高等学校电子信息类“十二五”规划教材

程序设计基础

李 军 编著

西安电子科技大学出版社
http://www.xduph.com

高等学校电子信息类“十二五”规划教材

程序设计基础

李　军　编著

西安电子科技大学出版社

内 容 简 介

本书主要介绍 C 语言环境下进行程序设计的基本思想、方法和技巧。本书共 12 章，内容包括程序设计概论、C 语言概述、算术运算程序设计、逻辑运算与流程控制、常用基础算法与程序设计、模块化程序设计技术、批量数据处理程序设计、文本信息处理程序设计、结构数据类型、在磁盘上存取数据、位运算、编写大型程序等。

本书内容丰富、取材新颖、叙述通俗易懂，可作为计算机及其他理工科各专业的程序设计课程教材，也可作为编程爱好者的参考书。

图书在版编目(CIP)数据

程序设计基础/李军编著.
—西安：西安电子科技大学出版社，2014.1
ISBN 978-7-5606-3233-9

Ⅰ. ① 程… Ⅱ. ① 李… Ⅲ. ① C 语言—程序设计—高等学校—教材
Ⅳ. ① TP312

中国版本图书馆 CIP 数据核字(2013)第 249027 号

策　　划　毛红兵
责任编辑　王　瑛　毛红兵
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　2014 年 1 月第 1 版　2014 年 1 月第 1 次印刷
开　　本　787 毫米×1092 毫米　1/16　印张 20.5
字　　数　485 千字
印　　数　1～3000 册
定　　价　36.00 元
ISBN 978 – 7 – 5606 – 3233 – 9 / TP
XDUP 3525001 – 1

前　言

“程序设计基础”是一门理论与实践相结合的普通理工科大学生计算机入门课程，在计算机学科的教学中具有十分重要的作用。本课程能够培养学生运用程序设计语言求解问题的基本能力，让学生了解高级程序设计语言的结构，掌握计算机问题求解的基本思想以及基本的程序设计过程和技巧，熟悉并适应计算机问题求解模式，即从描述问题、建立模型、设计算法到编写程序、测试程序、分析结果的一系列过程，培养学生将问题抽象化，设计与选择解决方案，以及用程序设计语言实现方案并进行测试和评价的能力。

目前，国内关于程序设计基础的教材较多，按照教材内容的组织方式概括为两种，或以语言知识为主线，或以程序设计为主线。以语言知识为主线的教材内容组织大多是以某个具体的程序设计语言(如 C 语言)知识及其自身体系为脉络展开的，对语言语法知识的介绍较为细致，对程序设计思想方法的介绍较少，而且教材中所配程序是为了验证和解释语言的语法，学生使用这类教材后普遍反映掌握语法规则容易，编程应用困难。以程序设计为主线的教材，以算法为导向，重点介绍程序设计的思想与方法，对程序设计语言知识的介绍比较粗略，这类教材起点高，初学者使用起来比较困难。

本书在展示 C 语言强大程序设计功能的同时讲授了程序设计的基本思想、方法和技巧。本书的特点如下：

1. C 语言知识介绍方面的特点

(1) 从易到难，循序渐进。C 语言的各个主题顺序是根据初学者的学习需求来安排的，用了比较多的篇幅从应用的角度讲解 C 语言的基础知识。第 2～4 章重点介绍 C 语言程序的基本结构、变量、指针、表达式、程序的基本流程控制结构，通过这 3 章内容的介绍，读者能够很快上手，并有能力编写一些处理简单数据结构问题的程序；第 6 章介绍函数的概念，以便读者掌握模块化程序设计的工具；第 7～10 章介绍复杂数据结构、用文件进行数据的存取等知识，以便读者掌握处理复杂数据的方法；第 11 章介绍位运算的相关知识。

(2) 突出重点，详略得当。C 语言的语法比较庞杂，有些语句可以相互代替，有的语法应用较少。本书重点介绍基本的、常用的、不可或缺的 C 语言知识，而对一些非重点的 C 语言要素进行淡化处理。如本书带“*”号的章节可作为自学内容。

(3) 分散难点，化难为易。指针既是 C 语言的重点，又是一个应用难点，它与变量、函数参数、数组、数组下标和字符串等概念密切相关，增加了 C 语言学习的难度。本书没有设置专门的章节介绍指针，而是伴随着变量、函数、数组等概念从不同角度来分散介绍指针，分解指针知识的教学难度，以便读者能够逐步掌握关于指针的知识，增加应用指针的机会。

