普通高等教育“十二五”规划教材
高等学校公共课计算机规划教材

C语言程序设计教程
（第3版）

■ 张敏霞　孙丽凤　王秀鸾　主编
■ 迟春梅　副主编

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY　[http://www.phei.com.cn]

高等学校公共课计算机规划教材

C 语言程序设计教程

（第 3 版）

张敏霞　孙丽凤　王秀鸾　主编

迟春梅　副主编

罗　容　白清华　编

電子工業出版社

Publishing House of Electronics Industry

北京 · BEIJING

内 容 简 介

本书内容分为基础篇、提高篇和实验篇3部分。基础篇的主要内容包括程序设计和C语言基础知识，顺序、选择和循环结构程序设计，以及编译预处理，使读者初步建立起利用C语言进行简单程序设计的思想，学会进行简单的程序设计。提高篇的主要内容包括数组等构造型数据类型、指针类型、函数、位运算以及对文件的操作，使读者学习并体会C语言模块化的编程思想及对数组、指针类型的应用，学会使用构造型数据类型和指针类型处理问题，学习对文件进行操作。实验篇设计了9个实验，以加强编写程序的实战能力。

本书在编写时兼顾了全国计算机等级考试的要求。书中例题丰富，注重实用，且均在Visual C++ 6.0环境下调试通过。各章都配有丰富的习题。本书程序源代码、配套课后习题指导、参考答案和教学用电子课件，请通过华信教育资源网免费索取。

本书可作为高等学校本科、高职高专计算机专业及相关专业程序设计的入门教材，也可作为全国计算机等级考试的辅导教材，还可供广大程序设计初学者自学使用。

图书在版编目 (CIP) 数据

C语言程序设计教程 / 张敏霞，孙丽凤，王秀鸾主编. —3版. —北京：电子工业出版社，2013.10
高等学校公共课计算机规划教材
ISBN 978-7-121-21351-9

I. ①C… II. ①张… ②孙… ③王… III. ①C语言—程序设计—高等学校—教材 IV. ①TP312

中国版本图书馆CIP数据核字(2013)第206136号

策划编辑：王羽佳
责任编辑：王羽佳　　特约编辑：曹剑锋
印　　刷：北京天宇星印刷厂
装　　订：三河市鹏成印业有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路173信箱　　邮编：100036
开　　本：787×1092　1/16　印张：17.5　字数：510千字
印　　次：2013年10月第1次印刷
印　　数：5000册　　定价：39.00元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：(010)88254888。

质量投诉请发邮件至zlts@phei.com.cn，盗版侵权举报请发邮件至dbqq@phei.com.cn。

服务热线：(010)88258888。

前　言

计算机程序设计基础是高等学校各专业开设的一门必修计算机基础课程，课程的重点在于培养学生的程序设计思想和程序设计能力，以适应当今社会对人才的需求。C 语言由于自身的简洁、紧凑和灵活性的特点，以及具备其他高级语言所不具备的低级语言的特性，而使得它成为一种在计算机软件设计和计算机程序设计教学中备受欢迎的程序设计语言。

本书集结了众位编者多年的教学经验和应用 C 语言的体会，根据教育部高教司非计算机专业计算机教学指导委员会提出的高等学校计算机基础课程教学基本要求，在广泛参考有关资料的基础上编写而成。

本书内容分为基础篇、提高篇和实验篇。

基础篇分成 6 章，主要内容包括程序设计及 C 语言基础知识，顺序、选择、循环结构程序设计，以及编译预处理等，使学习者初步建立起利用 C 语言进行简单程序设计的思想，学会进行简单程序设计。相对于第 2 版，本书增加了编译预处理的内容，让学习者尽早理解文件包含和宏定义的思想及其在程序设计中的应用。

提高篇分成 6 章，主要内容包括数组等构造型数据类型、指针类型、函数、位运算以及文件的操作等，使学习者理解并体会 C 语言模块化的编程思想以及对数组、指针类型的应用，学会使用构造型数据类型和指针类型处理问题，学会对文件进行操作。相对于第 2 版，本书调整了部分章节的顺序，首先介绍数组和指针这两种新的数据类型，接着介绍函数的内容，在“函数”一章中突出介绍模块之间信息的传递。

实验篇设计了 9 个综合性实验项目以开拓学习者的思路，激发学习者的学习兴趣。

同时，本书在编写时兼顾了全国计算机等级考试的要求。

本书在结构组织上符合学习逻辑，内容循序渐进，每个知识点的介绍都以引起学生的学习热情和兴趣为出发点，以提高学生的程序设计思想和能力为目标，既注重理论知识，又突出实用性。书中例题丰富，注重实用，且均在 Visual C++ 6.0 环境下调试通过。各章配有丰富的习题，以帮助读者深入理解教材内容，巩固基本概念，达到培养良好的程序设计能力和习惯的目的。

