普通高等教育“十一五”国家级规划教材

C++程序设计系列教材

C++程序设计教程(第二版)习题及解答

钱能 著

清华大学出版社

普通高等教育“十一五”国家级规划教材

C++程序设计系列教材

C++程序设计教程(第二版)习题及解答

钱能 著

清华大学出版社
北京

内 容 简 介

本书是主教材《C++程序设计教程（第二版）》的配套书。由于第二版主教材对第一版做了根本性的改动，使得本书也与第一版的《C++程序设计教程习题及解答》大相径庭。本书从习题练习出发，引导读者从机器运行的角度来思考问题，以编写出能够实战的程序代码。本书也是作者《C++程序设计教程（第二版）实验指导》一书的对照和补充。

全书突出 C++编程能力培养，全局把握抽象编程观，潜窥语言和系统的内在特性，力图与同类书相区别。书中还介绍了测试数据制作、各种策略之代码演变、细节优化以及各个编译器性能差异的比较。在面向对象程序设计中，演绎了一个融概念设计和系统实现于一体的模型，并对动态链接库的实现方法进行了讨论。

所有的代码都经过作者调试，体现了独特的代码风格，给出了注重实战的优化代码。所涉及的技巧与方法也许并不为人所知，但其代码性能在 acm.zjut.edu.cn 的提交系统的测试中，都名列榜首，所以能给读者以很好的参考。

代码强调 C++标准，均经 VC8、G++4、BCB6 三种编译器调试通过，并被做成电子文档，在清华大学出版社网站（http://www.tup.tsinghua.edu.cn）上供读者下载。

图书在版编目（CIP）数据

C++程序设计教程（第二版）习题及解答 / 钱能著. —北京：清华大学出版社，2009.10
（C++程序设计系列教材）
ISBN 978-7-302-20713-9

Ⅰ. C…　Ⅱ. 钱…　Ⅲ. C 语言－程序设计－高等学校－解题　Ⅳ. TP312-44

中国版本图书馆 CIP 数据核字（2009）第 139495 号

责任编辑：郑寅堃
责任校对：时翠兰
责任印制：杨　艳
出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　　编**：100084
社　总　机：010-62770175　　**邮　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质　量　反　馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 者：北京国马印刷厂
经　　销：全国新华书店
开　　本：185×260　**印　张**：18.5　**字　数**：456 千字
版　　次：2009 年 10 月第 2 版　　**印　　次**：2009 年 10 月第1 次印刷
印　　数：1～5000
定　　价：29.00 元

第二版前言

《C++程序设计教程（第二版）习题及解答》是主教材《C++程序设计教程（第二版）》的配套书。由于第二版主教材对第一版做了根本性的改版，《C++程序设计教程（第二版）习题及解答》也与第一版大相径庭，其涉及的知识面更广，内容更多，从易到难的幅度更大，所以其解答也相应更详尽、具体。同时，它也是对《C++程序设计教程（第二版）实验指导》一书内容的对照和补充。

撰写意图

本书从习题练习出发，引导读者从机器运行的角度来思考问题，以编写出能够实战的程序代码。

C++主教材虽然涉及一些数学方法描述的算法，但是它终究不是一本算法书，所以只在简单明了的情况下，偶尔表现一下算法技巧。习题解答也沿袭了主教材的这一著述原则，在代码和解答中，侧重表现C++语言的描述技巧，而无长篇大论展开算法描述之意图。因此，所有的解答都不是晦涩难懂的。如果对解答的代码颇感新奇，那就是作者在充分运用C++语言的特性，贯彻作者强调启发性的意图。

习题解答的另一个目的是让读者明白，大量的撰写和调试代码，是优秀程序员的必然经历。要让代码不但能正确运行，而且还要运行得出色，这个能力很大一部分是基于测试和调试技术。学会了自己控制整个开发平台，自己制作测试数据，自己调试各种代码，本身便是具有了一定程度的对新开发软件的学习能力。语言逻辑表达能力只是解决程序结构和框架的问题，但是能力的更现实体现是如何将正确的程序得以在某个平台上测试和运行。虽然最终是要展开编程中的数学逻辑（即算法）研究，但是，不会调试和测试对编程能力的提高将是一个很大的障碍。

