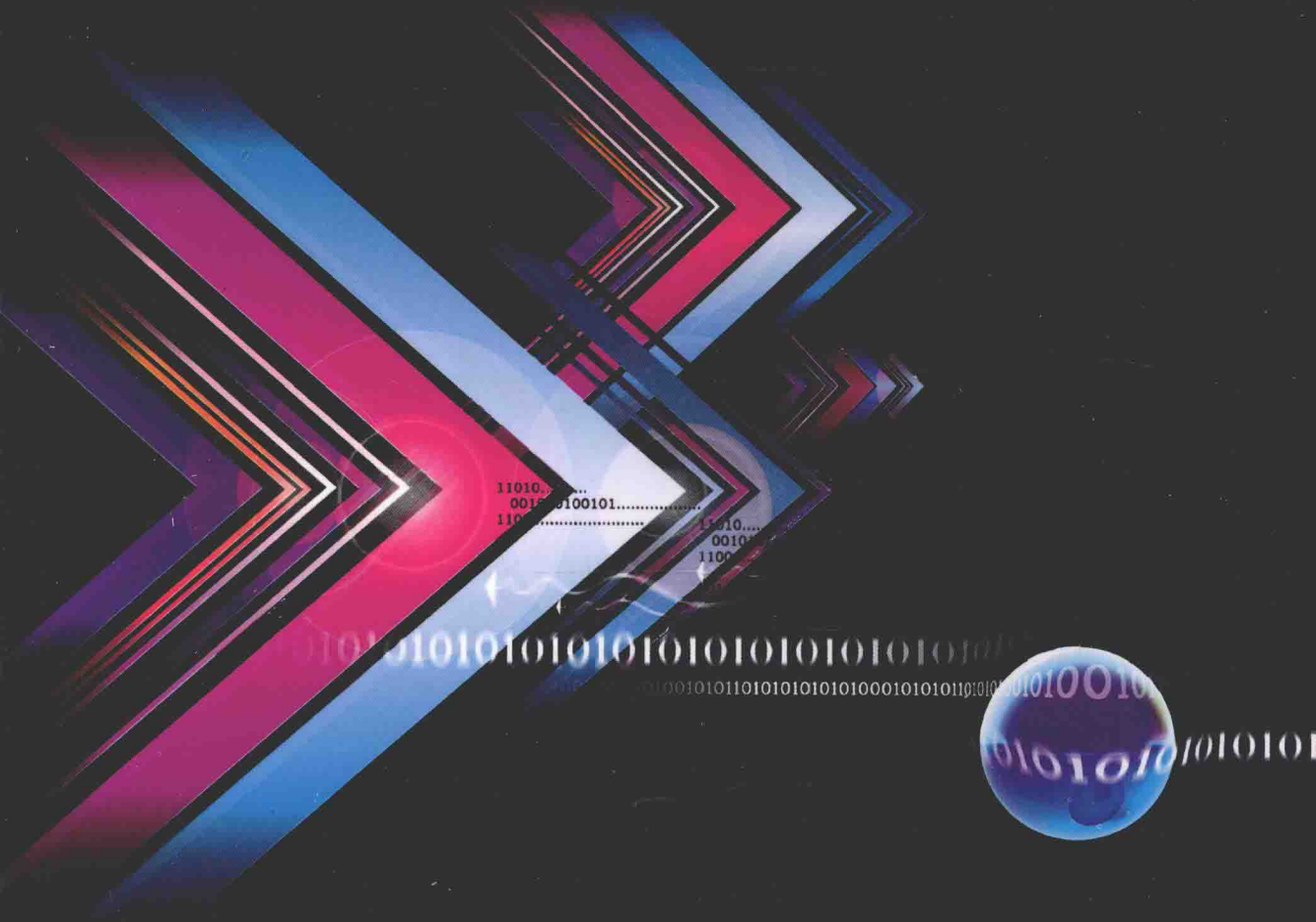
中国电子教育学会高教分会推荐

普通高等教育电子信息类“十三五”课改规划教材



C程序设计与系统开发

主编　祁建宏　张志斌

西安电子科技大学出版社
http://www.xduph.com

中国电子教育学会高教分会推荐

普通高等教育电子信息类规划教材

C程序设计与系统开发

主　编　祁建宏　张志斌

参　编　刘子江　文　静　朱小军

张海瑞　郭　媛

西安电子科技大学出版社

内 容 简 介

本书采用案例教学法，将繁琐而抽象的语法规则融入具体例子当中，通过所给出解决实际问题的典型例子，使读者达到掌握语言本身相关规则的目的，同时培养读者解决实际问题的能力。课后配以针对性的习题，以巩固对相关知识点的理解掌握。

