21st CENTURY
实用规划教材

全国本科计算机应用创新型人才培养规划教材

数据结构与算法

主　编　佟伟光

北京大学出版社
PEKING UNIVERSITY PRESS

全国本科计算机应用创新型人才培养规划教材

数据结构与算法

主　编　佟伟光

北京大学出版社
PEKING UNIVERSITY PRESS

内 容 简 介

本书系统地介绍了数据结构的基本概念和基本算法，主要内容包括：绪论，线性表，栈与队列，串，数组、特殊矩阵和广义表，树，图，排序，查找，算法的分析与设计，实验与上机指导。

本书特别注重突出应用性和实践性，实例和习题丰富，并在附录中给出了各章习题的答案。

本书适合作为应用型本科院校和成人教育计算机专业数据结构课程的教材，也可作为数据结构培训班的教材以及软件从业人员的自学参考书。

图书在版编目(CIP)数据

数据结构与算法/佟伟光主编. —北京：北京大学出版社，2009.8

(全国本科计算机应用创新型人才培养规划教材)

ISBN 978-7-301-15584-4

Ⅰ. 数… Ⅱ. 佟… Ⅲ. ①数据结构—高等学校—教材②算法分析—高等学校—教材 Ⅳ. TP311.12

中国版本图书馆 CIP 数据核字(2009)第 128034 号

书　　名：数据结构与算法

著作责任者：佟伟光　主编

策 划 编 辑：乐和琴

责 任 编 辑：程志强

标 准 书 号：ISBN 978-7-301-15584-4/TP・1044

出　版　者：北京大学出版社

地　　址：北京市海淀区成府路 205 号　100871

网　　址：http://www.pup.cn　http://www.pup6.com

电　　话：邮购部 62752015　发行部 62750672　编辑部 62750667　出版部 62754962

电 子 邮 箱：pup_6@163.com

印　刷　者：三河市欣欣印刷有限公司

发　行　者：北京大学出版社

经　销　者：新华书店

787mm×1092mm　16 开本　20.25 印张　470 千字

2009 年 8 月第 1 版　　2009 年 8 月第 1 次印刷

定　　价：32.00 元

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。

版权所有　侵权必究　　举报电话：010-62752024

电子邮箱：fd@pup. pku. edu. cn

序

本套教材经过全国几十所高等学校老师一年多的努力，终于与广大读者见面了。我相信，它一定会受到全国高等学校计算机界老师和同学们的热烈欢迎。

随着信息技术的飞速发展，单一培养模式已经不能满足社会对计算机专业人才多样化的需求。应对这一变化的最佳办法，就是采用多种模式的培养方式。当前，高等学校的计算机教育正处于从过去的单一培养模式向多种培养模式的转变过程中，多种模式的培养方式将是必然的发展方向。

多种模式的培养方式包括：培养人才的类型不同(研究型，应用型)；专业方向不同(计算机软件，计算机网络，信息安全，信息系统，计算机应用技术等)；课程设置的多样性等。

同时，高等教育对科技人才培养的要求是：不但要培养研究型科技人才，还要为国家培养更多的应用型科技人才(或称工程型科技人才)。也就是说，培养应用型科技人才是百分之九十以上的普通高等学校的主要任务。

本套教材正是为适应多种模式培养方式的要求，并且着重于培养计算机领域高级应用型科技人才的需求，而组织编写的。

本套教材具有如下特点。

1. 基础理论够用

计算机专业所需的基础理论知识以够用为准，不是盲目扩张。如数字系统的基础知识，计算机的基本组成原理和体系结构的基础知识，离散数学的基础知识，数据结构和算法的基础知识，操作系统的基础知识，程序设计的基础知识等，都进行了必要的讲解介绍。

2. 强调理论联系实际，学以致用

每本教材的编写都将“理论联系实际，学以致用”的原则贯彻始终。例如，《计算机组成原理和体系结构》结合现代的计算机讲解，使学生学完之后，确切掌握现代计算机的组成、结构和工作原理；又如，《程序设计》结合实例讲解，使学生学完之后，真正能够动手编写程序。

