新编计算机专业
重点课程辅导丛书

新编C++语言习题与解析

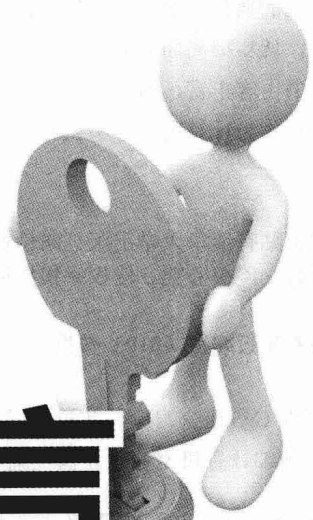
李春葆 喻丹丹 曾平 曾慧 编著

名师执笔 百万册畅销书全面升级

知识体系完整，以典型题目分析带动能力培养
应对：课程复习、考研、程序员面试、等级考试

清华大学出版社

新编计算机专业
重点课程辅导丛书



新编C++语言习题与解析

李春葆 喻丹丹 曾平 曾慧 编著

清华大学出版社
北京

内 容 简 介

本书根据计算机专业C++语言程序设计课程的教学大纲编写，全书共分10章，分别介绍C++语言概述、类和对象、引用、友元、运算符重载、模板、继承和派生、多态性和虚函数、C++的I/O流库和异常处理。每章由基本知识点和例题分析组成，前者高度概括和疏理本章应重点掌握的相关知识；后者详尽地解析精选的典型习题。本书将使学生充分掌握C++程序设计课程求解问题的技巧与方法，深化对基本概念的理解，切实提高面向对象的程序设计能力。

本书内容丰富，习题覆盖面广，不仅可以作为计算机专业本、专科C++语言程序设计课程的学习参考书，也可作为计算机水平考试和等级考试者的参考书。

图书在版编目（CIP）数据

新编C++语言习题与解析 / 李春葆等编著. —北京：清华大学出版社，2013.5
（新编计算机专业重点课程辅导丛书）

ISBN 978-7-302-30620-7

Ⅰ. ①新… Ⅱ. ①李… Ⅲ. ①C语言－程序设计－高等学校－教学参考资料 Ⅳ. ①TP312

中国版本图书馆CIP数据核字（2012）第272828号

责任编辑：夏非彼
封面设计：王　翔
责任校对：闫秀华
责任印制：何　芊

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦A座　　**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：三河市李旗庄少明印装厂
经　　销：全国新华书店
开　　本：190mm×260mm　　**印　　张**：21　　**字　　数**：534千字
版　　次：2013年5月第1版　　**印　　次**：2013年5月第1次印刷
印　　数：1～4000
定　　价：38.00元

产品编号：049938-01

《新编计算机专业重点课程辅导丛书》丛书序

“计算机专业教学辅导丛书——习题与解析系列”自 1999 年推出以来，一直被许多院校采用并受到普遍好评，广大师生也给我们反馈了不少中肯的改进建议，总印数超过百万册。这些都是我们修订、扩充该丛书的动力之源。同时，计算机科学与技术的持续发展和不断演化，使得传统的计算机专业教学模式也随之扩充与革新，随着计算机教材改革的不断深化，如何促进学生将理论用于实践，提高分析与动手能力，以及通过实践加深对理论的理解程度，都是 21 世纪计算机教学亟待解决的问题。正是基于这些需求，经过对原有丛书的使用情况的深入调研，并组织专家和一线教师对自身教学经验进行认真总结、提炼之后，我们重新修订了这套“21 世纪计算机专业重点课程辅导丛书”。

依据各门课程的最新教学大纲，对原有图书内容进行了全面的修订和扩充，使其更加完备、充实。修订之后的新版丛书几乎囊括了计算机专业的各个重点科目，与现行计算机专业课程体系更加吻合。

“新编计算机专业重点课程辅导丛书”包括：

- 《新编 C 语言习题与解析》
- 《新编 C++语言习题与解析》
- 《新编 Java 语言习题与解析》
- 《新编数据结构习题与解析》
- 《新编数据库原理习题与解析》
- 《新编操作系统习题与解析》
- 《新编计算机组成原理习题与解析》
- 《新编计算机网络习题与解析》

