A Laboratory Course in C++ (Third Edition)

大学实验课程丛书

C++

上机实践指导教程

（第三版）

[美] Nell Dale 著

马树奇 等译

Jones and Bartlett

美迪亚

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

A Laboratory Course in C++
(Third Edition)

大学实验课程丛书

C++上机实践指导教程

（第三版）

〔美〕 Nell Dale 著

马树奇 等译

電子工業出版社

Publishing House of Electronics Industry

北京 · BEIJING

内 容 提 要

随着软件项目复杂程度的日益增加，C++作为一种成熟的面向对象的程序设计语言已经在计算机软件工程领域具有了无与伦比的重要地位。本书对C++上机实践过程进行了科学的组织和安排，能够循序渐进地引导热心于C++技术的人们训练、提高自己的C++编程水平。尤其难得的是，本书的作者具有丰富的教学经验，真正按照科学的认知理论和实践来组织书中的内容，既有必要的知识点回顾，又有丰富的练习题，是广大C++初学者的良师益友。

本书适用于学习了C++基本编程知识之后，需要进一步提高编程水平的技术人员及在校学生；既适合在学校里作为上机教材，又可以满足已经工作的朋友自修的需要。

版权贸易合同登记号：01-2002-5464

图书在版编目（CIP）数据

C++上机实践指导教程（第三版）/（美）戴尔（Dale，N.）著；马树奇等译.—北京：电子工业出版社，2003.2

（大学实验课程丛书）

书名原文：A Laboratory Course in C++

ISBN 7-5053-8360-4

Ⅰ. C… Ⅱ. ①戴… ②马… 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（2002）第102818号

责任编辑：徐云鹏

印　　刷：北京天竺颖华印刷厂

出版发行：电子工业出版社　http://www.phei.com.cn

北京市海淀区万寿路173信箱　邮编：100036

北京市海淀区翠微东里甲2号　邮编：100036

经　　销：各地新华书店

开　　本：787×1092　1/16　印张：16.875　字数：420 千字

版　　次：2003年2月第1版　2003年2月第1次印刷

定　　价：25.00元

凡购买电子工业出版社的图书，如有缺损问题，请向购买书店调换，若书店售缺，请与本社发行部联系。联系电话：（010）68279077

简　介

学习C++过程中需要的支持

在此前约20年的时间里，Pascal语言一直作为计算机科学的入门课程而在各大院校讲授。但在刚刚过去的8到10年中，C++逐渐代替了Pascal的这种地位。然而即使是最极力提倡由C++代替Pascal的人士也不得不承认，对于初学者的学习而言，C++将会比Pascal难得多。

C语言以及其后的C++语言，都是为系统编程而设计的。系统程序员会很清楚自己想要做什么，以及自己编写的程序会起到什么作用。因此，C++执行的运行错误检查非常少，并且对一些非常离奇的代码也会进行编译。但另一方面对于初学者而言，他们经常不知道一些程序的准确含义，甚至经常不清楚自己编写的代码会起什么作用。因此，学生在学习过程中对于C++每一种构件语法和语义的理解就显得至关重要。在封闭的实验室进行系统的上机实践就是解决这一重要学习环节的理想途径。

计算机教学中采用的封闭实验室

著名的Denning Report[1]（Denning报告）中引入了封闭实验室（closed laboratories）一词，但没有给出其准确的定义。目前至少存在下列四种不同的定义：

1. 学生在监督下，根据计划完成其编程作业的这段时间。
2. 学生在监督下，根据计划解决一个小问题的这段训练和实践时间。
3. 使用专门准备好的实验室上机材料，使学生与计算机进行交互式学习，就像使用显微镜或者煤气灯一样。实验室应该监督并帮助学生发现相关的原理和解决方案。这种定义与Denning Report的思想最接近。
4. 上述两种或者两种以上定义的结合。

Curriculum '91[2]（91课程）报告出版后，人们在许多教育单位里都建议采用实验室上机练习。但是这里也没有包含关于应该进行哪些具体活动的准确内容。实际上，人们建议采用的许多训练活动在没有监督的（或者称为开放的）环境下也同样能够良好地进行。

本教程中定义的实验室上机活动是上述定义2和定义3相结合的产物。

开放实验室与封闭实验室的对比

尽管Denning Report和Curriculum '91隐含地表示实验室练习应该在监督下完成，但笔者认为这一点并不很重要。我的观点是，封闭的实验室练习之所以有价值的原因在于下列两方面：练习本身以及与同学、辅导教师的交流。如果没有条件进入封闭实验室环境，学生仍然可以从自己完成相关的练习中获益。

