教育部大学计算机课程改革项目规划教材

丛书主编 卢湘鸿

面向对象程序设计与C++

司慧琳 编著

清华大学出版社

教育部大学计算机课程改革项目规划教材

丛书主编 卢湘鸿

面向对象程序设计与C++

司慧琳 编著

清华大学出版社

北京

内 容 简 介

本书通过简单易懂的代码实例讲解、相应的课堂练习和大量的课后编程训练题帮助学生理解和掌握面向对象语言的三大特征：封装性、继承性和多态性。全书共12章，内容涉及类和对象、友元和静态、继承和组合、重载与多态性、异常处理等。

本书侧重面向对象程序设计的编程训练，为明确应训练的语法，所有编程题都提供了相应的输入输出测试用例，部分编程题目预设前置或后置代码。书后还附有4套模拟试卷和2个初学者问题集。

本书可配合Moodle平台使用。

本书适合作为高校计算机及相关专业面向对象程序设计(C++)课程的教材，还可作为广大读者学习面向对象程序设计的自学参考书。

图书在版编目(CIP)数据

面向对象程序设计与C++/司慧琳编著. —北京：清华大学出版社，2018
(教育部大学计算机课程改革项目规划教材)
ISBN 978-7-302-50310-1

Ⅰ. ①面…　Ⅱ. ①司…　Ⅲ. ①C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(2018)第112250号

责任编辑：谢　琛　战晓雷
封面设计：常雪影
责任校对：时翠兰
责任印制：宋　林

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
课件下载：http://www.tup.com.cn，010-62795954
印 装 者：三河市君旺印务有限公司
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：18.25　　**字　　数**：421千字
版　　次：2018年9月第1版　　**印　　次**：2018年9月第1次印刷
定　　价：49.00元

产品编号：074992-01

前言

面向对象程序设计(C++)是高等学校理工科专业课之一,是一门实践性很强的课程。本书适合作为高等院校“面向对象程序设计(C++)”课程的教材,还可以作为广大读者学习面向对象程序设计(C++)的自学参考书。

本书有5个特色:①每章围绕面向对象的一两个语法主题,提供简单易懂的代码实例讲解以及相应的课堂练习,来帮助学生理解和掌握本章内容;②侧重面向对象程序设计的编程训练,课后习题的代码总量超过3300行;③为明确应训练的语法,所有编程题都提供了相应的输入输出测试用例,部分编程题目预设前置或后置代码;④为帮助初学者检测学习效果,提供了4套模拟试卷;⑤提供了对于初学者来说具有很高参考价值的常见问题集锦,该内容来自作者多年教学过程中所整理的答疑记录,出版时作了精选和修订。

读者阅读本书之前需要具备基本的C语言程序设计能力,初步了解变量与数据类型、选择与循环、函数等基本语法。相比C语言的语法,C++面向对象的语法较晦涩难懂,初学者不容易掌握。为解决此问题,本书通过实例讲解和大量编程训练,帮助学生在实践经验积累的基础上领悟面向对象语言的三大特征:封装性、继承性和多态性。

本书共12章,涉及类和对象、友元和静态、继承和组合、重载与多态性、异常处理等内容。第1章介绍C++标准输入输出用法;第2章回顾结构体,介绍从结构体如何封装到类以及类图设计;第3章介绍对象初始化中各种构造函数的用法;第4章介绍字符串类、复制构造函数及析构函数;第5章介绍封装中基于项目的多文件管理以及文件与流操作;第6章介绍静态、友元等共享机制及保护;第7章介绍运算符重载;第8章介绍单继承和组合用法、继承中的同名覆盖规则;第9章介绍同名覆盖和赋值兼容以及继承中的基于项目的多文件管理;第10章介绍多继承及虚基类;第11章介绍虚函数的动态联编以及抽象类和纯虚函数等用法;第12章介绍C++的异常处理机制。使用本书完成教学大约需要68学时,其中包括34学时上机。

