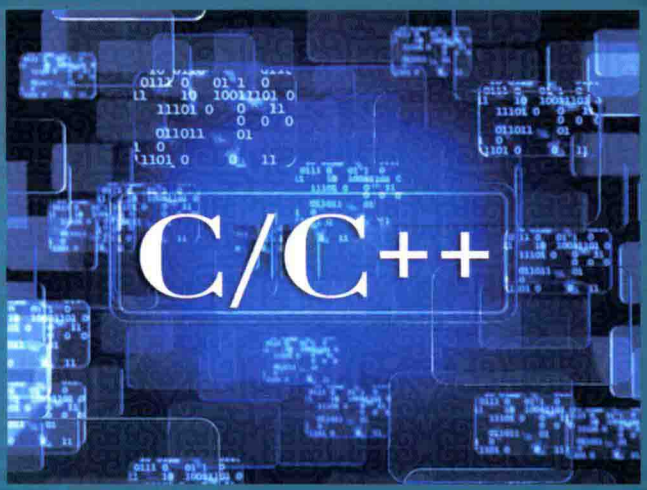
“十三五”普通高等教育规划教材

C/C++ 程序设计

第3版

陈卫卫 王庆瑞 编著

立体化·新形态教材



提供电子课件、教学视频

http://www.cmpedu.com

电子课件 教学视频 程序源代码 习题解答

机械工业出版社
CHINA MACHINE PRESS

高等教育规划教材

C/C++程序设计

第3版

陈卫卫　王庆瑞　编著

机械工业出版社

本书是学习C/C++语言程序设计的教材。内容包括：C语言基础、分支和循环、构造类型（数组、结构、联合、枚举、文件）、函数、指针类型、类和对象，以及用附录形式给出的数制和码制、ASCII码表、C/C++常用库函数、Visual C++ 6.0的基本用法、Visual C++ 2010的基本用法、部分习题参考答案等。

本书通过100多个例题和500多道习题讲解C/C++语言基本用法，向读者讲授程序设计的方法。

本书可作为普通高校计算机类专业程序设计语言课程教材，也可作为广大信息技术爱好者学习程序设计方法的参考书。

本书配有微课视频和拓展文件（包括：一部分附录和一部分例题、习题以及程序源代码等），扫描二维码即可观看。

本书另配有电子教案和习题解答源程序，需要的教师可登录www.cmpedu.com免费注册，审核通过后下载，或联系编辑索取（QQ：2966938356，电话：010-88379739）。

