普通高等教育“十一五”国家级规划教材

计算机程序设计基础

（第3版）

乔林 编著

高等教育出版社

普通高等教育

计算机程序设计基础

（第3版）

乔林 编著

高等教育出版社·北京

内容提要

计算机程序设计是高等学校计算机基础课程中的核心课程，具有大学基础课的性质。本书以C语言程序设计为基础，注重讲解程序设计的基本概念、方法和思路，培养读者的基本编程能力、逻辑思维与抽象思维能力。

本书主要内容包括：程序设计的基本概念、C语言的基本语法元素、程序控制结构、函数、算法、程序组织与库的设计、数组、字符串、结构体与指针等复合数据类型、文件与数据存储、程序抽象等。希望通过强调那些在程序设计与软件开发过程中起重要作用的思想与技术，使读者体会并初步掌握较大型或实用程序的编写与设计能力。本书行文严谨流畅，语言风趣幽默，示例丰富生动，习题难度适中。

本书可作为高等院校计算机及理工类各专业、成人教育院校程序设计课程的教材，也可供计算机应用开发人员及相关人员自学。

图书在版编目（CIP）数据

计算机程序设计基础/乔林编著. --3版. --北京：高等教育出版社，2018.1

ISBN 978-7-04-047766-5

Ⅰ. ①计… Ⅱ. ①乔… Ⅲ. ①C语言-程序设计-高等学校-教材 Ⅳ. ①TP312

中国版本图书馆CIP数据核字（2017）第112130号

策划编辑 刘 茜	责任编辑 唐德凯	封面设计 李卫青	版式设计 马 云
插图绘制 杜晓丹	责任校对 刘娟娟	责任印制 耿 轩	

出版发行	高等教育出版社	网 址	http://www.hep.edu.cn
社 址	北京市西城区德外大街4号		http://www.hep.com.cn
邮政编码	100120	网上订购	http://www.hepmall.com.cn
印 刷	北京鑫海金澳胶印有限公司		http://www.hepmall.com
开 本	850mm×1168mm 1/16		http://www.hepmall.cn
印 张	19.75	版 次	2009年8月第1版
字 数	430千字		2018年1月第3版
购书热线	010-58581118	印 次	2018年1月第1次印刷
咨询电话	400-810-0598	定 价	45.00元

本书如有缺页、倒页、脱页等质量问题，请到所购图书销售部门联系调换

物料号 47766-00

前言

本书写作目标可用8个字概括：开卷有益，开卷有趣。

一、本书旨趣

六年来的教学经验表明，学生在学习程序设计类课程时最难的地方不是掌握某种程序设计语言的语法规范，而是掌握程序设计的基本方法。

程序设计语言的语法规范是死的，并且与任何一种自然语言相比，程序设计语言的语法规范更为简单。因此，对于任何学生而言，只要花费一定量的时间（并且这个时间并不长），通过记忆并辅以上机验证，完全可以掌握甚至精通其语法规范。

然而，问题在于，在学生已经预习课程并认真听讲，自认为已掌握了程序设计方法后，却发现在解决实际问题时经常是毫无头绪，一筹莫展。这种心理落差与由此造成的心理恐慌对学生的学习热情来说是致命的——这事实上是课程教学无法获得应有成果的主要原因。因此，本书力图在学习语法规范与培养实际动手能力之间架设一座沟通的桥梁，以解决程序设计教学中理论与实践脱节的问题。

对于程序员而言，抽象贯穿程序设计与开发活动的始终，是应着重强调的最核心概念。事实上，如果让笔者仅使用一个词来表达本书旨趣，那只能是抽象。程序员对抽象的理解与把握左右着程序的质量与效率。所以，抽象正是本书所要架设的桥梁。

坦率地说，抽象思维能力的培养不是一本教材、一门课程所能够涵盖和阐释清楚的。但作为一种有益的尝试，本书希望能够通过一种有趣的、面目可亲的方式向读者说明抽象在程序设计中所能够和应该起到的作用，以及抽象思维能力对读者未来学习、工作的重要意义。

Vinoski 有云："Finding the right balance [between abstraction and pragmatism] requires knowledge, experience, and above all, thought." 翻译成汉语就是"于抽象与具象间寻觅正道需要知识、经验，更需要思想。"希望读者能够记住这句话。