本书作者多年为计算机及相关专业学生讲授“面向对象程序设计(C++)”课程,积累了丰富的教学资源和实践经验。本书配有相应的教学辅助课件以及程序实例,使之更符合面向对象程序设计课程的要求,有需要者可与出版社联系。

本书的所有示例程序均在DevCpp 5.11上调试通过。

标有*的课后习题有一定难度,供学有余力的同学选做。

在本书策划和写作过程中,孙践知、孙悦红、张迎新、陈红倩、姚春莲等提出了很多宝贵意见,同时本书也参考了许多国内外的面向对象程序设计书籍,作者在此一并表示衷心感谢。

限于作者水平,书中难免有疏漏和不妥之处,恳请读者批评指正!

作　者

2018年1月

目录

第1章 C++的输入输出

1.1 C++的输入输出示例

［例1.1］ C的基本数据类型变量输入输出。

```
//L1_1.cpp
#include <stdio.h>
int main()
{
    char a;
    int b;
    float c;
    double d;
    scanf("%c",&a);                    //输入 char 类型的数据
    scanf("%d",&b);                    //输入 int 类型的数据
    scanf("%f",&c);                    //输入 float 类型的数据
    scanf("%lf",&d);                   //输入 double 类型的数据
    //也可以一次输入多个:scanf("%c%d%f%lf",&a,&b,&c,&d);
    printf("%c\n",a);                  //输出 char 类型的数据
    printf("%d\n",b);                  //输出 int 类型的数据
    printf("%f\n",c);                  //输出 float 类型的数据
    printf("%f\n",d);                  //输出 double 类型的数据
    //也可以一次输出多个:printf("%c\n%d\n%f\n%f\n",a,b,c,d);
    return 0;
}
```

输入

```
A 12 12.34 12.3456789
```

输出

```
A
12
12.340000
12.345679
```

从例1.1的代码来看：C的输入函数 scanf 和输出函数 printf 需要针对不同的数据类型的变量编写相应的格式控制。

★**注意**：%f 格式控制在输出时不区分 float 或 double，默认都是 6 位小数，不足补 0，超过即四舍五入保留 6 位小数。

C++ 的输入对象 cin 和输出对象 cout 则可以根据变量的类型自动解析数据的对应类型，不需要考虑烦琐的格式控制，比 C 语言的输入函数 scanf 和输出函数 printf 简化了许多。

[**例 1.2**] C++ 的基本数据类型变量输入输出。本例是例 1.1 的 C++ 版。

```
//L1_2.cpp
#include <iostream>
using namespace std;
int main()
{
    char a;
    int b;
    float c;
    double d;
    cin>>a;                                    //输入 char 类型的数据
    cin>>b;                                    //输入 int 类型的数据
    cin>>c;                                    //输入 float 类型的数据
    cin>>d;                                    //输入 double 类型的数据
    //也可以一次输入多个,用>>分隔即可:cin>>a>>b>>c>>d;
    cout<<a<<endl;                             //输出 char 类型的数据
    cout<<b<<endl;                             //输出 int 类型的数据
    cout<<c<<endl;                             //输出 float 类型的数据
    cout<<d<<endl;                             //输出 double 类型的数据
    //也可以一次输出多个,用<<分隔即可:cout<<a<<endl<<b<<endl<<c<<endl;
    return 0;
}
```

输入

```
A12 12.34 12.3456789
```

输出

```
A
12
12.34
12.3457
```

★**注意**：float 类型或者 double 类型的数据有几位小数，cout 就默认输出几位小数，不自动补零。如果整数位数和小数位数之和超过 6 位，则四舍五入保留输出 6 位。如果想对有效位数进行控制，可以参考例 1.3 的代码。

