
5.2.3　多重循环 …… 87
5.3　程序设计举例 …… 89
习题 5 …… 97

第 6 章　数组 …… 98
6.1　概述 …… 98
6.1.1　数组的定义 …… 99
6.1.2　静态数组和动态数组 …… 101
6.2　一维数组的基本操作 …… 102
6.2.1　一维数组的赋值和输出 …… 102
6.2.2　一维数组的复制 …… 103
6.2.3　最大值问题 …… 103
6.2.4　一维数组的查找 …… 104
6.2.5　排序 …… 108
6.3　二维数组的基本操作 …… 111
6.3.1　二维数组的赋值与输出 …… 112
6.3.2　二维数组的复制 …… 113
6.3.3　二维数组的最大值问题 …… 116
6.4　程序设计举例 …… 117
习题 6 …… 123

第 7 章　过程 …… 125
7.1　概述 …… 125
7.2　函数过程 …… 127
7.2.1　函数过程的定义格式 …… 127
7.2.2　函数过程的建立 …… 128
7.2.3　函数过程的调用 …… 128
7.3　子程序过程 …… 130
7.3.1　子程序过程的定义格式 …… 130
7.3.2　子程序过程的建立 …… 131
7.3.3　子程序过程的调用 …… 131
7.4　参数传送 …… 132
7.4.1　形式参数与实在参数 …… 133

7.4.2 值传递与地址传递 …… 133
7.4.3 数组参数 …… 136
7.4.4 缺省参数和可变参数 …… 137
7.5 过程的嵌套调用 …… 139
7.5.1 嵌套调用 …… 139
7.5.2 递归调用 …… 140
7.6 变量的作用域与生存期 …… 143
7.6.1 变量的作用域 …… 143
7.6.2 变量的生存期 …… 146
习题 7 …… 147

第 8 章 数据文件 …… 148
8.1 概述 …… 148
8.1.1 文件的基本概念 …… 148
8.1.2 文件的基本操作 …… 149
8.2 数据文件及其结构 …… 152
8.3 数据文件的基本操作 …… 154
8.3.1 顺序文件的基本操作 …… 154
8.3.2 随机文件的基本操作 …… 157
8.3.3 二进制文件的基本操作 …… 160
8.4 文件操作举例 …… 161
8.4.1 顺序文件操作举例 …… 161
8.4.2 随机文件操作举例 …… 164
8.4.3 二进制文件操作举例 …… 169
习题 8 …… 172

第 9 章 面向对象程序设计 …… 174
9.1 面向对象的基本概念 …… 174
9.1.1 对象与实例 …… 175
9.1.2 什么是消息 …… 176
9.1.3 类 …… 177
9.1.4 事件及事件处理 …… 178
9.1.5 面向对象 …… 178
9.2 面向对象方法的特点 …… 178
9.2.1 继承 …… 178
9.2.2 封装 …… 179
9.2.3 多态性 …… 180
9.3 面向对象方法的优点 …… 180
习题 9 …… 181

第 10 章　窗体与基本控件 …… 182
10.1　可视化程序设计概述 …… 182
10.2　Visual Basic 6.0 窗体对象 …… 183
10.2.1　窗体的属性 …… 184
10.2.2　窗体事件 …… 188
10.2.3　和窗体有关的语句与方法 …… 190
10.2.4　多窗体应用程序 …… 191
10.3　命令按钮、标签与文本框 …… 193
10.3.1　命令按钮 …… 193
10.3.2　标签 …… 195
10.3.3　文本框 …… 197
10.4　选择性控件 …… 199
10.4.1　复选框与单选按钮 …… 199
10.4.2　框架 …… 202
10.4.3　列表框 …… 203
10.4.4　组合框 …… 206
10.4.5　水平滚动条与垂直滚动条 …… 207
10.5　计时器 …… 209
10.6　文件系统控件 …… 211
10.6.1　驱动器列表框 …… 211
10.6.2　目录列表框 …… 212
10.6.3　文件列表框 …… 212
10.7　控件编程的高级控制 …… 214
10.7.1　控件数组 …… 214
10.7.2　Tab 顺序 …… 218
10.7.3　键盘和鼠标事件 …… 220
习题 10 …… 223