图书在版编目（CIP）数据

C/C++程序设计 / 陈卫卫，王庆瑞编著. —3版. —北京：机械工业出版社，2019.1

“十三五”普通高等教育规划教材

ISBN 978-7-111-61405-0

Ⅰ. ①C… Ⅱ. ①陈… ②王… Ⅲ. ①C语言－程序设计－高等学校－教材 Ⅳ. ①TP312.8

中国版本图书馆CIP数据核字（2018）第280553号

机械工业出版社（北京市百万庄大街22号 邮政编码100037）

策划编辑：和庆娣 责任编辑：和庆娣 胡 静

责任校对：张艳霞 责任印制：张 博

河北鑫兆源印刷有限公司印刷

2019年1月第3版·第1次印刷

184mm×260mm·20印张·490千字

0001－3000册

标准书号：ISBN 978-7-111-61405-0

定价：59.00元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

电话服务

服务咨询热线：（010）88379833

读者购书热线：（010）88379649

网络服务

机 工 官 网：www.cmpbook.com

机 工 官 博：weibo.com/cmp1952

教育服务网：www.cmpedu.com

金 书 网：www.golden-book.com

前　言

本书自 2008 年第 1 版和 2013 年第 2 版发行以来，在计算机基础教学中发挥了应有的作用，深受读者欢迎，对此，作者表示感谢。

图灵奖获得者 Dijkstra 说过：“我们所使用的工具影响着我们的思维方式和思维习惯，从而也将深刻地影响着我们的思维能力”。计算机（包括一切智能设备）无疑是当今人们最为依赖的工具，这就促使我们必须用一种新的思维方式——计算思维，去理解人类的行为，探寻求解问题的方法，设计更多更好的计算机处理算法和程序，从而让计算机更好地为人类服务。培养人们的计算思维能力也是近年来国际学术界和教育界所倡导的一种新的教育理念。要具备这种思维能力，就必须了解计算机，懂得计算机的工作原理，掌握程序设计方法（就这个意义上说，计算思维也可理解为程序思维，或者说，计算思维包含程序思维）。而程序设计方面的训练是其中的重要环节。故而，本次再版力求将这种教育理念更好地融入教学内容当中。

作者根据长期从事算法设计与分析、数据结构、汇编语言和多种高级语言等有关计算机课程教学的经历，总结出这样的经验：要学好用好程序设计语言，必须把握好“记、读、仿、练、操”5 个环节。

记，是指学生先要粗记基本语法和程序框架，然后再通过上机练习，在理解中加以记忆和巩固，起到事半功倍的效果。尤其是对那些内容繁杂的、比较抽象和“绕人”的内容，如输入/输出格式、函数参数传递、指针等不能盲目死记。

读，就是熟读教材中的示例程序，细心体会其中的算法设计方法和程序设计技巧。大多数例题都有一定的代表性和渐近性，需要熟读。

仿，就是在第二步熟读的基础上，多多模仿编写与示例内容相近、结构相近的程序，逐步“仿造”出“好”程序来。

练，就是多做练习题，特别要独立完成程序阅读题和程序填空题。程序阅读题是用人脑模拟计算机，跟踪程序的执行，从而得出答案。跟踪过程对巩固语法规则很有帮助。完成程序填空题，需要弄清程序的功能和大体结构，根据上下文，“猜出”应该填写的语法成分，这对提高学生的程序思维能力大有好处。

操，即上机操作。在纸上编写的程序是“静止”的“死”程序。只有上机操作，才能让程序“动”起来、“活”起来，学会在调试过程中找出程序中的语法错误和逻辑错误。只有学会了在机器上编程并调试，才算真正地学会了编程。

作者正是按照如何在教学中紧扣上述 5 点打造本书的。具体体现于如下几点。

1．全书始终以介绍程序设计方法为主线，语法概念仅作为支撑程序设计的工具，而不是单纯拿语法“说事”。适当地弱化语法概念，缩减单纯语法规则所占篇幅，将语法概念更多地融入例题和习题之中。为此，本书选用了 100 多道例题（包括拓展文件中的例题），分支、

循环、数组等重点内容都配有单独一节程序设计示例，以突出这条主线。另有 500 多道习题与之配合，服务于这条主线，作为正文内容的补充和延伸，为教师留下较大的教学空间，以突出重点，强调应用。

2．将相关内容融合在一起，既体现出共性，压缩了篇幅，也便于归纳和总结。例如，运算符和表达式、构造类型、参数传递方式等。

3．每个例题都经过精心挑选，在多（量大）、经（经典）、精（精巧）、广（涉及面广）的选题指导思想基础上，本着循序渐进的原则，从解题的算法设计思路、算法的自然语言描述、算法的流程图、算法的实现程序、注释、对程序的解释说明等多角度对例题解法、所用到的数据、每一步的实现方法，以及程序结构和主要语句的含义等方面加以剖析与讲解，一步步引导读者进行计算思维，使之容易读懂程序，逐步学会程序设计方法。

4．有的例题先后多次出现，以体现不同的解法和使用不同的语法工具（如是否用数组、是否用函数、是否递归、是否用指针、是否用文件等）。通过这些范例，展示如何用语法描述问题和求解问题。

5．每道习题都经过精心设计，不仅习题量大，题型多（包括一般概念题、选择题、改错题、程序填空题、程序跟踪题、程序设计题等），有层次，而且自成体系，能够起到对教学内容“消化、细化、深化”的作用。在附录部分给出部分习题的参考答案，这些答案大多具有代表性，以冀起到指导作用。