本套丛书具有如下特点：

☑ 以典型题目分析带动能力培养

本丛书注重以典型题目的分析为突破口，点拨解题思路，强化各知识点的灵活运用，启发解题灵感。所有例题不仅给出了参考答案，还给出了详细透彻的分析过程，便于读者在解题过程中举一反三，触类旁通，从而提高分析问题和解决问题的能力。

☑ 全面复习，形成知识体系

本丛书以权威教材为依托，对各知识点进行了全面、深入地剖析和提炼，构成了一个完备的知识体系。在各类考试中，一个微小的知识漏洞，就可能造成无法弥补的损失，因此复习必须全面扎实。

☑ 把握知识间的内在联系，拓展创新思维

把握知识点之间的关系，这样，掌握的知识就能变“活”。本丛书通过对知识点的分解，找出贯穿于各知识点之间的内在联系，并配上相关的例题，阐明如何利用这些内在联系解

决问题，从而做到不仅授人以“鱼”，更注重授人以“渔”。

☑ **紧贴计算机专业考研大纲要求，提高考研成绩**

自 2009 年以来计算机科学与技术专业实行全国联考，统一命题和阅卷，联考内容涵盖数据结构、计算机组成原理、操作系统和计算机网络。本丛书的相关课程均以最新联考大纲为基础进行编写，并收录了最新的联考试题。另外，各高校计算机专业研究生复试的常见课程有高级语言程序设计和数据库原理等，这些课程的内容也涵盖在本丛书中。

本套丛书由长期坚持在教学第一线的教授和副教授编写，他（她）们结合自己的教学经验和见解，把多年的教学实践成果无私奉献给读者，希望能够提高学生素质、培养学生的综合分析能力。

如果说科学技术的飞速发展是 21 世纪的一个重要特征，那么，教学改革将是 21 世纪教育工作不变的主题，也是需要我们不断探索的课题。要紧跟教学改革，不断更新，真正满足新形势下的教学需求，还需要我们不断地努力实践和完善。本套教材虽然经过细致的编写与校订，仍然难免有疏漏和不足之处，需要不断地补充、修订和完善。我们热情欢迎使用本套丛书的教师、学生和读者朋友提出宝贵意见和建议，使之更臻成熟。

本套丛书的编写工作得到湖北省教学改革项目——计算机科学与技术专业课程体系改革的资助，武汉大学计算机学院也给予了大力支持，在此表示衷心感谢。

2013 年 3 月

前　言

面向对象的程序设计技术是目前最热门、最实用的软件开发手段。与传统的面向过程的程序设计技术完全不同，面向对象的程序设计技术把现实世界的问题抽象为“类”，而要解决的问题是对类所生成的对象的一系列操作。它的出现是程序设计方法学的一场革命，它注重数据和程序之间不可分割的内在联系，并把它们进行数据抽象，封装成一个统一的整体，使程序员将精力主要集中于要处理的对象的设计和研究上，大大提高了软件开发的效率。

C++语言是一种混合型的面向对象的程序设计语言。它既具有独特的面向对象的特征，可以为面向对象的技术提供全面支持；又具有对传统 C 语言的向后兼容性，具备结构化程序设计特征；特别为学习和掌握 Visual C++、Java 等软件开发工具提供了坚实的理论基础。因此，目前越来越多的高校改革计算机专业教学大纲，直接以 C++语言代替 C 语言作为基础程序设计语言，使学生尽快掌握面向对象的程序设计方法，提高实际的软件开发能力（因为目前的软件开发工具都是采用面向对象的方法进行软件开发的）。