注1：参见ACM通信第32卷No.1第9~第23页的Denning, P. J. (chair) "Computing as a Discipline（编程训练）"。

注2：Tucker, A. B. (Ed.) 编写的 "Computing Curricula 1991: Report of the ACM/IEEE-CS Joint Curriculum Task Force（计算课程1991：ACM/IEEE-CS联合课程组报告）"，12月17日最终草案。ACM序号为201910。IEEE Computer Society Press序号为2220。

本教程的组织

每一章都包含三类活动：上机之前、上机实践和课后练习。“上机之前”活动包括一份作业说明以及简单的书面练习。“上机实践”活动又分为一系列课程，每一课都代表该章节中的一个概念。每一课又分为一系列练习，这些练习会详细地展示相关的概念。“课后练习”中布置的练习是与各章内容相适应的一些课外编程作业。每个练习都要求学生应用在该章节中介绍过的概念。

如果你计划把本教程用于封闭式的实验室环境，我建议“上机之前”这一部分应该在学生来实验室之前完成。学生可以在刚进入实验室的几分钟内检查他们的答案（每一章的第一课）。“上机实践”部分的活动设计为大约两小时，这也是封闭实验室通常采用的时间安排。当然，教师也可以根据情况对相关的章节进行取舍，比如只布置部分练习或者缩短上机时间。

“课后练习”的活动会给出一些编程项目。我并不建议要把所有这些内容都布置成作业。在大多数情况下只留一道题就够了，除非练习中的多个问题彼此相关。

如果不是把本教程应用在封闭实验室环境中，那么教师也可以把“上机实践”作业的全部或者一部分内容布置给学生，让他们独立完成（参见下文“灵活处理”一节）。不论是在封闭式环境中还是在开放环境中，“上机实践”和“课后练习”中的许多练习都可以按小组为单位来完成。

训练活动的理论基础

Benjamin Bloom的研究成果是我决定把每一章分成三类活动的理论基础。他在认知领域3制定了一种分类方法，把学习进步的过程分成六个难度逐渐提高的层次[3]。在确定本教程的练习活动时，我把Bloom制定的六类结合成了三类。这些类别在下面通过具体的算法（或者与程序语言相关的其他结构）为例进行了说明。

认识：学生可以发现相关的算法，并且确定对于一组给定的数据其输出应该是什么（不做变化）。

生成：学生可以生成一个类似的算法（稍有变化）。

举一反三：学生可以修改这个算法以便进行一种重要的更改（主要变化），能够把该算法应用在不同的上下文环境，能够把相关的算法结合起来，以及能够对这些算法进行比较。

“上机之前”活动处于“认识”阶段。大多数的“上机实践”活动属于“生成”阶段，适当的时候可以包含一些“举一反三”阶段的活动。“课后练习”相关的活动属于“举一反三”阶段。

这些活动还吸取了Kolb及其他一些人士关于学生如何学习的研究成果[4]。学生在学习过程中越是主动参与，他们也就越会有更大的收获。读和写都是主动参与的具体形式。因此，“上机之前”活动从阅读一份简介开始，并且有许多练习题会要求学生以书面形式解释发生了哪些现象。仅仅观看程序的运行及其答案是一个被动的过程，但由于要把其答案写下来，

注3：Bloom, Benjamin编写的“Taxonomy of Educational Objectives: Handbook I: Cognitive Domain（教育目标分类：手册I：认知领域）”，纽约David McKay于1956年出版。

注4：Svinicki, Mrilla D.和Dixon Nancy M.著“The Kolb Model Modified for Classroom Activities（针对课堂教学而修改的Kolb模型）”，选自College Teaching的Vol. 35, No. 4, Fall, 第141~146页。

这个练习就会成为一个主动参与的过程。

灵活处理

本书的设计思想之一就是为教师提供最大的灵活性。每一章都有一份作业单，这是一份表格形式的清单。“作业单”表格的第一列中列出的是该章的训练活动；在第二列中学生可以检查布置了哪些作业；第三列供他们记录需要上交什么输出结果；第四列供教师评定成绩。

第三版所做的修改

本版的主要变化是增加了新的一章，介绍模板（Template）和异常（Exception）。这一章的内容尽管插在“链接结构”的后面，但实际上其中的练习只与基于数组的表有关。

