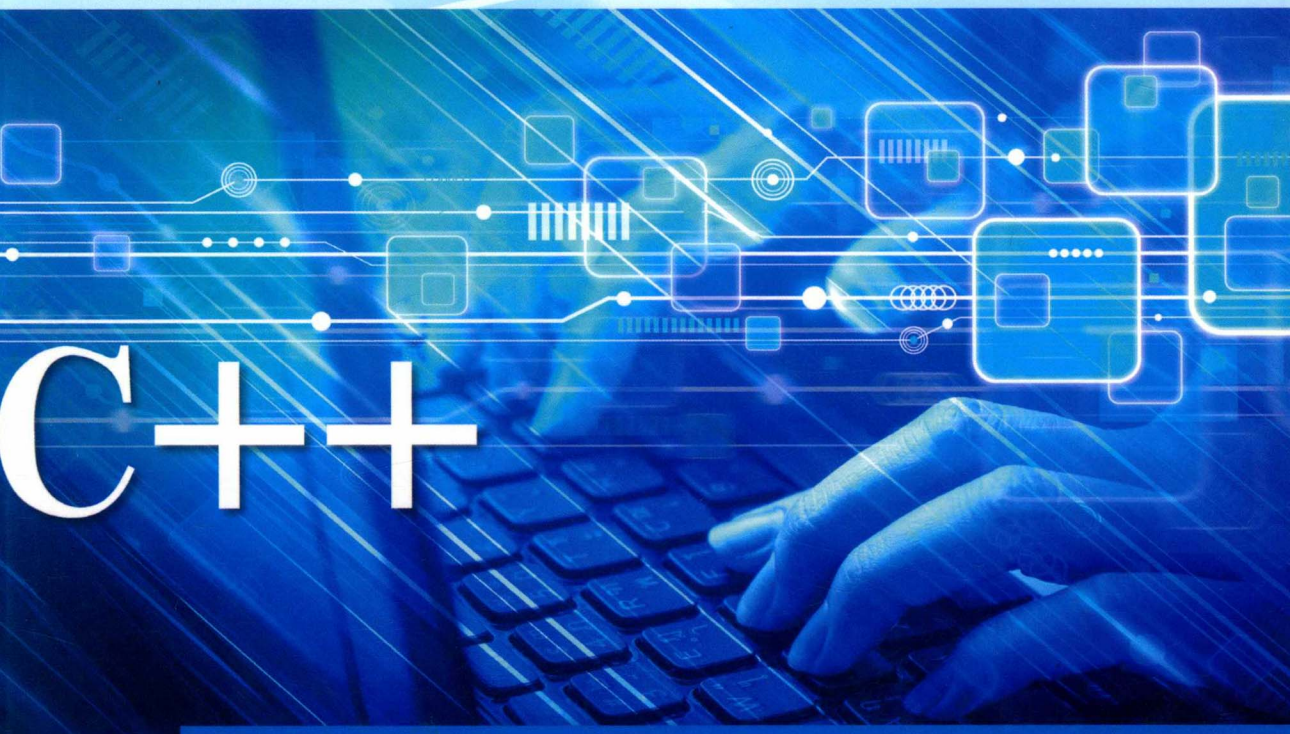
国家开放大学
THE OPEN UNIVERSITY OF CHINA



C++语言程序设计

C++ YUYAN CHENGXU SHEJI

（第2版）

李 宁 主 编

中央广播电视大学出版社

国家开放大学
THE OPEN UNIVERSITY OF CHINA

C++语言程序设计

（第2版）

李宁 主编

中央广播电视大学出版社·北京

图书在版编目（CIP）数据

C ++语言程序设计／李宁主编. —2 版. —北京：中央广播电视大学出版社，2015. 8

ISBN 978 - 7 - 304 - 07236 - 0

Ⅰ. ①C… Ⅱ. ①李… Ⅲ. ①C 语言—程序设计—开放大学—教材 Ⅳ. ①TP312

中国版本图书馆 CIP 数据核字（2015）第 159852 号

C ++语言程序设计（第 2 版）

C ++ YUYAN CHENGXU SHEJI

李 宁 主 编

出版·发行：中央广播电视大学出版社

电话：营销中心 010 - 66490011　　总编室 010 - 68182524

网址：http：//www. crtvup. com. cn

地址：北京市海淀区西四环中路 45 号　　**邮编**：100039

经销：新华书店北京发行所

策划编辑：邹伯夏　　**版式设计**：赵 洋

责任编辑：白 娜　　**责任校对**：宋亦芳

责任印制：赵连生

印刷：北京明月印务有限责任公司　　**印数**：1001~2000

版本：2015 年 8 月第 2 版　　2015 年12月第 2 次印刷

开本：787mm × 1092mm 1/16　　**印张**：17. 5　　**字数**：389 千字

书号：ISBN 978 - 7 - 304 - 07236 - 0

定价：26. 00 元

第2版前言 PREFACE

C++语言是在C语言的基础上发展起来的。C语言一诞生，就由于其高效性、灵活性和适应性被广泛应用，很快成为软件开发最重要的语言之一。C语言的灵活性虽然有利于算法的实现和程序设计技巧的发挥，但却显得不够规范，初学者不易掌握，因此很少将其作为第一门程序设计语言学习。随着C语言的不断发展，其规范化程度也在不断提高，发展到现在的C++语言已今非昔比，不但具备了支持面向对象程序设计的各种手段，其规范化程度也达到了一个新的水平。因此，只要在教学中有所取舍、有所侧重，完全可以将其作为程序设计的第一教学语言，将“程序设计的学习”与“实用工具与实用技能的掌握”真正统一起来，使有限的学时得到更充分的利用。

