挑战 C++程序语言

蔡文辉　黄国峰　张真诚　著

机械工业出版社

C++程序语言是目前最受欢迎的面向对象程序语言之一，不但保留了原来 C 语言的许多优点，另外加入了面向对象所需的一些机制。本书主要以简单的实例来说明 C++语言的结构、数据类型的种类、流程的控制方法、面向对象程序设计方法、对象的继承机制、运算符的重载以及多态性等，而且还介绍了软件工程的知识，使读者对于如何开发大型的软件系统能有一个清晰的概念。另外本书提供了完整的范例程序，供读者学习参考之用。

本书通俗易懂，理论与实践紧密结合，适用于作为大专院校学生 C++语言课程的教材，也可作为广大计算机爱好者学习 C++的参考书。

图书在版编目（CIP）数据

挑战 C++程序语言/蔡文辉等著. —北京：机械工业出版社，2002.8

ISBN 7-111-10776-4

Ⅰ. 挑… Ⅱ. 蔡… Ⅲ. C 语言-程序设计 Ⅳ. TP312

中国版本图书 CIP 数据核字（2002）第 061214 号

机械工业出版社（北京市百万庄大街 22 号 邮政编码 100037）

责任编辑：朱英彪

北京机工印刷厂印刷・新华书店北京发行所发行

2002 年 9 月第 1 版・第 次印刷

787×960 1/16・ 20 印张・ 459 千字

0001-5000 册

定价：31.00 元

凡购本图书，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）68993821、68326677-2527

推　荐　序

本书是针对 C++程序设计初学者而写的一本好书，从最基本的 C++语法，到完整的面向对象程序设计方法，全部都用实例来说明，讲得非常清楚，对于每一位没有学过程序设计的初学者都可以很容易看懂。

尤其本书列举了许多流程图范例，而且对于每一个流程图都附有对应的 C++程序，可提供给读者作为设计程序时的参考。对于不曾使用计算机程序来解决问题的人来说，流程图无疑是最佳的学习途径，而市面上介绍程序设计的书，多半忽略了这一既简单又有效率的工具，由此可见作者对于本书编著的用心。

如果你企盼成为新时代最有效率的程序设计员，本书亦针对如何开发大型系统有简单扼要的介绍，在熟悉了 C++语言之后，对于如何进入专业的系统开发，本书亦提供了一个良好的桥梁。

整体而言，对于有志从事软件开发的人来说，本书绝对是一本值得极力推荐的好书。

李家同

2002 年 1 月

序

随着计算机制造技术的突飞猛进，计算机价格已经可以被一般的社会大众所接受。再加上计算机网络应用的日益普及，这使得使用者对于各类应用软件的需求日益增加。然而，传统的软件开发方法在许多大型系统上的应用已经出现了缓不济急的情况。而面向对象的软件开发观念，强调软件组件的再使用，可以让软件的开发像机械设备的组装一样容易，因此，近年来已倍受软件开发人员的重视，并且也已实际应用在许多成功的案例上。因此，面向对象程序设计的观念俨然已经成为软件专业人员所必备的知识了。

C++程序语言无疑是目前最受欢迎的面向对象程序语言之一，主要原因是 C++是一个源自近二十年来最成功的程序语言——C 语言。C++不但保留了原来 C 语言的诸多优点，另外还加入了因应面向对象所需的一些机制。然而，本书内容已经涵盖了大部分 C 程序语言的语法说明，因此，即使读者不具备 C 语言的基础也可以轻松地从本书的内容中学会 C++程序语言。本书的编排内容，主要涵盖了计算机硬件的简介、流程图的绘制方法、C++语言的结构、数据类型的种类、流程的控制方法、面向对象的观念介绍、对象的继承机制、运算符的重载以及多态性等，而最后一章则是软件工程的简介。相信读者在阅读完本书的内容，并且亲自练习过每一章的习题之后，不仅可以具备利用 C++语言来开发软件的基本能力，而且对于发展大型软件系统也将具有基本的认识。

