21世纪普通高等教育规划教材

# 计算机程序设计基础

熊壮 主编

配有电子教案

机械工业出版社
CHINA MACHINE PRESS

21 世纪普通高等教育规划教材

# 计算机程序设计基础

主编　熊　壮
参编　刘慧君　伍　星
主审　朱庆生

机械工业出版社

计算机程序设计基础是计算机专业技术基础系列课程中的重要组成部分。全书共分10章：基本概念；程序设计中的数据基础；结构化程序基础；模块化程序设计基础；程序设计中数据对象地址的处理方法；构造类型数据处理基础；字符串数据处理基础；二进制位数据的处理基础；文件数据处理基础；程序设计的深入话题。每章后配有习题与思考题，便于读者学习和理解。

本书从结构化程序设计技术出发，以C程序设计语言为载体，通过本书的学习，读者可以了解计算机程序设计所需要的基本知识，掌握计算机结构化程序设计的基本概念、基本技术和方法。

本书可供高等院校计算机专业作为计算机程序设计基础、计算机程序设计技术或计算机程序设计语言等计算机技术基础课程教材，也可供非计算机专业本、专科学生以及计算机应用开发人员在学习程序设计语言和程序设计技术时作为参考。

**图书在版编目（CIP）数据**

计算机程序设计基础/熊壮主编. —北京：机械工业出版社，2005.9
21世纪普通高等教育规划教材
ISBN 7-111-17447-X

Ⅰ.计… Ⅱ.熊… Ⅲ.程序设计-高等学校-教材
Ⅳ.TP311

中国版本图书馆CIP数据核字（2005）第109610号

机械工业出版社（北京市百万庄大街22号 邮政编码100037）
策划编辑：刘丽敏
责任编辑：刘丽敏 版式设计：霍永明 责任校对：吴美英
封面设计：马精明 责任印制：洪汉军
北京京丰印刷厂印刷
2005年10月第1版·第1次印刷
787mm×1092mm $^{1}/_{16}$·13.75印张·335千字
定价：20.00元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
本社购书热线电话（010）68326294
封面无防伪标均为盗版

# 前　言

计算机程序设计基础课程是计算机专业技术基础系列课程中的重要组成部分，引导学生进入计算机程序设计的广阔空间，培养学生的逻辑思维能力、抽象能力和基本的计算机程序设计能力是计算机程序设计基础课程的主要任务。本书从结构化程序设计技术出发，以C程序设计语言为载体，通过对典型问题的分析、算法描述以及相应C语言代码描述展现了在程序设计过程中如何对问题进行分析，如何组织数据和如何描述解决问题的方法，展现了在计算机应用过程中如何将方法和编码相联系的具体程序设计过程，进而向读者介绍计算机结构化程序设计的基本概念、基本技术和方法。

计算机程序设计主要任务之一就是将需要用计算机处理的实际问题抽象为数学模型，并设计出解决这个问题所需要的方法和步骤即算法。在处理实际问题的程序中应该包含两个方面的内容：要处理的对象和对这些对象的处理方法。对于那些要处理的对象，用数据和数据之间的关系即数据结构来表示，而对对象的处理方法用算法来表示。本书中用了较多实例向读者介绍了算法的基本概念和算法的特征，以及算法的描述方法。

计算机程序设计的另外一个主要任务就是用合适的计算机程序设计语言对所设计的算法进行编码处理，即编制程序。C语言功能丰富、表达能力强、程序执行效率高、可移植性好；C语言既有高级计算机程序设计语言的特点，同时又具有部分汇编语言的特点，因而C语言具有较强的系统处理能力；C语言是一种结构化程序设计语言，支持自顶向下、逐步求精的程序设计技术，通过C语言函数结构可以方便地实现程序的模块化；在C语言的基础上发展起来的面向对象程序设计语言，如C++、Java、C#等与C语言有许多的共同特征，掌握C语言对学习进而应用这些面向对象的程序设计语言有极大的帮助。综上所述，C语言应该是学习计算机结构化程序设计技术的首选语言。

本书主要内容分为三个大部分：第一部分（第1~2章）主要讨论了进位计数制、数制之间的转换、数在计算机中的表示方法、算法描述的方法及若干算法描述实例；第二部分（第3~9章）主要讨论了结构化程序设计的基本思想和方法，并以C程序设计语言作为载体讨论了程序设计所涉及的主要技术，如数据的基本类型、程序控制结构、数组、指针、字符串、标准库、模块化程序设计技术及C程序设计语言对模块化技术的支持和实现等问题，以帮助读者尽快提高使用程序设计技术解决实际问题的能力；第三部分第10章主要讨论了C程序设计语言特有的一些知识，如命令行参数、预处理器等，同时还讨论了在程

