

皮德常 编著

**面向对象**

**C++**

**程序设计**

清华大学出版社



皮德常 编著

**面向对象**

**C++**

**程序设计**

清华大学出版社  
北京

## 内 容 简 介

本书详细介绍了 C++ 面向对象的核心编程思想和方法, 特别注重程序设计的实用性, 使读者具备运用面向对象的方法分析和解决实际问题的能力。

本书以面向对象的程序设计贯穿始终, 共 9 章, 主要包括: C++ 程序设计基础、文件操作、类的基础、继承、多态、虚函数、对象组合、异常处理、标准模板库 STL(主要介绍编程常用的 string 类、容器类、迭代器及其算法等) 以及通过 ODBC 对数据库进行编程等, 为后继课程的学习和课程设计打下坚实的基础。书中列举了数百个可供直接使用的程序示例代码, 并给出了运行结果。

本书语言流畅、实例丰富, 讲解了 C++ 程序设计的核心内容。全部代码都在 Visual Studio C++ 2010 环境下调试通过, 并配有大量的习题, 同时在网站提供了该书的电子教案和程序示例源码, 特别适合作为高等学校 C++ 编程和面向对象程序设计课程的教材。

本书封面贴有清华大学出版社防伪标签, 无标签者不得销售。

版权所有, 侵权必究。侵权举报电话: 010-62782989 13701121933

### 图书在版编目(CIP)数据

面向对象 C++ 程序设计/皮德常编著. —北京: 清华大学出版社, 2017  
ISBN 978-7-302-45892-0

I. ①面… II. ①皮… III. ①C 语言—程序设计 IV. ①TP312.8

中国版本图书馆 CIP 数据核字(2016)第 298822 号

责任编辑: 张瑞庆  
封面设计: 常雪影  
责任校对: 焦丽丽  
责任印制: 李红英

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](mailto:c-service@tup.tsinghua.edu.cn)

质量反馈: 010-62772015, [zhiliang@tup.tsinghua.edu.cn](mailto:zhiliang@tup.tsinghua.edu.cn)

课件下载: <http://www.tup.com.cn>, 010-62795954

印 装 者: 清华大学印刷厂

经 销: 全国新华书店

开 本: 185mm×260mm

印 张: 18.75

字 数: 457 千字

版 次: 2017 年 2 月第 1 版

印 次: 2017 年 2 月第 1 次印刷

印 数: 1~2000

定 价: 39.00 元

产品编号: 069435-01

# 前言 Foreword

C++ 是一种广泛使用的计算机编程语言,常用于系统开发、引擎开发等应用领域,是至今为止最受广大程序员喜爱的、最强大的编程语言之一,它支持封装、继承和重载等面向对象的重要特性。同时,C++ 也是高校学生学习程序设计的一门专业必修课程,学好 C++ 语言,可以很容易地触类旁通其他语言,如 Java、C# 等。

本书是作者总结近 20 年的教学和实践经验编著而成的,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。适合用作为高校 C++ 程序设计和面向对象程序设计课程的教材,也可供具有 C 语言编程基础的自学者使用。

## 本书特点

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

(2) 本书作者近 20 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,如 C、C++ 和 Java,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,具有针对性。

(3) 该书内容与与时俱进,讲解了 C++ 的许多新内容。例如,string 类、体现了泛型程序设计思想的 STL,以及基于 STL 的基本程序设计方法、通过 ODBC 对常规数据库的编程方法等。作者认为,随着 C++ 的发展,教材也应当与之同步。本书另辟新章专门讲解了这些内容,并结合实例给出了具体应用和综合举例。为读者采用 C++ 进行课程设计和项目研发打下坚实的基础。

(4) 作业安排从易到难,环环相扣。许多学生学过 C++,却不会编程。因此,本书设计了许多与实际有关的习题,并且它们彼此相关。

(5) 课程设计。C++ 课程往往都有课程设计,为便于教师组织教学和学生理解课程设计要求,本书的最后给出了课程设计的基本要求和文档模板,为课程设计的顺利进行提供了便利。

(6) 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,将复杂的概念采用浅显的语言讲述,便于读者理解和掌握。

## 本书编排特点

(1) 每章开始均引出本章要讲解的内容和学习要求。

(2) 每章安排的习题都具有很强的操作性,能通过计算机编程验证。

(3) 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。

(4) 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,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。

(5) 本书包含了大量的程序示例,全部采用 Microsoft Visual C++ 2010(Express) 版本给出了运行结果。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。