3. 强调教材的配套性

根据多年组织教材的经验，只有配套性好的教材才最受教师和学生们的欢迎。我们这套教材，尽量做到了课堂教材、实训教材和教学课件完全配套，以方便教学使用。

另外，本套教材提供的是一套应用创新型计算机教育系列教材，可供不同类型学校依照自己的教学计划，根据自身的需要进行选用。

现在把这套教材奉献给全国计算机界的朋友们，真诚希望大家能够喜欢。本套教材难免会有诸多缺点或不到之处，还希望得到大家的批评和指正。

全国高等学校计算机教育研究会课程与教材建设委员会主任

李大友

2009 年 3 月

前　言

数据结构和算法在计算机学科中的地位十分重要，是设计系统程序和大型应用程序的重要基础。数据结构与算法课程是计算机科学及其相关专业的一门核心课程，本课程的学习将为学生后续的操作系统、软件工程、数据库概论、编译技术等专业基础课和专业课程的学习，以及软件设计水平的提高打下良好的基础。

数据结构与算法是一门理论与实际密切结合的课程，本课程的教学目的是教会学生将数据结构的算法理论与编程实践相结合，并能够灵活地应用在工程实践中。为了满足应用型本科院校的教学需要和社会对计算机应用人才的广泛需求，作者以热忱的工作态度投入了大量的精力和时间进行数据结构与算法课程的教学研究，尽力完善各个教学环节，力求建立针对应用型本科教学的全新课程体系，并在此基础上编写了本书。

本书内容宽泛、重点突出，既涵盖了数据结构的所有基本知识点，又考虑到应用型本科计算机专业学生学习的特点，更突出了最重要、最基本的内容，便于学生牢固掌握，灵活应用数据结构的知识解决实际问题。

本书特别注重实用性和实践性。本书在讲述每一种典型的数据结构之后，都介绍了此数据结构在计算机科学领域和软件设计中的相关应用；每一章都在应用示例与分析环节中给出大量的例题，并详细进行了解答；每章后均给出了大量的各类习题，并在书后对所有习题给出了答案，方便学生练习和参考。

本书概要地介绍了算法分析和一些常用的、经典的算法设计技术。在教学中可根据学时选讲或安排学生自学此部分内容，使学生进一步巩固所学的知识，以提高程序设计的质量，奠定扎实的软件开发基础。

本书针对各个典型数据结构问题，精心设计了实验与上机指导。每个实验都包括了【基础实验】、【提升实验】、【综合应用实验】三个部分，遵循循序渐进、由易到难的原则，旨在逐步引导学生，不断深入地学习，掌握基本数据结构的具体实现技术，学以致用，灵活地解决一些实际问题，提高程序设计的能力。

本书简明、生动、通俗易懂，在讲述上深入浅出，对学生学习中的难点均给出了较为浅显和易于理解的解释，使那些晦涩难懂的概念和算法更加清晰，便于学生接受和掌握。

本书所有算法都以 C 语言的函数形式表示。为了有效地提高编程的效率和质量，书中所有算法均在 Visual C++环境下编辑、运行。这样，既避免了 DOS 界面单一、编辑不方便等缺点，也尽量回避了 C 语言中关于指针等学生难于理解的知识。

书中的所有算法，均在 Visual C++下运行通过，无须做任何修改，学生可直接上机运行，验证这些算法。

本书的编写参考了国内外数据结构的最新教材和研究成果。例如，在树的应用中增加了关于等价类的问题，并将排序二叉树、平衡树等内容前移到树的这一章中，这样，更突出了树、图这两个学习难点的重要性，便于学生更深入地学习、理解、掌握这些知识。

此为试读，需要完整PDF请访问：www.ertongbook.com

本书由佟伟光任主编、杨政任副主编，参加本书大纲讨论和部分章节编写的还有费雅洁、史江萍、谢爽爽、赵忠诚、柴军等。李大友教授及应用型本科教材编委会、部分应用型本科院校的老师对本书的编写大纲提出了宝贵的修改意见。本书在编写过程中，得到了北京大学出版社有关同志的关心和大力支持，谨此一并表示衷心的感谢。