序设计中构造复杂数据类型的方法及其应用问题。

本书每章均提供了与内容紧密相关的习题以帮助读者巩固所学知识，同时还提供了常用C标准库函数、ASCII码对照表等常用教辅资料。本书所有例题和习题解答可以在机械工业出版社网站上下载。

本书可供高等院校计算机专业作为计算机程序设计基础、计算机程序设计技术或计算机程序设计语言等计算机技术基础课程教材，也可供非计算机专业本专科学生以及计算机应用开发人员在学习程序设计语言和程序设计技术时作为参考。

本书由熊壮主编，朱庆生主审。各章节编写分工如下：熊壮（第1章、第4章、第7章、第10章），刘慧君（第2章、第5章、第9章），伍星（第3章、第6章、第8章），全书由熊壮进行内容调整、修改，统一定稿。

限于编者水平，书中错误和不妥之处在所难免，恳请读者不吝指教。

联系地址：重庆，重庆大学计算机学院。

E-Mail：xiongz@cqu.edu.cn，xiongz@cqcnc.com

编 者

2005年9月

# 目　录

# 第 1 章　程序设计的基本概念

## 1.1　程序设计语言

### 1.1.1　程序设计语言概述

计算机应用本质上就是以计算机作为工具解决人类社会中的实际问题，其中一个关键的步骤就是让计算机能够理解、执行人类解决某种问题的方法，从而达到解决实际应用问题的目的。为了能够使计算机理解人的意图，就必须解决人类和计算机相互交流的问题，将人解决问题的思路、方法和手段通过某种计算机能够理解的形式告诉计算机，使得计算机能够根据人的指令去一步一步地工作进而完成某种特定的任务。这种人和计算机之间交流的语言就称为计算机程序设计语言。从计算机发明至今，随着计算机硬件技术和软件技术的发展，计算机程序设计语言也经历了机器语言、汇编语言、面向过程的程序设计语言以及面向对象的程序设计语言等阶段。

**1. 机器语言**

在计算机系统中，一条机器指令规定了计算机系统硬件的一个特定动作。一个系列的计算机在硬件设计制造时就用若干指令规定了该系列计算机能够进行的基本操作，这些指令集合在一起构成了该系列计算机的指令系统。在计算机应用的初期，程序员使用机器的指令系统来编写计算机应用程序，这种程序称为机器语言程序。编写机器语言程序要求程序员对计算机系统的所有细节如指令系统、存储容量等都要十分熟悉，这使得计算机程序设计成为了一种非常艰苦的工作。使用计算机机器语言编制的程序，由于其每条指令都对应计算机一个特定的基本动作，所以程序占用内存少、执行效率高。但其缺点也十分明显，例如：编制程序的工作量大、容易出错、依赖具体的计算机系列，从而程序的通用性、移植性都很差。

**2. 汇编语言**

为了解决使用机器语言编写应用程序所带来的一系列问题，人们首先想到了使用助记符号来代替不容易记忆的机器指令，用助记符号来表示计算机指令的语言称为符号语言，亦称为汇编语言。在汇编语言中，每一条用符号来表示的汇编指令与计算机机器指令都一一对应，但记忆难度大大减小了，与机器语言相比，不仅易于检查和修改程序错误，而且指令、数据的存放位置可以由计算机系统自动分配。用汇编语言编写的程序称为源程序，计算机系统不能直接识别和处理源程序，必须通过某种方法将它翻译成计算机系统能够理解并执行的机器语言，执行这个翻译工作的程序称为汇编程序。

使用汇编语言编写计算机程序，程序员仍然需要十分熟悉计算机系统的硬件结构，所以从程序设计本身上来看仍然是繁琐的、低效率的。但正是由于汇编语言与计算机硬件系统关系密切，迄今为止在某些特定的场合如对时空效率要求很高的系统核心程序以及实时控制程

序等，汇编语言仍然是十分有效的程序设计工具。

**3. 高级语言**

计算机程序设计高级语言是一类接近于人类自然语言和数学语言的程序设计语言的统称。高级语言按照其程序设计的出发点和方式不同分为了面向过程的语言和面向对象的语言。Fortran 语言、C 语言等都是面向过程的语言，到目前为止还具有强大的生命力，是设计中小型计算机程序的优秀程序设计语言，同时也是面向对象程序设计语言的基础；而以 C++、Smalltalk 等为代表的语言与面向过程语言有着许多不同，这些语言支持“程序是相互联系的离散对象集合”这么一种新的程序设计思维方式，主要具有封装性、继承性和多态性等特征。