本书共 12 章，分别为算法及其描述方法、程序设计基础、数组及字符串、复杂数据类型、指针、函数、文件、系统开发与链表、面向对象程序设计入门、可视化程序设计、位运算、预处理。前 7 章介绍 C 语言编程基础知识；第 8 章主要是对前述内容的综合应用，讲述完整系统开发的一般流程及单链表的相关操作；第 9 章介绍了面向对象程序设计的基础知识；第 10 章重点介绍了利用 MFC 开发 Windows 图形界面风格软件的一般方法，以及 C/S 模式软件开发的一般方法；第 11 章介绍了位运算，以重点满足利用 C 语言进行通信、控制等领域软件开发的用户的需求；第 12 章讲述预处理，用于增强软件可移植性。

本书结构新颖、内容丰富、条理清晰、重点突出，可用作高等院校计算机相关专业的教材，也可作为社会培训教材或自学参考书。

图书在版编目(CIP)数据

C 程序设计与系统开发/祁建宏，张志斌主编. —西安：西安电子科技大学出版社，2016.8
普通高等教育电子信息类“十三五”课改规划教材
ISBN 978-7-5606-4144-7

Ⅰ. ① C… Ⅱ. ① 祁… ② 张… Ⅲ. ① C 语言—程序设计—高等学校—教材 Ⅳ. ① TP312

中国版本图书馆 CIP 数据核字(2016)第 184977 号

策　　划　刘王芳
责任编辑　阎　彬　祝婷婷
出版发行　西安电子科技大学出版社(西安市太白南路 2 号)
电　　话　(029)88242885　88201467　　邮　　编　710071
网　　址　www.xduph.com　　电子邮箱　gsqls@163.com
经　　销　新华书店
印刷单位　陕西天意印务有限责任公司
版　　次　2016 年 8 月第 1 版　2016 年 8 月第 1 次印刷
开　　本　787 毫米×1092 毫米　1/16　印　张　25
字　　数　596 千字
印　　数　3000 册
定　　价　43.00 元

ISBN 978-7-5606-4144-7/TP

XDUP 4436001-1

前　言

自 2010 年《国家中长期教育改革和发展规划纲要(2010—2020 年)》颁布以来，教育部就开始筹划高等教育体制和结构改革，核心是改变以往单一的学术型或研究型办学模式，将高职、新建地方本科院校、独立学院纳入现代职业教育体系，对高等学校实行分类管理。2014 年 6 月，国务院出台《关于加快发展现代职业教育的决定》，同时，教育部等 6 部委联合印发《现代职业教育体系建设规划(2014—2020 年)》，今后高校将分为研究型高校、应用技术型高校、高等职业学校。另外，2014 年 3 月教育部表示新的高考改革方案包含两种模式：技术技能人才高考和学术型人才高考。这些措施都表明我国对应用型人才培养的重视程度在逐渐提高。

程序设计的特点之一是技术实践性很强，应用型特征明显。但长期以来，高校中对该类课程的教学都以理论为主，培养的人才实践能力较差。为满足市场对程序设计人才的实际需求，改变传统授课模式，培养出既掌握理论知识、又具有很强实践动手能力、能综合运用所学相关知识高效而准确地解决实际问题的软件开发人才，是高校计算机人才培养的一个重要课题。

C 语言自推向市场，就以其丰富的数据类型及运算符、自由灵活的编程风格、强大的硬件编程能力等独特优点，始终牢牢占领着编程市场很大的份额。时至今日，在许多学校的计算机语言教学和通信、控制等领域的软件开发中，C 语言都成为首选。

本书共 12 章，另加 5 个附录，内容涉及 C 语言编程相关知识、数据结构基础知识、面向对象程序设计基础知识、可视化程序设计基础知识、软件工程基础知识、计算机硬件基础知识等，顺应了目前流行的课程整合思想。与同类书相比，本书更注重培养学生综合运用相关知识解决实际问题的能力。本书全面介绍了 C 语言的相关内容，另外还增加了“数据结构”、“软件工程”、“计算机组成原理”的部分内容。

