福建省高校计算机等级考试规划教材(二级)

C语言程序设计教程

福建省高校计算机教材编写委员会 组织编写

叶东毅 主编

厦门大学出版社
XIAMEN UNIVERSITY PRESS

福建省高校计算机等级考试规划教材(二级)

C语言程序设计教程

主　编　叶东毅

副主编　张　莹　谢丽聪

编写者(以姓名笔画为序)

叶东毅　江凤莲　杨　升　余文森

陈庆强　严宣辉　张　莹　陈维斌

姜德森　黄朝辉　谢丽聪

厦门大学出版社

图书在版编目(CIP)数据

C语言程序设计教程/叶东毅主编. —厦门:厦门大学出版社,2009.5
福建省高校计算机等级考试规划教材
ISBN 978-7-5615-3251-5

Ⅰ.C… Ⅱ.叶… 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(2009)第070143号

厦门大学出版社出版发行
(地址:厦门市软件园二期望海路39号 邮编:361008)
http://www.xmupress.com
xmup @ public.xm.fj.cn
沙县方圆印刷有限公司印刷
2009年5月第1版 2009年5月第1次印刷
开本:787×1092 1/16 印张:20
字数:512千字 印数:1~6 000册
定价:28.00元
本书如有印装质量问题请直接寄承印厂调换

前 言

本书是福建省高等学校学生计算机应用水平等级考试委员会根据其第七届第二次全体会议的精神，组织来自省内多所高校的骨干教师重新编写的，主要面向福建省高等学校非计算机类专业的学生。

C语言是一种使用广泛、功能强大的高级程序设计语言，学习C语言有助于学生更好地掌握程序设计的基本方法并逐步形成正确的程序设计思想。然而，相对其他一些程序设计语言来说，C语言比较不容易掌握，学习的难度比较大，学生的学习成绩总体上不够理想，近几年的福建省高校学生计算机应用水平二级C语言考试通过率都偏低。因此，如何把握C语言教学过程中的一些难点和重点，如何在保持C语言内涵和特色的前提下有效地进行教学内容的改革，如何提高学生的学习兴趣和学习成绩，成为大家普遍关心的问题。通过这几年的教学实践，福建省内多所高校相关课程的教师对C语言教学过程中存在的问题和如何提高教学质量有了比较清楚的认识，对C语言教材的内容框架、重点难点以及知识点的选择基本达成共识，希望能够重新编写一本教材，既着重于程序设计语言共性的内容和程序设计的基本思想方法，又能较好地反映C语言的特色，同时还可满足《福建省高等学校学生计算机应用水平等级考试——C语言程序设计考试大纲》的要求，兼顾等级考试的通过率。本书的编写正是基于这个想法。

本书主要包括C语言的基本数据类型、运算、基本控制结构、数组、指针、函数、用户自定义数据类型以及文件等内容，其中有关指针和函数的内容作了一定程度的简化。在参考多本现有C语言程序设计教材的基础上，书中也融入了编者在多年教学实践中的认识和体会，强化了对算法和程序设计基本思想的分析，并附有大量的实例，对学生在学习过程中容易混淆的概念和经常会犯的错误进行解剖和说明。

本书建议授课学时为40学时，上机学时为40学时，总计80学时。

本书共分9章。福州大学叶东毅教授任主编，并负责第1章的编写。第2章由龙岩学院江凤莲老师编写，第3章由福建师范大学严宣辉副教授编写，第4章由福州大学张莹副教授和谢丽聪副教授编写，第5章由莆田学院黄朝辉副教授编写，第6章由福建工程学院陈庆强副教授编写，第7章由泉州师范学院姜德森教

授编写，第8章由华侨大学陈维斌教授编写，第9章由武夷学院杨升副教授和余文森老师编写，全书由叶东毅、张莹和谢丽聪老师负责统稿。

本书的编写得到了福建省教育厅高教处和厦门大学出版社的大力支持和帮助；福建农林大学吴锤红教授提出了中肯而宝贵的意见，在此表示衷心的感谢！

本书虽经认真讨论、多次反复修改订正而定稿，但限于参编人数多，编者的学识和水平有限，疏漏、错误和不当之处在所难免，离重新编写教材期望达到的目标尚有差距，衷心希望任课教师、学生和其他读者不吝指教，以便对本书不断进行改进和完善。

