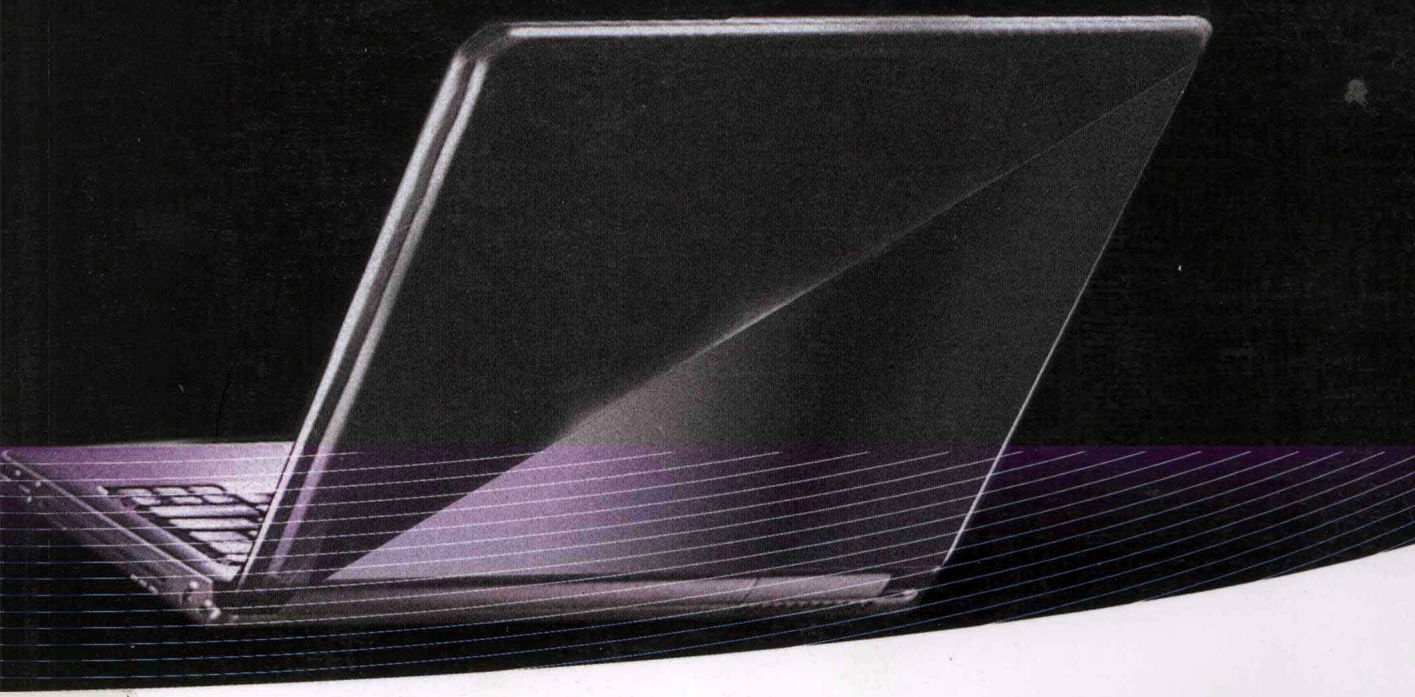
网博
黄金屋

金正海　余志勇　主编

C++实训教程
理论篇



東南大學出版社
SOUTHEAST UNIVERSITY PRESS

C++实训教程

理论篇

主　编　金正海　余志勇

东南大学出版社
·南京·

内容提要

本书着重介绍标准C++语言，从编译原理与内存管理角度，借助于汇编，深刻理解C/C++语言关键概念，并针对C++程序设计的各个知识点进行了全面、深入的剖析和提炼，为读者构建了一个完备的知识体系。本书中所有的实例都是Visual C++ 6.0集成开发环境下编辑、编译、调试通过的。

全书共18章，从最基础的编程语言概念讲起，前10章完整地介绍C/C++语言的基础语法知识和汇编的常用知识，包括过程设计、内存模型、编译器对地址的理解、复合数据类型、程序的文件组织、函数调用原理等。后8章介绍C++面向对象编程(OOP)以及面向对象高级特性，如类的设计和使用、多态、虚函数、继承、模板、异常、命名空间等知识，以及编程实践中要用到的C++标准库、STL容器、泛型算法等应用性内容，结合实例展开讨论。

本书具有由浅入深、通俗易懂和注重实例等特点，适合于没有或者缺乏C/C++程序设计经验的初学者作为标准C++语言的自学教材，同时也适合于已掌握汇编、C语言基础编程技术，需要提高C++实践能力以及对标准C++语言感兴趣的程序员参考阅读。