高级程序设计语言按照一定的语法规则，由表达各种意义的运算对象和运算方法构成。使用高级程序设计语言编写计算机程序的优点是：编制程序相对简单、直观、易理解、不容易出错。高级程序设计语言是独立于计算机系统的，因而用高级程序设计语言编写的计算机程序具有良好的通用性和可移植性。

用高级程序设计语言编写的程序也称为源程序，计算机系统不能直接理解和执行，必须通过一个语言处理系统将用高级程序设计语言编写的源程序转换为计算机系统能够认识、理解的目标程序才能被计算机系统执行。

## 1.1.2 语言处理程序概述

在程序设计过程中，除了使用机器语言之外，无论使用哪种计算机高级程序设计语言编写的计算机程序都称为源程序，它们都不能为计算机系统直接识别、理解和执行，都必须通过某种方式转换为计算机能够直接执行的机器语言程序。这种从用高级程序设计语言编写的源程序转换到机器目标程序的方式有两种：解释方式和编译方式。

解释方式下，计算机系统对用高级程序设计语言书写的源程序一边解释一边执行，不形成对应的目标文件和执行文件。

编译方式下，首先通过与所使用的程序设计语言相对应的编译程序对源程序进行处理，经过对源程序的词法分析、语法分析、语意分析、代码生成和代码优化等阶段将所处理的源程序转换为用二进制代码表示的目标程序。然后通过连接程序处理，将程序中所用的函数调用、系统功能调用等嵌入到目标程序中，构成一个可以连续执行的二进制执行文件。调用这个执行文件就可以实现程序员在源程序文件中所指定的相应功能。用编译方式处理高级程序设计语言源程序的简单过程如图 1-1 所示。

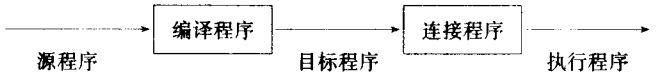


图 1-1 高级语言源程序处理过程

## 1.1.3 程序设计方法概述

随着计算机技术的发展和计算机技术应用领域的不断扩展，计算机程序设计方法也得到了迅速的发展，出现了结构化程序设计、面向对象程序设计以及组件程序设计等计算机程序设计方法。在这些程序设计方法中，结构化程序设计方法占有十分重要的地位，了解和掌握结构化程序设计的概念和方法将为学习和掌握其他程序设计方法打下良好的基础。

**1. 结构化程序设计方法**

结构化程序设计方法的基本思想之一就是抽象与分解，抽象实际上是人类认识世界的基本法则之一。人们在实践中认识到，现实世界中一定事物、状态、或过程之间总是存在着某些相似的方面，把这些相似的方面集中和概括起来，暂时忽略它们之间的差异，或者说抽出事物的本质特性而暂时不考虑它们的细节，这就是抽象。显然，这种抽象包含了系统的观点和分层的观点，即可以把程序设计所面临的问题看成是一个系统，这个系统可用高级的抽象概念来理解和构造，这些高级的抽象概念又可用较低级的抽象概念来理解和构造，如此进行下去，直到最低层次的模块可以表示成某种程序设计语言的语句为止。对系统的每一次抽象都是向更具体的方面推进的过程，也是进一步分解的过程。因此，在抽象的过程中，同时伴随着分解。当我们遇到一个较大的、较复杂的问题时，不是急于编写程序代码，而应该首先把问题自顶向下、逐步分解、细化成一个个程序模块，然后再对模块进一步细化。处于不同层次的模块应该只考虑自己的问题而不必考虑其他模块内部的问题，顶层的模块控制了系统的主要功能并影响全局；底层的模块则完成对数据的具体处理。这种自顶向下、模块化的、从抽象到具体的方式进一步表现了结构化程序设计方法的基本思想。

信息隐蔽和局部化是结构化程序设计方法的另一基本思想。这一思想是指：在模块化的过程中应该这样设计和确定模块，使得一个模块内包含的信息对于不需要这些信息的模块来说是不能访问的。这也告诉我们，模块的划分应该遵循模块独立性原则，即模块彼此间互相依赖的程度要小，而模块内部各个元素彼此结合的程度要大。

结构化程序设计方法的基本思想还包括了尽量使用三种基本结构、保持单入口和单出口形式、限制使用 GOTO 语句等，使得程序容易理解、容易维护和容易验证其正确性。

**2. 面向对象程序设计方法**

面向对象技术是程序设计方法的一次革命，在计算机软件开发史上具有里程碑的意义。面向对象程序设计方法的基本思想就是尽可能按照人类认识世界的方法和思维方式来分析问题和解决问题。现实世界是由许许多多的实体或对象构成的，这些对象彼此相关并能相互通讯，对象的活动构成了现实世界的活动。从面向对象的角度看，程序是对象的集合，对象之间的相互作用构成了一个软件系统。对象参与的交互动作称为事件，在事件中消息在对象之间发送，接收消息的对象调用相应的方法进行响应。如图 1-2 所示。

