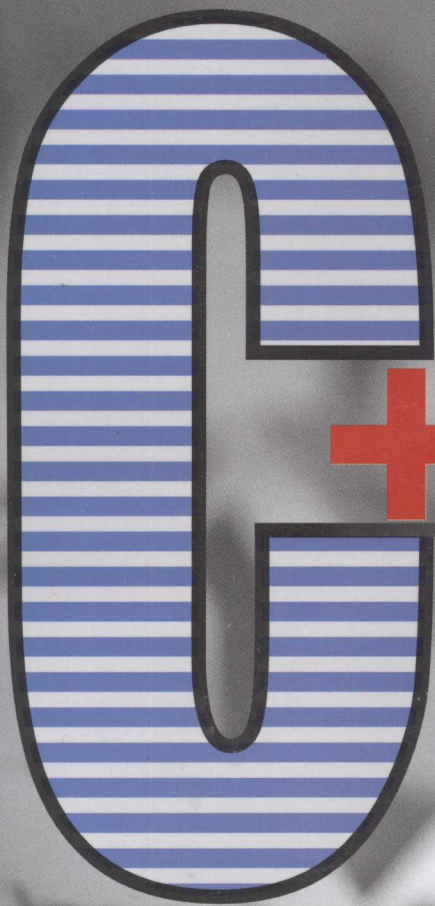




++

程序设计教程

皮德常 张凤林 编著



国防工业出版社

National Defense Industry Press

C++ 程序设计教程

皮德常 张凤林 编著

国防工业出版社

·北京·

++C 程序设计教程

图书在版编目(CIP)数据

C++ 程序设计教程 / 皮德常, 张凤林编著. —北京:
国防工业出版社, 2008.3 重印
ISBN 978-7-118-03583-4

I. C... II. ①皮... ②张... III. C 语言—程序设计—高等学校—教材 IV. TP312

中国版本图书馆 CIP 数据核字(2004)第 092268 号

※

国防工业出版社出版发行

(北京市海淀区紫竹院南路 23 号 邮政编码 100044)

新艺印刷厂印刷

新华书店经售

*

开本 787×1092 1/16 印张 18¼ 字数 418 千字
2008 年 3 月第 2 次印刷 印数 4001—6000 册 定价 29.00 元

(本书如有印装错误, 我社负责调换)

国防书店: (010) 68428422
发行传真: (010) 68411535

发行邮购: (010) 68414474
发行业务: (010) 68472764

前言

C++是一种实用的程序设计语言，是高校学生学习程序设计的一门必修专业课程，同时也是编程人员最广泛使用的工具。学好C++，可以很容易地触类旁通其他语言，如Java和C#等。

本书是在作者讲义的基础上，总结教学和实践经验编写而成的。全书以面向对象的程序设计贯穿始终，针对初学者和自学者的特点，结合实例讲解了C++的基本概念和方法，力求将复杂的概念用简洁、通俗的语言描述，做到深入浅出、循序渐进。因此，适合用于大学计算机专业和非计算机专业的教学，也可供具有C语言基础的自学读者使用。

本书特点

1. 本书主要讲解了面向对象的程序设计理论和编程方法，它是计算机科学与技术专业学生的编程基础。

2. 在内容安排上，提前讲解I/O操作（许多书都是最后讲解）。因为文件是很实用，也是比较难学的一章，这为学生进行课程设计和实验做了铺垫。

3. 本书是作者教学经验的结晶。作者近年来一直从事程序设计方面的教学和科研工作，主讲过程序设计方面的多门课程，为此积累了丰富的教学经验。“从实践到理论，再从理论到实践，循序而渐进”是作者教学的心得体会，编写教材也不例外，作者深知学生的薄弱环节和学习特点。将自己的知识、授课方法和经验整理成书，是作者的梦想。

