21世纪高等学校计算机基础实用规划教材

基于自然语言思想的递进C/C++教程（下册C++）

李春庚 编著

清华大学出版社

21世纪高等学校计算机基础实用规划教材

基于自然语言思想的递进C/C++教程（下册C++）

李春庚 编著

清华大学出版社
北京

内 容 简 介

本书下册虽然是建立在上册基础上的，但又有充分的独立性，完全可以单独使用。

本书以“选词造句，连句成段，多段成章，且词不断丰富，句式不断变化”的自然语言发展思想为学习线索，在C语言的“名词”——结构体基础上，发展得到C++语言的“标志性名词”——类和对象；将C语言中的指针和数组应用于C++的类对象，建立起了C和C++的连贯性；根据自然语言中，同一个名称或动词在不同的语境下有不同的意思，要根据上下文来理解的规律，讲解了C++语言中运算符的重载；根据C语言中名词的空间域和时间域，深入阐释了C++中类的封装、继承和派生。而模板、流、命名空间、标准模板库更是名词、动词及词的时间空间作用域融合发展的结果。

全书例题围绕“学校信息管理系统”展开，随着不断深入学习，不断完善系统的功能，使之接近实际开发。每章的习题都是例题的变化或功能的增强，以加强学生对知识的理解，锻炼程序开发能力。

本书适合作为普通高校C++语言课程的教材，也可以用作培训教材和自学教材使用。

图书在版编目(CIP)数据

基于自然语言思想的递进C/C++教程(下册C++)/李春庚编著. —北京：清华大学出版社，2011.7
(21世纪高等学校计算机基础实用规划教材)
ISBN 978-7-302-24469-1

Ⅰ. ①基… Ⅱ. ①李… Ⅲ. ①C语言—程序设计—高等学校—教材 Ⅳ. ①TP312

中国版本图书馆CIP数据核字(2010)第264703号

责任编辑：梁　颖　刘向威
责任校对：李建庄
责任印制：何　芊
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：010-62795954，jsjjc@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：北京市人民文学印刷厂
装 订 者：三河市溧源装订厂
经　　销：全国新华书店
开　　本：185×260　**印　张**：18　**字　数**：446千字
版　　次：2011年7月第1版　　**印　　次**：2011年7月第1次印刷
印　　数：1～4000
定　　价：28.00元

产品编号：037395-01

编审委员会成员

（按地区排序）

清华大学	周立柱　教授
	覃　征　教授
	王建民　教授
	冯建华　教授
	刘　强　副教授
北京大学	杨冬青　教授
	陈　钟　教授
	陈立军　副教授
北京航空航天大学	马殿富　教授
	吴超英　副教授
	姚淑珍　教授
中国人民大学	王　珊　教授
	孟小峰　教授
	陈　红　教授
北京师范大学	周明全　教授
北京交通大学	阮秋琦　教授
	赵　宏　教授
北京信息工程学院	孟庆昌　教授
北京科技大学	杨炳儒　教授
石油大学	陈　明　教授
天津大学	艾德才　教授
复旦大学	吴立德　教授
	吴百锋　教授
	杨卫东　副教授
同济大学	苗夺谦　教授
	徐　安　教授
华东理工大学	邵志清　教授
华东师范大学	杨宗源　教授
	应吉康　教授
东华大学	乐嘉锦　教授
	孙　莉　副教授
浙江大学	吴朝晖　教授
	李善平　教授

