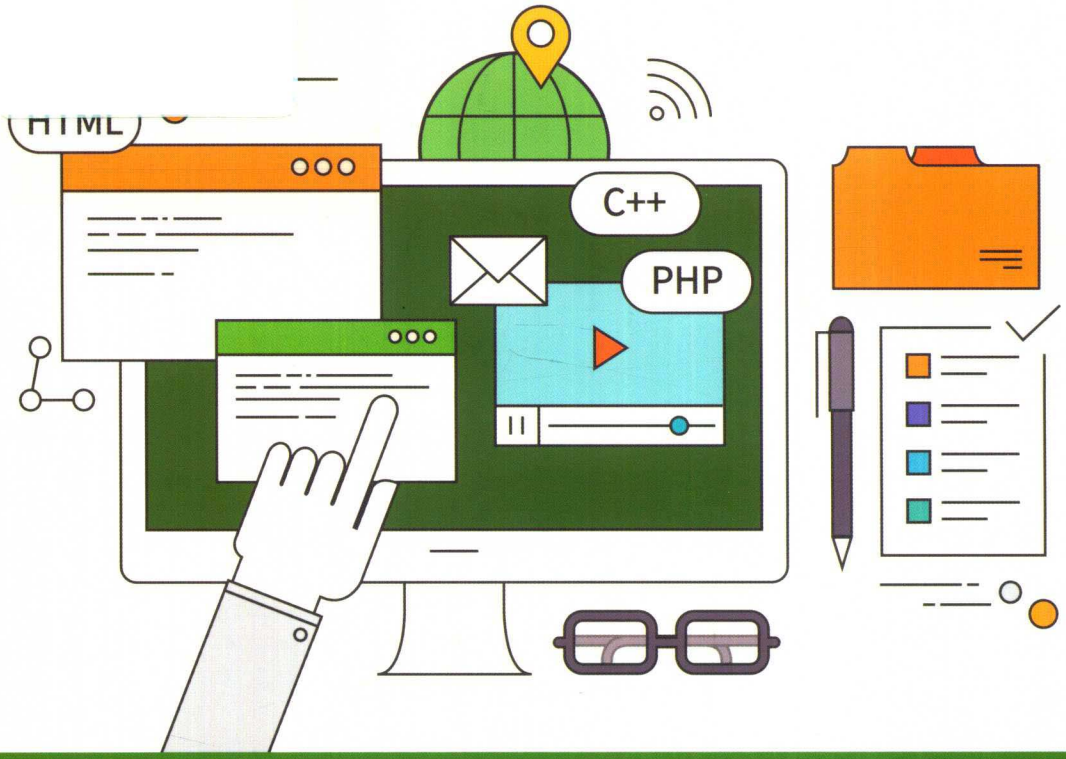
“十二五”普通高等教育本科国家规划教材
高等教育计算机学科“应用型”教材



C/C++程序设计教程——面向对象分册（第3版）

郑秋生◎主编　王黎明◎主审

中国工信出版集团
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

育本科国家规划教材
科“应用型”教材

C/C++程序设计教程——面向对象分册（第3版）

郑秋生　主　编
刘凤华　王文奇　李晓宇　副主编
王黎明　主　审

電子工業出版社
Publishing House of Electronics Industry
北京・BEIJING

内 容 简 介

本书属于C++程序设计系列教材，分为面向过程和面向对象两个分册。

面向对象分册详细阐述了C++语言中面向对象程序设计的语法和思想。主要内容包括类与对象、继承与派生、虚函数与多态性、异常处理、模板和STL标准模板库等内容。书中通过流行的UML工具描述C++类，内容讲解清晰，实例丰富，力避代码复杂冗长，注重程序设计思想。简短的实例和UML图特别有助于初学者更好地理解、把握解决问题的精髓，帮助读者快速掌握面向对象程序设计的基本方法。