与传统 C 语言教材相比，本书有以下优点：

(1) 开发环境采用目前 C 语言教学及考试的主流版本 VC 6.0；

(2) 采用案例教学法，将繁琐而抽象的语法规则融入具体例子当中，更有助于激发学习兴趣，培养学生解决实际问题的能力；

(3) 增加了“数据结构”的基础性内容，以提高读者的综合编程能力；

(4) 增加了“软件工程”的基础性内容，通过完整案例介绍了软件生命周期，以便读者掌握软件开发的一般流程；

(5) 增加了面向对象程序设计的介绍；

(6) 针对 C 语言学习中的难点(指针)及其最常见应用场合(链表)，专门设计了案例以加强对这部分内容的理解及掌握；

(7) 通过完整案例介绍了利用 MFC 实现可视化程序设计，以便开发 Windows 图形界面风格的软件；

(8) 通过完整案例介绍了 C/S 模式软件开发的一般方法；

(9) 习题以程序为主，通过大量的练习培养实践动手能力；

(10) 有实用的附录资料，以方便读者参考。

同时，本书有配套的实践指导书，重点介绍了以编程方式解决实际问题时常用的一些算法及典型软件系统的开发过程，可针对性地训练学生的实践动手能力。

由于作者水平有限，虽然编写过程严谨，但也难免存在不足之处，敬请广大读者朋友批评指正。

编　者

2016 年 5 月

目　录

第 1 章　算法及其描述方法 1
1.1　为什么要编写程序 1
1.2　算法的概念及基本特征 2
1.3　结构化程序设计方法 4
1.4　算法的几种描述方法 5
1.4.1　自然语言 5
1.4.2　流程图 5
1.4.3　N-S 图 7
1.4.4　计算机语言 7
1.4.5　伪代码 10
1.5　C 语言简介 10
1.6　VC 6.0 上机调试过程 11
1.6.1　C 语言程序的编制运行过程 11
1.6.2　VC 的启动 12
1.6.3　VC 的关闭 12
1.6.4　VC 中新程序的建立及调试运行 12
1.6.5　当前源程序及相关环境的关闭 15
1.6.6　已存在程序文件的打开及运行 16
习题 1 16

第 2 章　程序设计基础 18
2.1　信息处理流程概述 18
2.2　数据的输入、存储、加工处理及输出 18
2.2.1　数据的输入、存储、加工处理及输出流程示例 18
2.2.2　标识符及其起名规则 20
2.2.3　常量及变量 21
2.2.4　赋值运算符 25
2.2.5　算术运算符 26
2.2.6　格式化输出函数 printf 27
2.2.7　格式化输入函数 scanf 31
2.3　顺序结构程序设计 32
2.4　选择结构程序设计 34
2.5　循环结构程序设计 43
2.5.1　当型循环 44
2.5.2　直到型循环 46
2.5.3　for 循环 48
2.6　三种控制结构的综合应用 50
2.7　运算符及其优先级和结合性 61
2.7.1　运算符的优先级及结合性简述 61
2.7.2　常见运算符及其相关说明 63
2.7.3　有关结合性的解释 66
2.8　不同类型数据间的转换与运算 67
2.8.1　隐式自动转换 67
2.8.2　显式强制转换 69
习题 2 69

第 3 章　数组及字符串 74
3.1　C 语言中的数组 74
3.1.1　一维数组 74
3.1.2　二维数组 85
3.1.3　多维数组 88
3.2　字符串 89
3.2.1　字符串的本质 89
3.2.2　字符数组 89
3.2.3　常用字符串操作函数 90
3.2.4　字符串应用举例 93
习题 3 96

第 4 章　复杂数据类型 97
4.1　C 语言中的复杂数据类型概述 97
4.2　结构体 97
4.2.1　结构体类型的定义 98
4.2.2　结构体变量的定义 99
4.2.3　结构体变量的引用 101
4.2.4　结构体变量的赋值 102
4.3　共用体 105
4.3.1　共用体类型的定义 107
4.3.2　共用体变量的定义 107
4.3.3　共用体变量的引用 108
4.3.4　共用体变量的赋值 110

4.4 枚举类型 113
4.4.1 枚举类型的定义 113
4.4.2 枚举型变量的定义 114
4.4.3 枚举型变量的赋值 114
4.4.4 枚举类型有关说明 114
习题 4 116

第 5 章 指针 119
5.1 C 语言中的指针概述 119
5.1.1 指针与指针变量 119
5.1.2 指针相关运算 123
5.2 指针与数组的关系 124
5.2.1 数组的指针与指向一维数组元素的指针变量 124
5.2.2 一维数组与指针变量的关系 125
5.2.3 指向指针的指针变量 127
5.2.4 指针数组——元素类型为指针的数组 127
5.2.5 指针数组与指向指针的指针变量的关系 127
5.2.6 二维数组与指针变量的关系 130
5.3 指针与动态内存分配 134
5.3.1 C 语言内存管理概述 135
5.3.2 内存空间的动态分配 136
5.3.3 动态释放内存 137
5.3.4 动态内存分配的几种不同情形 137
5.3.5 常见内存错误及其对策 141
5.4 指针与字符串 141
5.4.1 用 C 语言处理字符串的两种不同形式 141
5.4.2 字符型指针变量与字符型数组 143
习题 5 144

第 6 章 函数 145
6.1 C 语言函数简介 145
6.1.1 函数的概念及分类 145
6.1.2 标准函数的使用 145
6.2 自定义函数 147
6.2.1 自定义函数概述 147
6.2.2 自定义函数的一般定义形式 149
6.2.3 自定义函数的调用 149
6.2.4 模块化程序设计 152
6.3 函数中参数的传递方式 155
6.3.1 按值传送 155
6.3.2 按地址传送 156
6.3.3 两种不同参数传递方式的选择 157
6.4 变量的作用域、生存期及存储类型 158
6.4.1 变量的作用域 158
6.4.2 变量的生存期 158
6.4.3 局部变量 158
6.4.4 全局变量 160
6.4.5 文件变量 161
6.4.6 变量的存储类型 162
6.4.7 外部变量在多文件系统中的应用 167
6.4.8 变量作用域、生存期及存储类型小结 168
6.5 函数的嵌套与递归调用 169
6.5.1 函数的嵌套调用 169
6.5.2 函数的递归调用 169
习题 6 174