题号前带“△”标记的是作者建议的上机练习题，这些题对巩固课堂知识具有典型的促进作用。多做上机练习是学好程序设计语言的最佳途径。因而，有能力的读者除完成那些带“△”标记的上机练习题之外，不妨从程序设计题中挑选出更多的习题进行上机练习。

6．与前两版的“章习题”模式不同，本版采用“节习题”模式，将独立训练单位由章细化到节，所配备的习题训练点的针对性更强，不仅便于教师布置作业，也便于学生自我练习。

7．书中所有程序（包括例题程序和习题中的程序）都进行了精心设计（并在 Visual C++ 6.0 环境下测试通过），具有良好的程序结构和程序设计风格，以起到示范作用。

8．考虑到 ANSI C（美国国家标准协会发布的 C 语言版本，也称 C89）作为一种优秀的教学语言被长期广泛使用，而单纯的 ANSI C 在现实中已极少用到（通常用 C++代替），故本书以 ANSI C 为主，将 C++的一些“好用的”（相对于基本 C）语法成分（如行注释、cin、cout 和传引用等）穿插其中，使学生从一开始就养成编写结构良好程序的习惯。特别是“传引用”的参数传递方式，对“净化程序”和增强易读性起到“不可代替的”作用。

9．简略介绍 C++的面向对象程序设计机制，使学生了解面向对象程序设计的基本原理，了解面向过程与面向对象之间的关系和差别，为以后掌握面向对象的程序设计方法打下基础。

全书共 6 章，前 4 章是基本内容，也是重点内容。第 5 章（指针）既是重点也是难

点，第 6 章（C++的类和对象）是引申的提高内容。标有“*”的章节为选讲（或略讲）内容。

本书由陈卫卫、王庆瑞编写。微课视频分别由下列教师讲授：张睿（第 1 章）、赵斐（2.1 节）、王家宝（2.2 节）、唐艳琴（第 3 章）、李志刚（第 4 章）、吴永芬（第 5 章）和雷小宇（附录 D、附录 E 和 Codeblocks 用法）。

在此，对曾经为本书的编写、出版给予帮助、支持和鼓励的所有人士表示感谢。特别感谢机械工业出版社陆军工程大学的一贯关注和支持。

对于书中不当甚至错误之处，恳请读者批评、指正。作者联系方式 njcww@qq.com。

编 者

目　录

第1章　C语言基础

本章讲述C语言程序设计的基础知识，其中，程序设计语言的发展和分类、C语言的产生和发展过程、程序设计风格（以及附录A数制和码制），均属于C语言程序设计的外围知识；基本语法成分是学习后续内容的基础，数据类型、运算符和表达式等尤为重要；数据的输入和输出函数以及编译预处理命令不属于本章重点训练内容，但很常用（每个程序都必不可少）。1.1.3节中通过示例展示的算法设计基本方法和程序实现方法，对学习后续内容均有指导作用。

1.1　一般概念

1.1.1　程序设计语言的发展和分类

1．程序设计语言的发展

众所周知，计算机是在程序控制下自动工作的，要让计算机完成某项任务，就必须编写相应的计算机程序。编程所用的语言叫程序设计语言，这种语言人和计算机都能“看懂”，因此，程序设计语言是人指挥计算机的工具。

计算机中直接参与计算的运算器和控制器等部件都是由逻辑电路构成的电子器件，而逻辑部件只“认识”0和1，所以程序的最终形式都是由0和1组成的二进制代码。

二进制代码形式的语言称为机器语言。早在计算机诞生之初，人们就是直接用机器语言编程。但是，这种在计算机看来十分明了的机器语言程序，在人看来却是一部“天书”。后来，人们又将3个二进制位合并在一起变成八进制。再后来，为了与字节对应，又将4个二进制位合并在一起变成十六进制。将机器语言程序写成八进制，或十六进制形式，要比二进制形式“好看”一些。但是，不管二进制、八进制，还是十六进制，用数字表示指令序列都不直观，不仅专业性极强，而且非常难读难用，编程工作效率低，又极易出错。好在当初计算机应用面很窄，编程工作量不大，矛盾并不十分突出。