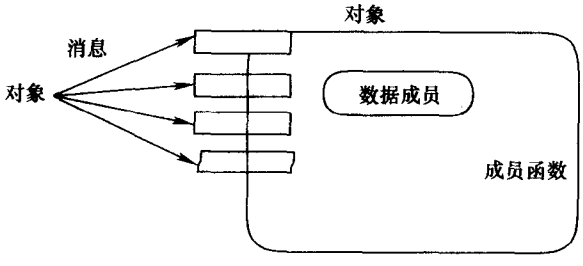


图 1-2　面向对象的程序结构

面向对象的程序设计最突出的特点是建立在对象和类的基础上，把问题所对应的现实世界中的事物抽象成对象或类，并建立对象之间的关系。每个对象或类不仅包含描述其特征的属性或数据结构，而且还包含对这些数据结构的操作（也称为方法或服务），如图 1-3 所示。对对象的操作可以是以某种方式处理数据，或者是执行某个计算，或者是监督一个对象控制事件的发生。通过对象关系可以明确对象之间的消息（消息也可理解为事件），通过定义每个对象所能够接收的消息以及对象接收消息时执行的相应操作可以建立对象间的接口，同时还可以描述对象的私有部分即对象的内部属性或数据结构以及操作的过程细节。当操作比较复杂时，仍然

可以使用传统的结构化设计技术将其模块化。

面向对象的程序中，操作始终是对类或对象中的属性进行。因此，能够反映对象属性的数据结构将决定在这些操作中所采用的算法。当我们对对象中的每一个操作进行描述时，可以把一个操作理解为一个“过程”，这个“过程”需要完成一定的任务，因此需要进行算法设计，面向对象程序设计中的算法设计与结构化程序设计中算法的设计类似。值得注意的是，面向对象程序设计中的“过程”与结构化程序设计中的过程本质上是两个不同的概念。结构化程序设计中，当过程对数据进行操作时数据是被动的；而在面向对象程序中，对象是主动的，对象中的“过程”即操作在接收到消息时进行启动并执行，以实现对对象中的属性进行改变的目的。例如，对于“盘子摔碎的问题”用结构化程序设计的思想可以描述为：“我将盘子从桌子上弄掉到地板上，盘子就被打碎了”；而用面向对象程序设计的思想则应描述为：“我推了一下盘子，盘子就在桌子上移动，盘子从桌子上落到了地板上，盘子自己就碎了”。在这里，“盘子（包含有属性）”是一个对象，除了收到“我推了一下盘子”这个消息（事件）之外，其余的动作全部是“盘子”的主动操作。

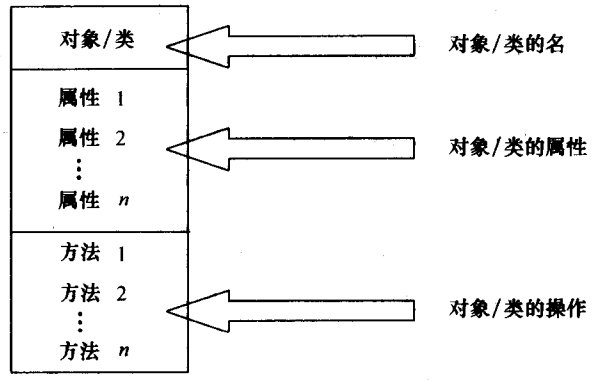


图 1-3 类与对象的示意图

从已有的对象类型出发建立一种新的对象类型，使新的对象类型继承原对象类型的特点和功能，这种思想是面向对象设计方法对软件开发技术的主要贡献。继承是对许多问题中分层特性的一种自然描述，继承就是创建一个具有其他类的属性和行为的新类。考虑如下“猫”和“白猫”的例子，可以用如图 1-4a 所示的方式来描述“猫”，在结构化程序设计中描述“白猫”的方法是将“猫”的所有属性和特征重新描述一遍，然后指出它的毛是白色的，如图 1-4b 所示；而在面向对象的程序设计中对“白猫”描述只需要表述出“毛是白色”的“猫”即可，如图 1-4c 所示。显然面向对象程序设计的描述方法既简练又能够表述出“白猫”和“猫”的内在联系，“白猫”具有“猫”的一切特征。从程序设计的角度上讨论这个问题，可以认为“白猫”这种数据类型从“猫”这种数据类型衍生而来，从而继承了“猫”数据类型的所有属性和行为。在面向对象的程序设计方法中，正是通过对类的继承实现程序代码的重用。