图书在版编目(CIP)数据

C++实训教程：理论篇/金正海，余志勇主编. —南京：东南大学出版社，2011.12

ISBN 978-7-5641-3195-1

Ⅰ.①C… Ⅱ.①金…②余… Ⅲ.①C语言—程序设计 Ⅳ.①TP312C

中国版本图书馆CIP数据核字(2011)第258309号

C++实训教程：理论篇

出版发行：东南大学出版社
社　　址：南京四牌楼2号　邮编：210096
出 版 人：江建中
责任编辑：丁志星
网　　址：http://www.seupress.com
经　　销：全国各地新华书店
印　　刷：江苏凤凰扬州鑫华印刷有限公司
开　　本：787mm×1092mm　1/16
总 印 张：51.5
总 字 数：1300千字
版　　次：2011年12月第1版
印　　次：2011年12月第1次印刷
书　　号：ISBN 978-7-5641-3195-1
定　　价：150.00元(共2册)

写给C或C++语言的初学者

常有初学者问“学习C++语言之前是否有必要先学C?”我认为这个问题是个伪问题，或者至少可以这样说：问题本身的答案并非是提出问题的人真正想要的。

回答这个问题非常简单：“当然不需要。”因为C++是一门独立的编程语言，它在语法、编译及运行环境上都不需要依赖于任何其他语言。至于说它是“C语言的超集”，那只是一种模糊的提法而已，更准确的说法应该是：C++语言从设计之初就充分考虑了对C语言的兼容性，结果使它在相当大的程度上兼容了C语言。如是而已。

但对于那样提问的初学者来说，以上会是他们想要的答案吗？我觉得不是，至少，问题没有这么简单。

C++语言支持多种编程范式：面向过程、面向对象和范型程序设计。它所兼容C的部分，也正是对面向过程进行支持的主要部分。因此，虽然C语言的基础有助于我们更快地掌握C++的面向过程部分，但面向过程也是C++语言直接支持的，因此我们完全可以从C++中直接学习面向过程编程，而无需先学C，至少理论上是这样。

但话还是要说回来，要注意到提这个问题的大多是初学者。一般讲解C++语言的教程，限于篇幅，在面向过程部分的详细程度，无法与一本优秀的C语言教程相媲美。于是，真正的问题来了：直接从C++语言开始学习的初学者，他是否愿意花费跟C语言初学者相同的时间和精力，来专门钻研面向过程程序设计？如果他不愿意，那么，他对面向过程程序设计的掌握，就比一个扎扎实实学习过C语言的人要花费更长的时间，即使只讨论“C++中的C”，结论也是一样。但我只是想强调这样一个事实，而并非要下一个“必须先学C”的结论。

我个人在初学C++语言时，觉得教程中学到的好多东西，包括面向对象程序设计，在实践中无法得心应手地加以运用。后来，因工作需要，又去学习Java语言，一下子就对面向对象有了一种开窍的感觉。再使用C++写程序时，发现OO(面向对象)起来也更顺畅了，我的好几位朋友也有过类似的感觉。究其原因，我想跟上面所分析的C语言的情况差不多。C++所支持的特性和编程范式多一些，初学者很难快速地消化全部。而Java在1.4版之前，可以被认为是纯面向对象的。也就是说，使用Java写程序，我们被迫实践面向对象编程，这就好比使用C写程序，我们被迫实践面向过程一样。而直接使用C++写程序，没有什么“被迫”，我们也就缺少了相应的“专项强化训练”。并非所有的人都需要这种“专项强化训练”，但对于初学者，面对着要掌握的复杂知识体系，它确实能起到“分而治之”的作用，从而促进对知识的消化和吸收。

还有人认为，学会了C++语言，自然就学会了C。而我认为这种观点是经不起推敲的。我们必须承认两点：第一，C语言是有用的，那么多C语言构筑起来的项目和复杂系统就是明证。虽然它没有对包括面向对象在内的各种更抽象的编程范式提供直接支持，但事实早已证明它是能够解决非常复杂的问题的。第二，C++语言的设计者所设计的是一门新式的

语言，而不仅仅是一个“更好用的C”。这就导致C++在继承C语言的同时要对它的各种特性做一些必要的扬弃。于是，就有了很多这样的情况：某种特性在C语言中非常重要，但C++语言却拥有一些新的特性来替代它，而且可能做得更好。这样，原来C语言中的那项特性在C++语言中虽然还被兼容，却被大大淡化，或不再提倡，甚至C++语言的教程中都不提那种特性。这方面的例子很多，比如“宏”，C++中有太多可以在各种不同的场合取代宏的东西，再比如可变参数，还有对指针的一些复杂使用等等。这些东西在C语言中可能是非常重要的，重要到成为C语言之所以“有用”的直接原因之一。因为这些原因，一个用惯了C++语言而从没单独学习过C语言的人，如果有一天突然被要求去负责一个C语言项目，我不认为他就一定能轻松搞定。所以，我认为“学会C++，自然也就学会了C”的观点是没有道理的。