本书适合作为高校 1~2 学期的 C++程序设计教材。由于本书内容的描述主要采用简单的范例作为抽象语法的使用说明，因此对于作为自修参考之用亦极为合适。

本书编排虽力求完美，但难免仍有少许谬误之处，欢迎专家、学者予以指教。

蔡文辉　黄国峰　张真诚

2002 年 1 月

C++程序语言

1 程序设计基本概念

1.1 计算机的过去与现在

1930 年末，Atanasoff 与 Berry 博士于 Iowa 州立大学创造了第一台电子计算器，主要设计用于核物理与数学的相关计算。1946 年，美国陆军与宾州大学合作开发出第一台通用型计算机，并被广泛用于天气预测、核能计算等。早期的计算机采用真空管为主要器件，但自从晶体管与大规模集成电路技术应用到计算机之后，计算机硬件的发展速度可说是一日千里。目前使用的个人计算机，其计算能力（每秒可完成数十亿个指令）已经比早期的大型计算机高出数百倍，而且价格也大幅降低。高性能、低价位的计算机在大多数国家已经像电视机一样普及，再加上计算机网络的蓬勃发展，计算机科技已经彻底改变了人类的生活方式。

目前常见的计算机，主要是以外观尺寸与性能来分类（参考图 1.1 所示）的。大型机（Supercomputer、Mainframe or Minicomputer）主要应用于需要高速运算的商业用途和学术实验室。其他比较偏向于个人使用的有工作站（Workstation）、个人计算机（Personal Computer）、笔记本电脑（Notebook or Laptop Computer）、平板式计算机（Tablet Computer）和掌上电脑（Palmtop Computer）等。

然而，如果没有适当的计算机软件来控制计算机硬件进行正确的工作，则不管硬件本身拥有多么高速的运算能力，它都将形同废铁一般。因此，如果视计算机硬件为骨架，则软件就是计算机的灵魂。由此可见，计算机软件的开发技术将会是本世纪众所瞩目的焦点，程序设计是软件开发的最基本要素，学会程序设计就等于拥有了搭上软件开发列车的一张车票。

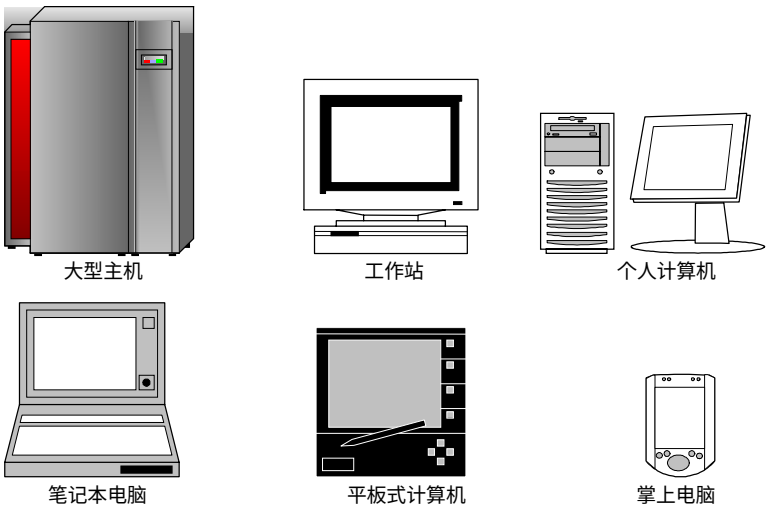


图 1.1　常见的计算机种类

1.2　计算机硬件结构

目前常见的计算机系统，其硬件基本包括下列各种组件：

1．主存储器（Main Memory）。

2．次存储器（Secondary Memory），如硬盘驱动器、软盘驱动器、磁带机、CD-ROM 等。

3．中央处理器（Central Processing Unit，CPU），负责最主要的算术与逻辑运算工作。

4．输入设备（Input Devices），如键盘、鼠标、扫描仪、数字化仪等。

5．输出设备（Output Devices），如监视器、打印机等。

图 1.2 是上述计算机硬件的架构示意图。计算机网络是指在计算机与计算机之间通过联机的方式，让彼此可以通过通信协议（Protocol）互相传递信息。所谓局域网（Local Area Network，LAN）指在有限区域（如同一建筑物）内的计算机设备所连成的网络（如图 1.3 所示）。在 LAN 中常见的网络架构为以太网（Ethernet）。近

年来，无线局域网（Wireless LAN）已逐渐普及，但相对于传统的有线网络，其通信频宽较为有限。将数个 LAN 通过卫星或越洋缆线连接在一起形成所谓的广域网（Wide Area Network，WAN，如图 1.4 所示），目前使用最普遍的广域网为因特网（Internet），其中所使用的通信协议为 TCP/IP。

