HZ BOOKS
华章教育

高等院校计算机教材系列

THE C PROGRAMMING LANGUAGE WITH PRACTICE

C语言程序设计与实验指导

苏莉蔚 主编

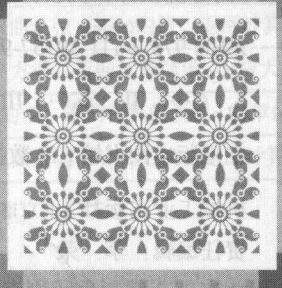
机械工业出版社
China Machine Press

高等院校计算机教材系列

THE C PROGRAMMING LANGUAGE WITH PRACTICE

C语言程序设计与实验指导

苏莉蔚 主 编
刘 威 副主编
白洪涛 姜 楠 王旭东 朱晓旭 孙 元 参 编



机械工业出版社
China Machine Press

本书主要介绍C语言的基础知识、语法特点并由浅入深地讲解C语言程序的开发过程，同时紧扣国家非计算机专业计算机等级考试大纲，以Visual C++ 6.0为程序运行环境，通过大量实例帮助初学者掌握C程序中的数据、表达式、控制语句、数组、指针、函数和文件等知识，以便尽快对C语言有系统和全面的认识。

本书概念清晰，例题针对性强，习题丰富，配有多种类型的实验题目，并对其进行详细分析和解释，帮助学生更好地理解和掌握C语言。

本书适用于大学本科理工类各专业学生学习C程序设计语言，同时也适用于自学C语言的读者。

图书在版编目（CIP）数据

C语言程序设计与实验指导／苏莉蔚主编. —北京：机械工业出版社，2012.8
（高等院校计算机教材系列）

ISBN 978-7-111-39157-9

Ⅰ. C… Ⅱ. 苏… Ⅲ. C语言-程序设计-高等学校-教材 Ⅳ. TP312

中国版本图书馆CIP数据核字（2012）第160021号

机械工业出版社（北京市西城区百万庄大街22号　邮政编码　100037）

责任编辑：李　荣

北京市荣盛彩色印刷有限公司印刷

2012年9月第1版第1次印刷

185mm×260mm·20印张

标准书号：ISBN 978-7-111-39157-9

定价：36.00元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

客服热线：（010）88378991；88361066

购书热线：（010）68326294；88379649；68995259

投稿热线：（010）88379604

读者信箱：hzjsj@hzbook.com

前言

随着科学的发展，社会的进步，计算机已深入日常生活的各个角落，信息的快速发展对人才的素质提出了更高的要求，非计算机专业的大学生已不仅仅局限于会操作和使用计算机。掌握一定的计算机技能，使用计算机进行程序设计，解决专业领域中的问题已经成为对学生的一项基本要求。

C 语言是广泛使用的程序设计语言，在各个领域中都有广泛的应用。它表达能力强，灵活方便，可移植性好，一直在软件开发过程中发挥着重要作用，并经常作为初学者学习程序设计的入门语言。

对于 C 语言，初学者都会感到困惑，不能很好地掌握语法，因而不能很快地自行编写 C 语言程序。考虑到 C 语言在当前计算机应用中的重要作用，既要保证课程内容的全面性，又要保证内容的针对性，还要强调内容的实践性，让学生多学多练，既不感到枯燥乏味，又能掌握理论知识和基本技能，使教学内容既符合教学实际，又满足社会的需求。

经过多年的教学与实践，我们组织编写了本书，在内容组织上本着充分调动学生的学习兴趣、提高学生分析问题和解决问题能力的精神，由浅入深，循序渐进。本书同时配有针对性强、简洁易懂的例题及类型丰富、特色鲜明的习题。此外，本书还提供了多种类型的实验题，并进行了详细分析和解释。同时，本书还提供了电子课件和习题答案（有需要者可登录华章网站 www.hzbook.com 下载），为教师授课和学生上机实习提供有效帮助。由于本书的内容非常贴近国家计算机等级考试中对 C 语言的要求，因此还能帮助学生轻松应对各种 C 语言考试。

本书的第 1 章、第 2 章和第 8 章由刘威编写，第 3 章由孙元编写，第 4 章和第 7 章由姜楠编写，第 5 章、第 6 章和第 9 章由苏莉蔚编写，第 10 章、第 11 章和第 12 章由白洪涛编写。电子课件的第 1 章～第 5 章及第 8 章由王旭东制件，第 6 章、第 7 章及第 9 章～第 12 章由朱晓旭制作。