曾有人说C语言是“结构化的汇编”，它的功能、效率和可移植性都很好地达到了它的设计初衷，它对现实项目中所面对的各种问题也都有它独特的解决方式，而同样的解决方式在C++语言中却未必是恰当的。换句话说，从解决实际问题的方式上来看，这两种语言谁也包含不了谁。而大多数人学习程序设计，最终不都是为了解决实际问题吗？所以，从实际意义上讲，C++语言从语法上几乎完全包含了C并不代表它“真正”包含了C。

反过来看，一个熟练掌握了C语言的程序员学习C++语言的情况又是怎样的呢？正如前面分析过的，C语言的面向过程“专项训练”非常有助于迅速掌握C++语言的面向过程部分；同时，大多数人最终都在实际的软件开发中运用编程语言，因此，任何语言的编程背景对其他新语言的学习无疑都会有帮助。但有一点仍需注意：C语言和C++语言终归体现了不同的编程思想，如果在学习C++语言的过程中不能适时地忘记并跳出C，有时可能阻碍对C++编程思想的理解和掌握。

最后，有必要提一下，C语言在其最新的标准中还加入了一些新的特性，它们当中有一些C++语言并没有办法直接“兼容”，甚至未来的C++语言也不一定会考虑兼容，比如栈上分配的动态数组。

想说的就这些了，总结一下：

(1) 学习C++语言不需要以任何其他语言的学习作为基础，包括C；

(2) 不要指望学会了C++语言，就等于同时掌握了C++跟C；

(3) 也不要指望学过一遍C++语言，面向过程的编程水平就一下子可以跟上专门学习过C语言的程序员；

(4) 学好了C，对学习C++有很大的帮助，但要更好地学习C++语言并掌握其编程思想，有时需要适时地忘记并跳出C。

再浓缩一下，其实只想说一句话：C和C++是两门不同的编程语言，只是它们有较大的联系。

目 录

第1章 走进程序的世界

1.1 CPU如何工作

在了解CPU工作原理之前，先简单谈谈CPU是如何生产出来的。CPU是在特别纯净的硅材料上制造的。一个CPU芯片包含上百万个精巧的晶体管。人们在一块指甲盖大小的硅片上，用化学的方法蚀刻或光刻出晶体管。因此，从这个意义上说，CPU正是由晶体管组合而成的。简单而言，晶体管就是微型电子开关，它们是构建CPU的基石。可以把一个晶体管当作一个电灯开关，它们有个操作位，分别代表两种状态：ON(开)和OFF(关)。这一开一关就相当于晶体管的连通与断开，而这两种状态正好与二进制中的基础状态“0”和“1”对应。这样，计算机就具备了处理信息的能力。

但不要以为，只有简单的“0”和“1”两种状态的晶体管的原理很简单，其实它们的发展是经过科学家们多年的辛苦研究得来的。在晶体管之前，计算机依靠速度缓慢、低效率的真空电子管和机械开关来处理信息。后来，科研人员把两个晶体管放置到一个硅晶体中，这样便创作出第一个集成电路，再后来才有了微处理器。

看到这里，你一定想知道，晶体管是如何利用“0”和“1”这两种电子信号来运行指令和处理数据的。其实，所有电子设备都有自己的电路和开关，电子在电路中的流动或断开完全由开关来控制，如果你将开关设置为OFF，电子将停止流动，如果你再将其设置为ON，电子又会继续流动。晶体管的这种ON与OFF的切换只由电子信号控制，我们可以将晶体管称为二进制设备，这样晶体管的ON状态用“1”来表示，而OFF状态则用“0”来表示，就可以组成最简单的二进制数。众多晶体管产生的多个“1”与“0”的特殊次序和模式能代表不同的情况，可将其定义为字母、数字、颜色和图形。举个例子，十进位模式中的1在二进位模式时也是“1”，2在二进位模式时是“10”，3是“11”，4是“100”，5是“101”，6是“110”等等，依此类推，这就组成了计算机工作采用的二进制语言和数据。组成的晶体管联合起来可以存储数值，也可以进行逻辑运算和数字运算。加上石英时钟的控制，晶体管组就像一部复杂的机器那样同步地运行它们的功能。

一、CPU的内部结构

现在你已经大概知道CPU是负责些什么事情，但是具体由哪些部件负责处理数据和运行程序呢？

1. 算术逻辑单元ALU(Arithmetic Logic Unit)

