第 1 章　计算机、程序设计和 C++语言概述

为了全面认识和理解 C++程序设计思想，简单回顾一下计算机的工作原理，数据在内存中的存储形式，以及程序设计的发展趋势是很有必要的。学习本章内容后，读者会对计算机、程序设计和 C++语言有个整体上的概念。

1.1　计算机概述

在世纪之交的 2000 年回顾一下计算机技术蓬勃发展的简史，有助于我们从第一计算机语言的角度掌握本课程的内容，提高计算机文化的素质，昂首进入充满生机的 21 世纪。20 世纪 40 年代，许多新的科技领域，如核反应堆的控制，导弹飞行轨迹的控制等，都要求在较短的时间内完成较复杂的计算工作，靠人工计算是不可能的，必须想办法依靠某种形式的计算机。美籍数学家冯·诺依曼（Von.Neuman）教授指出，如果用二进制而不用十进制进行数值运算，就可以利用电来进行运算，利用电路的开和关两种状态来表示二进制中的 0 和 1，用重叠组合的开关电路就可制成一种计算工具，于是在 1946 年诞生了世界上第一台电子计算机 ENIAC（Electronic Numerical Integrator And Computer）。它用二进制代替十进制完成了复杂的数值运算。但是运算指令和数据还是通过重新连接电路和设定开关来完成的。通过实践，冯·诺依曼又提出存储程序（Stored Program）的概念，即将运算指令和数据用数字的方式存放在电子计算机的存储器中，也就是首先提出了用软件来控制计算机的操作。这就形成了著名的冯·诺依曼原理——“二进制和程序存储控制”的计算机结构思想，为以后计算机的发展奠定了理论基础。

自 ENIAC 诞生以来，计算机的研制、生产和应用以迅猛的速度发展着，到 20 世纪末计算机已经历了电子管（第一代），晶体管（第二代），集成电路（第三代）和大规模集成电路（第四代）四个阶段。目前应用的计算机，它里面所用的集成电路芯片的集成度越来越高，有的被称为超大规模集成电路。这四个时期的计算机都基于冯·诺依曼原理，所以这四代计算机也称为冯·诺依曼计算机。

现在，计算机已是一种现代化的处理信息的工具。数字信息、文字信息、图像信息、动画信息和声音信息都可以通过计算机来进行存储处理。20 世纪 90 年代以来，许多国家已致力于集文（字）、图（形）、声（音）于一体的多媒体计算机 MPC（Multimedia Personal Computer）的研究，一台多媒体计算机可以具有电视机、录像机、计算机、电话机和录音机等的功能，计算机的应用开始进入每个家庭。加上计算机网络的发展，从小型的局域网到全球的 Inter 网，从传递信息的角度上讲，地球已变得越来越“小”了。与此同时，科学家们还在研制第五代计算机（非冯·诺依曼计算机），它采用完全新的工作原理和体系结构，它的工作接近于人们思考问题的方法，是一种能与人交谈，自我学习，自我推理并有决策能力的

智慧型计算机。这是计算机发展的方向，但近期还难以实现。本课程中，计算机指的是目前流行的大规模集成电路计算机。

计算机系统由硬件系统和软件系统两部分组成。

计算机硬件系统指的是计算机的具体设备，即实物。它由主机和外部设备两部分组成。而计算机主机又由中央处理单元 CPU（Central Processing Unit）和内存储器（简称“内存”）组成。CPU 包含运算器和控制器两部分，外部设备则由输入、输出设备和外存储器组成。

计算机的软件系统指的是程序。为了使计算机正常工作，程序是必不可少的。计算机做任何工作，都是通过存储在内存中的程序来实现的，可以说程序是计算机工作的“灵魂”。根据程序的不同用途，软件系统又可分为系统软件和应用软件两部分。系统软件一般是指生产厂家或公司在出售计算机时提供给用户的，它用来管理计算机的各项工作和为开发运行用户程序配备必要的软件设备。如操作系统 OS（Operating System），编译程序（Compiler），解释程序（Interpreter）以及一些集成（多功能）开发环境（软件）等。应用软件包括用户向厂家或公司定制的专用软件包和用户自己建立的各类源程序（高级语言编写的程序）或执行程序等。操作系统用来管理计算机内的所有资源（包括软、硬件资源）。如何充分应用计算机的全部资源和最大限度地发挥计算机各部分的作用是设计操作系统软件的指导思想。在操作系统的支持下，用户才能编辑源程序，并利用编译程序或解释程序翻译源程序成为机器能执行的二进制指令代码。集成开发环境是一个集编辑、编译、运行于一体的系统软件，它为用户编制应用软件提供很大的方便。

1.2 数据在计算机中的存储形式

1.2.1 位、字节、字的概念

在主机内的存储器，称内存储器，简称内存。要运行的程序和数据都存放在内存中。那么内存是由什么组成的呢？它原来是由千千万万个具有二个状态的电子开关组成的。电子开关打开时的状态为“1”，闭合时的状态为“0”。每个电子开关用计算机术语“位（bit）”来称呼，它可以代表二进制数（逢二进一）的一个基本单元。计算机内存就是一个庞大的二进制基本单元——电子开关的集合体。

要存放信息，必须把这些电子开关有机地组织起来，一般用若干个二进制“位”组成一个“字节”(byte)，多数计算机用 8 位组成一个字节，如图 1.1 所示。

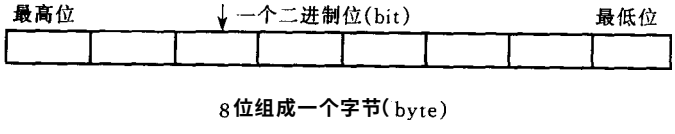


图 1.1 字节示意图

一个字节可以存放一个字符，当然这个字符是用一个二进制数来表示的，这个二进制数就是这个字符的二进制代码。计算机上所应用的全部字符（字母、数字或其他专用字符）都有相应的二进制代码，如字母“a”，它的二进制代码是：