本书实验指导中的实验一和实验五由王旭东编写，实验二和实验八由刘威编写，实验三和实验七由姜楠编写，实验四和实验十一由朱晓旭编写，实验六和实验九由苏莉蔚编写，实验十和实验十二由白洪涛编写，全书由苏莉蔚统稿。

由于作者水平有限，疏漏和不足之处在所难免，期望读者批评指正。

教学建议

教学章节	教学要求	课时
第1章 程序语言和C语言概述	了解C语言的发展及特点 了解程序设计的基本方法 掌握算法及其表示方法 了解C语言程序的基本结构及语法单位 熟悉C语言程序的开发环境	4
第2章 数据类型、运算符及表达式	掌握常量与变量的含义 掌握C语言基本数据类型及类型转换 掌握C语言的各种运算符与表达式	4
第3章 顺序结构程序设计	了解顺序结构程序设计的基本思想 掌握字符输入/输出函数 掌握格式化输入/输出函数	4
第4章 选择结构程序设计	了解选择结构程序设计的基本思想 掌握关系运算与逻辑运算的规则 掌握if语句的几种形式 掌握条件运算符与条件表达式的使用方法 掌握switch语句的语法及特点	6
第5章 循环结构程序设计	掌握while语句的语法及特点 掌握do…while语句的语法及特点 掌握for语句的语法及特点 掌握break和continue语句的特点	6
第6章 数　组	掌握一维数组的定义及元素的引用 掌握二维数组的定义及元素的引用	6
第7章 函　数	掌握函数的定义及调用 掌握函数的参数传递形式 掌握局部变量和全局变量的含义 了解变量的存储类别 了解内部函数与外部函数的区别	10
第8章 预处理命令	掌握宏定义的含义及应用 掌握文件包含的使用 了解条件编译	4
第9章 指　针	掌握指针的含义 掌握指针与指针变量的关系 掌握数组的指针及指向数组的指针变量的应用 掌握函数的指针及指针变量作函数参数时的使用方法	10

（续）

教学章节	教学要求	课时
第 10 章 结构体与共用体	掌握声明结构体类型和定义结构体变量的方法及结构体数组的使用 掌握结构体指针及指向结构体指针变量的应用 了解链表的建立及操作方法 掌握共用体类型变量的使用 了解枚举类型变量的使用 了解用 typedef 定义新类型的方法	10
第 11 章 位　运　算	了解二进制数、位和字节的含义 了解位运算的含义和使用 了解位域的含义和使用	4
第 12 章 文　　件	掌握文件及文件类型指针的含义 掌握文件的打开与关闭方法 了解文件的顺序读写函数 了解文件的随机读写函数 了解文件检测函数	4
总　课　时	基础知识课时	72
	综合实训课时	30

说明：

1）建议课堂教学全部在多媒体机房内完成，实现“讲－练”结合。

2）建议教学分为基础知识部分和综合实训部分，其中基础知识部分建议教学学时为 72，综合实训部分建议学时为 30，不同学校可以根据各自的教学要求和计划学时数对教学内容进行取舍。

目　录

第 1 章　程序语言和C语言概述

本章从程序设计语言的发展入手，介绍 C 语言的基本特点，算法及其表示，C 语言程序的基本结构、语法单位以及 C 语言程序的集成开发环境 Microsoft Visual C++ 6.0。

1.1　程序与计算机语言

在任何基于冯·诺依曼体系结构的计算机上，程序从某种外部设备（通常是硬盘）载入内存，然后按指令序列顺序执行，直到执行了一条跳转或转移指令，或者出现一个中断。基于这种体系的计算机如果没有程序支持将无法工作。计算机中的程序都是由计算机语言编写而成的。本节将介绍程序的概念、计算机语言的发展以及 C 语言的发展和特点。

1.1.1　程序的概念

我们知道，计算机是一种具有内部存储能力的自动、高效的电子设备，它最本质的使命就是执行指令所规定的操作。如果我们需要计算机完成什么工作，只要将其步骤用诸条指令的形式描述出来，并把这些指令存放在计算机的内部存储器中，需要结果时就向计算机发出一个简单的命令，计算机就会自动逐条顺序执行操作，全部指令执行完就得到了预期的结果。这种可以被连续执行的一条条指令的集合称为计算机的程序。也就是说，程序是计算机指令的序列，编制程序的工作就是为计算机安排指令序列。