ALU是运算器的核心。它是以全加器为基础，辅以由移位寄存器及相应控制逻辑组合

而成的电路，在控制信号的作用下可完成加、减、乘、除四则运算和各种逻辑运算。就像刚才提到的，这里就相当于工厂中的生产线，负责运算数据。

2. 寄存器组 RS(Register Set 或 Registers)

RS 实质上是 CPU 中暂时存放数据的地方，里面保存着那些等待处理的数据或已经处理过的数据，CPU 访问寄存器所用的时间要比访问内存的时间短。采用寄存器，可以减少 CPU 访问内存的次数，从而提高 CPU 的工作速度。但因为受到芯片面积和集成度的限制，寄存器组的容量不可能很大。寄存器组可分为专用寄存器和通用寄存器。专用寄存器的作用是固定的，分别寄存相应的数据。而通用寄存器用途广泛并可由程序员规定其用途，其数目因微处理器而异。

3. 控制单元(Control Unit)

正如工厂的物流分配部门，控制单元是整个 CPU 的指挥控制中心，由指令寄存器 IR(Instruction Register)、指令译码器 ID(Instruction Decoder)和操作控制器 OC(Operation Controller)三个部件组成，对协调整个电脑有序工作极为重要。它根据用户预先编好的程序，依次从存储器中取出各条指令放在指令寄存器中，通过指令译码(分析)确定应该进行什么操作，然后通过操作控制器，按确定的时序，向相应的部件发出微操作控制信号。操作控制器中主要包括节拍脉冲发生器、控制矩阵、时钟脉冲发生器、复位电路和启停电路等控制逻辑。

4. 总线(Bus)

就像工厂中各部位之间的联系渠道，总线实际上是一组导线，是各种公共信号线的集合，作为电脑中所有组成部分传输信息共同使用的“公路”。直接和 CPU 相连的总线可称为局部总线，包括：数据总线 DB(Data Bus)、地址总线 AB(Address Bus)、控制总线 CB(Control Bus)。其中，数据总线用来传输数据信息；地址总线用来传送 CPU 发出的地址信息；控制总线用来传送控制信号、时序信号和状态信息等。

二、CPU 的工作流程

由晶体管组成的 CPU 是处理数据和运行程序的核心，即中央处理器。首先，CPU 的内部结构可以分为控制单元、逻辑运算单元和存储单元(包括内部总线及缓冲器)三大部分。CPU 的工作原理就像一个工厂对产品的加工过程：进入工厂的原料(程序指令)，经过物资分配部门(控制单元)的调度分配，被送往生产线(逻辑运算单元)，生产出成品(处理后的数据)后，再存储在仓库(存储单元)中，最后等着拿到市场上去卖(交由应用程序使用)。在这个过程中，注意到从控制单元开始，CPU 就开始了正式的工作，中间的过程是通过逻辑运算单元来进行运算处理，交到存储单元代表工作的结束。

三、数据与指令在 CPU 中的运行

前面已经为大家介绍了 CPU 的部件及基本原理，现在，来看看数据是怎样在 CPU 中运行的。数据从输入设备流经内存，等待 CPU 的处理。这些将要处理的信息是按字节存储的，也就是以 8 位二进制数或 8 比特为 1 个单元存储。这些信息可以是数据或指令，数据可以是二进制表示的字符、数字或颜色等等，而指令告诉 CPU 对数据运行哪些操作，比如完成加法、减法或移位运算。

假设在内存中的数据是最简单的原始数据。首先，指令指针会通知 CPU，将要运行的

指令放置在内存中的存储位置。因为内存中的每个存储单元都有编号(称为地址),可以根据这些地址把数据取出,通过地址总线送到控制单元中,指令译码器从指令寄存器中拿来指令,翻译成 CPU 可以运行的形式,然后决定完成该指令需要哪些必要的操作,它将告诉算术逻辑单元(ALU)什么时候计算,告诉指令读取器什么时候获取数值,告诉指令译码器什么时候翻译指令等等。

假如数据被送往算术逻辑单元,数据将会运行指令中规定的算术运算和其他各种运算。当数据处理完毕后,将回到寄存器中,CPU 通过不同的指令将数据继续运行或者通过 DB 总线送到数据缓存器中。

基本上,CPU 就是这样去运行读出数据、处理数据和往内存写数据这三项基本工作的。但在通常情况下,一条指令可以包含按明确顺序运行的许多操作,CPU 的工作就是运行这些指令,完成一条指令后,CPU 的控制单元又将告诉指令读取器从内存中读取下一条指令来运行。这个过程不断快速地重复,快速地运行一条又一条指令,产生你在显示器上所看到的结果。我们很容易想到,在处理这么多指令和数据的同时,由于数据转移时差和 CPU 处理时差,肯定会出现混乱处理的情况。为了保证每个操作准时发生,CPU 需要一个时钟来控制 CPU 所运行的每一个动作。时钟就像一个节拍器,它不停地发出脉冲,决定 CPU 的步调和处理时间,这就是我们所熟悉的 CPU 的标称速度,也称为主频。主频数值越高,表明 CPU 的工作速度越快。

