高等院校电类专业新概念教材·卓越工程师教育丛书

C程序设计高级教程

周立功　主编
周攀峰　陈明计　编著

北京航空航天大学出版社

高等院校电类专业新概念教材·卓越工程师教育丛书
编 委 会

前　言

一、创作起因

1. 软件危机

近年来，随着代码量越来越大，不可移植且不可复用的代码导致产品上市时间一再延误，在将软件交付使用之前依然无法找到所有的错误，代码的维护难度也越来越大，软件开发与维护的过程仍旧难以度量，开发成本居高不下，软件企业被竞争对手打败的现象屡见不鲜，软件行业呈现危机。作者从业30多年来，不仅见证了IT技术的高速发展，而且也经历了新产品的开发由单兵作战到团队开发的痛苦转变过程，事实上软件在创新中所起到的作用越来越大。

2. 人才危机

微软公司招聘人才遵循“三好学生”原则——态度好、数学好、编程好，其中重要的一条就是要求编程好。但现实的情况是，教学内容陈旧，缺乏工程价值，与工业界的要求严重脱节，从而导致大量学生因为编程能力差而找不到理想的工作。具有讽刺意味的是，每年有几百万的大学生毕业，而企业却招聘不到满意的人才。

3. 教学危机

虽然通过灌输知识可以让人具有很强的考试能力，但是教育的本质在于培养学生的创造力、好奇心、独特的思考能力与解决工程实际问题的能力，去启迪和唤醒人类的智慧。那么，什么是好的教学方法呢？就是去启迪具有不同性格的人、不同背景的人、不同文化的人、不同思考模式的人，去唤醒人类的智慧，发现学生们的强项，在传授知识的同时帮助学生树立正确的人生观与价值观并重建理想。走出大学校门的学生不应仅仅追求拥有一个填饱肚子的好“饭碗”，而应勇于直面社会的千变万化，并从中领悟人生真谛，尽而在改造社会的过程中享受奋斗的无穷乐趣。这可能是今天与未来创新教育所面临的最大挑战。

二、本书特点

如何让教学更有效率、学习更有效果、教育更有效益，这是高等教育面临的根本问题。到底哪些内容是培养合格工程师和卓越工程师所必需的，如何与工业界无缝连接，这是我们制订教学大纲与培养计划时必须认真思考的问题。

传统C语言教材中的示例重在解释语法，很少涉及软件工程的设计思想和方法。一般来说，通过这种传统方法培养出来的学生编写有效代码的能力较差。因此培养一名卓越的工程师需要讲授哪些关键的知识点，成为教材创作需要重点考虑的因素。

1. 将程序设计贯穿始终

作者认为，C语言教材的内容应该具有一定的工程价值，这样便于读者在后续的学习中，灵活地设计最优化的代码，而不是为了验证语法而编程。现有的教材与教学内容严重滞后于软件工程技术的发展，大多数教师也是这样“被教”出来的，因此他们教出来的学生很难满足现

代企业的需求。这就是教学危机带来人才危机与软件危机的根源。

2. 指针无处不在

其他的高级语言在输出参数和数组方面具有独立的语言结构，但C语言将这些知识渗透在它的指针概念中，因此相对来说学习难度会大得多。本书在相关章节分散介绍指针，从而简化了学习过程，使得学生能够由浅入深地逐渐吸收指针用法的精髓。这种方法便于用传统的高级语言术语（输出参数、数组、数组下标和字符串）来表达基本概念，对于没有汇编语言背景的学生来说，更容易掌握指针的多种用法。

虽然大多数C语言教材会专门用一章的篇幅来介绍指针，但往往出现在后半部分，这是远远不够的。本书不仅从第一章就开始介绍指针，而且在第5章“深入理解指针”中，结合一维数组、字符数组、结构体、结构体数组与枚举详细阐述了指针，同时在第7章“深入理解函数”、第8章“深入理解数组与指针”、第9章“深入理解结构与指针”以及第11章“创建可重用软件模块的技术”几章中，全面描述了指针在不同的上下文环境中的有效用法，展示了使用指针编程的惯用方法。