本书是为国家开放大学“C++语言程序设计”课程编写的专用教材，目标是通过学习C++语言中的数据类型、语句结构以及程序设计的基本方法，掌握程序设计中最基本的概念、方法和理论，了解面向对象程序设计的基本概念和方法，获得运用C++语言解决一般问题的实用技能。

本书在编写过程中仔细斟酌了教学内容的取舍，在满足教学大纲的前提下不刻意追求“系统、严格、完整”。在教学内容的表现形式上，本书注意适应“以自学为主、业余学习为主、分散学习为主”的教学特点，适时插入“学习目标”“注意”“经验与技巧”“小结”等助教提示，以帮助学生理解教学内容，少走弯路。本书在大部分章节后都提供了自测题，使学生能随时自我检查学习效果，并获得相应的反馈信息。与本书配套的内容还有实验和录像教材。本书使用了下列助学提示标志符：


自测题


注意


经验与技巧


思考题


上机测试

学生在利用本教材时应注意以下几点：

(1) 以结构化程序设计为主线。尽管C++语言全面支持面向对象程序设计，但本课程选用C++作为第一教学语言，主要是由于它的实用性和规范性，按照教学大纲的要求，仍以学习结构化程序设计为主。具体地说，本书应以前7章为重点，后3章只要求一般了解、掌握。

(2) 注重体现良好的程序设计风格，养成良好的程序设计习惯。不去挖掘C++的所有

功能或表现手法，因为挖掘出来的东西往往是原C语言中的不规范部分；不过分追求“技巧”，因为它往往依赖于一些不规范手法。

（3）注重实践。这是一门时间性很强的课程，只有多上机，才能真正掌握所学的知识。

本书第1章～第6章由首都经贸大学李宁副教授执笔，第7章～第10章由国家开放大学王娇讲师执笔，全书由李宁统合修订。参加本书审定工作的学科专家有中国石油大学陈明教授、清华大学郑莉教授和北京理工大学马锐副教授；诸位学科专家对本书初稿提出了宝贵的修改意见，特此致谢。

由于学识水平和时间的限制，不妥之处在所难免，欢迎批评指正。

李 宁

2015年4月

目　录　CONTENTS

计由C语言时代的结构化程序设计逐步过渡到面向对象程序设计。但同时也要认识到，面向对象程序设计是对结构化程序设计的丰富和提升，在面向对象的框架下，具体程序流程的设计仍然离不开结构化程序设计的理念和手段。