1.2 计算机语言与编译技术

一、编程语言

计算机语言的种类非常的多,总的来说可以分成机器语言、汇编语言、高级语言三大类。如果按语种分,可以分为英文符号语言和汉语符号语言(典型的如:易语言、易语言、飞扬)两类。电脑做的每一次动作,每一个步骤,都是按照已经用计算机语言编好的程序来运行的,程序是计算机要运行的指令的集合,而程序全部都是用我们所掌握的语言来编写的。所以人们要控制计算机一定要通过计算机语言向计算机发出命令。目前通用的编程语言有两种形式:汇编语言和高级语言。

1. 机器语言

机器语言是直接用二进制代码指令表达的计算机语言。指令是用 0 和 1 组成的一串代码,它们有一定的位数,并分成若干段,各段的编码表示不同的含义,例如某台计算机字长为 16 位,即由 16 个二进制数组成一条指令或其他信息。16 个 0 和 1 可组成各种排列组合,通过线路变成电信号,让计算机运行各种不同的操作。如某种计算机的指令为 1011011000000000,它表示让计算机进行一次加法操作,而指令 1011010100000000 则表示进行一次减法操作。它们的前 8 位表示操作码,而后 8 位表示地址码。从上面两条指令可以看出,它们只是在操作码中从左边第 0 位算起的第 6 和第 7 位不同。这种机型可包含 256(2 的 8 次方)个不同的指令。

特点:

机器语言或称为二进制代码语言,计算机可以直接识别,不需要进行任何翻译。每台机器的指令,其格式和代码所代表的含义都是硬性规定的,故被称为面向机器的语言,也称为

机器语言。它是第一代的计算机语言。机器语言对不同型号的计算机来说一般是不同的。

缺点：

(1) 大量繁杂琐碎的细节牵制着程序员，使他们不可能有更多的时间和精力去从事创造性的劳动，运行对他们来说更为重要的任务。如确保程序的正确性、高效性。

(2) 程序员既要驾驭程序设计的全局又要深入每一个局部直到实现的细节，即使智力超群的程序员也常常会顾此失彼，屡出差错，因而所编出的程序可靠性差且开发周期长。

(3) 由于用机器语言进行程序设计的思维和表达方式与人们的习惯大相径庭，只有经过较长时间职业训练的程序员才能胜任，使得程序设计曲高和寡。

(4) 因为它的书面形式全是“密”码，所以可读性差，不便于交流与合作。

(5) 因为它严重地依赖于具体的计算机，所以可移植性差，重用性差。

这些弊端造成当时的计算机应用未能迅速得到推广。机器语言是用二进制代码表示的计算机能直接识别和运行的一种机器指令的集合。它是计算机的设计者通过计算机的硬件结构赋予计算机的操作功能。机器语言具有灵活、直接运行和速度快等特点。不同型号的计算机其机器语言是不相通的，按照一种计算机的机器指令编制的程序，不能在另一种计算机上运行。用机器语言编写程序，编程人员要首先熟记所用计算机的全部指令代码和代码的涵义。手编程序时，程序员得自己处理每条指令和每一数据的存储分配和输入输出，还得记住编程过程中每步所使用的工作单元处在何种状态。这是一件十分繁琐的工作，编写程序花费的时间往往是实际运行时间的几十倍或几百倍。而且，编出的程序全是些 0 和 1 的指令代码，直观性差，还容易出错。现在，除了计算机生产厂家的专业人员外，绝大多数程序员已经不再去学习机器语言了。

2. 汇编语言

为了减轻使用机器语言编程的痛苦，人们进行了一种有益的改进：用一些简洁的英文字母、符号串来替代一个特定的指令的二进制串，比如，用“ADD”代表加法，“MOV”代表数据传递等等，这样一来，人们很容易读懂并理解程序在干什么，纠错及维护都变得方便了，这种程序设计语言就称为汇编语言，即第二代计算机语言。然而计算机是不认识这些符号的，这就需要一个专门的程序专门负责将这些符号翻译成二进制数的机器语言，这种翻译程序被称为汇编程序。汇编语言的实质和机器语言是相同的，都是直接对硬件操作，只不过指令采用了英文缩写的标识符，更容易识别和记忆。它同样需要编程者将每一步具体的操作用命令的形式写出来。汇编程序通常由三部分组成：指令、伪指令和宏指令。汇编程序的每一句指令只能对应实际操作过程中的一个很细微的动作，例如移动、自增，因此汇编源程序一般比较冗长、复杂、容易出错，而且使用汇编语言编程需要有更多的计算机专业知识。但汇编语言的优点也是显而易见的，用汇编语言所能完成的操作不是一般高级语言所能实现的，而且源程序经汇编生成的可执行文件不仅比较小，运行速度也很快。