4. 内容与与时俱进，讲解了C++的许多新内容，这是其他教材所不具备的。例如：向量、string类和STL，许多教材都讲解不全，甚至不讲解这些新内容。作者认为，随着C++的发展，教材也应当与之同步。

5. 作业安排从易到难，环环相扣。许多学生学过C++，却不会编程。作者在教学中认识到了这一点，因此，设计了许多与实际有关的习题，并且它们彼此相关。

6. 分布实施课程设计。C++课程往往都有课程设计，许多院校是将课程设计放在课程结束以后，这就具有一个弊病：时间跨度太长，学生容易忘记学过的知识。不会做的学生，就抄袭别人的。为解决这个问题，本书直接在每章末尾给出课程设计的一部分，以使学过的知识能及时得到巩固。

7. 力求通俗易懂。编写本书的目的是让读者通过自学或在教师的讲授下，能够运用C++语言的核心要素，进行面向对象的程序设计。因此，本书围绕着如何进行C++编程展开。为了便于读者的学习，作者力求该书的语言通俗易懂，将复杂的概念用浅显的语言讲述，便于读者理解和掌握。

8. 在编排上, 本书具有以下特点:

- 每章开始均引出本章要讲解的内容和学习要求。
- 每章安排的习题都具有很强的操作性, 能通过计算机验证。
- 对书中重要的内容采用黑体标记, 特别重要的内容采用下面加点标记。
- 本书强调程序的可读性。书中的程序全部采用统一的程序设计风格。例如: 类名、方法名和变量名的定义做到“望名知义”; 语句的末尾或下一句的开头放上左大括号, 而右大括号自成一行, 并采用缩排格式组织程序代码; 此外, 对程序中的语句还进行了尽可能多的注释。希望读者模仿这种程序设计风格。
- 本书强调程序的可移植性, 不以某个 C++ 开发工具为标准。
- 本书包含了大量的程序示例, 并给出了运行结果。凡是程序开头带有程序名编号的程序, 都是完整的程序, 可以直接在计算机上编译运行。
- 本书采用了醒目的标记来显示知识点, 这些标记是: 警告、注意和思考等, 它们穿插在内容中, 帮助读者尽快找到重要的信息。



警告: 这是警告信息, 它们往往是容易混淆的知识点。



注意: 值得你关注的地方, 其级别略次于警告。



思考: 提出问题, 引导你思考, 以培养你思考的能力。

教学支持

本书的电子教案采用 PowerPoint2000 制作, 可以在讲课时用多媒体投影演示, 这可部分取代板书。教师不仅可以使本教案, 还可以方便地修改和重新组织其中的内容以适应自己的教学需要。使用本教案可以减少教师备课时编写教案的工作量, 以及因板书所耗费的时间和精力, 从而提高单位课时内的知识含量。

我们向使用本教材的教师免费提供本书的电子教案。需要本教案的教师可以直接与国防工业出版社联系。

本书的各章与其配套教材《C++习题解答与课程设计指导》一书中实验内容相配合, 相辅相成。

感谢读者选择本书, 欢迎您对本书的内容提出批评和修改建议, 作者将不胜感激。作者的联系地址如下:

电子邮件地址: dc.pi@nuaa.edu.cn

通信地址: 江苏省南京市南京航空航天大学信息科学与技术学院 顾皮德常 (收)

邮政编码: 210016

内 容 简 介

C++是一种实用的程序设计语言，是高校学生学习程序设计的一门必修专业课程，同时也是编程人员最广泛使用的工具。学好C++，可以很容易地触类旁通其他语言，如Java和C#等。全书以面向对象的程序设计贯穿始终。

全书共7章，主要包括：C++程序设计基础；文件操作；类的基础部分；类的高级部分；继承、多态和虚函数；异常处理以及模板等等。书中列举了数百条可供直接使用的程序示例代码，并给出了运行结果，同时配有大量习题，还提供了该书的电子教案，使学生在学时更为直观。

