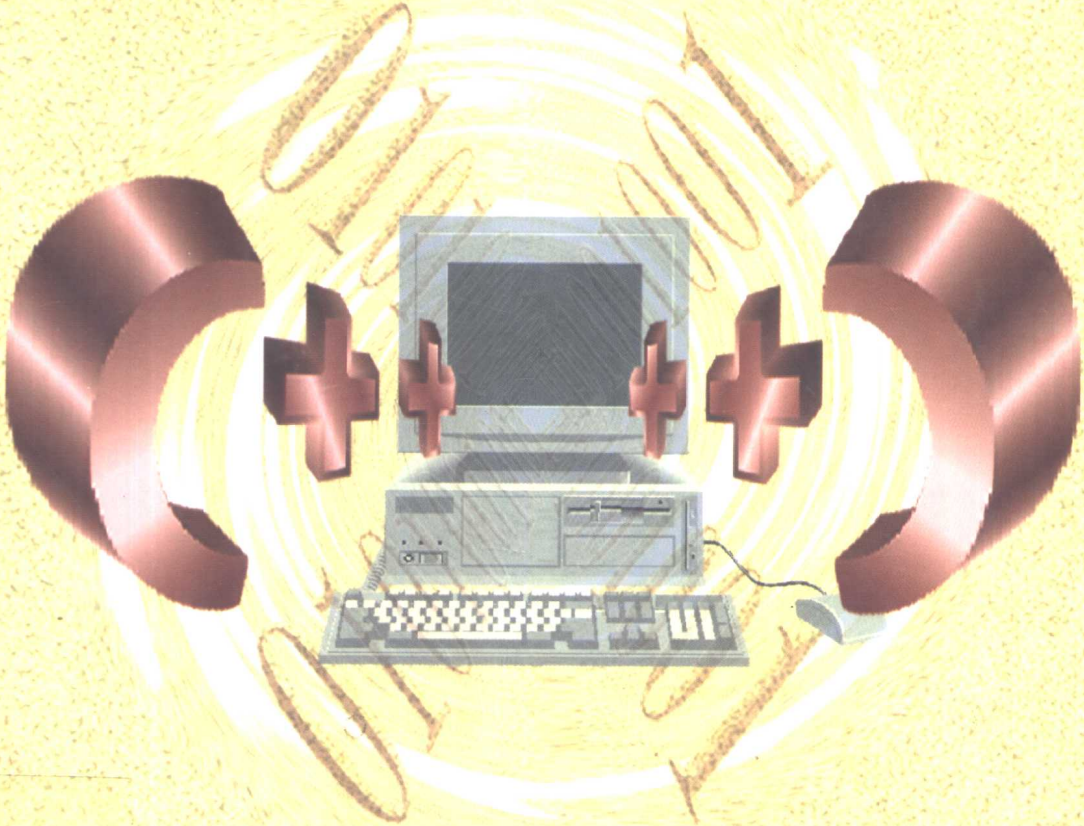
高等学校教学用书

C++面向对象程序设计实用教程

钟蓓 顾建鹏 编著



北京航空航天大学出版社

C++面向对象程序设计实用教程

钟　蓓　顾建鹏　编著

北京航空航天大学出版社

内 容 简 介

面向对象程序设计是近年来最热门的软件开发技术。C++ 编程语言是目前最常用的面向对象程序设计语言。本书由浅入深，使用大量程序实例，融 C++编程语言的基本概念和面向对象程序设计的基本原理为一体，系统地讲述了 C++ 的语言特性和用 C++进行面向对象程序设计的技术。必要的章节附有编程中最常见的错误和有关的解释，帮助读者加深理解和避免类似的错误，或作为编程时查错的参考。本书面向实用，内容新颖、丰富，材料充实，自成体系，通俗易懂，是兼顾深度和广度的 C++ 编程的教科书，适合作高等院校教材，也可作广大计算机工作者的参考书。

