说C解C

耿　楠　聂艳明　李建良　冯　妍　著

西安电子科技大学出版社
http://www.xduph.com

C 语言程序设计丛书

说C解C

耿楠 聂艳明 李建良 冯妍 著

西安电子科技大学出版社

内 容 简 介

本书针对目前国内高校在C语言程序设计教学中对部分重要语法细节的关注不足、对综合应用的分析不够深入、缺少工程化意识和忽略程序DEBUG等问题，提出了程序设计“三观”的概念，即内存观、代码观和调试观，选取25个实用性强的专题如DEBUG的概念及使用、scanf()函数及键盘缓冲区、数据类型的本质、浮点数及其使用、函数参数的单向值传递、泛型排序程序设计和第三方库的安装与使用等，以科技论文的撰写方式对所列专题进行了较为深入细致的讨论。

本书图文并茂，实例丰富，语言活泼，以期为深入学习和掌握C语言程序设计的读者提供指导和帮助，为承担C语言教学的教师提供参考和启发。

图书在版编目 (CIP) 数据

说C解C/耿楠等著.—西安：西安电子科技大学出版社，2021.3
ISBN 978-7-5606-5578-9

I. ①说… II. ①耿… III. ①C语言—程序设计 IV. ①TP312.8

中国版本图书馆CIP数据核字 (2020) 第088817号

策划编辑 陈婷
责任编辑 苏镇镇 陈婷
出版发行 西安电子科技大学出版社 (西安市太白南路2号)
电 话 (029)88242885 88201467 邮 编 710071
网 址 www.xduph.com 电子邮箱 xdupfxb001@163.com
经 销 新华书店
印刷单位 陕西天意印务有限责任公司
版 次 2021年3月第1版 2021年3月第1次印刷
开 本 787毫米×1092毫米 1/16 印 张 19.75
字 数 459千字
印 数 1～3000册
定 价 46.00元
ISBN 978-7-5606-5578-9 / TP
XDUP 5880001-1

献给：那些喜欢C的人，把C当作乐趣的人。为了C，你可能茶饭不思；为了C，你可能面红耳赤；为了C，你可能义无反顾。C是一种精神，一种说不清楚的理想，痛苦并快乐。

序

受耿楠教授等人之盛邀，为他们所著的《说 C 解 C》一书作序，荣幸之余，心存惶惶。

C 语言作为程序设计语言中的一员，具有特殊的身份和地位。自从 D.M.Ritchie 和 Ken Thompson 开发出这种不依赖于具体机器系统的语言后，它就焕发出蓬勃的活力。近些年来，各种新的计算机程序设计语言层出不穷，特别是以面向对象为特征的、有利于开发者理解的、有利于高效撰写与复用的语言，往往一经推出就迅速普及开来，以至于在 C 语言教学与学习中，有许多人质疑其在最近和将来一段时期内的价值。我就此问题谈谈个人的看法：第一，目前唯一能够完全衔接低级语言（机器码和汇编语言）和其它各种高级语言的中级语言是 C 语言；第二，目前的主流操作系统无一不是以 C 语言为基础开发的；第三，在硬件控制中，最能高效驱动且易于移植，并使开发者易于理解的是 C 语言。如果不掌握这个不可替代的有力工具，在软件和硬件方面做本质上的创新，无异于沙滩上筑高楼。

要想学好 C 语言，第一，要了解计算机的基本架构和特点，就像要学好外语最好先了解那个国家的文化渊源；第二，要掌握 C 语言的基本语法规则和运行流程，不仅仅是死记硬背，要从逻辑上明白为何是这样而不是那样；第三，要经常从机器和人两个视角考查并理解一些容易出现歧义的规定；第四，要多练习，不能机械地将代码从书本或屏幕上挪移到计算机中，应该在头脑中深思熟虑（综合考虑操作流和数据流）后，再输入计算机中，而且要在输入的同时预测出错的地方和输出的结果；第五，要多总结，将人和机器的思维模式对应起来，要掌握扎实的编译技巧，同时要具备深厚的编译功底。