每个习题解答都先将练习目的摆明，然后直奔主题，介绍解答思路。虽然习题解答的描述在结构上比较自由，但都是从要害上着手。有些习题还给出了设计指导和框架图示，目的是给读者更多的编程思路。解答展开了许多编程的细节和经验，这些经验在纯粹看书学习中也许很难消化成为自己的编程能力，只有自己亲身体验，才能把握。这一方面是想让读者更深入地理解实验之重要性，另一方面，由于有些习题已经演变成了《C++程序设计教程（第二版）实验指导》一书中的内容，所以本习题解答还旨在帮助读者强化实践能力。

编写形式

本书按主教材每章后面的全部习题为素材，逐一进行解答。由于每章的学习内容侧重点不同，或重方法，或重分析，或重理解、重实践等，要求不一，目的不一，所以在撰写习题解答和按从易到难的顺序逐一展开的时候，也对其内容做了详略不等的描述。

习题解答多以代码和解释的形式展开，因而代码的风格和可读性至关重要。本书的代

码风格沿袭了主教材的代码风格。函数、类型和结构定义，全局数据描述、编译指令描述都予以区别，循环控制按锯齿形框架分层，代码结构力呈优雅，关键之处给予注释。因代码本身就是设计框架和动作流程的通俗展示，伴随着语言描述的细节，比之算法流程的逐条文字描述更显立体感和形象化。

主教材从贯穿全书的编程方法主线中频频诱导快乐学习心态，它以体系架构和学习引导为己任，而对语法却不拘小节，并不强调面面俱到。在习题解答中，还涉及主教材中并没有提及的一些基本编程技术、资源和知识。例如，介绍和使用了一些 C 函数库、STL 技术和相关技术手段。这一方面是对主教材的横向补充，同时也拓展了一些 C++的新颖知识。

本书在叙述到基本编程（第 5~7 章）阶段时，便需要更多的编程经验。其中，有选择地重点介绍第 6 章的练习 8，正是为了揭示 C++内部特性，授之以独到的编程心得。该练习介绍了测试数据制作、各种编程策略之代码演变、细节优化以及各个编译器性能差异之比较。在方法介绍中，刻意关注代码在运行时间和空间代价上的互偿性。所有表现不同特点的代码，其版本有数十个之多。除此之外，在第 13 章的抽象类中，还演绎了一个面向对象的程序设计，融概念设计和系统实现为一体，并注重面向对象设计的实现细节，对动态链接库的实现方法进行了多角度的讨论。

全书与主教材遥相呼应，仍然突出 C++编程能力培养中的两个方面：一是全局把握抽象编程观，二是潜窥语言和系统内在特性。通过其丰富的技术展示，再一次表现了编程本质，亦突出了区别于其他同类书之特色。

代码说明

所有的代码都由作者撰写并调试。这一方面是说，代码解答不是为了解答而解答，而是在体现代码风格之个性的同时，给出注重实战的独特代码。代码中所使用的技巧与方法也许并不为人所知，但是，因为它们中许多都在 acm.zjut.edu.cn 的提交系统中，经过耗用时间、占用空间以及代码长度的较量而名列榜首，所以即使读者会写解决某个问题的代码，书中所给出的代码仍然能给读者以很好的参考。另一方面是说，代码调试的工作量浩大，书中每个习题都给出了至少一种经调试通过的参考代码。调试代码的工作甚至比撰写书稿更费时费力，有时为了找出更好的描述方法或者为了更准确地评价性能价值，还会将几种方法在不同调试环境中查看机器语言的中间结果。

习题解答的代码都是基于控制台运行环境，对于面向对象编程实践中的动态链接库也不例外。解答中均提供完整的可运行代码，目的是让读者获得整体感，便于对各种平台进行比较和自学。全部代码均经 VC8、G++4、BCB6 三种编译器调试通过，也就是强调了代码的 C++标准性。它们均被做成电子文档，在清华大学出版社网站 http://www.tup.tsinghua.edu.cn 上供读者下载。