编　者

2009年3月

目 录

第 1 章

程序设计概述

计算机由硬件系统和软件系统两部分组成。只有通过运行软件，计算机才能发挥作用，解决实际应用问题。一个软件是由计算机程序及其相关文档构成的，因此，程序设计是实现计算机应用的核心和基础。

本章简要介绍程序设计的基本概念和相关知识。

1.1　程序和程序设计语言

1.1.1　程序与程序设计的概念

作为一种能自动计算的机器，计算机通过执行一系列指令来完成给定的计算工作。因此，要让计算机完成某项任务，就必须将完成这项任务的方法和具体步骤编写成计算机可以直接或间接执行的一系列指令，使之执行这些指令后，就可以完成给定的任务。这样的一系列指令的集合就称为计算机程序或简称程序，编写这些指令就是程序设计。因此，程序可以看成是借助计算机的一种问题求解的过程。计算机通过运行不同的程序来完成各种不同的问题求解任务，从而实现计算机在各个领域的应用。

1.1.2　程序设计语言

程序设计语言是一组用来定义计算机程序的语法规则，用来向计算机发出指令。人们借助程序设计语言来编写程序，解决不同的问题。程序设计语言按照语言级别可以分为低级语言和高级语言。低级语言有机器语言和汇编语言，主要由机器基本指令集构成。它依赖于所使用的计算机硬件，即与特定的机器有关。它具有运行效率高的特点，但编写复杂、费时，容易出差错，而且程序修改维护困难。高级语言的表示方法比较接近于自然语言，在一定程度上与具体的计算机硬件无关，相对来说易于学习和使用，而且也便于维护，但是运行效率不如低级语言。下面对程序设计语言做进一步的解释。

1. 第一代程序设计语言(1GL)：机器语言

计算机所使用的是由“0”和“1”组成的二进制数，能直接识别的指令也是二进制编

码。因此，最早也是最底层的程序设计语言是以二进制编码形式的机器基本指令集为基础的语言，称为机器语言，也称为第一代程序设计语言(1GL)。在用机器语言编写的程序中，每一条指令都是二进制形式的指令代码，规定了计算机要完成的某个操作。在指令代码中，一般包括操作码和地址码，其中操作码告诉计算机做何种操作，即“干什么”，地址码则指出被操作的对象存放在哪儿。由于用机器语言编写的程序直接针对计算机硬件，因此，它的执行效率比较高，能充分发挥出计算机的速度性能，这是机器语言的最大优点。然而，机器语言也存在明显的缺点。因为机器语言程序由二进制代码组成，编写起来非常繁琐，难学、难记、难编，而且编出的程序不便于阅读和理解，容易出错，修改起来很困难。此外，机器语言强烈依赖于计算机硬件的指令系统，而每台计算机的指令系统往往各不相同，在一台计算机上执行的程序，要想在另一台计算机上执行，往往要重新编写程序，可移植性差。针对这些问题，人们设计了若干更有利于程序编写的程序设计语言，这些程序设计语言通常用于编写计算机间接执行的指令，然后借助专门的语言转换处理程序自动地将这些指令转换为机器语言，即计算机直接可以执行的二进制指令序列。根据发展阶段和功能特点的不同，这些程序设计语言分为第二代程序设计语言(如汇编语言)、第三代程序设计语言(也称为高级语言，如本书介绍的C语言)和第四代程序设计语言(也称为面向问题的语言)等。

2. 第二代程序设计语言(2GL)：汇编语言

为了克服机器语言难读、难编、难记和易出错的缺点，人们用与代码指令实际含义相近的英文缩写词、字母和数字等符号取代指令代码，例如，用ADD代表加法，用MOV代表数据传递等，这样，人们能较容易读懂并理解程序，使得纠错及维护变得方便了，这种程序设计语言称为汇编语言，即第二代程序设计语言。然而计算机是不认识这些符号的，这就需要一个专门的程序负责将这些符号翻译成二进制数的机器语言，这种翻译程序称为汇编程序。

汇编语言仍然是面向机器的语言，使用起来还是比较繁琐，通用性也差。但是，用汇编语言编写的程序，其目标程序占用内存空间少，运行速度快，有着高级语言不可替代的用途。

3. 第三代程序设计语言(3GL)：高级语言

为了克服前面提到的机器语言和汇编语言存在的缺点，人们开始寻求和设计一些尽量不依赖机器硬件、与人类自然语言相接近并且借助某种语言转换程序可被计算机接受和执行的程序设计语言。这种程序设计语言称为高级语言，也称为第三代程序设计语言。