1.1.2　计算机语言的发展

人和人之间有多种交流和沟通方式，其中最常用的就是语言。每个国家、地区和民族都有各自的沟通语言。人们在使用和控制计算机的过程中，需要有一种或多种人和计算机都能识别的语言来编程。因此，计算机语言通常被称为“程序语言”，一个计算机程序总是用某种程序语言编写的。

计算机程序语言的发展大致经历了机器语言、汇编语言和高级语言三个阶段。

机器语言和汇编语言均是完全依赖于具体机器特性的，是面向机器的语言，我们称之为低级语言（low level language）。而高级语言主要是相对于汇编语言而言的，它是较接近于自然语言和数学公式的编程，基本脱离了机器的硬件系统，用人们更易理解的方式编程。

1．机器语言

因为计算机内部的所有工作都是基于二进制的，所以计算机只能识别由 0 和 1 组成的二进制指令代码。这种计算机能直接识别和执行的二进制代码称为机器指令（machine instruction），而机器指令的集合称为机器语言（machine language）。每台机器的指令格式和代码所代表的含义都是硬性规定的，故称之为面向机器的语言，也称为机器语言。它是第一代计算机语言。不同型号计算机的机器语言是不兼容的，按某种计算机机器指令编制的程序，不能在其他种类的计算机上执行。

用机器语言编程，编程人员要首先熟记所用计算机的全部指令代码和代码的含义。编程时，程序员必须自己处理每条指令和每个数据的存储分配及输入/输出，还必须记住编程过程中每步所使用的工作单元的状态，这是一件十分繁琐的工作，编程花费的时间往往是实际运行时间的几十

倍或几百倍。综上所述，虽然机器语言具有灵活、能直接执行和速度快等优点，但是机器语言的程序设计思维和表达方式与人们的习惯大相径庭，并且具有程序可靠性差、开发周期长、可移植性差、重用性差和可读性差等诸多缺点。因此，除了计算机生产厂家的专业人员外，绝大多数程序员已经不再学习和使用机器语言了。

2．汇编语言

为了克服机器语言的诸多缺点，人们使用助记符（英文字母和数字）来替代特定的二进制指令。例如，用“ADD”代表加法，“MOV”代表数据传递，等等。这样，人们阅读程序、调试程序及维护程序都方便了很多。这种程序设计语言就称为汇编语言（assemble language），即第二代计算机语言。然而计算机并不能直接识别和执行这种助记符号指令，需要一个专门的程序，负责将这些符号指令翻译成二进制数的机器指令，这种翻译程序称为汇编程序。

汇编语言也是一种面向机器的语言，通常是为特定的计算机或系列计算机而专门设计的。汇编语言保持了机器语言的优点，具有直接和简捷的特点；可以有效地访问、控制计算机的各种硬件设备，如磁盘、存储器、CPU 和 I/O 端口等；目标代码简短，占用内存少，执行速度快，是高效的程序设计语言。汇编语言离不开具体计算机的指令系统，因此，不同型号的计算机，有着不同结构的汇编语言。而且，对于同一问题所编制的汇编语言程序在不同种类的计算机间是互不相通的。针对计算机特定硬件而编制的汇编语言程序，能准确发挥计算机硬件的功能和特长，程序精练而质量高，所以至今仍是一种常用且强有力的软件开发工具。汇编语言程序经常与高级语言配合使用，应用十分广泛。

汇编语言仅优于直接手工编写的二进制机器指令码，因此不可避免地存在一些缺点。例如，汇编语言编写的代码非常难懂，不好维护；很容易产生 Bug，难以调试；只能针对特定的体系结构和处理器进行优化；开发效率很低，时间长且单调。

3．高级语言

由于汇编语言依赖于硬件体系，且助记符量大难记，于是人们又发明了更加易用的高级语言。这种语言的语法和结构更类似于普通英文，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。高级语言通常按其基本类型、代系、实现方式和应用范围等分类。