扬州大学	李　云　教授
南京大学	骆　斌　教授
	黄　强　副教授
南京航空航天大学	黄志球　教授
	秦小麟　教授
南京理工大学	张功萱　教授
南京邮电学院	朱秀昌　教授
苏州大学	王宜怀　教授
	陈建明　副教授
江苏大学	鲍可进　教授
中国矿业大学	张　艳　副教授
	姜　薇　副教授
武汉大学	何炎祥　教授
华中科技大学	刘乐善　教授
中南财经政法大学	刘腾红　教授
华中师范大学	叶俊民　教授
	郑世珏　教授
	陈　利　教授
江汉大学	颜　彬　教授
国防科技大学	赵克佳　教授
	邹北骥　教授
中南大学	刘卫国　教授
湖南大学	林亚平　教授
西安交通大学	沈钧毅　教授
	齐　勇　教授
长安大学	巨永锋　教授
哈尔滨工业大学	郭茂祖　教授
吉林大学	徐一平　教授
	毕　强　教授
山东大学	孟祥旭　教授
	郝兴伟　教授
中山大学	潘小轰　教授
厦门大学	冯少荣　教授
仰恩大学	张思民　教授
云南大学	刘惟一　教授
电子科技大学	刘乃琦　教授
	罗　蕾　教授
成都理工大学	蔡　淮　教授
	于　春　讲师
西南交通大学	曾华燊　教授

出版说明

随着我国改革开放的进一步深化，高等教育也得到了快速发展，各地高校紧密结合地方经济建设发展需要，科学运用市场调节机制，加大了使用信息科学等现代科学技术提升、改造传统学科专业的投入力度，通过教育改革合理调整和配置了教育资源，优化了传统学科专业，积极为地方经济建设输送人才，为我国经济社会的快速、健康和可持续发展以及高等教育自身的改革发展做出了巨大贡献。但是，高等教育质量还需要进一步提高以适应经济社会发展的需要，不少高校的专业设置和结构不尽合理，教师队伍整体素质亟待提高，人才培养模式、教学内容和方法需要进一步转变，学生的实践能力和创新精神亟待加强。

教育部一直十分重视高等教育质量工作。2007 年 1 月，教育部下发了《关于实施高等学校本科教学质量与教学改革工程的意见》，计划实施“高等学校本科教学质量与教学改革工程（简称‘质量工程’）”，通过专业结构调整、课程教材建设、实践教学改革、教学团队建设等多项内容，进一步深化高等学校教学改革，提高人才培养的能力和水平，更好地满足经济社会发展对高素质人才的需要。在贯彻和落实教育部“质量工程”的过程中，各地高校发挥师资力量强、办学经验丰富、教学资源充裕等优势，对其特色专业及特色课程（群）加以规划、整理和总结，更新教学内容、改革课程体系，建设了一大批内容新、体系新、方法新、手段新的特色课程。在此基础上，经教育部相关教学指导委员会专家的指导和建议，清华大学出版社在多个领域精选各高校的特色课程，分别规划出版系列教材，以配合“质量工程”的实施，满足各高校教学质量和教学改革的需要。

本系列教材立足于计算机公共课程领域，以公共基础课为主、专业基础课为辅，横向满足高校多层次教学的需要。在规划过程中体现了如下一些基本原则和特点。

（1）面向多层次、多学科专业，强调计算机在各专业中的应用。教材内容坚持基本理论适度，反映各层次对基本理论和原理的需求，同时加强实践和应用环节。

（2）反映教学需要，促进教学发展。教材要适应多样化的教学需要，正确把握教学内容和课程体系的改革方向，在选择教材内容和编写体系时注意体现素质教育、创新能力与实践能力的培养，为学生的知识、能力、素质协调发展创造条件。

（3）实施精品战略，突出重点，保证质量。规划教材把重点放在公共基础课和专业基础课的教材建设上；特别注意选择并安排一部分原来基础比较好的优秀教材或讲义修订再版，逐步形成精品教材；提倡并鼓励编写体现教学质量和教学改革成果的教材。

(4) 主张一纲多本,合理配套。基础课和专业基础课教材配套,同一门课程可以有针对不同层次、面向不同专业的多本具有各自内容特点的教材。处理好教材统一性与多样化,基本教材与辅助教材、教学参考书,文字教材与软件教材的关系,实现教材系列资源配套。