辅导材料

相关的程序、程序外壳（部分程序）以及数据文件在下列网址提供，可以由学生下载到自己的计算机上：

http://www.problemsolvingcpp.jbpub.com

大多数程序以及程序外壳都会在有相关应用的练习之前列出，但这里没有列出用于调试练习的程序。因为有些练习要求学生返回以前的一个程序或者程序外壳中，因此我建议学生制作两份复本，在复本上完成相关练习。

学生辅导材料中又有一些子目录，每一章一个目录。相关的程序和程序外壳存储在扩展名为.cpp的程序文件中。头文件存储在扩展名为.h的文件中。

如果学生使用的是某种标准化以前的C++版本，那么请查看下列网址：

http://www.problemsolvingcpp.jbpub.com

以便了解如何把磁盘上的相关程序转换成可以在自己的计算机上运行的形式。

致　谢

任何作者都不是凭空写作的。我也经常会收到一些同事正式的或者非正式的反馈意见。感谢我所在的部门中耐心地“顺便”为我解答C++相关问题的人们，还要感谢下面这些同事，他们阅读了本书的初稿：奥古斯塔学院的Mary D. Medley、北佛罗里达大学的Susan Wallace、宾西法尼亚Millersville大学的Paul Ross、阿尔伯克基市新墨西哥大学的Jeanine Ingber、Bradley大学的James C. Miller、西北拿撒勒学院的Ed Korntved、陶森州立大学的Charles Dierbach、奥玛哈市内布拉斯加大学的Mansar Zand、圣·玛丽大学的Porter Scobey、波特兰州的Kenrick J. Mock以及孟卡多市明尼苏达州立大学的Lee D. Cornell。特别要感谢Wisconsin-La Crosse大学的Mark Headington，感谢他及时而详细地解答了我的问题。

感谢Michael、Amy、Mike、Sigrid、Bobbie、Tara和Theresa，谢谢你们。

目　录

第1章　编程和解决问题的方法综述

目标：

- 能够登录到一台计算机上。
- 能够在计算机上完成下列任务：
  - 更换活动（工作）目录。
  - 列出目录中的文件。
- 能够使用编辑器及C++编译器完成下列任务：
  - 装入一个包含程序的文件。
  - 更改一个包含程序的文件。
  - 保存一个文件。
  - 编译一个程序。
  - 运行一个程序。
  - 更改一个程序并且重新运行它。
  - 改正一个有错误的程序。
  - 进入和运行一个程序。
  - 退出系统。

作业单

姓名_____　日期_____

小组_____

填写下表，表中显示了每一课都安排了哪些作业并且可以用来检查学生提交的结果：（1）实验报告、（2）输出文件清单和/或（3）程序清单。指导教师或者辅导教师（teaching assistant，TA）可以使用“成绩”列来评定学生的成绩。

活动	布置的作业：检查或列出练习编号	提交 (1)	(2)	(3)	成绩
上机之前					
复习					
上机之前的练习					
上机实践					
课1-1：检查上机之前的练习					
课1-2：基本文件操作					
课1-3：程序的编译和运行					
课1-4：程序文件的编辑、运行和打印					
课1-5：运行有错误的程序					
课1-6：输入、编译和运行新程序					
课后练习					

上机之前的准备

复习

计算机是一种可以编程的电子设备，它可以存储、检索和处理数据。这里所谈到的存储、检索和处理操作与计算机的五个基本物理部件有关，它们分别是：存储器单元、算术/逻辑处理单元、控制单元、输入设备和输出设备。这些物理部件称为计算机硬件。可以在计算机上运行的程序称为软件。编写构成软件的具体程序的过程就称为编程。

编程

程序就是一系列编写出来完成一项具体任务的指令。编程就是定义这样的指令序列的过程。这个过程包括两个阶段：确定需要完成的任务，以及用一系列指令来表达相应的解决方案。

编程的过程总是从一个具体问题开始。程序的编写不是彼此隔离的，而是编写出来以解决具体问题。确定需要做些什么就意味着列出一个具体问题的解决办法。这是第一阶段，因此也是解决问题的阶段。

