21世纪高等学校计算机规划教材

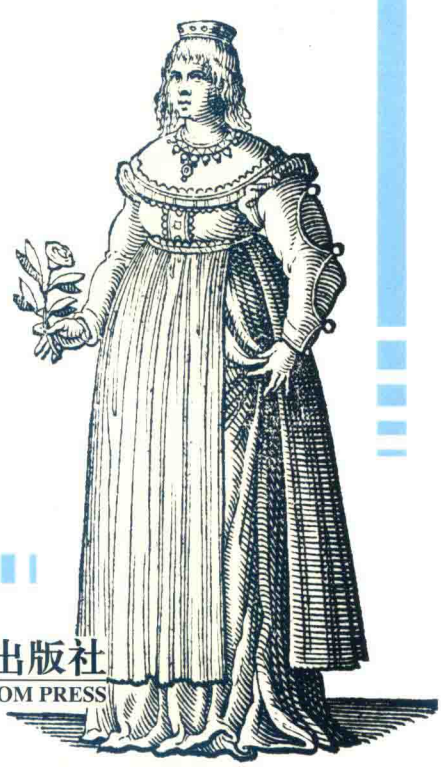
COMPUTER

C++面向对象程序设计教程

C++ Object-Oriented Programming Course

杜青 解芳 李春颖 王丹华 编著

- 重点突出、难点化解
- 例题丰富、传授编程思想
- 提供实训、注重编程实践
- 内容符合全国计算机等级考试二级



中国工信出版集团

人民邮电出版社
POSTS & TELECOM PRESS

21世纪高等学校计算机规划教材

COMPUTER

C++面向对象程序设计教程

C++ Object-Oriented Programming Course

杜青 解芳 李春颖 王丹华 编著

人民邮电出版社

北京

图书在版编目（CIP）数据

C++面向对象程序设计教程 / 杜青等编著. -- 北京 : 人民邮电出版社, 2017.1
21世纪高等学校计算机规划教材
ISBN 978-7-115-44411-0

Ⅰ. ①C… Ⅱ. ①杜… 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(2017)第000278号

内 容 提 要

本书介绍了 C++面向对象程序设计的基本概念和编程方法，内容包括类与对象、静态成员与友元、运算符重载、继承与派生、输入输出流、异常处理等，详细阐述了 C++面向对象程序设计的四个特性，即抽象性、封装性、继承性和多态性。

本书给出了大量的例题，通过简单的例题，分析面向对象程序设计基本概念的内在含义，使抽象的概念具体化、形象化；同时将难点问题分散到多个例题中，结合具体实例，由浅入深进行讲述，便于初学者在短时间内理解和掌握面向对象程序设计的思想和方法。每章还给出一定数量的习题，方便读者对本章内容的复习、巩固。

本书可作为高等学校“C++面向对象程序设计”课程的教材，也可作为具有 C 语言程序设计基础的开发人员进一步学习 C++面向对象程序设计的参考书。

◆ 编　　著　杜　青　解　芳　李春颖　王丹华
责任编辑　张孟玮
执行编辑　李　召
责任印制　沈　蓉　彭志环

◆ 人民邮电出版社出版发行　　北京市丰台区成寿寺路 11 号
邮编　100164　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
三河市海波印务有限公司印刷

◆ 开本：787×1092　1/16
印张：17　　2017 年 1 月第 1 版
字数：445 千字　　2017 年 1 月河北第 1 次印刷

定价：42.00 元

读者服务热线：(010)81055256　印装质量热线：(010)81055316
反盗版热线：(010)81055315

前 言

本书是为具有C语言程序设计基础,想要进一步学习C++面向对象程序设计的读者而编写的。

全书共分9章，介绍了C++面向对象程序设计的基本概念和编程方法，内容包括类与对象、静态成员与友元、运算符重载、继承与派生、输入输出流、异常处理等，详细阐述了C++面向对象程序设计的四个特性，即抽象性、封装性、继承性和多态性。