本书可作为高等学校本科、高职高专计算机专业及相关专业的程序设计的入门教材，也可作为全国计算机等级考试的辅导教材，还可以供广大程序设计初学者自学使用。

本书具有以下特点：

① 突出算法理解，重视实际操作。

② 加强对学生程序设计思想和实际编程能力的培养，以适应信息社会对人才的需求。

③ 提供多样式的学习环境。本书同时提供丰富的教学资源，包括课程学习网站（http://211.64.192.109/jpkc）、程序源代码、配套教学电子课件及习题指导与参考答案，以上资源向使用本书作为教材的教师免费提供，请通过华信教育资源网 www.hxedu.com.cn 索取。

④ 注重可读性。本书的编写小组由具有丰富教学经验的、多年来一直从事计算机基础教育的一线资深教师组成，内容组织合理，语言使用规范，符合教学规律。

本书的编写得到了青岛理工大学各级领导的关心和支持，在此一并表示深深的感谢。

由于作者水平有限，书中难免有误漏之处，敬请专家、教师和广大读者批评指正。

编　者

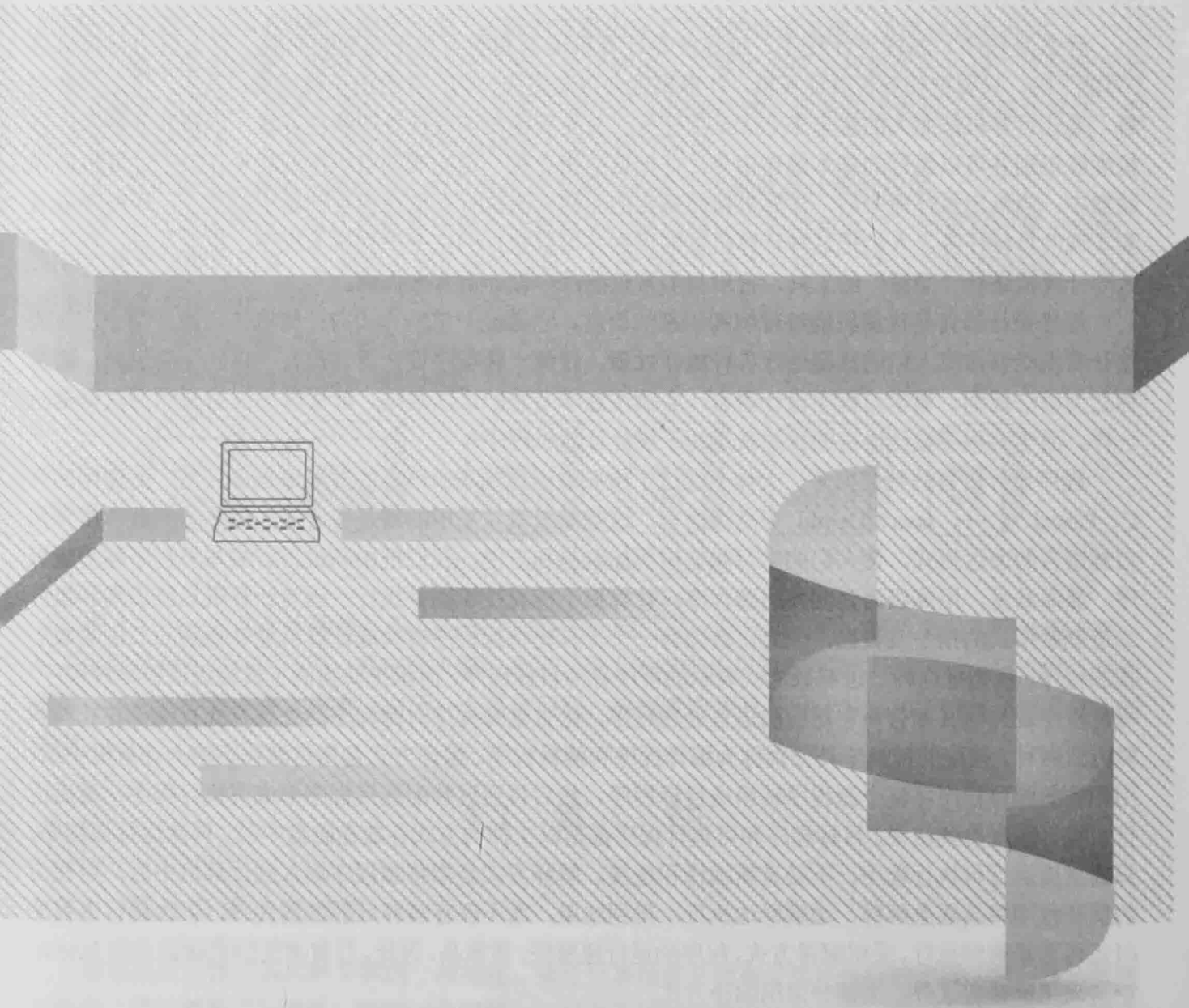
目　录

基础篇

提高篇

实验篇

基　础　篇



第1章　程序设计及C语言概况

本章主要介绍程序设计的基本概念、算法描述及结构化程序设计方法、C语言的起源与发展、C语言的特点、简单的C语言程序和在Visual C++ 6.0环境下C语言程序的开发过程。通过本章的学习，读者可对程序设计的基本概念和C语言有一个大致的了解。

