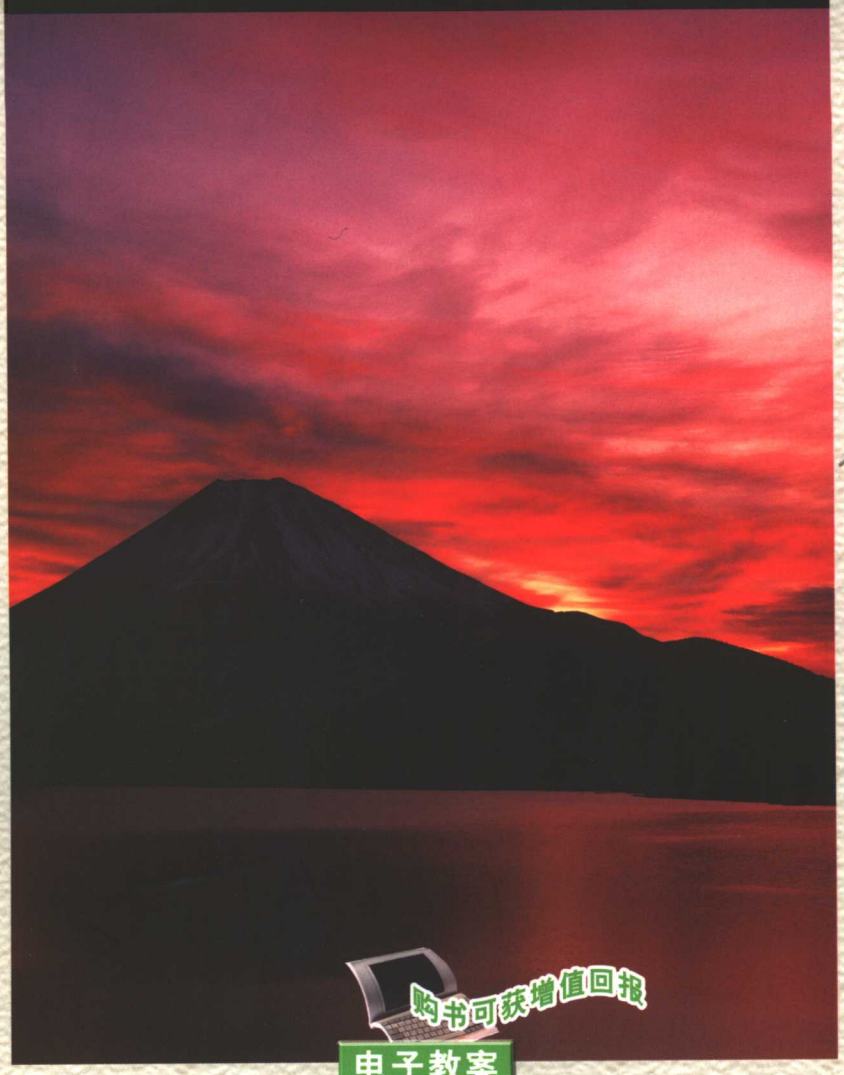
高等院校计算机基础教育规划教材



购书可获增值回报

电子教案

徐士良 主编

葛 兵 徐 艳 编著

C++程序设计

机械工业出版社
CHINA MACHINE PRESS

高等院校计算机基础教育规划教材

C++程序设计

徐士良 主编
葛兵 徐艳 编著

机械工业出版社

本书是C++的入门教材。全书分三部分，共16章，主要介绍了C++的语法知识，C++面向过程的程序设计和C++面向对象的程序设计。本书将重点放在对问题处理的方法和具体编程的方法，通过通俗易懂的例题分析使读者快速掌握C++的特点和程序设计方法。

本书可作为各专业的学生或工程技术人员学习C++程序设计的教材。

图书在版编目（CIP）数据

C++程序设计/徐士良主编. —北京：机械工业出版社，2006.6
（高等院校计算机基础教育规划教材）
ISBN 7-111-18958-2

Ⅰ.C... Ⅱ.徐... Ⅲ.C语言－程序设计－高等学校－教材
Ⅳ.TP312

中国版本图书馆CIP数据核字(2006)第034000号

机械工业出版社（北京市百万庄大街22号　邮政编码　100037）
策　　划：胡毓坚
责任编辑：陈振虹
责任印制：李　妍

北京地质印刷厂印刷
2006年5月第1版·第1次印刷
184mm×260mm·24.25印张·602千字
0001—5000册
定价：34.00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)68326294
编辑热线电话：(010)88379739
封面无防伪标均为盗版