0	1	1	0	0	0	0	1

这些字符和二进制代码之间的对应关系，很多计算机系统采用 ASCII（American Standard Code for Information Interchange 即“美国标准信息交换码”）代码。这个代码与字符之间的对应关系见附录 I。

计算机内存就是由很多排列整齐的字节组成，为了管理方便，每个字节都有相应的位置编号，这个编号就是这个字节的“地址”，通过地址可以找到内存中任何一个字节的内容。

一个字节可以存放一个字符，但要存放一个整数或一个实数，一个字节就不够了，有的系统上整数用 2个字节或 4 个字节来存放，而实数用 4个字节来存放。由一个或若干个字节组成一个“字”，一个字可以用来存放一个数据或一条指令。

1.2.2 内存单元、内存单元地址和指针的概念

一个内存单元可以用来存放一个数据或一条指令，由于内存中每个字节都有自己的地址，因此每个内存单元也有自己相应的地址，由一个字节组成的内存单元的地址，就是这个字节的地址，而由多个字节组成的内存单元的地址则指的是组成这个内存单元几个字节中第一个字节的地址。假定程序中已定义了三个短整型变量 i、j、z，编译时系统分配给它们各一个内存单元，每个内存单元占两个字节，如图 1.2所示，图中每个长方形框代表两个字节。如果分配给变量 i 的内存单元地址是 5000 （这只是假定的地址），那么变量 i 占用地址为 5000 开始的两个字节（即地址为 5000 和 5001 两个字节），5000 是变量 i 的内存单元地址，简称变量 i 的地址，C/C++语言中用符号“ &i”表示，这里 & 表示取地址，&i 的值是 5000。若定义变量时 j是紧跟在变量 i 之后，变量 z 又紧跟在 j之后，那么系统分配给这些变量的内存单元一般是连续的，即分配给变量 j 的内存单元地址是 5002 ，分配给 z 的内存单元地址是 5004，所以变量 j的地址 &j 的值是 5002（变量 j 占用地址为 5002 和 5003 两个字节),而变量 z 的地址 &z 的值为 5004 （变量 z 占用地址为 5004 和 5005 两个字节），所以，当程序定义了某一变量以后，变量名和它占用的内存单元地址有直接的对应关系。

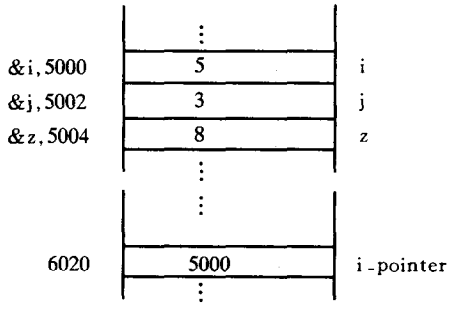


图 1.2 内存单元分配示意图

请注意，要区别一个内存单元的地址和这个内存单元的内容（即变量值）这两个概念。对变量值的存取是通过地址来进行的，根据变量名和地址的对应关系，找到变量的地址，然后从这个地址所标识的存储单元中进行存取数据的操作。图 1.2 中，设变量 i，j 的值分别是

5和3，要执行一个操作 z=i+j 时，过程是这样的：从变量 i 存储单元的地址 5000 开始的两个字节中取出变量 i 的值 5，再从变量 j 存储单元的地址 5002 开始的两个字节中取出变量 j 的值 3，通过 CPU 将它们相加后得 8 送到变量 z 内存单元中，即送到地址为 5004 开始的两个字节中。这种按变量地址存取变量的方式称为“直接访问”方式，这里的“访问”指的是在内存单元中存取（或称读写）数据的意思。

总之，通过变量的地址能找到变量的内存单元，因此，把变量的地址称作变量的“指针”，如地址 5000 是变量 i 的指针，地址 5002 是变量 j 的指针。通过变量的指针可以找到变量的内存单元来对变量进行读写操作。指针指的是某个内存单元的首地址（或说一个内存单元的入口地址），因此 C 语言中应用的指针是有数据类型的（详见第 2 章），而每一种数据类型的变量在内存中存放的数据是以内存单元为单位的。

与“直接访问”相对应的另一种访问内存的方式是“间接访问”方式。C 语言中可以定义一种变量专门用来存放地址，假定我们定义一个变量 i_pointer 是用来存放整型变量地址的，系统编译时分配给这个变量的地址假定是 6020，那么可以通过下面赋值语句将变量 i 的地址赋给变量 i_pointer：

```
i_pointer = &i;
```

变量 i 的地址值是 5000，所以通过以上赋值语句就把地址值 5000 放入地址是 6020 开始的两个字节中，i_pointer 的值就是 5000。这样，我们就可以对变量 i 进行间接访问了，过程是：先从变量 i_pointer 中提取变量 i 的指针 5000，然后到地址是 5000 和 5001 两个字节中存取 i 的值。图 1.3 表示直接访问和间接访问的示意图。