1.1　程序设计的基本概念

1.1.1　程序和程序设计语言

什么是程序？怎样设计程序？这往往是计算机语言初学者首先遇到的问题。有人以为计算机是“万能”的，只要把任务告诉计算机，计算机就会自动完成一切，并给出正确结果。其实，这是一种误解，要让计算机按照人的意志来完成某项任务，首先要编制该项任务的解决方案，再将其分解成计算机能够识别并可以执行的基本操作指令，并把这些指令按一定的规则组织排列起来存放于计算机内存储器中，当给出执行命令后，计算机按照规定的流程依次执行存放在内存储器中的指令，最终完成所要实现的目标任务。人们把这种计算机能够识别并可以执行的指令序列称为程序。也就是说，程序是人与计算机进行“交流”的工具，它用我们常说的程序设计语言来描述。

程序设计语言是计算机能够理解和识别的语言。它通过一定的方式向计算机传送操作指令，从而使计算机能够按照人们的意愿进行各种操作处理。任何一种程序设计语言都有一定的使用规则，通常称之为语法规则。要学习程序设计语言，必须注意学习它的语法规则，就像学习汉语要学习汉语语法一样。而学习程序设计语言的目的就是为了设计计算机程序。

程序设计语言的种类很多，大体上经过了由低级语言到高级语言的发展过程，目前广泛使用的有C、Pascal、C++、Java、Delphi 等高级语言，这些高级语言采用的都是接近于人们熟悉的数学语言和自然语言的表达形式，使人们的学习和使用更加容易和方便。我们把由高级语言编写的程序称为源程序。显而易见，用高级语言编写的源程序，计算机不能直接识别并执行，因为计算机只能识别和执行二进制数形式的指令或数据，因此，必须有一个工具先将源程序转换成计算机能够识别的二进制数形式的程序，我们把这种二进制数形式表示的程序称为目标程序，而承担转换的工具称为语言处理程序。每种程序设计语言都有与它对应的语言处理程序。语言处理程序对源程序的处理方式有编译方式和解释方式两种，相应的转换工具分别称为编译程序和解释程序。编译方式是指将源程序输入计算机中后，用相应的编译程序将整个源程序转换成目标程序，然后再通过装配连接程序形成可执行程序，最后运行可执行程序得到结果。目标程序和可执行程序都是以文件的方式存放在磁盘中的，再次运行该程序，只需直接运行可执行程序，不必重新编译和连接。解释方式是指将源程序输入到计算机中后，用相应的解释程序将其逐条解释，边解释边执行，得到结果，而不保存解释后的机器代码，下次运行该程序时还要重新解释执行。采用编译方式，程序的运行速度快，效率高。因此，目前常用的高级语言除BASIC语言采用解释方式外，大部分采用编译方式。

1.1.2　程序设计

对于初学者来说，往往把程序设计简单地理解为只是编写一个程序，这是不全面的。程序设计是指利用计算机解决问题的全过程，它包含多方面的内容，而编写程序只是其中的一部分。利用计算机

解决实际问题，通常先要对问题的性质与要求进行深入分析并确定求解问题的数学模型或方法，然后考虑数据的组织方式和算法，并用某一种程序设计语言编写程序，最后调试程序，使之运行后能产生预期的结果，这个过程称为程序设计。程序设计的基本目标是实现算法和对初始数据进行处理，从而完成对问题的求解。

有些初学者，在没有把所要解决的问题分析清楚之前就急于编写程序，结果编程思路混乱，很难得到预期的效果。因此，为了用计算机解决一个实际问题，作为设计人员，从拿到任务到得出正确结果，往往要经过以下 6 个设计阶段。

① 分析问题。即分析问题需求，也就是弄清楚该问题有哪些已知数据，程序运行需要输入什么数据，需要输出什么结果，需要进行哪些处理等。

② 确定处理方案。如果是数学问题，就要根据该问题的数学解法，考虑所用的数学公式或相关函数；如果是工程问题，就要先建立该问题的数学模型，把工程问题转化成数学问题，以便用计算机解决。对同一个问题可以用不同的方案来处理，不同的方案决定了不同的处理步骤，效率也有所不同。

③ 确定操作步骤。根据选定的处理方案，具体列出让计算机如何进行操作的步骤。这种规定的操作步骤称为算法，而这些操作步骤之间的执行顺序就是控制结构。通常使用流程图来描述算法，把算法思想表达清楚，比较简单的问题可直接进入编写程序阶段。

④ 根据操作步骤编写源程序。用计算机语言编写的操作步骤就是计算机程序。

⑤ 运行调试程序。将计算机程序输入计算机中，经过编译、连接和运行，如果程序是正确的，应该能得到预期的结果。如果得不到正确的结果，应检查程序是否有错误，改正后再调试运行，直到得出正确的结果为止。