第 11 章　图形与界面设计 …… 225
11.1　图形控件和图形方法 …… 225
11.1.1　坐标系统 …… 225
11.1.2　绘图属性 …… 228
11.1.3　图形控件 …… 230
11.1.4　图形方法 …… 232
11.2　常见的 ActiveX 部件 …… 237
11.2.1　加载 ActiveX 控件 …… 237
11.2.2　公共对话框控件 …… 237
11.2.3　RichTextBox 控件 …… 244
11.3　Visual Basic 6.0 用户界面设计 …… 247

11.3.1 菜单设计…… 247
11.3.2 多文档界面…… 252

第1章　程序设计概论

本章概要：

在本章里将学习以下内容：

- 程序和程序设计的有关知识；
- 程序设计语言和程序设计方法方面的有关知识。

学习完本章后，将能够：

- 了解程序、程序设计的基本概念、基本内容、基本步骤；
- 了解程序设计语言方面的基本知识；
- 了解程序设计方法方面的基本知识；
- 懂得如何评价软件的质量。

要利用计算机完成某项工作，必须事先编制指挥计算机工作的程序，只有这样，计算机才能按照人的意志高速并且正确地工作，程序设计语言就是编写计算机程序所使用的语言。程序设计语言作为程序设计工具存在并被人们使用着，程序设计语言可以帮助人们描述、实现解决问题的想法和操作，但其本身是不会产生想法、不会自主决定操作的。程序设计是利用程序设计工具(程序设计语言)描述如何解决特定问题的过程，成果是程序。对于程序来说，设计者的意愿是程序的灵魂，一序列的命令只是程序的躯壳。

学习设计程序，首先应该了解程序及程序设计的有关概念、性质、特点，了解程序设计的过程并掌握一定的程序设计方法，了解和熟悉程序设计工具和环境，对程序设计有较全面的认识，学会从总体上把握和设计程序，避免由于过早地把注意力陷入程序中语句、参数等细节中去，从而导致只会模仿而不懂得设计。

在本章中，将对程序设计的基本概念、程序设计语言、程序设计方法、程序设计环境和实现步骤予以介绍，为进一步学习程序设计做好准备。

1.1　程序设计的基本概念

要学习设计程序，首先应该对程序和程序设计的有关概念、程序设计的过程、程序设计工作的性质和特点等有所了解。由于每位利用计算机程序设计语言进行程序设计的人，在对程序、程序设计以及相关知识和技术的理解和掌握上存在着许多差异，因而其程序设计的结果也存在着很大的差别，很难想像，连什么是程序、什么是程序设计等基本概念都不清楚的人能写出好的程序。本节将介绍一些基本而又重要的概念。

1.1.1　程序与程序设计

计算机和我们的日常生活息息相关。是工作时的好帮手；学习时的好师友；休闲娱乐时的好伙伴。由于计算机在各个方面的出色表现，使得很多人都以为计算机是个天才，但其实它并

不是!

从使用计算机解决实际问题的用户角度看,计算机确实是个天才,它无所不会、无所不能,动作敏捷、做事利落,而且干得漂亮。但是从程序员的角度去看,计算机不但不是天才,简直就是一个傻瓜!它做事的速度过人但智力低下,如果你让它每次加 1 地数数,它会老老实实地数。但是,如果你不让它停止,它就会这么一直数下去,不知自动停止。因为"数数"和"停止"是两个不同的操作,计算机所执行的每一个操作都需要程序员下达指令(由于某种原因进入"逻辑混乱状态"的例外),否则它就不知道下一步该做什么。可见,计算机所会的,仅仅是忠实、严格地执行程序员事先安排的指令而已。

为了解决某个实际问题而编排的指令序列称之为程序。程序运行时,计算机将严格按照程序中各个指令所指定的动作进行操作,从而逐步地完成预定的任务。例如,如果要让计算机从 1 加到 10000,那你得发出如下指令序列(程序):

"让 s=0";

"让 x=1";

"将 x 加到 s 上";

"将 x 加 1";