自测题

填空题

1.1-1 C++语言是在（　　）语言的基础上发展起来的。

1.1-2 C++语言是一种典型的全面支持（　　）特性的语言。

1.2 简单的C++程序

1.2.1 一个只显示输出一行文字的程序

例1.1：设计一个程序显示“同学们，早上好!”。

```
1: //主程序文件 goodmorning.cpp
2: #include <iostream>
3: using namespace std;
4: int main(){
5:     cout <<"同学们,早上好!" <<endl;
6:     return 0;
7: }
```

注意：上面程序清单中的行号是本书为了便于讲解而加上去的，不是程序文件内容的一部分。执行这个程序屏幕上将显示：

同学们，早上好!

这个程序虽然简单，但借助它可以说明以下一些问题。

1. 程序和程序文件

程序就是用计算机语言对如何完成一项任务（程序的功能）进行的描述。每个程序都必须保存在一个文本文件中，该文本文件称为程序文件。对于C++语言来说，程序文件约定的扩展名是cpp。文件名最好有一定的提示作用，能使人联想到程序内容或功能。例如，例1.1的程序文件就可以命名为goodmorning.cpp。

2. 函数和函数定义

函数就是具有特定功能的程序模块，是C++程序的重要组成部分。函数定义就是用特定的格式对函数功能进行描述，是建立函数必不可少的步骤。函数定义的一般格式如下：

```
类型修饰符 函数名(形参声明表) {函数体}
```

例 1.1 实际上只包含了 1 个函数定义，从第 4 行开始到第 7 行结束。第 4 行行首的 int 是一个 *类型修饰符*，表示函数的返回值是整数类型。main 是 *函数名*，系统将通过这个名称调用该函数。对于格式中的 *形参声明表*，程序中没有出现，表示该声明表是空的，没有任何参数。这里所说的形参是形式参数的简称。尽管 *形参声明表*是空的，但是函数名后面的括号不能省略，因为它是识别函数的重要标志。格式中大括号中的 *函数体*是一系列 C++语句，函数的功能正是由这些语句实现的。在例 1.1 中，*函数体*包含两个语句。基本语句都以分号作为结束符。

3. 主函数

每个程序至少要有 1 个函数，这个不可缺少的函数称为主函数，约定的函数名是 main。主函数（或称为 main 函数）是程序的入口，一个应用程序的执行就是从主函数中第 1 个语句开始的。显然，例 1.1 就是一个只包含主函数的程序。函数的返回值一般被调用它的其他函数所利用，但主函数的调用者是操作系统，它的返回值被操作系统所利用。对于主函数的返回值，一般用 0 表示正常结束，用非 0 整数表示非正常结束。本教材不讨论操作系统如何利用 main 函数的返回值问题，只需记住，本教材要求 main 函数的最后一个语句始终是：

```
return 0;
```

含有主函数的程序文件一般称为主程序文件。

4. 字母的大小写

在 C++程序中，字母的大小写是有区分意义的，因此 main、Main、MAIN 等都是不同的名称，作为主函数函数名的只能是 main，不能混同。

⚠ 在有些高级语言中，字母的大小写是不加以区分的，因此一些学过其他高级语言的同学可能已养成在写程序时不区分大小写的习惯，在编写 C++程序时应特别注意。

⚠ C++语言中除了字符串常量中引号内的字符以及注释文字之外，其他字符都必须是西文字符（或称为半角字符）。尤其注意不要误用中文标点，如把"同学们，早上好!"错写成“同学们，早上好!”。

5. 程序书写风格

C++程序在书写格式上是比较自由的，一行中可以有多个语句，一个语句也可以分布在连续的若干行中。例如，例 1.1 的主函数定义也可以写成：

```
int main(){cout <<"同学们,早上好!"<<endl;return 0;}
```

或

```
int main()
   {
```

```
        cout
        <<"同学们,早上好!"<<endl;
        return 0;
    }
```