二、篇章结构

本书的篇章结构与传统C语言教材不同。本书如此编排知识点的根本出发点在于，笔者

希望能够以培养读者解决实际问题能力和抽象思维能力为主线，而不是以语言语法知识点为主线。这么做的好处是，读者不会在一开始就接触到过多的C语言语法规范的细节，避免在实际编程时受到“学习知识过多”所造成的干扰，从而能够将注意力集中到解决实际问题中去。

本书共分10章，具体组织方式如下。

第0章简单回顾C语言的历史，介绍C语言与程序设计的一些基本概念，然后通过两个小例子说明了C语言编程的基本流程。

第1章与第2章介绍C语言的基础知识，这些知识包括数据与数据类型、表达式与语句、运算、基本输入输出功能、枚举类型与布尔类型等用户自定义类型、典型的分支与循环结构等，它们是程序员构建C程序宏伟大厦的一砖一瓦。此外，第1章还专辟一节讨论程序设计风格问题，良好的程序设计风格是编写优秀代码的要件之一。作为后续章节的先导，第2.7节讨论问题求解与结构化程序设计的基本方法，以使读者大概了解结构化程序设计的思路。

第3章与第4章着重研究函数与算法，包括函数声明与调用、函数定义、函数调用规范、程序的结构化与模块化、程序测试与代码优化、算法的概念与特征、算法的描述方法、算法设计与实现、递归算法、容错、算法复杂度等内容。这些知识同样是基本的。

第5章讨论程序组织与软件工程，主要研究库与接口的概念、作用域与生存期、宏与条件编译等知识。通过两个具体的实例——如何设计随机数库以及如何使用随机数库设计一个猜测价格的游戏程序说明程序开发的基本流程以及在进行程序开发时要着力关注的问题。

第6章研究C语言提供的复合数据类型，主要涵盖字符与字符串、数组、结构体三方面知识。这些知识其实并不复杂，在本书力图淡化细节的主旨下尤其如此。此外，本章还讨论简单数据集上的简单查找与排序操作。

第7章研究指针。C语言的指针非常灵活，事实上它与其他语法要件（例如函数、数组、字符串、结构体等）有着千丝万缕的联系。专列一章讨论指针是非常有必要的，它起到了承前启后的作用。

第8章讨论文件与数据存储。在引入文件的基本概念与基本操作之后，本章研究文件的4种读写方法以及实际编程时的数据存储策略。

第9章研究程序抽象。本章是本书最重要的一章，前面8章对抽象的认知将在本章得到进一步提升，通过学习数据抽象与算法抽象的基本方法与原则，透彻地理解抽象数据类型、链表数据结构、函数指针在构造抽象程序时的意义，掌握程序抽象的思考方法。

本书的组织结构也与传统C语言教材不同。双色双栏的排版格式主要不是为了美观，而是为了与全书知识结构相协调。笔者特意为本书安排两个线索：一个以程序抽象机制为主线的主体结构着重研究程序设计方法，与程序抽象有关的部分C语言语法元素得以详细展开；另一个为辅线，补充部分基础知识，对某些知识点进行额外阐述，或对主线结构下的知识点按照便于查阅的方式重新编排。笔者希望这样富有特色的编排体系能够突出本书的重点和难点，为读者的学习带来方便。

此外，特别需要说明的是，为了在保持本书结构完整性的前提下完成抽象思维能力培养的教学目标，本书绝大多数例题与习题需要使用笔者实现的函数库。函数库已在Turbo C、Borland C++ Builder、Dev-C++（使用GCC编译器）、Microsoft Visual C++等编译器或开发环境中调试通过，读者可直接使用。为降低成本，库的源代码没有随书提供，读者可以从高

等教育出版社网站获得，具体网址为 http://www.hep.com.cn；配套的习题解答与上机指导书也包含了函数库的全部源代码，读者也可以参照该书。

三、学习建议

很多学生曾问，什么时候应该学习一门程序设计语言，以及如何才能学好它？对这样的问题，三言两语很难解释清楚。

理想情况下，语言的学习应该遵从实际的需要。当需要使用该语言解决实际问题时，才应该花费宝贵的时间学习它。但是，很多学生其实并不了解自己是不是需要学习某种特定的程序设计语言，或者并不知道自己未来会不会需要用这样的程序设计语言来解决实际问题。显然，如果要读者在需要使用该语言解决实际问题时再来学习它，笔者估计大多数读者都会反对这样的观点。另一方面，一旦面临找工作的压力，如果这个没有学，那个也没有学，履历看上去就会有些不美观了。更何况还有不同专业的教学计划呢——现实情况是，一旦教学计划定下来，什么应该学，什么可以不学其实并没有多少选择的余地！