第 7 章 文件 175
7.1 文件概述 175
7.1.1 文件概述 175
7.1.2 文件的分类 175
7.2 文件操作 177
7.2.1 文件指针 178
7.2.2 文件操作基本步骤示例 178
7.3 文件操作相关函数 180
7.3.1 文件的打开：fopen()函数 180
7.3.2 文件的关闭：fclose()函数 182
7.3.3 文件格式化输出函数：fprintf() 182
7.3.4 文件格式化输入函数：fscanf() 182
7.3.5 判断是否到文件尾函数：feof() 183
7.3.6 文件数据块读/写函数：fread()和 fwrite() 183
7.3.7 文件内部指针的定位 183
7.3.8 清除文件缓冲区函数：fflush() 188
习题 7 189

第 8 章　系统开发与链表 190
8.1　软件工程简介 190
8.1.1　软件工程概述 190
8.1.2　软件工程中的瀑布模型 191
8.2　“班级基本信息管理系统”开发示例 193
8.2.1　问题的定义 194
8.2.2　可行性研究 194
8.2.3　需求分析 194
8.2.4　系统设计(概要设计) 195
8.2.5　详细设计 195
8.2.6　编程 198
8.2.7　测试 204
8.2.8　运行及维护 205
8.3　链表与系统开发 205
8.3.1　问题的定义 205
8.3.2　可行性研究 205
8.3.3　需求分析 205
8.3.4　系统设计(概要设计) 205
8.3.5　详细设计(算法设计) 208
8.3.6　编程 212
8.3.7　测试 219
8.3.8　总结 220
习题 8 220

第 9 章　面向对象程序设计入门 221
9.1　面向对象程序设计概述 221
9.1.1　面向过程的程序设计方法 221
9.1.2　面向对象的程序设计方法 222
9.1.3　面向对象程序设计的特点 223
9.1.4　面向对象与面向过程程序设计比较 225
9.1.5　三种常用的面向对象程序设计语言 226
9.1.6　面向对象程序设计示例 226
9.2　面向对象程序设计方法的技术实现 227
9.2.1　类的定义 227
9.2.2　成员的存取控制类别 230
9.2.3　构造函数和析构函数 230
9.2.4　面向对象程序设计实例 230
习题 9 238

第 10 章　可视化程序设计 239
10.1　MFC 简介 239
10.1.1　MFC 的定义 239
10.1.2　一个简单的 MFC 应用程序例子 242
10.2　MFC 中的控件 253
10.2.1　静态控件 253
10.2.2　按钮 256
10.2.3　编辑框 260
10.2.4　旋转按钮控件 262
10.2.5　列表框 268
10.2.6　组合框 275
10.2.7　进度条 281
10.2.8　滚动条 284
10.2.9　滑动条 285
10.2.10　日期时间控件 290
10.2.11　图像列表控件 290
10.3　用 MFC 进行可视化系统开发 291
10.4　C/S 模式的“班级基本信息管理系统”的开发 318
10.4.1　在 VC 中用 ODBC 访问数据库 318
10.4.2　MFC 的 ODBC 类 322
10.4.3　MFC ODBC 数据库编程的一般处理流程 322
10.4.4　系统说明 323
10.4.5　开发过程 323
习题 10 340

第 11 章　位运算 341
11.1　C 语言中的位运算 341
11.2　位运算实际应用 342
11.2.1　按位与、或、异或及取反 342
11.2.2　移位运算 344
11.2.3　典型应用案例 345
11.3　位域 348
11.3.1　位域类型定义 349
11.3.2　位域变量定义 350
11.3.3　位域的使用 350
11.3.4　位域的空间分配 351
习题 11 353

第 12 章　预处理 354

12.1 预处理简介 354
12.2 宏定义 354
12.2.1 无参宏定义 355
12.2.2 带参宏定义 355
12.2.3 取消宏定义 357
12.3 文件包含 357
12.4 条件编译 357
习题 12 360

附录 A 常见标准函数 361
附录 B C 语言关键字 378
附录 C ASCII 码表 379
附录 D C 语言常见错误举例说明 383
附录 E 学习建议 391
参考文献 392

第 1 章　算法及其描述方法

本章重点：

1. 算法的概念及主要特征
2. 算法的几种常见描述方法
3. 算法的三种基本结构
4. C 语言简介

1.1　为什么要编写程序

先来看一个趣味题：将数字 1、3、5、7、9、11、13 共 7 个数填入图 1-1，使得每个圆圈内 4 个数字相加的和都相等，每种确定的填充方案中每个数字只能出现一次，请找出所有符合条件的填充方案。

