C程序设计与应用

施荣华　刘卫国　编著

中　国　铁　道　出　版　社

1999年·北京

(京)新登字 063 号

内 容 简 介

本书以 C 语言为编程工具，详细介绍了程序设计的基本方法与技巧，着力培养读者的程序设计能力。为突出综合应用能力的训练，本书增加了 C 语言综合应用方面的内容。鉴于面向对象程序设计代表了新颖的程序设计方法和思维方式，本书介绍了 C++ 语言和面向对象程序设计方面的基本知识。

本书既可以作为高等学校计算机程序设计课程的教材，也可以作为各种计算机培训班的教材，还可以作为计算机应用方面的技术人员的参考书。

图书在版编目(CIP)数据

C 程序设计与应用/施荣华，刘卫国编著 .-北京：中国铁道出版社，1999
ISBN 7-113-03373-3

Ⅰ.C… Ⅱ.①施… ②刘… Ⅲ.C 语言-程序设计 Ⅳ.TP312

中国版本图书馆 CIP 数据核字(1999)第 22482 号

书　　名：C 程序设计与应用
著作责任者：施荣华　刘卫国
出版·发行：中国铁道出版社(100054，北京市宣武区右安门西街 8 号)
策划编辑：殷小燕
责任编辑：殷小燕
封面设计：李艳阳
印　　刷：北京彩桥印刷厂
开　　本：787×1092 1/16　印张：16.75　字数：421 千
版　　本：1999 年 8 月第 1 版　1999 年 8 月第 1 次印刷
印　　数：1—2000 册
书　　号：ISBN7-113-03373-3/TP·371
定　　价：28.00 元

前　言

大多数人已取得这样一个共识：程序设计是计算机应用人员的一项基本功，也是大学生的一项基本思维方式训练。通过学习程序设计，可以编写出适合自己需要的程序，让计算机完成自己指定的任务，具体了解计算机是怎样进行工作的，知道计算机能做什么和不能做什么，而不至于在使用计算机时知其然而不知其所以然。同时，学习程序设计可以提高分析问题、解决问题的能力，开发人们的聪明才智，促进事业的发展。只有掌握程序设计的知识，才能具有一定的应用开发能力。学习程序设计是成为计算机高级用户的必由之路。

教育部对高等学校非计算机专业计算机基础教学提出了三个层次的要求，即“计算机文化基础”、“计算机技术基础”和“计算机应用基础”。其中，在“计算机技术基础”这一层次上要求学生具有高级语言程序设计方面的知识。《C程序设计与应用》一书就是作者在多年来从事计算机程序设计教学工作的基础上，适应计算机基础教学实际需要而编写。

学习高级语言程序设计涉及两方面的问题，一是要掌握一门高级语言，这是编程的基础，二是要掌握程序设计的基本方法，这是学语言的目的，也是学习高级语言程序设计的重点和难点。以往对初学者选学哪一门高级语言有过争论，我们认为这并不是大的原则问题，因为语言只是一种编程工具，只要能解决问题就行。更重要的是要掌握程序设计的基本方法，培养较强的程序设计能力。本书以C语言作为编程工具，编写时突出以下几点：

第一，力图讲清程序设计的基本方法、基本思路，努力培养读者的编程能力。而不过分死抠语言细节，对于语言的语法细节引导读者通过上机去掌握。

第二，突出综合应用能力的培养。教学实践表明，在课程全部内容讲授完毕后，增加一个综合练习，对提高学生的应用能力是极为有益的。本书增加了C语言综合应用方面的内容。

第三，增加新的内容。鉴于面向对象程序设计代表了新颖的程序设计方法和思维方式，本书介绍了C++语言和面向对象程序设计方面的知识。

全书共分为十三章。第一、二章介绍程序设计的基本概念、过程以及C语言的初步知识，使读者对C程序设计有一个直观的认识；第三章介绍C的基本运算对象的表示方法，这是程序设计必须具备的知识；第四章介绍C的语句以及程序设计的基本方法；第五章介绍构造数据类型——数组的概念与应用；第六章介绍函数的定义与应用；第七章介绍C的另一种重要数据类型，即指针类型的概念、定义与应用；第八、九、十章介绍C语言另外两类构造数据类型——结构体和共用体的概念与应用；第十一章介绍文件的概念以及文件的基本操作；第十二章通过几个实例介绍C的综合应用；第十三章介绍C++和面向对象程序设计的初步知识。

本书由施荣华、刘卫国主编。第一、二、八、九、十、十一、十二、十三各章及附录由施荣华编写，第四、五、六、七各章由刘卫国编写，第三章由毛锦编写。参加本书编写及文字录入整理工作的还有曹刚、盘善海、孙景伟、王国才、程思杰、甘元驹、彭春华、周跃飞等。