(5) 依靠专家,择优选用。在制定教材规划时依靠各课程专家在调查研究本课程教材建设现状的基础上提出规划选题。在落实主编人选时,要引入竞争机制,通过申报、评审确定主题。书稿完成后要认真实行审稿程序,确保出书质量。

繁荣教材出版事业,提高教材质量的关键是教师。建立一支高水平教材编写梯队才能保证教材的编写质量和建设力度,希望有志于教材建设的教师能够加入到我们的编写队伍中来。

21 世纪高等学校计算机基础实用规划教材

联系人:魏江江 weijj@tup.tsinghua.edu.cn

前言

1. 本书编写背景

C语言是应用面广，效率高的结构化高级计算机语言。C++语言是在C语言基础上发展起来的、面向对象高级计算机语言，它不但继承了C语言的所有优点，兼容C语言的所有语法，更增加了安全性高、适应性强和编程高效率的特点。通常将C语言和C++语言合并，泛称C++语言。C++语言也成为全世界多数大学理工类专业计算机编程训练的首选教学语言。

怎样才能通过教和学，高效率地掌握C++语言呢？当前有以下两种教学模式：

(1) 先学习C语言，然后学习C++语言。该教、学模式，知识内容掌握得比较深入，但耗费的时间比较长，效率不高。另外，容易带来“结构化”和“面向对象”两种编程思想的对立。因为在C语言的教材中，通常只讲结构化的编程，而C++教材通常强调面向对象的优点和其与结构化思想的不同。忽略了“结构化”和“面向对象”的联系，及“面向对象”是如何从“结构化”发展而来的。

(2) 模糊C语言和C++语言的界限，直接学习C++语言。这种教、学模式，效率比较高，耗费的时间较短，但教、学效果往往不够理想。因为面向对象编程，程序语句排列的先后顺序和程序执行的顺序不一致，具有“超文本的跳跃性”，对于没有结构化程序设计基础的初学者理解难度较大。

为了克服以上两种教学模式的不足，我们寻着计算机编程语言产生发展的脉络，提出了“基于自然语言思想的递进C/C++教学”思路。因为计算机编程语言从机器语言到汇编语言再到高级语言的发展过程，就是越来越逼近人类自然语言的过程；从C语言到C++语言的发展过程，就如同人类自然语言不断丰富的发展过程，它是一个递进的补充、完善过程。

人类自然语言是由词构成短语，多个词或短语构成句子，多个句子形成段落，多个段落就是文章。基于人类自然语言的思想，我们将C语言中的类型、常量和变量看做名词，将各种运算符看做动词，将if…else、switch…case、while等看做连词，表达式就是短语，程序语句就是句子，程序模块就是段落，整个工程的程序就是文章。而数组、结构体、函数和指针是名词的发展，语句嵌套、函数的递归等是句式的发展。沿着这样的理解思路，C++中的类自然是C中结构体类型的发展，类的对象是结构体变量的发展，当以类和对象为着眼点编程时，是面向对象的编程，但类的构建过程，即类的内部实现却常常是结构化的。这样我们就清晰了从C发展到C++的过程，找到了贯穿C和C++的一致性知识框架体

系,使 C 语言和 C++语言无缝衔接。使学生在学习过程中,不去区分 C 和 C++,更多地学习 C 和 C++的递进性、一致性,而在课程结束后,可以深入地理解 C 和 C++的不同,领略 C++对 C 的补充和增强。深入理解 C 语言和 C++语言的知识内容,真正做到融会贯通,培养编程实践能力。