2. 程序设计方面的特点

本书在程序设计思想、方法和技巧等方面采用直接介绍与隐含融入相结合的方式。

(1) 第 1 章直接介绍了程序设计方法的一些基础知识，讨论了问题求解与软件开发之间的联系，为读者进行程序设计做了必要的铺垫；第 5 章以常用算法为纽带，实现逻辑思

维与程序设计方法的有效融合，针对迭代策略、穷举策略，以不同的有趣实例对算法进行综合应用，让读者体验程序设计的实现过程，既反映问题难度及求解规模上的变化，又彰显知识和求解方法的多样性。

(2) 将程序设计方法的讲解融入到例题与案例中。书中的例题不是对事先编写好的程序进行解释，而是对例题采用启发式的解决方式启发读者根据问题的描述对问题进行分析，从中找出解决问题的方法，进而设计算法，编写程序代码。

(3) 融入了过程抽象与数据抽象的概念。本书在第 6～9 章中融入了过程抽象与数据抽象的概念及方法，第 12 章对过程抽象与数据抽象进行了总结。

3. 其他方面的特点

(1) 几乎每章都有一个案例，均可用该章所学的 C 语言知识及软件开发方法来解决。案例采用逐步扩展、加入条件的方式，从描述问题、建立模型、设计算法、编写程序等方面进行介绍，贴近实际，不断地激发读者对知识的探索欲望。

(2) 几乎每章都有一节自学内容。自学内容主要讨论常见的编程错误与查找错误的方法，介绍程序测试和调试的简单方法与技巧。

(3) 例题尽可能地采用完整的程序或函数，很少使用不完整的程序片段。

(4) 程序编写规范，风格良好。几乎所有的程序都有紧扣主题、意义明确的注释，用来说明程序或者程序段的功能和变量在程序中的含义与作用，以便引导读者阅读程序，使读者养成良好的编程习惯。

本书第 1 章、第 4～9 章、第 11 章由李军编写，第 2 章由曹记东编写，第 3 章由林勇编写，第 10 章由魏佳编写，第 12 章由郭天印编写。全书由李军策划并最终统稿。李建忠教授审阅了本书，并提出了许多宝贵意见，在此表示衷心的感谢。

编者力图在书中充分反映程序设计思想方法与 C 语言知识，使得二者并重，为初学者提供具有一定特色的教材，但由于编者水平有限，书中难免存在不妥之处，欢迎广大读者多提宝贵意见。

编　者

2013 年 10 月

目　录

第 1 章　程序设计概论

程序是在解决问题时依据时间或空间顺序安排的工作步骤。例如，对会议议题的安排顺序称为会议程序，将基础建设、主体、屋面、装修、设备安装等施工步骤称为建筑施工程序。不同的领域都有该领域内的工作程序，在不同的工作中常常要进行工作步骤的编排，这种工作步骤的编排过程被称为程序设计。

自从电子计算机诞生以来，计算机科学得到了迅猛的发展，用计算机解决问题的应用领域越来越广泛，解决问题的规模越来越大。用计算机解决问题，需要事先确定问题的求解步骤，并将这些步骤用计算机指令或者计算机语言描述出来，其描述结果称为计算机程序；用计算机语言对所要解决问题中的数据以及处理问题的方法和步骤进行描述的过程称为程序设计。将设计好的程序交给计算机，计算机会按照程序中规定好的具体操作步骤对数据进行处理，直至得出最终结果，从而给出问题的答案。

程序设计要解决的核心任务包括：对给定问题进行有效的描述并给出问题的求解方法(计算机科学中称为**算法**)，正确地组织数据(计算机科学中称为**数据结构**)，运用程序设计语言进行编码、调试和测试等。

作为全书的导引，本章简要介绍程序设计要解决的这些核心问题的基本概念，以便读者在后续各章的学习中能够深入理解相关内容。

1.1　问题求解的思维过程

计算机求解问题的过程实际上是计算机模拟人类解决问题的过程，这一过程涉及人的一般思维活动、人对数据的组织及处理过程。因此，在介绍如何运用计算机求解问题之前，有必要了解人类是如何解决问题的，重点要了解解决问题的思维过程。