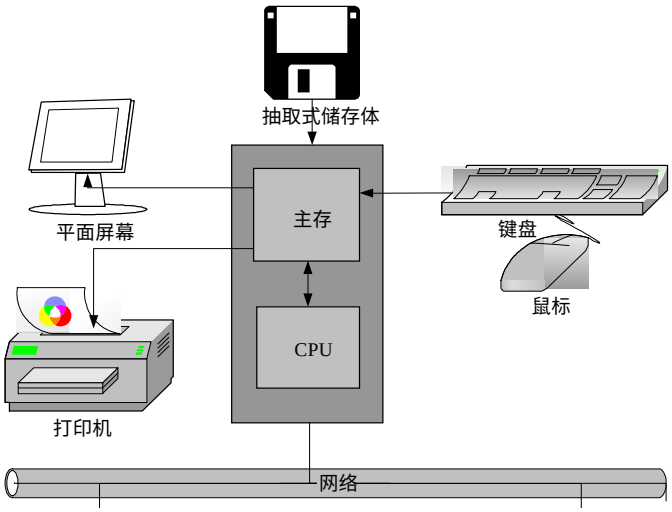


图 1.2　计算机硬件的架构示意

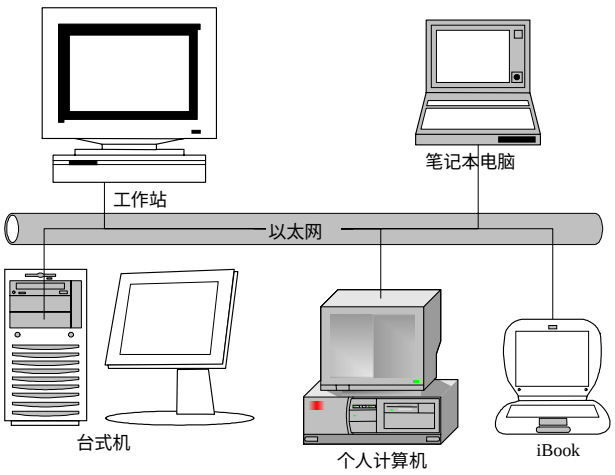


图 1.3　局域网

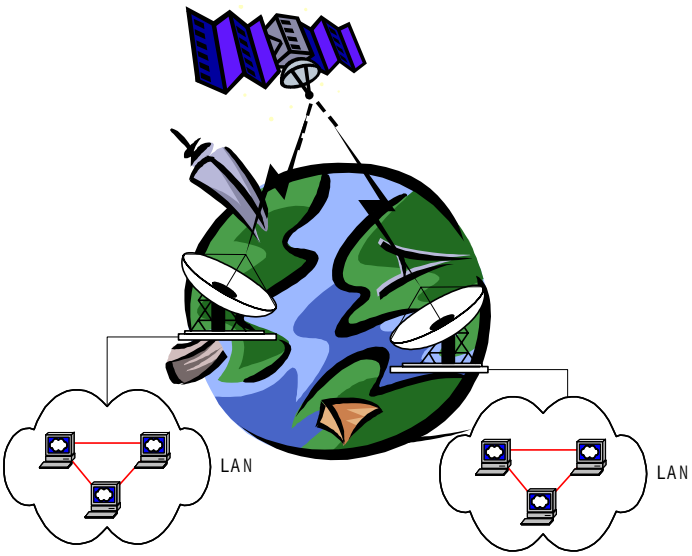


图 1.4　广域网

1.3　计算机软件

在第一节中已经说过，如果计算机没有了软件，即使拥有再强大计算能力的硬件也将是英雄无用武之地，可见软件在计算机应用领域中是何等的重要。事实上，计算机软件是由一整套完整的程序所构成的。而所谓的程序（Program）是指一组指示计算机硬件执行特定工作的指令。计算机软件主要针对某一类型问题的需求而开发，因此计算机软件按其功能可以简单地区分为系统软件和应用软件。

1.3.1　操作系统

系统程序主要处理使用者与计算机硬件之间的接口工作，例如打印机驱动程序、键盘驱动程序等。而操作系统（Operating System）则是整合必备的系统程序，

提供一个工作平台，让使用者及程序设计人员有一个更便捷的工作环境。常见的操作系统包括 Windows 98/2000/XP、Linux、Unix、Macintosh OS 等。