随着计算机应用面不断地扩大，程序需求量大增，编程工作量也越来越大，人们便产生了用符号代表机器指令的想法，设计出汇编语言（Assemble Language，又称符号语言）。比如，用ADD表示加法指令，用SUB表示减法指令等，要比用二进制0/1序列“00111011”表示某一条指令直观得多。用汇编语言编写的程序称为汇编源程序。将汇编源程序送入计算机，由计算机自动地将汇编源程序翻译成计算机能够直接执行的二进制程序，而计算机的自动翻译功能是由专用软件——汇编程序（Assembler，又称汇编器）完成的，当然，这个软件也是人们事先编写好，并安装在计算机系统中的。一台计算机配上了汇编程序就相当于人们“教会”计算机认识汇编语言了。再后来，人们又设计出反汇编程序，它能将机器语言程序反过来翻译成汇编语言程序。通过反汇编，人们就可以读懂安装在计算机中的可执行程序。

使用汇编语言减轻了不少人们的编程工作量，但是，汇编语言仍然十分原始，一条汇编语句（也称汇编指令）对应一条机器指令，易读性仍然很差。编制一个程序，哪怕只是用来完成简单计算任务的程序，通常需要成百上千条汇编指令。不仅编程效率低，程序不易调试，而且容易出错。更为麻烦的是，这种语言是完全按照计算机硬件设计的，不同种类的计算机都有自己特有的机器语言和汇编语言，一种类型的机器无法识别另一种类型机器的机器语言，所以，汇编源程序缺乏可移植性。

人们把机器语言和汇编语言归属为低级语言。

汇编语言的出现具有划时代的意义，它启发人们，可以设计出更好用的语言，只需要通过翻译器，将新语言的源程序翻译成机器语言程序。于是，人们期待的，脱离计算机机种的高级程序设计语言（以下简称高级语言）便陆续被设计出来。

第一个高级语言是 20 世纪 50 年代中期出现的 FORTRAN（FORmula TRANslator）语言，它的取名就意味着它所起的角色（翻译器）。该语言特别适用于编写科学计算（即纯数值计算）程序，直到今天仍然在发挥着作用，可见其生命力之强大。

后来人们又相继设计出用于商业事务处理的 COBOL 语言（Common Business Oriented Language），适合于算法描述的算法语言 ALGOL（ALGOrithmic Language），面向初学者的 BASIC 语言（Beginner's All-purpose Symbolic Instruction Code）、PL/1 语言（Programming Language/1）以及 Niklaus Wirth 根据结构化程序设计思想设计的 PASCAL 语言（Philips AutomaticSequence CALculator，同时也为了纪念最早实用计算器的发明者、数学家 Blaise Pascal 而命名），以及 C 语言、C++语言和一些更好用、“级别更高”的语言，这些语言在程序设计语言的发展历程中都起到十分重要的作用。

高级语言引入了变量、数组、分支、循环、子程序，以及接近数学语言的表达式等语法成分，用接近英语口语的语句描述处理步骤（例如，if…then…else…），不仅容易理解和记忆，而且一条语句的处理能力相当于几条、几十条，甚至几百条汇编指令，大大地提高了编程效率。

2．程序设计语言的大致分类

高级语言有两种形式，一种是编译型的，另一种是解释型的。

用编译型的高级语言编写的源程序（也称源代码）经过编译程序（Compiler，又称“编译器”）对其编译，产生出机器语言程序（称目标程序），再将一个或几个目标程序与标准库函数程序连接起来，最终构成一个完整的可执行程序。FORTRAN、PASCAL、C 等都属于编译型的语言。

BASIC 是典型的解释型高级语言，采用解释的方法执行源程序中的语句。通过解释程序（Interpreter，又称解释器）对源程序中的语句边解释边执行，而不产生目标程序文件。目前最流行的网络编程语言 Java 也属于解释型高级语言。