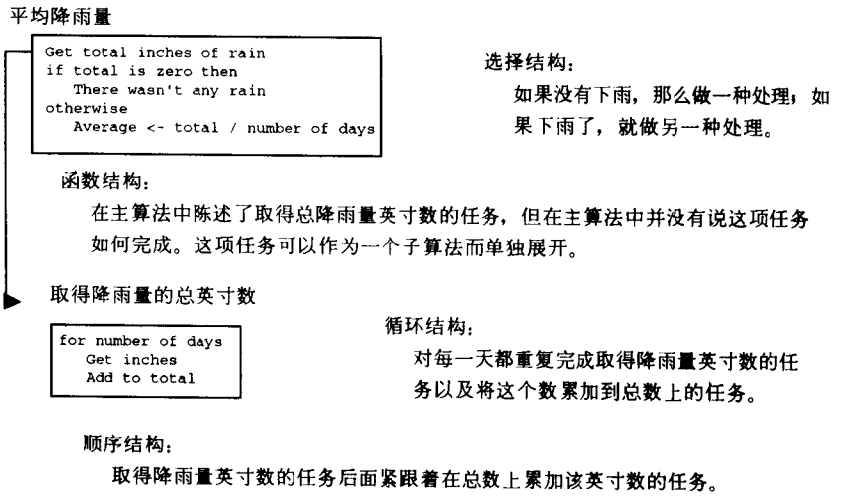
第二阶段需要把这个解决方案表达成一系列的指令，因此是实现阶段。在这里，前面解决问题阶段中指出的一般性的解决方案会被转换成一种具体的解决方案（采用一种具体编程语言编写的一段程序）。这两个阶段中都要包含测试工作。在把一般性的解决方案翻译成一段程序之前，必须要证明该方案是正确的。

我们通过下面的问题来说明这一过程。

问题：计算一定天数内的平均降雨量。

讨论：如果是手工完成这项工作，就需要把每天下雨的英寸数记下来，然后把这些数字加在一起，再将总数除以总天数。这就是我们在程序中要使用的算法。

算法：



下面列出的是实现这个算法的C++程序。如果现在还不太理解这段程序也不用担心。现在我们也并不期望大家能够理解此程序。不久之后，大家就能够理解程序中的全部内容了。

需要指出的是，位于符号/*和符号*/之间的信息是供阅读程序的人看的。这种信息称为注释，会被C++编译器忽略。从符号//到该行末尾的信息也是注释信息。

```
// Program Rain calculates the average rainfall over a period
// of days.  The number of days and the rain statistics are in
// file Rain.in.

#include <iostream>
#include <fstream>
#include <iomanip>
using namespace std;

void GetInches(ifstream&, int&, float&);
// Rain statistics are read from a file; the average is
// returned.

int main()
{
    float  average;          // average rainfall
    float totalRain;         // total accumulated rain
    int numberOfDays;        // number of days in calculation
    ifstream  rainFile;      // data file
```

```
    cout  << fixed  << showpoint;
    rainFile.open("Rain.In");
    GetInches(rainFile, numberOfDays, totalRain);
    if (totalRain == 0.0)
        cout  << "There was no rain during this period."
              << endl;
    else
    {
        average = totalRain / numberOfDays;
        cout  << "The average rain fall over "
              << numberOfDays;
        cout  << " days is " << setw(1)  << setprecision(3)
              << average  << endl;
    }

    return 0;
}

//*******************************************************

void  GetInches(ifstream& rainFile, int& numberOfDays,
      float& totalRain)
// Pre:  rainFile has been opened.
//       numberOfDays is the first value on rainFile, followed
//       by numberOfDays real values representing inches of
//       rain.
// Post: numberOfDays is read from rainFile.
//       Number of inches of rain for numberOfDays are read
//       and their sum is returned in totalRain.

{
    float  inches;          // Day's worth of rain
    int  counter;           // loop control variable

    rainFile >> numberOfDays;
    totalRain = 0.0;
    counter = 1;
    while (counter <= numberOfDays)
    {
        rainFile >> inches;
        totalRain = totalRain + inches;
        counter++;
    }
}
```

开始

有些C++系统提供了一个集成环境，可以用来生成和执行C++程序。例如Turbo C++和CodeWarrior，这两种软件可以在PC机和Mac机上运行，它们就提供了这样的环境。编译器、编辑器和实时运行系统都捆绑在一个系统中。另外一些C++编译器是单独提供的，因此你必须使用一个通用的编辑器来输入自己的程序，然后再对这个程序进行编译，然后再运行。可以使用的具体系统种类繁多，我们不可能一一进行详细的说明。在本文中我们采用一种模拟的形式来表示输入程序和运行程序的过程。