图书在版编目(CIP) 数据

C++ 面向对象程序设计实用教程/钟蓓等编著. —北京
：北京航空航天大学出版社，1999.4
ISBN 7-81012-867-1

Ⅰ. C… Ⅱ. 钟… Ⅲ. C 语言-程序设计-教材 Ⅳ. TP312

中国版本图书馆 CIP 数据核字(1999)第 05530 号

C++面向对象程序设计实用教程

钟　蓓　顾建鹏　编著

责任编辑　许传安

责任校对　陈　坤

北京航空航天大学出版社出版发行

北京市学院路 37 号 (100083) 发行部电话：82317024

http://www.buaapress.cn.net

E-mail:pressell@publica.bj.cninfo.net

北京宏文印刷厂印刷　各地书店经销

*　*　*

开本：787×1092　1/16　印张：14　字数：354 千字

1999 年 4 月第 1 版　1999 年 4 月第 1 次印刷　印数：5000册

ISBN 7-81012-867-1/TP・331　　定价：19.00 元

前　言

近年来，无论学术领域还是工业技术领域，面向对象的编程技术有了飞速发展，成为最热门的实用软件开发技术。面向对象的编程技术与传统的方法完全不同，它把现实世界的问题抽象为“类”的概念，而解决问题的过程就是完成对类所生成对象的一系列操作。面向对象编程有四个显著的特点：数据抽象、信息隐藏(封装)、继承和动态连编。这些特点使面向对象编程技术较传统编程方法有明显的优越性，比如：数据抽象在对象的定义和实现之间提供了清晰的分界；继承性允许软件开发者在已存在类的基础上定义新的类。数据抽象是面向对象编程的主要特色之一。用面向对象方法所编制的程序最大优点是程序结构严密、清楚，有条理性；其系统功能可以非常容易地被增加、删除及修改。例如，通过派生类的应用，可以非常方便地赋予原有基类新的使用价值，以扩展系统功能。

C++ 编程语言是目前最常用的面向对象程序设计语言。虽然 C++ 在结构上和规则上是 C 语言的增强与扩展，但从本质上讲，作为一种新兴的计算机语言，C++ 是不同于C语言的。它与 C 语言的最大区别就在于对数据抽象和对面向对象编程技术的支持。因此，没有C语言的基础，对读者直接学习 C++ 和用 C++ 进行面向对象编程并没有多大的影响。

本书由浅入深，使用大量程序实例，融 C++ 编程语言的基本概念和面向对象程序设计的基本原理为一体，系统地介绍了C++ 的语言特性和用 C++ 进行面向对象程序设计的技术，具体说明了如何用 C++ 建立和使用数据抽象、信息隐藏、继承和多态性以及 C++ 是如何支持面向对象编程中的这些主要概念的，帮助读者真正掌握 C++ 语言与面向对象编程概念的对应关系。通过学习该书，读者无需参考别的资料便可在短期内掌握 C++ 编程语言和面向对象程序设计的精髓，得到十分明晰的用 C++ 进行面向对象程序设计的思路。从而可以解决许多人在学习了有些面向对象编程概念的书籍和 C++ 编程语言的书籍以后，仍不知如何使用 C++ 进行实际的面向对象编程的问题。

本书共分九章，分别给出了 C++入门的基本概念、数据类型、复合语句和程序块、派生数据类型、函数、重载函数及函数样板、文件类、类的派生与继承、面向对象编程等内容。必要的章节附有编程中最常见的错误和有关的解释，帮助读者加深理解和避免类似的错误。附录中分别给出了 ASCII 码表、操作符优先级和常用的 C++ 函数，供读者编程时参考。

本书是作者在多年积累的教学经验和实际编程经验的基础上编写的。选材从实用出发，兼顾了深度和广度。另外，书中的程序例子均已在计算机上调试通过。愿本书能帮助读者在短期内掌握 C++ 编程语言和面向对象程序设计的技术。书中难免有不足之处，望用者不吝指正。

编著者

1998 年 11 月

目　录

第1章 C++入门

1.1 C++简史及其特点

C++是一种新兴的编程语言。它是被用于一些最新的，旨在使软件更易于开发、易于扩展并尽可能有效的软件工程方法。它在结构和规则上是C语言的增强与扩展形式，并且又引用了一些相关语言——Simula, Pascal 和 Ada 等面向对象编程语言的特性。

80年代早期，C++的第一个版本在 AT & T Bell 实验室由 Bjarne Stroustrup 开发，并在1985年发行。在一年之中，它立即引起许多公司的极大兴趣并在20多种不同的硬件系统上实现，其范围从家庭微机到大型机。

