信息科学与技术丛书

安国双　编著

C++面向对象程序设计

- 将作者多年的开发经验以章节的形式汇集而成
- 内容的叙述采用归纳和推理的方式进行，使得对知识的理解能逐步深入
- 书中的样例代码均来自实际开发环境中
- 提出很多以往不容易想到或接触到的观点，具有启发意义

机械工业出版社
CHINA MACHINE PRESS

信息科学与技术丛书

C++面向对象程序设计

安国双　编著

机械工业出版社

本书从学习和使用并重的角度叙述了面向对象的编程方法，包括C++基础、面向对象编程思想、类型以及类层次结构4个主要部分，涵盖了类型分析、类对象、运算符重载、内存布局、类型转换和虚拟机制等面向对象的编程知识。书中的代码示例都是经过作者精心选择和设计的，可以为读者带来实际工作中的第一手资料。通过书中具体内容的学习，读者可以在短时间内快速提高自己的编程能力。

适用本书的读者为学习过编程语言的在校大学生，有编程基础的软件工程师，从C语言向C++语言转型的软件工程师，想集中学习面向对象知识的软件工程师，以及其他C++编程爱好者。

图书在版编目（CIP）数据

C++面向对象程序设计/安国双编著. —北京：机械工业出版社，2011.7
（信息科学与技术丛书）
ISBN 978-7-111-35527-4

Ⅰ. ①C… Ⅱ. ①安… Ⅲ. ①C语言－程序设计 Ⅳ. ①TP312

中国版本图书馆CIP数据核字（2011）第155714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：郝建伟
责任印制：乔 宇

三河市宏达印刷有限公司印刷

2011年9月第1版·第1次印刷
184mm×260mm·21.5印张·527千字
0001-3500册
标准书号：ISBN 978-7-111-35527-4
定价：49.00元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

电话服务
社服务中心：（010）88361066
销售一部：（010）68326294
销售二部：（010）88379649
读者购书热线：（010）88379203

网络服务
门户网：http://www.cmpbook.com
教材网：http://www.cmpedu.com

出版说明

随着信息科学与技术的迅速发展，人类每时每刻都会面对层出不穷的新技术和新概念。毫无疑问，在节奏越来越快的工作和生活中，人们需要通过阅读和学习大量信息丰富、具备实践指导意义的图书来获取新知识和新技能，从而不断提高自身素质，紧跟信息化时代发展的步伐。

众所周知，在计算机硬件方面，高性价比的解决方案和新型技术的应用一直备受青睐；在软件技术方面，随着计算机软件的规模和复杂性与日俱增，软件技术不断地受到挑战，人们一直在为寻求更先进的软件技术而奋斗不止。目前，计算机在社会生活中日益普及，随着Internet延伸到人类世界的方方面面，掌握计算机网络技术和理论已成为大众的文化需求。由于信息科学与技术在电工、电子、通信、工业控制、智能建筑、工业产品设计与制造等专业领域中已经得到充分、广泛的应用，所以这些专业领域中的研究人员和工程技术人员越来越迫切需要汲取自身领域信息化所带来的新理念和新方法。

针对人们了解和掌握新知识、新技能的热切期待，以及由此促成的人们对语言简洁、内容充实、融合实践经验的图书迫切需要的现状，机械工业出版社适时推出了“信息科学与技术丛书”。这套丛书涉及计算机软件、硬件、网络和工程应用等内容，注重理论与实践的结合，内容实用、层次分明、语言流畅，是信息科学与技术领域专业人员不可或缺的参考书。

目前，信息科学与技术的发展可谓一日千里，机械工业出版社欢迎从事信息技术方面工作的科研人员、工程技术人员积极参与我们的工作，为推进我国的信息化建设做出贡献。

机械工业出版社

前 言

C++语言是最早支持面向对象程序开发和设计的语言，自从诞生之日起，它就受到了广大软件开发和设计者的普遍认可，并成为众多操作系统以及各种系统软件的首选开发工具。现在C++语言在这些领域应用更为广泛和更加深入，这也说明了C++语言的重要程度和它顽强的生命力。因此，我们有必要尽力去学习并掌握C++语言。