等。但我们在编写程序时也不应那么随意。一个具有良好书写风格的程序便于阅读、理解和维护，它至少有以下特点：

- 每一个程序行都比较简单（尽量不包含多个语句）。
- 有规律地利用文本缩进体现语句间的逻辑关系。
- 适时添加必要的注释（见“1.2.2节的2. 程序的注释”中的解释）。
- 书写风格前后统一。

6. 常量

程序中的"同学们，早上好!" 是一个字符串常量。此外，C++程序中还可以有字符常量（如'A'）、整型常量（如345）、实型常量（如3.14）等。常量是程序中数据的一种重要表现形式，常量的值在程序运行中是不可改变的。

7. 数据的显示输出

例1.1中第5行是一个输出语句。其中cout是一个连接显示器的C++输出流对象，<<是与之相联系的插入操作符，两者配合即可完成简单的显示输出。endl实现回车换行，如执行如下语句

```
cout <<35 <<' '<<36 <<endl <<37 <<"Hello";
```

的输出效果是：

```
35 36
37Hello
```

8. 预处理命令 #include

预处理就是编译系统在编译一个程序之前通过预处理程序对程序文件中所有的预处理命令所进行的处理。以#开头的行都是预处理命令，如例1.1的第2行就是以#打头，因此它是一个预处理命令。该例中的这个预处理命令是#include，称为包含命令，功能是把一个文本文件的内容包含（插入）到该命令处。<iostream>是包含命令的参数，给出了要包含的那个文件的文件名，即iostream。

9. 头文件

在例1.1第5行显示输出语句中，用到了cout、<<和endl这样一些在另外的文件中声明的符号，这个另外的文件就是系统提供的文件iostream。程序的第2行用#include命令包含该文件，实际上也就是把针对cout、<<、endl等的声明插入程序的开始处。此类文件通常插入程序的头部，因此称为头文件。一般来说，如果程序中使用的函数、变量、常

量、类、数据类型等的声明是由别的文件提供的，就必须在程序文件的开始部分用#include 命令把相关的头文件包含进来。系统提供的头文件没有扩展名，要求用一对尖括号括起来（如<iostream>）。而程序设计者自己建立的头文件应以 h（header 的字头）为扩展名，要求用一对引号括起来（如"userheader.h"）。头文件都是文本文件。头文件实际上也是一种程序文件，但由于其特殊性，总是把它称为头文件，便于和以 cpp 为扩展名的程序文件区分。

10. 名字空间

例 1.1 中的第 3 行语句是一个 using namespace 语句，表示这个程序将采用被称为 std 的名字空间。关于名字空间，本教材不做进一步的说明，但应记住，每个程序都应在开始处使用一个如下这样的 using 语句：

```
using namespace std;
```

1.2.2 一个包含函数定义和调用的程序

例 1.2：设计函数 hello，它通过显示“×××，你们好!”，向指定的人或人们问好。

```
1: //主程序文件 hello.cpp
2: #include<iostream>
3: using namespace std;
4:
5: void hello(char*s){     //这是函数 hello 的定义
6:     cout<<endl/*从下一行行首开始显示*/<<s<<",你们好!";
7: }
8:
9: int main(){
10:     hello("同学们");
11:     hello("朋友们");
12:     cout<<endl;
13:     return 0;
14: }
```

运行这个程序，屏幕上将显示如下内容：

```
同学们，你们好!
朋友们，你们好!
```

通过这个程序要说明以下几个问题。

1. 函数调用

在例 1.2 中有两个函数定义：第 5～7 行对函数 hello 的定义和第 9～14 行对主函数

main的定义。虽然函数hello在主函数main的前面，但程序仍然从主函数main开始执行。主函数是由操作系统调用执行的，而其他函数必须通过程序中的函数调用才能执行。函数调用的一般格式是：

```
函数名(实参表)
```

这里的实参是实在参数的简称。格式中，*实参表*中的参数必须在数量和类型上与相应函数定义中的*形参声明表*吻合。例如，hello的*形参声明表*是char*s，参数变量s被定义为一个字符串；而10、11两行的两个函数调用中的*实参表*分别是“同学们”和“朋友们”，是两个字符串常量，因此与char*是吻合的。在调用过程中，实在参数和形式参数相结合，完成参数数据的传递。例如，通过第10行的调用，实参“同学们”与hello的形参变量s相结合，因此在执行第6行的语句

```
cout<<endl<<s<<",你们好!";
```