⑥ 整理输出结果，写出相关文档。

图 1.1 所示为程序设计的一般过程。

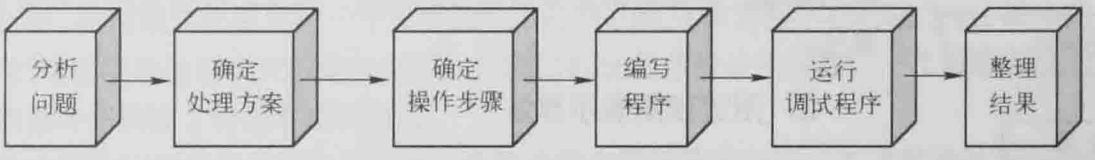


图 1.1　程序设计的一般过程

图 1.1 中，前两个步骤类似于人们解决问题的一般过程，即分析问题，然后确定一种处理方案。后 4 步则是程序设计的环节，其中最关键的是第③步“确定操作步骤”，或称算法设计。只要算法是正确的，编写程序就不会太困难。对于一个具体问题，编程者应该具备设计算法和正确使用已有算法的能力。

1.2　算　　法

1.2.1　算法及算法的特性

算法是对具体问题求解步骤的一种描述。做任何事情都必须事先想好执行的步骤，然后按步骤进行操作，才能避免产生错乱。对同一个问题，可以有不同的解题方法和步骤。有的方法需要的步骤很少，而有些方法需要的步骤则较多。一般来说，希望采用过程简单明了和思路清晰正确的方法。因此，为了有效地解题，一个算法应具有下列 5 个特性。

① 有穷性。一个算法必须在执行有限个操作步骤之后结束，且每步都可以在有穷时间内完成。

② 确定性。算法中的每条指令必须有确切的含义，不会产生二义性。

③ 可行性。算法中描述的操作在计算机中都是可以实现的。

④ 输入。一个算法应有零个或多个输入。

⑤ 输出。一个算法应有一个或多个输出。

1.2.2 算法的描述工具

为了使算法表达得更清晰，更容易实现算法的编写，在程序设计时通常使用专门的算法表达工具对算法进行描述。对于复杂的问题，可以先用算法表达工具对算法进行描述，再进行编程，算法的最终实现应该是计算机程序。算法的评价标准涉及很多方面，但正确性和清晰易懂性永远是一个好算法的基本条件。

计算机算法的表达工具通常有以下4种。

1. 用自然语言表示算法

用自然语言表示算法就是把算法的各个步骤用人们所熟悉的自然语言依次表示出来。但要注意，用自然语言所表示的每个操作步骤必须是计算机能够实现的。

【例1.1】 求两个整数 m 与 n 的和，用自然语言求解该问题的步骤如下。

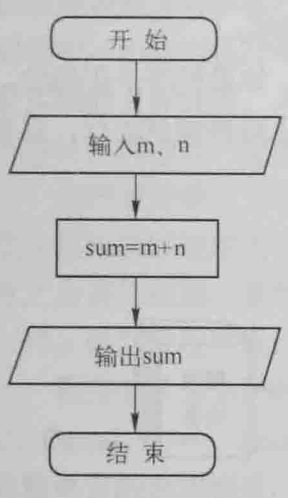


图1.2 例1.1算法

步骤1：输入整数 m 和 n。

步骤2：求和 sum=m+n。

步骤3：输出两数之和 sum。

以上3个步骤都是在计算机中可以实现的，所以是一个正确的算法描述。用自然语言表示算法，人们比较容易理解，但书写较烦琐，而且在某些场合，由于自然语言含义的不确切性，容易引起歧义，造成误解；另外，对比较复杂的问题，用自然语言又难以表达准确。因此，较少采用自然语言表示复杂算法。

2. 用流程图表示算法

用流程图表示算法就是用一些大家共识的专用图形符号和带有箭头的流程线来表示算法。用图形符号表示算法必须有一组统一规定的、含义确定的专用符号。表1.1所示为传统流程图所用的基本符号。图1.2所示为例1.1中求两个整数 m 与 n 之和的传统流程图算法。

表1.1 传统流程图所用的基本符号表

图形符号	符号名称	说 明
	起始、终止框	表示算法的开始或结束
	输入、输出框	框中标明输入、输出的内容
	处理框	框中标明进行什么处理
	判断框	框中标明判定条件，并在框外标明判定后的两种结果的流向
	流程线	表示从某一框到另一框的流向
	连接点	表示算法流向出口或入口连接点

可见，用流程图来表示算法，直观形象，绘制简单方便，流程清晰，各种操作一目了然，而且不会产生“二义性”。缺点是，占用面积大，由于使用流程线指出各框的执行顺序，且对流程线的方向没有任何限制，可以随意转向，往往使人弄不清楚流程的思路。

针对上述问题，1973 年由美国计算机科学家 I.Nassi 和 B.Shneiderman 提出了结构化程序设计流程图，又称 N-S 流程图。N-S 流程图保留了传统流程图可以形象直观地表示算法的优点，去掉了流程线，即不允许流程任意转移，只能从上到下顺序进行，算法的每步都用一个矩形框来描述，把这些矩形框按执行的次序连接起来就是一个完整的算法。这种流程图规定了几种基本结构作为构造算法的基本单元，将在 1.3 节中结合 3 种基本结构来介绍。

3. 用伪代码表示算法

所谓伪代码是指用介于自然语言和程序设计语言之间的一种代码来描述算法。它的表示形式比较自由灵活，而且由于与程序设计语言比较接近，因此可以比较容易地转换成计算机程序。伪代码无统一的语法，只要自己或别人能看懂就行。