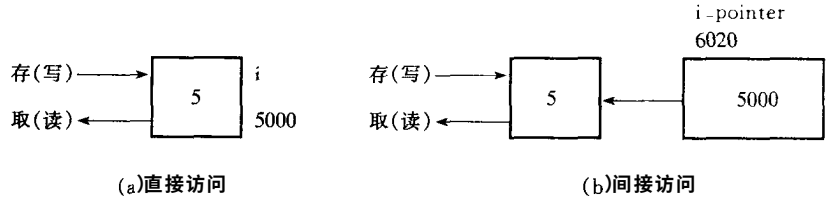


图 1.3 直接访问和间接访问

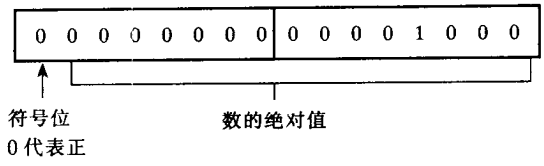
图 1.3 (a) 表示对变量 i 的值直接进行读写操作（直接访问）；图 1.3 (b) 表示对 i_pointer 所“指向”的内存单元进行读写操作（间接访问）。因为 i_pointer 的值是变量 i 的地址，所以称 i_pointer 指向变量 i，借助于 i_pointer 可以对变量 i 进行读写操作。当 i_pointer 的值用变量 j 的地址 &j 重新赋值，那么借助于 i_pointer 可以对变量 j 进行读写操作因为 i_pointer 的值可以改变，所以称它为“指针变量”。关于指针变量的定义详见第 2 章，这里只作概述，目的是说明借助于它访问内存的方式（间接访问的方式）。

1.2.3 原码、反码和补码的概念

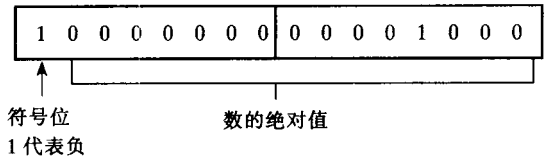
内存中的每一个数都占用一个内存单元，而这个内存单元中的最高位用来表征数的符号，以 0 代表正数，以 1 代表负数，其余各位代表数的绝对值。

一个数的代码有原码、反码和补码三种表达方式。而正数的原码、反码和补码的形式都相同，没什么差别，如一个短整数，在内存中占两个字节，假定它们在内存中是并排的，则

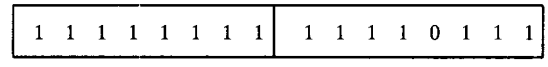
+8 的原码、反码和补码都具有如下的形式：



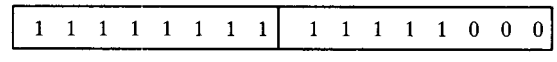
请读者注意，负数的原码、反码和补码的表示形式是不同的，如 −8 的原码是：



负数的反码规定：符号位不动，其余各位对原码取反。如 −8 的反码是：



负数的补码规定：符号位不动，其余各位对原码取反，然后加 1。也就是说，一个负数的补码是它的反码加 1。如 −8 的补码是：



计算机内存中的数都是以补码的形式存放的，因为用补码表达一个数时，0 的代码是惟一的，同时用补码进行运算时，数的符号位用不着单独处理。当已知一个数的补码反过来要求其原码时，只要记住“一个负数补码的补码就是它的原码”，即把补码当“原码”，再求一次补码就得原码。

1.3 程序设计初步

1.3.1 程序设计语言

要计算机按人们预先安排的步骤进行工作，就要解决人一机语言交流问题，人们给计算机一系列的指令，计算机按人们给定的指令一步一步的工作，这些指令的集合就是程序设计语言。

计算机内的数据和指令都是以二进制形式存放的。计算机内部传递信息并实现各种操作，也是由二进制的指令来完成的，也就是说，计算机只能识别和执行二进制指令。因此，在计算机发展初期，人们就是用手工编写出一条条二进制指令，这些二进制指令的集合就称为“机器语言”。这种机器语言看上去都是由简单的 0 和 1 两个数字组成，但实际使用起来却非常繁琐，容易搞错，除了计算机专业人员会去研究和使用它们外，其他人极少过问。

为简化机器语言，人们想到用符号来代替二进制形式的指令，符号指令便于记忆和使用。这些符号也称作“助记符”。例如用 ADD 代表两数相加，用 MOV 表示数据传送等。由这些符号组成的指令集称为“符号语言”，也称“汇编语言”。计算机不能直接接受汇编语言，要通过“翻译”，把汇编语言翻译成二进制的机器语言，机器才会执行指令，这个翻译是通过一个事先装入计算机中的“汇编程序”软件来完成的。

汇编语言比机器语言简单易学，为专业人员设计系统软件提供了很大的方便。但对于一些非专业人员来说，仍感到难以掌握，而且不同的机器都有自己的一套编码，这就严重地影响了计算机的进一步推广应用。直到 20 世纪 50 年代，世界上出现了第一个“高级语言”——FORTRAN语言以后，计算机的应用才得到了飞速的发展。所谓高级语言指的是它更接近于人们习惯的自然语言和所用的数学公式。如要计算机打印输出写“ print”，要 打开某文件的通道写“ open”，要计算 x 的正弦函数用 sin(x)等，这为我们广大非专业人员应用计算机提供了极大的方便。继 FORTRAN 语言以后，高级语言发展迅速，至今已有几百种不同的高级语言，其中常用的有： FORTRAN 语言适用于数值计算； BASIC 语言适用于初学者；COBOL语言适用于商业和管理； PASCAL语言适用于教学；C语言适用于编写系统软件和工程应用等。

C语言除具有高级语言的优点外，还具有强有力的处理硬软件的接口和低级语言的特点，因此，可用它来编写系统软件。 C语言简洁、方便、灵活，且具有各种控制流结构，以满足下文所提到的结构化程序设计的需要。 C语言的代码质量高，程序运行速度快。

在 C语言的基础上 1983年首次推出 C++ 语言，C++语言是 C 的超集，它继承了 C 全部的优点，并提出程序设计的新概念和新方法，自 20 世纪 90 年代以来开始成为程序设计的新潮流。