用高级语言编写的程序一般称为源程序。计算机不能直接运行一个源程序，必须通过一种“翻译程序”将源程序翻译成机器语言形式的目标代码，计算机才能识别和执行。这种“翻译”通常有两种方式，即编译方式和解释方式。

编译方式是指在源程序执行之前，就将源程序代码“翻译”成目标代码(机器语言)，因此目标程序可以脱离其语言环境独立执行，使用比较方便，效率较高。但应用程序一旦需要修改，必须先修改源程序代码，再重新编译生成新的目标程序才能执行。

解释方式是源程序代码一边由相应语言的解释器“翻译”成目标代码(机器语言),一边执行,因此效率比较低,而且不能生成可独立执行的文件,应用程序不能脱离其解释器,但这种方式比较灵活,可以动态地调整、修改源程序。

高级语言的出现使得计算机程序设计语言不再过度地依赖某种特定的机器或环境。无论哪种机型的计算机,只要配上相应的高级语言的编译或解释程序,则用该高级语言编写的程序就可以在计算机上运行。

1954 年诞生了第一个不依赖机器硬件的高级语言 FORTRAN。此后,共有数百种高级语言问世,其中 Basic、Pascal、C、C++、Java 等是近年来广泛使用的程序设计语言。

早期的高级语言大多是面向过程的,面向过程的程序设计以函数或子程序为中心,用函数或子程序来作为划分程序的基本单位,程序的执行是对一系列函数或子程序的调用,数据往往处于从属的位置。20 世纪 80 年代初,在软件设计思想上发生了一次变革,由此产生了面向对象的程序设计。面向对象的程序设计以数据为中心,类作为表现数据的工具构成划分程序的基本单位,而函数成为了类的接口。面向对象程序设计可以更直接地描述客观世界存在的事物(即对象)及事物之间的相互关系。C++就是一个典型的面向对象程序设计语言。

4. 第四代程序设计语言(4GL):面向问题语言

第四代程序设计语言的提出主要是为了成数量级地提高软件生产率,缩短软件开发周期,它通常具有“面向问题”、“非过程化程度高”等特点。确定一个语言是否是一个第四代程序设计语言,主要考察以下几个方面:

(1)生产率标准:4GL 应比 3GL 提高生产率一个数量级以上;

(2)非过程化标准:4GL 基本上应该是面向问题的,即只需告知计算机“做什么”,而不必告知计算机“怎么做”;

(3)用户界面标准:4GL 应具有良好的用户界面,简单、易学,使用方便、灵活;

(4)功能标准:4GL 要具有生命力,不能适用范围太窄,在某一范围内应具有通用性。

以数据库管理系统为基础的查询语言 SQL 就是一种典型的第四代计算机语言。

1.2　算法概述

1.2.1　算法的概念

一般来说,为解决一个问题而采取的方法和步骤,就称为算法。计算机算法则是用计算机求解一个具体问题或执行特定任务的一组有序的操作步骤(或指令),是构成计算机程序的核心部分。著名瑞士计算机科学家 N. Wirth 曾经提出一个公式:

程序＝数据结构＋算法

其中,数据结构主要是数据的类型和数据的组织形式,是对程序中数据的描述。算法则是

对程序中操作的描述，也就是操作步骤。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。由此可见算法是计算机程序设计中的灵魂。

需要注意的是，算法一般只是对处理问题思想的一种描述，不是计算机可以直接执行的程序代码。因此算法本身是独立于计算机的，算法的具体实现则由计算机完成。从这个意义上说，程序设计的本质就是要将算法转化为计算机程序。这个转化过程往往因人而异，即，对于同一个算法，不同的人可以编出不同的程序，其运行效率也不相同。读者在后续章节的学习过程中会逐步体会和认识到这点。

处理一个问题，可以有不同的算法。以求1到100之间所有偶数的和这个十分简单的问题为例，我们可以有多种不同的算法。例如，一种是逐个考察1到100之间的数，如果是偶数则累加，如果是奇数就不累加，最后的累加数即为所求。显然这样做效率是比较低的。另一种是从2开始累加，每次加进的数是上一次的数加2，累计50次可得。还有一种是先求出1到50的和再乘以2而得。当然，还可以有其他的算法。由此可见，设计和选择算法是至关重要的。不仅要保证算法正确，还要考虑算法的质量和效率。