(6) 本书采用醒目的标记来显示知识点。这些注意和思考的标记,都穿插在内容中,帮助读者尽快找到重要的信息。

**【注意】**值得读者关注的地方,往往是容易混淆的知识点。

**【程序运行结果】**给出当前示例的运行结果。

**【程序解析】**对示例程序中的难点给予分析,帮助读者理解程序。

**【思考】**在对示例程序理解的基础上,进一步提出问题,引导读者思考,以培养其思考能力。

## 教学支持

我们向使用本教材的教师免费提供本书的用 PowerPoint 2003 制作的电子课件和全部程序示例源代码,需要的教师可以在 <http://www.tup.tsinghua.edu.cn> 网站上获取。

感谢读者选择本书,欢迎提出批评和修改建议,作者联系电子邮件 [dc.pi@163.com](mailto:dc.pi@163.com)。

作者

2017年1月

<b>第 1 章 C++ 程序设计基础 .....</b>	<b>1</b>
1.1 为什么要学习 C++ 程序设计 .....	1
1.2 过程化程序设计和面向对象程序设计 .....	2
1.3 简单的输出和输入方法 .....	2
1.3.1 cout 对象 .....	2
1.3.2 cin 对象 .....	4
1.4 标识符 .....	7
1.5 布尔类型 .....	8
1.6 培养良好的编程风格 .....	8
1.6.1 风格对比 .....	9
1.6.2 注释方法 .....	9
1.7 格式化输出 .....	11
1.7.1 采用操作符实现格式化输出 .....	12
1.7.2 采用函数成员实现格式化输出 .....	17
1.7.3 对函数成员的初步讨论 .....	19
1.8 格式化输入 .....	19
1.8.1 指定输入域宽 .....	19
1.8.2 读取一行 .....	20
1.8.3 读取一个字符 .....	21
1.8.4 读取字符时容易出错的地方 .....	22
1.9 函数的默认参数 .....	23
1.10 引用作函数参数 .....	25
1.11 函数重载 .....	27
1.12 内存的动态分配和释放 .....	30
思考与练习 .....	33
<b>第 2 章 文件操作 .....</b>	<b>36</b>
2.1 文件的基本概念 .....	36
2.1.1 文件命名的原则 .....	36
2.1.2 使用文件的基本过程 .....	36
2.1.3 文件流类型 .....	37



- 2.2 打开文件和关闭文件..... 37
  - 2.2.1 打开文件 ..... 38
  - 2.2.2 文件的打开模式 ..... 39
  - 2.2.3 定义流对象时打开文件 ..... 40
  - 2.2.4 测试文件打开是否成功 ..... 40
  - 2.2.5 关闭文件 ..... 41
- 2.3 采用流操作符读写文件..... 41
  - 2.3.1 采用<<写文件 ..... 41
  - 2.3.2 格式化输出在写文件中的应用 ..... 43
  - 2.3.3 采用>>从文件读数据 ..... 45
  - 2.3.4 检测文件结束 ..... 46
- 2.4 流对象作为参数..... 47
- 2.5 出错检测..... 49
- 2.6 采用函数成员读写文件..... 51
  - 2.6.1 采用>>读文件的缺陷 ..... 51
  - 2.6.2 采用函数 getline 读文件 ..... 52
  - 2.6.3 采用函数 get 读文件 ..... 53
  - 2.6.4 采用函数 put 写文件 ..... 54
- 2.7 多文件操作..... 55
- 2.8 二进制文件..... 57
  - 2.8.1 二进制文件的操作 ..... 57
  - 2.8.2 读写结构体记录 ..... 58
- 2.9 随机访问文件..... 62
  - 2.9.1 顺序访问文件的缺陷 ..... 62
  - 2.9.2 定位函数 seekp 和 seekg ..... 62
  - 2.9.3 返回位置函数 tellp 和 tellg ..... 65
- 2.10 输入输出文件 ..... 67
- 思考与练习 ..... 71

**第3章 类的基础部分..... 73**

- 3.1 过程化程序设计与面向对象程序设计的区别..... 73
  - 3.1.1 过程化程序设计的缺陷 ..... 74
  - 3.1.2 面向对象程序设计的基本思想 ..... 74
- 3.2 类的基本概念..... 75
- 3.3 定义函数成员..... 78
- 3.4 定义对象..... 79
  - 3.4.1 访问对象的成员 ..... 79
  - 3.4.2 指向对象的指针 ..... 79
  - 3.4.3 引入私有成员的原因 ..... 81

3.5	类的多文件组织	82
3.6	私有函数成员的作用	84
3.7	内联函数	85
3.8	构造函数和析构函数	87
3.8.1	构造函数	87
3.8.2	析构函数	89
3.8.3	带参构造函数	91
3.8.4	构造函数应用举例——输入有效的对象	93
3.8.5	重载构造函数	95
3.8.6	缺省构造函数的表现形式	97
3.9	对象数组	98
3.10	类的应用举例	101
3.11	抽象数组类型	106
3.11.1	创建抽象数组类型	106
3.11.2	扩充抽象数组类型	109
	思考与练习	114