本书的特点是实例丰富，重点突出，叙述深入浅出，分析问题透彻，既有完整的语法，又有大量的实例，突出程序设计的思想和方法，将C语言程序设计和C++程序设计有机地统一起来。本书特别适合作为计算机学科各应用型本科、专科的C语言程序设计和C++语言程序设计教材，也可作为其他理工科各专业的教材，还适合作为相关专业技术人员的自学参考书。

图书在版编目（CIP）数据

C/C++程序设计教程. 面向对象分册/郑秋生主编. —3版. —北京：电子工业出版社，2019.1
ISBN 978-7-121-33047-6

Ⅰ. ①C… Ⅱ. ①郑… Ⅲ. ①C语言－程序设计－高等学校－教材②C++语言－程序设计－高等学校－教材 Ⅳ. ①TP312.8

中国版本图书馆CIP数据核字（2017）第280705号

策划编辑：张贵芹
责任编辑：康 霞
印 刷：三河市君旺印务有限公司
装 订：三河市君旺印务有限公司
出版发行：电子工业出版社
北京市海淀区万寿路173信箱 邮编 100036
开 本：787×1092 1/16 印张：22.75 字数：582.4千字
版 次：2008年2月第1版
2019年1月第3版
印 次：2019年1月第1次印刷
定 价：47.00元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至zlts@phei.com.cn，盗版侵权举报请发邮件至dbqq@phei.com.cn。

本书咨询联系方式：（010）88254511；82470125@qq.com。

前　言

本书主要作者都是有着丰富教学经验的一线教师，从事 C/C++语言程序设计课程教学多年，深知学生在学习 C++语言程序设计这门课程后，对程序设计方法、算法设计、调试程序、习题解答的茫然和疑惑，因此本书在介绍理论知识、相关概念和语言语法时，始终强调其在程序设计中的作用，使语言语法与程序设计相结合。同类书籍大部分偏重于对语言语法和概念的介绍，虽然在书中有针对一个语法和知识点的程序例子，但学生对每章内容在实际程序设计中的作用缺乏了解，而本书每章节后都附有针对性较强的应用实例分析，尽可能使初学者在学习每章的内容后，拿到题目就能够独立进行程序设计、解决实际问题，而不至于无从下手。本书有以下五个鲜明特点。

第一，改变了传统的教学模式：先讲 C 语言程序设计，再讲 C++对 C 语言的扩展、面向对象的程序设计。本书将 C/C++ 语言的学习很好地融在一起，让读者把面向过程和面向对象的程序设计方法有机地结合在一起，面向过程和面向对象两分册都统一使用 Visual Studio 2013 编译器。

第二，改变了传统教材以语言、语法学习为重点的缺陷，从基本的语言、语法学习上升到程序的“设计、算法、编程、调试”层次。为了让学生更好地掌握程序开发思想、方法和算法，书中提供了大量简短、精辟的代码有助于初学者学习解决问题的精髓，并且在每章后都有一节关于程序综合设计的内容，有一个或多个较大的程序，帮助读者更好地掌握程序设计方法和解决实际问题的能力。

第三，强调程序的设计方法，大量例题有流程图、N-S 图和 UML 图，即突出程序的算法和设计而不仅仅是语法和编程，培养学生程序设计能力和程序调试技能，养成好的编程习惯，为专业程序员的培养打下好的基础。

第四，培养学生面向对象程序设计的能力，引导学生建立程序设计的大局观，帮助学生掌握从客观事物中抽象出 C++类的方法。通过系统的学习，使学生的编程能力上一个台阶，具备解决复杂问题的程序设计能力。