本书讲稿已经在大连海事大学的教学中使用,得到良好的教学效果。

2. 本书特色

(1) 以自然语言的思想,讲授计算机语言,形象易懂。

(2) 篇幅短小,中心突出,练习题通常是例题的发展和变化,少而精,非常适合于课堂教学和初学者自学。

(3) 仅用几个例题,贯穿本书始末,将学习的各种语法递进加入例题,使之不断完善,丰富,一步步接近实际开发。

(4) 本书上册引入四则运算数学方法,讲解二维数组指针难点,使语言的学习成为简单的推导,深入而准确。

(5) 本书上册提出表达式、函数是复合的变量,常量是变量的"瞬间快照"等观点,蕴涵了"泛化"的思想,潜移默化训练学生的思维方式,为 C++的泛型程序设计打下思想基础。

(6) 本书下册延续了上册的讲解思路。发展 C/C++语言中的名词,得到类、对象、模板和命名空间;发展 C/C++语言中的动词,得到运算符的重载;当各种词性融合发展时,得到标准模板库。

(7) 本书下册的例题,是以面向对象的编程方法对上册结构化编程例题的重新编写和进一步补充、发展。充分展示了结构化编程和面向对象编程的区别和联系。

感谢大连海事大学信息科学技术学院,计算机系"C/C++语言课程组"的史一民老师、刘宁老师、刘亚清老师、于纯妍老师、宋梅萍老师、黄健老师和郭浩老师给予的大力支持,感谢遥感信息研究所的安居白教授给予的大量帮助。

感谢读者选择使用本书,欢迎您对本书内容提出批评和修改意见,我们将不胜感激。作者联系地址如下:

电子邮件地址:licgsh@163.com;licgsh@sina.com; licgsh@dlmu.edu.cn

通信地址:辽宁,大连,大连海事大学,信息科学技术学院,计算机系　李春庚

邮政编码:116026

编　者

2011 年 5 月

目 录

第 10 章　C++给 C 语言打的“补丁”

10.1　“补丁”(1)——C++的基本输入输出

C 语言的输入输出语句使用的是各种输入输出函数，对应输入输出不同类型的数据要么使用不同的函数，要么需要复杂的函数参数。C++语言具有很好的自适应性，这在输入输出语句中得到充分的体现。通常同一个输入输出语句，不用作任何的改动，就可以自适应地输入输出各种类型的数据。这是因为，C++有专门用于输入输出的“流类”。“流类”本身是类，由于我们还没有学习“类”的概念，所以没办法详细阐述，只能在学习“类”后再详细学习。现在只介绍简单的输入输出流，如果您不理解“流”，暂时可以不准确地认为它是一种输入输出“命令”或“语句”。

C++的基本输入是由标准输入流 cin 和提取运算符>>配合使用完成的；基本输出是由标准输出流 cout 和插入运算符<<配合使用完成的。

[例 10.1]　使用标准输入输出流 cin 和 cout，对各种类型数据进行输入和输出操作。

```
#include<iostream.h>                          //包含输入输出流 cin 和 cout 的头文件
void main()
{
 int id;
 char name[10];
 float score[2];
 cout<<"请输入学生编号:"<<endl;               //输出字符串后，输出回车换行符
 cin>>id;
 cout<<"请输入学生姓名:"<<endl;
 cin>>name;
 cout<<"请输入两门成绩，空格分割:"<<endl;
 cin>>score[0]>>score[1];                     //多个数据的级联输入，可以是不同类型的
 cout<<"您输入的学生信息是:"<<endl;
 cout<<id;
 cout.width(10);                              //设定下面输出数据所占列宽度为 10 列
 cout<<name;
 cout.width(5);
 cout<<score[0]<<'\t'<<score[1]<<endl;        //成绩间输出一个制表符
 cout<<name<<"同学的总成绩是:"<<score[0]+score[1]<<endl;
//多个数据的级联输出，还可输出表达式的值
}
```

本例运行结果如图 10.1 所示。

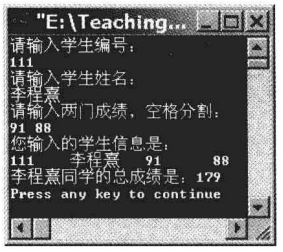

图 10.1 例 10.1 的运行结果

我们可以这样理解:cin 和 cout 是名词,是流对象,是我们的程序和操作系统间做数据交换的名词,<<和>>是动词,是将程序中的数据"送入"cout 中和从 cin 中"提取"数据的意思。