未雨绸缪总是不错的，那么读者和笔者就达成了一个基本共识，那就是要学习“程序设计基础”这门课程——读者如果不是必须或者希望学习这门课程，就不会看到这部分文字，不是吗？那么，读者能够从一门未来极有可能很少甚至不会使用的程序设计语言中学习什么东西，才能够获得尽可能多的知识和经验呢？

类似于自然语言的学习，如果某个单词或语法结构在未来的学习和工作中永远不会被用到，学习它的意义显然几乎为零。程序设计语言也一样，记住 C 语言的全部语法规范细节但却从来不用它同样也是毫无意义的。

笔者以为，在课程中读者要学习三部分内容：第一，一旦掌握了某种语法规范，可以使用它解决什么样的问题；第二，一旦解决问题的方法和途径需要使用还没有掌握的知识，这些知识如何获得；第三，如何进行正确的思考，寻找解决问题的方法和途径。

第一条属于知识记忆、理解与复现范畴，要解决的是跨越程序设计领域门槛的问题。在此，笔者以为，这些知识应该很好地裁减以适应学习需要，也就是说，门槛不能定得太高。事实上，贯穿本书的思想就是读者只需要了解最基本的知识就够了，这足以解决本书中大多数有趣并有实际意义的问题。更重要的是，能够在很短的时间内完成这些知识的学习并应用于实践，这对于获得成就感与愉悦感，延长学习热情至关重要。本书之所以如此大胆地去除部分 C 语言知识点，起因即为此。

第二条属于再学习能力培养问题。一般而言，学习到的知识未来能够直接使用到的不到 20%，余下的知识怎么办？学习，学习，再学习，仅此而已。这同样表明，所有未雨绸缪的努力都不可能面面俱到。在解决实际问题时，应该学会“大胆假设，小心求证”。例如，如果需要为班级里 30 位学生创建一个通信录程序，并提示将每位学生的姓名、年龄、电话号码等具体信息存储在一起才合乎逻辑，那么解决问题时就应该采用这样的方法来进行，即使到目前为止还不知道 C 语言中应该采用什么样的语法规范来描述上述逻辑，但是应该相信这样的语法规范一定是存在的。有了这样的认知，再去学习结构体与数组，就会发现这些语法结构是那么直接而富有效率，哪里有那么多的难以理解之处？

第三条属于创造性的知识应用问题。这一条实在太重要了，学习知识和技能的唯一目的

是为了解决问题，否则，人与书橱和 Internet 有何差别？那么，如何解决实际问题呢？知识储备固然重要，但更重要的是发现问题以及提出解决问题的方法。显然，仔细分析问题，对问题进行抽象描述，在此基础上形成最终解决方案并得出结论是创造性解决实际问题的必由之路。

例如，一个苹果加上两个苹果是 3 个苹果，抽象成算术就是 1 + 2 = 3。这很简单，而 $a + b = c$ 就并不那么简单了。从实物运算抽象到算术，进一步抽象到代数，再抽象到抽象代数，人们对数与算的认识逐渐深化。程序设计也同样，程序抽象活动并不是简单的、一蹴而就的，为获得问题的满意解答，可能需要多层次、多角度的抽象。例如，从文字常数抽象出量，从算术运算抽象出表达式，从操作序列抽象出语句与语句序列，再进一步抽象出程序控制结构，抽象出一个又一个的函数；从数据的性质抽象出各种各样的数据关系，并通过数组、字符串、结构体、指针、文件等数据类型或数据结构固定下来，从数据与代码的关系抽象出函数指针，以构造灵活的能够适应未来需要的抽象程序代码。这些都是抽象。可以这么说，读者对事物的认识越深刻，越关注于一般问题的解答，就越能理解抽象在程序设计中所起的关键作用，也越能从抽象中获得益处。

因此，关于课程学习，笔者有如下建议。

第一，阅读。多阅读程序代码，尤其是 C 语言标准库以及大量开源软件的源代码。有诗曰："熟读唐诗三百首，不会作诗也会吟。"阅读大量优秀源代码，对于培养良好的编程风格与编程习惯，了解典型的数据组织方式与算法实现策略有极大的好处。