本章阐明C++所具有的从其它语言继承而来的特点。从本质上讲，作为一种计算机语言，C++不只是“扩展的C”，也不是“C的新版本”。C++语言的定义不仅在小的和大的方面修改了C，同时还兼有和开发了其它语言的软件特点。

1.1.1 计算机语言的Algol家族

C++是Algol语言家族的最新成员之一。所有这些语言是“强制”语言。从根本上来说，“强制”语言是由给计算机一组有序命令来完成一项任务的。众所周知，这些命令叫语句。

C++从其家族语言的先前成员演绎而来，形成了它本身的结构和特点。

Algol

Algol (1960)是第一种能够使程序控制结构化流程有效的语言。Algol提供结构语句诸如：

```
if (条件为真)  do  某事1    else  某事2.
```

应用Algol语言，可以得益于程序控制的进程。在程序的任意地方，可被引出用于完成整个程序任务的子任务的一个过程（其它某处的一段程序编码)。当子程序被完成，程序控制返回到原处。在C++中，这个过程称作“函数”。

Pascal

Pascal 是一种具有很强类型化的语言，就像 Ada 和 Modula-2。用这种语言编制的程序，编译器在程序被翻译之前和在检查数据对象被妥善用于传递正确类型的操作数给过程之前，获知所有数据对象的类型。

Pascal 提供一组提前定义的数据对象类型。编译器获知这些类型和用于某一特定类型对象的操作。这些类型包括整数型(正整数和负整数)，实数型 (数字包括小数诸如 4.37 和 -0.001)，字符型 (字母、数字、标点符号和控制字符)，数组(数据对象列表、依据列表中的位置、每个数据对象可被选择)，指针 (记忆数据对象的地址)和文件(通常是提供数据的文本文件)。

C——影响最大的“双亲”

C 作为一种书写系统软件的语言而被开发。这种软件使程序员可完全和容易地利用计算机硬件。C 语言被首先用于开发 UNIX 操作系统。

近 20 年来，C 语言成为最重要的编程语言之一。它强调高级语言的优点与机器语言的灵活性和有效性相结合。

一个 C 编译器不一定检查数据对象是否被正确管理 (该语言是弱类型化语言)，因此容易产生运行错误，而用这里提及的多数其它语言编写的程序就不太会出现类似的错误。

C 语言是一种相对的小型语言，可以在多数计算机上毫无困难地执行。很多在 PC 机上开发的软件是由 C 语言写成的。C 成为一种有普遍用途的语言。C 语言有两种基本预定义：整数型和浮点型(实数)。这些类型的对象由操作符管理，如算术操作符 +，-，* 和 /，以及关系操作符 >，>=，<，<=，==(等于)和 !=(不等于)。

C++ 语言有与 C 语言相同的类型定义和操作符，以及相同的结构语句形式。

C++ 通常是由提供 ANSI C 标准库来实现的。这个标准库定义了许多处理需频繁执行子任务的非常有用的函数。C++ 还增加了 iostream 库(被用来完成输入和输出)到 C 库。多数 C++ 编译器也可编译 C 程序代码。用 C 编写的程序文件可被连编到一个 C++ 程序。

Simula——影响最大的“祖父母”

高级计算机语言首先应用的领域之一是模拟分离系统，如机场、港口或前面带有汽油泵的汽车修理厂。这些系统在计算机上被模拟，用来确定飞机的跑道、停泊船只的港湾或哪台油泵被需要。

如果我们考虑机场系统，知道这些飞机可能是波音 747 系列或两坐小型飞机。描述一架飞机可用一般术语：它的数据成员，像发动机个数和门的个数，以及它的成员函数诸如起飞、飞行航线、降落等等。

60 年代初，在挪威，模拟语言 Simula 被开发用来描述和模拟这样的系统，其作用是考查系统中数据对象的生命周期(life cycle)。因此，在开始构造系统之前一个最合适的结果就有可能被获得。挪威应用 Simula 寻找系统状态发展的最佳答案已有多年。

Simula 语言是基于 Algol，又增加了“类”(class)概念的语言。应用“类”的概念，就有可能定义一个“类”(例如：飞机)和这个类的一般的对象(例如：6 架波音 747 和 2 坐小型