## 第4章 类的高级部分 115

4.1	静态成员	115
4.1.1	静态数据成员	116
4.1.2	静态函数成员	118
4.2	友元函数	121
4.3	对象赋值问题	125
4.4	拷贝构造函数	127
4.4.1	默认的拷贝构造函数	129
4.4.2	调用拷贝构造函数的情况	129
4.4.3	拷贝构造函数中的常参数	131
4.5	运算符重载	131
4.5.1	重载赋值运算符	132
4.5.2	this 指针	134
4.5.3	重载运算符时要注意的问题	137
4.5.4	重载双目算术运算符	138
4.5.5	重载单目算术运算符	140
4.5.6	重载关系运算符	141
4.5.7	重载流操作符<<和>>	142
4.5.8	重载类型转换运算符	144
4.5.9	重载[]操作符	149
4.5.10	操作符重载综合举例——自定义 string 类	154
4.6	对象组合	163



思考与练习	165
<b>第 5 章 继承、多态和虚函数</b>	<b>166</b>
5.1 继承	166
5.2 保护成员和类的访问	171
5.3 构造函数和析构函数	174
5.3.1 缺省构造函数和析构函数的调用	175
5.3.2 向基类的构造函数传参数	176
5.4 覆盖基类的函数成员	179
5.5 虚函数	182
5.6 纯虚函数和抽象类	185
5.6.1 纯虚函数	185
5.6.2 抽象类	186
5.6.3 指向基类的指针	189
5.7 多重继承	190
5.8 多继承	192
思考与练习	195
<b>第 6 章 异常处理</b>	<b>198</b>
6.1 异常	198
6.1.1 抛出异常	199
6.1.2 处理异常	199
6.2 基于对象的异常处理	201
6.3 捕捉多种类型的异常	203
6.4 通过异常对象获取异常信息	205
6.5 再次抛出异常	207
思考与练习	208
<b>第 7 章 模板</b>	<b>209</b>
7.1 函数模板	209
7.1.1 从函数重载到函数模板	209
7.1.2 在函数模板中使用操作符需要注意的地方	212
7.1.3 在函数模板中使用多种类型	213
7.1.4 重载函数模板	213
7.1.5 定义函数模板的方法	214
7.2 类模板	215
7.2.1 定义类模板的方法	215
7.2.2 定义类模板的对象	217
7.2.3 类模板与继承	219

思考与练习	222
<b>第 8 章 标准模板库 STL</b>	<b>223</b>
8.1 标准模板库简介	223
8.2 string 类型	226
8.2.1 如何使用 string 类型	226
8.2.2 为 string 对象读取一行	226
8.2.3 string 对象的比较	227
8.2.4 string 对象的初始化	227
8.2.5 string 的函数成员	228
8.2.6 string 对象应用举例	230
8.3 迭代器类	231
8.4 顺序容器	233
8.4.1 矢量类	234
8.4.2 列表类	239
8.4.3 双端队列类	242
8.5 函数对象与泛型算法	244
8.5.1 函数对象	245
8.5.2 泛型算法	248
8.6 关联容器	251
8.6.1 集合和多重集合类	251
8.6.2 映射和多重映射类	253
8.7 容器适配器	255
8.7.1 栈容器适配器	255
8.7.2 队列容器适配器	256
8.7.3 优先级队列容器适配器	257
思考与练习	258
<b>第 9 章 数据库程序设计</b>	<b>259</b>
9.1 数据库简介	259
9.2 SQL 语句	260
9.2.1 定义表	260
9.2.2 查询	260
9.2.3 插入	261
9.2.4 删除	261
9.2.5 修改	261
9.3 数据库连接	262
9.3.1 ODBC 简介	262
9.3.2 ODBC 驱动程序	262

9.3.3	创建数据源	262
9.4	数据库编程中的基本操作	264
9.4.1	数据库编程的基本过程	264
9.4.2	数据库查询	265
9.4.3	插入记录	266
9.4.4	修改记录	267
9.4.5	删除记录	268
9.5	数据库编程综合举例	269
	思考与练习	276
<b>附录 A</b>	<b>课程设计要求</b>	<b>278</b>
A.1	课程设计简介	278
A.2	程序结构	282
A.3	程序的主要特点	283
A.4	操作说明	283
A.4.1	收银模块	283
A.4.2	书库管理模块	284
A.4.3	报表模块	284
A.4.4	退出系统	285
<b>附录 B</b>	<b>课程设计报告格式</b>	<b>286</b>
	<b>参考文献</b>	<b>288</b>

C++ 是在 C 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,它是对 C 的扩充。

本章的学习目标:

- 了解 C++ 的发展历史,明白 C++ 编程的重要性。
- 掌握基本的输入和输出方法。
- 理解函数的默认参数。
- 掌握引用作函数参数。
- 掌握内存的动态分配和释放方法。