高级语言并不是特指某一种具体的语言，而是包括很多编程语言。自从 1954 年第一个完全脱离机器硬件的高级语言 FORTRAN 问世以来，共出现了几千种高级语言。其中，具有重要意义且影响很大的有 FORTRAN、ALGOL、COBOL、BASIC、LISP、SNOBOL、PL/1、Pascal、C、Prolog、Ada、C++、VC、VB、Delphi 和 Java 等。

高级语言与计算机的硬件结构及指令系统无关，它具有更强的表达能力，可方便地表示数据的运算和程序的控制结构，能更好地描述各种算法，而且容易学习和掌握。但高级语言编译生成的程序代码一般比用汇编语言设计的程序代码要长，执行的速度也慢。所以汇编语言适合编写一些对速度和代码长度要求高的程序和直接控制硬件的程序。

高级语言程序“看不见”机器的硬件结构，不能用于编写直接访问机器硬件资源的系统软件或设备控制软件。为此，一些高级语言提供了与汇编语言之间的调用接口。用汇编语言编写的程序，可作为高级语言的一个外部过程或函数，利用堆栈来传递参数或参数的地址。

高级语言的发展也经历了从早期语言到结构化程序设计语言，从面向过程到非过程化程序设计语言的过程。相应地，软件的开发也由最初的个体手工作坊式的封闭式生产，发展为产业化、流水线式的工业化生产。

高级语言的下一个发展目标是面向应用，也就是说，只需要告诉程序你要干什么，程序就能自动生成算法，自动进行处理，这就是非过程化的程序设计语言。

1.1.3 C语言的发展及特点

1. C语言的发展概况

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计算机上实现。

C语言是在一种称为B语言的基础上开发的语言，克服了B语言依赖于机器又无数据类型等局限性。在1970年，美国贝尔实验室的Ken Thompson以BCPL语言为基础，设计出了一种既简单又接近于硬件的B语言，并用它写成了第一个基于PDP-7计算机的UNIX操作系统。B语言是取了BCPL语言的第一个字母。而BCPL语言（Basic Combined Programming Language）是1967年英国剑桥大学的M. Richard基于CPL语言（Combined Programming Language）提出的一种改进性语言。而CPL语言又是英国剑桥大学于1963年根据ALGOL60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ALGOL60，它的演变过程如图1-1所示。

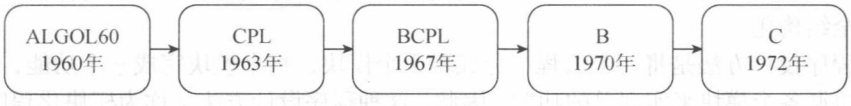


图1-1 C语言的演变过程

最初的C语言就是为了描述和实现UNIX操作系统而产生的一种工具语言，它集中了高级语言和低级语言的优点。1973年，贝尔实验室的Ken Thompson和D. M. Ritchie合作使用C语言修改了UNIX操作系统，即UNIX第5版本。原来的UNIX操作系统是用汇编语言写的，改写后UNIX操作系统90%以上使用了C语言。从此，C语言的命运与UNIX操作系统便有着密切的联系，随着UNIX操作系统的发展和推广，C语言也被广泛地使用和发展。

在1975年UNIX第6版本公布以后，C语言开始引起人们的注意，它的优点逐步被人们所认识。1977年出现了与具体机器无关的C语言编译文本，推动了UNIX操作系统在各种机器上的迅速实现。随着UNIX日益广泛地使用，C语言也得到了迅速的推广。1978年以后，C语言先后被移植到大、中、小和微型机上，很快成为世界上应用最广泛的计算机语言之一。

1978年又推出了UNIX第8版本，以该版本中的C编译程序为基础，B.W.Kernighan和D. M. Ritchie（被称为K&R）合作出版了《The C Programming Language》一书。但是，在此书中并没有定义一个完整的标准C语言。1983年，美国国家标准化协会（American National Standard Institute，ANSI）对C语言的各种版本进行了扩充，推出了新的标准，称为ANSI C。它对原来的标准C进行了改进和扩充。1987年，ANSI又公布了87 ANSI C新版本。