汇编语言同样十分依赖于机器硬件，移植性不好，但效率仍十分高，针对计算机特定硬件而编制的汇编语言程序，能准确发挥计算机硬件的功能和特长，程序精炼且质量高，所以至今仍是一种常用而强有力的软件开发工具。

3. 高级语言

从最初与计算机交流的痛苦经历中，人们意识到，应该设计这样一种语言：接近于数学语言或人的自然语言，同时又不依赖于计算机硬件，编出的程序能在所有机器上通用。经过

努力，1954 年，第一个完全脱离机器硬件的高级语言——FORTRAN 问世了。之后的多年，共有几百种高级语言出现，有重要意义的有几十种，影响较大、使用较普遍的有 FORTRAN、ALGOL、COBOL、BASIC、LISP、SNOBOL、PL/1、Pascal、C、PROLOG、Ada、C++、VC、VB、Java 等。

二、目前流行的主流编程语言简介

如果你想学习编程，却又不知从何入手，使用什么工具，那么不妨看看下面的几种学习方案，可能会给你一些启示吧。

1. C++语言

C++，应用最广泛、成熟、强大、复杂、流行的程序设计语言。你目前正在使用的 Windows 或 Linux 操作系统的大部分内容均出自 C++的程序员之手，C++非常强大近乎无所不能，而 C++代码经过编译后将成为计算机的二进制代码的可执行程序，所以在兼容性、性能上均为优秀。当今如果你学习 C++，再配合 WIN32 SDK、MFC 或是. NET CLR，绝对是开发桌面程序的首选。从名字可以看出，C++改进自 C 语言，特别是在面向对象方面的扩展，但学习 C++不需要且不建议先学习 C 语言基础，可以把C++当作一门全新的编程语言。

如在 Windows 平台下，C++首选的集成开发环境(IDE)自然是 Visual C++，它包含在 Microsoft Visual Studio 之中，最新版本是 2010 版。你可以选择微软官方提供的 Visual Studio 2010 专业版，这是最好的选择！微软提供 90 天的试用版，安装镜像较为庞大，与正式版没有区别，同时包含了 MSDN Library(最权威的开发文档)，唯独只是授权序号的区别，下载安装后即可使用。当然，你也可以进行简单的操作将其变为正式版。不过在此不是鼓励大家使用盗版软件，如在非商用的情况下，这种手段是可行的，也会加快你的学习进度与质量。此外微软也提供了 Visual Studio 2010 速成版，又称为学生版，它是免费的，如仅是学习的话完全够用。

除了微软的 IDE，你还有其他的选择。在 Windows 下搭建 Eclipse+CDT+MinGW，Linux 和 Unix 下可以使用 Eclipse+GCC++，这些都是开源、免费的。或者，更基本的你需要一个记事本与 C++编译器。

优点：

(1) C 语言灵活性好，效率高，可以接触到软件开发比较底层的东西。

(2) 微软的 MFC 库博大精深，学会它后可以随心所欲地进行编程。

(3) VC 是微软制作的产品，与操作系统的结合更加紧密。

缺点：

对使用者的要求比较高，既要具备丰富的 C 语言编程经验，又要具有一定的 Windows 编程基础，它的过于专业使得一般的编程爱好者学习起来会有不小的困难。

综述：VC 是程序员用的东西。如果你是一个永不满足的人，而且可以在编程上投入很大的精力和时间，那么学习 VC 你一定不会后悔的。

2. Java 语言

Java，面向对象、安全、跨平台、强大稳健、流行的程序设计语言与环境，由 Sun 公司开发，目前由 Java Community Process 控制。Java 近些年来非常流行且稳定，未来生命周期较长。它的语言风格较为接近 C++与 C#，而最为人熟知的便是其跨平台性。Java 的跨平

台性已得到了广泛的认可，在计算机的各种平台、操作系统，以及手机、移动设备、智能卡、消费家电中均已迈入成熟的生产化，且国内的 Java 人才需求前景目前也是大好。

Java 的官方网站是 http：//java. sun. com/。Java 分为 JavaSE(标准版，面向初学者与桌面开发)、JavaEE(企业版，也称 J2EE，面向企业级开发、网络开发，包括了为人熟知的 JSP，并包含了 JavaSE 的所有内容)、JavaME(微型版，用于手机、PDF、机顶盒、消费家电等嵌入设备开发)，目前还有正在发展的 JavaFX(一种富 Internet 应用程序开发的脚本语言)。Java 的运行环境是 JRE，开发环境是 JDK，均可以在其官方站点下载。开发平台的构建较为简单，开发者下载并安装 JDK 即可，目前 JDK 最新版本是 1. 6. 15。最重要的，它们都是免费、开源的。而 Java 不需要指定集成开发环境(IDE)，JDK 和记事本足以完成。不过在此推荐使用 Notepad++或 UltraEdit 作为代码文本编辑器，并使用 Eclipse 与 NetBeans 这两款免费而又强大成熟的 Java 集成开发环境(IDE)。