"如果 x 没有超过 10000,则回到第三步,否则,执行下一个命令";

"报告结果 s"。

计算机执行上述指令,就完成了累加的任务。编制让计算机完成某项任务要执行的指令序列的工作称为程序设计。

程序设计不是一件很容易的事。因为,程序员面对的计算机不但不是一点就明的天才,而是这么一个具有三岁幼儿般智力却有着惊人工作速度和能力的怪物。程序员的任务就是要通过编制指令序列来将这个怪物"调教"成一个"无所不会、无所不能"的天才般的工具。

所幸的是,随着计算机技术的不断发展,特别是程序设计语言的发展,计算机的"智力"也有一定程度的提高。就是说,现在的计算机能够听得懂较为复杂的指令了。比如,像上面那样让计算机计数求和的指令序列,现在用类似于下面的一条指令就行了:

"从 1 到 10000 求和"。

这样,程序员的工作将会轻松许多。

1.1.2 程序设计的过程

程序设计过程主要包括问题分析、确定算法和编码实现等三个步骤。

1. 问题分析

问题分析是程序设计的第一步。在进行程序设计之前,必须要对程序将要解决的问题进行深入分析,弄清楚程序应该完成哪些功能?有哪些输入数据?需要输出哪些信息?

2. 确定算法

弄清楚程序应该"做什么"之后,就需要考虑"怎么做"了。这时需要寻找解决问题的方法(算法),如果该问题有多种解法,还应该对各种解法进行分析比较,从中选择一个比较好的解法。

在寻找和确定算法的时候,可以用合适的方法和工具来记录和描述算法。

3. 编码实现

编码实现就是将用程序流程图(或其他工具)描述的算法转换成某种程序设计语言表示的程序,并且在计算机上的实际测试、运行得以通过。

一般说来,程序设计过程不会是一蹴而就的,上述三个步骤往往需要反复多次。图 1.1 较为准确地描述了这一过程。

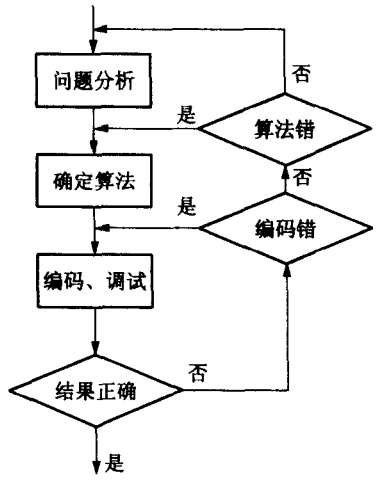


图 1.1　程序设计过程

1.1.3　程序的运行方式

按照运行方式,程序可以分为线性程序和事件驱动程序两种。

线性程序是按线性顺序(虽然局部可能出现跳跃和循环)执行的,即从程序的第一条指令开始执行,一刻不停地执行下去,直到最后一条指令为止。例如,DOS 环境中的程序基本上都是线性程序。

事件驱动程序的执行方式是,程序启动后不是顺序地往下执行,而是进入等待事件发生的状态,当某种事件发生时,系统就去执行相应的事件处理代码,执行完事件处理代码之后,再次进入等待事件发生的状态,……直到用户终止程序的执行为止。例如,Windows 环境中的程序都是事件驱动程序。

1.1.4　程序设计的特点

程序设计是一项非常具有挑战性的智力活动,与其他工作相比,它有一些独特的性质。

1. 构造性

程序员设计程序的工作与作家写文章、音乐家创作乐谱有很多相似的地方。作家通过调遣一个个简单的字词而创作出华丽的篇章;音乐家通过组织一个个简单的音符而谱写出优美动听的乐曲;程序员则通过编排一个个简单的操作而构造出能使计算机具有天才般本领的程序。这些工作的共同特点就是构造性。

构造性决定了程序设计结果(即程序)的多样性。因为只要程序能够正确地解决问题,怎么构造都行。因此,为了解决同一个问题,不同的程序员会设计出面貌迥异的程序来。甚至,同一个程序员,在不同的时期设计出的解决同一个问题的程序也会大相径庭。