耿楠教授等人在多年的教学实践中对许多技术细节做了比较深入的调查研究，同时对学生的学习习惯有比较全面的掌握，基于“详尽说明、透彻讲解、深入浅出、旁征博引”的原则撰写了这本《说 C 解 C》，期望能对想学好 C 语言的诸位有所帮助。使科技易于学习和掌握不只是 C 语言教授者和学习者的心愿，更是所有相关技术研究者们的宏念。孜求进步，果待自心！是为序。

张志毅

2020 年 12 月

陕西杨凌

前　言

本书根据作者教学团队多年从事“C 语言程序设计”“数据结构”等课程的教学和科研工作经验，以当前广泛使用的轻量级免费跨平台开发工具 Code::Blocks(MinGW) 为 IDE 开发环境，针对 C 语言程序设计教学过程中常见的重点和难点进行了深度剖析。

本书并没有系统讲解 C 语言的语法和编程方法，而是针对选定的 25 个专题，将 C 语言中的重点、难点和疑点的细节从计算机编程理念、工程化规范意识、开发思路与技巧等方面，进行了深入的解读。通过讲解大量实践案例和技巧，揭示 C 语言中那些鲜为普通学习者或开发者所知的秘密，并简要介绍了 C99 语言规范的新特性及不同平台架构下数据表示的差异，旨在让已经具备 C 语言基础的读者真正掌握 C 语言，从而撰写出更加高效和稳定的 C 语言代码，同时也为承担 C 语言程序设计教学的教师提供必要的参考和启发。

本书的另外一个特点是形式新颖，内容有趣易读，原理讲解细致深刻，知识覆盖面广，并提供书中所有示例源码及其讨论平台，不但是一本提升 C 语言编程水平的读本，也可以为提高读者科技写作能力提供参考。本书适合有一定 C 语言基础的程序员阅读学习，也可作为学习 C 语言程序设计的辅导材料，同时还可作为 C 语言程序设计的教学参考书。无论是 C 语言新手还是有 C 语言基础的程序员，都可以从本书中汲取需要的知识和创意。

本书第 3、4、9 章由聂艳明撰写，第 13、14、15 章由李建良撰写，第 2、8 章由冯妍撰写，其余各章由耿楠撰写，全书由耿楠完成统稿。耿楠、李建良分别编写了配套示例代码，由耿楠对代码进行了系统集成，冯妍对代码进行了审核和校对。西北农林科技大学信息工程学院研究生江旭参加了书稿校对和代码调试工作。

本书的撰写参考了一些在线资料和共享文档以及相关文献，本书基于 LaTeX 工作室提供的开源“ChenLaTeXBookTemplate”LaTeX 模板实现排版。

鉴于作者的学识水平，书中谬误之处在所难免，敬请读者不吝指正。本书的出版得到了西安电子科技大学出版社领导的关怀和支持，陈婷编辑也为本书的出版付出了大量心血，在此一并表示衷心的感谢。

耿楠

2020 年 10 月

陕西杨凌

目录

第 1 章　C 语言的“三观”和提问的智慧

针对 C 语言程序设计在数据组织、代码管理和程序调试中普遍存在的问题，本书提出了 C 语言程序设计的“三观”，即内存观、代码观和调试观。另外，在学习 C 语言程序设计过程中遇到困惑时，如何才能详细、精确地描述问题并及时得到准确答复也是很重要的。根据多年的教学工作经验，在参考相关文献的基础上，本章也总结了 C 语言程序设计中提问的智慧。本章旨在帮助读者建立正确的程序设计“三观”和提问策略，探索获得答案的有效途径。