本书是在作者多年讲授 C/C++语言的基础上编写的一本 C++语言教学辅导书。全书分为 10 章，第 1 章为 C++语言概述，介绍 C++语言的数据类型、运算符、3 种控制语句、函数设计方法等；第 2 章为类和对象，介绍 C++面向对象的程序设计基础；第 3 章为引用，介绍引用的相关概念和使用方法；第 4 章为友元，介绍友元的相关概念和使用方法；第 5 章为运算符重载，介绍运算符重载的相关概念和使用方法；第 6 章为模板，介绍模板的相关概念和使用方法；第 7 章为继承和派生，更深入地介绍面向对象的设计方法；第 8 章为多态性和虚函数，介绍面向对象的另一种特性，即多态性；第 9 章为 C++的 I/O 流库，介绍 C++文件操作方法等；第 10 章为异常处理，介绍 C++异常处理机制和方法。每章的内容分为两部分，前半部分介绍本章的主要概念、使用语法和相应的程序实例，后半部分是与本章内容紧密相关的例题分析，分为单项选择题、填空题、判断题、问答题和编程题，其中包含近几年一些 IT 公司的笔试题。大部分题目给出了较为详细的分析过程。附录 A 给出了一份本科生考试试题，附录 B 给出了近几年全国计算机等级考试二级 C++试题。

本书的所有程序均在 Visual C++ 6.0 环境中调试通过。

本书重点突出，知识点解析详细，习题分析具有启发性，既可作为大专院校各专业 C++语言程序设计课程的教学辅导书，也可作为计算机水平考试和等级考试者的参考书。

参与本书编写人员除了封面署名人员以外，还有金晶、陶红艳、马玉琳、余云霞和喻卫等人。由于水平有限，尽管编者不遗余力，书中难免存在错误和不足之处，敬请有关专家和广大读者批评指正。

编者

2013 年 3 月

目　录

第1章　C++语言概述

本章学习要点

- 掌握C++语言的特点。
- 掌握C++的各种数据类型及其基本运算。
- 掌握C++的各种控制结构及其使用技巧。
- 掌握C++数组和指针的使用方法。
- 掌握C++函数的相关概念和设计方法。
- 灵活运用本章的相关知识进行综合程序设计。

1.1　基本知识点

1.1.1　面向对象语言的要素

面向对象系统包含3个要素，即对象、类和继承，这3个要素反映了面向对象的传统观念。面向对象的语言应该支持这3个要素。

- 对象：从实现形式上讲，对象是一个状态和操作（或方法）的封装体。状态是由对象的数据结构的内容和值定义的，方法是一系列的实现步骤，它是由若干操作构成的。对象实现了信息隐藏，对象与外部是通过操作接口联系的，操作接口提供了这个对象的功能。对象通过消息与另一个对象传递信息，每当一个操作被调用，就有一条消息被发送到这个对象上，消息带来了这个将被执行的操作的详细内容。
- 类：类是创建对象的样板，它包含着所创建对象的状态描述和方法的定义。类的完整描述包含了外部接口、内部算法以及数据结构的形式。由一个特定的类所创建的对象称为这个类的实例，因此类是对象的抽象及描述，它是具有共同行为的若干对象的统一描述体。类中要包含生成对象的具体方法。
- 继承：类提供了说明一组对象结构的机制，再借助于继承这一重要机制扩充了类的定义，实现了面向对象计算的优越性。继承提供了创建新类的一种方法，这就是说，一个新类可以通过对已有类进行修改或扩充来满足新类的要求。新类共享已有类的行为，而自己还具有修改的或额外添加的行为。因此，可以说继承的本质特征是行为共享。

1.1.2　词法及词法规则

1. C++的字符集

C++的字符集由下列字符组成：

- 大小写英文字母，即a~z和A~Z。

- 数字字符，即 0～9。
- 特殊字符，即空格、!、#、%、^、&、*、_（下划线）、-、+、=、～、<、>、/、\、|、.、，、、:、；、?、′、"、（）、[]、{}。