操作系统主要负责的工作项目如下：

1．用户与计算机之间的沟通，例如接受使用者的指令加载某一程序并执行。

2．内存、中央处理器、打印机等硬件资源的使用管理，例如程序占用 CPU 的时间分配。

3．从键盘、鼠标等输入设备接受信息，并将其传给正在执行的程序。

4．将程序执行的结果，正确输出到指定的输出设备，如屏幕、打印机等。

5．储存设备（如硬式磁盘驱动器、磁带机）的管理使用。

1.3.2 应用软件

应用软件主要用来协助使用者完成某项特定的工作。例如，文字处理软件主要用来处理文件的排版工作，常见的有 Word、WordPerfect、LaTex 等。而像 Ms-Excel 和 Lotus 1-2-3 等电子表格软件，则主要用来处理统计资料的分析、计算与图表的绘制等。其他常见的应用软件包括简报软件、图（音）像处理软件、绘图软件、数据库管理系统以及程序开发工具软件等。一般而言，这些应用软件是依附在某一特定的操作系统上的。举例来说，在 Windows 操作系统上的软件无法直接移到 Linux 上运行，然而若程序语言本身具有良好的可移植性（Portable），则可以将源程序使用适当的编译程序重新编译后即可在不同的操作系统上执行。下一小节将简单介绍各种常见的程序语言。

1.3.3 程序语言

其实计算机只识别一种程序语言，即机器语言（Machine Language）。不同的 CPU 会有不同的机器语言，如果计算机软件采用机器语言来开发，则该软件将只能在特定的硬件上执行。因此，不具有移植性是其主要的缺点之一。除此之外，因为机器语言纯粹以数字来表示，所以并不适合人类阅读，当然用机器语言来开发程序

将是非常困难的一件事。

汇编语言（Assembly Language）是稍具可读性的程序语言，采用符号式的指令来代表指令，但仍然对硬件具有相当高的依赖性，因此移植性不佳。具有较佳移植性的程序语言，也是目前为大多数程序设计员所采用的程序语言并被归类为高级程序语言（High-Level Language），例如 Pascal、C/C++、Fortran、COBOL、Lisp 等，这些程序语言采用较接近自然语言的方式来描述程序中的逻辑与流程控制。使用高级程序语言所编写的程序代码称为源程序代码（Source Code），它必须经过特定的编译器（Compiler）翻译成机器语言，才使得特定的计算机硬件可以正确地执行程序指令。

如果在不同的机器上，均有针对该特定程序语言所设计的编译器，则同样的源程序可在不同的机器上执行，这样也使得该程序语言具有较佳的移植性。

本书主要介绍 C++程序语言的设计技巧，而常用的 C++语言编译器包括 GNU C++ Compiler、Borland C++ Compiler 及 Microsoft Visual C++。其中后两者为商业软件，而 GNU C++ Compiler 则为自由软件，程序设计者可以自行到 GNU 网站下载相关的编译程序，GNU 的网址为 http://www.gnu.org。另外，Borland C++ Compiler 也针对其编译器部分（不包含整合开发环境）提供给程序设计者下载使用，请参考该公司网站 http://www.borland. com/。有关编译器的使用流程，请参考本书第 3 章图 3.1 的说明。

1.4　程序设计方法

程序设计是一种解决问题的行为，简单地说就是利用计算机来处理现实生活中所遇到的问题。软件工程是一门专门的学科，主要在于探讨如何开发与管理比较大型且复杂的软件系统，在本书的 16 章中将会有软件工程的简介。然而，无论是多么复杂的系统，都可以将问题分为数个较小的部分，再分别加以解决。程序分析员的主要工作就是将整个系统的问题理清，并写出系统的规格与需求，而程序设计员则负责将用来解决这些问题的计算机程序编写出来。下面将介绍如何从分析问题、设计程序到最后编写程序的一套简单的方法，相信这对于程序设计初学者会有相当大的帮助。

程序设计的过程，主要分为下列几个项目：

1．问题定义

2．问题分析

3．算法设计

4．程序编写

5．程序测试

6．程序维护及升级

接下来，将针对每一个项目的工作内容做一个简单扼要的说明，之后再以一个实际的例子来补充说明这些工作项目的意义。

问题定义的主要用意在于明确定义出所要解决的问题到底是什么，必须避免其中含有涵义不清的语句。通常问题的定义是由对该问题有相当了解的专家所描述的，而程序设计者必须对其叙述进行深入的了解。