在ANSI C标准确立之后，C语言的规范在很长一段时间内都没有大的变动。1990年国际标准化组织，（International Standard Organization，ISO）接受87 ANSI C 为ISO C的标准ISO 9899—1990。1994年ISO修订了C语言标准。1995年 WG14小组对C语言进行了一些修改，成为1999年发布的ISO/IEC 9899:1999标准，通常称为C99。但是各个公司对C99的支持所表现出来的兴趣不同。当GCC和其他一些商业编译器支持C99的大部分特性的时候，微软和 Borland却似乎对此不感兴趣。

目前流行的各种C语言编译系统大多数以ANSI C为基础，但各有不同。当前微机上使用的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。本书的内容和实例均以ANSI C为基础，但读者在学习过程中也应了解不同版本编译系统的特点和区别（可参阅相关的学习资料）。

2. C 语言的特点

C 语言能发展得如此迅速，而且成为最受欢迎的程序设计语言之一，主要因为它具有以下众多特点。

（1）兼顾高级语言和低级语言的特性

C 语言把高级语言的基本结构和语句与低级语言的实用性结合起来。C 语言可以如汇编语言一样对位、字节和地址进行操作，而这三者是计算机最基本的工作单元。由于 C 语言允许直接访问物理地址，可以直接对硬件进行操作，因此它既具有高级语言的功能，又具有低级语言的许多功能，此特点有助于用 C 语言来开发系统软件。

（2）结构化程序设计模式

结构化程序设计语言的显著特点是代码及数据的分隔化，即程序的各个部分除了必要的信息交流外彼此独立。这种结构化方式可使程序层次清晰，便于使用、维护以及调试。C 语言是以函数形式提供给用户的，这些函数可以方便地调用，并具有多种循环和条件语句控制程序流向，从而使程序完全结构化。

结构化程序设计方法是将一个大程序分成若干个模块，每个模块完成一个功能，由一个总控模块控制和协调各个模块来实现总的功能。因此，这种程序设计方法又称为模块化程序设计方法。

（3）简洁紧凑，使用方便

C 语言是一种非常简练的语言，用 C 语言编写的程序十分简洁。其简洁性表现如下：

. 1）C 语言中类型说明符采用缩写形式。例如，整型用 int 而不用 integer，字符型用 char 而不用 character。

2）C 语言中关键字较少，只有 32 个。

3）运算符丰富，共有 34 种。C 语言把括号、赋值、逗号等都作为运算符处理，从而使 C 的运算类型极为丰富，可以实现其他高级语言难以实现的运算。

4）预处理功能将简化程序书写内容。

（4）功能强大

C 语言的功能强大表现在它同时具有数值运算和非数值运算的功能，并且在处理非数值数据时更加方便和灵活。C 语言还提供了丰富的数据类型，除了基本数据类型之外，还提供了构造数据类型，例如，数组、结构、联合和枚举等。使用这些数据类型可以很方便地实现各种复杂的数据结构（例如，链表、栈、树等）的操作。

C 语言还引入了指针的概念，使程序效率更高。另外，C 语言也具有强大的图形功能，支持多种显示器和驱动器，而且计算功能和逻辑判断功能也比较强大，可以实现决策目的。

（5）可移植性好

在诸多高级语言中，C 语言的移植性尤为突出。这是因为其编译系统较小，另外，预处理功能对移植也提供了方便。因此，C 语言本身只需稍加修改便可用于各种型号的机器和各类操作系统中，这也是 C 语言得以广泛应用的原因之一。

（6）目标代码质量高，程序执行效率高

一般情况下，C 语言程序只比汇编程序生成的目标代码效率低 10%~20%。

（7）语法限制不太严格，程序设计自由度大

C 语言的语法比较灵活，允许程序编写者有较大的自由度。

3. C 语言的不足

在学习 C 语言的过程中，除了要了解和掌握 C 语言的上述特点外，还应该知道 C 语言所存在的不足，从而避免出现一些莫名其妙的错误。

（1）运算符多，难用难记

C语言中有34种运算符，又有15种优先级和两类结合性，这无疑给数据的运算和处理提供了方便。但是，诸多运算符和不同的优先级势必难以记忆和使用。例如，有些功能不同的运算符却使用相同的符号，如“*”，它作为单目运算符表示取内容，作为双目运算符表示两个操作数相乘，与斜线符“/”连在一起（/*或*/）表示注释符。另外，它用在说明语句中，用来表示它右边的变量为指针变量，等等。此外，如果对运算符的15种优先级记忆混淆，会造成表达式计算结果的错误。