人们在认识客观世界时，其思维方式总是遵循着从特殊到一般的变化，从形象到抽象的跃升。例如，儿童能够计算出 3 + 5 = 8，是基于头脑中的 3 个苹果和 5 个苹果，或者 3 个糖果和 5 个糖果等实物形象地相加而得出结论的。这个过程是从一个个的特例经过抽象得出一般规律的思维创造，通常人们是经过这种思维方式直接得到规律的。这种思维方式对程序设计具有深远意义，它给出的是一类问题的通用解决办法。

又如，从 N(N 为任意整数)个数中找出最大数。这种找数的方法是经过了一个从特殊到一般的过程。首先来看特殊情况，假设有如下排放的 5 个正整数：

8，12，15，5，13

这是一组数量较少、数值确定的数。凭借已有的数学知识和概念，能够轻松地得出答案。即便是将这组数中的数据个数扩至 10 个，例如：

8，12，15，5，13，25，18，17，20，19

人们也能轻松地得出正确答案。但是要说出这个答案是如何得出的，或者说出同类问题是如何解决的，就不是一件简单的事情了。

为了回答如何获得这类问题的求解规律，我们假定仍然有 5 个数，它们分别是

a_1，a_2，a_3，a_4，a_5

这种表示形式是一个比上面的一组具体数值稍复杂的问题，但是，它却给出了求解这类问题的抽象表示。找出它的解决方法将会给出同类问题的一般求解方法，任何一组有 5 个具体数值的排列都是它的一个特例，因此可以用相同的方法加以解决。下面分三个阶段来获得这类问题的通用解法。

第一阶段：特例阶段。

假设 a_1、a_2、a_3、a_4、a_5 的值分别是 8、5、15、12、13 等，a 表示在问题求解过程中找到的最大数，则求解最大数的步骤如下：

第一步 检查第一个数 a_1 ($a_1=8$)，由于它是遇到的第一个数，是目前为止最大的数，因此令 $a=a_1$。

第二步 检查第二个数 a_2 ($a_2=5$)，由于 $a_2<a$，所以 a (a = 8)仍为最大数，不必改变 a 的值。

第三步 检查第三个数 a_3 ($a_3=15$)，由于 $a_3\geqslant a$，也就是说目前 a (a = 8)已不再是这组数中的最大数了，所以用 a_3 的值替换 a 的值，令 $a=a_3$。

第四步 检查第四个数 a_4 ($a_4=12$)，由于 $a_4<a$，所以 a (a = 15)仍为最大数，不必改变 a 的值。

第五步 检查第五个数 a_5 ($a_5=13$)，由于 $a_5<a$，即 a (a = 15)仍为最大数，不必改变 a 的值。

经过上述五步得出了最大数为 15 的结论。第一步把最大数 a 设为 a_1，第二步至第五步依次将 a_i ($i=2, 3, 4, 5$)与 a 比较，如果 $a_i\geqslant a$，则令 $a=a_i$。若再给出一组数值，仿照上述五步进行测试，看看能否得到正确答案？

第二阶段：细化处理。

为了能使上述方法求解同类问题中的任何问题，有两点需要注意。首先，第一阶段中第一步的动作和其他步骤的动作不一样，没有进行比较；其次，第二步至第五步的功能一样，但是描述语言却不一样。所以，需要对上述方法进行改进，使其描述具有一致性和精确性。我们称这一阶段为细化阶段。

在开始时，由于最大数 a 还没有初始化，所以用 a_1 的值初始化 a 的值，这便形成了第一步这个特殊步骤；后四步可用一种较为一致的语言“如果当前的数值大于最大数，那么就将当前数作为最大数”来描述。这样第一阶段中的五步就可以描述如下：

第一步 令最大数 a 等于 a_1。

第二步 如果 $a_2>a$，则令最大数 a 等于 a_2。

第三步 如果 $a_3>a$，则令最大数 a 等于 a_3。

第四步 如果 $a_4>a$，则令最大数 a 等于 a_4。

第五步 如果 $a_5>a$，则令最大数 a 等于 a_5。

上述五步便给出了从任意 5 个数中找出最大数的一般方法。现在需要将此问题泛化，假使要从 N(N 为任意整数)个数中找出最大数，按照本阶段的五步描述那样，一步步地罗列出 N 个步骤，从理论上来说是可以的，但是，当 N 的值特别大时，一步步地罗列是不现实的，所以需要将上面的问题泛化，形成一个具有广泛指导意义的方法。