4. 用程序设计语言表示算法

用自然语言和用图形符号所表示的算法有一个共同的缺点就是计算机都不能识别和执行。只有用计算机能理解和执行的程序设计语言把算法表示出来，然后把程序输入计算机中并执行，计算机才能按照预定的算法去解决问题。

1.3　结构化程序设计方法

结构化程序设计是指，为使程序具有一个合理的结构以保证程序正确性而规定的一套如何进行程序设计的原则。结构化程序设计的原则是：采用自顶向下、逐步求精的方法；程序结构模块化，每个模块只有一个入口和一个出口；使用 3 种基本控制结构描述程序流程。其中，模块化是结构化程序设计的重要原则。所谓模块化就是把一个大型的程序按照功能分解为若干相对独立的、较小的子程序（即模块），并把这些模块按层次关系进行组织。按照结构化程序设计的原则，一个程序只能由顺序结构、选择结构和循环结构这 3 种基本结构组成。

人们解决复杂问题普遍采用自顶向下、逐步求精和模块化的方法，在这种设计方法的指导下开发出来的程序，具有清晰的层次结构，容易阅读和维护，软件开发的成功率和生产率可极大提高。因此，使用结构化方法设计出的程序等于数据结构加算法。

已经证明，任何复杂的算法都可以由顺序、选择、循环这 3 种结构组合而成。所以，这 3 种控制结构称为程序的 3 种基本控制结构。

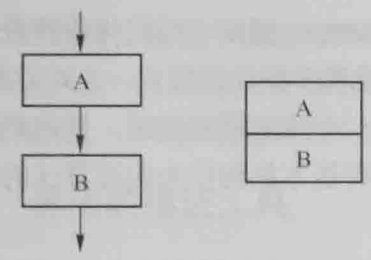
1. 顺序结构

顺序结构如图 1.3 所示，图（b）是 N-S 流程图。其中 A 和 B 是顺序执行的关系，即先执行模块 A 操作，再执行模块 B 操作。

图 1.2 就是例 1.1 中求两个整数 m 与 n 之和的传统流程图结构。它只需顺序结构就能解决问题。

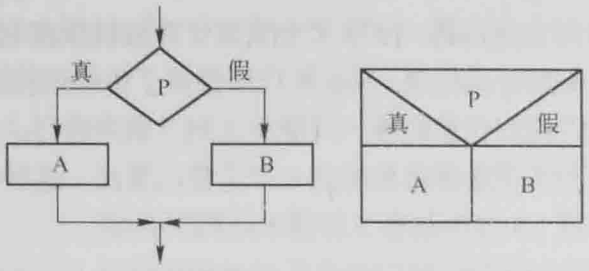
2. 选择结构

选择结构又称为分支结构，如图 1.4 所示，图（b）是 N-S 流程图。其中，P 代表一个条件，当条件 P 成立时（或称为“真”时），执行模块 A，否则执行模块 B。注意，只能执行 A 或 B 之一，两条路径汇合在一起结束该分支结构。通过下面的例子，读者可以了解如何用自然语言、N-S 流程图描述分支结构。



(a) 传统流程图　(b) N-S流程图

图 1.3　顺序结构



(a) 传统流程图　(b) N-S流程图

图 1.4　选择结构

【例 1.2】　求 a、b 两个整数中较小的数。

用自然语言求解该问题的步骤如下。

步骤 1：输入整数 a 和 b。

步骤 2：进行判断，如果 $a<b$，则 min=a，否则 min=b。

步骤 3：输出两个数中较小的数 min。

用 N-S 流程图求解该问题的过程如图 1.5 所示。

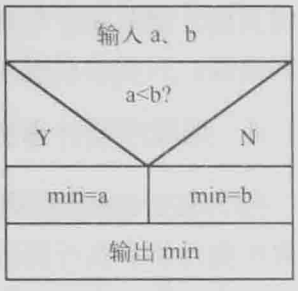
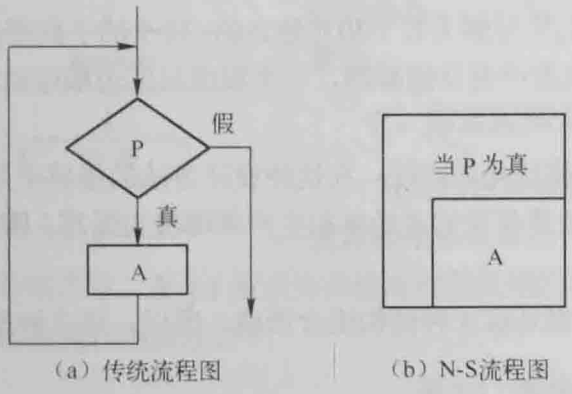