本书在内容编排上具有如下特点。

（1）重点突出

由于C++面向对象程序设计涉及的知识点非常多，为了帮助初学者理解和领会面向对象程序设计的思想和方法，本书对重要的概念，如类的概念、类的继承机制等，进行详细的阐述。通过简单的例题，分析基本概念的内在含义，每个例题后面给出运行结果和程序说明，使抽象的概念具体化、形象化，便于初学者在短时间内理解和掌握。

（2）难点化解

对于难点问题，如构造函数、运算符重载等内容，本书结合具体实例，由浅入深进行分析，将难点问题分散到多个例题中，降低初学者学习的难度。

（3）例题丰富

本书的每一章，都给出了大量的例题，其中不仅有简单的基础例题，使读者理解面向对象程序设计的基本概念，还有较复杂的拓展例题，使读者掌握面向对象程序设计的基本方法。在最后一章给出了两个综合实例，介绍了用面向对象程序设计方法开发应用程序的基本方法与步骤，读者可参照实例，写出自己的应用程序。

每章还提供了一定数量的习题，方便读者在学完本章内容后复习所学知识。

本书由杜青、解芳、李春颖、王丹华编写。其中第1章由李春颖编写，第2、3章由解芳编写，第4、5、9章及附录由杜青编写，第6章由李春颖、解芳共同完成，第7、8章由王丹华编写，全书由杜青策划，杜青、解芳统稿。

本书凝聚了作者多年来积累的教学经验。本书在编写过程中得到了南京工程学院计算机工程学院各级领导和同事的大力支持，徐金宝、廖雷老师也为本书提出了很多好的建议和意见，在此表示衷心的感谢。

目 录

第1章 C++基础知识

1.1 C++概述

C++是一种优秀的面向对象程序设计语言。C++以 C 语言为基础，既保留了传统的结构化程序设计方法，又对流行的面向对象的程序设计提供了完整支持。此外，C++语言还具有许多 C 语言不支持的新功能和新特性。

1.1.1 C++语言的历史和特点

1. C++语言的历史

C++是由 AT&T Bell 实验室的 Bjarne Stroustrup 及其同事于 20 世纪 80 年代初在 C 语言的基础上成功开发的。C++保留了 C 语言原有的所有优点，并增加了面向对象的机制。由于 C++对 C 的改进主要体现在增加了面向对象程序设计的“类”，因此最初它被 Bjarne Stroustrup 称为“带类的 C”。后来为了强调它是 C 的增强版，使用了 C 语言中的自加运算符“++”，改称为 C++。

C++在 C 语言的基础上增加的主要特性有：公有和私有成员的区分、类及派生类、类的赋值运算符、构造函数和友元、内联函数、析构函数及重载等。

1985 年公布的 C++语言中增添了其他一些重要特性，如：函数和运算符的重载、虚函数的概念等。新增加的内容有：对象的初始化与赋值的递归机制、多重继承等。

1993 年的版本对 C++语言进一步完善，其中最重要的特性是模板，而且解决了多重继承产生的二义性问题和相应的析构函数的处理问题。

1998 年的 C++版本得到了国际标准化组织和美国标准化协会的批准，标准模板库中增加了命名空间的概念、标准容器类、通用算法类和字符串类型等让 C++语言变得更实用。

2003 年通过的 C++版本，是一次技术性修订，修订了第一版中的错误，并减少了多义性。

2. C++语言的特点

在众多的高级程序设计语言中，C++有其独特之处。

（1）C++是 C 语言的超集

C++是 C 语言的超集指的是 C++中包含了 C 语言的全部语法特性。它比 C 语言更简洁、高效地接近汇编语言特点，对 C 系统进行了扩充，C++的编译系统能检查出更多的类型错误，因此 C++比 C 安全。