本书是在作者讲义的基础上，总结过去的教学和实践经验编写而成的。其中结合实例讲解了C++的基本概念和方法，力求将复杂的概念用简洁、通俗的语言描述，做到深入浅出、循序渐进。本书适合用做大学计算机专业和非计算机专业，并且学习过C语言的C++程序设计课程教材，也可供具有C语言基础的自学者使用。

目 录

24	常量表达式	1.4.1
44	常量表达式	1.4.2
84	常量表达式	1.4.3
44	常量表达式	1.4.4
12	常量表达式	1.4.5
22	常量表达式	1.4.6
22	常量表达式	1.4.7
第1章 C++程序设计基础		1
72	1.1 为什么要学习 C++ 程序设计	1
92	1.2 过程化程序设计和面向对象程序设计	2
20	1.3 简单的输出和输入方法	2
20	1.3.1 cout 对象	2
20	1.3.2 cin 对象	4
20	1.4 标识符	7
00	1.5 布尔类型	8
00	1.6 培养良好的编程风格	9
00	1.6.1 风格对比	9
80	1.6.2 注释方法	10
00	1.7 格式化输出	12
00	1.7.1 采用操作符实现格式化输出	13
05	1.7.2 采用函数成员实现格式化输出	19
15	1.7.3 对函数成员的初步讨论	21
15	1.8 格式化输入	21
05	1.8.1 指定输入域宽	21
25	1.8.2 读取一行	22
05	1.8.3 读取一个字符	23
85	1.8.4 读取字符时易于出错的地方	24
08	1.9 函数的缺省参数	25
58	1.10 引用作函数参数	27
58	1.11 函数重载	29
08	1.12 内存的动态分配和释放	33
28	1.13 string 类型	36
08	1.13.1 如何使用 string 类型	36
78	1.13.2 为 string 对象读取一行	38
08	1.13.3 string 对象的比较	38
08	1.13.4 string 对象的初始化	39
10	1.13.5 string 的函数成员	41
40	1.13.6 string 对象应用举例	42
40	1.14 STL 矢量	44

1.14.1	定义矢量的方法	45
1.14.2	访问矢量中的元素	46
1.14.3	使用函数成员 <code>push_back</code>	48
1.14.4	获取矢量的大小	49
1.14.5	删除矢量中元素	51
1.14.6	清空矢量	53
1.14.7	检查矢量是否为空	53
1.14.8	矢量的其他函数成员	55
	思考与练习	57
	课程设计之一	59
第 2 章	文件操作	65
2.1	文件的基本概念	65
2.1.1	文件命名的原则	65
2.1.2	使用文件的基本过程	65
2.1.3	文件流类型	66
2.2	打开文件和关闭文件	66
2.2.1	打开文件	66
2.2.2	文件的打开模式	68
2.2.3	定义流对象时打开文件	69
2.2.4	测试文件打开是否成功	69
2.2.5	关闭文件	70
2.3	采用流操作符读写文件	71
2.3.1	采用流插入操作符写文件	71
2.3.2	格式化输出在写文件中的应用	73
2.3.3	采用流提取操作符从文件读数据	75
2.3.4	检测文件结束	76
2.4	流对象做参数	78
2.5	出错检测	80
2.6	采用函数成员读写文件	82
2.6.1	采用流提取操作符读文件的缺陷	82
2.6.2	采用函数 <code>getline</code> 读文件	83
2.6.3	采用函数 <code>get</code> 读文件	85
2.6.4	采用函数 <code>put</code> 写文件	86
2.7	多文件操作	87
2.8	二进制文件	89
2.8.1	二进制文件的操作	89
2.8.2	读写结构体记录	91
2.9	随机访问文件	94
2.9.1	顺序访问文件的缺陷	94