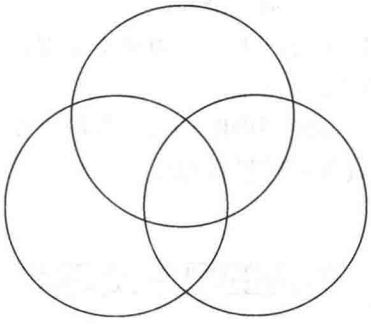


图 1-1　趣味图

分析：如果仅找出一种方案，此问题很简单，但现在要找出所有方案，如何实现呢？为便于讨论，我们对图 1-1 中的七个填充位置按图 1-2 所示进行标注。

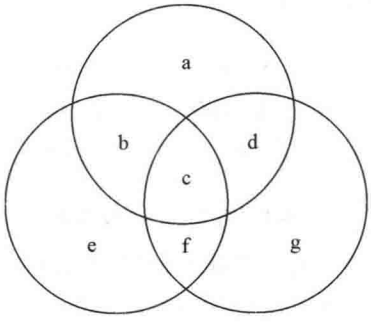


图 1-2　标注过的趣味图

由题可知，每个位置可供选择的数字是 1、3、5、7、9、11、13 七个数中的某一个，即每个位置有七种取值的可能性，那么七个填充位置共有 7^7(即 823 543)种可能的情形，只要将这些可能的情形全部列举出来，再将其中符合题目要求的情形挑选出来，就可以找出所有的填充方案了。部分可能情形如表 1-1 所示。

表 1-1　部分可能情形

a	b	c	d	e	f	g	是否符合填充条件
1	3	5	7	9	11	13	否
1	3	5	7	9	13	11	否
1	3	5	7	11	9	13	否
1	3	5	7	11	13	9	否
1	3	5	7	13	9	11	否
1	3	5	7	13	11	9	否
…	…	…	…	…	…	…	…

但是，由于此问题的可能情形共有 823 543 种，用传统的手工方式去寻找所有答案速度太慢。对于手工解决明显太慢的问题，可以利用计算机来解决，也就是说需要编写程序。

相对于人而言，计算机具有速度快、信息存储容量大、检索速度快、数据精确度高等优势。对于一些按传统方式不好解决，难以达到速度、精度等方面要求的问题，就可以考虑发挥计算机的优势，通过编写程序来解决问题。

利用计算机编程解决实际问题时，大体分为两个步骤：

第一步：针对具体问题制订解决方案；

第二步：根据制订的解决方案编写程序，在计算机上调试运行程序，得出结果。

其中，第一步实际上就是针对具体问题设计算法。

1.2　算法的概念及基本特征

解决实际问题时，都要制订一个针对具体问题的解决步骤和方法，以此为据去实现目标。将为了解决实际问题而制订的步骤、方法称为算法(Algorithm)。

【例 1-1】 有分段函数如图 1-3 所示。

$$y=\begin{cases} x+1 & x>0 \\ x-100 & x=0 \\ 20x & x<0 \end{cases}$$

图 1-3　例 1-1 的分段函数

如果 x 及对应的 y 都为整数，则手工计算也比较方便，但如果是实数，且位数较多，则手工计算的难度会大大增加，此时便可考虑编程解决。通过分析手工做题过程，可以总结出根据 x 计算 y 值的一个算法如下：

第一步：输入 x 的值；

第二步：判断 x 是否大于 0，若大于 0，则 y 值为 x + 1，然后转第五步；否则进行第三步；

第三步：判断 x 是否等于 0，若等于 0，则 y 值为 x - 100，然后转第五步；否则进行第四步；

第四步：y 值为 20x (若第二、三步所对应的条件均不成立，则第四步所对应条件肯定成立)，继续向下进行；

第五步：输出 y 的值后结束。

需要注意的是，针对同一个问题，可能有多个不同的算法，如此例就可以设计出另外的算法，请读者自行尝试完成。

有些算法可以在计算机中实现，有些则无法在计算机中实现，我们以后所提到的算法都是指能在计算机中实现的算法。

算法具有如下基本特征：

1. 有穷性

算法中所包含的步骤必须是有限的，不能无穷无止，应该在人们所能接受的合理时间段内产生结果。

设计算法是为了解决实际问题，如果一个算法中包含的步骤是无限的，也就意味着按此算法得到答案的时间是无限的，这样的算法明显没有意义。

另外，对于有些算法，虽然其步骤有限，但其实现的时间满足不了实际要求，也不能算是合格的算法。如我们设计了一个预测天气状况的算法，只要将有关天气情况的历史数据和当前数据输入进去，经过复杂的计算推导，就可以精准地对以后的天气状况进行预测，但其运算量特别大，速度慢，即使采用计算机来实现，对于第三天的天气状况的预测结果也要到第四天才能出来，这样的算法还有意义吗？

2. 确定性

算法中的每一步所要实现的目标必须是明确无误的，面对不同的人或机器，要有相同的理解，不能有二义性，否则就会导致同一个算法对于同一组数据产生不同的结果，这显然是不合情理的。

3. 有效性

算法中的每一步如果被执行了，就必须被有效地执行。例如，有一步是计算 X 除以 Y 的结果，如果 Y 为非 0 值，则这一步可有效执行，但如果 Y 为 0 值，则这一步就无法得到有效执行。