第三阶段：泛化处理。

从第二阶段的后四步可以看出，每一步的处理方式都是相同的，若将第二步重复 4 次，也能解决问题。由此可以推广到对于 N 个数，除第一步外其余 N–1 步用同样的描述方法重复进行 N – 1 次，即可解决问题。所以，可以将这个问题的处理方法泛化如下：

第一步　将最大数 a 置为 a_1。

第二步　重复下述方法 N – 1 次：

如果 $a_i > a$，则将最大数 a 置为 a_i(其中 $i = 2, 3, \cdots, N - 1$)。

经过泛化处理后，就给出了解决从 N 个数中找出最大数问题的一种通用方法，并且能够很容易地将其转换成计算机程序。

在解决上述查找最大数问题的求解过程中有两个问题需要注意：第一个是数值的排列与存放问题；第二个是求解步骤问题。它们在程序设计中代表了同一事物既相互区别又相互联系的两个不同的侧面，即数据结构与算法设计问题。下面就这两个问题进行简要介绍。

1.2　算 法 基 础

1.1 节描述了在查找最大数问题时采用的一种通用的方法和步骤，通常将这种通用的方法和步骤称为**算法**。从本质上说，算法体现了人类解决某类问题时的思维过程，描述了人类解决同类问题所依据的规则。如果找到了解决某类问题的算法，人类在解决同类问题的具体实例时就可以根据算法对所给定问题进行相应的处理，最终得出答案。

1.2.1　算法的概念及其特征

算法是对特定问题的求解步骤的一种描述，能够对符合规范的输入在有限的时间内获得所要求的输出。在计算机程序设计中，算法的输入和输出都被编码成数字，因此算法对信息的处理实际上表现为对数据的某些运算。也可以说，算法描述了一个运算序列或数据处理过程。它强调问题求解的步骤和思想，而不是答案。例如，1.1 节中描述从 N(N 为任意整数)个数中找出最大数的三个阶段中的步骤和方法均为算法，只不过第三阶段进行泛化处理后形成的算法对同类问题具有普遍的适用性。

由上述对算法概念的描述可以看出算法具有如下五个特征：

(1) **有穷性**。算法必须在有限步之后结束，每步必须在有限的时间内完成。一个需要无限时间才能解决问题的方法等于没有解决问题。

(2) **确定性**。算法的每一步必须有确切的含义，进而整个算法的功能才能是确定的。一个没有确切含义的操作步骤，会给算法带来不确定性。没有确定性的算法无法解决问题。

(3) **可行性**。算法的所有操作都能够用已知的方法来实现。如果算法中含有不可实现的操作，则它无法求解问题。

(4) **输入**。一个算法必须要有输入，以刻画算法的初始状态。只有两种情况下算法没有输入：一是算法本身已经设置了初始条件；二是这个算法不能解决问题。对没有输入的算法一定要严格考察，判断其属于哪种情况。

(5) **输出**。一个算法必须要有一个或多个输出，以刻画算法进行问题求解的结果。一个没有输出的算法是没有意义的。

算法与程序既有联系又有区别，满足算法五个特征的程序肯定是算法，但程序并非全部满足算法的五个特征。例如，计算机的操作系统可以不停地运行，它总是逗留在一个永不终止的循环中，等待有新的作业输入，操作系统中这段循环程序就不满足算法的有穷性特征。

1.2.2 算法的基本结构

作为对特定问题求解步骤的描述，算法是由许多具体的操作构成的，虽然操作的内容千变万化，但是一些操作之间具有内在联系，这些联系控制着各操作步骤的执行顺序，使得按照书写顺序排列的操作步骤不一定在操作顺序上相邻。算法中各步骤的执行顺序问题称为算法的流程控制问题。算法的基本结构分为顺序结构、选择结构和循环结构三种。

1. 顺序结构

顺序结构就是算法中一组操作步骤的执行顺序是按照书写顺序依次进行的，并且每一步骤只执行一次。

2. 选择结构

选择结构也称为分支结构。选择结构往往由若干组操作组成，根据某个条件的成立与否，选择其中的一组执行，这样每组操作就形成了一个分支。选择结构中每次只有一个分支被执行，其余分支不会被执行。分支结构中的每个分支也可以是算法的三种基本结构中的任意一种，也就是说每个分支中除了有顺序结构外，还可以有进一步的分支结构，也可以有循环结构。