Java 提供了一个功能强大语言的所有功能，但几乎没有一点含混特征。C++安全性不好，但 C 和 C++被大家接受，所以 Java 设计成 C++形式，让大家很容易学习。

Java 去掉了 C++语言的许多功能，让 Java 的语言功能很精炼，并增加了一些很有用的功能，如自动收集碎片。

Java 去掉了以下几个 C 和 C++功能：

(1) 指针运算

(2) 结构

(3) typedefs

(4) #define

(5) 需要释放内存

这将减少平常出错的 50%。而且，Java 很小，整个解释器只需 215K 的 RAM。

另外，Java 实现了 C++的基本面向对象技术并有一些增强(为了语言简单，删除了一些功能)。Java 处理数据方式和用对象接口处理对象数据方式一样。

3. C#语言

C#，读作 C Sharp。微软的 C#就好似是 C++、Java、Delphi 与 Visual Basic 的结合体，是新兴、易学、强大的程序设计语言，它更像 Java 完全面向对象，开发与运行都在 . NET Framework 环境中。使用微软强大的 Visual Studio 集成开发环境，这是快速开发 Windows 平台桌面应用程序的最好选择。听起来有点像 VB? 不过 C#比 VB 可强大多了。但 C#编译后的程序如 Java 一样是中间语言，运行程序的计算机需要安装 . NET Framework 运行环境，该环境不算大也不算小，但在微软的 . NET 推出多年后的今天，它还没有较好的普及开，可能对你的程序传播会有一定影响。同时 C#也可以开发 ASP. NET 的动态网页程序，这是曾经风光多年的 ASP 的替代产品。开发 C#程序，使用微软的 Visual Studio 是最好的，也是几乎唯一的选择。同 Visual C++的环境搭建基本雷同，在此不再复述。目前 . NET 环境已经发展到 3. 5，C#已经发展到 C#4. 0，学习 C#请选择一本实时性、专业性、全面性的好教程。微软的 Visual Studio 2010 近期也将发布，届时将搭载更为强大的 C#语言与集成开发环境。

4. Ruby 语言

Ruby，较 Python 来说更新兴的面向对象脚本语言，由日本人开发。实质上它与 Py-

thon 是同类，具有共同的一些特性、脚本语言，面向对象、免费开源、简洁强大、跨平台性，所以在此就不多作介绍。Python 与 Ruby 都是目前的新兴流行脚本语言，不过它们作为两款出生与成长不同的孩子，终究不同，选择哪一个还得取决于你的应用需要。可在 Ruby 的官方中文站点获取更多信息，下载与安装 Ruby 开发与运行环境。

5. Perl 语言

Perl，最成熟、最具灵活性的脚本语言。学习起来较为容易，但却非常强大。Python 语言的许多特性便是有借鉴自 Perl。Perl 拥有一个大型的第三方代码库 CPAN，极大地方便了程序人员的开发和使用。在官方网站下载 Perl 环境，所有的 Linux 系统都几乎集成了 Perl。脚本语言的开发都推荐在 Notepad＋＋等高级文本编辑器中完成。

6. Pascal 语言

Pascal，曾经风靡全球的高级程序设计语言，特点是语言简明、结构严谨、灵活性较高，名气不下于 C＋＋。前几年流行的 Delphi（一种 Windows 应用程序快速集成开发环境）使用的便是 Pascal 语言，它也是国际信息学奥林匹克竞赛、中国信息技术奥林匹克竞赛的编程项目主要语言。如今 Pascal 语言在实际应用领域已逐渐被人舍弃，但依然是非常适合初学者的编程语言。你可以下载使用 Free Pascal 这个目前最好的 Pascal 编译环境，也可以使用 Delphi。

7. Basic 系列语言

Basic，或称为 Basic 系列语言，主要有 QuickBasic、QBasic、VBScript、Visual Basic、Visual Basic. NET，甚至有适于儿童学习的 Small Basic。如今 PC（个人计算机）中流行的 Basic 语言，正是鼎鼎大名的比尔·盖茨先生编写的，但最初的 Basic 语言来自上世纪 60 年代两位美国计算机科学家。