出版说明

计算机基础教育在经济建设与社会发展中，发挥着非常重要的作用。我国高等院校十分重视计算机基础教育，在指导思想、教学设置及安排、优化知识结构等方面进行了大量的工作，其目的是为国家培养合格的高素质人才。为了满足教育的需求，机械工业出版社组织编写了这套“高等院校计算机基础教育规划教材”。

在组织编写过程中，我社聘请了高等院校承担计算机基础教育工作的主讲教授和骨干教师，对教学经验进行了总结和提炼，并对前瞻性课题和内容进行了研讨。针对课程特点，总结出课程中的知识点、重点和难点，并融入教材的编写中。

本套系列教材与课程紧密结合，定位准确，注重理论教学和实践教学相结合，逻辑性强，层次分明，叙述准确而精炼，图文并茂，习题丰富，非常适合各类高等院校、高等职业技术学校及相关院校的计算机基础教育，也可作为各类培训班的教材或自学参考书。

机械工业出版社

前　言

C++是从C语言发展演变而来的一种面向对象的程序设计语言。面向对象的程序设计相对于面向过程的程序设计来说，是一种新的方法，面向对象技术有许多抽象的语法和较难理解的名词术语，如果要求读者先掌握这些抽象的语法，弄清楚各个名词术语的含义，然后再将这些抽象的语法和术语应用到程序设计中去，这对于非计算机专业的读者来说，显然是困难的，实际上也没有必要这样去做。

基于上述原因，作者对本书作如下定位：

- 读者是非计算机专业的，没有C++基础，甚至连C语言都没有学过，或者即使学过C语言但并不熟练；
- 不作抽象的名词解释，利用实例直接介绍如何编程。例如，用编程实例介绍什么是继承和派生，而不是先介绍继承和派生的概念，再介绍抽象的语法，最后才介绍具体的编程。
- 本书的目标是学会用C++进行编程，通过读者自己的练习尽量达到熟练。因此，本书是入门性的，有关C++更深入的内容不作更多的介绍。

本书叙述简明扼要，通俗易懂，例题丰富，有利于读者自学。书中所有程序实例均在Visual C++ 6.0环境下调试通过。本书可作为各专业的学生或工程技术人员学习C++程序设计的教材。

由于作者水平有限，书中错误和不当之处，请读者批评指正。

为了配合本书的教学，机械工业出版社为读者提供了电子教案，读者可以在www.cmpbook.com上下载。

作　者

目　录

出版说明
前言

第 1 部分　C++ 基础知识

第 1 章　C++ 简介 …… 1
1.1　C 程序与 C++ 程序的简单比较 …… 1
1.2　C++ 程序的构成与书写格式 …… 3
1.3　如何运行一个 C++ 程序 …… 8
1.4　习题 …… 14
第 2 章　基本数据类型 …… 16
2.1　数据在计算机中的表示 …… 16
2.1.1　计算机记数法 …… 16
2.1.2　计算机中数的表示 …… 21
2.2　常量 …… 28
2.3　变量及其定义 …… 30
2.4　符号常量的定义 …… 35
2.5　习题 …… 36
第 3 章　运算符与表达式 …… 37
3.1　赋值运算 …… 37
3.2　算术运算符与算术表达式 …… 37
3.3　关系运算符与关系表达式 …… 39
3.4　逻辑运算符与逻辑表达式 …… 40
3.5　其他运算符 …… 42
3.6　表达式求值顺序的讨论 …… 45
3.7　习题 …… 48
第 4 章　基本的输入与输出 …… 51
4.1　输入流与输出流操作 …… 51
4.2　putchar 与 getchar 函数 …… 52
4.3　printf 与 scanf 函数 …… 54
4.4　习题 …… 56