图 1.5　例 1.2 算法

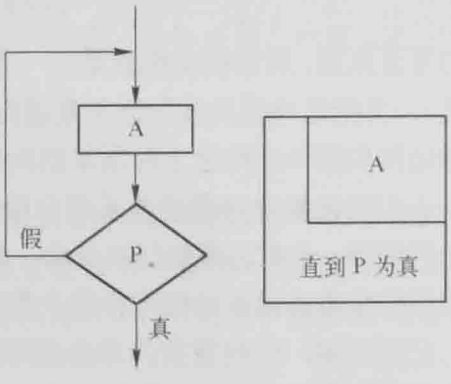
3. 循环结构

循环结构又称为重复结构，有两种循环形式。一种是当型循环结构，如图 1.6 所示。其中，P 代表一个条件，当条件 P 成立（为“真”）时，反复执行模块 A 操作，直到 P 为“假”时才停止循环。另一种是直到型循环结构，如图 1.7 所示。先执行模块 A 操作，再判断条件 P 是否为“假”，若 P 为“假”，再执行 A，如此反复，直到 P 为“真”为止。



(a) 传统流程图　(b) N-S流程图

图 1.6　当型循环结构



(a) 传统流程图　(b) N-S流程图

图 1.7　直到型循环结构

下面通过例题，使读者了解如何用自然语言、N-S 流程图描述循环结构。

【例 1.3】　计算 1+2+3+4+…+100。

用自然语言求解该问题的步骤如下。

步骤 1：定义变量 sum 用来存放和值，并将初值 0 赋给 sum，使 sum 的值为 0；定义变量 k，用来存放每项的值，并将 1 赋给 k。

步骤 2：判断 k 的值是否小于或等于 100，如果是，则继续执行步骤 3，否则转到步骤 5，退出循环。

步骤 3：将 sum 与 k 的和赋给 sum。

步骤 4：将 k 的值增 1，返回步骤 2 重复执行。

步骤 5：输出和值 sum。

用 N-S 流程图求解该问题的过程如图 1.8 所示。

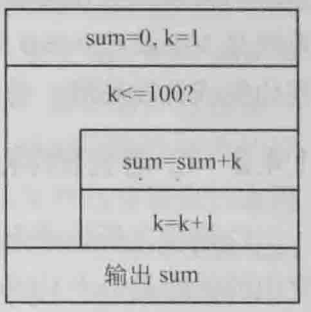


图 1.8　例 1.3 算法

可以看到，3 种基本控制结构共有的特点是：有一个入口，有一个出口；结构中每一部分都有被执行到的机会，也就是说，每一部分都有一条从入口到出口的路径通过它（至少通过一次）；没有死循环（无终止的循环）。

结构化程序要求每个基本控制结构具有单入口和单出口的性质是非常重要的，这是为了便于保证和验证程序的正确性。在设计程序时，一个结构一个结构顺序地写下来，整个程序结构如同砌墙一样顺序清楚，层次分明；在需要修改程序时，可以将某个基本控制结构单独取出来进行修改，由于其具有单入口单出口的性质，不会影响到其他的基本控制结构。可以把每个基本控制结构看作一个算法单位，整个算法则由若干个算法单位组合而成。这样的算法称为结构化算法。而这样设计出的程序清晰易读，可理解性好，容易设计，容易验证其正确性，也容易维护。同时，由于采用了“自顶向下、逐步细化”的实施方法，能有效地组织人们的思路，有利于软件的工程化开发，提高编程工作的效率，降低软件的开发成本。

1.4　C 语言的初步知识

1.4.1　C 语言的起源与发展

C 语言诞生于 1972 年，由美国电话电报公司（AT&T）贝尔实验室的 D.M.Ritchie 设计，并首先在一台使用 UNIX 操作系统的 DEC PDP-11 计算机中实现。

C 语言是在 B 语言的基础上发展起来的，早在 1970 年，美国贝尔实验室的 K.Thompson 就以 BCPL 语言（Basic Combined Programming Language）为基础，设计出一种既简单又接近于硬件的 B 语言，并用它编写了第一个 UNIX 操作系统。而 BCPL 语言是英国剑桥大学的 Matin Richards 于 1967 年基于 CPL 语言（Combined Programming Language）提出的一种改进语言。CPL 语言又是英国剑桥大学于 1963 年根据 ALGOL 60 推出的一种接近硬件的语言。由此可见，C 语言的根源可以追溯到 ALGOL 60，其演变过程如下：

ALGOL 60（1960 年）→CPL（1963 年）→BCPL（1967 年）→B（1970 年）→C（1972 年）

C 语言问世以后，在应用中多次进行了改进。1973 年贝尔实验室的 K.Thompson 和 D.M.Ritchie 用 C 语言将 UNIX 系统（即 UNIX 第 5 版）重写了一遍，增加了多道程序设计功能，使整个系统，包括 C 语言的编译程序都建立在 C 语言的基础上。第 5 版 UNIX 系统（UNIX V5）奠定了 UNIX 系统的基础。到 1975 年，UNIX 第 6 版问世。随着 UNIX 的巨大成功和被广泛移植到各种机器上，C 语言也被人们所接受，并移植到大型、中型、小型和微型机上，它很快风靡全世界，成为世界上应用最广泛的计算机程序设计语言之一。