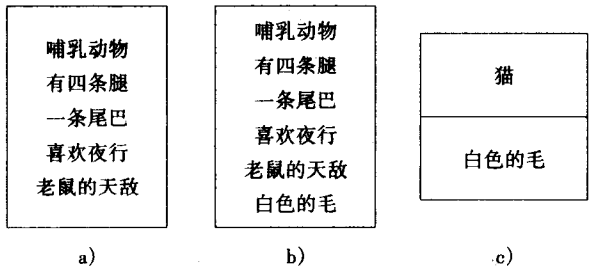


图 1-4 “白猫”的不同描述方法

综上所述，面向对象的程序设计方法可以简单地表示为：“面向对象程序设计方法 = 对象 + 类 + 继承 + 消息通信”，面向对象的程序设计方法既使用对象又使用类和继承机制，而且对象之间仅能通过消息实现彼此之间的通信。

**3. 组件（COM）程序设计**

组件程序设计方法继续并发展了面向对象的程序设计方法。它把对象技术应用于系统设计，对面向对象的程序设计的实现过程作了进一步的抽象。我们可以把组件化程序设计方法用作构造系统的体系结构层次的方法，并且可以使用面向对象的方法很方便地实现组件。在结构化程序设计方法和面向对象的程序设计方法中都只能在源程序级别对程序代码进行重用，而不能在二进制级别（可执行代码级）重用，组件程序设计方法正是从这个层面解决问题的。

## 1.2　计算机中数据表示方法

通用的计算机系统只能直接识别和处理二进制数据信息，所以在计算机系统内部存储和处理的数据都是二进制数据。但在计算机程序设计特别是在使用高级程序设计语言进行程序设计的过程中，为适应人类的自然需求习惯上通常使用十进制形式的数据，在特殊的情况下也采用八进制或十六进制数据。因此，有必要了解各种进制数据的表示形式、各种进制数据之间的转换方法以及数据在计算机系统内部的存储方法。

### 1.2.1　进位计数制和数制之间的转换

**1. 进位计数制的基本概念**

在计算机中常用到的数制有十进制、二进制、八进制和十六进制。任意的 $R$ 进制数据使用的数码为 $R$ 个，它们是 0，1，2，…，$R-1$，逢 $R$ 进位，用公式可以表示为

$$(N)_R = \left( \sum_{i=-m}^{n-1} k_i R^i \right)_R \qquad (0 \leqslant k_i \leqslant R-1)$$

式中，$n$ 表示整数位数；$m$ 表示小数位数；$k_i$ 表示 $R$ 个数码中的任意一个。

当基数 $R=10$ 时表示十进制数。使用的数码为 0，1，2，…，9，逢 10 进位，十进制数据可以用公式表示为

$$(N)_{10} = \sum_{i=-m}^{n-1} k_i \cdot 10^i \qquad (0 \leqslant k_i \leqslant 9)$$

例如，$135.25 = 1\times10^2 + 3\times10^1 + 5\times10^0 + 2\times10^{-1} + 5\times10^{-2}$

当基数 $R=2$ 时表示二进制数。使用的数码为 0 和 1 两个，逢 2 进位，二进制数据可以用公式表示为

$$(N)_2 = \sum_{i=-m}^{n-1} k_i \cdot 2^i \qquad (k_i = 0,1)$$

例如，$11011.11 = 1\times2^4 + 1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0 + 1\times2^{-1} + 1\times2^{-2}$

同样，当基数 $R=8$ 时表示八进制数。使用的数码为 0，1，2，…，7，逢 8 进位；当基数 $R=16$，时表示十六进制数。使用的数码为 0，1，2，…，9，a，…，f，逢 16 进位；读者可以参考其他专业书籍，此处不在赘述。

**2. 数制之间的转换**

（1）十进制整数转换为 $R$ 进制数　十进制整数转换为 $R$ 进制时采用“除 $R$ 取余法”。

“除 $R$ 取余法”的基本方法是用被转换的数除以 $R$，所得到的余数为取出的 $R$ 进制数码；然后用上次得到的商除以 $R$，得到的余数为取出的 $R$ 进制数码；直到数据全部转换完为止。最先取出的是 $R$ 进制数据的最低位，最后取出的是 $R$ 进制数据的最高位。

**例 1-1**　将十进制数据123转换为八进制数。

| | | | |
|---|---|---|---|
| 8 | 123 | （余数） | |
| 8 | 15 | …3 | （转换后的最低位） |
| 8 | 1 | …7 | |
| | 0 | …1 | （转换后的最高位） |