编译和解释的最大区别是，前者得到一个完整的机器语言程序，执行时可以脱离翻译环境，所以运行速度快；后者则不能脱离解释器单独执行，因而执行速度慢。

从另一角度，又可将高级语言分为面向过程的和面向对象的两大类。凡提供面向对象程序设计手段的语言都属于面向对象的，否则，属于面向过程的。

早期的高级语言（如 PASCAL、ALGOL 和 C 等）都是面向过程的。这类语言基于数据类型和处理数据的函数（或称过程）定义。数据在其作用域内可被任何函数访问，数据和处

理数据的函数是分离的，缺乏数据保护机制。

面向对象的语言提供的机制支持面向对象的程序设计方法。以类（实际上是类的对象）作为基本单位，把数据以及处理这些数据的代码严密地封装在一起形成类，外部只知有类，但不知（也不必知道）类是如何处理的，只能通过类访问和处理类中的数据，而不能绕过类去访问类中的数据。这样，类内部的处理细节对外是屏蔽的，从而保护了类中的数据，有效地保证数据的完整性和一致性（详见第 6 章）。

20 世纪 60 年代开发的 Simula 67 语言提出了对象和类的概念，被认为是面向对象语言的鼻祖。20 世纪 70 年代出现的 ADA 语言（为纪念第一位有文字记载的女程序员 Augusta Ada Lovelace 而命名）支持抽象数据类型，是基于对象的语言。但由于它并不全面支持继承，所以仍算不得真正面向对象。再后来的 Simulatalk 语言和 Java 语言进一步丰富了面向对象的概念，将信息隐藏得更加严密，通过向对象发送信息的方式进行程序设计。C++语言，以及当前流行的网络编程语言 C#、Python 等都属于面向对象的程序设计语言。

另外，为了满足不同行业、不同人群的需要，人们还设计出各种各样的专用语言（比如，数控语言、各种辅助设计语言、数据库操作语言），以及专门用来软件开发的各种开发工具（有的也属于编程语言）等，这里不能一一列举。

总之，语言的发展促进了编程技术的发展，而编程技术的发展和编程要求同样也激励着编程语言的发展。易学、好用、功能强大，对使用者所应具备的计算机专业知识要求不高，是未来语言的发展方向。

3．C 语言的产生和发展过程

1967 年，Martin Richards 为编写操作系统和编译器开发了一种特殊的语言 Basic Combined Programming Language（缩写为 BCPL）。后来，贝尔实验室的 Ken Thompson 对 BCPL 进行修改，并取 BCPL 的首字母 B 作为名称，即 B 语言。

1972 年，贝尔实验室的 Dennis Ritchie 再次对 B 语言进行修改，取 BCPL 的第二个字母 C 作为修改后的语言名称，即 C 语言，并成功地在 DEC PDP-11 计算机上加以实现（即编写出该语言的编译器）。C 语言继承了 BCPL 语言和 B 语言的优点，克服其不足。继而，贝尔实验室用 C 语言开发 UNIX 操作系统大获成功，引起计算机界高度关注。从此，人们对 C 语言开发低层软件的能力有了充分的认识。

1978 年，Kernighan 和 Ritchie 在 Prentice Hall 出版的《The C Programming Language》一书引起了人们对 C 语言的广泛兴趣，C 语言得以普及和发展。到 20 世纪 70 年代末期，C 语言已逐步发展成今天所说的“传统 C”（也称 Kernighan & Ritchie C）。

1983 年，在美国国家标准委员会（ANSI）下属计算机和信息处理分会（X3）指导下，成立了 X311 技术委员会，该委员会对 C 语言进行了扩充，制定了统一的 C 语言标准（即 ANSI C）。后来又对 ANSI C 进行修订，产生了新的 ANSI C。此后，虽然产生多种不同的 C 语言版本，但它们基本上都符合 ANSI C 标准。

20 世纪 80 年代初，贝尔实验室又在 C 语言基础上，扩充了支持面向对象的程序设计功能，取名为 C++。

历史上比较有影响的 C 语言和 C++语言版本（大多带集成开发环境）有 Microsoft 公司出品的 Microsoft C（简称 MSC），Borland 公司出品的 Turbo C（简称 TC）和 Borland C++（简称 BC），以及 Microsoft 公司出品的 Visual C++（简称 VC）等。