飞机)。正是“类”的概念，被后来的语言沿用，包括 C++。就这方面来说，C++ 又是 Simula 的新翻版。

一组数据对象是一类，以及处理这些数据对象的操作，这些概念是抽象数据类型的基础。我们可以定义抽象的“类”——飞机，并且声明与应用这个类的实际的对象(实例)，如波音一号……小型飞机一号……

Ada

Ada是由美国国防部资助，在 70 年代中期开发的一种语言。当时需要计算机系统有一种标准语言，用于控制一个大型系统，诸如导弹、飞机或生产线。

尽管在许多方面，Ada 语言基于 Pascal 语言，但它大得多并包含程序包的概念。一个 Ada程序包通常被定义在一个独立的库里。一个库由两个文件组成：定义文件和实现文件。定义文件给出由库提供的类和处理这些类的对象的过程；实现文件包含这些类及过程如何被实现的详细步骤。这些程序包支持“信息隐藏”(information hiding)。详细的实现过程用户是不知道的，他们只是通过它的定义文件访问一个库。

Ada 语言允许用户建立已经调试过的程序包的库，以便该程序包再次被以后的程序用到。比如，在一个编程队伍中，某个成员可能被安排开发一个独立的程序包。对于整个的软件任务，通过它的库定义文件可以知道哪个程序包应被提供，而其它的成员则并不在意这一实现过程。

Ada 允许开发样板(template)过程和程序包。这是一些子程序或提供一个“template”的程序包。通过“template”，实际的、具有许多相同之处的过程或程序包可以被定义，但各自也可具有细微差别。编译器决定所需要的特殊定义在程序码何处。我们可以考虑一个一般的程序包“飞机”，从该程序包可得到实际需要的程序包，如“波音飞机”或“两坐小型飞机”。

Ada 也能使过程重载(overloading)。overloading 允许一个过程可以有几个定义。在翻译过程中，编译器决定所需过程的特殊定义在程序码的位置，这由该过程所处理的数据类型给出(这种形式的过程定义约束叫早期或静态约束)。一个类似的例子是“飞机降落”这一过程。“波音飞机”和“两坐小型飞机”都需要“降落”，而二者有轻微的不同(如，波音飞机“降落”需一条较长的跑道……)。

1.1.2 Smalltalk——一种面向对象的语言

Smalltalk是用于面向对象编程(Object-Oriented Programming)技术主要的开发程序语言。Smalltalk 提供给程序员基本的“类”定义，并且对于一个特定的应用可由此派生出“类”层次的定义。一个“类”的对象也可产生。

类定义包括处理该类对象的方法(methods)。

一个程序由一系列送往对象的“消息”(messages)组成。对象以不同的形式响应一个消息，因为执行一个消息是由这种方法来完成的：送这个消息给相应已定义了的类的对象。当程序运行时，何种方法就能被确定。程序运行期间，方法定义与码的联结称作晚期连编(late

binding)或动态连编(dynamic binding)。

“对象以不同形式响应一个消息”这一概念叫做“多态性”(polymorphism)。多态性和类定义的继承性是面向对象编程的关键性概念。

面向对象编程技术鼓励编程人员集中精力于应用程序的对象类集。这些对象中类定义或许已存在并且正在被应用，同时类定义还会派生出相应新的类定义。因此程序易于扩展。

在应用中一旦对象的类被建立，就需要处理这些对象的方法，那么可用一些工具编写应用软件。

1.1.3 C++ 概述

C++——一种将在本书做进一步讨论的语言，从它家族的其它语言继承了许多特点。从C派生的C++结构语句形式和功能是从Algol家族来的。因此，结构式编程技术可用于开发程序源码而确保软件总是以一定的顺序执行。

应该注意到，C++在函数定义上有些变化，如**inline**函数(function)定义和传递变量给函数的形式(传值调用和引用调用)。对于从C派生出来的数据对象类型，C++沿用了提前定义的数据类型和操作数定义，也从C继承了派生的类型，只是又增加了引用类型。

通过Ada，C++从Simula继承了“类”的概念。因此，用户可以定义他们自己的数据类型。如定义一个“类”，这个“类”包括处理属于这个“类”的数据对象的函数定义。C++比Ada又进了一步，它允许使用多数操作数以及函数重载，以便当一个操作数是类的一个对象时，操作符的操作可被重新定义。