3. 循环结构

循环结构是指算法中的一组操作在一定条件下被反复多次执行。被反复执行的部分称为循环体。循环结构也需要判断条件，当条件满足时，算法进入循环体执行；当条件不满足时，循环结束，执行循环结构之后的其他步骤。循环结构中循环条件的设定非常重要，设置不当，循环永不结束，即出现死循环。与分支结构类似，循环体可以是算法的三种基本结构中的任意一种。当循环体只执行一次时，可以将循环结构简化成顺序结构。

另外，还有一种称为递归的结构，具体内容详见第 6 章。

1.2.3 算法的描述方法

在构思和设计了一个算法之后，必须清楚、准确地将所设计的求解步骤记录下来，即描述算法。算法与它的描述之间是有差别的，这就好像一个理论与刊载这个理论的书之间的差别一样，理论本质上是一种思想性的东西，而书则是这个理论的某种语言文字的载体，一本书可以翻译成另外的语言文字，如英文、俄文等，以不同的版面格式再版，这只不过是改变了理论的表达形式，而它的思想内容并未改变。

算法可以用不同的方法来描述。人类的自然语言(如中文、英文、俄文)是一种最直接的表示法，便于人们阅读。例如，1.1 节中查找最大数的算法就是用中文进行描述的。但是用自然语言描述算法不够严谨，因为有时用同样的文字描述一句话，不同的人会有不同的理解。也就是说，用自然语言描述算法容易引起歧义，会降低算法的确定性。算法也可用程序设计语言来描述，程序设计语言严谨、规范，不容易引起歧义，但是不便于人们阅读，并且掌握它还需要进行专业训练。因而，在计算机科学领域中，通常采用流程图和伪代码来描述算法。

1．流程图

流程图法是采用规格化的图形符号结合自然语言以及数学表达式进行算法描述的。其特点是简明、直观，便于理解，与程序设计语言无关，同时又很容易细化成具体的程序。流程图中一些常见的图框及流程线如图 1-1 所示。

图 1-1　流程图中常见的图框及流程线

起止框：表示程序的开始或结束。作为起始框时，它没有入口，只有一个出口；作为终止框时，它没有出口，只有一个入口。

数据框：表示数据的输入或输出。它有一个入口和一个出口。

处理框：表示数据运算及其处理的图框。它有一个入口和一个出口。

判断框：对给定的条件进行判断，根据条件成立与否决定如何执行程序的后续操作。它有一个入口和两个出口(根据判断条件成立与否选择其中一个)。

流程线：表示操作流程的去向，一般用带箭头的线段或者折线来表示。

注释框：是为了对算法的某些地方作必要说明而引进的，以帮助程序设计人员阅读算法或使程序设计人员更好地理解流程图的作用。它是流程图中的可选元素，并非必备元素。

图 1-2 所示为 1.1 节中查找最大数的算法流程图。

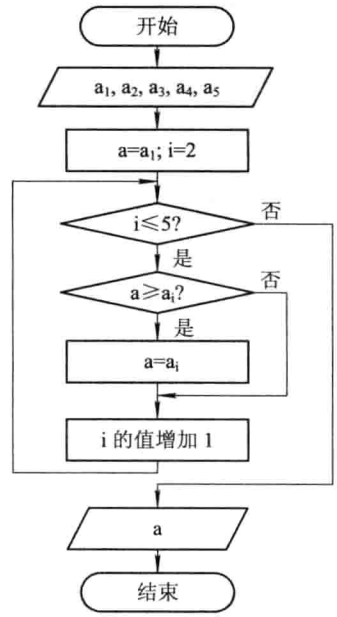

图 1-2　查找最大数的算法流程图

2．伪代码

伪代码法是在程序设计语言的基础上，简化并放宽其严格的语法规则，保留其主要的逻辑表达结构，并结合自然语言和一些数学的表达方式，形成的类似于程序设计语言的一种描述方式。采用这种方式描述的算法很容易细化为具体的程序。最具影响力的伪代码是类 Pascal 语言和类 C 语言描述法，它们看上去与 Pascal 语言或者 C 语言程序非常接近。伪

代码比流程图更接近程序。

计算机科学领域对伪代码没有形成共识，只是要求以懂得程序设计语言知识的人都能很好地理解为原则，因而伪代码法没有一个统一的标准。本书中的程序设计是采用 C 语言来完成的，因此，本书中的伪代码采用的是 C 语言的流程控制语句、赋值语句并结合自然语言与数学语言的一种综合描述方法。下面将本书中用到的伪代码的一些具体事项做一约定。