（2）C++支持面向对象的程序设计方法

C++语言支持面向对象程序设计特性主要包括抽象数据类型、封装和信息隐藏、以继承和派生方式实现程序的重用、以虚函数实现多态性、以模板实现类型的参数化等。

C++正是从软件的可靠性、可重用性、可扩充性、可维护性等方面体现出它的优越性。

1.1.2 C++程序与C程序

C++语言与C语言保持兼容，从程序结构上看，C++程序与C程序有很多相同之处。为了对C++程序结构有一个初步了解，下面通过例子比较C++程序与C程序有何异同。

例 1-1 一个简单程序。

（1）用C语言编写的一个简单程序。

```
//这是一个简单的C程序：simple.c
#include<stdio.h>
void main()
{   printf("Hello World!\n");    }    //输出字符串
```

（2）用C++语言编写的一个简单程序。

```
//这是一个简单的C++程序：simple.cpp
#include<iostream>
using namespace std;
int main()
{   cout<<" Hello World! "<<endl;    //输出字符串
    return 0;                        //没有此句会有警告。最好加上此句
}
```

程序运行结果：

```
Hello World!
```

程序说明：

（1）C程序源文件的扩展名为C，而C++程序源文件的扩展名为CPP。

（2）C++程序系统头文件不带后缀.h，在“#include<iostream>”语句下增加“using namespace std;”。输入输出通过使用输入输出流对象（如cin、cout）来完成。

例 1-2 用函数和类来实现简单程序。

（1）用一般函数编写的一个简单程序。

```
#include<iostream>
using namespace std;
void output()
{  cout<<" Hello World! "<<endl;}
int main()
{   output();
    return 0;
}
```

（2）用类编写的一个简单程序。

```
#include<iostream>
using namespace std;
class Simple
{public:
```

```
    void output()
    {    cout<<" Hello World! "<<endl; }
};
int main()
{   Simple s;
    s.output();
    return 0;
}
```

程序运行结果：

```
Hello World!
```

程序说明：

在该程序中用到了类，这在其后内容中会着重介绍。

1.1.3 C++对C的扩充

1. 函数原型声明

在C++中，如果函数调用的位置在函数定义之前，则要求在函数调用之前必须对所调用的函数作函数原型声明，这是强制性的。这种声明在标准C++中称为函数原型（function prototype），函数原型给出了函数名、返回类型以及在调用函数时必须提供的参数个数和类型。

函数原型的语法为：

<返回类型><函数名>(<参数类型列表>)；（注意在函数原型后要有分号）

如有定义：

```
int max(int a,int b)
{   return a+b;   }
```

声明时必须写成：

```
int max(int a,int b);或 int max(int ,int);
```

在C++中，调用任何函数之前，必须确保它已有函数原型声明。函数原型声明通常放在程序文件的头部，以使得该文件中的所有函数都能调用它们。实际上，标准函数的原型声明放在了相应的头文件中，这也是在调用标准函数时必须包含相应的头文件的原因。

2. 函数的重载

在C语言中，同一作用域中不能有同名的函数，每个函数必须有其唯一的名字，这样有时会令人生厌。例如，求几个数中的最大值，由于要求命名唯一，会起不同的名字。

```
int max(int,int);
int smax(int,int, int);
float fmax(float,float);
```

以上函数都是求最大值，函数却起了不同的名字，C++为了方便程序员编写程序，特别引入了函数重载的概念来解决此问题。

C++允许在同一作用域中用同一函数名定义多个函数，这些函数的参数个数或参数类型不相同，这些同名的函数用来实现不同的功能，这就是函数的重载。

例如，上述几个函数的声明可以改为如下。

```
int max(int,int);
int max(int,int, int);
float max(float,float);
```

C++用一种函数命名技术可以准确判断出应该使用哪个 max()函数。

在调用一个重载函数时，编译器必须搞清所调用的函数究竟是哪个函数。这是通过实参类型和所有被调用函数的形参类型比较来判定的。按下述 3 个步骤的先后顺序找到并调用该函数。