温馨告白

在本系列丛书主教材出版之后，许多学生和读者出于对学习的渴望，对本书的出版抱有厚望，从各个渠道与作者联系，索要本书的初稿，提出各种建议，这些都促成了本书的应时出版；“C++程序设计”作为省级精品课程来建设，学校领导的大力支持，保证了课程

教学和教材建设的外部环境，使作者得以深入从事相关教学研究与实践，也促进了本书的撰写工作。

这里，特别感谢恩师王国东先生一直以来给予的细雨润物般的关怀、教诲和启迪。

本书因启动时间晚，所以迟至今日才与读者见面。本书撰写历时虽短，但还是把主教材中学习 C++的重要问题和经验理了出来，把自己的风格和特色体现了出来。

描述问题的解答涉及 C++语言、环境、标准、技巧、风格等各个方面。学习和研究角度不同，则看法和解决方法便不同，有些或许还可以做得更周全或更精彩，欢迎读者来信讨论。

作者电子邮件地址：qianneng@mail.hz.zj.cn

钱 能

2009 年秋于杭州自在居

第一版前言

学习程序设计方法，并且要实质性地提高编程能力，除了看书理解之外，有两个关键的因素：一个是做编程书面练习，一个是上机做实验。在初级程序设计的能力培养中，唯有多练，才能真正找到编程的感觉，才能培养出宝贵的编程经验。这些感觉和经验，一定程度上决定了日后成为什么样的计算机人才。所有计算机高级人才，都曾走过风风雨雨的编程历程。

本书既为那些想提高编程能力的初学者准备，又为设计、开发软件的程序员提供了编程方面的素材。书中共有习题 67 例，每个习题解答都是精选的讲课例子，对程序设计中存在的问题具有一定的代表性。有些解答还提供了不止一种方法。书中除了个别问答题外，全部解答都通过 BC++5.0 的调试和运行，并配有运行结果，以便读者在不同环境下运行比较。

程序的解答可以各种各样，越吃透程序结构、吃透语言的内在含义和联系，就越能准确表达解决问题的方法，编制的程序也就越简练，对日后深入学习程序设计理论的领悟性也越高。读者可以先自己尝试解答，然后再参考书中解答。书中的解答与系列丛书《C++程序设计教程》中的习题具有相似性，可以起到提示和帮助读者解答教程中习题的作用。当然书中的解答并不一定是最优的，它们只是具有代表性而已。欢迎读者提出自己的见解，编出更高质量的程序。

虽然这只是迈出了一小步，但作者愿与程序设计爱好者们一起努力，在明年的再版中，更上一层楼。

作者的学生刘雪芬、沈雪飞和徐晓萍做了大量习题中程序的录入和调试工作，在此表示深深的感谢。

作者的电子邮件地址是：

qianneng@mail.hz.zj.cn

目　录

第一部分　基 础 编 程

第二部分 过程化编程

第三部分 面向对象编程技术

第四部分　高 级 编 程

第一部分　基 础 编 程

第1章　概述 练习解答

EX0101

这是本书的第一个C++代码，其解答为：

```
//=====================================
// EX0101.cpp
// simplest program with output
//=====================================
#include<iostream>
//-------------------------------------
int main(){
  std::cout<<"I am a student.\n";
}//=====================================
```

前面附着#的语句行：

```
#include<iostream>
```

说严格些，它不是C++语句。也就是说，它并不规定机器做什么，而是规定编译器在编译时做什么，它们也称为伪指令，用斜体区分。

输出用 cout，它本是应用了 C++提供的标准输入、输出的流资源，因此使用包含 iostream 的指令（用斜体表示）。凡是涉及C++资源的，都在名为 std 的名表中注册，所以若默认使用C++资源，必须在包含指令后面再加：

```
using namespace std;
```