由于作者水平有限，书中难免出现错误和不妥之处，请读者不吝指正。

作　者

1999年4月

目　录

第一章　程序设计概述

第一节　程序设计的概念

什么是程序设计呢？最通俗的不严格的说法是：用计算机语言编写程序。那么什么是程序呢？著名的计算机专家沃思给出了一个有名的公式：

程序＝算法 ＋ 数据结构

也就是说，要编写程序必须要研究如何组织数据，如何进行操作。数据是程序操作的对象，因此首先应确定要对其施加操作的是什么样的数据，然后设计出对其操作的具体步骤，操作步骤就是算法。数据结构和算法是程序的两大要素。打个通俗的比方，一本菜谱介绍各种烹调方法，对每一种菜肴来说，需要说明用什么原料，然后介绍操作步骤，二者缺一不可。数据结构相当于原料，算法就是操作步骤。对不同的数据结构，算法是不同的。例如，对 20 个数排序，用数组处理和不用数组（只用简单的变量）来处理，算法差别是很大的。选择合适的数据结构可以降低算法的复杂程度。

上述公式当时是针对较小规模的程序而提出的，对于规模较大的程序而言，还要考虑如何组织程序设计。例如把一个大的任务先分解为若干较小的子任务，然后分别对每一个小的任务（即模块）进行程序设计，这就降低了程序设计的复杂程度。这就要研究程序设计方法。近年来推广的结构化程序设计方法，目的是使程序设计工作规范化，改变过去一个时期中程序设计无章可循，把程序当作个人的“艺术品”，从而降低程序可读性的状况。结构化程序设计方法的要点包括：自顶向下；逐步细化；模块化。每一模块都由具有良好结构特征的基本结构组成。

如果要处理的是一个复杂庞大的任务，按照软件工程的要求，需进行系统分析、系统设计和程序设计，一个软件的生存周期要经历以下几个阶段：

（1）问题定义与需求分析。说明问题是什么？系统必须做什么？

（2）总体设计（或概要设计）。将一个大的任务划分为若干个子任务即划分模块。概括说明应该如何解决所面对的问题。

（3）详细设计。根据给定子任务（模块）的要求，设计算法和数据结构，即说明怎样具体地实现这个系统。

（4）编写程序和单元测试。生成正确的程序模块。

（5）综合测试和确认运行。生成符合要求的软件。

（6）系统维护。保证软件能持久的满足用户的要求。

以上六个阶段称为“软件生存周期”，其中的（1）称为“软件定义时期”；（2）～（5）称为“软件开发时期”；（6）称为“软件维护时期”。一般所说的“程序设计 ”大体包括以上（3）、（4）两个阶段。

因此进行程序设计应该掌握程序设计方法，也就是要用结构化程序设计方法编写程序。

此外，一个程序必须是用一种计算机语言来编写的，要有必要的环境支持。例如，要用到操作系统提供的一些命令和系统软件。因此，我们认为：

一个合格的程序设计人员应当具备算法、数据结构、程序设计方法和程序设计语言及其支持环境这四个方面的知识。

在这四个方面中，数据结构是一门专门的课程，包含丰富的内容，在本书中不能作专门的详细的论述。只能结合具体例题加以讨论。本书重点是讨论算法，根据算法编写程序，在本书中所有的程序都是按结构化程序设计方法编写的。本书介绍 C 语言的基本概念和使用方法，包括有关的环境支持。

程序设计的全过程应该包括：分析问题、划分模块、建立模型(物理模型和数学模型)、选择数据结构和算法、描述算法(例如画出流程图)、编程序、上机调试程序、分析运行结果、写出程序说明书等文档。不要把程序设计简单地认为只是“编程序”而忽略其他方面的内容。

第二节　程序设计语言及其支持环境

程序设计语言是人与计算机进行信息交流的工具。程序设计是要在一定的语言环境下进行的。

一、程序设计语言的发展历程

从计算机诞生到今天，程序设计语言也在伴着计算机技术的进步不断升级换代，大体上可以说现在已经发展到了第四代。

(一)机器语言

一种 CPU 的指令系统，亦称该 CPU 的机器语言，它是该 CPU 可以识别的一组由 0 和 1 序列构成的指令码。下面是某 CPU 指令系统中的一条加法指令：

10000000

用机器语言编程序，就是从所使用的 CPU 的指令系统中挑选合适的指令，组成一个指令序列，这种程序是机器可以直接理解、执行的。但是，这种程序序列太长、不直观，而且难记、难认、难理解，不易查错；只有少数专业人员才能掌握；程序的生产效率很低，质量难以保证。

(二)汇编语言

为减轻人们在编程中的劳动强度，50 年代中期人们开始用一些“助记符号”来代替 0，1 码编程。如前面的机器指令可以写为：

ADD A, B

这种用助记符号描述的指令系统，称为符号语言或汇编语言。

与机器语言相比用汇编语言编程的生产效率及质量都有所提高。但是汇编语言指令是机器不能直接识别、理解和执行的。用它编写的程序经检查无误后，要先翻译成机器语言程序才能被机器理解、执行。这个翻译转换过程称为“汇编”。汇编后得到的机器语言程序称为目标程序(object program)，以前的程序，称为源程序(source program)。由于汇编语言指令与机器语言指令基本上具有一一对应的关系，所以汇编语言源程序的汇编可以由汇编系统以查表的方式进行。

汇编语言与机器语言，都是依 CPU 不同而异的，它们都称为面向机器的语言。程序员用它们编程时，不仅要考虑解题思路，还要熟悉机器的内部结构，并且要“手工”地进行存储器分配。这种编程的劳动强度仍然很大，使计算机的普及推广存在很大的障碍。