1.1　C 语言的“三观”

长期以来，在 C 语言程序设计教学中普遍存在着为了减轻学习压力而“简化”教学的现象。例如，用浮点数代替整数，将所有代码都写在“main()”函数中，随意在代码中添加“printf()”“getch()”“system(PAUSE)”函数等。类似这样对 C 语言程序的简化在一定程度上回避了学生在 C 语言学习中学习曲线陡峭的问题，提高了学生学习的积极性。但是往往会使学生忽略 C 语言程序设计的本质，产生“只见树木，不见森林”的片面认识，无法将学到的知识灵活应用于工程实践中。

为此，本书提出了 C 语言程序设计“三观”的概念，即内存观、代码观和调试观，以期读者能够更好地从宏观上掌握程序设计的基本思想。

1.1.1　内存观

程序设计是采用 IPO(Input Processing and Output) 模式对数据进行加工处理和输出的过程。如何高效、准确地组织和管理程序设计中的数据，这不只是使用 C 语言而是使用任何一种语言进行程序设计时，首先应该认真对待的问题。

众所周知，C 语言严格来讲是一种介于汇编语言和高级语言之间的“中级语言”。它是一种强类型的程序设计语言，为数据处理的严谨和灵活提供了必要的语句和语法。但由于目前多数教科书对 C 语言的数据类型只是进行了简单的分类，没有深入分析数据类型是内存组织与管理的本质，甚至一些教材中在整型变量无法满足需要时，用浮点型变量替换整型变量来保存数据。类似这种不完整、不准确的概念和提法，往往会造成读者“死记硬背”且无法灵活运用 C 语言语句和语法的现象。

另外，C 语言中地址类型指针变量的存在，使得在 C 语言程序设计中可以灵活、精确地管理和使用内存。但由于指针的复杂性，一些教材甚至是教师在教学中刻意回避使用指针，并强调对于能用普通变量解决的问题，不要用指针来解决，这对于真正掌握 C 语言程序设计毫无裨益。

以上现象违背了C语言程序设计的基本理念。在C语言程序设计中，建立正确的“内存观”非常重要。在编写任何代码前，一定要深入分析数据的类型，设计合理的数据组织与管理方案，然后再进行后续的程序设计。只有这样，才能合理、准确、高效地使用内存，避免出现“内存浪费”和“内存泄漏”等问题。

1.1.2 代码观

在目前的C语言教学中，为了简化教学任务和应付考试，往往在完成C语言的学习后，所有的代码都写在一个“main.c”文件中甚至在一个“main()”函数中。更有甚者，一些教材中会出现“双击这个.c文件编译/运行程序”的提法。

实质上，C语言是一种结构化的程序设计语言，“函数”是其程序设计的“基本单元”。在实际工程中，往往要求任何规模的程序设计都必须基于函数来实现。

另外，对于实际中规模较大的程序设计，需要按照“工程”的方式，采用“多文件结构”的形式组织程序中涉及的各类代码、数据和其它文件，所有文件相互协作，通过编译/链接，生成最终的可执行文件。

所以，在C语言程序设计中，一定要建立正确的“代码观”，以便更为合理地进行任务分解，完成函数设计，进行代码的分文件组织，用工程的方式管理代码。只有这样才能避免将所有代码都写在一个“main.c”文件甚至是一个“main()”函数中。也只有这样才能建立起“工程”的概念，在保存文件时，不至于只保存一个“main.c”文件，造成工程设置丢失的问题。

1.1.3 调试观

不只是C语言，在使用任何程序设计语言进行程序设计时，总会有BUG发生，因此，在程序设计中不断进行DEBUG是一个不可回避的任务（关于BUG和DEBUG的详细介绍见第4章）。

同样为了简化教学任务和应付考试，大都建议学生不断地用“printf()”“putc()”“puts()”等输出函数输出程序运行的中间结果，甚至会使用“getchar()”“system(PAUSE)”等导致系统无端挂起的语句中断程序运行，以便分析程序的运行状态。这些方式往往无法全面跟踪和剖析程序的运行过程，只能是“头痛医头，脚痛医脚”，并且由于系统无端挂起，会造成更多无法预知的错误。