即 $(123)_{10}=(173)_8$

**例 1-2**　将十进制数据123转换为二进制数。

| | | | |
|---|---|---|---|
| 2 | 123 | （余数） | |
| 2 | 61 | …1 | （转换后的最低位） |
| 2 | 30 | …1 | |
| 2 | 15 | …0 | |
| 2 | 7 | …1 | |
| 2 | 3 | …1 | |
| 2 | 1 | …1 | |
| | 0 | …1 | （转换后的最高位） |

即 $(123)_{10}=(1111011)_2$

（2）十进制小数转换为 $R$ 进制数　十进制小数转换为 $R$ 进制时采用“乘 $R$ 取整法”。“乘 $R$ 取整法”的基本方法是用被转换的十进制小数乘以 $R$，所得到的整数部分为 $R$ 进制数的数码；然后再用上次得到的小数部分乘以 $R$，得到的整数部分为取出的 $R$ 进制数码；直到数据转换完成或达到所需的精度要求为止。最先取出的是 $R$ 进制小数的最高位，最后取出的是 $R$ 进制小数的最低位。

**例 1-3**　将十进制小数0.63转换为八进制(取 4 位小数)。

| | （整数部分） | |
|---|---|---|
| $0.63\times8=5.04$ | 5 | （转换后的小数最高位） |
| $0.04\times8=0.32$ | 0 | |
| $0.32\times8=2.56$ | 2 | |
| $0.56\times8=4.48$ | 4 | （转换后的小数最低位） |

即 $(0.63)_{10}\approx(0.5024)_8$

**例 1-4**　将十进制小数0.63转换为二进制（取 4 位小数）。

| | （整数部分） | |
|---|---|---|
| $0.63\times2=1.26$ | 1 | （转换后的小数最高位） |
| $0.26\times2=0.52$ | 0 | |
| $0.52\times2=1.04$ | 1 | |

$0.04 \times 2 = 0.08$　　0　　（转换后的小数最低位）

即$(0.63)_{10} \approx (0.1010)_2$

当需要将一般的十进制实数转换为 $R$ 进制实数时，只需将整数部分和小数部分分别转换后拼接在一起即可。例如，$(123.63)_{10} = (1111011.1010)_2$。

（3）$R$ 进制数转换为十进制数　对于任意的 $R$ 进制数据转换为十进制数据，最简单的方法是“按权相加法”。“按权相加法”的基本方法是将被转换的数据按权展开为多项式，然后将展开的多项式按十进制计算求和即可。

**例 1-5**　将二进制数据$(1111011.1010)_2$转换为十进制。

$$\begin{aligned}(1111011.1010)_2 &= (1\times2^6+1\times2^5+1\times2^4+1\times2^3+0\times2^2+1\times2^1+1\times2^0+1\times2^{-1}+\\ &\quad 0\times2^{-2}+1\times2^{-3}+0\times2^{-4})_{10}\\ &= (64+32+16+8+0+2+1+0.5+0+0.125+0)_{10}\\ &= (123.62)_{10}\end{aligned}$$

**例 1-6**　将八进制数据$(173)_8$转换为十进制。

$$(173)_8 = (1\times8^2+7\times8^1+3\times8^0)_{10} = (64+56+3)_{10} = (123)_{10}$$

## 1.2.2　数在计算机中的表示方法

一个带符号的二进制数据表示称为该数据的真值，在计算机系统中数据的符号也需要用数码来表示。根据不同的需要一个数可以用原码、反码或补码来表示。

**1. 数的原码表示**

二进制数据的原码表示是将该二进制数据真值的符号用数码来表示，符号位为 0 表示正数，符号位为 1 表示负数，$X$ 的原码用标记 $[X]_{原}$ 来表示。例如，以 8 位二进制数据为例

$[+123]_{原} = 01111011$

$[-123]_{原} = 11111011$

特别需要注意的是，在数的原码表示方法中，0 有两种表示形式，即

$[+0]_{原} = 00000000$　　$[-0]_{原} = 10000000$

数据的原码表示方法简单，与数据的真值转换方便。但在进行两个正数相减或两个不同符号数相加的运算时需要比较两个数据的绝对值才能决定被减数、减数和结果的符号。

**2. 数的反码表示**

二进制数据 $X$ 的反码用标记 $[X]_{反}$ 来表示，数据反码生成的规则是：正数的反码与其原码相同；负数的反码是在其原码的基础上符号位保持不变，数值位按位取反（0 变成 1，1 变成 0）。例如，以 8 位二进制数据为例