1.3.2 结构化程序设计

20 世纪 60 年代末到 70 年代初随着科技的高速发展，程序的规模越来越大，可是当时的编程技术却仍然停留于手工业方式。软件设计各自为政，效率很低，可读性差，无章可循，错误百出，调试艰难。面对这一严重情况，荷兰学者 E.W.Dijkctra 提出结构化程序设计（Structured Programming ，简称 SP）的理论。经过多年的实践，结构化设计的理论和实施方法日益完善，现已被很多人接受。结构化程序设计方法提出了一些大家都要遵循的原则，我们把它归纳为 32 个字：“自顶向下，逐步细化；清晰第一，效率第二；书写规范，缩进格式；基本结构，组合而成”。

“自顶向下，逐步细化”对一个大程序的设计特别有用，意思是根据题目的总要求划分出若干个相对独立的模块，而每个模块根据它所完成的功能又可细分成若干个子模块，由每个子模块根据它所完成的功能又可细分成若干个子模块，由每个子模块再考虑具体的程序段，如此从上而下，逐步细化，一直可以细化到不能再分割为止。

“清晰第一，效率第二”，这是从提高程序的可读性，调试时改错容易，便于软件交流和维护等观点出发的，认为程序在清晰的基础上，再考虑它的效率。

“书写规范，缩进格式”，这是程序设计风格的重要内容之一。写程序时不要把语句“堆”在一起，而是按程序的不同层次写成锯齿形。具体例子可看各章的例题。

“基本结构，组合而成”，程序不论大小，简单还是复杂，都规定用三种基本结构组成，

它们是顺序结构、选择结构和循环结构。实践证明，用这三种基本结构可以组成各种模块实现任何复杂的算法。

C语言完全支持结构化程序设计，它有多种结构化语句供用户在不同场合选用。C++语言也继承了这种方法。

1.4 C++语言概述

1.4.1 C/C++语言简史

像操作系统这样的系统软件，以前是不能用高级语言来编写的，因为它需要跟硬件打交道，像 UNIX 操作系统，过去也都是用汇编语言编写的。由于汇编语言依赖于计算机硬件，程序员编程的劳动强度很大，程序的可读性和可移植性都比较差。为了克服这个缺点，用高级语言来编写是个好办法，可是一般高级语言“离硬件远”，不能对硬件进行操作，如内存地址的操作、位操作等。那么是否可以找到一种既具有高级语言的优点，又具有低级语言特性的语言呢？经过多年努力，人们终于研制出 C语言，它不仅可以用来编制用户的应用软件，还可以用来编制系统软件。C 语言是怎样发展而来的呢？简单地说，C语言是由 B语言发展而来的，而 B语言又是由“A语言”发展而来。

“A 语言”指的是高级语言 Algol 60 。1960年出现的 Algol 60是一种面向问题的过程式高级语言，它离硬件远，不宜用来编写系统软件。1967年英国剑桥大学的 Matin Richards在 Algol 60基础上推出了比较接近硬件的 BCPL（Basic Combined Programming Language ）语言，1970年美国贝尔实验室的 Ken Thompson 以 BCPL语言为基础又设计出简单而又很接近硬件的 B语言（取 BCPL 的第一个字母),并用 B语言写了 UNIX操作系统初版。1973年贝尔实验室的 D.M.Ritchie 在 B语言的基础上设计出 C语言（取 BCPL 的第二个字母），它保留了 B语言的精炼，接近硬件的优点，又克服了它们过于简单、无数据类型等缺点。后来 K.Thompson 和 D.M.Ritchie 合作进一步改写 UNIX 操作系统，到 1977 年 UNIX操作系统得到日益广泛使用，与此同时 C语言也迅速得到推广。1978 年以后，C语言已先后移植到大、中、小微型机上，C语言已风靡全世界，得到越来越多人的赞誉。

1978 年 Brian W.Kernighan和 Dennis M.Ritchie（简称 K&R)合著了《The C Programming Language》一书，这本书成为后来广泛使用的 C语言的基础，被称为标准 C。后来，美国国家标准协会 ANSI（American National Standard Institute）又推出了新的标准 C，它比原来的标准 C又有了较大的发展，1988 年 K&R按国家标准重写了他们的经典著作《The C Programming Language》。

尽管 C语言具有很多优点，但随着 C语言的广泛应用，也逐渐显示出它的局限性。首先，它仍是一种面向过程的程序设计语言（Procedure Oriented Language 简称 POL），程序中数据和操作是相互分离的，缺乏可靠性和安全性，再则是代码的可重用性差，而且程序的规模达到一定程度时，程序的复杂性大大增大，调试极为困难等。由此，1980年贝尔实验室的 Bjarne Stroustrup 开始对 C 进行改进和扩充，1983 年正式取名为 C++ （使用了 C语言中的自增运算符 ++ ，表示是 C 的进一步扩充和改进），以后再经几次修订，于 1994 年制订了 ANSI C++ 标准草案。目前，C++仍在不断发展中。由于 C++的出现，有人称原来的 C为

“传统的 C”。

C++是 C 的超集，C是 C++ 的基础。C++ 继承了传统 C 的全部特性，也就是说支持面向过程的程序设计，同时也支持面向对象的程序设计（Object Oriented Programming 简称 OOP)。对小型程序使用 C++面向过程程序设计完全可以，但对于中大型程序应该使用 C++ 面向对象程序设计。

1.4.2 C++ 中对象、类和抽象的概念

对象（object）代表着一个待处理问题的实体，它一般有自己的属性和行为。世界是由各种各样的对象（如汽车、火车、自行车、人、博士生、研究生、大学生、硕士生、工人、动物、狗、猫、鸡、飞机、飞船、卫星等）所组成的。一个对象可以通过消息传递与其他对象实现通信。