2.9.2	定位函数 seekp 和 seekg	95
2.9.3	返回位置函数 tellp 和 tellg	99
2.10	输入输出文件	100
	思考与练习	105
	课程设计之二	107
第 3 章	类的基础部分	108
3.1	过程化程序设计与面向对象程序设计的区别	108
3.1.1	过程化程序设计的缺陷	108
3.1.2	面向对象程序设计的基本思想	109
3.2	类的基本概念	110
3.3	定义函数成员	113
3.4	定义对象	114
3.4.1	访问对象的成员	115
3.4.2	指向对象的指针	115
3.4.3	引入私有成员的原因	117
3.5	类的多文件组织	118
3.6	私有函数成员的作用	121
3.7	内联函数	122
3.8	构造函数和析构函数	124
3.8.1	构造函数	124
3.8.2	析构函数	127
3.8.3	带参构造函数	128
3.8.4	构造函数应用举例——输入有效的对象	131
3.8.5	重载构造函数	133
3.8.6	缺省构造函数的表现形式	135
3.9	对象数组	136
3.10	类的应用举例	139
3.11	抽象数组类型	145
3.11.1	创建抽象数组类型	145
3.11.2	扩充抽象数组类型	148
	思考与练习	154
	课程设计之三	155
第 4 章	类的高级部分	156
4.1	静态成员	156
4.1.1	静态数据成员	157
4.1.2	静态函数成员	160
4.2	友元函数	162
4.3	对象赋值问题	167
4.4	拷贝构造函数	169

20	4.4.1 缺省的拷贝构造函数	172
90	4.4.2 调用拷贝构造函数的情况	172
001	4.4.3 拷贝构造函数中的常参数	174
201	4.5 运算符重载	174
701	4.5.1 重载赋值运算符	175
801	4.5.2 this 指针	177
801	4.5.3 重载运算符时要注意的问题	180
801	4.5.4 重载双目算术运算符	181
901	4.5.5 重载单目算术运算符	183
011	4.5.6 重载关系运算符	185
811	4.5.7 重载流操作符<< 和 >>	185
411	4.5.8 重载类型转换运算符	187
211	4.5.9 重载[]操作符	194
211	4.5.10 操作符重载综合举例——自定义 string 类	200
711	4.6 对象组合	211
811	思考与练习	213
151	课程设计之四	214
5	第 5 章 继承、多态和虚函数	215
451	5.1 继承	215
451	5.2 保护成员和类的访问	221
751	5.3 构造函数和析构函数	225
851	5.3.1 缺省构造函数和析构函数的调用	225
161	5.3.2 向基类的构造函数传参数	226
881	5.4 覆盖基类的函数成员	229
281	5.5 虚函数	233
881	5.6 纯虚函数和抽象类	237
981	5.6.1 纯虚函数	237
241	5.6.2 抽象类	237
241	5.6.3 指向基类的指针	241
841	5.7 多重继承	242
421	5.8 多继承	244
221	思考与练习	248
821	课程设计之五	250
6	第 6 章 异常处理	252
721	6.1 异常	252
001	6.1.1 抛出异常	252
501	6.1.2 处理异常	253
701	6.2 基于对象的异常处理	255
901	6.3 捕捉多种类型的异常	257

6.4 通过异常对象获取异常信息	259
6.5 再次抛出异常	261
思考与练习	262
课程设计之六	262
第7章 模板	264
7.1 函数模板	264
7.1.1 从函数重载到函数模板	264
7.1.2 在函数模板中使用操作符需要注意的地方	268
7.1.3 在函数模板中使用多种类型	268
7.1.4 重载函数模板	268
7.1.5 定义函数模板的方法	270
7.2 类模板	270
7.2.1 定义类模板的方法	270
7.2.2 定义类模板的对象	273
7.2.3 类模板与继承	275
思考与练习	278
参考文献	279

第 1 章 C++程序设计基础

C++是在 C 语言的基础上扩充而成的，以其独特的机制在计算机领域有着广泛的应用。本章主要讲述 C++的基本知识，它是属于对 C 语言的扩充，因此有着承前启后的作用。