（2）类型转换灵活，语法检查不严格，易出错

C语言中类型转换比较灵活，如int型与char型可以自动转换，这些自动转换的规定为计算提供了一定的方便。例如，在C语言中允许一个字符与一个整数进行加减运算，'a'+2是合法的，该表达式的值用字符型表示为'c'，用整型数表示为99。为了转换上的方便，在许多情况下将不作类型检查。例如，在函数调用中，要求形参和实参类型一致，如果类型不一致也不判错仍可通过，有时会造成运算结果错误，为了避免这类错误发生，可以对函数进行原型说明，这时编译时将会对形参和实参的类型进行检查，增强其安全性。另外，也可使用强制类型转换运算符来限制不必要的自动转换。

（3）数组下标越界不检查，易造成数据混乱

在C语言中数组在动态赋值时不作越界检查。因此，在数组元素个数少于实际赋值的项数时，编译系统不会报越界错误，而是继续按地址进行赋值，这样容易造成数据混乱。为了降低出错率，应尽量避免动态赋值越界。对数组的静态赋值（即赋初值）不会发生越界赋值的现象，出现越界赋值将报错。

（4）求值顺序与二义性

C语言为了优化等原因，允许不同的编译系统在表达式或参数表内重新安排求值顺序。这样对于一般的表达式值和参数表内各项参数值是没有影响的，例如，改变表达式中各操作数的计算顺序不会改变表达式的值。但是，对一些具有副作用的运算符（如增1减1运算符和赋值运算符）来讲，不同的求值顺序将会造成表达式的值不同。这一点在编程时应特别注意，避免出现那些可能有二义性的表达式和参数表。

4. C语言版本

目前最流行的C语言有以下几种：

1）Microsoft C或称 MS C。

2）Borland Turbo C或称 Turbo C。

3）AT&T C。

这些C语言版本不仅实现了ANSI C标准，而且在此基础上各自作了一些扩充，使之更加方便、完美。

1.2 程序设计方法

程序设计的基本目标是用算法对问题的原始数据进行处理，从而获得所期望的效果。但这仅仅是程序设计的基本要求。全面提高程序的质量和编程效率，使程序具有良好的可读性、可靠性、可维护性以及良好的结构，编制出好的程序，应当是每位程序设计工作者追求的目标。而要做到这一点，就必须掌握正确的程序设计方法和技术。

1.2.1 结构化程序设计方法

结构化程序设计方法是公认的面向过程编程应遵循的基本方法和原则。结构化程序设计方法主要包括：

1）只采用三种基本的程序控制结构来编制程序，从而使程序具有良好的结构。

2）程序设计自顶向下。

3）用结构化程序设计流程图表示算法。

有关结构化程序设计及方法有一整套不断发展和完善的理论和技术，对于初学者来说，完全掌握是比较困难的。但在学习的起步阶段就了解结构化程序设计的方法，学习好的程序设计思想，对今后的实际编程很有帮助。

1．结构化程序设计特征

结构化程序设计的特征主要有以下几点：

1）以三种基本结构（顺序、分支、循环）的组合来描述程序。

2）整个程序采用模块化结构。

3）有限制地使用 goto 转移语句，在非用不可的情况下，也要十分谨慎，并且只限于在一个结构内部跳转，不允许从一个结构跳到另一个结构，这样可缩小程序的静态结构与动态执行过程之间的差异，使人们能正确理解程序的功能。

4）以控制结构为单位，每个结构只有一个入口和一个出口，各单位之间接口简单，逻辑清晰。

5）采用结构化程序设计语言书写程序，并采用一定的书写格式使程序结构清晰，易于阅读。

6）注意程序设计风格。

2．自顶向下的设计方法

结构化程序设计的总体思想是采用模块化结构，自顶向下，逐步求精。首先把一个复杂的大问题分解为若干相对独立的小问题。如果小问题仍较复杂，则可以把这些小问题又继续分解成若干子问题，这样不断地分解，使得小问题或子问题简单到能够直接用程序的三种基本结构表达为止。然后，对应每一个小问题或子问题编写出一个功能上相对独立的程序块来，这种像积木一样的程序块称为模块。每个模块各个击破，最后再统一组装，这样，对一个复杂问题的解决就变成了对若干个简单问题的求解。这就是自顶向下、逐步求精的程序设计方法。