C++语言是一门相对复杂和精深的语言，这意味着在使用这门语言获得益处的同时，也要为掌握这门语言付出艰辛的努力。因此，我们需要一种刻苦钻研的精神和勇于实践的学习态度，这主要体现在两个方面：一是对C++语言知识的学习，另一方面是来源于具体的工作经验。如果能将二者结合起来，那对我们掌握这门语言无疑大有益处。这本书就是基于这样的出发点编写的，是作者将多年的开发经验以章节的形式汇集而成的。目的是希望能为读者提供实际开发工作中的第一手资料，让读者能从实际工作的角度去学习C++语言。

本书所介绍的内容来自于实际工作当中，对于有C++语言基础但没有实际工作经验的读者来说，能体会到书本中学习到的知识和实际工作中得来的知识之间的区别，这种区别使得读者能够将所学知识从实际应用的角度去看待，这是对实际工作的提前适应过程，这是有益处的。对于有C++开发和设计经验的读者来说，书中的许多内容都会是熟悉的和需要的，也会感觉到这些知识是适合于有经验的读者的。

在内容叙述上，本书具有如下特点：一是本书对内容的叙述采用归纳和推理的方式进行，使得对知识的理解能逐步深入；二是书中的样例代码都来自实际开发环境中，对实际的开发工作有很好的参考意义；三是书中的很多观点都是以往不容易想到或接触到的，会有一些启发意义；四是本书的内容是全面的，C++程序设计的各个主要知识点在书中都能找到。

在内容组织上，本书共12章，从整体上又可划分为四部分。第一部分包括第1章，用一定的篇幅对语言的基础知识做了简单介绍，以作为后续阅读的基础；第二部分包括第2、3章，第2章从整体上介绍了类型，第3章从整体上介绍了类型的思想；使得读者能够对基于对象和面向对象的程序设计有一个宏观上的认识；第三部分包括第4～6章，第4章全面介绍了对象，第5章全面介绍了运算符重载，第6章以一个具体的例子全面介绍了基于对象的程序设计；第四部分包括第7～12章，第7章全面介绍了继承，第8章全面介绍了内存布局，第9章全面介绍了类型转化，第10章全面介绍了虚拟机制，第11章以一个具体的例子全面介绍了面向对象的程序设计，第12章使用多个样例全面展示了各种类型的设计方法。

在阅读本书的过程中，请注意书中的【例X－Y】程序、【例X－Y】分析、【例X－Y】片段、【例X－Y】类型、【例X－Y】类层次结构等部分，这些就是在实际工作中经常用到

的经验和技巧，通过认真的阅读和理解可以掌握面向对象程序设计的典型做法。这里的“程序”是指可直接运行并产生输出的代码；“分析”是指为了说明特定问题的解释性代码；“片段”是指为了说明特定开发技巧的示例性代码；“类型”是指有代表意义的类或结构体的代码；“类层次结构”是指存在继承关系的一组类型的代码。

由于编者水平有限，加上时间仓促，错误和不妥之处在所难免，敬请广大读者不吝指正。

编者

目　录

第1章 C++基础

前面的学习，无不是为进一步的学习打基础；
积跬步于基础，才能致千里于类型。

C++语言是从C语言扩展而来的。因此，有人就把C++中C语言的部分作为C++语言的基础知识，只能说这种想法是基本正确的，因为C语言中的一些知识在C++语言中已经被改变了看待方法和使用方式。例如C++语言把结构体和共用体提升到了类的高度，使用常变量取代宏定义形式的常量等，都足以改变我们的编程习惯和编程方法。因此，读者有兴要认真地学习C++语言的基础知识，这对以后学习更为复杂的内容很有帮助。

本章要点

- 基于对象的程序设计方法
- 面向对象的程序设计方法
- 分支与循环控制结构
- 枚举、结构体和共用体
- 内存开辟与指针
- 指针数组和数组指针
- 函数参数与返回值
- 指针函数和函数指针