4. 有零或多个输入

根据算法的不同，有的算法在实现过程中需要输入一些原始数据，而有些算法可能不需要输入原始数据。

5. 有一个或多个输出

设计算法的最终目的是解决问题，因此，每个算法至少应该有一个输出来反映问题的最终结果。

1.3 结构化程序设计方法

E. W. Dijikstra 在 1965 年提出的结构化程序设计(structured programming)方法，是软件发展的一个重要里程碑。它采用“自顶向下、逐步细化、模块化”的程序设计方法，将一个大问题从全局到局部逐层分解为若干个子模块，直到每个子模块的功能及规模相对简单为止。

该方法的要点有以下几点。

(1) 一个入口，一个出口。

算法中的每一个步骤都是只有一个入口、一个出口，这样便于跟踪算法的流程；反之，如果允许多个入口或多个出口，就会出现类似迷宫的情况，到一个位置以后，下一步会有多种选择，这会大大增加跟踪算法流程的难度，不利于算法的分析及查错。

结构化程序设计方法提倡一个算法由三种基本控制结构组成，这三种控制结构都是只有一个入口和一个出口。这样做的目的是使算法的流程更清晰简洁，以方便用户读懂算法，增强算法的可读性。这三种基本结构分别如下：

① 顺序结构。这种结构为按从前向后的顺序逐步执行的控制结构，这也是现实中最常见、最自然的一种结构。如我们每天的生活，就是按一定顺序逐步进行的。

② 选择结构。这种结构又称分支结构，是根据指定条件作出决策，在两条或多条分支路径中选择其中一条执行的控制结构。

③ 循环结构。这种结构为根据是否满足指定的条件而决定是否重复执行指定操作的控制结构。

根据理论证明，任何复杂的处理过程都可以用这三种控制结构组合实现。也就是说，一种计算机语言中如果有了这三种控制结构，就可以解决任何复杂的问题了。因此，掌握这三种控制结构的基本思想是学习程序设计的基础。有关这三种控制结构的具体介绍见后续各章节内容。

(2) 严格控制无条件转移语句 GOTO 的使用以避免算法结构混乱。

无条件转移语句 GOTO 可使算法流程进行任意跳转，这在有些情况下会提高算法设计的灵活性，但同时也会导致算法的流程严重混乱，给算法的分析增加难度，不利于排错。

(3) 采用“自顶向下、逐步细化”的方法，将复杂问题分解为若干个小模块进行处理。

现实中人们处理复杂问题的一种通用办法就是进行分解——化整为零。这种做法一方面有利于将复杂问题简单化，一个复杂问题分解成多个子模块之后，每个子模块只对应其中的一部分，复杂程度肯定会降低；另一方面，将问题分解开之后，也有利于多人分工合作。

(4) 采用主程序员组的组织形式。

主程序员组的组织形式指开发程序的人员组织方式应采用由一个主程序员(负责全部技术活动)、一个后备程序员(协调、支持主程序员)和一个程序管理员(负责事务性工作，如收集、记录数据，文档资料管理等)为核心，再加上一些专家(如通信专家、数据库专家)及其他技术人员所组成的小组。

综上所述，其中前两条可解决程序结构的规范化问题，第三条可解决将大化小、将难化简的求解方法问题，第四条可解决软件开发的人员组织结构问题。

1.4　算法的几种描述方法

有多种方法来描述具体算法，下面进行具体的介绍。

1.4.1　自然语言

描述算法最简单的一种工具就是自然语言，如汉语、英语等。其优点是通俗易懂，易于掌握，一般人都会用。但也有其缺点：一是繁琐，不直观；二是容易产生歧义。此种方法在计算机领域内很少被使用。

1.4.2　流程图

流程图使用一些图框表示各种类型的操作，用线条指示操作的执行顺序。相对于自然语言，流程图所表示的算法显得更简洁、直观、易懂。常用的图框符号如图 1-4 所示。

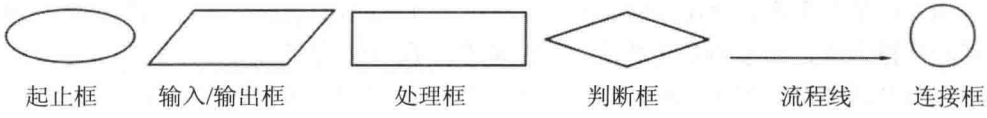


图 1-4　流程图常用图框符号

对流程图中所用的符号作如下说明。

(1) 起止框：椭圆形状，用于标明算法的开始及结束位置，每个算法的开头和结尾各放置一个。

(2) 输入/输出框：在算法中进行输入及输出的位置可放置此框，框内注明要具体进行的输入及输出操作。

(3) 处理框：用于算法中间处理数据或计算。

(4) 判断框：用于给出判断的依据，框内标明判断条件，用于“分支”及“循环”两种控制结构。

(5) 流程线：用于将上述各框连接起来并指明执行的方向。按一般约定，如果是从上向下、从左向右，则箭头可以省略；但如果是从下向上或从右向左，则箭头不能省略。

(6) 连接框：为正圆形状，以便与起止框相互区分。有时我们要绘制的流程图可能比较大，一个页面放不下，需要放到多个页面中，这会导致流程图断开。此时，可在前页中断开的地方放置一个连接框，框内用符号标注；接下来在后页中与前页断开处相对应的位置也放置一个连接框，框内用与前面断开处相对应的符号进行标注，这样，通过使用连接框就可以将断开的位置联系起来了。