1978 年又推出了 UNIX 第 7 版，以该版本中的 C 语言编译程序为基础，B.W.Kernighan 和 D.M.Ritchie 合作（被称为 K&R）出版了“The C Programming Language”一书。该书介绍的 C 语言被称为标准 C，这本书成为后来被广泛使用的 C 语言的基础。1983 年，美国国家标准化协会（ANSI）对 C 语言的各种版本做了扩充和完善，推出了新的标准，被称为 ANSI C，它比原来的标准 C 有了很大的改进和发展。1987 年，ANSI 又公布了 87 ANSI C 新版本。当前流行的各种 C 语言编译系统都是以 87 ANSI C 为基础的。在微机上使用的 C 语言编译系统多为 Microsoft C、Turbo C、Borland C 和 Quick C 等，它们略有差异，但按标准 C 语言书写的程序，基本上都可运行。读者若要了解不同版本的编译

系统的特点和规定可参阅有关手册。目前，全国计算机等级（二级 C 语言）考试中采用的 C 语言编译系统是 Visual C++ 6.0（简称 VC）。Visual C++ 6.0 是一个集程序编辑、编译、连接和运行为一体的可视化集成开发环境，是计算机界公认的最优秀的应用开发工具之一。

1.4.2 C 语言的特点

C 语言之所以成为世界上应用最广泛、最受人们喜爱的计算机高级语言之一，与它本身所具有的突出的优点是分不开的。C 语言的主要特点概括如下。

① 语言简洁、紧凑，使用方便、灵活。C 语言一共只有 32 个关键字，9 种控制语句。程序书写形式自由，主要用小写字母表示。

② 支持结构化程序设计。C 语言具有结构化的控制语句，以函数作为程序的模块单位，这使得它可以很方便地实现这种构造。

③ 运算符丰富。C 语言除了拥有一般高级语言都有的运算符外，还拥有不少独特的运算符，可以实现其他高级语言不能实现的运算，增强了 C 语言的运算功能。

④ 数据类型丰富。C 语言能方便地实现各种复杂的数据结构，具有很强的数据处理能力。

⑤ 较强的编译预处理功能。C 语言的编译预处理功能，为开发规模较大的程序提供了方便，极大地提高了程序开发的效率。

⑥ C 语言的可移植性好。C 语言本身只需稍加修改便可用于各种型号的计算机和各类操作系统，因此，用 C 语言编写的程序也可以很方便地用于不同的系统，这也是 C 语言得以广泛应用的原因之一。

⑦ C 语言本身既有一般高级语言的优点，又有低级（汇编）语言的特点。C 语言可以允许直接访问内存地址，可以进行位（bit）运算，也可以直接对机器硬件进行操作。因此，既可以用它来编写大型系统软件，也可以用它编写各种应用软件，许多以前只能用汇编语言处理的问题，现在可以改用 C 语言处理了。

⑧ 语法限制不太严格，程序设计自由度大。C 语言由于放宽了语法检查，使程序员有较大的自由度。例如，对数组下标越界不做检查，要由程序设计人员自己保证其正确性。因此，用 C 语言编写程序，对程序设计人员的要求相应地要高一些。

上述 C 语言的特点，在初学时也许还不能深刻理解，待学完 C 语言之后就会有比较深的体会和感受。

1.4.3 C 语言程序的构成

下面通过两个简单的 C 语言程序，使大家能够对 C 语言程序有一个最基本的认识。

【例 1.4】 求两个整数 m 与 n 的和。

程序中，m 和 n 分别表示两个整数，sum 表示两个整数的和。

程序代码如下：

```
#include  "stdio.h"                  /* 文件包含命令，将头文件 stdio.h 包含进来 */
main()                               /* 主函数 */
{  int  m,n,sum;                     /* 定义变量 m, n, sum*/
   m=5;                              /* 给变量 m 赋值 5 */
   n=3;                              /* 给变量 n 赋值 3 */
   sum=m+n;                          /* 求 m+n 的值，并赋给变量 sum */
   printf("sum is %d \n",sum );      /* 输出 sum 的值 */
}
```

以上程序由一个称为主函数的 main( )函数构成。C 语言规定必须用 main 作为主函数名，其后的一对圆括号“( )”中间可以是空的，但这一对圆括号不能省略。主函数后面是函数体，由一对花括号“{ }”括起来，左花括号“{”表示函数开始，右花括号“}”表示函数结束。“/* …… */”表示注释

部分，它只是起到帮助阅读程序的作用，对编译和运行不产生任何影响。为方便理解，可使用汉字来进行注释，注释部分可以出现在程序的任何地方。程序第 3 行是变量定义语句，定义 m、n 和 sum 为整型（int）变量；程序第 4、5 行是两条赋值语句，使 m 和 n 的值分别为 5 和 3；程序第 6 行也是一条赋值语句，使 sum 的值为表达式 m+n 的值；程序第 7 行是输出语句，其中，printf( )是 C 语言的标准输出函数（当引用 C 语言的标准函数时，需在程序开始处将该函数所对应的头文件包含进来。本程序第 1 行就是该函数所对应的头文件），双引号中的%d 表示在此位置以十进制整数类型输出变量 sum 的值，\n 为回车换行符，双引号中其他内容将作为字符串原样输出。因此，该程序的运行结果如下：

```
sum is 8
```