例 1.2 代码涉及的 C++ 语法简介：

(1) cpp 是 C++ 源文件的扩展名。

(2) C++ 的头文件扩展名为 h 或者没有扩展名，一般自定义的头文件使用扩展名 h，使

用 C++ 标准库的头文件没有扩展名，例如：#include <iostream>，这表示允许程序中使用 cin 和 cout。

★注意：当使用<iostream>时，由于 C++ 的标准库中所有标识符都被定义在一个名为 std 的 namespace 中，因此其后必须带上语句“using namespace std;”。

(3) 行注释以两个连续的斜线(//)开始，其注释范围仅限当前行。注释的文字说明在程序编译时被编译器忽略。

(4) cin 是 C++ 语言的标准输入流对象，在默认情况下是从键盘输入。cout 是 C++ 语言的标准输出流对象，在默认情况下是输出到显示器。

cin 的基本用法为

```
cin>>V1>>V2>>…>>Vn;
```

>>是预定义的提取符，用于从一个输入流对象获取字节。

在输入时，应注意用空格或制表符(Tab 键)将输入的数据分隔开。例如：

```
cin>>x>>y;
```

执行到该语句时，输入 x 和 y 值，即从键盘输入

```
3 5
```

3 和 5 之间用空格或制表符分隔。

★注意：输入数据的类型应与接收该数据的变量的类型相匹配，否则输入操作将会失败或者得到的是一个错误的数据。

cout 的基本用法为

```
cout <<E1<<E2<<…<<Em;
```

<<是预定义的插入符，用于传送字节到一个输出流对象。

★注意：输出时要恰当使用字符串和换行符 endl，提高输出信息的可读性。

例如：

```
cout <<x <<" +" <<y <<" =" <<sum <<endl;
```

[**例 1.3**]　有效位数输出控制。

```
//L1_3.cpp
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{   float a=23.4538769;
    cout <<"a="<<a<<endl;
    cout<<"a="<<setprecision(4)<<a<<endl;      //输出整数+小数的有效位数,a=23.45
    cout<<"a="<<fixed<<setprecision(4)<<a<<endl;
                                               //输出小数部分有效位数,a=23.4539
    return 0;
}
```

输出

```
a=23.45
a=23.4539
```

从例 1.3 的代码来看，为了使精度设置的含义体现为小数位数，必须提前把浮点数的输出方式设置为定点方式，即 fixed。

[例 1.4] 字符串输入输出。

```
//L1_4.cpp
#include <iostream>
using namespace std;
int main()
{
    char s[100];
    cin>>s;                    //输入不带空格的字符串数据，如果带空格，需要用 cin.get(s,100);
    cout<<s<<endl;             //输出字符串数据，可带空格，也可不带空格
    return 0;
}
```

输入

```
Hello
```

输出

```
Hello
```

[例 1.5] 普通数组的输入输出。

```
//L1_5.cpp
#include <iostream>
using namespace std;
int main()
{   int t[5],i;
    for(i=0;i<5;i++)
        cin>>t[i];             //输入数组元素
    for(i=0;i<5;i++)
        cout<<t[i]<<endl;  //输出数组元素
    return 0;
}
```

输入

```
1 2 3 4 5
```

输出

```
1
2
3
```

```
4
5
```

[例 1.6]　结构体变量的输入输出。

```
//L1_6.cpp
#include <iostream>
using namespace std;
struct STUSCORE {
    char strName[12];                    //姓名
    int iStuNO;                          //学号
    float fScore;                        //某门课程成绩
};
int main()
{   STUSCORE one;
    cin>>one.strName>>one.iStuNO>>one.fScore;
    cout<<"姓名:"<<one.strName<<",学号:"<<one.strStuNO
    <<",成绩:"<<one.fScore<<endl;
    return 0;
}
```