3. 算法基础

开发具有健壮性的软件需要高效的算法，然而很多程序员往往等到问题发生之时，才会求助于算法，这就有点晚了。事实上，“数据结构＋算法＝程序”是阐明计算机软件技术的经典表达式，而属于数据结构与算法的范畴非常广泛，如何在有限的时间内，选择合适的学习内容并打下坚实的基础则是至关重要的。因此，在教材中精选用例是非常重要的，这正是本书的特点之一。比如：

①“求二进制数中1的个数”示例，看起来这是一个非常简单的算术运算示例，但实际上却是“用于通信的奇偶校验”的常用算法。

②“去掉最大值与最小值求平均值”示例，看起来不难，但实际上却是数据采集信号处理中常用的“去极值平均数字滤波”算法。去掉一个最大值，即相当于去掉“毛刺型干扰”，去掉一个最小值，即相当于去掉“噪声型干扰”，最后通过求平均值得到有效采样值。

③“优先编码器的实现”示例，虽然没有几行程序，但却是嵌入式实时操作系统中常用的“判定任务就绪优先级最高”的算法。

④“冒泡排序”是最简单的算法，从性能上看，似乎没有存在的理由，但其实现简单，而简单的程序通常更可靠。在数据量不大时，所有排序算法性能差别不大。而高级排序算法只有在元素个数多于1 000时，性能才显著提升。在90%的情况下，存储元素的个数只有几十到上百个而已，比如，进程数、窗口个数和配置信息等的数量通常都不大，此时冒泡排序算法是最好的选择。

由于经典算法思想相对来说比较成熟且容易掌握，因此本书侧重于结合经典算法思想学习如何设计可重用的代码。而选择排序、插入排序、希尔排序、归并排序、快速排序、顺序查找与二分查找等算法，尽管其实现方法不一样，但都可以抽象为具有相同形参的接口，因此调用者在更换算法时，就无需修改应用程序的源代码。

⑤直观的“连加和”算法几乎成为所有C语言教材的经典示例，如何提高程序的运行效率和健壮性，是程序设计的重点。

4. 数据结构基础

由于信息技术的高速发展，程序设计与数据结构以及算法的边界越来越模糊，因此，结合数据结构的相关知识点来提升算法的性能也是本书的重要特征之一。

通过一题多解、由浅入深、循序渐进，详细地阐述了栈、队列与链表的来龙去脉，以及各种重用软件的实现方法，使学生日后能用于工程项目之中。

5. 软件工程方法

其实在软件开发过程中，用户改变需求并不过分。虽然可能仅仅改动几个字，但对于有些已经完成的代码，却几乎需要重头编写。原因就在于其编写的代码不容易维护，灵活性差，不容易扩展，更谈不上重用。

由于硬件新技术的不断涌现，产品功能越来越复杂，随之而来的是代码量也越来越大。虽然“算法＋数据结构”依然是软件技术的核心，但当代码越来越大时，开发过程的浪费也越来越大。许多情况下由于缺乏规划，以致于编写出来的代码缺陷成堆并难以发现，用流程来规范这种行为就成为当务之急，这就是软件工程的本质。由此可见，软件设计方法已经显得尤为重要。因此“算法＋数据结构＝程序”必将被“系统”所取代，即“程序＝算法＋数据结构＋方法”，且更重要的是，计算机学科必将扩展和融合更多的学科知识。这将是未来最大的变化。

(1) 重　构

软件并非灵光一闪就能创造出来，人们总是将兴奋点放在编写代码上。要想成为一名优秀的程序员，只知道什么是好代码是远远不够的，重要的是要知道如何将代码变得更漂亮。实际上，傻瓜编写的代码只有计算机才能理解，而优秀的程序员编写出来的代码可以让其他人也能看懂。

重构是改变软件系统的过程，它不会改变代码的外部行为，但是可以改善其内部结构。这样一来，就能将引入 bug 的风险降得更低。因此，有了重构就有可能将一个糟糕甚至混乱的设计，逐渐转变为良好的设计。

(2) 软件分层技术

分层设计就是将软件分成具有某种上下级关系的模块，由于每一层都是相对独立的，因此只要定义好层与层之间的接口，每一层就都可以单独实现。基于分层的架构具有以下优点：

- 有利于降低系统的复杂度与隔离变化。
- 有利于自动测试。由于每一层都具有独立的功能，因此更易于编写测试用例。
- 有利于提高程序的可移植性。通过分层设计将各种平台的不同部分放在独立的层中，那么当软件移植到不同的平台时，只需要实现不同的部分即可，中间层都可以重用。