1.1 为什么要学习 C++程序设计

随着计算机软硬件技术的发展，计算机应用规模不断提高，在软件开发语言和工具方面不断地推陈出新，新语言、新工具层出不穷。目前，国内许多高校，无论是计算机专业或者是非计算机专业，都在开设 C 语言的基础上，陆续开设了 C++语言，并且将它作为学过 C 语言后的一门专业必修课程。

为了解决程序设计的复杂性，贝尔实验室于 1980 开始研制一种“带类”的 C，到 1983 年才正式命名为 C++。在计算机刚发明时，人们采用打孔机直接进行机器指令程序设计，当程序有几百条指令时，采用这种方法就困难了。后来人们设计了用符号表示机器指令的汇编语言，从而能够处理更大、更复杂的程序。20 世纪 60 年代出现了结构化程序设计方法（目前的 C 语言就采用这种方法），这使得人们能够容易编写较为复杂的程序。但是，一旦程序设计达到一定的程度，即使结构化程序设计方法也变得无法控制，其复杂性超出了人的管理限度。例如，一旦 C 程序代码达到了 25000 行~100000 行，系统就变得十分复杂，程序员很难控制，而发明 C++的目的就是为了解决这个问题，其本质就是让程序员理解和管理更大、更复杂的程序。因此，采用支持面向对象的 C++是时代发展的需要。

C++吸收了 C 和 Simula67（一个古老的计算机语言）的精髓，它具有 C 所无法比拟的优越性。C++在维持 C 原来特长（如效率高和程序灵活）的基础上，借鉴了 Simula67 面向对象的思想，将这两种程序设计语言的优点相结合。C++的程序结构清晰、易于扩展、易于维护同时又不失效率。目前，C++已超出了当初设计其的目的，成功地应用到数据库、数据通信等系统，并成功地构造了许多高性能的系统软件。C++与 C 相比，具有三个重要的特征，从而使其优越于 C：

- 第一个特征是支持抽象数据类型 ADT（Abstract Data Type，简称 ADT），在 C++中 ADT 表现为类，是对对象的抽象，而对象是数据和操作该数据代码的封装体，它提供了对代码和数据的有效保护，可防止程序其他不相关的部分偶然或错误地使用对象的私有部分，这是 C 所无法实现的。

- 第二个特征是多态性，即一个接口，多重算法。C++既支持早期联编又支持滞后联编，而 C 仅支持前者。

• 最后一个特征是继承性。继承性一方面保证了代码复用，确保了软件的质量，另一方面也支持分类的概念，从而使对象成为一般情况下的具体实例。

以上三个特性，我们将在后面的章节给予详细的讲解。

目前许多系统软件，如操作系统、数据库管理系统 DBMS 等都采用 C++ 所写，所以从事有关软件开发和计算机应用的人员，不掌握 C++ 简直寸步难行。因此，掌握 C++ 编程已成为许多专业学生的必然选择。

1.2 过程化程序设计和面向对象程序设计

C++ 支持过程化程序设计和面向对象的程序设计，它们是两种不同的编程模式。

在过程化程序设计中，程序员编写函数（也有些书称之为过程）。这些函数是执行某个特定任务的程序语句的集合，每个函数一般都包含局部变量，甚至还有全局变量。过程化程序设计是以过程（函数）为核心，而面向对象程序设计（Object Oriented Programming，简称 OOP）是以对象为核心。一个对象是一个包含数据和对数据操作的封装体。对象包含信息和对信息操作的能力，并且对信息的操作基于消息传递。

随着我们对 C++ 学习的逐步深入，将会深刻理解过程化程序设计和面向对象程序设计。

1.3 简单的输出和输入方法

C++ 除了具有 C 语言的输出和输入方法之外，还具有自己的输出和输入方法。简单的输出是通过 cout 对象，输入是通过 cin 对象，下面分别讲述它们的使用方法。