10.2 "补丁"(2)——引用 &

引用是 C++对 C 在名词上的扩充。引用就是给已存在变量定义一个新名字,它代表的就是那个已存在的变量(后面要学习,也可以是一个存在的对象)。所以通常说:"引用是变量(或对象)的别名"。引用本身不是变量(或对象)实体,系统不会为它分配内存空间,它不具有"变量的三层含意"(请参考上册 4.1.1 节)。引用本身是和程序中的 main、if 等代码一起,存放在程序的代码内存区,而变量是存储在数据内存区的。我们学过的普通变量、指针变量和各种复合类型变量都是存储在程序的数据内存区,它们都有变量的三层含意。引用不具有变量的三层含义,它仅有一层名字含义。对编程人员来说,引用仅仅是一个名字而已,它的地址就是它所代表的(引用的)那个变量(或对象)的地址,它的值就是它代表的(引用的)那个变量(或对象)的值。一句话,引用就是它所代表的(引用的)那个名词(变量或对象)实体。所以,通过引用对变量的任何操作,就是对变量实体的操作。

[例 10.2] 引用的定义和使用。

```
#include <iostream.h>
void main()
{
 int x = 10;
 int& r = x;                    //定义引用 r,同时让它引用已有同类型变量 x,这个过程必须同时完成
 cout <<"x = "<< x << endl;
 r = 100;
 cout <<"r = 100 后 x = "<< x <<'\t'<<"r = "<< r << endl;
}
```

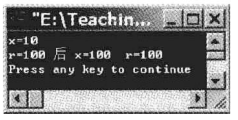

图 10.2 例 10.2 的运行结果

本例运行结果如图 10.2 所示。

[例 10.3] 对上册例 5.4 节的补充。以引用作为函数的参数,编写交换两个变量值的函数 swap4。

```
#include <iostream.h>
void swap1(int a, int b) {                    //参数是简单变量的函数
 int c;
 c = a;                                       //通过中间变量 c 交换 a 和 b 的值
 a = b;
 b = c;
}
```

```
void swap2(int *a, int *b) {                    //参数是指针变量的函数
 int c;                                         //通过中间变量 c 交换 *a 和 *b 的值
 c = *a;
 *a = *b;
 *b = c;
}
void swap3(int *a, int *b) {                    //参数是指针变量的函数
 int * c;                                       //通过中间指针变量 c 交换 a 和 b 的值
 c = a;
 a = b;
 b = c;
}
void swap4(int &a, int &b) {                    //参数是引用的函数
 int c;                                         //通过中间变量 c 交换 *a 和 *b 的值
 c = a;
 a = b;
 b = c;
}

void main(){
 int x = 2;
 int y = 100;
 cout<<"x 的初始值:"<<x<<'\t'<<"y 的初始值:"<<y<<endl;
 swap1(x, y);                   //以普通变量值为实际参数,调用形式参数是普通变量的函数
 cout<<"swap1: x = "<<x<<'\t'<<"y = "<<y<<endl;
 swap2(&x, &y);                 //以普通变量的地址为实际参数,调用形式参数是指针变量的函数
 cout<<"swap2: x = "<<x<<'\t'<<"y = "<<y<<endl;
 swap3(&x, &y);                 //以普通变量的地址为实际参数,调用形式参数是指针变量的函数
 cout<<"swap3: x = "<<x<<'\t'<<"y = "<<y<<endl;
 swap4(x, y);                   //以普通变量值为实际参数,调用形式参数是普通变量引用的函数
 cout<<"swap4 x = "<<x<<'\t'<<"y = "<<y<<endl;
}
```