## 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++ 语言程序设计课程,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 年开始研制一种“带类”的 C,到 1983 年才正式命名为 C++。在计算机发明初期,人们采用打孔机直接进行机器指令程序设计,当程序长度有几百条指令时,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。到了 20 世纪 60 年代出现了结构化程序设计方法(目前的 C 就采用这种方法),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 25 000~100 000 行,系统就变得十分复杂,程序员很难控制,而研制 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 Simula 67 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 Simula 67 的面向对象的思想,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++ 已超出了当初设计它的目的,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 3 个重要特征,从而使其优越于 C。

第一个特征是支持抽象数据类型(Abstract Data Type, 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。

第三个特征是继承性。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3个特性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++编写,所以从事有关软件开发和计算机应用的人员,若不掌握C++简直寸步难行。总之一句话: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(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, 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。

随着对C++学习的逐步深入,将会逐渐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对象用来输出数据。cout对象是C++的标准输出对象,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:

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

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。

**【例 1-1】** 输出数据的方法。

```
#include <iostream>
using namespace std;
int main()
```

```
{
    cout << " I like programming language " << " C++ " ;
    return 0;
}
```

### 【程序运行结果】

I like programming language C++

**【注意】** 程序的开头必须包含 `iostream` 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,将例 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例 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,但是程序仍在同一行上显示信息。除非指定输出方式,否则送给 `cout` 的信息将会连续显示。例如:

```
cout << "我最喜爱的东西: " ;
cout << "computer " ;
cout << " & tea " ;
```

程序段的运行结果:

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。有两个换行的方法,一个方法是在 `cout` 语句后加一个流操作符 `endl`;另一种方法是加入 `'\n'` 换行符。

### 【例 1-2】 换行输出。

```
#include <iostream>
using namespace std;
int main()
{
    cout << "我最喜爱的东西: " << endl ;
    cout << "computer " << '\n' ;
    cout << " & tea \n" ;
    return 0;
}
```

### 【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。此外,不要把反斜线\和正斜线/弄混,'\n'是不会换行的;同时,也不要再在反斜线和字符 n 之间加空格,例如\ n'是错误的。

### 1.3.2 cin 对象

cin 对象是 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。

**【例 1-3】** 标准输入和输出。

```
#include <iostream>
using namespace std;
int 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" ;
    return 0;
}
```

**【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
```

**【程序解析】** 这个程序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。流提取操作符>>将输入的数据传给变量,而流插入操作符<<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',在将该数据存储到变量之前,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 这样的不能储存在整型变量中的数;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on应该向用户提示该输入什么样的数据。下面的例 1-4 由于缺乏提示用户的信息,不是一个好的程序,在写程序中要杜绝这种现象。

**【例 1-4】** 缺乏输入提示信息的写法,这是一种不好的写法。

```
#include <iostream>
using namespace std;
int main()
{
    int length, width, area;

    cin >>length;
    cin >>width;
    area =length * width;
    cout <<"矩形的面积为: " <<area <<"\n";
    return 0;
}
```

**【程序解析】** 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:

```
int length, width, area;
cout <<"请输入矩形的长和宽,中间用空格隔开: ";
cin >>length >>width;    //给两个变量读取值
```

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,用户也可以按照如下形式输入:

```
10      20
```

但要注意的是,在最后一个数输入后要按 Enter 键。

**【注意】** cin 对象读取数据的特点和 C 中的 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作过详细的探讨,因为这些不是 C++ 的核心和重点。

采用一个 cin 对象,也可以同时为多个不同类型的变量读入数据。

**【例 1-5】** 采用 cin 对象同时为多个不同类型的变量输入数据。



```

#include <iostream>
using namespace std;
int 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;
    return 0;
}

```

### 【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]
整数: 100
浮点数: 3.14159
字符: Y

```

**【程序解析】** 从上例的输出可以看出,各个数值分别储存在各自的变量中。如果用户的输入如下:

```
5.7 4 B
```

那么,程序将把 5 存储在变量 whole 中,将 0.7 存储在变量 fractional 中,把 4 存储在变量 letter 中,所以必须以正确的格式输入各个数值。

cin 读入字符串的方式与 scanf 函数类似,并且也是采用字符数组存储字符串,例如:

```

char company[12];
cin >> company;

```

该数组最多能存储 12 个字符,并且最后一位应是 '\0',表示字符串的结束。

**【注意】** 如果用一个字符数组存储字符串,要确保该字符数组足够大,能够存储字符串中的所有字符(包括空字符 '\0')。

**【例 1-6】** 采用 cin 对象读取一个字符串。

```

#include <iostream>
using namespace std;
int main()
{
    char name [21];

    cout << "What is your name? ";
    cin >> name;
}

```