第二，实践，尽可能多地实践。仅仅完成代码阅读是不够的，要想将知识转化为自己的实际技能，最重要的检验准则就是能否使用它解决实际问题。很多初学程序设计的学生向笔者诉苦，他们对于编程的最深刻体验就是"一看就明白，一用就不会"。最根本的原因就是实践太少。按照笔者的理解，要想真正精通一门程序设计语言，花费 10 倍的编程实践时间（与书本学习时间相比）是必要的。

第三，在实际编程时，建议读者回答下述两个问题：① 我所编写的程序能够解决本问题吗？② 我所编写的程序能够不加修改或很少修改就解决另一个看上去与前一问题似乎完全不同的问题吗？回答了这两个问题，就可以骄傲地宣称自己已得窥堂奥，进入了程序设计的自由天地。

为了锻炼读者的实际动手能力，本书提供了大量习题，其中绝大多数习题都非常有趣。这些习题基本涵盖了知识复现、编程验证、知识融会贯通训练与挑战性项目实践等多个方面。建议读者在时间允许的情况下完成本书所有习题，即使部分挑战性项目一开始看上去似乎过于困难，但认真思考总会有宝贵的收获。

总之，对于程序设计课程学习而言，会记固然重要，会想更重要；知识固然重要，能力更重要。笔者希望读者在学完本书之后，能够得出"这本书不仅深入，而且浅出；不仅深刻，而且好玩"的结论。

四、教学建议

在教学方面，应以课堂讲授与上机实践相结合的手段来进行。理想的授课时数与上机时数为 48 + 64 学时（48A），经过合适的裁减与教材组织，本书同样可以适应 48 + 48 学时（48B）、

32 + 48 学时（32A）与 32 + 32 学时（32B）。具体课时安排可参考表 1。

表 1 课时安排

章	48A	48B	32A	32B
第 0 章 C 语言概述	2 + 2	2 + 2	2 + 2	2 + 2
第 1 章 C 语言基本语法元素	3 + 3	3 + 3	3 + 3	3 + 3
第 2 章 程序流程控制	6 + 6	6 + 4	4 + 6	4 + 4
第 3 章 函数	4 + 6	4 + 4	3 + 4	3 + 3
第 4 章 算法	4 + 6	4 + 4	3 + 4	3 + 3
第 5 章 程序组织与软件工程	7 + 9	7 + 7	3 + 6	3 + 3
第 6 章 复合数据类型	6 + 6	6 + 6	4 + 6	4 + 4
第 7 章 指针	6 + 8	6 + 6	4 + 6	4 + 4
第 8 章 文件与数据存储	3 + 6	3 + 3	2 + 3	2 + 2
第 9 章 程序抽象	7 + 12	7 + 9	4 + 8	4 + 4

说明：

（1）如果上机课时只有 48 或 32 个学时，则可以选择难度较低的习题或者少布置部分习题。

（2）如果授课课时不足，则部分内容可以简要论述或略过。

① 对于第 2 章，第 2.7 节可以简要论述，不再详细展开。

② 对于第 3 章，第 3.4 节可以简要论述，第 3.5 节可以省略。

③ 对于第 4 章，第 4.6 节可以简要论述或略过。

④ 对于第 5 章，第 5.4～5.6 节可以简要论述。

⑤ 对于第 6 章，第 6.5 节可以简要论述或略过，并适当压缩其他各节内容。

⑥ 对于第 7 章，第 7.5 节可以简要论述或略过。

⑦ 对于第 8 章，第 8.4 节可以简要论述或略过。

⑧ 对于第 9 章，第 9.1 与 9.4 节可以简要论述。

教学组织上，建议以有趣的习题展开知识点的讲授，适当补充一些课本以外的例题更好。课堂上的师生互动非常必要，这有助于引导学生自己获得正确的解决方案。如果能够在课堂上利用例题穿插一些游戏，效果会更好。

成绩考核与评定上，建议以大作业为主，并辅以平时作业和笔试。如果时间允许，能够安排全部或部分学生进行大作业答辩，效果会更好。笔试不是必要的，但如果不进行笔试，教师对教学效果心里没底，适当安排小测验或期末考试也是可行的。此时建议试卷尽量考核基本知识点，检验学生抽象思维能力培养方面的效果，而不应将注意力集中到语法细节上。当然，如果学生人数较多，没有助教或助教人手严重不足，大作业并不可行，则平时作业与期末考试的比重就应仔细权衡。此外，能够组织上机考试显然非常好，不过这对学校的硬件条件提出很高的要求，属于不能强求之列。