(三)面向过程的高级语言

汇编语言和机器语言是面向机器的，随机器而异。1954 年出现的 FORTRAN 语言以及随后相继出现的高级语言，不再是面向具体的机器，而是面向解题的过程。即用人们易于理解的形式来表示解题的过程。使用高级语言编程序时人们不必熟悉计算机内部的具体构造和熟记机器指令，而把主要精力放在算法上面。因此，面向过程的高级语言又称为算法语言。

下面是一个计算圆面积的 C 语言程序片段：

```
main()                          /* 告诉编译器 C 程序由此开始执行  */
{                               /* 这一程序段开始  */
int r;                          /* 半径 r 为整数  */
float s;                        /* 面积 s 为实数  */
s=3.14159 * r * r;              /* 计算面积 s  */
printf("%f",s);                 /* 输出面积 s 的值  */
}                               /* 程序结束  */
```

我们看到这个程序是很好理解的，其中计算面积的语句与我们所习惯的数学式子没有什么根本的区别("/ * " 与" * /" 之间的内容称为注释，目的是让程序更容易被理解)。显然，使用高级语言编程可以较大的降低编程过程的劳动强度，提高编程的效率。

高级语言的出现是计算机技术发展道路上的一个里程碑。它把计算机从少数专业人员手中解放出来，使之开始成为大众化的普通工具。从 1954 年第一种高级语言问世后不久，不同风格、不同用途、不同规模、不同版本的高级语言便风起云涌。据统计，全世界已有成千种的高级语言，其中使用较多的有近百种。在这些语言中，C 语言以其高效、灵活、功能丰富、表达力强、移植性好而受青睐，被称为十年来在计算机程序设计实践中做出了重大贡献的一种语言。

(四)非过程化的高级语言

使用过程化的语言解题，人们首先要理解问题要求我们"做什么"，然后去构造"怎么做"的解题过程。非过程化的高级语言又称"更高级语言"。用它编程序不但无须考虑机器的构造，而且无须去构造"怎么做"的解题过程，用户只需告诉计算机"做什么"，而不必告诉它"怎么做"，机器便会替你完成解题过程。例如，你只要告诉机器"找出平均成绩 60 分以下的学生名单"，机器便会按你的意图去完成这项工作。

非过程化的高级语言的解题效率更高，使用更为简便，更适合于非计算机专业人员使用。但是，它的运行效率及灵活性都不如过程化的语言，而且，目前所使用的大多数程序都是用过程化编写的。因此，学习用过程化的高级语言进行程序设计仍是计算机应用人员的一项基本训练。

二、程序设计语言的支持环境

要运行一个程序，除了必须的计算机设备(即计算机硬件系统)以外，还必须有软件环境的支持。

一个计算机系统包括硬件系统和软件系统两大部分。所谓软件系统(或称程序系统)是各种程序的总称。广义地说，软件泛指程序、运行时所需的数据以及程序的有关文档资料。软件系统着重研究如何管理机器和使用机器的问题，也就是研究怎样通过软件的作用更好地发挥计算机的功能。为什么需要软件系统呢？一个不包含任何软件的计算机称为"裸机"。在裸机上只能运行机器语言程序。显然它的功能是很有限的，机器效能得不到有效的发挥。软件分为系统软件和应用软件。系统软件是由计算机厂商提供的，它的作用是实现系统的功能，以充

分发挥计算机的效能，支持用户的应用程序运行，例如操作系统、编译程序等属系统软件。应用软件则是为了实现应用系统的专用功能而开发的程序，适用于特定的应用目的。

自从有了符号语言和高级语言，就有了汇编程序和编译程序，以便将符号语言程序或高级语言程序翻译为机器指令。从这里可以看到，一个高级语言程序不能直接在计算机上运行，而必须有一个软件的运行环境。

运行一个高级语言程序一般需要的软件支持环境有：操作系统、编辑程序、翻译程序及连接安装程序等。

1. 操作系统(简称 OS, Operating System)

现在几乎每一种计算机都配备了操作系统，它是各种软件中最重要的一种，是各种软件的核心与基础。早期的计算机工作方式一般为“独占”方式，即一个用户独自占用一台计算机。由于人工操作(包括输入程序和数据)速度慢而计算机的运行速度极快，因此计算机的利用效率极低，为了提高计算机的利用效率，大中型计算机采用“成批处理”(或称“批处理”)方式。将一批程序(以及运行该程序时所需的数据)和操作命令一起输入到计算机系统。计算机系统按规定顺序对程序一个一个地进行处理，不必人工干预。在批处理的基础上又发展了“多道处理”方式，即一个计算机“同时”执行多个程序。实际上，一个 CPU 在一个瞬间只能处理一条指令。对“多道”的程序作业是采用“中断”的方式来处理的。

如果在第一个程序处理过程中计算机出现异常或等待(例如需要打印)，计算机就自动地“中断”对该程序的处理，转而处理第二个程序……这样可以提高计算机工作效率，减少计算机“空闲”时间。

还有一种提高计算机工作效率的方式称为“分时方式”。在分时系统中一个计算机和许多台直接提供给用户使用的输入输出设备(称为终端设备)相联。最简单的终端设备就是一台显示器，也可以有打印机等。许多用户分别坐在自己的终端设备面前同时使用同一个计算机资源。每一个用户如同“独占”计算机一样地使用计算机，而实际上分时系统按一定的规律将一个时间单位分成若干段，轮流分配给各用户。由于计算机的运算速度极快，因此用户感觉不到同时还有别的用户在使用计算机。