$[+123]_{反} = 01111011$

$[-123]_{反} = 10000100$

特别需要注意的是，在数的反码表示方法中，0 也有两种表示形式，即

$[+0]_{反} = 00000000$　　$[-0]_{反} = 11111111$

**3. 数的补码表示**

二进制数据 $X$ 的补码用标记 $[X]_{补}$ 来表示，数据补码生成的规则是：正数的补码与其原码相同；负数的补码是在其原码的基础上先求其反码，然后在反码的最低位加 1。例如，

以 8 位二进制数据为例

$[+123]_{补} = 01111011$

$[-123]_{补} = 10000101$

在数据的补码表示中，0 的表示方法是惟一的，即

$[+0]_{补} = [-0]_{补} = 00000000$

二进制数据用补码表示时运算简单，符号位可以作为数据的一位参与运算而不必单独处理，二进制数据的减法运算可以用其补码的加法实现，因而在通用计算机系统中，数据都是用二进制补码来存储和处理的。

为了能够得到用补码表示的二进制负数的真值，可以通过对二进制数据的补码再求补得到该二进制负数的原码来实现，即 $[[X]_{补}]_{补} = [X]_{原}$。例如

$$[10000101]_{补} = 11111011 = [-123]_{原}$$

**4. 字符数据的编码**

计算机系统除了能够对数值数据进行处理外，还能够处理非数值信息（例如符号、文字、图形、图像等）。由于计算机系统内部只能处理二进制编码，因此必须将非数值信息用二进制编码来表示。对于字符数据，最常用的编码方式是 ASCII 码，即美国国家标准信息交换码（American Standard Code for Information Interchange）。扩展的 ASCII 码用 8 位二进制数编码，可以表示 $2^8 = 256$ 个字符。此外，常用的编码方式还有 EBCDIC 码、Unicode 码等。

**5. 程序设计中数据的表示**

在程序设计中数据的表示方式是非常重要的，数据在程序中用于描述程序所处理的对象。对数据的描述需要从三个方面进行，即数据的名称、数据的特征以及数据的特征值。更具体地说，在程序中被处理的数据需要有一个名字来表示；而且在对其处理之前还必须知道该数据的特征，即它可以表示哪一类值且表达范围如何；如果将该数据用于参加某种运算或者用于某种判断还必须知道它的特征值，即所具有的内容。

在程序中所处理的数据不但要命名以便于操作该数据，而且还必须确定该数据能够表示的数据种类、能够表示的数据范围以及能够参加何种操作(运算)。在程序设计中声明或定义一个数据对象时就必须用数据类型来指定上述信息，以便编译程序在处理这些数据对象时能够为它分配合适的存储空间，并能够检查对其操作的正确性。

程序中所用到的数据在程序的运行过程中都占据一个或一段特定的存储单元，这些存储单元在存储器中都有一个起始位置，这个起始位置称之为数据在内存中的存储地址。在程序设计中定义或者声明变量，其作用之一就是为了能够建立用变量名字所表示的数据与具体内存地址之间的对应关系。

在程序中使用的数据主要有两个大类：常量和变量。在程序的运行过程中，其值不允许发生改变的数据称为常量，在各种计算机程序设计语言中都有相应的常量表示方法。在程序的运行过程中，值有可能(允许)发生变化的数据称为变量。一般地，变量是指在程序中程序员用符号来显示命名的数据对象。大多数程序设计语言都是强制声明的，即规定变量必须先声明后使用。在程序中定义变量的目的就是指定该变量所表示的数据在程序中的三个方面特征。在具体的程序设计语言中，对变量的声明(定义)方法有所不同，几乎所有高级程序设计语言对变量解释方法都是类似的，即变量具有名字和数据类型；变量在所处程序被编译处理

后与一个相应的内存地址相联系；变量具有值，其值可以是确定的，例如 C 语言中的全局变量或静态变量；其值也可能是不确定的，例如 C 语言中的自动变量；确定的值可以参加相应的操作或用于判断，对于不确定的值，其操作后的结果是不可预知的。

## 1.3　算法特征及算法的描述方法

计算机程序设计的目的是通过计算机解决实际问题。程序设计主要有两个方面的任务：首先是将需要用计算机处理的实际问题抽象为数学模型，并设计出解决这个问题所需要的方法和步骤即算法；然后用合适的计算机程序设计语言对所设计的算法进行编码处理，即编制程序。在处理实际问题的程序中应该包含两个方面的内容：要处理的对象和对这些对象的处理方法。对于那些要处理的对象，用数据和数据之间的关系即数据结构来表示，而对对象的处理方法用算法来表示。著名计算机科学家沃思（Nikiklaus Wirth）用公式："程序 = 数据结构 + 算法"诠释了计算机程序设计的本质。