2. 单词及词法规则

C++共有以下 6 种类型的单词。

- 标识符：是由程序员定义的单词，用来命名程序中的一些实体。常见的有函数名、类名、变量名、常量名、对象名、标号名、类型名等。C++规定，标识符是由大小写字母、数字字符（0～9）和下划线组成的，并且以字母或下划线开始，其后跟零个或多个字母、数字字符或下划线。
- 关键字：是系统已预定义的单词。它们在程序中都有着不同的用途，用户不可再重新定义。
- 运算符：是系统预定义的函数名称，这些函数作用于被操作的对象，将获得一个结果值。运算符通常由一个或多个字符组成。
- 分隔符：又称为标点符号。分隔符是用来分隔单词或程序正文的，它用来表示某个程序实体的结束和另一个程序实体的开始。在 C++中，常用的分隔符有空格符、逗号、分号和冒号等。
- 常量：是在程序中直接使用符号表示的数据。在 C++中，常量有数字常量、字符常量、字符串常量等。
- 注释符：在 C++中，采用了两种注释方法。一是使用“/*”和“*/”括起来进行注释（C 语言的注释方式）。二是使用“//”，从“//”开始，直到它所在的行尾，所有字符都被作为注释处理。这种方法用来注释一行信息。

1.1.3 数据类型

1. 基本数据类型

在 C++中，每种基本数据类型都使用一个关键字来表示。C++的基本数据类型描述了机器硬件所支持的对象和可以对这些对象执行的操作。表 1.1 是 C++在 32 位机编译环境下所提供的基本数据类型及其取值范围。

在 C++中，数据类型也可以看成类，也有构造函数等，例如：

```
int a(5);
```

这里定义了一个整型变量a，通过调用int的构造函数将其值赋值为5。也可以简单地理解为C++中变量赋初值的一种方式。

表 1.1　C++的基本数据类型及其取值范围

数据类型	占用字节数	取值范围
bool	1	true 或 false
int	4	-2^{31}～$2^{31}-1$
short int	2	-2^{16}～$2^{16}-1$
long int	4	-2^{31}～$2^{31}-1$
unsigned int	4	0～$2^{32}-1$

（续表）

数据类型	占用字节数	取值范围
unsigned short int	2	$0\sim2^{16}-1$
unsigned long int	4	$0\sim2^{32}-1$
char	1	-128～127
signed char	1	-128～127
unsigned char	1	0～255
float	4	-3.4E+38～3.4E+38
double	8	-1.7E+308～1.7E+308
long double	10	-3.4E+4932～3.4E+4932

2. 复合数据类型

C++中还提供了几种复合数据类型，这些复合数据类型包括数组、结构体、共用体、枚举、类和用户自定义类型。

（1）数组

数组是具有统一数据类型的对象的集合。每一个单独的对象并没有名字，但是它们可以通过其在数据中的位置来确定。

在 C++中，不仅可以定义一维数组，还可以定义多维数组。多维数组中的每一维的范围大小都用一个中括号括起来。例如，下面的语句定义了一个二维数组，该数组的第一维共有 10 个元素，第二维共有 20 个元素。

```
int Array[10][20];
```

在定义数组时，系统将自动为它分配一块连续的内存空间。此时，数组名指向这块空间的起始点，并且在程序中不可改变。也就是说，一个数组名相当于一个常量。

（2）结构体

结构体将某些相关的具有不同类型的数据组织到一个数据类型中。定义一个结构体类型的一般形式为：

```
struct 结构体名
{   结构体成员 1 定义;
    结构体成员 2 定义;
    …
    结构体成员 n 定义;
};
```

结构体成员可以嵌套，即结构体的成员可以是另一个结构体变量。

（3）共用体

共用体使几种不同的数据类型的变量共占同一内存单元。共用体类型变量的定义形式为：

```
union 共用体名
{   共用体成员 1 定义;
    共用体成员 2 定义;
    …
    共用体成员 n 定义;
```

```
} 变量列表;
```

共用体和结构体的定义相似，但它们的含义是不同的，二者的主要区别在于：结构体变量所占内存长度是各成员所占内存长度之和，每个成员分别占有其自己的内存单元；而共用体变量所占的内存长度等于最长的成员的长度。