（1）寻找一个严格的匹配，如果找到了，就用该函数。

（2）通过内部转换寻求一个匹配，只要找到了，就用该函数。

（3）通过用户定义的转换寻求一个匹配，若能查出有唯一的一组转换，就用该函数。

如：

```
max(-10,5);              //调用 int max(int,int);
max(-10,5,-20);          //调用 int max(int,int, int);
max(-12.23,1.2);         //调用 float max(float,float);
max(1.23f,5);            //调用 float max(float,float)
```

定义重载函数时，需要注意以下几点：

（1）C++的函数如果在参数类型、参数个数、参数顺序上有所不同，则认为是不同的。但重载函数如果仅仅是返回类型不同，则是不够的。

例如，下面的声明是错误的。

```
int  max(int ,int);
void  max(int ,int);
```

编译器无法区分函数调用“max(3,5)”是上述哪一个重载函数。因此重载函数至少在参数个数、参数类型或参数顺序上有所不同。

（2）typedef 定义的类型与一个已存在的类型相同，而不能建立新的类型，所以不能用 typedef 定义的类型名来区分重载函数声明中的参数。

例如，下面的代码实际上是同一个函数。

```
typedef INT int;
void func(int x){//...}
void func(INT x){//...} //error: 函数重复定义
```

编译器不能区分这两个函数的差别，INT 只不过是 int 的另一种称呼而已。

（3）让重载函数执行不同的功能，是不好的编程风格。同名函数应该具有相同的功能。如果定义一个 abs()函数而返回的却是一个数的平方根，则该程序的可读性受到破坏。

例 1-3 用一个函数求 2 个正整数或 3 个正整数的最大者。

```
#include<iostream>
using namespace std;
int max(int a,int b)
{   return a>b?a:b;
}
int max(int a,int b,int c)
{   return (a>b?a:b)>c?(a>b?a:b):c;
}
int main()
{   int a,b,c;
    cin>>a>>b>>c;
```

```
    cout<<max(a,b,c)<<"   "<<max(a,b)<<endl;
    returne 0;
}
输入3 5 4
```

程序运行结果：

```
5
```

程序说明：

（1）例中分别给 max 函数 2 个实参和 3 个实参，程序运行时，根据所传递的实参个数不同，调用相应的函数，得到了期望的结果。

（2）如果函数只是返回类型不同，而其他完全相同（参数的个数及类型），则不能作为重载函数来使用。

3. 有默认参数的函数

（1）默认参数的目的

C++可以给函数定义默认参数值。通常调用函数时，要为函数的每个参数给定对应的实参。如有以下函数声明和函数定义：

```
void delay(int loops); //函数声明
void delay(int loops) //函数定义
{    if(loops==0) return;
     for(int i=0;i<loops,i++);//为了延时
}
```

无论何时调用 delay()函数，都必须给 loops 一个实参以确定时间。但有时需要用相同的实参反复调用 delay()函数。C++可以给参数定义默认值。如果将 delay( )函数中的 loops 定义成默认值 1000，只需简单地把函数声明改为：

```
void delay(int loops=1000);
```

这样，调用 delay()函数时，不给出实参，程序会自动将实参当作值 1000 进行处理。

```
delay(2500);  //loops 设置为 2500
delay();      //ok: loops 采用默认值 1000
```

调用中，若不给出参数，则按指定的默认值进行工作。允许函数默认参数值，是为了让编程简单，让编译器做更多的检查错误工作。

（2）默认参数的声明

默认参数在函数声明中提供，当既有声明又有定义时，定义中不允许出现默认参数。如果函数只有定义，则默认参数可出现在函数定义中。

```
void point(int=3,int=4); //声明中给出默认值
void point(int x,int y) //定义中不允许再给出默认值
{    cout <<x<<endl;
     cout <<y<<endl;
}
```