用户定义库可被开发。每个库由两个文件组成：一个头文件和一个实现文件。一旦这些库被开发和测试以后，它们在任何必要的时候都能被重复利用。库中的类定义支持数据抽象(abstraction)。

C++沿用了Ada中的**templates**的概念，但目前所有这些概念仍未有各方面都一致的统一标准。

C++也能使程序员利用面向对象编程技术来开发软件。派生出来的类(derived classes)可被定义。这些类从一个或多个基础类(base classes)继承所有的特点，并可再定义或增加这些特点。类的层次能被开发，因此一个派生类可从几个基础类多重继承而来。

更进一步，C++比C有着更强的类型化的特点。特别是，在许多情况下，编译器测定了传递给某函数的实际变量的类型，这决定特定的函数定义以函数调用的方式(static binding)插入到程序代码中。当然，这不如Pascal和Ada语言那么强类型化。例如，缺乏对预定义类型的范围检查可导致偶然发生运行错误，等等。

由于“多态性”，C++在类的层次中执行虚拟(virtual)函数。它们是带有几个有关定义的成员函数。这些定义在运行期间被确定(dynamic binding)，而不是由编译器确定。这些虚拟函数相当于Smalltalk中的方法(methods)。C++选择一个虚拟函数相当于Smalltalk中送出

一个消息(message)：响应取决于该类中被用于选择函数的对象。所以，面向对象语言更进一步的特点是多态性(polymorphism)。

C++ 允许程序员把传统的结构化编程技术与面向对象编程技术结合起来。以这种方式，程序员取两种编程技术所长而开发大型软件系统。

C++ 程序员应首先利用面向对象的技术，考虑实际中对象的“类”(classe) 和这些对象所要应用到的方法。然后，对这些类进行定义或对派生类进行定义。作为应用结构化编程技术的函数而定义的每种方法确保了执行这种方法的一系列步骤是正确的。

一旦实际应用中对象的类被确定和被完全定义，用结构化编程技术编写的应用程序再一次保证了程序运行期间每个步骤的正确性。

1.2 一个C++程序

下面是一个非常简单的 C++ 程序，它的功能是把信息“Hello,world!”在一个标准输出设备(通常是一个屏幕)上写出。根据这段程序，1.3，1.4 和 1.5 小节分别介绍其组成部分。

```
//file hello writes out a message

#include <iostream.h>    //include this header file for i/o

void main ( )
{
    cout<<"Hello,world!\n";   //output message with a newline finish
}
```

1.3 C++程序的一般形式

一个 C++ 程序由一个或多个函数所组成。对于由多个函数所组成的程序，可以把这些函数一起放在一个文件里，也可以分成几个文件来储存。通常，一个程序必须包含一个程序起始执行的主函数(main function)。**Main** 是这个主函数的标识符。Main 接下来的圆括号( ) 向编译器提示：这是一个函数。

函数体放于花括号{ }之中。括号中包含的语句序列能够使函数的任务被完成。序列的每条语句用分号来结束。

以后的章节，我们将学习如何编写其它的函数以及带有在圆括号中用来传递数据的变

量的函数。

上节程序中，用于 main( )前面的关键字 void 指明该函数不带有返回值。

1.4 注释头文件和 #include 伪指令

我们用注释语句来解释程序，使它更容易读懂。通常，注释头语句给出文件名和整个程序任务的描述。编译器忽略注释语句，当遇到注释语句时就跳过去而编译其它语句。

对短于一行的注释语句，通常是由符号 // 开头，后面接注释语句的内容。、

对长于一行的注释语句，可以把它的内容放在符号 /* 和符号 */ 之间。这些符号界限对长于一行的注释是很有用的。例如，如果调试程序——寻找程序错误原因时，也许我们希望编译器忽略几行程序代码。几乎所有的程序都需要从键盘输入和从屏幕上输出，因此这些程序要利用 C++ 中的 iostream库。iostream.h 头文件(header file)是用 #include 伪指令插入到程序始端的。编译一个 C++ 程序的第一步由预处理器执行。伪指令是面向预处理器的命令。所有预处理器的伪指令由符号#打头，预处理在编译之前执行，即程序被翻译之前执行。