代码中在 cout 前加了 std::，表示由于没有默认使用，只得在使用名字时加前缀，或者说，单一使用只须加前缀，便可以省略上述的名空间说明。

一般来说，语句中往往涉及C++各种资源的诸多使用，为了默认使用，在初学者的程序里，通常都会加上上述名空间使用语句。

另外，上述解答代码中，一些加了双斜杠的行是注释行，它不起执行的作用，只是给人阅读。代码其实可以写成没有注释语句的形式：

```
#include<iostream>
```

```
int main()
{
  std::cout<<"I am a student.\n";
}
```

本解答代码追求规范、个性化、可理解，习题与解答在题号上对应，所以在代码头上加上了一些注释。

代码中双引号括起来的是字串。如果字串中本身含有双引号，则需要在双引号字符前加引导符\，例如：描述“I say "OK!"”，用C++语句写为：

```
std::cout<<"I say \"OK!\"\n";
```

\n是换行字符，它是用两个字符来描述一个控制行为的控制符。语言的描述都是用有形可见的字符符号，也就是编程中使用的字符，恰如英语中的英文字符用以构成英语。如果要用这些符号描述无形的控制行为则要加引导符'\'。控制符在主教材 CH3.2.1 的表3-3上描述了一些，一般不常用。

```
int main(){
  ⋮
}
```

是一个函数描述，专业地说，为函数定义。它是一个函数名(main)，加上一对小(圆)括号，附上返回类型(int)，再加上一对花括号，里面写上若干条语句。左花括号的位置只要跟在右小括号后面，写在哪里都行。不同的书写方式反映了编程的不同风格。

C语言风格的代码为：

```
//=====================================
// EX0101.cpp
// simplest program with output
//=====================================
#include<stdio.h>
//-------------------------------------
int main(){
  printf("I am a student.\n");
}//=====================================
```

它使用了不同的资源（头文件）。头文件是指以扩展名.h结尾的文件，头文件多在C语言中使用，C++当然可以兼用，但是C++更多使用的是没有.h扩展名的资源。包含不同的资源，将导致代码中的输出采用不同的方式。

语句以分号结束，C++编译器以分号区分各语句单位。语句总是写在函数中，表示计算或者输入、输出等操作。

EX0102

可由若干行语句来完成本问题的字符图形输出的工作。

```
//=======================================
// EX0102.cpp
// 简单字符图形输出
//=======================================
#include<iostream>
using namespace std;
//---------------------------------------
int main(){
  cout<<"   *\n";
  cout<<"  ***\n";
  cout<<" *****\n";
  cout<<"*******\n";
  cout<<" *****\n";
  cout<<"  ***\n";
  cout<<"   *\n";
}//=======================================
```

main()函数中的语句是要求机器执行的动作序列。这里是由 7 个输出语句构成。每个语句负责输出一行字符，这是为了直观。

然而，C++输出语句的能力并非一定按行为单位进行，可以用一条语句执行几行的输出。下列是用一个输出语句通过反复执行<<操作，将字串送往输出设备，从而将该图形整个输出。

```
cout<<"     *\n"<<"    ***\n"<<"  *****\n"<<"*******\n"<<"  *****\n"<<"
***\n"<<"   *\n";
```

我们看到，一条语句在一行写不下时，可以由接着下一行继续写。编译器在识别到分号时才认可整条语句的结束。所以，可以通过添加一些空格、空行，人为地构造优雅的代码格式。上述语句可以这样写：

```
cout<<"   *\n"
    <<"  ***\n"
    <<" *****\n"
    <<"*******\n"
    <<" *****\n"
    <<"  ***\n"
    <<"   *\n";
```

虽然增加了语句行，但是形象，只要注意在最后一行才打上分号，以表示前面的代码行只是一条语句的组成部分而已。

我们还看到，该语句是将各个字串按行分段表示和逐段输出的。这有点极端，事实上，可以将各段字串拼合：

```
cout<<"   *\n  ***\n *****\n*******\n *****\n  ***\n   *\n";
```