1.3.1 cout 对象

cout 对象只能用来输出数据，同时也称为标准输出对象，它的作用是使用标准输出设备（即显示器）输出信息。



注意：cout 可以看作是 console output 英文单词的缩写。

cout 是输出流中的一个对象，因此也称为流对象。它的操作对象是数据流，要输出信息，只需将数据流传给 cout，例如：

在上述语句中，“<<”是流插入操作符，它的功能是将字符串“I like programming language C++”送给 cout。程序 1-1 给出了输出数据的另外一种方法。

程序 1-1

```
#include <iostream.h>

void main()
{
    cout << "I like programming language " << "C++";
}
```

程序运行结果:

I like programming language C++

 注意: 程序的开头必须包含 `iostream.h` 文件, 因为 `cout` 对象 (和下一节的 `cin` 对象) 就定义在该文件中, 这就像 C 程序必须包含 `stdio.h` 文件一样。

通过上例可以看出, 使用 “<<” 可以传送多个数据给 `cout`。例如, 将程序 1-1 的 `cout` 语句修改如下:

```
cout << "I like programming language ";
cout << "C++";
```

程序段的运行结果和程序 1-1 相同。但我们要理解一个重要的概念: 虽然输出分解为两个语句, 但程序仍在同一行上显示信息。除非你指定输出方式, 否则送给 `cout` 的信息将会连续显示, 程序 1-2 说明了这一点。

程序 1-2

```
#include <iostream.h>

void main ()
{
    cout << "我最喜爱的东西: ";
    cout << "computer ";
    cout << "& tea ";
}
```

程序运行结果:

我最喜爱的东西: computer & tea

输出结果与源代码中字符串的安排是不同的, `cout` 完全按照提交数据的方式输出。在上面的源代码中使用了 3 个输出语句 `cout`, 但它仍在一行上输出信息, 这是因为如果不加入换行符, `cout` 不会自动换行。我们有两种换行的方法: 一种方法是在 `cout` 语句后加一个流操作符 **endl**; 另一种方法是加入 “\n” 换行符。程序 1-3 就是这样定义的。

程序 1-3

```
#include <iostream.h>

void main ()
{
    cout << "我最喜爱的东西: " << endl;
    cout << "computer " << "\n";
    cout << "& tea \n";
}
```

程序运行结果:

我最喜爱的东西:

computer

& tea

 注意: endl 是 end of line 的缩写, 它和 '\n' 的功能一样都是换行。当 cout 遇到 '\n' 时, 将输出光标移到下一行的开头。



警告: 不要把反斜线 \ 和正斜线 / 弄混, '\n' 是不会换行的; 同时也不要不要在反斜线和字符 n 之间加空格, 例如 '\ n' 就是错误的。

1.3.2 cin 对象

cout 是 C++ 的标准输出对象, cin 是 C++ 的标准输入对象, 它的功能是从 I/O 控制台 (即键盘) 接收输入数据, 如程序 1-4。

程序 1-4

```
#include <iostream.h>
```

```
void main ()
```

```
{
```

```
    int length, width, area;
```

```
    cout << "计算矩形的面积 \n";
```

```
    cout << "输入矩形的长: ";
```

```
    cin >> length;
```

```
    cout << "输入矩形的宽: ";
```

```
    cin >> width;
```

```
    area = length * width;
```

```
    cout << "矩形的面积为: " << area << "\n";
```

程序运行结果:

计算矩形的面积

输入矩形的长: 10 [Enter]

输入矩形的宽: 20 [Enter]

矩形的面积为: 200

这个程序可以用来计算一个矩形的面积。当程序运行时, 用户输入的数据将存储在 length 和 width 两个变量中, 如下两行:

```
cout << "输入矩形的长: ";
```

```
cin >> length;
```

cout 在屏幕上显示“输入矩形的长:”，cin 为 length 变量输入值。其中“>>”称为流提取操作符，它从左边输入流对象 cin 中读一个数，并把它存储在>>右边的变量中。在上面的程序语句中，cin 将从键盘输入的数据存储在 length 变量中。