对象按其属性和行为的不同可以分成各种各样的类（别），如汽车、火车、自行车属车类，硕士生、大学生、中小学生属学生类，帆船、轮船、气垫船属船类，鸡、狗、猫属动物类，飞机、飞船、火箭属航天类，数控机床、磨床、冲床属机床类等。每个类（别）的对象在属性和行为上有它们的共性。属性表现为其内部的各种数据的性质，而行为表现为其处理这些数据的过程或方法。面向对象程序设计中把两者封装在一个类（class）的代码中，属于该类的对象具有该类的共性，但对象拥有自己具体的数据值和自己的行为结果。

面向对象程序设计的工作很大程度上决定于类的设计。因为 OOP 中具体的对象是要根据类型来定义的，这类似于传统 C 中用标识符 int(integer）来定义整型变量一样。类是具有相同属性和行为（共性）的对象的抽象，如“学生”概念上是抽象的，它具有硕士生、大学生、中小学生等的共性。但仔细一想，硕士生、大学生、中小学生还可以认为是抽象的，如大学生具有理、工、文科大学生的共性。即大学生是理、工、文大学生的抽象。由此可见，抽象是具有层次的，可分高层次抽象和低层次抽象，它们的区别在于低层次抽象比高层次抽象更具体一些。抽象（abstract）是人们处理复杂问题的强有力的工具。

面向对象程序设计中由类来定义对象。如理科类学生张三、李四，工科类学生王鑫、郑玉等。可以说类是抽象的，对象是具体的。

1.4.3 C++ 面向对象的特性

1.封装性（encapsulation)

抽象的过程就是建立类的过程，在 C++面向对象程序设计中，每个类的属性和行为都封装在一个模块中，对外是不透明的，只有通过一定的对外接口才能访问内部的信息，这就好象是我们要知道某人的情况只有通过某种方式（如打电话、发 Email 或亲自去拜访他等）才能获得。这就是所谓 OOP 的封装性和数据隐藏，这大大提高了程序模块的可靠性和安全性。它是 OOP 的重要特性之一。

使用类定义对象时，人们无须关心类中内部的机理，即不必知道该类对象内部是如何工作的，人们关心的是该类对象的功能。类的对象实际内部工作情况和数据变化都被隐藏起来，如要知道对象内部具体的数据须通过定制的对外接口才能获得。

抽象有层次，类也有层次，不同层次的类又可定义不同的对象，所以对象有高层次类定义的对象和低层次类定义的对象的区别。在不要求十分具体的问题上，可用高层次类定义的

对象，而在要求比较具体的问题上，则可用低层次类定义的对象。前面我们提到的世界是由各种各样的对象所组成的，这里的对象包含着高层次类的对象和低层次类的对象。读者也许会问，前面的对象例子中，哪些是高层次类的对象？哪些又是低层次类的对象呢？这要针对某一个具体事例的设计才能分清。但有一点是共同的，即所谓的层次是相对的。

2. 继承性(inheritance)

C++中的继承性提供了创建新类的一种方法。就是说可以在已有类的基础上进行修改或扩充来建立一个新类。新类可以共享被继承类的属性和行为，同时还可以额外扩充新的行为和数据，使之具有更具体、更完善的功能。继承的本质是行为共享和代码可重用性。新类称为被继承类的子类或派生类，而被继承类称为父类或基类。父类和子类的关系就是高层次抽象和低层次抽象的关系。抽象是有层次的，也就是说类是有层次的，层次体现在它的继承性中。例如上面提到的学生类是硕士生类、大学生类和中小学生类的父类或基类，而硕士生类、大学生类和中小学生类则是学生类的子类或派生类。大学生类可以共享学生类的方法和属性，同时具有大学生所特有的行为和属性。

由派生类的继承性决定类的上下级层次关系，同时也决定了相应的类所定义的对象是父类（基类）对象还是子类（派生类）对象。继承性也是 OOP 的重要特性之一。

3. 多态性(polymorphism)

不同对象对相同的消息有不同意义的解释，从而使它们表现的行为和动作也不相同。这种现象称多态性。如同样一个打印消息，送到图像对象时，打印一幅图像，而送到文本对象时，则打印出一篇文章。现实世界中，就相当于同样一个任务给不同的单位或个人去完成，它们所采用的方法和行为可以不一样，从而得到的结果也不完全相同。这里下达同样一个任务，在 OOP 中就是发同样一个消息，而不同的单位和个人在 OOP 中就是不同的对象。这就是多态性。

继承性和多态性相结合，可以产生一系列种类繁多的类（别）和特性各异的对象，而通过消息实现对象之间的通信。可以说“面向对象的程序 = 对象 + 消息”。

OOP 的封装性、继承性和多态性接近于人们对事物的认识规律和思维方法，这就是 OOP 当前成为程序设计新潮流的主要原因。

1.5 C++ 字符集和标识符

1.5.1 C++ 字符集

C++程序由下列字符所组成的单词或标识符写成：

(1) 小写英文字母　　a～z

(2) 大写英文字母　　A～Z

(3) 数字字符　　0～9

(4) 27 个特殊字符

+　－　*　/　=　：　；　？　\　～　|　！　#　%　&

()　[]　{}　^　<　>　_（下划线）空格　，　·　″　′

1.5.2 C++ 标识符

标识符是程序员用来命名程序中的一些单元或模块，如类型名、对象名、函数名、变量名、常量名等等。C++ 规定：标识符由字母、数字字符和下划线组成，并以字母或下划线开头，其后可以跟零个或多个字母、数字字母或下划线组成。

定义标识符时应注意：

(1) 标识符中的大小写字母是有区别的。如 Print，print，Stu，stu，X，x，Year，year 等都代表不同的标识符。

(2) 标识符的长度（字符个数）可以任意，但不同的编译系统识别标识符的长度是有限的，如有的编译系统只能识别标识符的前 32 个字符。

(3) 实际编程时，提倡使用有意义的英文单词，最好能做到“见名知义”，帮助提高程序的可读性。

(4) 定义标识符时，避免使用系统已经定义过的标识符。这些标识符是系统的保留字，用来定义系统的关键字。系统的关键字对 C++编译程序来说有着特殊的含义，详见附录Ⅱ.