(3) 接口与实现

创建可重用软件是现代程序设计的基本概念。当问题达到一定的复杂度时，则可以将其抽象出来，定义一个简单的接口，封装成为隐藏复杂性的库。对于用户来说，在使用这个库时不必仔细理解抽象的细节，从而简化了程序的结构。由此可见，汇编子程序与 C 语言函数不一定就是模块，只有分离接口与实现的代码才是可重用的软件模块。

(4) 分离容器与算法的迭代模式

传统的 C 语言程序设计将数据结构与算法放在了一起，从而导致对每一个数据结构都要开发一套与之匹配的算法。而迭代模式就是将容器（最通用的几种数据结构）与算法单独设

计，即算法不依赖于容器的实现，算法不会直接在容器中进行操作，从而有效地实现了代码重用。

本书是一本引领学生学习C语言程序设计的教材，而并非一本软件工程技术方面的专著，提供给学生的仅仅是一个起点，最终都要落实于实践，因为只有在实践与创新中才能不断地进步。从某种意义上来说，学习的过程就是一个人不断认识自己、了解自己和超越自己的过程，同时也是思考人生和不断自我完善的过程。我相信，对于每一个学生来说，无论你有什么样的目标和追求，本书都会对你的成长有所启迪。

三、教学内容的组织与安排

本书由三部分组成，即基础篇、提高篇和综合篇，其中基础篇与提高篇是必修的内容。虽然综合篇的内容有一定难度，但对于学生来说，应该记住：学从难处学，用从易处用。实际上，程序设计课程不是听会的，也不是看会的，而是练会的，是在充分上机动手编程的过程中逐步学会的，因此，只要将基础篇与提高篇中的示例全部经过上机调试，则学习综合篇也就很容易了。综合篇的特点是融合了前面所学的基础知识，重在提高学生的程序设计能力。建议综合篇的学习以上机练习为主、教师精讲为辅。

1. 基础篇

基础篇包括第1～4章，为学习“C语言程序设计”应知应会的基础知识，其所有的示例尽管很简单，但却很实用。如果读者仅仅是满足于看懂了，到头来合上书还是两眼一抹黑。因此，读者必须一一上机实践，只有这样才能达到熟能生巧的目的。

第1章——程序设计基础。本章重在引导初学者入门，在思想和方法上给初学者以启发。对内存的理解可以说是C语言程序员的基本素质之一，很多学生之所以学会了C语言的语法，却仍然无法写出正确的程序，主要原因在于其对内存的理解不够透彻，所以本章从变量的存储开始陆续介绍内存的基本知识。

第2章——简单函数。通过实参与形参在内存中的存储方式来阐述传值与传址函数调用中数据传递的本质，以及如何精确地返回到调用点的函数调用机制，并全面掌握用return语句与指针返回函数的结果(即值与地址)以及指针作为输入/输出参数的习惯用法。

第3、4章——选择结构程序设计与循环结构程序设计。主要描述每种语句的正式结构，并详细描述其语法和语义，以及在程序设计中的习惯用法。

2. 提高篇

提高篇包括第5～10章。

第5章——深入理解指针。虽然指针的威力无穷，且可以直接访问硬件，但只有在深入理解它的基础上，才能成为一名优秀的程序员。如果使用恰当，则可以大大简化算法和提高效率；如果使用不当，则很容易引起错误，且非常难以发现。

第6章——变量与函数。主要讨论全局变量与局部变量、内部函数与外部函数的作用域与可见性，以及变量的存储方式与生存期。

第7章——深入理解函数。重点介绍函数指针以及用回调函数实现隔离，其中用回调函数实现隔离是一个极其重要的知识点。有了回调函数，通过对软件进行分层设计，不仅可降低系统的复杂度，而且能隔离层与层之间的变化，有利于提高软件的可移植性。