（3）默认参数的顺序规定

如果一个函数中有多个默认参数，则形参分布中，默认参数应从右至左逐渐定义。当调用函数时，只能向左匹配参数。例如：

```
void func(int a=1,int b,int c=3, int d=4);  //error
void func(int a,int b=2,int c=3,int d=4);  //ok
void func(int a,int b,int c=3,int d=4);   //ok
void func(int a,int b,int c,int d=4);     //ok
```

例 1-4 阅读程序，观察调用默认参数的顺序。

```
#include<iostream>
using namespace std;
void f(int a,int b=2,int c=3,int d=4);
int main()
{   f(5);
    f(5,6);
    f(5,6,7);
    f(5,6,7,8);
    return 0;
}
void f(int a,int b,int c,int d)
{   cout<<a<<"  "<<b<<"  "<<c<<"  "<<d<<endl; }
```

程序运行结果：

```
5  2  3  4
5  6  3  4
5  6  7  4
5  6  7  8
```

程序说明：

该程序说明默认参数从右至左逐渐定义，调用函数时，向左匹配参数。

（4）默认参数与函数重载

默认参数可将一系列简单的重载函数合成为一个。如例 1-4 中的函数 f 是下面 4 个重载函数。

```
void f(int a)
{   cout<<a<<"  "<<2<<"  "<<3<<"  "<<4<<endl;}
void f(int a,int b)
{   cout<<a<<"  "<<b<<"  "<<3<<"  "<<4<<endl;}
void f(int a,int b,int c)
{   cout<<a<<"  "<<b<<"  "<<c<<"  "<<4<<endl;}
void f(int a,int b,int c,int d)
{   cout<<a<<"  "<<b<<"  "<<c<<"  "<<d<<endl;}
```

如果一组重载函数（可能带有默认参数）都允许相同个数实参的调用，将会引起调用的二义性。例如：

```
void f(int);                          //重载函数之一
void f(int,int=2,int=3,int=4);        //重载函数之二，带有默认参数
void f(int,int ,int =3,int=4);        //重载函数之三，带有默认参数
f(10);                  //error:到底调用 3 个重载函数中的哪个?
f(20,30)                //error: 到底调用后面 2 个重载函数中的哪个?
```

例 1-5 将例 1-3 不用重载，改用带有默认参数的函数实现。

```
#include<iostream>
using namespace std;
int max(int a,int b,int c=0)
{   return (a>b?a:b)>c?(a>b?a:b):c;
}
```

```
int main()
{   int a,b,c;
    cin>>a>>b>>c;
    cout<<max(a,b,c)<<"   "<<max(a,b)<<endl;
    return 0;
}
```

输入 3 5 4

程序运行结果：

```
5   5
```

程序说明：

本例题说明默认参数可将一系列简单的重载函数合成为一个具有默认参数的函数，完成重载功能。

1.2 C++的输入输出

C++为了方便用户，除了可以用 printf 和 scanf 函数进行输入与输出外，还增加了标准输入与输出流对象 cout 和 cin。它们是在文件 iostream 中定义的。

实际上，C++在内存中为每一个数据流开辟一个内存缓冲区，用来存放流中的数据。当用 cout 和插入运算符<<向显示器输出数据时，先将这些数据插入到输出流中送到输出缓冲区保存，直到缓冲区满了或者遇到 endl，就将缓冲区全部数据送到显示器显示出来。在输入时，从键盘输入的数据先放在键盘的缓冲区，形成 cin 流，然后用提取运算符>>从输入缓冲区提取数据送给程序中的有关变量。总之，流是与内存缓冲区相对应的，或者说，缓冲区中的数据就是流。

输入和输出是数据传送的过程，数据就如流水一般从一处流向另一处。C++形象地把这个过程称为流（stream）。

前面曾多次说明 cout 和 cin 并不是 C++语言中所提供的语句，它们是 iostream 对象。在未学习类对象时，在不致引起误解的情况下，为叙述方便，把它们称为 cout 语句和 cin 语句。