输入

```
wang 111 78.5
```

输出

```
姓名:wang ,学号:111, 成绩:78.5
```

★注意：

(1) C++ 中结构体的关键字 struct 在定义变量时是可以省略的，例如定义结构体变量 one 时，直接使用 STUSCORE one 即可。

(2) 在结构体定义中，成员都是公开的，因此在外部(如 main 函数中)使用直接采用“结构体变量名.成员名”即可，比如 one. strName。

1.2　面向过程的程序设计

面向过程(procedure oriented)是一种以过程为中心的编程思想。在面向过程的程序设计中，程序是处理数据的一系列过程。过程(或函数)定义为实现特定功能的一组指令，其设计思想是功能分解并逐步求精，数据与程序过程分开存储。在面向过程的程序设计中，程序块是由函数构成的。

面向过程的程序设计由顺序、选择、循环 3 种逻辑结构组成，通过函数和模块来组织程序，因此编程的主要技巧在于模块之间的调用关系及数据的变化上。

[例 1.7]　计算学生 3 门课的平均成绩。

```
// L1_7.cpp
```

```
#include <iostream>
using namespace std;
struct STUSCORE {
    char strName[12];                                    //姓名
    int iStuNO;                                          //学号
    float fScore[3];                                     //3门课程成绩
};
float GetAverage(STUSCORE one)                           //计算平均成绩
{   return (float)((one.fScore[0]+one.fScore[1]+one.fScore[2])/3.0);
}
int main()
{   STUSCORE one={"LiMing",21020501,{80,90,65}};        //结构体变量定义省略 struct
    cout<<one.strName<<"的平均成绩为:"<<GetAverage(one)<<endl;
    return 0;
}
```

输出

```
LiMing的平均成绩为:78.3333
```

★**注意**：在结构体定义中，只描述了数据的不同属性特征(成员)，因此，相关操作(函数)要访问数据的属性时需要参数传递，即GetAverage必须是带参数的函数。

C++新增了引用的用法。引用是一种特殊类型的变量，可以被认为是另一个变量的别名。引用运算符 & 用来说明一个引用。定义引用的语法格式如下：

数据类型 &引用名 =已定义的变量名;

例如：

```
int a =10;
int &i =a;                                   //引用 i 与变量 a 占用同一个内存单元:i=10,a=10
i =i +100;                                   //i=110,a=110
```

★**注意**：引用不是普通变量，它本身没有值和地址值，引用的地址值是它被绑定的变量或者对象的地址值，它的值也是被绑定变量的值。引用与指针不同，指针存储目标单元的地址，引用直接指向目标单元。

由于增加了引用，函数的实参与形参有3种结合方式：值调用、传址调用和引用调用。

(1) 值调用：形参和实参都是普通变量或值，在函数调用时，将实参的值赋给形参，但形参值的变化不影响实参，这一特点被称为单向值传递。

[**例1.8**] 交换两个整形变量的值(值调用)。

```
// L1_8.cpp
#include <iostream>
using namespace std;
void fun (int x, int y)                      //形参是普通变量
{   int tmp =x;
    x =y;
```

```
    y =tmp;
}
int main()
{   int a =1,b =2;
    cout <<"Before exchange:a=" <<a <<",b=" <<b <<endl;
    fun(a, b);                          //实参是普通变量
    cout <<"After  exchange:a=" <<a <<",b=" <<b <<endl;
    return 0;
}
```

输出

```
Before exchange:a=1,b=2
After  exchange:a=1,b=2
```

从例 1.8 的输出来看，值调用(单向的值传递)不能实现交换两个整形变量的值。

(2) 传址调用：形参和实参是指针或者地址，在函数调用时，形参的指针获取实参传来的地址，形参通过地址访问间接改变实参地址单元中存放的值，这一特点被称为双向地址传递。

[**例 1.9**]　交换两个整形变量的值(传址调用)。

```
// L1_9.cpp
#include <iostream>
using namespace std;
void fun (int * x, int * y)             //形参是指针
{   int tmp = * x;
    * x = * y;
    * y =tmp;
}
int main()
{   int a =1,b =2;
    cout <<"Before exchange:a=" <<a <<",b=" <<b <<endl;
    fun(&a, &b);                        //实参是地址
    cout <<"After  exchange:a=" <<a <<",b=" <<b <<endl;
    return 0;
}
```