第 2 部分　C++ 面向过程的程序设计

第 5 章　结构化程序的三种基本结构 …… 57
5.1　程序设计的基本过程 …… 57

5.2 流程图 …… 61
5.2.1 传统流程图 …… 61
5.2.2 结构化流程图 …… 62
5.3 选择结构程序设计 …… 63
5.3.1 语句与复合语句 …… 64
5.3.2 if 语句 …… 67
5.3.3 if…else 结构 …… 70
5.3.4 条件运算符 …… 74
5.3.5 switch 结构 …… 76
5.4 循环结构程序设计 …… 83
5.4.1 当型循环与直到型循环 …… 83
5.4.2 while 语句 …… 84
5.4.3 do-while 语句 …… 86
5.4.4 for 语句 …… 87
5.4.5 循环的嵌套与其他有关语句 …… 89
5.5 算法举例 …… 94
5.5.1 列举与试探 …… 94
5.5.2 密码问题 …… 98
5.6 习题 …… 99
第 6 章 编译预处理 …… 103
6.1 宏定义 …… 103
6.1.1 标识符定义 …… 103
6.1.2 带参数的宏定义 …… 104
6.2 文件包含命令 …… 108
6.3 条件编译命令 …… 109
第 7 章 函数 …… 116
7.1 模块化程序设计的基本概念 …… 116
7.2 函数的定义 …… 117
7.3 函数的调用 …… 119
7.4 函数间的参数传递 …… 121
7.4.1 形参与实参的结合方式 …… 121
7.4.2 局部变量与全局变量 …… 124
7.4.3 动态存储变量与静态存储变量 …… 126
7.5 内联函数 …… 128
7.6 函数的重载与函数模板 …… 130
7.6.1 函数的重载 …… 130
7.6.2 函数模板 …… 132
7.7 默认参数的函数调用 …… 133
7.8 函数的递归调用 …… 135

7.9 习题 …… 139
第8章 数组 …… 141
8.1 数组的基本概念 …… 141
8.2 数组的定义与引用 …… 143
8.2.1 一维数组 …… 144
8.2.2 二维数组 …… 145
8.2.3 数组的初始化 …… 145
8.3 字符数组与字符串 …… 148
8.3.1 字符数组的定义与初始化 …… 148
8.3.2 字符串 …… 149
8.3.3 字符数组与字符串的输入与输出 …… 150
8.3.4 字符串处理函数 …… 152
8.3.5 字符串变量 …… 154
8.4 数组作为函数参数 …… 157
8.4.1 形参数组与实参数组的结合 …… 157
8.4.2 二维数组作为函数参数 …… 160
8.5 排序与查找 …… 163
8.5.1 有序表的二分查找 …… 163
8.5.2 冒泡排序 …… 164
8.5.3 选择排序 …… 165
8.5.4 插入排序 …… 166
8.6 习题 …… 167
第9章 指针 …… 169
9.1 指针变量 …… 169
9.1.1 指针的基本概念 …… 169
9.1.2 指针变量的定义 …… 170
9.1.3 指针变量作为函数参数 …… 172
9.1.4 指向指针的指针 …… 175
9.2 指针数组 …… 176
9.3 数组与指针 …… 177
9.3.1 一维数组与指针 …… 177
9.3.2 二维数组与指针 …… 180
9.3.3 数组指针作为函数参数 …… 182
9.4 字符串与指针 …… 185
9.5 函数与指针 …… 188
9.5.1 用函数指针变量调用函数 …… 188
9.5.2 函数指针变量作为函数参数 …… 190
9.5.3 返回指针值的函数 …… 192
9.6 引用 …… 192

9.6.1 引用的基本概念 …… 192
9.6.2 引用作为函数参数 …… 194
9.7 习题 …… 195
第 10 章 自定义数据类型 …… 198
10.1 结构体类型 …… 198
10.1.1 结构体类型变量 …… 198
10.1.2 结构体与函数 …… 204
10.1.3 结构体数组 …… 207
10.1.4 结构体与指针 …… 212
10.2 链表 …… 216
10.2.1 链表的基本概念 …… 216
10.2.2 链表的基本运算 …… 219
10.2.3 多项式的表示与运算 …… 222
10.3 联合体 …… 227
10.4 枚举类型 …… 228
10.5 自定义类型名 …… 231
10.6 习题 …… 232
第 11 章 位运算 …… 235
11.1 二进制位运算 …… 235
11.2 位段 …… 240
11.3 习题 …… 243