为此，本书提出了程序设计“调试观”的概念，强调无论何处何时，程序设计中的BUG务必要使用相应的DEBUG工具分析、定位和提出解决方案。只有建立正确的调试观，才能避免在代码里随处使用“printf()”的问题，养成只有在最终输出或输出函数中才使用“printf()”函数的习惯，才能有效地避免随意使用“getchar()”或“system(PAUSE)”挂起程序的恶习！

内存观、代码观和调试观这“三观”是C语言程序设计中重要的“思想”和“理念”，每一个学习和使用C语言的人都应该树立正确的C语言程序设计“三观”。在程序设计中，要树立正确的内存观以有效组织数据；要树立正确的代码观以建立清晰的代码组织结构；要树立正确的调试观以准确定位并解决各类错误。

1.2　提问的智慧

任何人，任何时候，学习任何知识，都不可避免地要碰到各种各样的问题。向有经验的人提问和请教，是任何学习任务中都不可或缺的一个环节。如何高效地提出问题，通过与对方进行交流和沟通“挣”来一个正确和完美的答案，是学习者必须掌握的方法。

1.2.1　提问之前应该做的事情

1. 查阅教材和参考文献

对于大多数的问题，一些教材和参考文献能为你提供标准的解答。因此在提问之前，最好先去阅读相关的教材和参考文献。如果能够在提问的同时表明自己已经阅读过教材和参考文献，但是依旧留有困惑，别人会更加愿意为你解答。如果你不知道如何查找参考文献，那么抓紧时间去学习，这是一门非常重要的课程，并且没有人专门给你讲授这门课程！

2. 检索互联网

C 语言是一个成熟且经典的计算机程序设计语言，从 1969 年开始，人们使用它已有五十多年的历史。初学者遇到的绝大部分问题都能在互联网上找到成熟的答案。因此，在提问之前，最好是先在互联网上搜索。

常用的网站有：

- http://en.cppreference.com/w/；
- https://stackoverflow.com/；
- https://www.csdn.net/；
- 北大、浙大等的 OJ 系统 (Online Judge 系统)。

3. 认真 DEBUG

DEBUG 是 C 语言程序设计中一个不可或缺的工具，通过 DEBUG 可以剖析一个程序中的所有细节，从而判断程序中 BUG 出现的位置、种类、严重程度等。解决这些 BUG，程序便能够正常运行。DEBUG 特别适用于解决程序中的逻辑错误。

在提问之前一定要认真完成程序的 DEBUG，通过设置断点暂停程序的运行，以查看或改变程序的变量值、函数调用结果、CPU 的状态和内存的状态等，从而实现程序运行过程的剖析、错误的定位和错误的分析。在提问之前，应该准备 DEBUG 的过程截图、内容记录等信息。

4. 相互讨论

三人行必有我师，在学习过程中，身边一定有许多精于此道的同学和朋友，要多向这些同学和朋友请教，与同学和朋友展开讨论和分析。

理不辩不明，讨论分析得多了，对问题的理解也就更加深入，可以更为清晰地分析和解决问题了。也许经过与同学和朋友的讨论，很多问题及相关疑问就轻易得到了解决和解答。

1.2.2 提问模板

为了能够详细、精确、及时地得到问题的分析和处理方法，给回答的人提供全面、有效的信息是非常必要的。根据作者多年的教学工作经验，以 C 语言课程的提问为例，这里给出两个基本的提问模板。

1. 语法错误 (编译错误)

语法错误是在程序编写过程中产生的，编译器通过词法和语法分析完全可以精确地定位错误的位置和错误的类型。因此在出现语法错误时，需要提供全面的编译器编译内容和链接错误及警告信息，可参照下面的模板进行提问。