问题分析主要针对解决问题时需要输入哪些信息和输出结果的规范。另外，对于解决问题时所需的假设与限制也必须明确地在这个阶段加以制定。

算法设计主要是设计如何解决问题的详细步骤，这一个阶段通常比较困难。原则上，当问题本身稍微复杂时，建议在设计算法的一开始，并不需要刻意地将所有的细节都写出来，可以采取由上而下（Top-Down Design）的方法将问题细分成许多独立的小问题，接着再针对每一个小问题设计出算法来。最后，可以进一步改进（Refinement）算法，使其更具效能或修正错误。值得一提的是，在设计算法时可以采用流程图（Flowchart）来表示计算的方法与流程。流程图是一个相当实用的工具，本书第 2 章将介绍流程图中所使用的各种符号的意义，并且举出许多具有代表性的范例来说明流程图的绘制方法。

程序编写阶段主要是将上一步骤中所设计的算法，以程序语言实际编写出来。此阶段的工作内容可以视为将设计完成的算法转换成使用程序语言来表示。

程序测试阶段主要用来验证程序的正确性。对于比较复杂的程序，多半需要有专人进行全面测试，以避免因程序设计者本身的盲点而导致无法确保程序的完整性与正确性。

程序维护阶段通常是为了因新的需求而必须对已完成的程序进行适当的修改。除此之外，这一阶段的目的也是为了修正上一个步骤所未发现的错误。

范例：公里与英里的转换

问题定义：假设正在设计一个用来分析汽车油耗的程序，然而不同国家所惯用

的单位会有所不同，因此在这个程序中需要一个能够转换公里与英里的程序。

问题分析：在刚开始分析此问题时，必须先理清到底是要将公里数转换成英里数，还是将英里数转换成公里数，如果两者都需要，则必须有一个分辨的机制。例如设计两个程序，一个用来将公里数转换成英里数，另一个用来将英里数转换成公里数；或者，也可以只设计一个程序，但其中必须用一个额外的参数来指明要做哪一种转换。关于将公里数转换为英里数的计算，所需输入的数据为公里数（Kms），而输出为英里数（Miles），两者的换算公式为：一公里等于 0.6215 英里，即一英里等于 1.609 公里，两者均须使用浮点数数据类型来储存。

算法设计　根据上一个步骤的分析，算法的设计如下：

步骤一　输入公里数，存入变量 Kms。

步骤二　代入换算公式，计算出对应的英里数，将结果存入变量 Miles。

步骤三　将结果输出。

程序编写：在这个阶段，可以依据上一阶段所设计的算法，采用事先选用的程序语言编写，如下所示。

【公里转换英里程序范例】

```
// 程序功能：转换公里数为英里数
#define Km2Miles （0.6215）

double Km_To_Miles(double Kms) //在此输入公里数
{
   double Miles;

   Miles = Kms * Km2Miles; //公里数乘上 0.6215 等于英里数

   return Miles; //返回计算结果
}
```

程序测试：如果编写的程序含有语法错误（Syntax Error），可以程序语言的编译器（Compiler）进行检测。因此，程序测试阶段主要是查找程序中的语意错误（Semantic Error），如误将公里数乘以 1.609 以为是对应的英里数。针对上一步骤编写的程序，可以利用下列程序来测试其正确性。

【公里转换英里程序测试范例】

```
// 程序功能：转换公里数为英里数
#include <iostream.h>
double Km_To_Miles(double Kms);

int main(void)
{
  double Kms;

  cout <<"输入公里数:";
  //从标准输入设备取得公里数，并将其记录在变量 Kms 中
  cin >> Kms;

  // 输出转换结果
  printf("%5.2f 公里等于%5.2f 英里\n", Kms, Km_To_Miles(Kms));
  return 0;
}
```

【执行结果】

```
输入公里数：100
100.00 公里等于 62.15 英里
```

关于上述范例程序的语法及其意义，本书第 3 章之后将会有详细的说明。

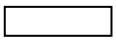
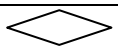
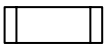
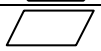
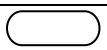
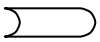
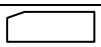
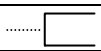
C++ 程序语言

2 流 程 图

2.1 简　　介

流程图（Flow Chart）主要用来描述解决问题的方法与步骤，已被广泛应用于许多不同的领域。常用的流程图符号及其意义如表 2-1 所示。

表 2-1　常用的流程图符号

符　　号	用　　途
	处理
	比较
	已设定的子程序
	准备符号
	输入或输出
	起点或终点
	连接符号
	流向符号
	磁盘或在线储存
	磁带
	人工输入
	报表纸
	接下一页
	屏幕
	卡片
	说明

在日常生活中，如果要说明某件事情的处理步骤，可以利用适当的流程图符号连接，有系统地逐步进行描述。例如，如图 2.1 所示就是从饮料自动售货机购买可