第 3 部分 C++ 面向对象的程序设计

第 12 章 类与对象 …… 245
12.1 面向对象程序设计概述 …… 245
12.2 类的声明与对象的定义 …… 246
12.2.1 结构体类型与类 …… 246
12.2.2 类的声明 …… 249
12.2.3 对象的定义 …… 251
12.2.4 对象中数据成员的访问 …… 253
12.2.5 成员函数的调用 …… 255
12.2.6 在类外定义成员函数 …… 256
12.2.7 C++ 程序的结构 …… 258
12.2.8 程序举例 …… 260
12.3 构造函数与析构函数 …… 264
12.3.1 构造函数 …… 264
12.3.2 用参数初始化表对数据成员初始化 …… 272
12.3.3 析构函数 …… 272
12.4 对象的赋值与复制 …… 274

12.4.1 对象的赋值 …… 274
12.4.2 对象的复制与复制构造函数 …… 275
12.5 对象数组与对象指针 …… 279
12.5.1 对象数组 …… 279
12.5.2 对象指针 …… 281
12.5.3 this 指针 …… 283
12.6 共享数据的保护 …… 284
12.6.1 常对象 …… 284
12.6.2 const 修饰的类成员 …… 286
12.6.3 对象的常指针与常引用 …… 290
12.7 类的静态成员 …… 291
12.7.1 静态数据成员 …… 291
12.7.2 静态成员函数 …… 296
12.8 类的友元 …… 297
12.8.1 友元函数 …… 297
12.8.2 友元类 …… 301
12.9 类模板 …… 301
12.10 习题 …… 304
第 13 章 继承与派生 …… 307
13.1 继承与派生的基本概念 …… 307
13.2 派生类的声明 …… 309
13.3 派生类成员的访问控制 …… 311
13.4 派生类的构造函数与析构函数 …… 315
13.4.1 派生类的构造函数 …… 315
13.4.2 派生类的析构函数 …… 320
13.5 多重继承 …… 320
13.5.1 多重继承的概念 …… 320
13.5.2 虚基类 …… 323
13.6 习题 …… 326
第 14 章 多态性 …… 328
14.1 什么叫多态性 …… 328
14.2 运算符重载 …… 328
14.2.1 运算符重载的基本概念 …… 328
14.2.2 运算符重载的方法 …… 330
14.2.3 运算符重载的规则 …… 332
14.2.4 运算符重载实例 …… 333
14.3 虚函数 …… 340
14.3.1 问题的提出 …… 340
14.3.2 虚函数的声明 …… 343

14.3.3 虚析构函数 …… 345
14.4 纯虚函数与抽象类 …… 346
14.4.1 纯虚函数 …… 346
14.4.2 抽象类 …… 346
14.5 习题 …… 348
第 15 章 流类库与输入/输出 …… 349
15.1 流类库的基本概念 …… 349
15.2 标准输入流与输出流 …… 350
15.2.1 用流对象进行输入/输出 …… 350
15.2.2 用输出流对象的成员函数进行输出 …… 351
15.2.3 用输入流对象的成员函数进行输入 …… 352
15.3 文件流类与文件操作 …… 353
15.3.1 文件流对象 …… 353
15.3.2 文件的打开与关闭 …… 354
15.3.3 文件的读写操作 …… 356
15.3.4 文件的定位 …… 360
15.4 习题 …… 362
第 16 章 命名空间 …… 363
16.1 命名空间的声明与使用 …… 363
16.1.1 命名空间的概念 …… 363
16.1.2 使用命名空间解决名字冲突 …… 366
16.1.3 使用命名空间成员的方法 …… 368
16.1.4 无名的命名空间 …… 370
16.2 标准命名空间 …… 370
附录 …… 372
附录 A 基本 ASCII 码表 …… 372
附录 B 常用库函数 …… 373
参考文献 …… 378