要对用户作业进行有效管理，使它们协调无误和高效率地工作，这是一件极为复杂的工作。它要根据计算机运行时出现的瞬息万变的情况，迅速地、准确地作出决策和发出各种命令，使计算机执行相应的工作。这个工作是由“操作系统”来完成的。简单地说，操作系统相当于一个“总调度”，它控制着所有在该计算机上运行的程序并管理这个计算机的所有资源，最大限度地发挥计算机系统各个部分的作用。操作系统的基本特征是“多任务并行和资源共享”。根据功能的不同，操作系统可分为批处理操作系统、分时操作系统和实时操作系统等。操作系统本身是一个庞大的程序，它由以下五个部分组成：

(1) CPU 管理模块。它的任务是控制 CPU 的工作，接收“中断”信号，为用户提供 CPU 处理，充分发挥 CPU 的效能。它是操作系统的核心。

(2) 存储管理模块。它的任务是合理地为用户分配存储空间，有效地管理内存。

(3) 设备管理模块。它的任务是有效地管理计算机的各种设备，使之有效地进行工作。

(4) 文件管理模块。它的任务是管理外存中的程序(包括系统提供的程序和用户程序)和数据，在需要时及时将它们调入内存使用。

(5) 作业管理模块。它的任务是合理调度用户作业。所谓作业是指用户请求计算机所做的一个完整的工作。操作系统中用来控制作业执行的一组控制程序称为作业管理程序，包括

组织批处理作业或分时作业。

操作系统是所有软件的核心。现代的计算机系统，如果没有操作系统的支持，一切程序都无法工作。程序运行的每一步都必须由操作系统发出命令，计算机才执行各种操作。

不同的操作系统的功能和工作方式是不同的。目前广泛流行的操作系统有 Windows、Windows NT、VMS、OS/2 、Unix、DOS 等。有汉字输入输出功能的为中文操作系统，如 UCDOS、中文之星等。新一代操作系统用图形界面代替字符界面，如 Windows 98 等。

2. 编辑程序

为了方便用户将源程序输入计算机系统，许多计算机系统为终端用户提供了称为“编辑程序”的软件。它提供了一些方便的编辑功能，例如可以先输入一个源程序，然后可对它进行各种修改，如插入、删除某些字符，甚至可以将指定的某些字符或字符块从一处移到另一处。这如同对一篇文章进行编辑一样。用户可以在终端前通过键盘使用编辑程序来输入和修改源程序，直至满意为止。修改完的源程序存放在磁盘上(由用户指定文件名)，以后需要时再调入计算机进行编译。应当说明，编辑程序只有编辑功能，没有查错和翻译的功能，经过编辑的源程序仍然是以 ASCII 码表示的字符的集合，而不是机器指令。

在 PC 机上可以使用的编辑程序有 EDLIN 行编辑程序，也可用“全屏幕编辑程序”(如 Word, WPS, Edit)在整个屏幕范围内在任何位置上进行各种修改，使用更方便。关于各种编辑程序的使用方法，请参阅计算机文化基础类教材。

3. 翻译程序

翻译程序的功能是将符号语言和高级语言源程序翻译成机器语言程序。有三种翻译程序：

(1) 汇编程序

将符号语言程序翻译成机器语言程序。它对符号语言源程序逐行扫描，将机器码代替助记符号。同时进行语法检查，如有错误，输出错误信息。

(2) 编译程序

将高级语言源程序转换成机器语言程序。它实际上包括翻译和查错两个功能。具体功能包括：词法分析、语法分析、语义分析、生成目标程序及优化目标程序等。如果发现有错，向用户报告出错信息，不生成目标程序，用户必须重新调用编辑程序对源程序进行修改，然后再次进行编译。如还有错，则还要再用编辑程序修改源程序，然后再编译，直到正确得到目标程序为止。

(3)解释程序

它也是将高级语言源程序翻译成机器指令，但它与编译程序不同，它是边翻译边执行的，即输入一句，翻译一句，不产生整个的目标程序。翻译工作进行到遇到错误为止，如果没有错误，则一直进行到全部执行完毕。例如源程序有 10 个语句，第 5 个语句存在一个严重的错误，则前面 4 个语句都能正常翻译和执行，进行到第 5 个语句时输出错误信息，停止工作，用户必须修改此语句，然后从头重新运行程序。一般高级语言(如 C, FORTRAN, COBOL, PASCAL 等)都用编译方式，而 BASIC 语言及 FoxPro 等数据库语言多用解释方式(目前已出现编译版本)。解释方式使用灵活方便，占内存少，但执行时间长、效率低，编译方式得到的目标程序执行速度快、但占内存多。