第五，根据当前实际大型软件项目开发的需要，加大了异常处理、模板等内容，新增 STL 标准模板库，并通过流行的 UML 工具设计 C++类。

本书的编写充分考虑了目前应用型本科 C/C++语言程序设计课程教学的实际情况和存在的问题：

第一，学生在大一阶段的基础课程较多，不可能投入过多的精力来学习本门课程。

第二，大学生对这门课学习的期望值很高，但对学习时可能遇到的困难估计不足。

第三，大学生现有的上机实践条件大大改善，特别有利于贯彻先进的精讲多练的教学思想。

第四，学生学会了语言的语法，仍不具备解决实际问题的能力，学生的程序设计、算法设计、编程、调试能力相对较差。

本书正是考虑了学生的这些实际问题，进而精心编写的一套面向应用型本科的 C/C++语

言程序设计教程，特别适合于分两个学期系统讲授 C/C++语言程序设计。第一学期讲授面向过程分册，第二学期讲授面向对象分册。

本分册共分 9 章，第 1～4 章主要阐述面向对象程序设计的重要概念：类和对象、继承与派生、虚函数与多态性。第 5 章介绍输入/输出流技术，第 6 章主要介绍异常的概念、异常的产生及异常的处理机制，第 7 章和第 8 章介绍模板和 STL 标准模板库，第 9 章主要讲述面向对象的分析与设计方法，并以实例形式详细介绍如何用 C++语言进行程序设计。

为了方便使用本教材的教师备课，我们还提供了配套的电子教案公开放在网站上，供任课教师自由下载使用。相信我们多年的教学经验会对广大师生的教和学有所帮助。建议本分册的教学学时为 60 学时，其中理论教学 44 学时，课内上机实践 16 学时，课外上机不少于 32 学时。

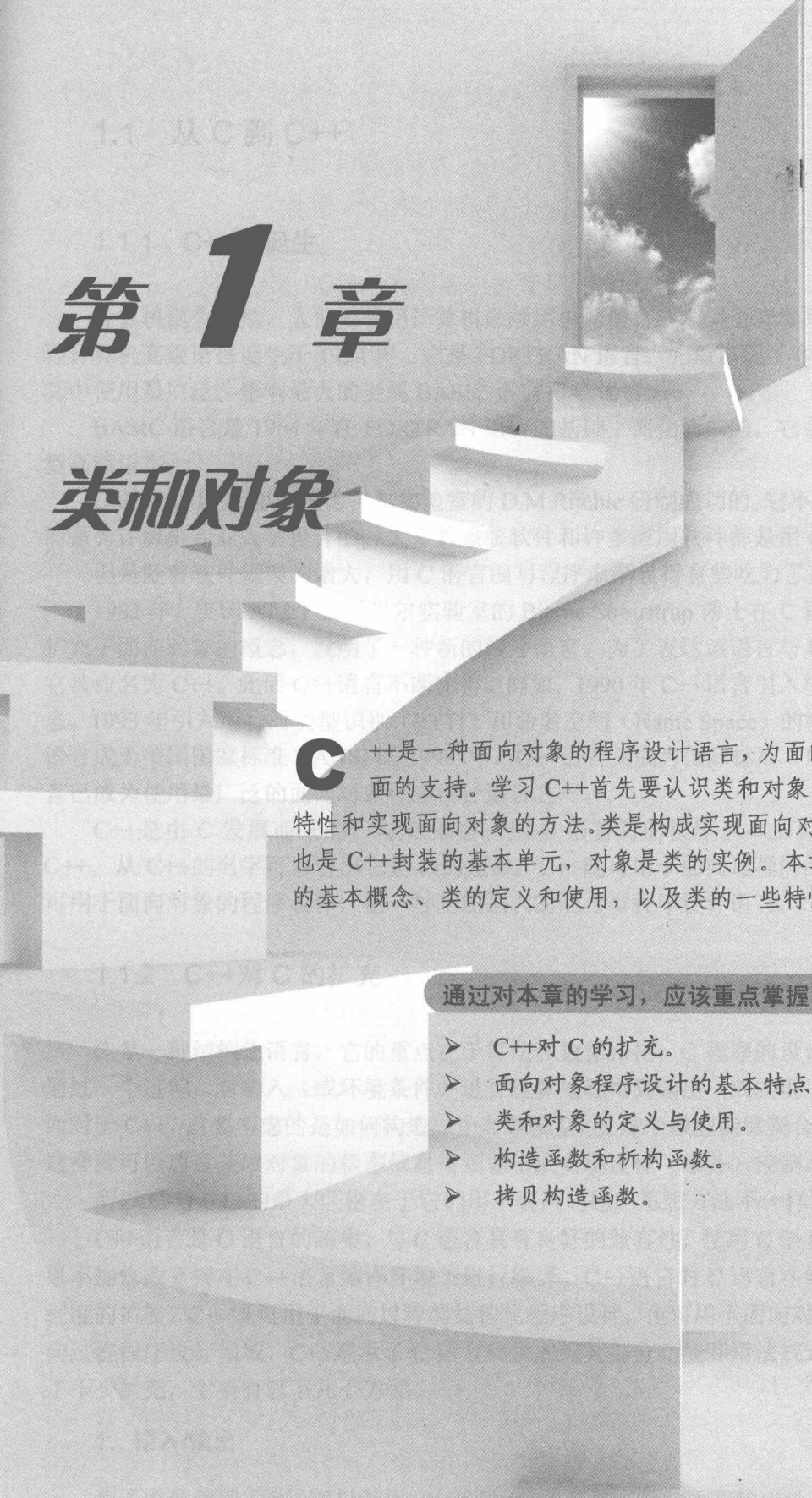
本书的编写得到了河南省计算机学会的大力支持，其组织河南多所高校编写了高等教育计算机学科“应用型”系列教材。参编本书的高校有中原工学院、郑州大学、河南科技大学、郑州轻工业学院。