（4）枚举

所谓枚举是指将变量的值一一列举出来，变量的值只限于在列举出来的值的范围内。

定义枚举类型由关键字 enum 开始。例如，下面的语句定义一个名为 color 的枚举类型，并定义了一个枚举类型的变量 mycolor：

```
enum color {Red, Green, Blue, Yellow};
enum color mycolor;
```

3. 指针

指针也是一种数据类型，具有指针类型的变量称为指针变量。一个指针变量所存储的信息是一个对象在内存中的地址。通过指针可以间接地访问对象。

（1）指针变量定义

每一个指针变量都有一个相应的基类型，基类型用以说明这个指针所指向的地址中存放数据的数据类型。指针的定义如下：

```
基类型  *指针变量;
```

例如：

```
int *pi;              //pi 是一个指向 int 型变量的指针
float *pl;            //pl 是一个指向 float 型变量的指针
char *pc;             //pc 是一个指向 char 型变量的指针
char (*pa)[3];        //pa 是一个指向一维数组的指针
int (*pf)();          //pf 是一个指向函数的指针，该函数的返回值为 int 型数值
int **pp;             //pp 是一个指向指针的指针
```

一个指针变量所占的内存空间大小与一个内存地址所占空间相等。也就是说，一个int型指针变量与一个double型指针变量占用同样大小的内存空间。

一个指针所指向的空间可以通过 malloc 函数或 new 运算符来分配，例如，以下语句让指针 p 指向一个长度为 4 的整数空间：

```
int *p = new int[4];
```

或

```
int *p=(int *)malloc(sizeof(int)*4);
```

当不再需要指针变量所指空间时，可用 free 或 delete 释放所分配的空间。

提 示

如果用new分配单个数据空间，可用delete释放；如果用new分配一个含多个元素的数组空间，可用delete []释放，其中方括号的作用是通知编译器要释放数组中的所有元素（若是对象数组，delete []根据由new []创建的对象数调用相同数目的析构函数），否则delete只释放了该指针所指的单个数组元素，其后的数组元素仍没有被释放。

在C++中，数组的元素可以用下标表示，也可用指针表示，但是最好还是用指针表示，因为指针表示要比下标表示处理起来更快。因此，C++程序中，尽量使用指针来引用数组元素。下面介绍数组元素的指针表示法。

（2）一维数组的指针表示法

例如：

```
int a[5];          //a 是一维数组名，它有 5 个 int 型变量
```

当用指针方法表示时，*(a+i)与 a[i]（其中，i=0，1，2，3，4）是相同的。

常量指针与变量指针是有区别的。例如：

```
int a[10], *p;
p=a;
```

表达式p+1、a+2、p=p+1、p-a等都是合法的。而表达式a=a+1、a=a-1等是非法的，因为p是变量指针，而a是常量地址（指针）。

（3）二维数组的指针表示法

例如：

```
int b[2][3];      //b 是二维数组名，它有 6 个 int 型变量
```

当用指针方法表示时，*(*(b+i)+j)与b[i][j]（其中，i=0，1；j=0，1，2）是相同的。

一个二维数组可以看作是一个一维数组，它的元素又是一个一维数组。对 b[2][3]来讲，可以看成是具有两个元素的一维数组，即称行数组，每个元素（即每个行数组）又是具有三个元素的一维数组，称为列数组。因此，b[2][3]可以看成为两个元素的一维行数组和三个元素的一维列数组组成。前面讲过了一维数组的指针表示，将二维数组的行、列的一维数组都用指针表示，便得到如下形式：

```
*(*(b+i)+j)
```

这是一个二级指针。将二维数组的行数组用下标表示，列数组用指针表示，得到如下形式：

```
*(b[i]+j)
```

再将二维数组的行数组用指针表示，列数组用下标表示，又得到如下形式：

```
(*(b+i))[j]
```

另外，按二维数组各个元素在内存中存放的顺序，用指向数组首元素的一级指针表示如下：

```
*(&b[0][0]+3*i+j)
```

其中，&b[0][0]是b数组的首元素地址。

对于二维以上数组的指针表示与此类似。

1.1.4　常量定义

在 C++中，常量的定义和变量相似，只是要在前面加上 const 关键字，const 意味着“只读的”。例如：

```
const int a=1;
```

```
int const a=1;
```