4. 装配连接程序

经过编译得到的目标程序是不能直接执行的，因为目标程序可能要调用内部函数、外部函

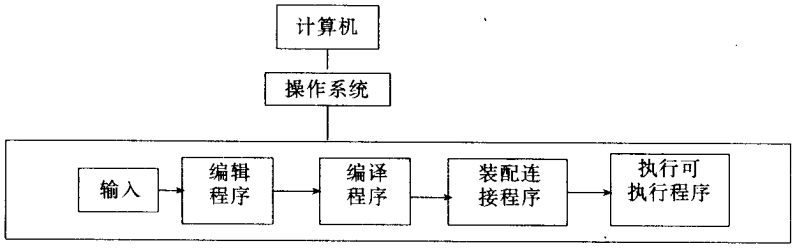

图 1- 1 程序运行的支持环境

数、子程序及系统提供的过程库中的程序，因此，在运行之前还需要用“装配连接程序”将所有的目标程序(如主程序、子程序)和系统提供的函数库、过程库等连接在一起成为一个整体。生成可执行的目标程序。

因此，在写好一个源程序以后，要经过编辑→编译→装配连接→运行，才能得到结果。程序运行的环境可用图 1－1 表示。计算机－操作系统－编译系统－源程序－用户之间的关系如图 1－2 所示。用户要使用计算机，先要编写好源程序，再经过编译、装配连接得到可执行的目标程序，在操作系统的统一调度与安排下由计算机执行。从表面上看，用户好像直接操纵计算机，其实，用户是通过操作系统与计算机打交道的。

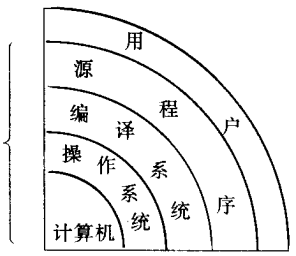

图 1- 2 用户与系统的关系

第三节 算法知识简介

一、算法的概念

做任何事情都有一定的步骤。例如，你要考入大学，首先要填报名表，交报名费，体检，拿到准考证，按时参加考试，得到录取通知书，到指定学校报到注册等。这些都是按一系列的顺序进行的步骤，缺一不可，次序错了也不行。因此，我们从事各种工作和活动，都必须事先想好进行的步骤，以免产生错乱。事实上；在日常生活中。由于已养成习惯。所以人们并不意识到需要事先设计“行动步骤”。但事实上都是按照一定的规律进行的，只是人们不必每次都重复考虑它而已。

广义的说，为解决一个问题而采取的方法和步骤，称为“算法”(Algorithm)。例如，描述南拳动作的图解。就是“南拳的算法”。一首歌曲的乐谱，也可以称为该歌曲的算法，因为它指定了演奏该歌曲的每一个步骤，按照它的规定就能演奏出预定的曲子。

对同一个问题，可以有不同的解题方法和步骤。例如，求 $1+2+3+\cdots+100$，即 $\sum n$，有的人可能先进行 1 十 2，再加 3，再加 4，一直加到 100。而有的人采取这样的方法 $\sum n=100+(1+99)+(2+98)+\cdots+(49+51)+50=100+50+49\times100=5050$。还可以有其他的方法，如：$100(101-1)/2=5050$。显然，方法有优劣之分。有的方法只需进行很少的步骤，而有些方法则需要较多的步骤。一般希望采用方法简单，运算步骤少的方法。因此，为了有效地进行解题，下仅需要保证算法正确，还要考虑算法的质量，选择合适的算法。

要完成一件工作，包括设计算法和实现算法两个部分，例如作曲家创作一首曲谱就是设计一个算法，但它仅仅是一个乐谱，并未变成音乐，而作曲家的目的是希望使人们听到悦耳动人

的音乐。由演奏家按照乐谱的规定进行演奏，就是“实现算法”。设计算法的目的是为了实现算法。因此，我们不仅要考虑如何设计一个算法，也要考虑如何实现一个算法。

本书所关心的当然只限于计算机算法，即计算机能执行的算法。例如，让计算机算$\sum n$，或将200个学生的成绩按高低分次序排列，是可以做到的。而让计算机去执行“替我理发”目前是不可能的。

计算机算法可分为两大类别，数值运算算法和非数值运算算法。数值运算的目的是求数值解。例如求方程的根、求一个函数的定积分等，都属于数值运算范围。非数值运算包括的面十分广泛，最常见的是用于事务管理领域，例如图书检索、物质管理、学籍管理、人事管理、行车调度管理等，计算机在非数值运算方面的应用远远超过了在数值运算方面的应用。由于数值运算有现成的模型，可以运用数值分析方法，因此对数值运算的算法的研究比较深入，算法比较成熟。对各种数值运算都有比较成熟的算法可供选用。常常把这些算法汇编成册（写成程序形式），或者将这些程序存放在磁盘或磁带上，供用户调用。例如有的计算机系统提供“数学程序库”使用起来十分方便。而非数值运算的种类繁多，要求各异，难以规范化，因此只对一些典型的非数值运算算法（例如排序算法）作比较深入的研究。其他的非数值运算问题，往往需要使用者参考已有的类似算法重新设计解决特定问题的专门算法。

二、算法的构成

算法含有两大要素：

一是操作，即构成一个算法的操作取自哪个操作集。它与使用的工具系统有关。如算盘上的操作集由进、退、上、下、去等组成；驾驶汽车的操作包括：踩离合器、踩油门、开电门、换挡、左转、右转、开灯、关灯等。计算机算法要由计算机实现，组成它的操作集是计算机所能进行的操作。而且这些操作的描述与程序设计语言的级别有关。在高级语言中所描述的操作主要包括：算术运算（+、-、*、/）、逻辑运算（“与”、“或”、“非”等），关系运算（==、>=、<=、>、<、!=）、函数运算、位运算、I/O操作等。计算机算法就由这些操作组成。