C++将所有的程序都以下面这两行代码开始。

```
#include<iostream>
using namespace std;
```

这两行代码使 iostream 流进入可用状态，cin 和 cout 的定义就包含在这个文件里。using namespace std;这个特定的指令表明程序准备使用 std 命名空间。

1.2.1 用 cout 进行输出

cout 是由 c 和 out 两个单词组成的，代表 C++的输出流对象，C++预定义 cout 代表标准输出设备，即显示器。标准流是不需要打开和关闭文件即可直接操作的流式文件。cout 是输出流对象的名字。输出操作由操作符<<来表达，其功能是将紧随其后的双引号中的字符串输出到标准输出设备（显示器）上。传递给 cout 对象的任何值将在屏幕上显示。cout 必须和输出运算符<<一起使用。<<在 C 语言中作为位运算中的左移运算符，在 C++语言中又被赋予新的含义为“插入”（inserting），作为输出信息时的插入运算符。

cout 语句的一般格式为：**cout<<表达式 1<<表达式 2<<……<<表达式 n;**

当程序需要在屏幕上显示输出时，可以使用插入操作符<<，向 cout 输出流中插入字符。例如：cout<<"This is a program.\n"。

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，<<运算符的结合方向是自左向右的，因此各输出项按自左向右顺序插入到输出流中。

用 cout 和<<可以输出各种类型的数据。如:

```
float a=3.45;
int b=5;
char c='A';
cout<<"a="<<a<<","<<"b="<<b<<","<<"c="<<c<<endl;
```

每输出一项都要用一个输出流符号。不能写成“cout<<a,b,c”这种形式。

可以看出来的是：在 C++在实现输出的时候一般会自动地按照数据的类型进行输出。

例 1-6 使用流插入运算符输出基本数据类型。

```
#include<iostream>
using namespace std;
int main()
{   cout<<88<<"\t"<<59.5<<"\n";
    cout<<"ZWQ"<<ends<<endl;
    int n=1,*p=&n;
    cout<<p<<"  "<<(unsigned long)p<<endl;
    char* s="ZWY";
    cout<<s<<"\t"<<(void*)s<<endl;
    return 0;
}
```

程序运行结果：

```
88  59.5
ZWQ
0012FF60  1245024
ZWY   00417700
```

程序说明：

通过本例可以看出各输出项按自左向右顺序插入到输出流中。

1.2.2 用 cin 进行输入

在 C++中，从输入设备向内存流动的数据流称为输入流，cin 是输入流对象的名字。用 cin 来实现从系统默认的标准输入设备（键盘）向内存流动的数据流称为标准输入流。用>>运算符从输入设备键盘取得数据并送到输入流 cin 中，然后送到内存。在 C++中，这种输入操作称为提取。所以>>常被称为提取运算符。

cin 语句的一般格式为：**cin>>变量 1>>变量 2>>……>>变量 n;**

当程序需要执行键盘输入时，可以使用提取操作符">>"，从 cin 输入流中提取字符。从键盘输入的数据类型应和变量一致。也可以连续用>>，实现从键盘到多个变量的输入形式。各数据之间要有分隔符，分隔符可以是一个或多个空格键、制表键或回车键。

例如：

```
int a;          //定义整形变量 a
float b;        //定义浮点型变量 b
```

```
cin>>a>>b;        //从键盘接受一个整数和一个实数，注意不要写成cin>>a,b
```

如果在运行时从键盘输入

```
20 30.45                    (两个数字以空格分开)
```

这时变量a，b分别获得值20和30.45。用cin和>>输入数据同样不需要指定数据类型。

例1-7 cin的使用。

```
#include <iostream>
using namespace std;
int main()
{   char d;
    int i;
    float z,q;
    cin>>i>>z>>q;
    d=i;
    cout<<"d="<<d<<"\ti="<<i;
    cout<<"\tz="<<z<<"\tq="<<q<<endl;
    return 0;
}
```