1.6 C++程序的基本结构

1.6.1 两个简单的 C++程序

为了了解 C++程序的基本结构及其特点，让我们来示范两个简单的 C++程序。一个是面向过程的程序设计，求两数之和。另一个稍微复杂一点，简单地说明面向对象程序设计中类的设计，它包含封装性和数据隐藏以及怎样通过对外接口函数跟封装在内部的数据实现通信。

【例 1.1】 求两数之和。

```
// ex1_1.cpp   the sum of a+b
#include "iostream.h"
void main()
{
    int a,b;
    int add(int,int);
    cout<<"Enter two int numbers: \n";
    cin>>a>>b;
    int c=add(a,b);
    cout<<"a+b="<<c<<endl;
}
int add(int x,int y)
{
    return x+y;
}
```

程序运行结果是：

```
Enter two int numbers:
11  17  ↓
a + b = 28
```

【例 1.2】求五边形的周长。

该程序的任务是计算直角坐标中五边形的周长，每边的长度由两点间的距离来决定。直角坐标中的点有 x 和 y 两个坐标值，在尚未给出具体的 x 和 y 之前，点（设为 p（x，y））的概念是抽象的，可以用类来表达，我们不妨称为点类。它具有直角坐标中所有点（如p1(x1，y1)，p2(x2，y2)，p3(x3，y3)…等）的共性：都有 x 坐标值和 y 坐标值。坐标值不能随便被更改，否则它就不代表原来的点了。因此在设计点类时，坐标值 x 和 y 应把它们隐藏起来，不能随便被外界访问（读／写），而只能通过特定的一些方法才能读写它们，即提供跟外界交互的接口（函数）。假定处理坐标值的方法（不同的用户可以有不同的设计）有下列三个：

（1）设定点的初值

（2）获得横坐标值

（3）获得纵坐标值

C++ 中类的设计由关键字 class 开始，后跟类名，假定用 CPoint，下面是一对大括号，大括号中的内容即为该类属性（数据）和行为（函数）的封装体，其中 public 关键字下面的方法是对外的接口函数，private 关键字下面是被隐藏的数据（这里是坐标值 X 和 Y）。例 1.2 中示范了点类 CPoint 的设计，通过类 CPoint 可以定义具体的点对象 p1(5,7)、p2(2,3)等。关于类和对象定义的详细内容见第 7 章。

```
// ex1_2.cpp
#include "iostream.h"                                    //编译预处理
#include "math.h"
class CPoint                                             //类的定义
{
    public:
        CPoint(int x,int y) {X=x;Y=y;}
        int getX() {return X;}
        int getY() {return Y;}
    private:
        int X,Y;
};
void main()                                              //主函数的设计
{
    CPoint p1(3,4),p2(7,3),p3(8,7),p4(6,4),p5(3,8);
    double d(CPoint &,CPoint &);
    double s;
    s=d(p1,p2)+d(p2,p3)+d(p3,p4)+d(p4,p5)+d(p5,p1);     //语句部分
    cout<<"s="<<s<<endl;
}
double d(CPoint &pa,CPoint &pb)                          //函数的设计
{
```

```
    int dx,dy;
    dx = pa.getX() - pb.getX();
    dy = pa.getY() - pb.getY();
    double c = sqrt(dx * dx + dy * dy);                    //语句部分
    return c;
}
```

程序运行结果是：

```
s = 20.8518
```

1.6.2 C++程序的组成

由上面的示范程序，可看出C++程序从总体上来说是由类和函数组成，但从基本成分的角度，C++程序也可以说是由类的定义、函数、编译预处理命令、语句和注释等部分组成。

1. 类的定义

例1.2中我们简单地定义了一个在两维坐标中表达点的类型，面向对象的程序设计中，类的设计是核心，一个稍具规模的C++程序，类的定义往往是集中在一个头文件中。这部分内容详见第7章。

2. 函数

C++程序中拥有大量的函数，这是程序设计模块化的体现。函数用来完成某个特定的操作，如例1.1中的main()函数和add()函数。函数也可以出现在C++程序的类内，如例1.2中的CPoint类，其中有getX()和getY()函数表达该类的某种行为。函数也可以出现在程序的其他地方，如例1.2中的d()函数。一个程序可以包含很多函数，但一个程序中只允许有一个主函数main()(基于Windows的程序以WinMain()为主函数)，程序总是从主函数开始执行，其他函数只能通过主函数或被主函数已经调用的函数调用。注意类内函数的调用还受具体对象的限制。函数在调用前一般要说明其原型，以免调用出错。函数可以带参数，也可以不带参数。具体细节在第5章和第7章中展开。

3. 编译预处理命令

C++程序中可以看到有些程序行以“#”符号开头，这些程序行就是预处理命令。C++提供三类预处理命令：文件包含、宏定义和条件编译（详见第5章第10节）。两个示范程序中都有文件包含命令：

```
#include "iostream.h"
```

其中iostream.h是一个头文件，也称输入输出流文件。程序中由于要用到数据的输入和输出，因此要包含该头文件，所谓包含就是把头文件代码引入程序中，由于这个工作是在编译程序（见1.7节）前完成的，所以称编译预处理命令。例1.2的程序中除包含输入输出头文件以外，还包含一个数学库函数的头文件math.h，因为程序中要用到数学函数中的开方sqrt()函数。