当你刚刚进入一台计算机（登录进入）时，操作系统就是当前运行的软件。你可以把这个操作系统看做是一个门厅，这里连接着其他所有的软件。你输入自己想要使用的软件名，操作系统就会提供这个软件。当你使用完毕之后，必须先回到操作系统（门厅），然后才能接着使用其他软件。

每一个软件就像一扇门。操作系统负责打开这扇门，并且引导你进入保存着你想要使用的软件的房间。例如在编辑器中，你可以生成一个书面信息文件。这个文件可以包含一段程序，或者是一段程序要使用的数据。

对于此前没有用过编辑器的人而言，可以把编辑器看做是一个程序，你通过这个软件可以像使用非常聪明的电子打字机一样使用键盘和显示屏。文件是指你通过键盘输入的信息。你可以从显示屏上看到自己输入了什么内容。编辑器中的命令可供你像亲手操作打字机一样做自己想做的事情。这些命令使你能够更改字母、单词和句子，以及对它们重新进行组织。文件驻留在二级存储区中，有自己的文件名，用来保存一系列数据组成的集合。数据本身也可以被称为是文件。

当你对自己输入的内容满意后，可以给自己的文件起个名字，并且告诉编辑器把它保存起来。为文件起个名字就像是把一个写了相关信息的标签插在文件夹上。你可以取出这份文件并且带着它从一间屋子走到另一间屋子。

当你离开编辑器的时候，操作系统会重新显示一个提示符，表示它已经准备好听从你的吩咐接着做下面的工作。如果你已经生成了一个C++程序，你可以告诉操作系统说自己想编译（翻译）这个C++文件（需要到C++房间）。操作系统会打开那个房间的房门，让你拿着自己的文件夹走到C++编译器旁。

C++编译器会试图翻译你的程序。如果这个程序中含有语法错误，编译器会如实汇报。这时你就需要返回编辑器房间更改这些错误。当你的C++程序最终编译正确之后，C++编译器会把目标程序（翻译后的代码）存放在一个文件中。

你可以告诉操作系统自己已经准备好想要运行这个目标程序了。包含翻译后的程序的文件会被带到执行房间，在那里会运行这个程序。正是在这个执行程序的地方，你为了解决某个问题而编写的这个程序才真正开始为你解决相关的问题。

当你完成所有工作准备退出（注销登录）的时候，可以告诉操作系统。操作系统会打开一扇标有退出字样的门，你就可以离开了。

有些系统把进行编辑、编译和执行程序的房间都捆绑起来形成一个豪华型的套间。例如在Turbo C++中你可以在操作系统中输入“TC”进入这个套间；而在CodeWarrior中你可以告诉操作系统要进入“CodeWarrior IDE”。接着你就会看到一个菜单，上面列出了可以

做的各种事情：生成一个新文件、修改一个旧文件、编译一个程序或者运行一个程序，甚至有一个选项可以寻求帮助。你可以使用一个鼠标（一种点取设备）或者在键盘上按一个功能键来做出选择。这些选项中有许多会给出二次菜单供你进一步选择。这种集成系统是一种对用户友好的系统：你不会被孤伶伶地留下没有人理睬。显示屏顶部的输入（编辑）、编译和运行菜单会一直引导着你完成各项工作。

上机之前的练习

姓名_____ 日期_____

小组_____

指导教师会提供一份书面材料，上面会介绍你正在使用的系统。

练习1：你使用的是什么计算机？

练习2：你使用的是什么操作系统？

练习3：你使用的是何种C++系统？

练习4：你使用的C++系统是一种集成系统还是编辑器与编译器分开的系统？

练习5：如果你的编辑器是独立的，那么这是一种什么编辑器？

练习6：如果你使用的不是一种集成系统，那么应该使用什么命令进行编译、运行和编辑？

课1-1：检查上机前的练习

姓名_____ 日期_____

小组_____

练习1：你使用的是什么计算机？在这个课程中常用的计算机有与IBM兼容的PC、Macintosh计算机和UNIX工作站。你也可能使用计算机网络上的一个终端。

练习2：你使用的是什么操作系统？Windows（95、98或者2000）、DOS、Macintosh和UNIX都是常用的操作系统。

练习3：你使用的是何种C++系统？这里有多种选择，但它必须与你正在使用的操作系统兼容。

练习4：你使用的C++系统是一种集成系统还是编辑器与编译器分开的系统？在此你只有两种选择：集成系统或者非集成系统。