算法的另一要素是控制结构，即如何控制组成算法的各操作的执行顺序。结构化程序设计方法规定：一个程序只能由三种基本控制结构（或由它们派生出来的结构）组成。这三种基结构是：

①顺序结构。该结构中各个操作是按书写的顺序执行的。

②选择结构。根据指定的条件进行判断，根据判断的结果在两条分支路径中选取其中的一条执行。

③循环结构。或称重复结构。根据给定条件是否满足决定是否继续执行循环体中的操作。

由这三种基本结构还可以派生出“多分支结构”，即根据给定条件多个分支路径中选择执行其一的操作。

1966年Bohm和Jacopini证明，由这三种基本结构可以组成任何结构的算法，解决任何问题。

三、算法的描述

为了描述（表示）一个算法，可以用不同的方法。常用的有流程图、N-S图及PAD图等三种。

(一) 流程图

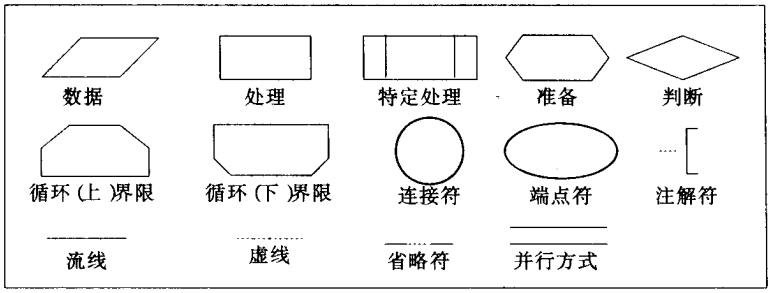

图 1- 3 流程图标准符号

流程图是一种流传很广描述算法的方法。这种方法的特点是用一些图框表示各种类型的操作,用线表示这些操作的执行顺序。我国国家标准 GB1526—89 中推荐的一套流程图标准化符号如图 1-3 所示。它与国际标准化组织 ISO (International Standard Organization) 提出的 ISO 流程图符号是一致的。

下面对其中一些主要符号作简要说明:

①数据。平行四边形表示数据,其中可注明数据名称、来源、用途或其他的文字说明。

②处理。矩形表示各种处理功能。例如,执行一个或一组特定的操作,从而使信息的值、信息形式或所在位置发生变化。矩形内可注明处理名称或其简要功能。

③特定处理。带有双竖边线的矩形,表示已命名的处理。该处理为在另外地方已得到详细说明的一个或一组操作,例如库函数或其他已定义的函数等。矩形内可注明特定处理名称或其简要功能。

④判断。菱形表示判断。菱形内可注明判断的条件。它只有一个入口,但可以有若干个可供选择的出口,在对定义的判断条件求值后,有一个且仅有一个出口被选择。求值结果可在表示出口路径的流线附近写出。

⑤流线。直线表示执行的流程。当流程自上向下或由左向右时,流程线可不带箭头,其他情况应加箭头表示流程。

⑥循环界限。循环界限表示循环的上界和下界。中间是要循环执行的处理内容,称为循环体。循环界限由去上角的矩形(表示的上界限)和去下角的矩形(表示的下界限)构成。

⑦端点。扁圆形表示转向外部环境或从外部环境转入的端点符。例如,程序流程的起始点。

⑧注解。注解是程序的编写者向阅读者提供的说明。注解符由纵边线构成,它用虚线连接到被注解的符号或符号组上。

图 1-4 中的虚线框,分别为用流程图表示的三种基本流程控制结构。这三种基本结构有一个明显的特征:单入口和单出口。从整体上看都相当于一个处理框。用它们所组成的程序来龙去脉十分清晰。

如图 1-5 所示,是用流程图表示一个求 n! 的函数模块。该函数返回 fac 的值就是 n! 的值。流程图中用到循环和判断,请读者自己消化理解此算法。

流程图灵活方便,它将算法用形象化的图形直观地展现出来。一个复杂的程序用流程图描述后,会变得一目了然、容易理解。它是目前应用较广的一种阅读、分析程序和算法设计的有力工具。