本章样例

- 【例1-1】程序：经典的C++问候
- 【例1-2】片段：贯通的分支语句
- 【例1-3】片段：枚举类型作为分支条件
- 【例1-4】类型：枚举类型作为索引
- 【例1-5】类型：用结构体实现结点（1）
- 【例1-6】类型：用结构体实现结点（2）
- 【例1-7】类型：共用体内嵌结构体类型
- 【例1-8】类型：自解释的句柄
- 【例1-9】片段：开辟二维数组
- 【例1-10】片段：释放二维数组

1.1 C++概述

C++作为一种开发语言，它的最终产物就是我们所熟知的程序。从文本形式的C++代码到可直接在计算机上运行二进制形式的C++程序，这一过程就是我们常说的编程和编译。而我们学习C++语言的目的亦是如此，或者说我们书写C++代码是希望能生成正确的程序。因此，我们站在程序这个结果的位置上，去审视C++编程将会变得有的放矢。本节的目的就是让大家对如何书写一个可运行的、完整的小程序有个基本的认识，同时这个小程序的生成过程也在很大程度上明显地代表了所有C++程序的生成过程。所以我们有必要认真学习这些知识。

1.1.1 程序

还记得C语言中那句经典的问候吧？是的，当以C++语言做出同样的表达时，你应该能够找到它们之间的些许差异。

样例“经典的C++问候”的代码如下：

【例1-1】 程序：经典的C++问候

```
#include <iostream.h>

int main()
{
    cout << "Hello World!" << endl;
    return 0;
}
```

对程序中的代码做出解释：

- 第一行代码是预处理指令，用于将iostream.h中的代码包含进来，作为HelloWorld.cpp的一部分，并参与最终的编译。
- 第二行代码是main函数的开始，main函数是所有C++程序必须存在的函数，是程序运行的入口点。
- 第三行代码开始是大括号，表示main函数定义的开始。
- 第四行代码是输出语句，它在iostream.h文件中定义，用于输出“Hello World!”字符串，并随后输出一个换行符。
- 第五行代码是返回语句，用以返回程序的运行状态。
- 第六行代码是结束大括号，表示main函数定义的结束。

程序运行时，将首先从main函数的第一条语句开始执行。cout是在iostream.h文件中定义的全局变量，它代表了一个输出设备（通常是显示器）。“<<”是一个操作符，用于向cout代表的设备输出数据。而endl代表的是一个换行符。因此，程序将先输出由双引号括起来的字符串“Hello World!”，紧接着输出一个换行符，使得新的输出位置移至“Hello World!”的下一行开始处。return是C++中的一个关键字，用于表示返回函数的调用处。

因为 main 是 C ++ 的入口函数，当程序运行到 return 语句时，表示程序应该返回了。返回值 0 用于表示程序的正常退出。

程序运行后将输出如下结果：

```
Hello World!
```

1.1.2 预处理指令

我们知道，C 程序的编译过程包括预处理、编译和连接，C ++ 程序也一样，因此它们都需要预处理指令。预处理指令就是在编译的预处理阶段由预处理器执行的命令。经常使用的预处理指令包括如下几种：

1. 头文件包含指令

头文件包含指令的格式是：

```
#include
```

该指令的基本用法是：

```
#include <header.h>
```

该指令用于将指定的 header.h 头文件中的代码插入到该预编译指令所在的位置，使得头文件中变量或函数的声明或定义在当前代码文件中可见，为后续的编译过程提供保障。