如果这时从键盘输入数据（一个整数和两个实数），中间用一个或多个空格键作为分隔符。如输入：

```
88  2.1  3.8
```

程序运行结果：

```
d=X  i=88  z=2.1  q=3.8
```

程序说明：

字符变量d和整型变量i的值都是88，但输出的形式不同。

1.3 引　　用

引用是C++引入的新语言特性，是C++常用的重要内容之一。首先，想要改变实参的值，可以用指针完成，但是使用引用之后程序变得更加简单。其次，传递函数参数的方式是传值，在函数域中为形参重新分配内存，而把实参的数值传递到新分配的内存中，或者实参是一个复杂的对象，要为形参重新分配内存，就会使得程序执行效率大大下降。使用引用之后可以使程序变得更加高效。

1.3.1 引用的概念

引用是C++对C语言的重要扩充，变量的引用就是变量的别名，对引用的操作与对变量直接操作完全一样，声明引用的方法与定义指针相似，只是用&代替了*。

引用的声明方法如下所示：**类型标识符 &引用名=目标变量名；**

举例如下：定义引用pa，它是变量a的引用，即别名。

```
float a;        //声明a为一个浮点型变量
float &pa=a;    //声明pa是一个浮点型变量的引用，它被初始化为a
```

这就声明了pa是a的“引用”，即a的别名。经过这样的声明后，使用a或pa的含义相同，都代表同一变量。注意：在上述声明中，&都是“引用声明符”，此时它并不代表地址。不要理解

为“把 a 的值赋给 pa 的地址”。对变量声明一个引用，引用不是变量，所以并不另开辟内存单元，pa 和 a 都代表同一个内存单元。在声明一个引用时，必须同时使之初始化，即声明它代表哪一个变量，和所关联的变量享受同等的访问待遇。

应用引用时需要注意以下几点。

（1）在声明一个引用时，必须告知到底是哪个变量被引用。引用必须在定义时同时被初始化，因为它必须是某个变量的别名，不能先定义一个引用后才初始化它。

例如下面语句是非法的。

```
int a=1,b=2;
int &tt=a;      //正确，引用 tt 被初始化
int &t;         //错误，引用 t 没有被初始化
```

（2）当一个引用声明完毕后，相当于目标变量名有两个名称，即该目标原名称和引用名称，在本函数执行期间，该引用一直与其代表的目标变量相联系，不能再把该引用名作为其他变量名的别名。

```
int a=1,b=2;
int &tt=a;      //tt 被初始化为 a 的引用
int &tt=b;      //不能将 tt 修改为 b 的引用
```

如果把 int &tt=b;改为 tt=b;，则该语句是正确的，并不能将 tt 修改为 b 的引用，但 tt 和 a 的值都可以变成 2。

（3）一个引用也可以有引用，也就是说一个变量可以有两个引用。

```
int a=3;
int &b=a;
int &c=b;
```

而这是合法的，这样，变量 a 就有了两个别名，b 和 c。

（4）引用不能为空。

引用必须是某个变量的别名，不能为空。例如：

```
int a=1;
int &tt=a;          //正确，引用 tt 是 int 的型变量 a 的别名
int &t=NULL;        //错误，引用 r 不能为空
```

1.3.2 引用的使用

通过下面的例子来了解引用的使用。

例 1-8 引用变量的简单使用。

```
#include <iostream>
using namespace std;
int main()
{   int x,y=36 ;
    int & refx = x , & refy = y ;
    refx=12;
    cout<<"x="<<x<<"  refx="<<refx<< endl ;
    cout<<"y="<<y<<"  refy="<<refy<< endl ;
    refx=y;
    cout<<"x="<<x<<"  refx="<<refx<< endl ;
    cout<<"&refx="<<&refx<<"  :  "<<"&x="<<&x<<endl ;
```