将前面所讲的三种控制结构用流程图来表示，如图 1-5 所示。图中的“Y”指“YES”，表示条件成立；“N”指“NO”，表示条件不成立；而“A”、“B”表示算法中的某一步骤。

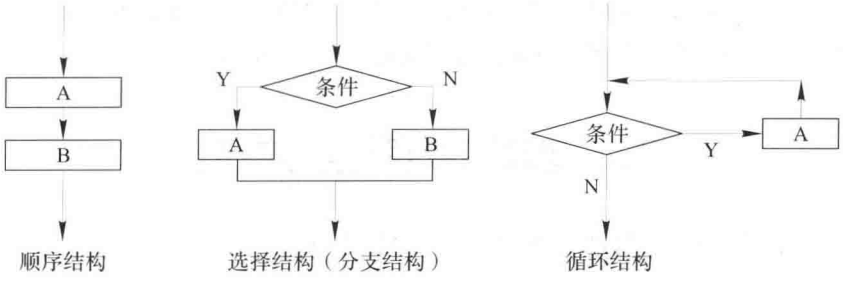


图 1-5　流程图表示三种基本控制结构

有关这三种结构作如下说明：

(1) 顺序结构。这种结构按从前往后的顺序执行，每个步骤的执行次数为固定的 1 次。

(2) 选择结构。这种结构先判断条件，若条件成立则执行分支“A”，条件不成立则执行分支“B”。可看出，条件判断进行 1 次，分支“A”及分支“B”的执行次数为 0 次(不执行)或 1 次(执行)。其实际效果就是根据条件从两个分支中选择一个执行。

(3) 循环结构。这种结构先判断条件，若条件成立则执行“A”，执行完后接着判断条件……这样就可以反复多次判断条件并执行“A”，若条件不成立则结束此循环并继续向下推进，“A”可能会反复执行多次，习惯上称之为“循环体”。可看出，条件判断至少 1 次，多则可能是很多次，“A”的最少执行次数为 0 次(第 1 次进行条件判断时就不成立)，多则可能是很多次，但不能是无数次，那样就会违反算法的有穷性原则。其实际效果就是根据条件对循环体进行多次执行，通常用于完成一些需要多次重复进行的操作。

需要指出的是，此处仅列出了三种控制结构的常见形式，实际应用中还有其他形式，在以后的章节中会进一步说明。

对于前面的例 1-1 用流程图所描述的算法如图 1-6 所示。

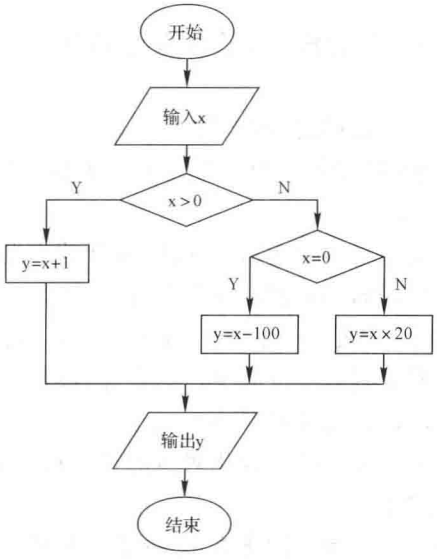


图 1-6　例 1-1 的算法流程图

1.4.3　N-S 图

N-S 图是描述算法的另一种常见工具，是对流程图的一种改进，最大改变是省掉了流程图中的流程线，使得图形更紧凑。用 N-S 图表示三种基本控制结构如图 1-7 所示。

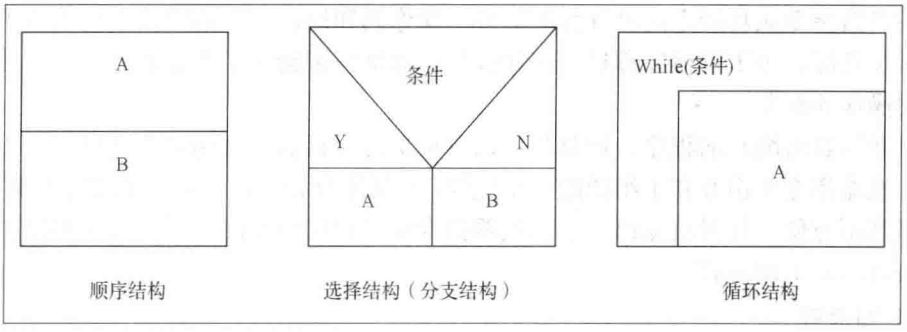


图 1-7　N-S 图表示三种基本控制结构

用流程图绘制的图形比较松散，占用的页面空间比较多。与流程图相比较，N-S 图能直观地用图形表示算法，自然地去掉了导致程序非结构化的流程线，实际绘制出来的图更紧凑，这样可以节省页面空间，但如果要进行修改，则显得不够方便。