(1) 条件选择结构采用 if-else 描述，循环结构采用 for、while、do-while 描述。

(2) 表达式赋值采用 C 语言的赋值号“=”，如多重赋值“a=b=c=e”是将表达式 e 的值赋给 a、b、c 三个变量。

(3) 比较运算采用以下符号：“<”表示小于，“≤”表示小于等于，“>”表示大于，“≥”表示大于等于，“==”表示等于，“≠”表示不等于。

(4) 逻辑运算采用以下符号：“AND”表示与运算，“OR”表示或运算，“NOT”表示否定。

(5) 有些无法形式化的描述用文字来表述。

下面是 1.1 节中查找最大数算法按照上述约定的伪代码表示。

```
给 a1,a2,a3,a4,a5 赋值
a=a1;i=2;
while(i≤5)
{
    if(ai≥a) a=ai;
    i 的值增加 1;
}
输出最大数 a;
```

无论采用何种方法来描述一个算法，一定要详略得当。一个算法要描述到什么样的细致程度，取决于交流算法的对象。如果是和一个计算机专家交流算法，只要能表达“找出五个数中最大数”这样的一个笼统而抽象的概念即可达到目的；如果是和一个初学者交流或者为了在计算机上执行算法，那么就要给出“找出五个数中最大数”的算法细节。

1.3 数据结构基础

数据是对客观事物的符号表示，在计算机科学中是指所有能输入到计算机中并被计算、加工和处理的对象。数据的类型既可以是数值类型的，也可以是非数值类型的；可以是单一类型的，也可以是复合类型的。数值类型可以是整数、实数等类型；非数值类型包括字符类型、文本型、图像类型、音频以及视频类型等。

数据元素是数据的基本单位，例如，在进行一个圆的计算时，半径、周长、面积等都是简单的数据对象，每个对象只需要一个数据元素，即实数来表示。在计算机程序中有些数据元素比较复杂，它具有底层结构，即每个元素由一个或多个数据项构成。**数据项**是具有独立含义的最小单位的数据。虽然有些数据元素具有底层结构，但是使用时将它看作一

个整体结构。例如，复数是由实部与虚部两个数据项构成的一个整体结构。

在进行程序设计时，经常会处理一批具有相同性质的数据元素，这种由一个或多个数据元素组成的复杂数据通常被称为**数据对象**。数据对象是一个具有底层结构的实体，常常被作为一个整体来引用。例如，一个 n × n 的实数矩阵就是一个数据对象，这个数据对象由 n^2 个实数类型的数据元素构成；再如，一个 n × n 的复数型矩阵也是一个数据对象，这个数据对象由 n^2 个复数类型的数据元素组成，而每个复数类型的数据元素又能分成实部和虚部两个数据项。

在任何情况下，描述事物对象的数据元素之间彼此不会是孤立的，它们之间总是存在着这样或那样的关系，这种元素之间的关系称为数据对象的**数据结构**。按照数据元素间关系的不同特征，通常有下列四种基本数据结构，如图 1-3 所示。

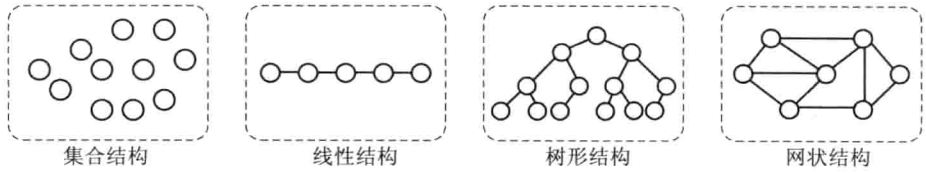

图 1-3　四种基本数据结构

(1) **集合结构**：构成数据对象的数据元素之间除了属于“同一集合”之外再没有其他关系。集合是元素关系极为松散的一种结构。

(2) **线性结构**：构成数据对象的数据元素之间存在一对一的关系。例如，数列、矩阵、表格等都是线性结构。

(3) **树形结构**：构成数据对象的数据元素之间存在一对多的关系。例如，家族的谱系图、一个单位的组织结构等都是树形结构。

(4) **网状结构**：构成数据对象的数据元素之间存在多对多的关系，这种结构也称为图结构。例如，一个国家的城市与城市之间的道路交通网络、一个人的社会交往关系等都属于网状结构。