本书由郑秋生任主编，第 1 章和附录 A 由张晓玲（河南科技大学）和夏敏捷编写，第 2 章和第 3 章由王海龙和苏安婕编写，第 4 章由王文奇编写，第 5 章由王璐编写，第 6 章和第 7 章由宋宝卫（郑州轻工业学院）编写，第 8 章由李晓宇（郑州大学）编写，第 9 章由刘凤华编写，附录 B 和 C 由郑秋生编写。全书最终由中原工学院郑秋生修改并统稿。郑州大学王黎明为本书提出了改进意见，在此谨向他们表示衷心的感谢。

由于编者水平有限，时间仓促，书中难免有错，敬请广大读者批评指正，在此表示感谢。作者邮箱：zqs@zzti.edu.cn。

编　　者

目　录



第1章 类和对象

C++是一种面向对象的程序设计语言，为面向对象技术提供了全面的支持。学习C++首先要认识类和对象，掌握它面向对象的特性和实现面向对象的方法。类是构成实现面向对象程序设计的基础，也是C++封装的基本单元，对象是类的实例。本章主要介绍类和对象的基本概念、类的定义和使用，以及类的一些特性。

通过对本章的学习，应该重点掌握以下内容：

- C++对C的扩充。
- 面向对象程序设计的基本特点。
- 类和对象的定义与使用。
- 构造函数和析构函数。
- 拷贝构造函数。

1.1 从 C 到 C++

1.1.1 C++的诞生

计算机诞生初期，人们要使用计算机必须用机器语言或汇编语言编写程序。世界上第一种计算机高级语言诞生于 1954 年，它是 FORTRAN 语言，先后出现了多种计算机高级语言，其中使用最广泛、影响最大的当属 BASIC 语言和 C 语言。

BASIC 语言是 1964 年在 FORTRAN 语言的基础上简化而成的，它是为初学者设计的小型高级语言。

C 语言是 1972 年由美国贝尔实验室的 D.M.Ritchie 研制成功的。它不是为初学者设计的，而是为计算机专业人员设计的。大多数系统软件和许多应用软件都是用 C 语言编写的。