> 编译遇到了错误，请问应该如何解决？是否还需要提供更多信息？谢谢！
> **编译报错：**
> 在这里填写编译报错信息截图（Code::Blocks 的 Logs & others 输出窗格中的 Build messages 标签中以 error 或 warning 开始的信息）。
> **最小工作示例代码：**
> 填写 MWE 请勿截图，切记要提供源代码！
> **问题描述：**
> 请使用缩进和空行等策略有条理、清晰地对遇到的问题进行罗列和表达。
> **编译链接参数：**
> 如有必要，请详细描述编译链接参数 (Code::Blocks 的 Logs & others 输出窗格中 Build messages 标签中的信息)。

2. 逻辑错误 (程序运行崩溃或运行结果与期望结果不符)

如果编译和链接没有错误，而是在程序运行中出现崩溃或者是程序运行结果与期望结果不相符，则这样的错误大多数是逻辑错误。逻辑错误当然是因为程序设计的逻辑不正确造成的，解决逻辑错误一般离不开 DEBUG，可参照下面的模板进行提问。

> 程序出现逻辑错误，请问应该如何解决？是否还需要提供更多信息？谢谢！
> **当前输出结果：**
> 提供相应控制台 (黑窗口) 输出截图。
> **功能描述与预期结果：**
> 对程序功能及预期结果进行详细、精准的描述，或提供预期效果截图 (也可以是其他人的正确结果截图)。
> **最小工作示例代码：**
> 填写 MWE 请勿截图，切记要提供源代码！
> **编译链接参数：**
> 如有必要，请详细描述编译链接中的参数 (Code::Blocks 的 Logs & others 输出窗格中的 Build log 标签中的编译日志)。

1.2.3　提问时的建议

1. 整理 MWE

MWE(Minimal Working Example) 意思是“最小工作示例”。顾名思义, MWE 的特点有三个: 简短 (不包含与问题无关的代码片段)、工作 (能够独立运行于他人的计算机上, 而不需要再添加额外的代码)、示例 (在他人计算机上的运行结果, 能完整地再现你所遇到的问题)。

时间对任何人都是一笔宝贵的财富。如果提供的代码冗长, 回答者势必要花费大量时间阅读不必要的代码。这样会浪费回答者的时间, 提问者会等待更久。因此有必要让示例足够简短, 而且尽量能够再现错误。

特别需要注意的是: Code::Blocks 中是以工程的方式管理代码, 提供完整的“工程文件”, 而不只是一个“*.c”或“*.h”文件, 要将“*.cbp”“*.layout”等与工程相关的文件一并提供, 以便他人能够运行代码。