它表示一个较长字串一次性送往输出设备去显示。字串中的每个“\n”表示换行，它控制设备在相应的输出中执行换行。所以整个字串的输出会在结果窗口中分段列在不同的行上。如果一个字串实在很长，一行写不下，可以在行尾插入一个斜杠符号，以示意编译器下行继续，从而不被当作语句不完整而出现编译错误。例如上述语句也可以这样写：

```
cout<<"   *\n  ***\n *****\n*******\n *****\n  *\
**\n   *\n";
```

由于下一行必定从头算起以作为上一行的继续，因此书写格式上没法通过加空格来修饰，可能会觉得不够优雅。一些超长的字串往往用数据文件的方式存取。所以这种代码的形式只有在翻读前人书写的代码时还能看到，现代编程中很少能看到这种代码。

下列作为极端的例子，用几条语句执行一行的输出：

```
cout<<" ";
cout<<" ";
cout<<" ";
cout<<"*";
cout<<"\n";
```

这 5 条语句构成了图形中第一行的输出字符。空三格，然后显示一个'*'字符，最后换行。

在运行结果的窗口中，显示的字符总是按语句执行的先后顺序，从左到右，从上到下。所以，输出一个图形，必须按顺序先输出最上面的字符行。为了到达行中显示可见字符的位置，需要先输出几个空格。为了输出下一行，在当前行必须先输出一个换行符（将控制符也作为一个字符进行输出），以使显示位置跃到下一行的开头。

逐个字符输出，用逐条语句描述，耗费了很多语句，才显示了很少的字符，因而从运行性能上来说划不来。但是在需要通过计算来获得字符个数、字符内容和字符位置，从而进行批量自动（循环控制）输出，而代替人工的时候，这种输出形式却是学习循环控制的基础。

例如，输出三个'*'字符，可以用循环语句来实现：

```
for(int i=1; i<=3; i=i+1)
{
  cout<<'*';
}
```

这是一个循环语句，花括号括起来的语句是被 for 循环描述，反复执行的语句序列，称为循环体，这里只含有一条语句。这条语句执行的次数由 for 循环的描述头说明，因为 i 变量从 1 到 3，不断地增 1 变化，说明该循环一共将进行 3 次，从而满足连输三个'*'的要求。由于循环体只含有一条语句，C++语法规则允许省略花括号：

```
for(int i=1; i<=3; i=i+1)
  cout<<'*';
```

该代码的语义与前面相同。在开始循环时，先令 i=1，然后判断是否 i<=3，如果判

断结果为假，则结束 for 循环语句，转去执行下一条语句。否则，进入并执行循环体语句。在这里，第一次判断时，i 为 1，所以是 1<=3，所以，执行 cout<<'*';语句，之后，循环变量 i 增 1 使其值为 2，接着再判断 i<=3，得到真值，于是继续进入并执行循环体语句，输出第二个'*'，周而复始，等到第三个'*'输出之后，再增 1 给 i 使其值为 4，判断 i<=3 时，判断结果为假，于是整个 for 循环语句宣告结束。这时候，便已经得到连续三个在屏幕上显示的'*'。

for 循环的循环体中可能会有许多语句，但其都归属在 for 循环语句体内，作为一个循环语句的整体，被视为一个语句单位。在循环语句的执行中，则自然按循环控制规则按序执行。

例如：输出一行 7 个'A'字符，可以循环执行输出 1 个'A'字符 7 次，再输出一个换行符：

```
for(int i=1; i<=7; i=i+1)
  cout<<'A';
cout<<"\n";
```

注意上面省略了包住循环体的花括号。“cout<<'A';”语句归属于 for 循环，因而被执行了 7 次，“cout<<'\n';”语句却是在 for 循环语句之后，因而按序执行中，它只被执行一次。这与代码：

```
for(int i=1; i<=7; i=i+1)
{
  cout<<'A';
  cout<<"\n";
}
```

是完全不同的。因为后者，每次输出'A'后都要换行，而前者在连续输出 7 次'A'后统一执行一次换行。从程序形态上看，因为所描述的循环体中有两条语句，所以花括号不能省，而前者可以省略花括号。