但是随着软件规模的增大，用 C 语言编写程序渐渐显得有些吃力了。

1982 年，美国 AT&T 公司贝尔实验室的 Bjarne Stroustrup 博士在 C 语言的基础上引入并扩充了面向对象的概念，发明了一种新的程序语言。为了表达该语言与 C 语言的渊源关系，它被命名为 C++。此后 C++语言不断完善。例如，1990 年 C++语言引入模板和异常处理的概念。1993 年引入运行时类型识别（RTTI）和命名空间（Name Space）的概念。1997 年，C++语言成为美国国家标准（ANSI）。1998 年，C++语言又成为国际标准（ISO）。目前，C++语言已成为使用最广泛的面向对象程序设计语言之一。

C++是由 C 发展而来的，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++。从 C++的名字可以看出它是 C 的超集。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，是一种功能强大的混合型程序设计语言。

1.1.2 C++对 C 的扩充

C 是一种结构化语言，它的重点在于算法和数据结构。C 程序的设计首要考虑的是如何通过一个过程，对输入（或环境条件）进行运算处理得到输出（或实现过程（事务）控制），而对于 C++，首要考虑的是如何构造一个对象模型，让这个模型能够契合与之对应的问题域，这样就可以通过获取对象的状态信息得到输出或实现过程（事务）控制。

所以 C 与 C++的最大区别在于它们用于解决问题的思想方法不一样。

C++语言是 C 语言的超集，与 C 语言具有良好的兼容性，使用 C 语言编写的程序几乎可以不加修改直接在 C++语言编译环境下进行编译。C++语言对 C 语言在结构化方面做了一定程度的扩展。C++既可用于面向过程的结构化程序设计，也可用于面向对象的程序设计。在面向过程程序设计领域，C++继承了 C 语言提供的绝大部分功能和语法规定，并在此基础上做了不少扩充，主要有以下几个方面。

1. 输入/输出

为了方便使用，除了可以利用 printf 和 scanf 函数进行输入和输出外，还增加了标准输入流/输出流 cin 和 cout。它们是在头文件 iostream 中定义的，标准流是不需要打开文件和关闭

文件就能直接操作的流式文件，而标准输入流是指从键盘上输入的数据，标准输出流是指向屏幕输出的数据流。

1）用 cout 进行输出

输出操作使用的是输出流对象 cout。采取的运算符是“<<”，称作“输出运算符”或“插入运算符”。

可以在一个输出语句中使用多个“<<”运算符将多个输出项插入到输出流 cout 中，“<<”运算符的结合方向为自左至右，因此多个输出项按自左至右的顺序插入到输出流中。可以使用多个“<<”将一大堆数据像糖葫芦一样串起来，然后再用 cout 输出。

格式：cout <<表达式 1[<<后跟一个表达式]

用 cout 和“<<”可以输出任何类型的数据，例如：

```
int a;
float b;
char name[10];
cout<<a<<b<<name<<"\n ";
```

也可以不用“\n”控制换行，在头文件 iostream 中定义了控制符 endl 代表回车换行操作，作用与“\n”相同。endl 的含义是 end of line，表示结束一行。

可以看到输出时并未指定数据的类型（如整型、浮点型等），系统会自动按数据的类型进行输出。这比用 printf 函数方便，在 printf 函数中要指定输出格式符（如%d, %f, %c 等）。

2）用 cin 进行输入

在 C++中，输入操作使用的是输入流对象 cin，右移运算符“>>”的含义被重写，称作“输入运算符”或“提取运算符”。

格式：cin>>变量 1>>变量 2…

“>>”是 C++的提取运算符，表示从标准输入设备取得数据，赋予其后的变量。从键盘输入数值数据时，两个数据之间用空格分隔或用回车分隔。

```
int a;
float b;
cin>>a>>b;
```