建议: 一般需提供 MWE 所在文件夹下除了 “bin”和“obj”文件夹的其余所有文件。

2. 给出完整准确的错误提示

在 Code::Blocks 中编译/链接错误信息在 Logs & others 输出窗格中的 Build messages 标签中输出, 如图 1.1 所示。

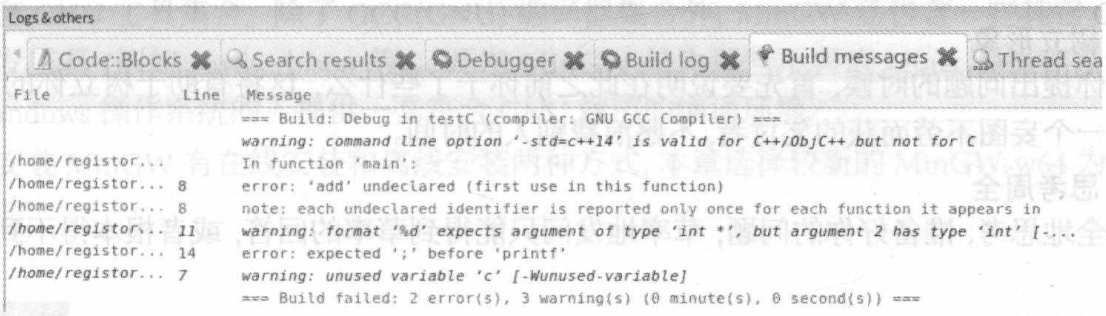

图 1.1　编译错误信息

错误截图应该完整, 若过长, 则要提供 Logs & others 输出窗格中的 Build log 标签中的编译/链接日志信息, 如图 1.2 所示。

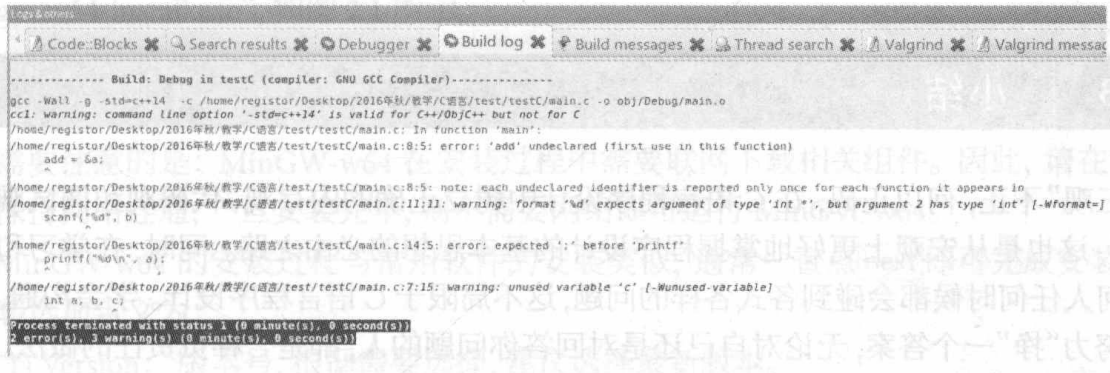

图 1.2　编译日志信息

3. 提问者常犯的错误

提问者常犯的错误: 一是只提供代码截图; 二是只提供代码片段; 三是对代码冗余不做删减。

(1) 永远不要提供代码截图，应直接提供源代码。C语言是一种非常灵活的语言，同时也意味着复杂。除了一些经典的问题，大多数问题通常无法通过“看一眼”就能解决。这意味着，回答者通常需要在自己的计算机上运行代码。如果只是给出了代码截图，回答者就不得不自己在计算机上重新输入你的代码。这样一方面会浪费很多时间(通常也不会有人乐意这样做)，另一方面，在重新输入代码的过程中，有可能会产生新的错误，从而无法完全复现所遇到的问题。

(2) 永远不要只提供代码片段，应提供完整的工作示例。众所周知，如果一个人感冒流涕，他不会只把自己的鼻子送给医生去检查。同样，提问者也不能只提供问题的一部分代码，因为问题可能出现于其它地方。如果只提供代码片段，则大多数的问题是无从解决的。特别是当涉及数据文件、第三方库等内容时，应该同时提供需要的数据文件以及需要的第三方库相关配置说明。

(3) 永远不要提供不加删减的冗长代码，应提供简短的工作示例。将未做删减的代码全部提供给别人，回答者往往要在工作、学习之余帮你解决问题，而没有人乐意在休息时间阅读一段乱码或垃圾代码。

1.2.4 提问者要谨记

1. 树立形象

当你提出问题的时候，首先要说明在此之前你干了些什么，这将有助于树立你的形象：你不是一个妄图不劳而获的乞讨者，不愿浪费别人的时间。

2. 思考周全

周全地思考，准备好你的问题，草率地发问只能得到草率的回答，或者根本得不到任何答案。

3. 尊重他人

决不要自以为够资格得到答案，你没这种资格！你要自己去“挣”回一个答案，靠提出一个有内涵、有趣、有思维激励作用的问题去“挣”到这个答案。

切记，你要自己去“挣”回一个答案，这一点极其重要！

1.3 小结

“三观”不正，何以正码，在C语言程序设计中树立正确的内存观、代码观和调试观非常有必要，这也是从宏观上更好地掌握程序设计的基本思想的必由之路。同时，在学习和工作中，任何人任何时候都会碰到各式各样的问题，这不局限于C语言程序设计。学会沟通、高效提问，努力“挣”一个答案，无论对自己还是对回答你问题的人，都是一种负责任的做法。