按照上述介绍，我们可以看出，一个数据对象的结构具有两个要素：一个是数据元素的集合；另一个就是数据元素之间的关系。集合结构是一种松散的结构，可以人为地给其数据元素之间加上一种关系，这样即可用其他三种结构来表示这种结构。通常最简单的做法是采用线性结构来描述集合结构。因此，数据结构可以粗略地分为**线性结构**与**非线性结构**。

上述介绍的数据结构是对数据对象的一种逻辑描述，描述的是数据元素之间的逻辑关系，因此也称为**逻辑结构**。数据的逻辑结构是从实际问题中抽象出来的数学模型，并非数据在计算机中的实际表示。数据在计算机中的表示与存储称为数据的**存储结构**，也称**物理结构**。存储结构既要表示数据元素，又要表示元素之间的关系。由于数据的逻辑结构有线性和非线性之分，而计算机的存储空间又都是线性的，要在线性的存储空间中表示数据的多种逻辑结构，就必须进行一系列的处理，这样既能保证数据在逻辑上的正确性，又能保证在计算机设备中的可实现性。

在计算机中存储与表示数据的逻辑结构有两种方式：顺序存储方式和链式存储方式。用这两种存储方式表示的数据的逻辑结构分别称为**顺序存储结构**和**链式存储结构**。

顺序存储结构是把逻辑上相邻的数据元素存储在物理位置相邻的存储单元中。顺序存储结构采用物理存储空间中位置自然相邻的关系来表示数据元素的逻辑关系，如图 1-4 所示。顺序存储结构通常借助于程序设计语言中的数组来实现。

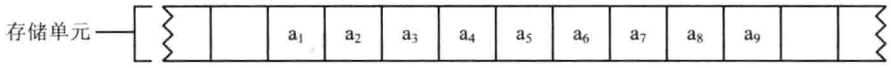

图 1-4　数据的顺序存储结构

链式存储结构对逻辑上相邻的元素并不要求其物理位置相邻，数据元素间的逻辑关系通过附设一个指针来指示，用指针指出与该数据元素有关的其他数据元素存放在何处。由于链式存储结构需要用指针来指示数据元素之间的关系，同顺序存储结构相比，它需要额外的存储空间来存储指针。链式存储结构通常借助于程序设计语言中的指针类型来实现，如图 1-5 所示。链式存储结构既能表示线性结构又能表示非线性结构。

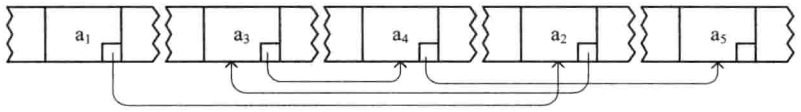

图 1-5　数据的链式存储结构

以上介绍的是最基本的数据结构，另外还有索引存储法以及散列法等。数据结构用来反映一个数据对象的内部结构，亦即一个数据对象由哪些元素构成，以什么方式构成。逻辑结构反映数据元素之间的逻辑关系，而物理结构反映数据元素在计算机内部的安排形式，这些都是程序设计需要解决的核心问题。

算法与数据结构是紧密相连的，在进行算法设计时必须确定相应的数据结构，数据结构选择得是否恰当，直接影响算法的效率。

1.4　程序设计语言概述

使用计算机解题，就是按照算法对数据进行处理，解决问题的算法必须以计算机能够读懂的形式表示出来，这就需要用计算机语言将算法描述出来。计算机语言(Computer Language)是指人与计算机之间通信的语言。计算机语言是人与计算机之间传递信息的媒介。计算机系统的最大特征是将指令通过一种语言传达给机器。为了使电子计算机能够进行各种工作，就需要有一套用以编写计算机程序的字符和语法规则，这些字符和语法规则组成了计算机语言的各种语句。本节主要介绍程序设计语言的发展历史、程序设计范型、过程型程序设计语言的语法元素及其基本功能。

1.4.1　程序设计语言的发展历史

1. 面向机器的程序设计语言

现代计算机系统包括硬件系统和软件系统。硬件系统是由运算器、控制器、存储器、输入/输出设备组成的。其中，运算器和控制器统称为中央处理器(CPU)，是计算机的核心。软件系统包括计算机运行所需的各种程序及其相关文档资料。