可以从键盘输入 60　88.99

a 和 b 分别获得 60 和 88.99。用 cin 和“>>”输入数据同样不需要在本语句中指定变量的格式控制符。

2．const 修饰符与#define 命令

用#define 命令定义符号常量是 C 语言所采用的方法，C++把它保留下来是为了和 C 兼容。C++的程序员一般喜欢用 const 定义常变量。虽然二者实现的方法不同，但从使用的角度看，可以认为都是用了一个标识符代表一个常量。

#define 与 const 的不同之处在于用#define 命令定义的符号常量只是用一个符号代替一个字符串，在预编译时把所有的符号常量替换所指定的字符串，它没有类型，在内存中并不存在以符号常量命名的存储单元；而用 const 定义的常变量具有变量的特征，它具有类型，在内存中存在以它命名的存储单元，可以用 sizeof 运算符测出其长度。与一般变量唯一的不同

是指定变量的值不能改变。

C++推荐使用 const 修饰符，因为 const 定义的常变量具有数据类型，而宏常量没有，编译器可以对 const 常量进行类型安全检查，而对宏变量只是进行字符替换，这就可能导致预料不到的错误。

3. 运算符 new 与 delete

在内存管理上，常常需要动态地分配和撤销内存空间。malloc/free 是 C/C++语言的标准库函数，而 new/delete 是 C++的运算符。C++推荐使用 new/delete，不仅是因为 new 能够自动分配空间大小，更为重要的一点在于对于用户自定义的对象而言，用 malloc/free 无法满足动态管理对象的要求。对象在创建的同时要自动执行构造函数，对象在消亡之前要自动执行析构函数。由于 malloc/free 是库函数而不是运算符，不在编译器控制权限之内，不能够把执行构造函数和析构函数的任务强加于 malloc/free，因此 C++需要一个能对对象完成动态内存分配和初始化工作的运算符 new，以及一个能对对象完成清理与释放内存工作的运算符 delete，简而言之，new/delete 能对对象进行构造和析构函数的调用，进而对内存进行更加详细的工作，而 malloc/free 不能。

new 运算符使用的一般格式为：

new 类型 [初值];

用 new 分配数组空间时不能指定初值。

例如：

```
new   int;                  //开辟一个存放整数的空间，返回一个整型数据的指针
new   int(80);              //开辟一个存放整数的空间，并指定该整数的初值为 80
new   char[10];             //开辟一个存放字符数组的空间，该数组有 10 个数组元素
                            //返回一个字符数据的指针
new   int[3][4];            //开辟一个存放二维整型数组的空间，该数组大小为 3*4
float  *p=new float(3.14159); //开辟一个存放实数的空间，该实数的初值为 3.14159
```

delete 运算符使用的一般格式为：

delete [ ] 指针变量;

如果要撤销上面用 new 开辟的存放实数的空间，应该用：

delete p;

前面用 new char[10]开辟的空间，如果把返回的指针赋给了指针变量 pt，则应该用以下形式的 delete 运算符撤销所开辟的空间：

delete [ ] pt;

4. 函数的重载

在前面的程序中用到了插入运算符“<<”和提取运算符“>>”。这两个运算符本来是 C 和 C++位运算中的左移运算符和右移运算符，现在 C++又把它作为输入/输出运算符，即一个运算符用于不同场合，有不同含义，这就叫运算符的“重载”（overloading），即重新赋予它新的含义，其实就是“一物多用”。

在 C++中，函数也可以重载。C++允许用一个函数名定义多个函数，这些函数的参数个数或者参数类型不相同。用一个函数名实现不同的功能，就是函数的重载。C++语言可实现

函数重载，即多个函数在同一作用域可以用相同的函数名，编译器在编译时可以根据实参的类型或个数来选择应该调用的函数。

参数的个数和类型也可以不同。重载函数的参数个数或类型必须至少有其中一个不同，函数的返回值类型可以相同也可以不同。

注意：不允许函数参数个数、参数类型都相同，只是函数返回值类型不同。因为系统无法从调用形式上判断调用与哪个函数相匹配。

5. 默认参数

一般情况下，在函数调用时形参从实参那里取得值，因此实参的个数应与形参相同。但是有时多次调用同一个函数时用的是同样的实参值，C++允许为函数的参数设置默认值，这时调用函数时，如果没有实参，就以默认值作为实参值。