第 1 部分　C++ 基础知识

第 1 章　C++ 简介

1.1　C 程序与 C++ 程序的简单比较

有的读者可能已经接触过 C 语言，那么用 C 语言与 C++ 语言编写的程序有什么区别呢？

C 语言是一种结构化和模块化的语言，它是面向过程的程序设计语言。面向过程的程序设计主要是围绕功能进行的，在 C 语言中，是用函数来实现一个功能。在这种面向过程的程序设计中，所有被处理的数据以及程序中所有的模块都是公开的，一个函数可以处理任意一组数据，同时一组数据也可以被多个函数所处理。例如，为了计算一批给定数据的平均值，可以用一个 C 函数实现，而求一组数据中的最大值可以用另一个 C 函数来实现，但实际上，利用计算平均值的 C 函数可以计算任意一组数据的平均值，即计算平均值的 C 函数对于所有的数据都是公开的；同样，对于一组数据来说，既可以通过调用计算平均值的 C 函数来计算这组数据的平均值，也可以通过调用求最大值的 C 函数求这组数据中的最大值，即一组给定的数据对于所有处理数据的函数是公开的。对于这种面向过程的程序设计，用户必须考虑程序中的每一个细节，即什么时候对什么数据进行操作。在程序的规模比较大，且数据比较多，处理又比较复杂的情况下，程序设计者就往往会感到力不从心，容易出错。

由上所述，传统的面向过程的程序设计方法，对于处理较小规模的问题是很方便的，但当程序规模较大时就不方便了。为此就产生了另一种程序设计的方法，即面向对象的程序设计方法。

在面向对象的程序设计中所面对的是一个个对象，所有的数据分别属于不同的对象。这种方法的一个重要特点是，将数据和对数据的操作封装在一个对象中，各对象之间相对独立，互不干扰。程序设计者只需要做两件事情：一是确定将哪些数据和操作封装在一起，即设计对象；二是确定如何通知对象活动，完成对相应数据的操作。例如，有多组数据需要处理，其中有的需要计算平均值和最大值，有的需要进行排序，那么就可以将需要计算平均值和最大值的那组数据以及计算平均值和最大值的操作代码封装在一起作为一个对象，将需要排序的那组数据以及排序的操作代码封装在一起作为另一个对象，此后，只需“启动”相应的对象就可以了。显然，在这种情况下，各个对象的操作完成了，整体的任务也就完成了。因此，这种面向对象的设计方法对于完成大型的任务是十分有效的。

C++ 是一种面向对象的程序设计语言，它在 C 语言原有功能的基础上作了扩充，还增加了面向对象的机制。C++ 是由 C 发展而来，它保留了 C 语言中的所有优点，因此，C++ 既可用于面向过程的结构化程序设计，又可用于面向对象的程序设计。

需要注意的是，与使用C语言需要有C编译系统一样，使用C++语言需要C++编译系统。另外，用C语言编写的源程序文件名的后缀为 .c，而用C++编写的源程序文件名的后缀为 .cpp。

下面看一个C++程序的例子。

例 1-1 编写一个C++程序，其功能是显示字符串“THE C++ PROGRAMMING LANGUAGE”。其C++程序如下：

```
#include <stdio.h>          /* 包含头文件 stdio.h,头文件带后缀 .h */
#include <iostream>         //包含头文件 iostream,头文件不带后缀 .h
using namespace std;        //使用命名空间 std
int main()                  //主函数为 int 型
{
   printf("THE C++ PROGRAMMING LANGUAGE\n");   /* 输出一行字符 */
   cout <<"THE C++ PROGRAMMING LANGUAGE\n";    //输出一行字符
   return 0;        //程序正常终止则返回 0 值
}
```

这是一个最简单的C++程序，其运行结果是在显示器屏幕的当前光标位置处显示如下两行字符串：

```
THE C++ PROGRAMMING LANGUAGE
THE C++ PROGRAMMING LANGUAGE
```

下面针对上述程序作以下几点说明。

1）在C程序中，可以用/*…*/的形式作注释；在C++程序中，除了可以用/*…*/的形式作注释外，还可以用以//开头作为注释。但必须注意，以//开头的注释部分不能跨行，即它是一种单行注释。