2. 宏定义指令

宏定义指令的格式是：

```
#define
```

宏定义指令有两个基本的用法，分别用于：

- 定义宏形式的常量。
- 定义宏形式的函数。

（1）宏常量

宏常量的基本用法是：

```
#define PI 3.1415926
```

该指令用于为文字量定义一个容易记忆的别名。在预处理阶段，代码文件中所有出现这个别名的位置都被替换为对应的文字量。

于是便可以这样使用这个常量：

```
double result = r * r * PI; //假设 r 已经预先定义
```

（2）宏函数

宏函数的基本用法是：

```
#define max((a), (b)) ((a) > (b) ? (a) : (b))
```

于是便可以以函数调用方式去使用它：

```
int val = max(6, 5);
```

3. 条件编译指令

条件编辑指令是由一组指令构成的，用以构成条件关系。这些指令包括：

```
#ifdef
#ifndef
#elif
#else
#endif
```

该指令的最简单的格式是：

```
#ifdef
//满足编译条件的代码片段
#endif
```

经常在跨编译环境（如32位和64位编译环境）、跨操作系统编译时使用条件编辑指令。除此之外，条件编译指令还有一些特定的使用方式，其中之一就是防止头文件的重复包含。如下：

```
//GenFun. h
#ifndef _GEN_FUN_H_
#define _GEN_FUN_H_

//头文件的代码

#endif //_GEN_FUN_H_
```

当头文件 GenFun. h 在一个源文件中被重复包含时，由于条件编译指令的存在，使得第二次被包含的操作无效。

1.1.3 输入与输出

输入与输出是一个程序不可或缺的部分，也是与用户交互的基本手段。准确地说，C++中的输入与输出不是语言本身的组成部分，但却是语言规范的组成部分，它被定义在语言规范的“标准语言库（STL）”中。

C++中的输入与输出使用的是流（stream）的概念，是使用面向对象思想设计的类型，包括输入流（input stream）、输出流（output stream）等更为具体的类型。另外，对于常规的应用，标准语言库中提供了三个标准的流对象，即对应标准输出设备（如显示器）的输

出流 cout、对应于标准输入设备（如键盘）的输入流 cin、和标准错误流 cerr（通常也输出到显示器）。

一般使用标准输入流和标准输出流即可完成常规的工作。为了使用它们，需要包含对应的标准流头文件。如下：

```
#include <iostream>
```

cout、cin 等标准流对象即定义于该文件中。但是，cout 和 cin 代表的是输出和输入设备，而不是输出和输入操作。对应的操作是通过操作符来完成，cout 对应的操作符是 <<，cin 对应的操作符是 >>。另外，还有一个经常用到的起到控制作用的操作符 endl，它代表的是换行操作。

在学了上述有关输入和输出的基础知识之后，就可以写出一个“星期打印机”的控制台程序，以便对输入和输出知识加以运用。如下：

```
#include <iostream>

int main()
{
    cout << "Week printer program." << endl;
    cout << "Please type number(0-6):" << endl;

    int num;
    while(true)
    {
        cin >> num;
        switch(num)
        {
            case 0: {cout << "SUNDAY" << endl;          break;}
            case 1: {cout << "MONDAY" << endl;          break;}
            case 2: {cout << "TUESDAY" << endl;         break;}
            case 3: {cout << "WEDNESDAY" << endl;       break;}
            case 4: {cout << "THURSDAY" << endl;        break;}
            case 5: {cout << "FRIDAY" << endl;          break;}
            case 6: {cout << "SATURDAY" << endl;        break;}
            default: {cout << "See you next time." << endl; return 0;}
        }
    }
}
```

当运行这个程序并输入 0 到 6 之间的数字时，将对应地打印出星期中的每一天。当输入其他数字时，程序将正常退出。

1.1.4 C++编程思想

C++语言是从C语言扩展而来的，自然提供了比C语言更为丰富的编程思想。整体而言，C++语言为我们提供了如下几种编程思想：

- 基于过程的编程思想。
- 基于对象的编程思想。
- 面向对象的编程思想。

用一句话概括每种思想是再恰当不过的做法了，任何貌似负责和追求叙述完美的做法都将使真实的思想迷失在一望无垠的文字中。在这里编者将给出自己的基于对象和面向对象的编程思想。