输出

```
Before exchange:a=1,b=2
After  exchange:a=2,b=1
```

从例 1.9 的输出来看，传址调用(双向的地址传递)通过访问地址间接实现交换两个整形变量的值。

(3) 引用调用：在形参名前加上引用说明符 & 即将其声明为引用，实参则直接采用一般的变量名。在函数调用时，形参就成了实参的别名，对引用的操作就等同于对主调函数中原变量的操作。

[例 1.10] 交换两个整形变量的值(引用调用)。

```
// L1_10.cpp
#include <iostream>
using namespace std;
void fun (int &x, int &y)                    //形参是引用
{   int tmp =x;
    x =y;
    y =tmp;
}
int main()
{   int a =1,b =2;
    cout <<"Before exchange:a=" <<a <<",b=" <<b <<endl;
    fun(a, b);                               //实参是变量
    cout <<"After  exchange:a=" <<a <<",b=" <<b <<endl;
    return 0;
}
```

输出

```
Before exchange:a=1,b=2
After  exchange:a=2,b=1
```

从例 1.10 的输出来看,引用调用通过引用(变量的别名)直接实现交换两个整形变量的值。引用调用时,实参要用变量名,形参作为实参变量名的引用,对形参的改变就是对实参的直接改变。因此引用调用的效果与传址调用相同,但是比其更方便、直接。

引用的目的主要是在函数参数传递中解决大对象的传递效率低和存储空间占用较多的问题。

★**注意**:C++ 中 swap 函数的参数就采用了引用的形式,因此为了避免与系统函数 swap 混淆,例 1.8、例 1.9、例 1.10 用 fun 作为函数名。

1.3 课堂练习

有一筐鸡蛋,1 个 1 个拿,正好拿完;2 个 2 个拿,还剩 1 个;3 个 3 个拿,正好拿完;4 个 4 个拿,还剩 1 个;5 个 5 个拿,还剩 1 个;6 个 6 个拿,还剩 3 个;7 个 7 个拿,正好拿完;8 个 8 个拿,还剩 1 个;9 个 9 个拿,正好拿完。筐里有多少鸡蛋?

C 语言程序代码如下:

```
#include <stdio.h>
int main()
{   int i;
    for(i=1;;i++)
    if(i%2==1&&i%3==0&&i%4==1&&i%5==1&&i%6==3&&i%7==0&&i%8==1&&i%9==0)
    {   printf("%d\n",i);
        break;
```

```
    }
    return 0;
}
```

(1) 将上述 C 语言代码改为 C++ 程序，相关输入输出使用 cin 或 cout。

(2) 如果需要找到 n 个满足条件的答案，如何修改？n 从键盘输入。例如输入 1，输出 1 个满足条件的答案(即 441)；输入 5，输出 5 个满足条件的答案(即 441、2961、5481、8001、10521)。

1.4　课后习题

本章训练侧重 cin/cout 的用法。部分题目提供前置或后置的预设代码，预设代码内容和顺序不可改变，只要在预设代码框架上将程序补充完整即可。

题 1.1　圆的面积计算。从键盘上输入半径，输出圆面积，π 取 3.14。例如，输入

```
10
```

输出

```
面积=314
```

题 1.2　两数求和。从键盘输入两个整数，输出其和。例如，输入

```
23 45
```

输出

```
68
```

题 1.3　子串分割。从键盘上输入一个满足格式要求的字符串(形如“A1,234”)，编程将其从分隔符(即逗号)位置分割成两个部分(如 A1 和 234 两个字符串)，并在屏幕上分两行依次显示分割后的结果。例如，输入

```
A1,234
```

输出

```
A1
234
```

题 1.4　字母大小写转换。从键盘输入一个大写字母，要求输出其小写字母。

提示：小写字母的 ASCII 码值为对应的大写字母的 ASCII 码值加 32。

例如，输入

```
A
```

输出

```
a
```