(二) N-S 图

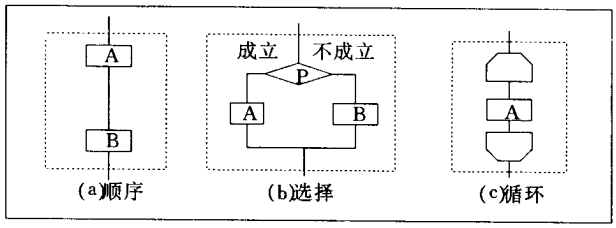

图 1 - 4 三种基本控制结构的流程图

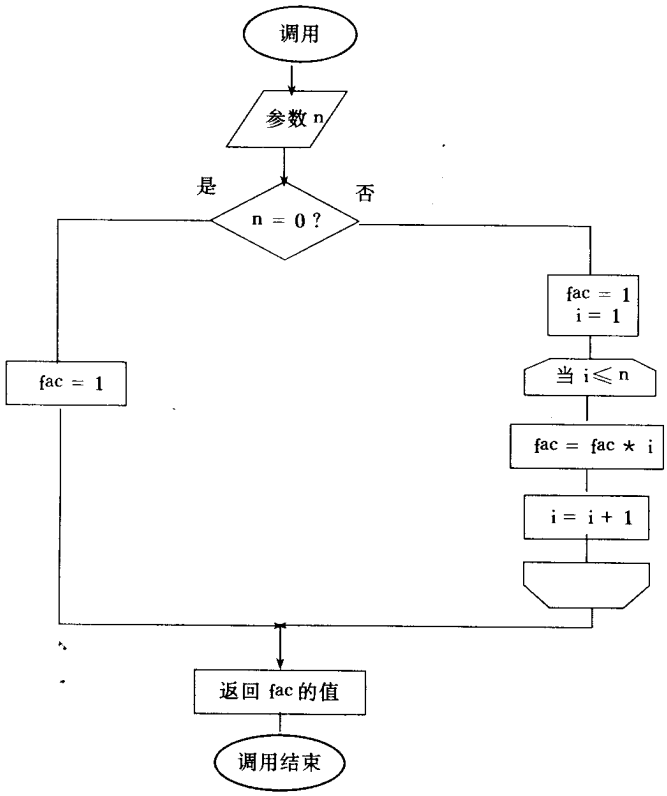

图 1 - 5 求 n ! 的函数模块流程图

灵活的流线是程序中隐藏错误的祸根。针对这一弊病，1973 年美国学者 I. Nassi 和 B. Shneiderman 提出了一种无流线的流程图，称为 N - S 图。用 N - S 图表示的三种基本结构如图 1 - 6 所示。

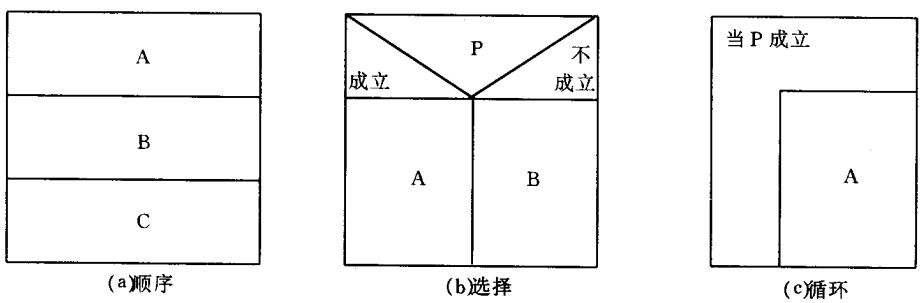

图 1 - 6 三种基本控制结构的 N- S 图

N－S图的每一种基本结构都是一个矩形框。整个算法可以像堆积木一样堆成。如图1－7所示为用N－S图描述的求n! 的算法。它总的是一个选择框，其中嵌套着一个顺序框，顺序框中又含有一个循环框，循环框中又嵌着顺序框。N－S图保留了流程图形象直观地用图形表示算法的优点，但去掉了容易导致程序非结构化的流线。人们用N－S图可以完全放心地设计出结构化的算法来。对初学者来说，使用N－S图可以强迫自己养成用三种基本结构构造算法的良好风格。但是N－S图的修改不大方便。

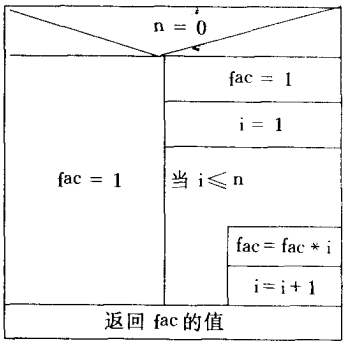

图1－7 求n!的N－S图

(三)PAD图

PAD(Problem Analysis Diagram——问题分析图)也是一种算法描述图形工具。图1－8所示为用PAD描述算法的三种基本结构。

PAD也没有流线，并且有规则地安排了二维关系：从上向下表示执行顺序，从左到右表示层次关系。图1－9所示为用PAD描述的求n! 算法。

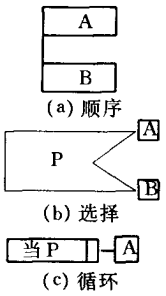

图1－8 三种基本控制结构的PAD图

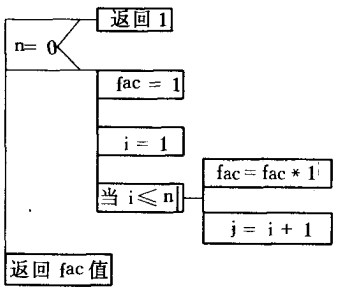

图1－9 求n!的PAD图

练习与思考

1. 什么是程序？请简述程序设计的全过程。
2. 什么是算法？试从日常生活中找出几个例子，描述它们的算法。
3. 试述三种基本结构的特点。
4. 试叙述什么是高级语言？什么是低级语言？
5. 试比较描述算法的三种基本方法。