对前述例 1-1 用 N-S 图进行描述的结果如图 1-8 所示。

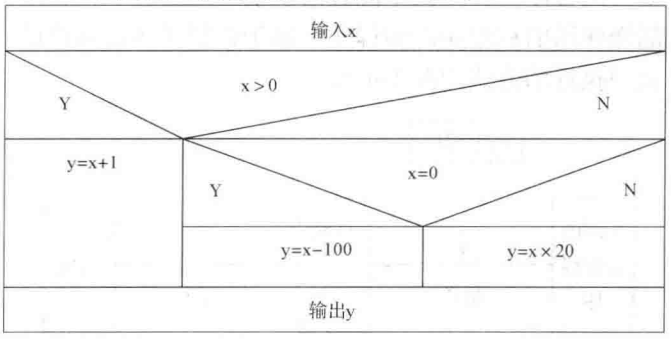


图 1-8　例 1-1 的算法 N-S 图

1.4.4　计算机语言

前述几种方法所描述的算法只能供人阅读，不能在计算机上运行，因此无法发挥出计算机的优势。若要借助于计算机解决问题，就需要将算法用计算机语言进行描述，即所谓的编程序，这就要用到计算机语言。

采用某种计算机语言，按其语法规则所描述出来的算法就是程序，也就是说程序实质体现的是一种算法。

人与人之间交流要用语言，即所谓的自然语言，而人与计算机进行交流也要用专门的、能被计算机直接或间接识别的语言，即计算机语言。

计算机语言通常是一个能完整、准确和规则地表达人们的意图，并用以指挥或控制计算机工作的“符号系统”。按其发展过程，计算机语言通常分为三类，即机器语言、汇编语言和高级语言。

1. 机器语言

机器语言是直接用二进制代码指令进行表达的计算机语言，其中所使用的指令称为机器指令。机器指令是由 0 和 1 组成的一串代码，它们有一定的位数，并分成若干段，各段的编码表示不同的含义。

一条机器指令从功能方面可分为两部分：操作码和地址码。其中操作码指明了指令的操作性质及功能，用于告诉计算机“干什么”；地址码则给出了操作数或操作数的地址，用于指明操作的对象。

用机器语言所编写的程序，计算机可以直接执行。机器语言相当于计算机的母语。

由于机器指令是由 0 和 1 组成的一串代码，非常不好记忆，若用机器语言去编写程序，用户会极其不方便。针对机器语言中的机器指令难以记忆的缺陷，后来又发展出了第二代计算机语言——汇编语言。

2. 汇编语言

汇编语言(Assembly Language)是一种符号语言，其基本思想是用英文单词或英文单词的缩写代替机器指令中的操作码和地址码，这种英文单词或英文单词的缩写通常称为助记符。相对于机器语言中 0 和 1 的代码串，汇编语言中的助记符更易于记忆，从而给编程人员带来了方便。

使用汇编语言编写的程序，计算机不能直接识别，需要有一种程序将汇编语言翻译成机器语言，这种起翻译作用的程序叫汇编程序，属于计算机系统软件的一种。用汇编语言设计的程序在计算机中执行的方式如图 1-9 所示。

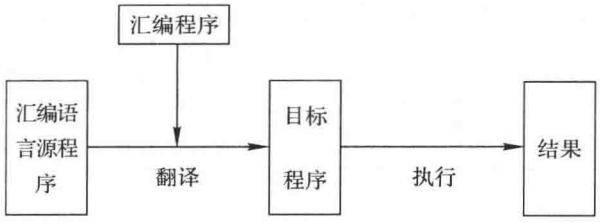


图 1-9　汇编语言程序执行方式

机器语言及汇编语言都与具体的计算机硬件联系紧密，严重依赖于具体的计算机类型，学习这两种计算机语言，要求对所使用的计算机硬件有十分深入的了解，这对于一般计算机用户来讲，很难做到。同时，不同公司所设计生产的计算机，其所使用的机器语言及汇编语言并不通用，这就意味着将在一种类型计算机上用机器语言或汇编语言编写的程序，直接移植到其他不同类型的计算机上去几乎是不可能的。程序的可移植性差，这也制约了这两类语言的实际应用。

为了克服上述机器语言及汇编语言的局限性，人们又开发出了第三代计算机语言——高级语言。

3. 高级语言

高级语言的语法和结构更接近于普通英文，且由于远离对硬件的直接操作，使得一般人经过学习之后都可以编程。高级语言并不是特指某一种具体的语言，而是包括很多，如