注意：插入操作符“>>”和提取操作符“<<”指定了数据的流动方向。插入操作符“>>”将输入的数据传给变量，而提取操作符“<<”将变量（或常量）传给 cout 输出。cin 对象在读取数据时，将暂停程序的运行，直到从键盘上输入数据并按 Enter 键确认。cin 对象能自动地将输入数据转换成与变量一致的数据类型，例如，用户输入“10”，cin 将分别读入字符'1'和'0'，在将该数据存储在变量之前，cin 能够自动地将它们转换成整数 10。cin 同样能够识别出，像“10.7”这样的数不能储存在整型变量中。如果用户输入一个浮点数给整型变量，那么小数点后的位数将被舍弃（也称截断）。如果用户输入浮点数，cin 通过截断浮点数后的小数部分，将整数部分存储在整型变量中。

如果程序要求输入数据，那么就应该向用户提示该输入什么样的数据。程序 1-5 是在程序 1-4 的基础上修改而成的，但缺乏提示用户的信息，因此它不是一个好程序。

程序 1-5

```
#include <iostream.h>

void main()
{
    int length, width, area;

    cin >> length;
    cin >> width;
    area = length * width;
    cout << "矩形的面积为: " << area << "\n";
}
```

当运行这个程序时，用户面对的是黑屏幕，不知道要做什么事。一个功能完善的程序应当及时、友好地给用户必要的提示信息。

采用 cin 对象可以一次读入多个变量的值，例如程序 1-6 采用 cin 同时给变量 length 和 width 变量读取值。

程序 1-6

```
#include <iostream.h>

void main()
{
    int length, width, area;

    cout << "计算矩形的面积 \n";
    cout << "输入矩形的长和宽，中间用空格隔开: ";
    cin >> length >> width; // 给两个变量读取值
```

```

    area = length * width;
    cout << "矩形的面积为: " << area << "\n";
}

```

程序运行结果:

计算矩形的面积

输入矩形的长和宽, 中间用空格隔开: 10 20 [Enter]

矩形的面积为: 200

下列语句等待用户输入两个数值, 并把输入中的第一个数赋给变量 `length`, 第二个数赋给变量 `width`:

```
cin >> length >> width;
```

在上例输出中, 用户输入 10 和 20, 10 就送给了变量 `length`, 20 送给了变量 `width`。当输入多个数值时, 数值之间要加空格。`cin` 读到空格时, 就能够区别输入中的各个数值, 数值间的空格数无所谓, 例如用户按照如下形式输入也可以:

```
10          20 [Enter]
```

但要注意的是, 在最后一个数输入后要按 Enter 键。



注意: `cin` 读取数据的特点和 `scanf` 函数类似, 在上例中的输入中, 如果先输入 10 并按 Enter 键, 然后输入 20 再按 Enter 键, 完全可以正确地输入数据。本书对数据的输入和输出格式不做过详细的探讨, 因为这些不是 C++ 的核心和重点。

采用一个 `cin`, 也可以同时为多个不同类型的变量读入数据, 程序 1-7 给出了示例。

程序 1-7

```
#include <iostream.h>
```

```
void main ()
```

```
{
```

```
    int whole;
```

```
    float fractional;
```

```
    char letter;
```

```
    cout << "请输入一个整数、一个浮点数和一个字符: ";
```

```
    cin >> whole >> fractional >> letter;
```

```
    cout << "整数: " << whole << endl;
```

```
    cout << "浮点数: " << fractional << endl;
```

```
    cout << "字符: " << letter << endl;
```

```
}
```

程序运行结果:

请输入一个整数、一个浮点数和一个字符: 100 3.14159 Y [Enter]