格式：形参类型 形参变量名 = 常数

功能：调用函数时，如果没有实参，就以常数作为该形参的值；如果有实参，则仍以实参的值作为该形参的值。

例如：编写计算圆柱体体积的函数 float volume (float h, float r = 12.5)。

调用可以采用以下任何一种形式：

```
volume( 45.6);          //相当于 volume( 45.6,12.5)
volume( 32.5, 10.5);    //h 的值为 32.5，r 的值为 10.5
```

函数参数结合从左到右，用第一种方式调用时，只有一个实参，圆半径的值取默认值 12.5；用第二种方式调用时，有两个实参，圆半径的值取实参的值 10.5。

注意：

（1）如果用函数原型声明，则只要在函数原型声明中定义形参的默认值即可。

（2）有默认值的形参必须放在形参表的右边，不允许无默认参数值和有默认参数值的形参交错排列。

```
int   f1(float a,int b=0,int c,char d='a');        //不正确
int   f2(float a,int c,int b=0,char d='a');        //正确
```

如果要调用上面的 f2 函数，则可以采取下面的形式：

```
f2(3.6,6,9, 'x')          //形参的值全部从实参得到
f2(3.6,6,9)               //最后一个形参的值取默认值'a'
f2(3.6,6)                 //最后两个形参的值取默认值，b=0,d='a'
```

（3）一个函数名不能同时用于重载函数和带默认形参值的函数。当调用函数时，如少写一个参数，则系统无法判断是利用重载函数还是利用带默认参数值的函数，会出现二义性。

6. 作用域运算符

在 C++语言中增加了作用域标识符（或称为名字解析运算符）::，用于解决局部变量与全局变量的同名重复问题。在局部变量的作用域内可用作用域标识符:: 对被其隐藏的同名全局变量进行访问。下面是一个简单的例子：

```
int x=0;
int test(int x)
```

```
{
    x=5;        //此处引用局部变量
    ::x=9;      //此处引用全局变量
}
```

7. 字符串变量

除了可以使用字符数组处理字符串外，C++还提供了字符串类类型 string，实际上它不是C++的基本类型，它是在 C++标准库中声明的一个字符串类，程序可以用它定义对象。

定义字符串变量格式如下。

格式：string 变量名表;

可以在定义变量时用字符串常量为变量赋初值：

```
string 变量名 = 字符串常量
string   string1;                  //定义 string1 为字符串变量
string   string2="china";          //定义 string2 为字符串变量同时对其初始化
```

注意：如用字符串变量，则在程序开始处要用包含语句把 C++标准库的 string 头文件包含进来，即应加上#include<string>。有关字符串变量的具体使用见第 2 章中的 string 类。

8. 变量的引用

C++提供了为变量取别名的功能，这就是变量的引用（reference）。引用是 C++对 C 的一个重要扩充。

1）引用的概念和简单使用

格式：类型 &变量 1 = 变量 2

变量 2 是在此之前已经定义过的变量，且与变量 1 的类型相同。这里为变量 2 定义一个别名变量 1，在程序里变量 1 和变量 2 就是同一个变量。

注意：两个变量不能用同一个别名。

例如：

```
int a = 3 ,b =4;
int &c = a;              // c 是 a 的别名
int &c = b;              // 错误的用法
```

一个变量可以有多个别名。

例如：

```
int a = 3;
int & b= a;
int & c= b;          //变量 a 有两个别名 b 和 c
```

2）引用的几点说明

（1）引用并不是一种独立的数据类型，它必须与某一种类型的数据相联系。声明引用时必须指定它代表的是哪个变量，即对它进行初始化。例如：

```
int a;
int   &b=a;              //正确，指定 b 是整型变量 a 的别名
```

```
int   &b;              //错误，没有指定 b 是哪个变量的别名
float  a;
int &b=a;              //错误，声明 b 是整型变量的别名，而 a 不是整型变量
```