上世纪末期与本世纪初期，Visual Basic 作为一款简单易学的 Windows 应用程序开发环境，曾在中国被大量学习使用，现在你在书店还可以看到许多 Visual Basic 考试书籍与题目的身影，不过这些已经是早已淘汰的 VB 6 了，我国的计算机等级考试大纲一直都存在严重问题（这是我的想法）。当时被流行的 VB 6 让很多人快速地迈入了程序员的队伍，但 VB 6 因为并不强大，并不能有效开发大型程序。VB 的成功并不在于改进后的 Basic 语言本身，而应是优秀的集成开发环境与开发 Windows 程序的便捷特性。

微软公司在 2002 年推出了 Visual Basic . NET，正式推出 . NET 环境并让 VB 作为首批成员。如今的 Visual Basic 已经非常强大，但已完全地依赖于 . NET Framework 环境，可以开发 ASP. NET 程序。而且 VB 所有优秀特点均已被 C＃吸收，它们如今的差别并不大。我更倾向推荐有意学习 VB 的朋友去学习 C＃。

程序语言并不适合于人，而适用于所要开发的领域或软件的需求。不能光从名称看出某一门编程语言的含义，也不能跟风别人学什么我就学什么。而在于你的兴趣，或者你要涉及的领域、开发的程序。如果你要开发桌面程序，那就可以学习 C＋＋、C＃、VB、Java；如果你要开发动态网页，就可以学习 C＃、VB、Java；如果你要开发手机程序，就可以选择 C＋＋、Java 或 C＃。

本书讲解的是 C/C＋＋语言，所以先来看看 C 和 C＋＋的历史。

1.3 C和C++的历史

一、C语言的历史

C语言是国际上广泛流行的、很有发展前途的计算机高级语言。它适合作为系统描述语言，即可用来编写系统软件，也可用来编写应用软件。

早期的操作系统等系统软件主要是用汇编语言编写的(包括UNIX操作系统在内)。由于汇编语言依赖于计算机硬件，程序的可读性和可移植性都比较差。为了提高可读性和可移植性，最好改用高级语言，但一般的高级语言难以实现汇编语言的某些功能(汇编语言可以直接对硬件进行操作，例如对内存地址的操作、位操作等)。人们设想能否找到一种既具有一般高级语言特性，又具有低级语言特性，集它们的优点于一身的语言。于是，C语言就在这种情况下应运而生了。

C语言是在B语言的基础上发展起来的，它的根源可以追溯到ALGOL 60。1960年出现的ALGOL 60是一种面向问题的高级语言，它离硬件比较远，不宜用来编写系统程序。1963年英国的剑桥大学推出了CPL(Combined Programming Language)语言。CPL语言在ALGOL 60的基础上接近了硬件一些，但规模比较大，难以实现。1967年英国剑桥大学的Matin Richards对CPL语言作了简化，推出了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，又作了进一步简化，设计出了很简单的而且很接近硬件的B语言(取BCPL的第一个字母)，并用B语言写了第一个UNIX操作系统，在PDP-7上实现。1971年在PDP-11/20上实现了B语言，并写了UNIX操作系统。但B语言过于简单，功能有限。1972年至1973年间，贝尔实验室的D. M. Ritchie在B语言的基础上设计出了C语言(取BCPL的第二个字母)。C语言既保持了BCPL和B语言的优点(精练、接近硬件)，又克服了它们的缺点(过于简单、数据无类型等)。最初的C语言只是为描述和实现UNIX操作系统提供一种工作语言而设计的。1973年，K. Thompson和D. M. Ritchie两人合作把UNIX的90%以上用C改写(UNIX第5版。原来的UNIX操作系统是1969年由美国的贝尔实验室的K. Thompson和D. M. Ritchie开发成功的，是用汇编语言写的)。

后来，C语言作了多次改进，但主要还是在贝尔实验室内部使用。直到1975年UNIX第6版公布后，C语言的突出优点才引起人们普遍注意。1977年出现了不依赖于具体机器的C语言编译文本《可移植C语言编译程序》，使C移植到其他机器时所做的工作大大简化了，这也推动了UNIX操作系统迅速地在各种机器上实现。例如，VAX、AT&T等计算机系统都相继开发了UNIX。随着UNIX的日益广泛使用，C语言也迅速得到推广。C语言和UNIX可以说是一对孪生兄弟，在发展过程中相辅相成。1978年以后，C语言已先后移植到大、中、小、微型机上，已独立于UNIX和PDP了。现在C语言已风靡全世界，成为世界上应用最广泛的几种计算机语言之一。

二、C++语言的历史

语言的发展是一个逐步递进的过程，C++是直接从C语言发展过来的，而C语言是从B语言发展过来的，B语言是BCPL的一个解释性后代，BCPL则是Basic CPL。其中最有