以上两个语句都定义了a是一个常整数，这样不能再修改a的值。

```
int b[]={1,2,3};
const int *a=b;
```

定义a是一个指向常整数的指针，a为数组b的地址，此时a指向的整数是不可修改的，如*a=2;是错误的，但可以通过b修改数组值，即不能通过a来修改其指向的值。而且a可以指向其他数组，如有一个数组定义int c={4,5,6}，可以执行a=c，还可以执行a++。

```
int b[]={1,2,3};
int * const a=b;
```

定义a是一个指向整数的常指针，a为数组b的地址，此时a指针指向的整数是可以修改的，如*a=2;是正确的，但指针a是不可修改的，不能让a指向其他数组，也不能执行a++语句。

在C++中可以用const定义常量，也可以用#define定义常量。

1.1.5 运算符

1. 算术运算符

C++提供如下的算术运算符。

- 单目算术运算符：-（取负）、+（取正）、++（增1）、--（减1）。
- 双目算术运算符：+（相加）、-（相减）、*（相乘）、/（相除）和%（取余数）。

其中，加、减、乘、除四则运算对int型、float型和double型变量都适用，而%运算符只用于int型运算。

与C语言不同，在C++中，++和--运算符可以作用于浮点数。

2. 关系运算符

关系运算符都是双目的，共有如下6种：

>（大于），<（小于），>=（大于等于），<=（小于等于），==（相等），!=（不相等）。

3. 逻辑运算符

- 单目逻辑运算符：!（逻辑求反）。
- 双目逻辑运算符：&&（逻辑与），||（逻辑或）。

4. 位操作运算符

位操作运算符是用来进行二进制位运算的运算符，它又分为两类，即逻辑位运算符和移位运算符。

（1）逻辑位运算符

- 单目逻辑位运算符：~（按位求反）。
- 双目逻辑位运算符：&（按位与）、|（按位或）、^（按位异或）。

（2）移位运算符

移位运算符有<<（左移）、>>（右移）。

左移是将一个二进制数按指定移动的位数向左移位，移掉的位被丢弃，右边移出的空位一律补 0。右移是将一个二进制数按指定移动的位数向右移位，移掉的位被丢弃，左边移出的空位或者一律补 0，或者补符号位，这要由机器而定。

5. 赋值运算符

- 简单的赋值运算符：=（赋值运算符）。
- 复合的赋值运算符：+=（加赋值），-=（减赋值），*=（乘赋值），/=（除赋值），%=（求余赋值），&=（按位与赋值），|=（按位或赋值），^=（按位异或赋值），<<=（左移位赋值），>>=（右移位赋值）。

6. 条件运算符

条件运算符的使用格式如下：

```
d1 ?  d2 :d3
```

其功能是先计算d1的值，并进行判断，如果为非零，则表达式的值为d2的值，否则表达式的值为d3的值。

7. 逗号运算符

逗号运算符的优先级是所有运算符中最低的。使用逗号运算符（,）可以将多个表达式组成为一个表达式。例如：

```
d1, d2, d3, d4
```

其中，d1、d2、d3和d4各为一个表达式。整个逗号表达式的值和类型由最后一个表达式决定。计算一个逗号表达式的值时，从左至右依次计算各个表达式的值，最后计算的一个表达式的值和类型便是整个逗号表达式的值和类型。

8. 强制类型运算符

该运算符用来将指定的表达式的值强制为所指定的类型，使用格式如下：

```
类型说明符(表达式)
```

或者

```
(类型说明符) 表达式
```

将所指定的“表达式”的类型转换为指定的“类型说明符”所说明的类型。这种强制转换可能使数值精度受到影响。

9. 长度运算符

长度运算符为 sizeof，它用于测试数据类型或变量所占用的字节数。在 C++中求一个变量的长度时还需考虑内存中数据对齐的问题，例如，有以下程序：

```
#include <stdio.h>
void main()
{   struct
    {   int n;
```