本例中 swap1～swap3 是上册例 5.4 节的内容,swap4 是本例增加的部分。

首先,对比 main 函数中 swap1～swap4 四个函数的调用形式。swap1 和 swap4 的实际参数形式相同,都是 int 型的普通变量,但 swap1 的形式参数是普通变量,swap4 的形式参数是 int &(int 引用)型。即给引用赋值(传递值)的形式和给普通变量赋值(传递值)的形式是相同的。swap2 和 swap3 的实际参数都是变量的指针,它们的形式参数也都是指针变量。

对比函数 swap1～swap4 的调用结果,只有 swap2 和 swap4 调用后将当前 x 和 y 的值做了交换,swap1 和 swap3 没能交换 x 和 y 的值。原因是 swap1 中交换了只在 swap1 函数模块内有效的普通变量 a 和 b 的值,swap3 中交换了只在 swap3 函数模块内有效的指针变量 a 和 b 的值。函数 swap1 和 swap3 不能通过函数内部形参值的交换而使函数外部实参值得到交换。值得注意的是:函数 swap2 通过指针变量的形式参数取 * 运算和普通变量的局部变量使函数模块外的实际参数的值得到交换;swap4 通过引用形式参数

和普通变量的局部变量使函数模块外的实际参数的值得到交换。

结论是：引用作为函数的形式参数时，实际参数就是类型对应的普通变量，这种传递形式与普通变量作为函数的形式参数相同，而本质上起到的作用，与指针变量作为函数的形式参数相同，可以改变函数模块外的实际参数。这无疑是形式上的简洁，本质上的高效。例 10.3 运行结果如图 10.3 所示。

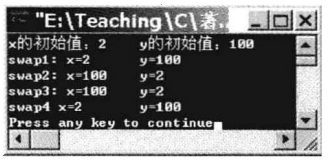

图 10.3 例 10.3 的运行结果

10.3 “补丁”(3)——形容词 const

C++中引入关键字 const，我们认为是“形容词”，它的意思是“常的”。通过形容词 const 的修饰，可以在 C 定义各种类型变量的基础上，定义具有“常”属性的各种类型变量——“常变量”。“常”和“变”本身是矛盾的。其实就像变量一样用一个由符号表征的名字来表示常量，而不是在程序中直接使用各种常数据，我们将这种符号名字的常量称作“符号常量”。符号常量可以增加程序的安全性、可读性和可维护性。

10.3.1 符号普通常量

符号常量是常量，但不是直接用值来体现，而是像变量那样有一个用符号表示的名字。符号普通常量就是用名字来表示一个具体的常量。C 中是采用预编译命令 # define 实现，如 # define PI 3.14 就是定义的 PI 为符号常量，其值是 3.14。内部执行机制是编译前进行替换，不通过编译器的语法检查，容易引入错误。

C++通过使用形容词性的关键字 const 修饰 C 中各种变量的定义形式来定义各种符号常量，我们可以极限地将由 const 定义的符号常量理解为“只读变量”。由于是常量，所以必须在定义时直接初始化，而且以后不能改变。上面预编译的等价形式是：const float PI=3.14。与预编译命令不同，此定义常量的方法是正常的语句，不是预编译命令，须要通过编译器的编译，提高了程序的安全性。

[例 10.4] 符号常量的定义和引用。

```
#include <iostream.h>
void main()
{
 const int x = 10;                    //定义 int 型的常量,其值是 10
 //int const  x = 10;                 //这个语句也是对的,const 在类型的前和后都正确
 char const name[] = "李程熹";        //定义 char[]型的常量 name,其值是"李程熹"
 //x = 100;// error C2166: l-value specifies const object
 //cin >> name;/ * binary '>>' : no operator defined
 which takes a right - hand operand of type 'const char [8]'
 (or there is no acceptable conversion) * /
 cout << x <<'\t'<< name << endl;
}
```