由于编者水平有限，加之时间仓促，书中难免存在不妥之处，请读者不吝指正。

如果读者有建议或要求，可与编者联系。编者 E-mail 地址：Weiguangt@sina.com。

编　者

2009 年 5 月

目　录

第1章 绪 论

本章导读

在计算机中如何有效地组织数据及处理数据是计算机科学的基本研究内容。本章主要讲述了数据结构中常用的基本概念、算法描述和分析方法，这些内容将贯穿数据结构课程的整个学习过程，掌握这些基本概念和方法，将有利于较好地学习以后章节的内容。

本章主要知识点

- 数据结构中常用的基本概念和术语
- 算法描述和分析方法
- Visual C++集成化开发环境

1.1 数据结构的基本概念

在计算机发展的初期，人们使用计算机的目的主要是处理数值计算问题，当时所涉及的运算对象主要是简单的整型、实型或布尔类型数据。随着计算机应用领域的扩大和软、硬件的发展，非数值处理问题显得越来越重要，据统计，当今非数值处理问题占用了90%以上的机器时间。这类问题解决的关键不再是数学分析和计算方法，而是必须研究数据间的相互关系及其对应的存储表示，并利用这些特性和关系设计出相应的算法和程序，以便有效地解决实际问题。

数据(Data)：一切能够由计算机接收和处理的对象。随着计算机技术的发展，数据这一概念的含义越来越广泛。不仅整数、实数、复数等是数据，字符、表格、声音、图形、图像等也都能够由计算机接收和处理，也都是数据。

数据元素(Data Element)：数据的基本单位，在程序中作为一个整体加以考虑和处理。换句话说，数据元素被当做运算的基本单位，并且通常具有完整、确定的实际意义。在数据结构中，根据需要，数据元素又被称为元素、顶点或记录。

数据项(Data Item)：数据的不可分割的最小单位。在有些场合下，数据项又称为字段或域。例如，将一个学生的自然情况信息作为一个数据元素，而学生信息中的每一项如学号、姓名、出生年月等为一个数据项。

数据结构(Data Structure)：数据之间的相互关系，即数据的组织形式。研究数据结构，是指研究数据的逻辑结构和物理结构。所谓数据的逻辑结构，是指数据元素之间的逻辑关系，如数组中各元素的先后次序、关系等；而数据的物理结构，是研究数据元素在计算机存储器中如何存储，如一个二维数组在内存中是如何安排的等。数据的物理结构又称为存储结构。

研究数据结构，除了要定义数据元素之间的相互关系，更重要的是要定义一组有关数据

元素的运算。例如，对于数组数据结构，除了要定义每个数组元素的类型和元素个数以外，还要定义有关数组的一些运算，如向数组中插入新元素或删除某个数组元素等。

算法(Algorithm)：对特定问题求解步骤的一种描述。它是一个有穷的规则序列，这些规则决定了解决某一特定问题的一系列运算。计算机依照这些规则对与此问题相关的一定输入，进行计算和处理，经过有限的计算步骤后得到一定的输出。

算法有以下特性。

(1) 有穷性。一个算法必须在有穷步之后结束，即必须在有限时间内完成。

(2) 确定性。算法的每一步必须有确切的定义，无二义性。算法的执行对应着的相同的输入仅有唯一的一条路经。

(3) 可行性。算法中的每一步都可以通过已经实现的基本运算的有限次执行得以实现。

(4) 输入。一个算法具有零个或多个输入，这些输入取自特定的数据对象集合。

(5) 输出。一个算法具有一个或多个输出，这些输出与输入之间存在着某种特定的关系。

算法的含义与程序十分相似，但又有区别。一个程序不一定满足有穷性。例如，操作系统，只要整个系统不遭破坏，它就永远不会停止，即使没有作业需要处理，它仍处于动态等待中。可见，操作系统不是一个算法。另外，程序中的指令必须是机器可执行的，而算法中的指令则无此限制。算法代表了对问题的解，而程序则是算法在计算机上的特定的实现。一个算法若用程序设计语言来描述，则它就是一个程序。