#include <iostream.h>伪指令使预处理器把 iostream.h 头文件插入到程序中写有伪指令的地方。当用此伪指令时，C++ 库的头文件总是包括在尖括号 < > 中。

1.5 C++ 简单的输入/输出

就 C++ 语言来说，输入从一串连续的信息序列(流：stream)读入，输出也被写入一串信息序列。当一个程序模式包含有 iostream.h 时，几个标准流就被自动地定义了，这就是用 iostream 库来输入和输出。

流cin被用于标准输入流，一般从键盘输入；流cout被用于标准输出流，一般送至屏幕。

在上面的程序中，输出一个字符串到屏幕，我们把这个字符串插入到输出流 cout 中。

1.5.1 “输出”操作符<< 被用于放一对象到输出流中

“输出”操作符(或称“插入”操作符) << 插入数据到输出流中，它有两个操作数。左边的操作数是一个输出流对象，而右边的是要输出的数据对象。在 C++ 程序执行过程中，可以输出所有预定义类型(pre-defined type)的数据对象。操作符对于这些类型操作数的定义均在 iostream 库中。

在 1.2 的程序例子中，用 cout 和 << 输出放于双引号 “” 中的一串字符。这串字符叫字符串常量或文本字符串。其中 \n 是换行符。

一系列的 << 操作符可被用于在一条语句中输出几个数据值(注意：其中没有空格或其它分隔符，除非额外定义)。操作符被放在每个要输出的对象之前，这些对象被插入到输出流中。例如，我们可用语句写出一条信息和换行符：

```
cout<<" Hello, world! " <<'\n';
```

其中单引号被用做把单一字符括起来。

输出流 cout 被送入缓冲区：插入到 cout 中的值不直接写出到输出设备上，而是放到缓冲区暂时存储。当缓冲区装满后，其内容以单一操作的形式被输出。

1.5.2 iostream 操作符

操作符被用于在缓冲区流执行的操作。

程序的例子常用到 endl 操作符。endl 操作符插入换行和刷新缓冲区的功能——尽管缓冲区没有满也输出它的内容。重新写出上述输出字符串的语句如下：

```
cout<<" Hello, world! " <<endl;
```

带变量的操作符会用到 iomanip 库。程序例子 2.2 显示了带有一个变量的操作符 setprecision。

1.5.3 “输入操作符” >>，从输入流中取出一个对象

对于编译器，“输入”操作符(或称抽取操作符)>> 也有几个定义，使编译器用于读入预定义类型的对象。这些操作符可被重载，意思是，其中每个操作符有几个定义。当编译器读到操作符的操作数时才断定要用到哪种定义。

“输入”操作符读入预定义对象直至遇到空白符(space)，'\t'(tab)，'\n'(换行符)，或对字符型对象抽取操作符只读入一单一字符。

cin 的应用为 cout 自动刷新缓冲器。

1.6 数据对象

本章描述数据项为对象。对象可以有变化的值或恒量值，并可以有或没有标识符。

一个对象是一个可存储量值的记忆区。我们很难对记忆单元的实际数量感兴趣(除非可

能在硬件的应用方面，但这已超出本书的范围)。C++ 提供指针来管理记忆的地址。指针控制对象的地址。第四章中要进一步讨论指针。

一个对象可以有零个或多个名字。这些可以是标识符，或可以更复杂(例如，作为数组矢量的元件的对象可以被当作矢量)。

数据对象的类型决定值的类型。这些值被存储于对象中。类型决定相应记忆区的大小，并且定义了对象被用于表达式中的操作。编译器包含预定义类型和派生类型(第四章中讨论)的定义。我们将看到，“类”即为用户定义的类型。

一个对象的值以二进制数的形式存于相应的记忆区。这个二进制数如何被理解依赖于对象的类型。

1.7 库的应用

许多库被提供用于每次程序的完成。通常情况下，这些库包括ANSIC库和一些专门用于操作系统和执行环境的其它的库。

利用库的头文件，我们可利用库所提供的函数及其它定义。

利用 stdlib 库函数编制随机数发生器的程序：

该程序包括 stdlib 库及 iostream 库的头文件。利用 stdlib 库中函数 rand ( )，从由 srand ( ) 函数生成的一组数中产生一个随机数。在每次运行过程中若给出新的整型数 *seed*，则 srand ( ) 函数便产生不同的随机数。通常情况下，*seed* 由计算机时钟给出的那一时刻记取，对每次运行程序时间给一新值；此程序要求一整数值。