1.2.2 算法的表示方法

描述一个算法可以有不同的方式，常见的有以下三种：

(1)使用自然语言描述算法；

(2)使用流程图描述算法；

(3)使用伪代码描述算法。

下面仍以求1到100之间所有偶数的和(记为sum)为例说明算法的3种描述方法。假设采用前面提到的最后一种算法，即先求出1到50的和再乘以2。

第1种：使用自然语言描述求sum的算法。

①假设初始值i为1；

②假设变量sum初始值为0；

③如果i≤50时，执行④，否则转出执行⑦；

④计算sum加上i的值后，重新赋值给sum；

⑤计算i加1，然后将值重新赋值给i；

⑥转去执行③；

⑦计算sum乘以2的值，输出sum的值，算法结束。

使用自然语言描述算法的方法比较容易掌握，但是有些操作不易表述清楚，例如循环操作。另外，还可能造成歧义，使他人对相同的一句话产生不同的理解。

第2种：使用流程图描述求sum的算法。

传统的流程图由一些特定意义的图形、流程线及简要的文字说明构成，它能明确地表示算法的运行过程。表1-1给出流程图中所使用的图形的含义。

用流程图描述的算法如图1-1所示。从图1-1中，可以比较清晰地看出算法的执行过程。

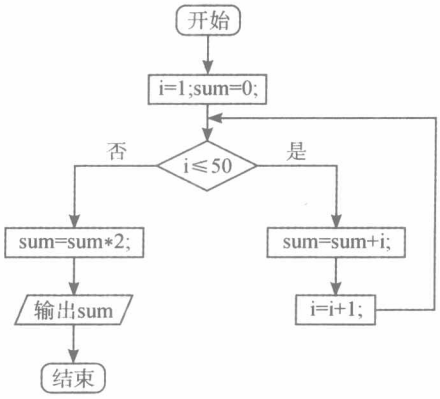
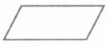


图 1-1　求 sum 的算法流程图

表 1-1　流程图的图形解释

图框	名称	含义
	起止框	表示算法开始或结束的符号
	流程线	表示算法流程的方向
	输入输出框	表示算法过程中的信息输入和输出
	判断框	表示算法过程中的选择分支结构。通常用上面的顶点表示入口，根据需要用其余的顶点表示出口
	处理框	表示算法过程中需要处理的内容。只有一个入口和一个出口

传统流程图的一个主要不足是流程线的用法缺乏规范。由于流程线可以转移流程的执行方向，如果使用不当或流程控制转移不明晰，容易导致程序的混乱和出错。为此，人们(I. Nassi 和 B. Schneiderman，1973 年)设计了一种新的流程图，它没有使用流程线，而是把整个算法写在一个大框图内，这个大框图由若干个小的基本框图构成，算法按照从上到下、从左到右的顺序执行。这种流程图简称 N-S 流程图。由于没有流程线，不可能任意转移控制，N-S 流程图具有结构化的特点，因此得到更普遍的使用。本书的后续章节中有许多例子都采用 N-S 流程图的方式描述算法。图 1-2 给出的是求 sum 算法的 N-S 流程图。

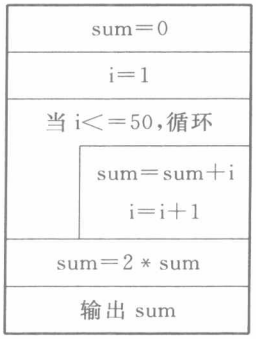


图 1-2　求 sum 算法的 N-S 流程图

采用流程图和N-S图的方式表示算法，直观形象，易于理解，是常用的描述算法的方法。然而，无论是使用自然语言还是使用流程图或N-S流程图描述算法，都只是表达了编程人员处理问题的一种思路，离真正的程序代码还有很大的差距。因此，人们引入了一种接近于计算机程序设计语言的算法描述方法——伪代码。