注意：不要把声明语句“int &b=a;”理解为“将变量 a 的值赋给引用 b”，它的作用是使 b 成为 a 的引用，即 a 的别名。

（2）引用与其所代表的变量共享同一个内存单元，系统并不为引用另外分配存储空间。实际上，编译系统使引用和其代表的变量具有相同的地址。

（3）当看到&a 这样的形式时，怎样区别是声明引用变量还是取地址的操作呢？请记住，当&a 的前面有类型符时（如 int &a），它必然是对引用的声明；如果前面没有类型符（如 p=&a），此时的&是取地址运算符。

（4）对引用的初始化，可以用一个变量名，也可以用另一个引用。例如：

```
int   a=3;
int &b=a;          //声明 b 是整型变量 a 的别名
int &c=b;          //声明 c 是整型引用变量 b 的别名
```

这是合法的，这样，整型变量 a 就有两个别名 b 和 c。

（5）引用初始化后不能再被重新声明为另一个变量的别名。例如：

```
int   a=3,b=4;
int   &c=a;        //正确，声明 c 为整型变量 a 的别名
      c=&b;        //错误，企图使 c 改变为整型变量 b 的别名
int   &c=b;        //错误，企图重新声明 c 为整型变量 b 的别名
```

9. 内联函数

C++提供了一种机制，在编译时，将被调用的函数代码嵌入调用函数代码中，在执行函数时省去了调用环节，提高了函数的执行速度。这种机制称为内置函数，也称内联函数。

格式：

```
inline 函数类型 函数名(形参表)
{ 函数体 }
```

inline 是 C++的关键字，在编译时，编译程序会把这个函数嵌入主调函数的函数体中。

调用格式：函数名（实参表）

【例 1.1】 计算 3 个整数中的大数。

```
#include<iostream>
using namespace std;
inline int max(int a,int b,int c)    //这是一个内联函数，求 3 个整数中的最大者
{
    if (b>a)   a=b;
    if (c>a)   a=c;
    return a;
}
int   main()
```

```
{
    int i=7,j=10,k=25,m;
    m=max(i,j,k);
    cout<<"max="<<m<<endl;
    return 0;
}
```

程序运行结果为：

```
max=25
```

由于在定义函数时指定它是内联函数，因此编译系统在遇到函数调用 max(i,j,k) 时就用 max 函数体的代码代替 max(i,j,k)，同时用实参代替形参。

调用语句 m= max(i,j,k) 就被置换成：

```
{
a=i ; b = j ; c= k;
if ( b>a)     a=b;
if ( c>a)    a=c;
m=a;
}
```

使用内联函数可以节省程序的运行时间，但增加了目标程序的长度。假设要调用 10 次 max 函数，则在编译时先后 10 次将 max 的代码复制并插入 main 函数，大大增加了 main 函数的长度，所以在使用时要衡量时间和空间上的得失。内置函数只用于规模很小而使用频繁的函数，可大大提高运行效率。

1.1.3 面向对象和面向过程的区别

1. 面向对象和面向过程的具体区别

面向过程就是分析出解决问题所需要的步骤，然后用函数把这些步骤一步一步实现，使用的时候一个一个依次调用就可以了。

面向对象是把构成问题事务分解成各个对象，建立对象的目的不是为了完成一个步骤，而是为了描述某个事务在整个解决问题步骤中的行为。

例如，五子棋，面向过程的设计思路就是首先分析问题的步骤：

（1）开始游戏；

（2）黑子先走；

（3）绘制画面；

（4）判断输赢；

（5）轮到白子；

（6）绘制画面；

（7）判断输赢；

（8）返回步骤（2）；

（9）输出最后结果。