【例 1.5】　求两个整数中的较小者。

程序中，x、y 分别表示两个整数，min 表示两个整数之中的较小值。

程序代码如下：

```
#include "stdio.h"
main()                              /* 主函数 */
{  int  x,y,min;                    /* 定义变量 */
   int  fun(int a,int b);           /* 当定义的函数在主函数之后时,要进行函数的声明*/
   printf("input x,y:");            /* 提示输入数据 */
   scanf("%d,%d", &x,&y);           /* 输入变量 x 和 y 的值 */
   min=fun(x,y);                    /* 调用 fun 函数,将 fun 函数的返回值赋给 min */
   printf("min=%d\n",min);          /* 输出 min 的值 */
}
int fun(int a,int b)                /* 定义 fun 函数,值为整型,a 和 b 为该函数的形式参数*/
{  int c;                           /* 函数中用到的变量 c 也要定义 */
  if(a<b)  c=a;                     /* 比较 a 和 b 的大小，将较小者赋给 c */
  else  c=b;
  return(c);                        /* 将 c 的值返回至调用处 */
}
```

该程序包括两个函数：主函数 main( )和用户自定义函数 fun( )。程序的第 6 行是输入语句。scanf( )是 C 语言的标准输入函数，它的作用是让用户从键盘上输入数据；双引号中的两个%d 是格式说明，表示用户输入的数据应该是两个整数；&x 和&y 中&的含义是取地址，它表示用户从键盘上输入的两个整数将分别送到变量 x 和 y 所对应的存储单元中，也就是输入给变量 x 和 y。第 7 行调用 fun()函数，调用时将 x 和 y 的值传送给被调函数 fun()中的形参 a 和 b，在被调函数 fun()中，求出两个数中的较小值赋给变量 c，最后由 return 语句返回 c 的值到主函数 main( )中的调用处，并将返回的值送给变量 min。

该程序的运行情况如下：

```
input x,y:10,2<Enter>
min=2
```

以上两例中，关于标准输入/输出函数的使用、函数调用、实参和形参的概念，大家可能还不十分清楚，可以先不予以深究，今后在有关章节中还要详细介绍。分析上述程序可以总结出 C 语言程序的基本构成如下。

（1）C 语言程序由函数构成

一个 C 语言程序至少应包含一个 main( )函数（如例 1.4），或者包含一个 main( )函数和若干个用户自定义函数（如例 1.5）。因此，函数是 C 语言程序的基本单位，相当于其他语言的子程序或过程。

C 语言的函数可以分为系统提供的标准函数（库函数，如例 1.5 中的 printf( )和 scanf( )函数）和

用户自定义函数（如例1.5中的fun( )函数）。各种C语言的编译系统所提供的标准函数的数量和功能以及函数名都不完全相同，标准C提供了100多个标准函数，Turbo C提供了300多个标准函数，函数调用使C语言很容易实现程序的模块化。

为了增加程序的可读性，可以用“/*……*/”在任何位置上对C语言程序的任何部分进行注释，一般在一个程序或函数的开始或某些程序的难点之处加上必要的注释（在Visual C++ 6.0环境下也可使用符号“//……”引出注释）。

（2）一个C语言函数通常由两部分组成：函数的首部和函数体

函数的首部包括函数类型、函数名、一对圆括号、函数参数（形参）名和参数类型的说明。如例1.5中fun( )函数的首部如图1.9所示。

图1.9 例1.5中fun( )函数的首部

注意：C语言规定，一个函数名后面必须紧跟一对圆括号，函数的所有形参都必须写在括号内。如果函数没有形参，则括号内可以是空的，但不能省略圆括号，例如，主函数main( )经常不带参数。

函数体就是函数首部后面一对花括号{}内的部分。如果一个函数内有多对花括号，则最外面的一对花括号{}所包含的内容就是该函数的函数体。函数体一般包括说明部分和执行部分。

① 说明部分

说明部分由若干条变量定义语句或函数声明语句组成。C语言规定，函数中使用的所有变量（或数组）必须在使用前进行定义，否则会在编译时出错。如例1.5中main( )函数的变量定义语句“int x,y,min;”。当然也可以没有说明部分。例如下面的程序代码：

```
main()
{  printf("It is fine today!" );}
```

该程序的作用是在屏幕上显示以下文字：

```
It is fine today!
```

这里没有用到任何变量，所以不需要变量的定义。

② 执行部分

执行部分由若干条执行语句组成。程序的功能就是通过执行部分实现的。C语言的任何语句都必须以分号“;”结束，分号是C语句的必要组成部分。例如：

```
y=x+b;
```

C语言的语句可以从任意位置开始书写，书写格式自由，一行内可以写多条语句，语句中多个空格与一个空格等效。

一个函数甚至可以既无说明部分，也无执行部分，例如：

```
comp()
{}
```

这个函数是一个空函数，什么作用也没有，但却是合法的。

（3）main( )函数

一个C语言的程序总是从main( )函数开始执行，并终止于main( )函数，而不论main( )函数在整个程序中处于什么样的位置（main( )函数可以放在程序的开头、最后，或某两个函数之间）。但是，为便于理解，通常将main( )函数放在程序的开头或最后。