构造性决定了不能用统一的标准来衡量程序的质量。为了表达同一个主题,一个作家写了一篇散文,而另一个作家却写了一首诗歌,显然不能用同一个标准来评价散文和诗歌。

构造性还决定了难以用形式化的方法来证明程序的正确性。由于对于解决同一个问题的程序存在着各种各样的构造方法,很难建立起像数学定理的证明和推导那样的形式化方法来证明程序的正确性。因此,程序的正确性只能通过测试来进行验证。然而,测试只能是有限的,不可能测试出程序可能隐含的所有问题,这就是目前大多数软件都不同程度地存在着错误的原因。

2. 严谨性

通常,讲演者偶然的口误和用词不当,往往不会对其表达的信息产生影响,因为听众有判断力,会在接受信息的过程中自动纠正讲演者的口误和用词不当。但是,和计算机打交道就不

同了，计算机虽然有一定的智力，但它毕竟不是人，它不会主动地对所接受的信息进行合理的修正。

计算机只会按照程序执行各指令所规定的操作，不会揣摩程序员的心思，也不会自动纠正程序员的笔误。因此，程序员编制的程序必须是非常严谨的，一点点的差错都有可能带来严重的后果。例如，在1963年美国用于控制火箭飞行的FORTRAN程序中，把一个指令“DO 5 I=1,3”误写为“DO 5 I=1.3”，在系统测试中又未能发现这个错误，结果这一“点”之差，导致飞往火星的火箭爆炸，造成1000万美元的损失。

程序严谨性的要求，决定了不能用自然语言来书写程序。因为任何一种自然语言都是不严格的，都会产生歧义。比如，“老赵对老钱说他的儿子考上大学了”，你能知道这句话的准确意思吗？如果没有其他补充信息，就不能知道到底是“老赵的儿子考上大学了”，还是“老钱的儿子考上大学了”。但如果这句话之前还有这样的话语“老赵的独生女去年大学毕业了……”，则你能立即知道，是“老钱的儿子考上大学了”。

编制程序所使用的程序设计语言是上下文无关的形式语言，而程序要解决的问题都是用自然语言陈述的。程序设计的任务就是，将用有歧义的自然语言描述的求解要求转换成用无歧义的形式语言表达的解题程序。这就需要程序员应具有认真的工作作风和严密的逻辑思维能力等基本素质。

3. 抽象性

现实世界中的事物，有着各种各样的特征，伴随着这些事物的存在和变化，大量的数据和信息不断地产生，为了能够用计算机来处理这些数据和信息，就必须对数据和信息进行抽象，因为计算机内部只能表示0和1组成的二进制代码。计算机中使用二进制代码来映射现实世界中的事物及其数据、信息。

现实世界中对事物进行的加工处理是千变万化的，就像我们眼中的世界一样绚丽多彩。而绚丽多彩的世界却是由三原色组合而成的。计算机可以完成各种各样的功能，而这些功能也是由有限的一些基本操作组合而成的。比如：读取一个数据、保存一个数据、计算一个算式的值、比较两个数值的大小、移动一下光标的位置、发出一个特定频率的声音等。

要让计算机来求解现实世界中的某一实际问题，程序员必须将该问题的求解方法和步骤进行归纳和抽象，将一系列的解题步骤编排成一组计算机能够执行的操作指令（即程序）。这就需要程序员应具有较强的概括和抽象的能力。

1.1.5 程序的质量标准

如前所述，程序设计的构造性特点决定了设计出的程序必然是多样的。那么，什么样的程序才算是好程序呢？至今还没有统一的标准。

实际上，随着计算机软硬件技术的飞速发展，人们所追求的目标也在不断的变化。20世纪80年代之前，不管程序结构多么混乱，只要运行的速度快、占用的内存少，就可算是好程序，因为那时计算机的硬件价格昂贵、性能差，而程序的规模较小。而后来人们越来越注重程序的易读性了，因为计算机硬件的性能价格比不断提高，程序的规模越来越大，结构混乱、不易阅读和理解的程序将会增大维护成本。