本书以 ANSI C 为基础介绍 C 语言程序设计方法，同时穿插地介绍一些“好用的”的C++语法成分（比如，输入 cin 和输出 cout，行注释符“//”，引用运算符“&”，运算符 new 和 delete 等）。这种以 C 语言为主，并含有一部分 C++语法成分的语言，本书称之为 C/C++语言。为便于叙述，后文中的“C 语言”多数情况下指的就是 C/C++语言。

本书第 6 章简单介绍 C++语言的类和对象的基本用法，以彰显面向过程和面向对象的程序设计的异同之处，使读者对面向对象的程序设计方法有一个粗浅的认识。

4．C 语言的特点

C 语言具有独特的风格，超短而精巧的源程序，灵活多变的 for 语句、break 语句和 continue 语句，“变化多端的”指针操作，超高的编译效率（源程序能被编译出效率非常高的目标代码）。

C 语言具有丰富的数据类型，以及多种多样的运算符和表达式。其中，自增自减运算、位运算、指针运算、取地址运算等与硬指令极为接近，使得 C 语言既具有一般高级语言的特征，又保留了部分汇编语言的特点，将高级语言和低级语言的优点融会在一起（所以特别适于开发底层系统软件）。正因为如此，C 语言被认为是一种带有低级语言色彩的、介于高级语言和低级语言之间的“中级”结构化程序设计语言。

编译预处理功能也是 C 语言的特色之一。

C 语言的语法格式十分灵活（如 for 语句），给程序员提供了极大的自由编程空间。

C 语言也存在不足之处。例如，因语法规定过于灵活而降低程序的易读性，甚至会造成隐蔽性错误；没有复数类型，纯数值计算能力差等。

另外，C 语言运算符的种类繁多，结合性和优先级等概念较复杂，初学者不易全面掌握。

1.1.2 C 源程序的基本结构

视频
1.1.2 C 源程序的基本结构

源程序结构包括物理结构和逻辑结构两个方面。物理结构指的是程序语句和程序块在源程序文件中的排列形式；而逻辑结构指的是程序语句和程序块的执行次序，即程序的流程。

1．物理结构

C 语言将一个独立的程序块称为一个函数。一个 C 源程序文件由一到多个函数组成，这些函数顺序排放在源程序文件中，排放次序并不十分重要（多少有点关系，详见第 4 章）。这种结构属于模块式结构[⊖]。函数是并列的、独立的，而不是嵌套的（一个函数内部不能定义另一个函数）。两个函数之间可以夹杂一些说明性语句，比如，编译预处理命令、全局量的定义或声明等。

每个函数都由一个函数头（包括函数类型、函数名、形参表等）和函数体（用一对花括号括起来的语句序列）组成。

图 1-1 给出 C 源程序物理结构示意图。其中，图 1-1a 是源程序文件结构，图中，双线条代表说明性语句，矩形代表函数；而图 1-1b 则是其中一个函数的结构。

⊖ 有些语言采用嵌套式的层次结构，如 PASCAL。

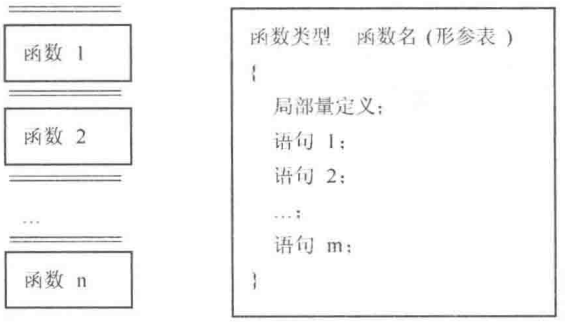


图 1-1 C 源程序的物理结构示意图

a) 源程序文件结构 b) 函数结构

一个完整的 C 源程序（可能由一个或多个源程序文件组成）必须有一个名为 main 的主函数，其余函数均是子函数（子函数名由程序员任取）。运行时总是从主函数的第一条可执行语句开始执行（无论主函数放在什么位置）。仅当被调用时，子函数才得以执行。当执行完主函数的最后一条语句时，这个程序的执行也便终止了。