第二章　C语言概述

第一节　C语言的发展历程

C语言是被广泛应用的计算机高级程序设计语言。C语言可用于编写各种复杂的应用程序，也能用它编写包括操作系统在内的系统程序。

因操作系统等系统程序依赖于计算机硬件，以前这类系统程序主要用汇编语言编写，因而程序的可读性和可移植性都很差，严重妨碍了系统程序的生产。为此，人们努力寻求一种程序语言，使它既具有高级语言的特性，能够编写可读性高、便于移植的程序，改善编写程序的环境；又具有某些必要的汇编语言特性，能描述对硬件的操作。例如能对内存地址的操作，对位的操作，对字的移位操作，以及对寄存器操作等。C语言就是在人们寻找集高级语言和汇编语言优点于一身的高级语言过程中产生的。

C语言是在B语言的基础上发展起来的。1963年，英国剑桥大学在ALGOL60程序设计语言基础上，推出了CPL(Combined Programming Language)。因其规模大，实现难等原因，1967年Matin Richards对CPL作了简化，推出了BCPL(B代表Basic)。以后，1970年美国Bell实验室的Ken Thompson又在BCPL的基础上，再次作进一步的简化，设计出既简单又接近硬件的B语言，并用B语言写了UNIX操作系统和大量的实用程序。因B语言只有单一的字类型，过于简单等原因而未能流行。D.M. Ritchie从1971年开始在B语言基础上设计了C语言，于1972年投入使用。1973年K. Thompson和D.M.Ritchie把UNIX系统用C语言重写了一遍，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。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、小、微型机上。C语言已风靡全世界，成为世界上应用最广的计算机程序设计语言之一。

1983年，美国国家标准化协会(ANSI)对C语言的各种版本作了扩充和完善，制定了C的标准，称为ANSI C。本书的叙述基本上以ANSI C为基础。目前广泛流行的各种版本C语言编译系统在非主要部分中稍有不同，在微型机上使用的主要有Microsoft C、TURBO C、Quick C等。因不同版本各有差异，因此读者在参考本书例题编写习题上机时应参考有关手册，了解你使用的计算机系统的C编译系统的特别规定。

第二节　C语言的特点

由于C语言集中了一般高级语言的优点和汇编语言的优点，能用它方便地编写不依赖于计算机硬件设施的各种应用程序，又能用它编写包括操作系统在内的各种系统程序。C语言具有多方面的特点，其主要特点有以下几个方面。

1.语言表达能力强

C语言包含丰富的运算符，有的运算符反映了当前计算机的性能，包含可直接由硬件实现的算术逻辑运算，有效到足以取代汇编语言编写各种系统程序和应用程序。众多的运算符

使C的运算类型极其丰富，可以表达数值运算、字运算、位运算和地址运算等。

2.数据类型丰富

C语言能在字符、整数、浮点数等基本类型的基础上按结构化的层次构造方法构造数组、结构和联合等各种结构化的数据类型。特别是C的指针类型灵活多样，非常有助于构造链表、树、栈、图等复杂的数据结构。另一方面，它的结构化程序控制结构符合结构化程序设计的要求，可编写结构非常良好的程序。此外，它的数据的静态和外部存储类机制有助于信息隐蔽、抽象的模块化结构程序设计。

3.语言简洁、紧凑、使用方便灵活

用C语言编写的程序通常比用其他高级语言编写的程序更简练，代码行少。语言的许多成分都通过显式函数调用完成。C语言没有I/O语句，也没有并行操作、同步或协同程序等复杂控制。另外，C语言程序在运行时所需要的支持少，占用的存储空间也小。

4.执行效率高

一个高级语言能否用来描述系统程序，除语言表达能力之外，还有能否产生高质量的代码这个重要因素。许多高级语言相对汇编语言而言其代码的执行效率要低得多，但C语言则不然，许多试验表明，用C语言描述较汇编语言描述，其代码执行效率只低约15%左右，而用C语言编程比汇编语言编程快得多，程序的可读性又高，特别是C语言程序比较容易移植。所以C语言成了人们描述系统程序和应用程序比较理想的工具。

5.所编程序的移植性较好

程序的可移植性是指在一个环境上运行的程序可以不加或稍加改动后在另一个完全不同的环境上运行。汇编语言是依赖于机器硬件的，用汇编语言编写的程序不可移植，而有些高级语言，因它们的编译程序不可移植，影响了用它们编写的程序的可移植性。目前在许多机器上都有C编译系统，且大部分是由C语言编译移植得到的。由于C语言的编译程序便于移植，也就提高了C程序的可移植性。

虽然C语言有许多优点，但也有一些不足之处。用C语言编程，自由度大，如对变量的类型约束不够严格，整型、字符型及逻辑型数据的通用，指针和数组的通用等。过多的通用性限制了编译程序对C程序作充分的句法和语义检查，可能会无视某些使用上的失误，依旧能正常编译，不能及时发现程序中的错误，给程序的调试和排错造成一些困难。另外，C语言的运算符优先级太多，不便于记忆，有些还与常规约定有所不同；类型检验弱，数据类型转换比较随便，进而影响了程序的安全性。

第三节　C语言程序结构

现用简单的C程序说明C程序的结构

例2-1

```
/* 一个只输出一行信息的C程序 */
main()/* 主函数 */
{
printf("This is a C programming . \n");
}
```

该程序只输出一行信息：