五、致谢

本书非一人之功也。特别感谢课程组的王行言、冯铃、黄维通、郑莉、刘宝林、孟威、余小沛老师，在课程建设与教材写作的过程中，与他们的讨论使作者获益良多，本书的最终

完成离不开从诸位同仁那里获得的宝贵经验和帮助。感谢李秀、田荣牌、汤荷美老师，她们为本书的完成提供了有力支持。几年来课题组的各位助教在承担繁重的上机辅导与批改作业任务之余，参与了本书例题与习题的设计与编程实现，感谢刘明亮、曾富涔、袁扣林、韩磊、吕璨、于雪璐、那盟、王炜、戴德承、刘芳琴、王倩、王永才、王桂玲、颜莉萍等诸位同学的辛勤劳动。

再次向上述同仁和助教表示真挚的谢意。

最后，衷心希望本书写作目标已达到。

乔 林

2017 年 1 月 31 日于清华园

目录

第0章　C语言概述

学习目标

1. 了解C语言的历史与特点。
2. 了解程序与程序设计的基本概念。
3. 初步认识算法在程序设计中的作用与地位。
4. 了解数据与数据结构在程序设计中的作用与地位。
5. 了解C程序的基本特征。
6. 掌握C程序的编译、调试与运行过程。

0.1 C语言简介

几十年来，作为计算机软件的基础，程序设计语言不断得到充实和完善，功能全面、使用方便的程序设计语言相继问世。在种类繁多的计算机程序设计语言中，20世纪70年代初诞生的C语言具有重要地位，它是描述、开发系统软件和应用软件（包括科学计算软件）的重要工具之一。

0.1.1 C语言简史

1972年，贝尔实验室的Dennis M. Ritchie发明了著名的C语言。1973年，Ken Thompson与Dennis M. Ritchie通力合作将UNIX中90%以上的代码用C语言重新实现。随着UNIX操作系统的广泛应用，C语言迅速得到推广，名闻天下。

随着C语言支持的计算机系统越来越多，C语言的变体也越来越多。这种C语言变体称为方言。C语言的方言有上百种，它们虽然功能基本一致，但实现上并不完全相同，因而有时并不能完全兼容。方言的存在对计算机应用技术发展十分不利。

有鉴于此，美国国家标准化协会（ANSI）于1983年专门成立了C语言标准委员会，并于1989年完成了C语言的标准化工作。此版本的C语言标准简称C89。1990年，ANSI C标准被国际标准化组织（ISO）接受为国际标准（ISO/IEC 9899：1990），称为ANSI/ISO Standard C，简称为C90。此后，1999年推出的C99标准在保留C语言特性的基础上，吸收了其继承者C++的部分特性，并增加了部分库函数。

目前可在微机上运行的C编译器和开发环境很多，如Microsoft Visual Studio、Intel C/C++、Dev-CPP、Borland C++ Builder、GCC等。

0.1.2 C语言特点

C语言之所以能够流行，主要因其有如下特点。

（1）C语言是介于高级语言与低级语言之间的“中级语言”。它既具有高级语言结构化与模块化的特点，也具有低级语言控制性与灵活性的特点。C语言语法简洁紧凑，使用方便灵活，操作符与数据类型丰富，语法限制不严格，程序设计自由度较大，允许对位、字节和地址这些计算机基本成分进行操作，生成的目标代码质量较好，程序运行效率较高。

（2）C语言是结构化程序设计语言。C语言提供了多种结构化语句，直接支持顺序、分支和循环3种基本程序结构，便于程序设计人员采用“自顶向下逐步求精”

的结构化程序设计技术。

（3）C 语言是模块化程序设计语言。C 语言程序由一系列函数组成，函数为独立子程序，这种结构便于将大型程序划分为若干相对独立的模块分别予以实现，模块间通过函数调用来实现数据通信。

（4）C 语言具有很好的可移植性。虽然 C 语言存在很多方言，但只要在编写程序时使用的是标准 C，就可以很容易地将为某种计算机系统编写的程序在另一种机器或另一种操作系统上编译运行起来。