模块是程序对象的集合，模块化就是把程序划分成若干个模块，每个模块完成一个确定的子功能，把这些模块集中起来组成一个整体，就可以完成对问题的求解。这种用模块组装起来的程序称为模块化结构程序。在模块化结构程序设计中，采用自顶向下、逐步求精的设计方法便于对问题的分解和模块的划分，所以，它是结构化程序设计的基本原则。

1960 年结构化程序设计思想诞生。C 和 Pascal 等语言都大力提倡这种程序设计的方法。结构化程序设计语言使得编写较复杂的程序变得容易。但是，一旦某个项目达到一定规模，即使使用结构化程序设计的方法，局势仍将变得不可控制。

在面向过程的程序设计方法中，问题被看做是一系列将被完成的任务，如读、计算和打印，许多函数用于完成这些任务，问题的焦点集中于函数。这种程序设计的基本任务是编写计算机执行的指令序列，并把这些指令以函数的方式组织起来。

3．程序设计的风格

程序设计风格从一定意义上讲就是一种个人编程时的习惯。而风格问题不像方法问题那样涉及一套比较完善的理论和规则，程序设计风格是一种编程的经验和教训的提炼，不同程度和不同应用角度的程序设计人员对此问题也各有所见。正因为如此，程序设计风格很容易被人们忽视，尤其是初学者。结构化程序设计强调对程序设计风格的要求，因为程序设计风格主要影响程序的可读性。

一个具有良好风格的程序应当注意以下几点：

1）语句形式化。程序语言是形式化语言，需要准确，无二义性。所以，形式呆板、内容活泼

是软件行业的风范。

2）程序一致性。保持程序中的各部分风格一致，文档格式一致。

3）结构规范化。程序结构、数据结构，甚至软件的体系结构要符合结构化程序设计原则。

4）适当使用注释。注释是帮助程序员理解程序、提高程序可读性的重要手段，对某段程序或某行程序可适当加上注释。

5）标识符贴近实际。程序中数据、变量和函数等的命名原则是：选择有实际意义的标识符，以易于识别和理解；要避免使用意义不明确的缩写和标识符，例如，表示电压和电流的变量名尽量使用 v 和 i，而不要用 a 和 b；要避免使用类似 aa、bb 等无直观意义的变量名。

1.2.2 面向对象程序设计方法

在程序设计方法的发展过程中，每一次重大突破都使得程序员可以应对更大的复杂性。在这条道路上迈出的每一步中，新的方法都运用和发展了以前的方法中最好的理念。今天，许多项目的规模又进一步发展。为了解决这个问题，面向对象程序设计方法应运而生。

发明面向对象程序设计（Object Oriented Programming，OOP）方法的主要出发点是弥补面向过程程序设计方法中的一些缺点。OOP 把数据看作程序开发中的基本元素，并且不允许它们在系统中自由流动。它将数据和操作这些数据的函数紧密地联系在一起，并保护数据不会被外界的函数意外地改变。

面向对象程序设计在程序设计模式中是一个新的概念，对于不同的人可能意味着不同的内容。其定义是“面向对象程序设计是一种方法，这种方法为数据和函数提供共同的独立内存空间，这些数据和函数可以作为模板以便在需要时创建类似模块的拷贝。这样的程序设计方法称为面向对象程序设计”。

从以上定义可以看到，一个对象被认为是计算机内存中的一个独立区间，在这个区间中保存着数据和能够访问数据的一组操作。因为内存区间是相互独立的，所以对象可以不经修改就应用于多个不同的程序中。

1.3 算法及其表示

计算机按照预先编制好的程序来完成操作任务。一个程序应包括以下两方面的内容：

1）对数据的描述，在程序中要指定数据的类型和数据的组织形式，即数据结构。

2）对操作的描述，即操作步骤，也就是算法。

数据是操作的对象，操作的目的是对数据进行加工处理，以得到期望的结果。作为程序设计人员，必须认真考虑和设计数据结构及操作步骤。著名的计算机科学家 Nikiklaus Wirth 提出了一个公式：