2）在C++程序中，除了可以使用C语言中的printf()函数输出信息外，还可以用cout输出信息。但在用cout输出信息时，要与<<运算符配合使用。cout的作用是将运算符<<右侧的内容输出到显示屏幕上。

除此之外，在C++程序中，除了可以用C语言中的scanf()函数从键盘输入数据外，还可以用cin从键盘输入数据。但在用cin从键盘输入数据时，要与>>运算符配合使用。cin的作用是将从键盘输入的数据送到运算符>>右侧的内存地址中。

在C++中，cout称为标准输出流，其隐含设备为显示屏幕；cin称为标准输入流，其隐含设备为键盘。

3）如果在程序中要使用printf()函数进行输出或使用scanf()函数进行输入，则需要包含头文件stdio.h。如果要使用cout进行输出或使用cin进行输入，则需要包含头文件iostream。

头文件带后缀 .h 是C语言的习惯用法，但在ANSI C++新的标准规定，系统头文件不带后缀 .h。

4）在C程序中，习惯将主函数说明为无类型（用void说明），以表示主函数没有返回值。但按照ANSI C++新的标准规定，C++程序中的主函数为整型（用int说明），如果程序正常终止应返回0值。

5）在上述程序中，语句

```
using namespace std;
```

表示在使用系统库时使用命名空间 std，这也是 ANSI C++ 新的标准规定。关于命名空间的概念将在本书的最后介绍。

特别需要说明的是，虽然 C++ 由 C 发展而来，与 C 兼容，用 C 语言写的程序基本上可以不加修改地用于 C++，但为了使初学 C++ 的读者能一开始就按照 ANSI C++ 新的标准规定编写程序，在本书以后的叙述中都将按照 ANSI C++ 新的标准规定介绍。

1.2 C++ 程序的构成与书写格式

首先介绍几个简单的 C++ 程序，使读者对 C++ 程序有一个大概的了解。下面的例子虽然简单，但反映了一般 C++ 程序的特点以及基本的组成。

例 1-2 编写一个 C++ 程序，其功能是显示字符串“How do you do!”。其 C++ 程序如下：

```
#include <iostream>          //包含头文件 iostream
using namespace std;         //使用命名空间 std
int main()                   //主函数
{
   cout <<"How do you do! \n";    //输出一行字符
   return 0;                 //程序正常终止则返回 0 值
}
```

这个程序与例 1-1 中的程序基本相同，其运行结果是在显示器屏幕的当前光标位置处显示如下一行字符串：

```
How do you do!
```

在本例的程序中，第 1 行“#include <iostream>”是“文件包含”命令；第 2 行语句“using namespace std;”的作用是在使用系统库时使用命名空间 std；由一对花括号{}括起来的部分是主函数体，其中的语句是实现程序的具体功能。

在本程序的函数体中，语句“cout <<"How do you do! \n";”的功能是将运算符<<右侧的内容（即字符串内容）输出到显示屏幕上。最后的“\n”表示换行。在 C++ 中，cout 称为标准输出流，其隐含设备为显示屏幕。如果在程序中要使用 cout，则要求包含头文件 iostream。

语句“return 0;”的作用是返回一个 0 值。因为在本例程序中的主函数为整型，C++ 标准规定当程序正常执行完后要返回一个 0 值。

例 1-3 从键盘输入一个直角三角形的两条直角边长 a 与 b，然后计算并显示输出该直角三角形斜边长 c。其 C++ 程序如下：

```
// 计算直角三角形斜边长
#include <iostream>          //包含头文件 iostream
#include <cmath >            //包含头文件 cmath
using namespace std;         //使用命名空间 std
```

```
int main()                                  //主函数
{ float a,b,c;                              //定义三个实型变量
  cout <<"input a and b :";                 //给出输入提示
  cin >>a >>b;                              //输入直角三角形两条直角边 a 与 b 的值
  c=sqrt(a*a+b*b);                          //计算斜边长
  cout <<" hypotenuse=" <<c <<endl;         //输出斜边长
  return 0;                                 //程序正常终止则返回 0 值
}
```