### 1.3.1　算法的概念与特征

**1. 算法的基本概念**

按照软件工程的基本原理，开发一个软件的过程可以分为软件分析阶段、软件设计阶段、软件编码阶段、软件测试阶段以及软件的运行和维护阶段。其中软件设计阶段的主要任务就是建立起软件的结构，确定软件的各个组成部分和实现这些部分的算法，在此基础上才能编制出程序最终的实际代码。从某种意义上说，算法设计的结果决定了程序的质量。

算法是用计算机解决实际问题的方法和步骤，是一组明确的可执行步骤的有序集合。例如，下面描述了解决"将两个实数按大小顺序排列"问题的方法和步骤。

**算法**：两实数排序算法。

第一步　算法开始；

第二步　输入原始数据：取得用于排序的两个实数 $a$ 和 $b$；

第三步　若 $a > b$，按先 $a$ 后 $b$ 的次序输出结果；否则，按先 $b$ 后 $a$ 的次序输出结果；

第四步　算法结束。

**2. 算法的基本特征**

算法必须同时具有以下五个特征。

（1）有穷性　一个算法必须能够在算法所涉及的每一种情况下，都能在执行有穷步操作之后结束。算法的有穷性特征是算法和计算方法之间最明显的区别，例如求正弦函数值的公式

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \cdots$$

没有表示出计算到何时为止，所以这样的描述只是计算方法而不是算法。当以某种方式表述了计算到多少项为止的时候（例如，某一项的绝对值小于 $10^{-5}$时停止计算）才能称之为算法。特别需要注意的是，并不是将解决问题的方法写成几个步骤就是算法，还需要注意这些步骤如果运转起来，以及是否能够正常结束。例如，下面所描述的解决问题步骤中，虽然有算法结束的步骤，但从这些步骤执行的情况来看，永远不会执行第五步，所以不满足于

有穷性，不能称为算法。

**算法**：数值累加。

第一步　算法开始；

第二步　$n \leftarrow 0$（即设置 $n$ 为 0）；

第三步　$n \leftarrow n+1$（将 $n$ 值加 1）；

第四步　转回第三步；

第五步　算法结束。

（2）确定性　算法的每一步操作，其顺序和内容都必须精确地惟一确定，不能有任何的歧义性。防止歧义性是程序设计中必须遵循的基本准则之一，无论使用何种符号、何种数据、何种操作都必须保证其惟一确定的精确意义。比如，在日常生活中描述某种需求的语句："给我一些某某东西"就不能用于描述算法，因为"一些"没有准确的数量描述。

（3）可执行性　算法所描述的每一步操作都必须是可行的，即必须是可以付诸实施并能够具体实现的基本操作。

（4）输入数据　所谓输入数据是指算法在执行的时候需要从外界获取的必要的数据信息，如算法的初始数据、加工数据等。一个算法有 0 个或多个输入，即有可能算法所需要的数据在设计算法时已经嵌入到算法之中，所以该算法在执行时不需要输入数据；当然算法在执行的时候也可能需要输入若干个数据。

（5）输出数据　算法是求解某种具体问题的方法和步骤，所以算法必须能够在执行后告知一个或一个以上的结果，即算法应有一个或多个输出数据，否则算法将没有任何意义。

**3. 算法的基本结构**

一般地说，由于需要解决的问题是各种各样的，所以算法的构造是千变万化的，其形式也是千姿百态的。但算法的最基本的组成形式和构造成分只有三种基本结构，任何复杂的算法都是这三种基本结构按某种方式的堆砌，这三种基本结构是：顺序结构、选择(分支)结构和循环结构。

（1）顺序结构　顺序结构是一种简单而基础的算法基本结构，其主要特点是：各个操作按其出现的先后次序依次顺序执行一遍。事实上，任何一个问题分解到了足够小的范围时都可以用一连串连续的操作加以解决，因而任何一个算法在一个适当的范围内都是顺序操作，所以顺序结构是构成算法最根本的基础结构。顺序结构算法执行过程如图 1-5 所示。

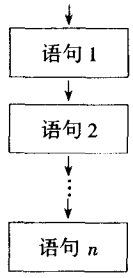


图 1-5　算法的顺序结构

（2）选择(分支)结构　在较复杂的算法设计中，必然要涉及到判断的问题，即具体执行何种步骤要根据某个或某些条件的判断结果来选择。选择结构是算法在描述稍复杂问题时所必不可少的基本结构，其主要特点是：根据给定条件的成立与否来选择执行不同的操作。最基本的选择(分支)结构有两种，如图 1-6 所示。

（3）循环结构　在算法中若要描述某些操作反复执行的概念，就要使用循环结构。循环结构的特点是：根据所给定的条件来判断是否需要重复地执行一组操作，当所给条件为真时执行，否则退出。算法的循环结构执行过程如图 1-7 所示。