4. 语句部分

C++程序中有各种各样的语句，一个函数主要由语句组成，一个语句的结束标志是分号，如一个表达式，后面加一个分号就成为表达式语句，表达式语句是C++程序中应用最

多的语句。其他还有通过类型定义变量或对象，后面也带分号，则可称为变量或对象的说明语句。只有一个分号而前面没有内容的语句称为空语句。

另外，还有选择结构语句，循环结构语句和无条件控制语句等。详见第 3 章。

5. 注释部分

C++ 程序中为了说明程序的功能或某一行的含义，可带注释。注释能帮助读者阅读和理解程序。程序编译时，注释被忽略，它不产生代码行。放在程序开头的注释，说明整个程序的功能，放在其他部分的注释，说明局部程序或语句的功能。 C++ 中可用如下两种注释方式：

```
//      ex1_1    the sum of  a + b
```

或

```
/*      ex1_1    the sum of  a + b */
```

其中带两个正斜杠“ // ”后面的注释内容要求在一行中结束，一行注释不够时，可用两行或连续几行，那么在它们的行首仍需要加“ // ”符。

注释内容还可以写在一对符号“ /* ”和“ */ ”之间，这是传统 C 中继承下来的注释方式，其中的内容可以是一行或几行。自符号“ /* ”开始到“ */ ”符号结束，其间的内容都被认为是注释内容。

1.7 C++程序的编辑、编译和连接

用计算机高级语言编写的程序，称源程序。计算机不能直接执行源程序，如前面已经提到过的要经过“翻译”，把高级语言翻译成机器语言，目前主要通过两种方式，一种是编译方式（由事先放入计算机内的编译程序来完成）；另一种是解释方式（由事先放入计算机的解释程序来完成）。这两种方式的不同点在于编译程序是把整个源程序翻译成机器语言的目标程序，运行这个目标程序得到运行结果。而解释程序是把源程序翻译一句，执行一句，再翻译一句，又执行一句，……如此一句一句进行的。显然，用编译方式，程序运行的效率高、速度快。目前，大多数高级语言都采用编译方式。源程序在编译以后生成二进制的目标代码，即机器语言指令，也被称为目标文件，通常以 .obj 作为文件的扩展名。

通常 C++ 程序中可能包含几个目标文件或库文件（扩展名为 .lib），它们都是二进制文件，应把它们连接起来，生成可执行的文件，这个连接工作由专门的软件——连接程序（又称连接器）来完成。连接以后，系统生成可执行文件，文件的扩展名是 .exe，所以也称 exe 文件。源程序通过编译和连接后产生一个可执行文件这个过程，一般来说不可能一次就获得成功，有一个反复调试的过程。从编辑源程序开始一直到产生一个可执行文件一般都由一个集成开发环境（Integral Development Environment 简称 IDE）来完成，IDE 集编辑、编译和连接于一体，在编译与连接过程中，根据是否有出错信息来决定是否要重新编辑或修改源程序。图 1.4 表示了产生 C++ 可执行文件的过程。

关于 Visual C++集成开发环境的介绍请看第 12 章。

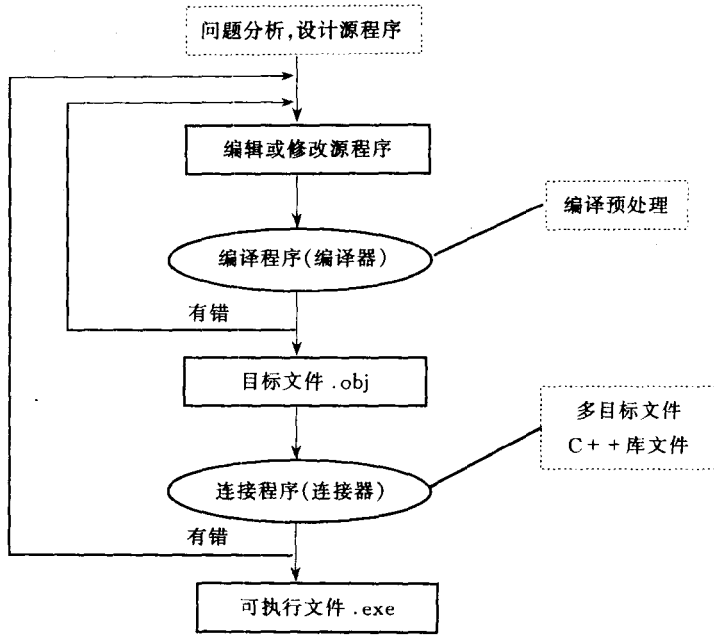


图 1.4 产生 C++可执行文件的过程

练　习

1.1 什么是冯·诺依曼原理？它在计算机的发展中起什么样的作用？

1.2 计算机内存的组织形式是怎样的？什么是计算机中的“位”、“字节”、“字”？

1.3 什么是“内存单元”？什么是“内存单元地址”？什么是“指针”？

1.4 “机器语言”、“汇编语言”和“高级语言”的区别是什么？

1.5 什么是“结构化程序设计”？它主要由哪三个基本结构组成？

1.6 什么是面向过程的程序设计？什么是面向对象的程序设计？两者的根本区别在哪里？

1.7 面向对象程序设计的核心是什么？它有哪些特性？

1.8 抽象在 OOP 中起什么作用？

1.9 C++语言的字符集由哪些字符组成？标识符的组成原则是什么？

1.10 C++程序有哪些基本组成部分？

1.11 C++程序如何实现？(自学第 12 章，上机验证本章的两个示范程序)

第 2 章　数据类型、运算符、表达式和常用的 I/O 流

数据类型是对一组变量或一组对象的抽象描述。C++语言中有十分丰富的数据类型。主要分两大类，一类是系统已经定义好的基本数据类型，如字符型、整型和实型等。另一类是用户参与定义的构造类型（也称导出类型），构造类型又称复合数据类型，具有更高意义的抽象，如结构（struct）类型，枚举（enum）类型，联合（union）和类（class）类型。C++语言中通过数据类型定义变量或对象。