0.2 程序设计的基本概念

在介绍 C 程序设计前，首先认识几个与程序设计有关的基本概念。

0.2.1 程序

所谓程序或应用程序，是指一系列遵循一定规则和思想并能正确完成指定工作的代码，也称为指令序列。通常，计算机程序主要描述两部分内容：一是描述问题中的全部对象及它们之间的关系；二是描述处理这些对象的规则。这里，对象及其关系涉及数据结构，而处理规则则涉及问题求解算法。因此，对程序的描述，经常有如下等式：

程序=数据结构+算法

一个设计合理的数据结构往往可以简化算法，而好的算法又使程序具有更高的运行效率，并使程序更容易阅读与维护。

0.2.2 程序设计与程序设计语言

所谓程序设计，就是根据计算机所要完成的任务，设计解决问题的数据结构和算法，然后编写相应的程序代码，并测试该代码运行的正确性，直到能够得到正确的运行结果为止。通常，程序设计应遵循一定的方法和原则，而不能是个人的随意行为。

良好的程序设计风格是程序具备可靠性、可读性、可维护性的基本保证。因此，可进一步对程序设计做如下定义：

程序设计=数据结构+算法+程序设计方法学

该描述强调了程序设计方法学在程序设计中的重要性。

在编写程序代码时，程序员必须遵照一定的规范来描述问题的解决方案和解决步骤，这种规范就是程序设计语言。像人们的自然语言一样，计算机程序设计语言也具有一些基本原则，具有固定的语法格式、特定的意义（语义）和使用环境（语

用），并且这些基本原则比自然语言更严格。

0.2.3 算法

所谓算法，就是问题的求解方法和步骤。通常，一个算法由一系列求解步骤组成。正确的算法要求算法规则和算法步骤的意义是唯一确定的，不能存在二义性。这些规则指定的操作是有序的，必须按算法指定的操作顺序执行，并能够在有限的执行步骤后给出正确的结果。

算法的二义性与模糊性

相传以前有位老先生对某位后生不满意，出一上联要后生对：

眉先生，须后生，后生却比先生长；

这里的“先生”与“后生”就是典型的双关语，具有二义性。那后生也是有点急智的，转眼对了下联：

眼珠子，鼻孔子，珠子反在孔子上。

这里的“珠子（朱子，指朱熹）”与“孔子”自然也是双关语，对得绝妙。在现实生活中，双关语不仅幽默，也非常有意义。

然而在设计计算机程序时，计算机只能死板地按照事先定义好的步骤一步一步执行，“先生”到底是指人呢，还是“先出现”的过程？像这样的二义性超出了计算机的可理解能力。

同时，算法也不允许存在模糊的概念。例如“来2克胡椒再加3克盐”有明确的概念，这在程序的计算或控制中是可以操作的，但“来一点胡椒加一点盐”就无法操作了。程序是按照人的具体要求来完成相应工作的，所以除非程序中能对“一点”进行明确的数值设定，否则“一点”一点意义都没有。

通常，算法的设计过程是逐步求精的。算法的设计者首先给出粗略的计算步骤框架，然后再对框架中的内容逐步细化，添加必要的细节，使之成为更加详细的描述。当然，在大多数情况下，细化工作不可能一步到位。此时，更进一步的细化工作就是必要的。这个过程一直到能够把需求通过编程语言完全描述清楚为止。

流程图是描述算法的常用工具。流程图也称为程序框图，是算法的图形描述。流程图比程序代码更直观，更易阅读和理解。流程图只是一种算法的表达工具，除非具有专业的应用程序，否则计算机很难识别流程图，更不能直接执行流程图。

与流程图相比，有经验的程序员在编程前更习惯使用伪代码进行纸面作业，完成程序代码的设计与实现。所谓伪代码就是类似于自然语言的说明性短语或符号。

一个问题可以有多种解决方案，因而解决问题的算法并不唯一。这些算法的执行效率有高有低，其差异甚至有可能大到人们无法想象的地步。

0.2.4 数据与数据结构

数据结构指数据对象以及这些数据对象的相互关系和构造方法。程序中的数据结构描述了程序中被处理的数据间的组织形式和结构关系。

数据结构与算法密不可分，一个良好的数据结构将使算法简单化；而明确了算法，才能更好地设计数据结构，两者相辅相成。

在解决实际问题时，程序员不仅要实现算法，还必须完整实现作为算法操作对