评价程序质量的优劣虽然没有统一的标准，但有一个基本共识，即应该从正确性、易读性、有效性、可维护性和适应性等几个方面来综合评价。

1. 正确性

正确性是对程序的最基本的要求。那么，什么样的程序才算是正确的呢？答案是：正确的程序完成且仅完成其“规格说明”中所列举的全部功能。换句话说，正确的程序除了完成其“规格说明”中所列举的全部功能以外，不做任何其他额外的事情。

知道了什么样的程序是正确的，并不意味着就能知道一个程序是否正确。从理论上说，按照一定的规则设计出的程序是可以证明其正确性的，但不幸的是，这些证明方法都异常的繁琐，以至于无法使用。

迄今为止仍然没有证明程序正确性的有效方法，人们能够做到的还是通过测试来验证正确性。然而，“测试只能找出程序的错误而并不能证明程序的正确”。因为不可能将程序的所有可能的输入、所有可能的操作和所有可能的结果都验证一遍，比如：输入有 1000 种可能，计算有 10 种可能，输出有 500 种可能，全部可能验证到就有 5000000 种组合(大型程序像这样验证，也许得花上几十年乃至几万年!)。

因此，在设计程序的过程中，程序员应该尽量使用好的程序设计方法，管理者应该加强管理和审查。这样，即使不能保证设计出的程序完全正确，但基本能够保证程序不出现严重问题。

2. 易读性

易读性也称简明性，是指在程序设计时应该尽可能地使程序容易阅读、容易理解。这是现在程序设计所追求的重要目标。

一般来说，程序不是编制完毕投入运行就万事大吉了，而是进入了运行维护期。统计数据表明，程序维护的工作量要比程序设计的工作量大得多。程序维护人员可能就是程序设计人员，但一般不是。要维护一个程序，首先要读懂这个程序。

程序的结构清晰、层次分明，程序中标识名称含义清楚、注释详细等都有助于对程序的阅读和理解，有利于程序的维护。

3. 有效性

有效性是指程序运行的速度快并且占用的空间少，这当然是我们所追求的目标。

一般情况下，时间和空间很难兼得：想节省时间，往往要以多占用空间为代价；想节省空间，往往要以增加处理时间为代价。因此，程序员应根据所解决问题的具体特点来决定时间和空间的取舍，以寻求一个合理配合。

值得注意的是，有效性和易读性往往是有冲突的。例如，要计算一个班级某门课程的平均分，在程序中可以写成：

班级平均分＝班级总分/班级人数

也可以写成：

A＝S/N

显然，前者的可读性要优于后者，但需要占用较多的空间。

应该指出，对于一般的应用程序而言，目前计算机的运行速度和存储空间已经基本上不用考虑效率问题，因此，当有效性和易读性发生冲突时，我们宁可要易读性。但对于一些实时控制系统(比如控制导弹飞行的程序)来说，时间效率仍然是要优先考虑的。

4. 可维护性

可维护性是指程序投入运行之后，如果发现错误或缺陷，应该能够纠正；如果有新的功能

要求，应该能够加入；如果程序的运行环境发生了变化，应该能使程序适应这种变化。可维护性也是好程序的一个重要指标。

开发一个软件需要投入大量的人力和财力，如果发现问题后能够修复并继续使用，这将延长软件的使用期限。

5. 适应性

适应性有两个方面的含义：一是指程序对需求变化的适应程度；二是指程序对运行环境变化的适应程度。

对于信息管理类的应用程序来说，程序都是用高级程序设计语言开发的，适应运行环境变化一般不会有问题，需要关注的是对需求变化的适应程度。比如，一个企业的经营范围扩大了，该企业的管理程序应该能够不做修改或稍加修改就能适应。

在分析问题时充分考虑未来可能的发展和变化，在程序设计时注意留有扩充的余地，可以提高程序适应性。

1.2 程序设计语言

计算机只是一种工具。要让计算机完成某项工作，必须事先编制指挥计算机工作的程序。只有这样，计算机才能按照人的意志正确工作。程序设计语言就是编写计算机程序所使用的语言。