源程序的任何地方都可以出现注释。

C++提供两种注释方式：行注释和段注释；ANSI C 只提供段注释，不提供行注释。

行注释以“//”开头，到本行末为止的内容均是注释。段注释从“/*”起到“*/”止，中间的内容都是注释。

行注释录入方便，但注释内容不能跨行。段注释则可以跨行，也可以夹在程序语句中间，而且段注释中还可以包含行注释。

对于大段大段的注释内容，若采用行注释，每行开头都要加“//”，所以采用段注释要方便些。

注释不是源程序语句，有无注释对程序执行结果没有丝毫影响，通过注释对程序语句、变量、程序段的功能加以注解，便于自己或他人阅读程序。

本书几乎所有的注释都采用行注释，当然，如果需要，也可改为段注释。

2. 逻辑结构

程序的逻辑结构指的是，同一函数内语句的执行次序，以及各函数的执行次序。

同一函数内的语句总是顺序排列的。执行时也总是从第一条语句开始一条一条地依次执行。但是，当执行一条控制语句（条件语句、转移语句、循环语句等）时，就会改变语句的执行次序，从而改变了程序的流程方向。

有 4 种不同的流程结构：顺序结构、分支结构、循环结构和函数调用结构。其中，前 3 种属于函数内结构，第 4 种是函数间的调用关系。

处在顺序结构中的语句，执行时严格按照语句的先后次序逐条执行，其流程如图 1-2 所示（箭头表示流程方向），即执行次序是：语句 1、语句 2、语句 3。

图 1-2 顺序结构流程图

分支结构有二分支和多分支两种，分别由 if 语句和 switch 语句实现，参见图 1-3。

分支结构的前部通常有一条用于测试判断条件的语句。当程序执行到该语句时，就对事先设置在程序中的判断条件进行测试，根据测试结果，从两个或多个分支中选择一个分支执行（其他分支不被执行）。

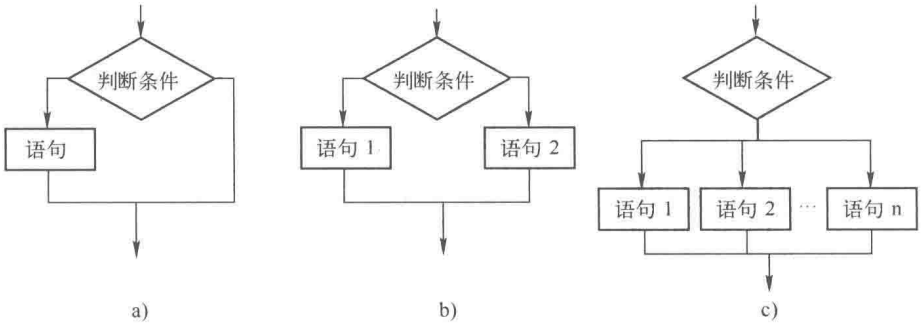


图 1-3　分支结构流程图

a) 二分支结构 1　b) 二分支结构 2　c) 多分支结构

循环结构也称重复结构。处在循环结构中的程序段称为循环体，在循环控制条件的控制之下，循环体可以反复执行。循环结构用于反复处理相同或相似的计算步骤。while 语句、for 语句和 do-while 语句都可以实现循环结构。