下面说明这个程序中由一对大括号{}括起来的函数体中的各语句功能及其执行情况。

第一个语句的功能是定义三个实型变量 a,b,c。执行到这个语句时,为这三个变量分配存储单元,以便存放相应的实数。

第二个语句的功能是给出输入提示信息。执行到这个语句时,在显示器屏幕的当前光标位置处显示如下字符串:

```
input a and b :
```

其作用是提醒用户需要从键盘输入两个实数给变量 a 与 b。

在这个语句中,cout 为标准输出流,其中用双撇号括起来的部分是需要输出的字符串内容。

第三个语句是输入语句。执行到这个语句时,系统就在此等待用户从键盘输入两个实数(中间用空格分隔,最后以输入回车结束)分别存放在为变量 x 与 y 所分配的存储单元中。输入完后将继续往下执行,否则一直在此等待输入。

在这个语句中,cin 为标准输入流。

第四个语句是赋值语句,其功能是计算斜边长(其中 sqrt()是求平方根函数)。执行到这个语句时,就进行计算,并将计算结果存放到变量 c 中。

第五个语句是输出语句。执行到这个语句时,将显示输出的结果。例如,如果在执行第三个语句时输入(有下划线的为输入部分)

```
input a and b :3.0 4.0<回车>
```

则在执行第五个语句时将显示输出

```
hypotenuse=5
```

在这个语句中,cout 为标准输出流,首先输出一个字符串"hypotenuse=",然后输出变量 c 的值,最后输出一个换行。其中 endl 表示换行。

在模块化程序设计中,往往要把一个大程序按人们能理解的大小规模进行分解。由于经过分解后的各模块比较小,因此容易实现,也容易调试。按照人类思维的特点,按功能来划分模块最为自然。在按功能划分模块时,要求各模块的功能尽量单一,各模块之间的联系尽量少。满足这些要求的模块有以下几个优点:① 其可读性和可理解性都比较好;② 各模块间的接口关系比较简单;③ 当要修改某一功能时,只涉及到一个模块;④ 其他应用程序可以充分利用已有的一些模块。

如果进一步将上述问题划分成两个子问题(即两个子功能):一是从键盘输入两条直角边

的长；二是计算并输出斜边长。在例 1-3 的程序中，这两个子功能是由一个函数（即主函数 main()）具体实现的。根据模块化程序设计的要求，可以将上述问题中的两个子功能分别由两个函数来实现。

例 1-4 编写一个函数，其功能是：根据给定的两条直角边，计算并输出该直角三角形的斜边长，再编写一个主函数，从键盘输入一个直角三角形的两条直角边长 a 与 b，然后调用计算并输出直角三角形斜边长的函数。其 C++ 程序如下：

```
#include <iostream>                          //包含头文件 iostream
#include <cmath>                             //包含头文件 cmath
using namespace std;                         //使用命名空间 std
void right_angled_triangle(float x, float y) //计算并输出斜边长的函数
{ float z;                                   //定义一个实型变量
  z=sqrt(x*x+y*y);                           //计算斜边长
  cout <<" hypotenuse=" <<z <<endl;          //输出斜边长 c
}
int main()                                   //主函数
{ float a,b;                                 //定义两个实型变量
  cout <<"input a and b :";                  //给出输入提示
  cin >>a >>b;                               //输入直角三角形两条直角边 a 与 b 的值
  right_angled_triangle(a, b);               //调用计算并输出斜边长的函数
  return 0;                                  //程序正常终止则返回 0 值
}
```

在这个 C++ 程序中，包含了两个函数：一个是根据给定的两条直角边计算并输出直角三角形斜边长的函数 right_angled_triangle()；一个是主函数 main()。在主函数中，首先从键盘输入直角三角形中两条直角边 a 与 b 的值，然后调用计算并输出斜边长的函数 right_angled_triangle()。

在例 1-4 中，主函数中定义的变量 a 与 b 可以通过调用关系传递给函数 right_angled_triangle()使用，并且函数 right_angled_triangle()也可以被其他函数调用（当需要时）。在这种机制下，函数中定义的数据是公用的，任何函数都可以通过调用关系操作这组数据。因此，这是一种面向过程的机制，这种机制在设计大型程序时是容易出错的。