程序设计语言是人和计算机交互的基本工具，其特性会影响人的思维和解决问题的方式，会影响人和计算机交互的方式和质量，也会影响其他人阅读和理解程序的难易程度。

每一种不同的程序设计语言都有着各自的“长处”和“短处”、适合于做什么和不适合于做什么。因此，程序员对程序设计语言的一些基本特性应该有所了解，以便能在编程时选择一种适当的程序设计语言。

1.2.1 程序设计语言的分类

据报道，自 1960 年以来人们已经设计和实现了数千种不同的程序设计语言，其中只有很少一部分得到了比较广泛的应用。现有的程序设计语言虽然五花八门、品种繁多，但它们基本上可以分为汇编语言和高级语言两大类。

1. 汇编语言

汇编语言的语句和计算机硬件操作有一一对应关系，每种汇编语言都是支持这种语言的计算机所独有的，因此，有多少种不同类型的计算机也就有多少种汇编语言。

2. 高级语言

高级语言使用的概念和符号与自然语言使用的概念和符号比较接近，它的一个语句（命令）往往对应若干条机器指令。一般地说，高级语言与计算机的类型无关。现在，编写程序所使用的基本上都是高级语言。

对于高级语言，从不同的角度，又可以进一步分类。根据其应用特点，可以分为通用语言、专用语言和数据库管理系统三类。

1）通用语言

通用语言的特点是应用范围广泛，可以编写解决各类问题的程序，具有很强的过程能力和

数据结构能力。比如，BASIC/Visual BASIC，C/C++/Visual C++，PASCAL 等都属于这类语言。

2）专用语言

专用语言的特点是应用范围比较狭窄，只能编写解决某类问题的程序。例如，APL 是为数组和向量运算设计的简洁而又功能很强的语言；BLISS 是为开发编译程序和操作系统而设计的语言；FORTH 是为开发微处理机软件而设计的语言；LISP 语言和 PROLOG 语言特别适合于编写人工智能领域的应用程序。

3）数据库管理系统

数据库管理系统也是一种高级语言，它的应用范围介于专用语言和通用语言之间，它特别适用于开发需要数据库技术支持的各类应用程序。目前常用的数据库管理系统有，SQL Server，Oracle，Visual FoxPro 等等。

按照源程序到目标程序的翻译方式，可以分为解释型语言和编译型语言两类：

1）解释型语言

用某种程序设计语言编写的程序称为该种语言的源程序。例如，用 C++语言编写的程序称为 C++源程序。源程序是不能被计算机直接执行的，只有将源程序“翻译”成目标程序（由二进制代码组成）后，计算机才能执行。

解释型语言源程序的执行方式是，由一个称为解释程序的语言处理软件，对源程序语句逐句进行解释（翻译成目标程序），解释一句，执行一句，解释并执行完毕时，也就得到了完整的运行结果。描述解释型语言源程序的执行过程如图 1.2 所示。

图 1.2　解释型语言源程序的执行过程

这种翻译方式执行速度较慢，因为它不产生可执行文件，每次运行都要重新解释一次。现在，纯粹的解释型语言已不多见，还在使用的大概就是 BASIC 语言了，Visual BASIC 不是纯粹的解释型语言，它也可以编译。

2）编译型语言

编译型语言源程序的执行方式是，由一个称为编译程序的语言处理软件，将整个源程序翻译成可以单独执行的目标程序，然后执行目标程序，得出运行结果。编译型语言源程序的编译和执行过程如图 1.3 所示。

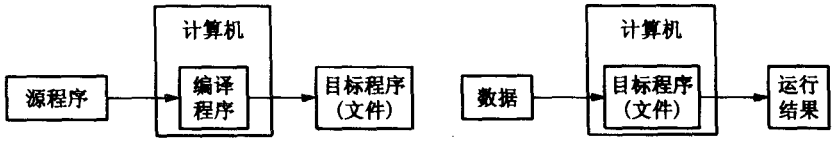


图 1.3　编译型语言源程序的编译和执行过程

编译生成了可执行文件之后，就不再需要源程序了，而且执行速度较快。目前使用的高级语言基本上都是编译型的。例如，C/C++/Visual C++，PASCAL，PROLOG 等。