时，s的值就是字符串常量"同学们"，因此才显示出“同学们，你们好!”。

在函数调用过程中，程序执行将从调用处转移到被调用函数处执行，待函数执行完毕，再返回调用处的下面继续执行。例如，主函数main中的第10、11两行就是两个由函数调用构成的语句，先后两次调用函数hello。在第一次调用时，程序从第10行转到第5行执行hello，显示“同学们，你们好!”，然后返回调用处的下面，即第11行；执行第11行意味着第二次调用hello，程序的执行即从第11行转到第5行，函数hello显示“朋友们，你们好!”，然后返回调用处下面，而该处（第12行）是一个输出回车换行的语句。

这里提到的有关函数的一些问题还将在第5章做专门讨论，这里只需简要加以了解就可以了。

2. 程序的注释

例1.2第5行行尾的“//这是函数hello的定义”是对该行程序的注释。注释以//开头，其后是注释文字，可一直延续到该行行尾。第1行的//出现在行首，因此整行都是注释文字。第6行中部的“/*从下一行行首开始显示*/”是程序注释的另一种方式，注释文字夹在/*和*/之间，这样的注释不但可以出现在行尾，也可以出现在一行中的其他位置，还可以跨越多行。

编译系统忽略（不处理）注释文字，因此注释文字可以是任意的。注释可使程序更容易理解，不会有其他副作用，因此在编写程序时随时添加注释是一种良好的习惯。

3. 主函数与程序调试

例1.2并不是一个完整的应用程序，其主角是函数hello，而主函数main仅仅是为了创造一个调用hello的程序环境。像这样的主函数main，实际上是一个调试程序，它的目的并不是利用hello函数去完成某种具有实用价值的任务，而是验证函数hello的正确性。在学习程序设计的过程中，主函数main常常担任“调试程序”的角色。

1.2.3　一个计算圆面积的程序

例 1.3：设计一个函数 Area，它可根据给出的圆的半径，计算圆的面积。设计相应的调试程序，以验证函数 Area 的正确性。

```
1: //主程序文件 area.cpp
2: #include <iostream>
3: using namespace std;
4:
5: double Area(double r){return 3.1416*r*r;}
6:
7: int main(){
8:     double radius,area;
9:     cout <<endl <<"请输入圆的半径:";
10:    cin >>radius;
11:    area =Area(radius);
12:    cout <<endl <<"圆的面积:" <<area <<endl;
13:    return 0;
14: }
```

这个程序从第 7 行的主函数 main 入口处开始执行。程序首先从键盘输入圆的半径（第 9、第 10 行），然后调用函数 Area 计算圆的面积，并将这个结果赋值给变量 area（第 11 行）；最后显示输出 area 的值（第 12 行）。这里的主函数 main 也是一个调试程序，其目的就是验证 Area 的正确性。关于这个程序，需要说明以下问题。

1. 函数的返回值

例 1.3 第 5 行定义的函数 Area 不是用 void 作为 *类型修饰符*，因此是一个有返回值的函数，其返回值的类型由 *类型修饰符* 确定。函数 Area 是以 double 作为 *类型修饰符* 的，表明它的返回值的类型是双精度实数；此外，函数返回值还可以是 char（字符）、char*（字符串）、int（整数）、long（长整数）、float（单精度实数）等类型。调用一个有返回值的函数可以获得一个值，这样的函数调用本身就是一种特殊的表达式。例如，在例 1.3 第 11 行的赋值语句中，等号右边的表达式就是一个对函数 Area 的调用，所获得的值被赋值给了变量 area。

2. 变量及变量的数据类型

变量是相对于常量来说的，它的值可以在程序运行过程中加以改变。变量必须先定义后使用。变量定义的基本格式是：

```
类型修饰符　变量名表;
```

其中，*类型修饰符* 主要包括 char、char*、int、long、float、double 等，分别