为了实现数据的各种操作，C++语言提供了丰富的运算符和形式多样的表达式，足以完成各种数值计算和非数值计算。

2.1　基本数据类型

C++语言的基本数据类型有 char（字符型）、int（整型）、float（实型或浮点型）和 double（双精度实型）。

字符型的说明符为 char，被 char 说明的变量占用内存一个字节。它的前面可以加 signed 或 unsigned 修饰符，表示有符号数据（有正负之分）或无符号数据（只有正数，没有负数）。读者会问，字符型的数据是字符，怎么会跟数值一样有正负呢？第一章中我们提到过字符在计算机中的存储形式，它们是用 ASCII 代码的方式来表达的，每个字符都有它的 ASCII 代码值，在 C/C++中字符型数据的输出形式可有两种，一种是以字符方式输出字符，另一种是以数值方式输出它的 ASCII 代码值。以后一种方式输出时，可有正负之分。其实在 C/C++中，字符可以跟其他数据直接参加运算，此时字符是以它的 ASCII 码参加运算的。

整型数据的说明符为 int，被 int 说明的变量在字长为 32 位的微机中占 4 个字节，而在字长为 16 位的微机中占 2 个字节。关键字 int 前面还可加 short 或 long 修饰符，short int 类型占 2 个字节，long int 类型占 4 个字节。同样在它们前面可以加 signed 或 unsigned 修饰符，由于加 unsigned 修饰符时的符号位（一个内存单元几个字节中的最高位）也当作数字位，因此正整数的范围扩大了一倍。

实型数据的说明符有单精度类型 float（被说明的变量占 4 个字节），双精度类型 double（被说明的变量占 8 个字节），double 前还可加修饰符 long，即 long double 长双精度类型（被说明的变量占 10 个字节）。

表 2.1 列出了 C++中所有的基本数据类型，它是根据 ANSI 标准给定的类型和对应于 32 位字长微机的字节数和数值范围。

表 2.1 C++的基本数据类型

类 型 名	含字节数	数值范围
[signed] char	1	-128~127
unsigned char	1	0~255
[signed] short [int]	2	-32768~32767
unsigned short [int]	2	0~65535
[signed] int	4	-2147483648~2147483647
unsigned [int]	4	0~4294967295
[signed] long [int]	4	-2147483648~2147483647
unsigned long [int]	4	0~4294967295
float	4	-3.4E38~3.4E38 7个有效位
double	8	-1.7E308~1.7E308 15个有效位
long double	10	-3.4E4932~1.1E4932 19个有效位

注：[ ]中的内容表示可以省略，如 signed long int 可以简写成 long等。

某数据类型的数据所含的字节数可用如下运算符计算：

sizeof（数据类型）

如 sizeof（int）在32位微机的值是4，sizeof（double）的值是8。

【例 2.1】 计算数据类型所含的字节数。

```
//ex2_1.cpp
#include "iostream.h"
void main()
{
    cout<<"char:"<<sizeof(char)<<endl;
    cout<<"short:"<<sizeof(short)<<endl;
    cout<<"int:"<<sizeof(int)<<endl;
    cout<<"double:"<<sizeof(double)<<endl;
}
```

运行结果是：

```
char: 1
short: 2
int: 4
double: 8
```

2.2 常量和 const 类型说明符

程序中不能被改变的量称为常量。C++ 中常量分字符常量、字符串常量、整型常量、实型常量和用 const 类型说明符说明的符号常量。

2.2.1 字符常量

字符常量是用一对单引号括起来的单个字符，如'p'、'*'、'?'、'!'、'$'、'5'等。这里单个

字符不能是单引号′、双引号″和反斜线\，要表达这几个字符需用转义字符。

转义字符用反斜线\开头，后跟字符的 ASCII 码值或需要改变其意义（转义）的字符。字符的 ASCII 码值可用 8 进制表示形式：′\ddd′，ddd 表示三位 8 进制值，也可以用 16 进制表示形式：′\xhh′，hh 为两位 16 进制数。如′\102′或′\x42′表示字符 ′B′。

转义字符常用于难以打印的字符：如′\007′或′\a′表示响铃符，执行输出语句：

```
cout<<′\007′;
```

计算机就会响一次铃。

C++中常用的转义字符如表 2.2 所示。

表 2.2 常用的转义字符

符　号	功　能
\ddd	8 进制数表示的 ASCII 字符
\xhh	16 进制数表示的 ASCII 字符
\a	响铃
\t	水平制表（横向跳格）
\n	换行
\r	回车
\b	退格（backspace 键）
\′	单引号
\″	双引号
\\	反斜线
\0	空字符

2.2.2 字符串常量

字符串常量（简称字符串）是用一对双引号括起来的字符序列，如″Welcome Freshmen!″等。当字符串一行写不下时，可用续行符反斜线\，如：

```
″This is \
a string.
```

在使用续行符时，系统会忽略续行符本身以及后面的换行符。

字符串中可以包含转义字符、空格符或其他字符。字符串的结束标志用′\0′（空字符）。注意：

（1）空字符（NUL）是 ASCII 码为 0 的那个字符，不要与空格字符（ASCII 码为 32）相混淆。

（2）字符串″a″和字符常量′a′的区别。″a″有字符串的结束标志，因此占内存的两个字节，而′a′只占内存的一个字节。

【例 2.2】 输出字符串常量和字符常量的示例

```
//ex2_2.cpp
#include "iostream.h"
void main()
```