（1）基于对象的编程思想

基于对象的编程思想就是从对象的角度去看待所有事物，并使用类为所有需要解决的问题建模。

> 在这种思想下，要考虑面向对象的封装性。

我们将在1.1.5节给出基于对象的程序设计方法，并在第6章给出一个完整的基于对象程序设计的例子。

（2）面向对象的编程思想

面向对象的编程思想就是从继承的角度去看待所有事物，并使用类层次结构为所有需要解决的问题建模。

> 在这种思想下，还要考虑继承性和多态性（也仍然包括封装性）。

我们将在第3章对C++编程思想的封装性、继承性和多态性做进一步的讨论，在1.1.6节给出面向对象的程序设计方法，并在第11章给出一个完整的面向对象的程序设计的例子。

1.1.5 基于对象的程序设计

按照基于对象的程序设计思想，可以得到基于对象程序设计的基本过程。

（1）问题抽象

问题抽象是首先需要解决的事情，亦即把每个需要解决的问题都抽象为具体的类型。这是一个从问题域向类型域映射的过程，在这一过程中需要做的事情是：

- 分析每个问题的特性，得到问题的属性和操作，但不是分析所有特性。
- 提炼得到属性和操作，保留需要的部分，以作为类型域的成员。
- 分析类型域中各个类型之间的关系，以用于类型之间的组织。

（2）数据封装

在这一步需要完成如下事情：

- 使用合适的数据类型去表示类型域中的每个属性，得到类型的各个数据成员。

- 确定每个数据成员的所属类型，是属于类的静态成员，还是属于对象的数据成员。
- 除非必要，我们把所有数据成员的可见性限定为 private。

（3）设计接口

这一步主要是确定类型域中每个类型上的操作：

- 使用合适的函数去定义类型域中的每个操作，得到类型的各个成员函数。
- 确定各个成员函数的所属类型，是属于类的静态成员函数，还是属于对象的成员函数。
- 确定每个成员函数的可见性，是对外访问的接口，还是类型内部的辅助接口。

（4）完善操作

在这一步为每个类型提供必要的操作符：

- 确定对象是否允许复制或赋值，是重载赋值运算符，还是禁用赋值运算符。
- 确定对象是否参与计算或比较，以便提供相应的算术和比较运算符。
- 其他更多有待确定的操作符。

（5）确定关系

类型域中各个对象不是孤立的，需要确定它们之间的关系：

- 确定对象之间的类型关系，是彼此相关的友元类型，还是存在包含的嵌套类型。
- 确定对象之间的存储关系，是作为数组的成员，还是作为静态的存储对象。
- 其他更多有待确定的关系。

（6）创建对象

抽象出来的类型最终将供程序使用，需要为对象的创建和使用做出保障：

- 确定对象的创建权限，是仅供内部使用的对象，还是供全局使用的对象。
- 确定对象如何被建立和销毁。
- 其他更多有待确定的与创建对象有关的问题。

至此，我们完成了基于对象的程序设计。我们从需要解决的问题中抽象出了每个具体的类，它们使用数据成员和成员函数对其做出表述，同时也确定了类之间的关系，提供了必要的访问接口，并且完善了每个类的运算（符），对类对象的使用也做出了交代。接下来，只需让这些类型产生它们的对象，并按照既定的关系组织起来即可。

当然，此时我们已进入了编码阶段，程序设计已告结束。

1.1.6 面向对象的程序设计

面向对象程序设计相对于基于对象的程序设计更为复杂，而它解决问题的能力更为强大。在进行面向对象程序设计时，除了考虑封装性之外，还要考虑它的继承性和多态性。

当然，对于一个具体的问题未必只有唯一的设计方法，可以存在类似的设计方法。特别是我们都以面向对象的思想去解决同一问题时，可能殊途同归。因此，在这里给出一个可行的设计过程以做抛砖，期望引玉。同时，即使各在殊途，期望同归。

面向对象程序设计的基本出发点依然是问题抽象，但它是逐步求精的过程，是从更为抽象的类型向更为具体类型的演进过程。最终，需要解决的问题被抽象为一组存在紧密关系的类层次结构。

下面，我们给出面向对象程序设计的基本过程。