RAND_MAX 在 stdlib 库头文件中定义，并在执行中给出最大随机整数值。

```
//file randnumb using stdlib library definitions

#include <iostream.h>
#include <stdlib.h>

void main ( )
{
    int seed;
    cout << "Random number are from 0 to" << RAND_MAX << endl;
    cout << "Input a seed to generate new numbers:";
    cin >> seed;
    srand (seed);// random numbers generated
    cout << "Random number is" << rand ( ) << endl;    /* output first random number in
                                                          this set*/
}
```

第2章 变量声明和基本数据类型

2.1 标识符和关键字

我们对对象和函数给出标识符 (Identifier)。实际上，标识符是变量和函数的名字(后面，将给“类”以标识符)。

一个标识符由字母、数字和底线符‘_’组成。标识符的首字符必须是一个字母或一个底线符‘_’。一般地，应避免用底线符‘_’开头或结尾，因为它或许用于特殊的目的。

一个标识符不应是 C++ 的关键字。

C++ 编译器对字母的大小写是敏感的。它可以识别大写和小写字母。如：可把 count，Count 和 COUNT 区分开来。也就是说，若程序中有 count，Count 和 COUNT 的话，它们分别代表不同的变量或者函数。

在 C++ 中规定了一些预定义的标识符为编译程序所用。它们有特殊的用途，不能被重新定义。这些标识符叫关键字。用户定义的标识符不能与关键字相同。常用的关键字有：

asm	**const**	**Enum**	**inline**	**public**	**struct**	**union**
auto	**continue**	**Extern**	**int**	**register**	**switch**	**unsigned**
break	**default**	**float**	**long**	**return**	**template**	**virtual**
case	**delete**	**For**	**new**	**short**	**this**	**void**
catch	**do**	**Friend**	**operator**	**signed**	**throw**	**volatile**
char	**double**	**Goto**	**private**	**sizeof**	**try**	**while**
class	**else**	**If**	**protected**	**static**	**typedef**	

2.2 变量声明和初始化

对一个变量进行声明就是为一个已认定的对象要求和获取存取空间，并且对该对象的值进行初始化。

可在程序的任何一个地方作变量声明(不像许多其它语言那样，只允许在程序顶端或函

数的初始部位有变量声明)。

当对象被声明时，初始化就是给该变量赋初值。

2.3 基本数据类型

C++ 从 C 继承了预定义类型。这些类型是整数类型和小数类型。类似于其它许多语言，C++ 规定一个对像在被用之前要给出它的类型，然后编译器可给该对象分配合适的存储空间。因为编译器知道了预定义类型，在我们需要新对象时声明所需存储量即可。

编译器掌握每种类型的定义和存储，如管理这些类型对象的操作符的定义。操作符诸如算术操作符 +，-，* 和 /；关系操作符 <，<=，== (等于)，!=(不等于)，>= 和 > ;以及逻辑操作符 && (与)，‖(或)和 !(非)。

C++ 中的基本数据类型有：整型、字符型、浮点类型和常量类型。

2.3.1 整 型

按长短论，整型中分为：

short int	短整型
int	整型
long int	长整型

若关键字 unsigned 放于上述三种类型变量之前，即为：

unsigned	short int:	无符号短整型
unsigned	int:	无符号整型
unsigned	long int:	无符号长整型

short int，int 和 long int 三种类型都用来表示整数变量。它们所占的内存空间有所不同。如：

int 通常在 PC 机上以 2 字节 (16位)表示。变量范围从 -2^{15} 到 $+(2^{15}-1)$。

unsigned int 范围从 0 到 $+(2^{16}-1)$。

long int 通常在 PC 机上以 4 个字节表示。变量范围从 -2^{31} 到 $+(2^{31}-1)$。

unsigned long int 变量范围 0 到 $+(2^{32}-1)$。

整型变量的声明是把上述这些关键字放在变量标识符之前，比如：

```
int number;
```