栈与函数的嵌套调用与递归调用通过内存紧密相连，很多时候总让人有一种看不见摸不

着的感觉。同时，递归这个概念对初学者来说，常显得颇为神秘，但本章通过图解法并结合内存对函数的嵌套调用与递归调用进行了详细的讨论。

第8章——深入理解数组与指针。虽然多维数组与复杂指针在教学中始终是一个难点，但要用好多维数组与复杂指针也并不难，理解和区分基本概念非常重要。当初学者具备一定的编程经验之后，根据需要再来重温本章的内容也就不难了。

第9章——深入理解结构与指针。结构的应用非常广泛，如果运用恰当，则事半功倍。针对结构体与指针，本章重点介绍复杂结构体类型成员与动态内存分配，以及与内存泄漏相关的知识与排除方法，最后详细介绍一个保存任意数据类型的通用单向链表。

第10章——流与文件。程序利用变量存储各种信息，但这些信息只是瞬间存在的，一旦程序停止运行，变量的值就丢失了。而在很多应用中，能够永久保存存储信息是相当重要的。而只有以文件的形式保存在一种永久存储介质中才是可行的。实际上，文本文件、游戏、可执行文件、源代码和其他一些存储在计算机上的永久数据对象都是以文件的形式存储的。本章重在介绍如何进行文件处理，以及如何使用标准函数库中的各个输入/输出函数。

3. 综合篇

第11章为综合篇。

第11章——创建可重用软件模块的技术。本章以栈、队列与链表为例展开论述，提出分离接口与实现的思想，详细介绍数据隐藏与隔离变化等设计方法。本章还通过需求引出了迭代模式，介绍分离容器（最通用的几种数据结构）与算法，从而有效地实现代码重用。

四、习题与实验指导

习题与实验指导将另行发布，这样便于有针对性地不断抽象出更有代表性的教学示例，以得到举一反三的效果。

其实，知识来自于别人的经验，智慧和能力来源于自己的经验。只有自己亲身体会，才能具有一定的智慧和能力，因此仅仅通过听课和看书是无法学会程序设计的，必须多上机编程才能熟而生巧。

五、更多资源

由于本教材的篇幅有限，因此可以通过配套的习题集、实验指导、电子版辅导资料与PPT课件来弥补其中的不足。如果仅局限于教材本身的内容，或完全依靠老师在规定学时之内传授的知识，对于学生来说是远远不够的，应该根据自己的兴趣加强课外学习。与本书密切相关的参考资料，请到周立功单片机网站（www.zlgmcu.com）中的“卓越工程师视频公开课”专栏下载。

六、面向对象

本书最早是为电类专业（包括电子信息工程、电气自动化、自动化、电子科学与技术、测控技术、通信、医学电子、机电一体化等专业）编写的，随着软件技术与嵌入式技术的高速发展，本书内容也成为计算机等相关专业教学内容的基础。因此，本书不仅适用于电类专业，同样也适用于计算机科学与技术、计算机应用与软件工程等专业。

七、结束语

本书由周立功、陈明计、周攀峰，历时5年的构思与实践，联合创作而成，是“高等院校电类

专业新概念教材·卓越工程师教育丛书”之一，由周立功担任主编，负责全书内容的组织策划、构思设计、修改完善以及最终的审核定稿。

周航慈教授提供了很多素材并指导了部分内容的写作；江西理工大学王祖麟教授、广州致远电子有限公司的朱旻，以及李先静也参与了本书的创作；西安邮电大学陈莉君教授在作者写作初期提出了很多中肯的意见。他们都是我的良师益友。在此，向这些卓有建树的专家、学者们深表谢意。

在本书的写作过程中，作者也参考和引用了本书“参考文献”中所列的经典图书的内容，在此，也向参考文献的作者们致以诚恳的谢意。

本书是作者从业 30 多年的工作总结。读者若有意见和建议，欢迎给我写信（QQ：2355327888，新浪微博：ZLG 周立功），作者期盼着与你们的交流。

周立功

2012 年 10 月 8 日

目　录

第1章 程序设计基础

本章导读

对内存的理解可以说是C语言程序员的基本素质之一，很多学生之所以学会了C语言的语法，而仍然无法写出正确的程序，主要原因在于对内存不够理解，所以本章从变量的存储开始介绍内存的基本知识。本书将基本的C语言语法安排在第1章“程序设计基础”与第5章“深入理解指针”进行完整的介绍，打破了传统教材的写作方法与传授C语言的思路。这样便于读者在后续的学习中，灵活地设计最优化的代码，而不是为了验证语法而编程。