算法 + 数据结构＝程序

求解一个给定的可计算或可解的问题，不同的人可以编写出不同的程序。这里存在两个问题：一是与计算方法密切相关的算法问题，二是程序设计的技术问题。算法和程序之间存在密切的关系。在 Nikiklaus Wirth 提出的公式的基础上，目前人们又总结出一个新的公式：

算法 + 数据结构 + 程序设计方法 + 语言工具和环境 ＝ 程序

上述公式中的 4 个方面是一个程序员所必须具备的知识。

1.3.1 算法的概念

人们在生产活动和日常生活中离不开算法，都在自觉或不自觉地使用算法。例如，人们要购买物品，首先要确定购买哪些物品，准备好所需的钱，然后确定到哪些商场选购、怎样去商场、

行走的路线，若物品的质量好如何处理，对物品不满意又怎样处理，购买物品后做什么等。由此可见，人们从事各种工作和活动都必须按照各种算法（即方法、计划或步骤）来依次有序地进行。

人们使用计算机，就是要利用计算机处理各种不同的问题，而要做到这一点，人们就必须事先对各类问题进行分析，确定解决问题的具体方法和步骤，再编制好一组让计算机执行的指令即程序，交给计算机，让计算机按人们指定的步骤有效地工作。这些具体的方法和步骤其实就是解决一个问题的算法。根据算法依据某种规则编写计算机执行的命令序列，就是编制程序。由此可见，程序设计的关键之一，是解题的方法与步骤，是算法。学习高级语言的重点，就是掌握分析问题、解决问题的方法，就是锻炼分析、分解，最终归纳整理出算法的能力。

因为具体的语言程序（如 C 语言程序）是算法的一个具体实现，所以在高级语言的学习中，一方面应熟练掌握该语言的语法，因为它是算法实现的基础，另一方面必须认识到算法的重要性，加强思维训练，以写出高质量的程序。

综上所述，给出算法的定义：算法（algorithm）是一组有穷的规则，规定了解决某一特定类型问题的一系列运算，是对解题方案的准确与完整的描述。算法是解题的步骤。

计算机算法可分为数值计算算法和非数值计算算法两大类。用于解决科学计算中的数值积分、解线性方程等的计算方法，就是数值计算算法；用于解决管理、文字处理、图形图像等的排序、分类和查找的算法，就是非数值计算算法。

1.3.2 算法的特性

作为一个算法，一般应具有以下几个基本特性。

1）确定性：算法的每一种运算必须有确定的意义，该种运算执行某种动作应无二义性，目的明确。这一性质反映了算法与数学公式的明显差别。在解决实际问题时，可能会出现这样的情况：针对某种特殊问题，数学公式是正确的，但按此数学公式设计的计算过程可能会使计算机系统无所适从，这是因为根据数学公式设计的计算过程只考虑了正常使用的情况，而当出现异常情况时，此计算过程就不能适应了。

2）可行性：要求算法中有待实现的运算都是基本的，每种运算至少在原理上能由人用纸和笔在有限的时间内完成。针对实际问题设计的算法，人们总是希望能够得到满意的结果。但一个算法又总是在某个特定的计算工具上执行的，因此，算法在执行过程中往往要受到计算工具的限制，使执行结果产生偏差。

3）输入：一个算法有 0 个或多个输入，在算法运算开始之前给出算法所需数据的初值，这些输入取自特定的对象集合。

4）输出：作为运算的结果，一个算法产生一个或多个输出，输出是同输入有某种特定关系的量。

5）有穷性：一个算法总是在执行了有穷步的运算后终止，即该算法是可达的。数学中的无穷级数，在实际计算时只能取有限项，即计算无穷级数值的过程只能是有穷的。因此，一个数的无穷级数表示只是一个计算公式，而根据精度要求确定的计算过程才是有穷的算法。算法的有穷性还应包括合理的执行时间的含义。这是因为，如果一个算法需要执行千万年，显然失去了实用价值。

1.3.3 算法的表示

原则上说，算法可以用任何形式的语言和符号来描述，通常有自然语言、程序语言、流程图、N-S 图、PAD 图和伪代码等。流程图、N-S 图和 PAD 图是表示算法的图形工具。其中，流程图是最早提出的用图形表示算法的工具，所以也称为传统流程图。它具有直观性强、便于阅读等特点，具有程序无法取代的作用。N-S 图和 PAD 图符合结构化程序设计要求，是软件工程中强调使用的图形工具。