第3种：使用伪代码描述sum的算法。

伪代码是一种用来书写程序或描述算法时使用的非正式表述方法，主要采用自然语言、数学公式和符号来描述算法的操作步骤，同时采用计算机高级语言（如C、Pascal、VB、C++、Java等）的基本控制结构（参见1.3节）来描述算法步骤的执行顺序。尽管伪代码不是一种编程语言，写的算法也不如流程图直观，但伪代码接近程序设计语言，因而用伪代码写出的算法比较容易转化为程序代码。下面是用伪代码描述sum的算法，并附有注释行。

```
BEGIN                    /* 算法开始 */
i←1;                     /* 为变量 i 赋初值 */
sum←0;                   /* 为变量 sum 赋初值 */
while i<=50              /* 当变量 i<=50 时，执行下面的循环体语句 */
{ sum←sum+i;
  i←i+1;}
sum←2 * sum;
输出 sum 的值
END                      /* 算法结束 */
```

1.3 结构化程序设计方法

高级语言的发展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语言的过程，也引发了多次程序设计方法的变革和进步，其中结构化程序设计方法就是一个很好的例子。20世纪60年代中后期，由于高级语言的出现，应用高级语言编写的软件越来越多，规模越来越大，然而程序的编写缺乏规范，编写风格和方法因人而异，可以说是想怎么编写就怎么编写。同时，缺乏科学规范的软件系统规划、测试和评估标准，导致许多软件系统含有错误而无法使用，软件的可靠性差。这种情形在计算机发展历史上被称为“软件危机”。经历了这次软件危机之后，人们逐渐意识到大型程序的编写不同于小规模程序的编写，它需要一种新的设计理念和准则，需要像设计大型工程那样来编写大型的程序。于是，结构化程序设计方法应运而生，随之在1970年诞生了第一个结构化程序设计语言——Pascal语言。

1.3.1 结构化程序基本控制结构

结构化程序设计方法由著名的计算机科学家E. W. Dijkstra和N. Wirth等人在20世纪60年代后期提出并逐渐发展起来，其基本思想是采用“自顶向下、逐步求精、分而治之”的原

则，将一个较为复杂的原问题分解成若干相对独立的小问题，依次细化，直至各个小问题获得解决为止。换句话说，就是以模块化设计为中心，将待编写的软件划分为若干个相互独立的子模块，每一个模块的功能单纯、明确，模块之间相互独立，在设计其中一个模块时，不会受到其他模块的影响。

按照结构化程序设计的观点，任何算法功能都可以通过三种基本控制结构以及它们的嵌套组合来实现，这三种结构就是顺序结构、选择(分支)结构和循环结构。

1. 顺序结构

顺序结构是依次执行指令的结构。即，程序中的指令按照顺序依次执行，每条指令都必须执行，且只执行一次，如图 1-3 所示，图 1-4 是相应的 N-S 流程图。

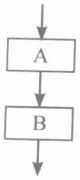


图 1-3　顺序结构

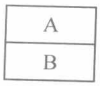


图 1-4　顺序结构 N-S 图

2. 选择(分支)结构

选择(分支)结构根据逻辑判断的结果，做不同的处理。一种典型的选择(分支)结构是双分支结构，如图 1-5 所示，图 1-6 是相应的 N-S 流程图。

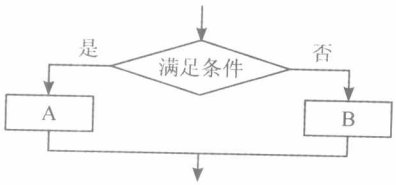


图 1-5　双分支结构

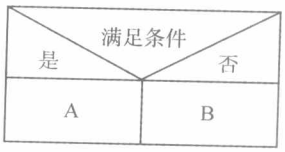


图 1-6　双分支结构 N-S 图

3. 循环结构

在循环结构中，当条件满足时，反复执行某条件语句或语句组的操作，直到条件不满足时为止。循环结构也称重复结构。根据条件设置方式和执行方式的不同，可以有两种不同的循环结构，分别为当型循环结构和直到型循环结构，如图 1-7 和图 1-8 所示，图 1-9 和图 1-10 分别是相应的 N-S 流程图。

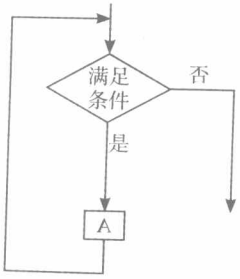


图 1-7　当型循环语句

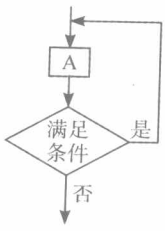


图 1-8　直到型循环结构