如果要输入一个整数，并将该整数的平方值随后输出，则其代码为：

```
#include<iostream>
using namespace std;
int main()
{
  int n;
  cout<<n*n<<"\n";
}
```

该程序的运行结果取决于输入的整数值。

程序运行结果往往并不由编程决定，而由运行中输入的数值来决定。如果要根据输入的整数值 n，输出一个 n*5 的'Z' 字符矩形，则因为无法在编程中确定 n 的值，而不能简单地用确定个数的输出语句来完成。也就是说，下述语句行数不知道：

```
cout<<"ZZZZZ";
cout<<"ZZZZZ";
cout<<"ZZZZZ";
cout<<"ZZZZZ";
cout<<"ZZZZZ";
cout<<"ZZZZZ";
```

n行?

这时候，就需要用循环语句，设置循环变量初值，到运行中获得的 n 之终值，不断进行测试、增量、再测试、再增量的步骤，来控制输出语句的执行次数。可以这样来实现输出 n*5 的字符矩形的任务：

```
#include<iostream>
using namespace std;
int main()
{
  int n;
  cin>>n;
  for(int k=1; k<=n; k=k+1)
  {
     cout<<"ZZZZZ\n";
  }
}
```

循环体中是输出 5 个字符'Z'并换行的语句，通过控制执行 n 次的循环，满足输出的要求。

如果将矩阵转过来，不是输出 n*5，而是输出 5*n 的字符矩阵，则因为数字 5 是确定的，可以用 5 个循环语句来做：

```
#include<iostream>
using namespace std;
int main()
{
  int n;
  cin>>n;
  for(int k=1; k<=n; k=k+1)  // 每个 for 循环都输出 n 个字符的行，共 5 个 for
     cout<<"Z";
  cout<<"\n";                // 换行不能少
  for(int k=1; k<=n; k=k+1)
     cout<<"Z";
  cout<<"\n";
  for(int k=1; k<=n; k=k+1)
     cout<<"Z";
  cout<<"\n";
  for(int k=1; k<=n; k=k+1)
     cout<<"Z";
```

```
  cout<<"\n";
  for(int k=1; k<=n; k=k+1)
     cout<<"Z";
  cout<<"\n";
}
```

因为行中'Z'字符的个数，取决于输入的n值，编程时n不确定，而循环的次数可以用不确定量来描述，所以可用循环来描述输出含n个'Z'字符的输出：

```
for(int k=1; k<=n; k=k+1)
   cout<<"Z";
cout<<"\n";
```

5*n的字符矩形输出用5个同样的代码来拼合，这样的代码显得冗长，完全可以通过循环语句的循环（二重循环）来简化：

```
for(int k=1; k<=5; k=k+1)
{
  for(int i=1; i<=n; i=i+1)
    cout<<'Z';
  cout<<"\n";
}
```

它是将输出一行n个'Z'字符的语句集合，重复执行5次，或者说，将该语句集合作为循环体，外套一个重复5次的循环语句。

例如，输出一个'x'字符的5*5正方形，可以循环5次输出一行5个'x'字符，而输出一行5个'x'字符，又是一个循环5次的单字符输出，再追加一个换行：

```
for(int k=1; k<=5; k=k+1)
{
  for(int i=1; i<=5; i=i+1)
    cout<<'x';
  cout<<"\n";
}
```

外循环的循环体内容便是输出一行5个'x'字符。它由两条语句构成，一条是for循环语句，另一条是单独输出换行的语句。这里用的for循环在结构上构成了内外二重循环。

需要注意的是内外二重循环的循环变量名最好不要相同，以示区别。这里外循环变量用k，内循环变量用i。

而因为内循环中输出5个字符并换行的执行在编程时是确定的，所以可以直接用“cout<<"xxxxx\n";”语句替代内循环而使代码简化：

```
for(int k=1; k<=5; k=k+1)
  cout<<"xxxxx\n ";
```

那么，从输入中获得n值，输出一个n*n的'x'字符正方形，又该怎么编程呢？想必读者会找到答案的。