有两种循环结构：当型循环和直到型循环，如图 1-4 所示。

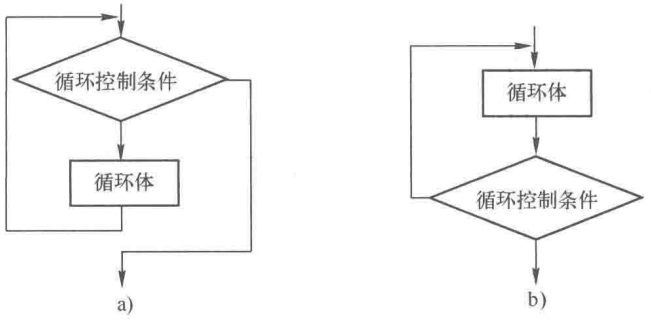


图 1-4　循环结构流程图

a) 当型循环结构　b) 直到型循环结构

当型循环的循环控制条件设在循环结构的前部，其执行流程是：先测试控制条件，然后根据测试结果，确定是否执行（或再一次执行）循环体，其意是当控制条件满足时就执行一次循环体；当控制条件不满足时就退出本循环结构。当然，每执行一次循环体之前都要重新检查一次控制条件。

直到型循环的循环控制条件设在循环结构的尾部，其执行流程是：先执行一次循环体，然后测试循环控制条件，确定是再次执行循环体，还是退出本循环结构。当然，也是每执行一次循环体之后都要重新检查一次控制条件。

当型循环和直到型循环的执行流程唯一区别在于，执行循环体前先测试循环控制条件，还是执行循环体后再测试循环控制条件。

顺序、分支、循环 3 种结构可以穿插出现在同一函数中，而分支结构、循环结构中也可以含有顺序结构、分支结构和循环结构，也就是说，这 3 种结构可以相互叠加，相互嵌套，形成复杂的程序结构。

函数调用可以改变程序的执行路径（即流程方向）。在执行一个函数时，如果遇到函数调用语句，那么控制（流程）从主调函数的调用点转移到被调函数，开始执行被调用函数。执行完被调函数后，返回到主调函数的调用点，继续执行主调函数中调用语句后面的语句。如图 1-5 所示。

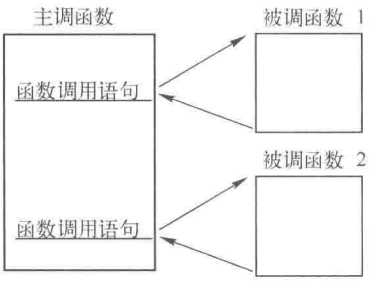


图 1-5　函数调用流程图

主函数可以调用子函数，子函数可以调用其他子函数，也可以调用自身（递归调用）。但任何函数（包括主函数本身）都不能调用主函数。

1.1.3　算法的描述和实现

1．程序和算法的关系


视频
1.1.3　算法的描述和实现

当人们着手求解某给定问题，设计求解此问题的处理程序时，先要确定解题方案，从中抽象出问题所涉及的数据及数据结构，设计出求解算法（即如何对这些数据一步步地进行处理，最后得到问题所需要的计算结果），然后按照算法中描述的计算步骤编写计算机程序（这一步称作算法的实现，而前一步则是算法的描述）。这就是著名的 Wirth 公式“算法+数据结构=程序”所概括的算法、数据结构和程序之间的关系。

按照这一角度，算法就是对计算步骤的描述，而程序则是实现计算步骤的（计算机可以执行的）指令序列，显然，程序中必含有算法。因此，有时候将算法称为程序，或将程序称为算法。

算法有多种描述形式，其中最常用的是自然语言描述形式（即文字叙述形式），以及下面介绍的流程图形式。

2．流程图

流程图也称框图，因为算法是程序的雏形，所以算法的流程图也就是程序的流程图。画流程图时要使用规范的流程图符号。

画流程图时要使用规范的流程图符号。

图 1-6 给出常用的流程图符号。其中，椭圆（图 1-6a）用于标注程序的起点（起始框）和结束点（终止框），框内通常书写“开始”和“结束”；矩形（处理框，图 1-6b）内书写程序的处理步骤；菱形（决策框，图 1-6c）用于书写程序分支或循环的判断条件；平行四边形（输入/输出框，图 1-6d）内书写程序的输入和输出数据；带箭头的线条（流程线，图 1-6e）用于指明程序的流程方向；小圆点（旁边带字母等记号，图 1-6f）用于指明流程图的连接点，当流程图太大，一处画不完时，通过连接点转到另一处再画，记号相同的，表示同一个流程点。