虽然本书介绍了很多有关C语言的细节问题，但并不需要读者完全掌握，主要是为了方便程序员遇到问题时查看，以加深理解。既要理解抽象的程序设计概念，又要深入了解一门语言，往往很难找到切入点，因此读者不宜将主要的精力投入到语言的学习上，否则在学习程序设计方法方面将会事倍功半。千万不要被C语言的细节所困扰，否则会导致“只见树木不见森林”的后果。

1.1 提前引用的概念

由于C语言中的很多概念互相交织，因此一些概念在还未说明前就可能引用了。为了便于理解，本节将对一些提前引用的概念进行简单的说明。读者看不懂不要紧，等到学习了后续内容再回过头来温习时，自然就明白了。

- 语句：语句是C语言的基本执行单元，详见1.12.1小节。
- 运算符：运算符是一些计算符号，详见1.7节。
- 表达式：表达式是C语言中有意义的式子，详见1.8节。
- 函数：函数是由一组语句组成的执行单元，详见第2章。
- 库函数：函数的一种存在形式，在语法角度与普通函数没有区别，详见2.6节。
- 标准库函数：标准库函数是C语言必须提供的库函数，详见2.6节。
- printf()：它是C语言提供的一个标准库函数，用于输出数据，详见1.12.3小节。
- scanf()：它是C语言提供的一个标准库函数，用于输入数据，详见1.12.4小节。

1.2 第一个C语言程序

1.2.1 Hello World

Hello World是一个经典的入门程序示例，同样也是深入研究计算机的极好题材，详见程

序清单1.1。

程序清单1.1 Hello World范例程序(hello.c)

```
#include<stdio.h>
int main(void)
{
    printf("Hello World!\n");                    //打印字符串 Hello World
    return 0;
}
```

尽管这个程序非常简单,但却反映了C语言的几个重要特点。

1. 注 释

从标准C99开始,注释可以从"//"字符开始,并延续到下一个行分隔符。例如,语句"//你的第一个C语言程序"用"//"开头,这个符号用作说明这一行程序从这个符号后的部分为"注释",双斜线注释常用于半行或单行的标注。

注释行主要是为了提高程序的可阅读性,通常用于概括算法、确认变量的用途或阐明难以理解的代码段。注释不会增加可执行程序的大小,编译器会忽略所有注释。

当然,所有的编译器都支持注释以"/*……*/"开头和结尾,注释可以包含任意数量的字符,并且总是被当作空白看待。

注意: 一个注释对不能出现在另一个注释对中。

2. stdio.h头文件

```
#include <stdio.h>
```

这是一条预处理程序指令,说明程序可能会使用一些标准输入/输出库函数,即一种标准库函数。也就是说,只要在程序的开始添加这一行,即可使用printf()函数输出字符串"Hello World!"。

在编译(详见1.2.2小节)程序之前,凡是以"#"开头的代码都先由"预处理器"予以处理。当预处理器看到这条指令时,它首先在编译器(详见1.2.2小节)指定的目录搜寻stdio.h文件。如果存在这样的文件,则由预处理器(详见1.2.2小节)将"标准输入/输出头文件(stdio.h)"中的内容包括到程序中,头文件stdio.h中包含了编译器在编译标准输入/输出库函数(如printf()、scanf())时要用到的信息和声明,还包含了帮助编译器确定调用库函数的程序格式是否正确的信息。如果不存在,则预处理器就会依次在当前目录和其他指定目录中寻找这个文件。如果仍然无法找到,则预处理器提示出错信息。

这些库函数是C编译器自带的,由其他程序员(创建者)编写的,用于实现特定的功能,其目的就是对用户(调用者)隐藏那些不必要知道的细节,防止用户程序将可能以想象不到的方式改变底层数据结构的值。

注意:

```
#include <stdio.h>
```

也可写成

```
#include "stdio.h"
```

两者的区别仅仅是搜索stdio.h文件的顺序有所不同,后者是先搜索当前目录,再搜索其