C++ 语言支持面向对象的机制，将数据和对数据的操作封装在一个对象中，各对象之间各自相对独立，互不干扰。在这种机制下，程序设计者只需要做两件事情：一是确定将哪些数据和操作封装在一起，即设计对象；二是确定如何通知对象活动，完成对相应数据的操作。下面的例子就是采用面向对象的方法来解决上述同样的问题。

例 1-5 采用面向对象的方法，从键盘输入一个直角三角形的两条直角边长，然后计算并输出直角三角形斜边长。其 C++ 程序如下：

```
#include <iostream>                          //包含头文件 iostream
#include <cmath>                             //包含头文件 cmath
using namespace std;                         //使用命名空间 std
class Right_Angled_Triangle                  //声明一个类，其类名为 Right_Angled_Triangle
```

```
{ private:                                    //以下为类中的私有部分
    float right_a;                            //私有数据变量 right_a 为实型(直角边)
    float right_b;                            //私有数据变量 right_b 为实型(直角边)
    float hypotenuse;                         //私有数据变量 hypotenuse 为实型(斜边)
 public:                                      //以下为类中的公用部分
    void set_rights()                         //定义公用函数 set_rights()
    { cout <<"input right_a and right_b :";   // 给出输入提示
      cin >>right_a;                          //输入直角边长
      cin >>right_b;                          //输入直角边长
    }
    void out_hypotenuse()//定义公用函数 out_hypotenuse()
    { hypotenuse=sqrt(right_a* right_a+right_b* right_b);  // 计算斜边长
      cout <<"hypotenuse=" <<hypotenuse <<endl;            // 输出斜边长
    };
};                                            //类的声明结束
//定义 Right_Angled_Triangle 的两个变量(称为对象)triangle1 与 triangle2
Right_Angled_Triangle triangle1, triangle2;
int main()                                    //主函数
{ triangle1.set_rights();                     //调用对象 triangle1 的函数 set_rights()
  triangle1.out_hypotenuse();                 //调用对象 triangle1 的函数 out_hypotenuse()
  triangle2.set_rights();                     //调用对象 triangle2 的函数 set_rights()
  triangle2.out_hypotenuse();                 //调用对象 triangle2 的函数 out_hypotenuse()
  return 0;
}
```

上述 C++ 程序的第 4 行声明了一个类,其类名为 Right_Angled_Triangle。类描述的是具有相似性质的一组对象。在 C++ 中,一个类由数据成员和对数据成员进行操作的成员函数组成。在上述 C++ 程序所声明的类中,包含的成员有:三个实型的私有数据变量(right_a, right_b 与 hypotenuse)以及两个公用的成员函数(set_rights()与 out_hypotenuse())。

在类中,所有的数据成员和成员函数分为两种:private(私有的)和 public(公用的)。被指定为公用的成员(数据或函数),既可以被本类中的成员函数调用,也可以被类外的语句所调用;而被指定为私有的成员(数据或函数),只能被本类中的成员函数调用,不能被类外的语句调用;在上述程序中,将所有数据成员(right_a,right_b 与 hypotenuse)均指定为私有的,所有成员函数(set_rights()与 out_hypotenuse())均指定为公用的。当然,根据需要也可以将一部分数据成员和成员函数指定为私有,而将另一部分数据成员和成员函数指定为公用。在一般情况下,总是将所有的数据成员指定为私有,以实现信息隐蔽。

类的声明结束后,定义类 Right_Angled_Triangle 的两个变量(称为对象)triangle1 与 triangle2。在本例中,这两个变量可以分别表示一个具体的直角三角形。在面向对象方法中,一个具体对象称为类的实例。例如,两条直角边分别为 3.0 和 4.0 的直角三角形就是类 Right_Angled_Triangle 的一个实例,具体的数据(直角边 right_a,right_b 与斜边 hypotenuse)可以存放在所定义的类 Right_Angled_Triangle 的变量中(如 triangle1 或 triangle2)。由此可知,类的一个变量可以存放类的一个实例。