算法与数据结构的关系紧密，在算法设计时先要确定相应的数据结构，而在讨论某一种数据结构时也必然会涉及相应的算法。

进行计算机程序设计，掌握数据结构知识和算法设计的有关知识是十分重要的。有人称数据结构和算法是计算机程序设计的两个支柱，瑞士著名的计算机科学家 N. 沃斯教授更是提出了“算法+数据结构=程序”的著名公式，并以此公式作为他所著的一本书的书名，由此可以看出算法和数据结构二者之间的密切关系。

1.2 算法的描述

算法需要用一种语言来描述，通常可有各种描述方法以满足不同的需要。例如，可以直接用某种高级程序设计语言(如 Pascal、C 语言等)来描述算法，也可以用框图、自然语言或类高级语言来描述算法。

- 框图描述算法：这种描述方法简单、直观、易懂，在算法研究的早期曾流行过。但使用框图来描述比较复杂的算法，有时就显得不够方便，甚至难于把算法清晰、简洁地描述出来。
- 自然语言描述算法：用中文、英文等语言直接来描述算法。这种描述方法直观、易懂，但描述比较复杂的算法，有时就显得不够简洁、清晰。
- 类高级语言描述算法：用类似高级语言，或称为伪高级语言来描述算法。类高级语言与程序设计语言相仿，包括了高级语言中的基本语言成分，但较为简单。用类高级语言描述的算法，虽不能在计算机上直接运行，但简明清晰，不拘泥于高级程序设计语言的细节，容易编写和阅读。如果需要也容易改写为对应的高级语言程序。
- 高级程序设计语言描述算法：应用某种高级语言直接来描述算法(如 Pascal、C 语言、C++、Java 等)。这种描述方法必须严格按照所使用的高级语言的语法规则来描述算法，通常这种算法也称程序，可直接在计算机上被调用运行，并获得结果。

此为试读，需要完整PDF请访问：www.ertongbook.com

考虑到目前许多学校以C语言作为基本的教学语言，学生对C语言较为了解，同时用C语言描述的算法易于上机运行并直接得到验证，本书将采用C语言描述算法，所有算法都以C语言的函数形式表示：

```
函数类型    函数名(参数表列)
{
    函数内部数据说明;

        语句序列;

    }
```

C++是C的扩展，是C 的超集，完全兼容C，为了方便对算法的描述，书中同时运用了C++对C的部分扩展功能。

为了能够避免由于特殊的语法知识而增加的算法的理解难度，本书在算法的实现中引入了C++中的“引用”作为函数的一种参数传递方式。所谓“引用”也可以理解为变量的别名，它和变量名指的是同一个存储空间，那么通过引用所做的修改也将会影响所引用的变量。下面简要介绍“引用”的定义和使用。

```
int a=10;
int ra=&a;  //定义ra作为a的引用
ra=20;      //通过ra来修改a的数据，此时a的值是20
```

引用定义的一般形式：

数据类型 引用名 =&变量名;

在函数的调用中，如果希望通过参数返回值，可以通过传“地址”进行调用，也可以通过传“引用”进行调用。例如：

```
void swap(int &ra , &rb)
{
    int temp;
    temp=ra;
    ra=rb;
    rb=temp
}
void main()
{
    int a=5,b=7;
    swap(a,b);
    printf("a=%d,b=%d",a,b);//输出的结果a=7,b=5
}
```

在此例中，通过“引用”作为参数，在swap( )函数中对ra和rb进行了交换，也就是对a和b进行了交换。

常用的C语言，通常在DOS环境的Turbo C下编译运行，很不方便。Microsoft公司推出的Windows 环境下的Visual C++集成化开发工具，集程序代码的编辑、编译、连接、运行和调试等功能为一体，为编程提供了完整、方便的开发界面，有效地提高了编程的效率和质量。

为了加深对所学理论知识的理解，同时顺利地应用到实践中，书中所有算法均在 Visual C++ 环境下编辑、编译、运行。对书中的所有算法，无须做任何修改，只需要自行编写主函数，直接上机运行调用本书中的函数，就可以获得运算结果，验证这些算法。

1.3 VC++ 6.0 开发工具简介

1. Visual C++ 6.0 的工作环境

打开 Visual C++，其工作环境如图 1.1 所示。

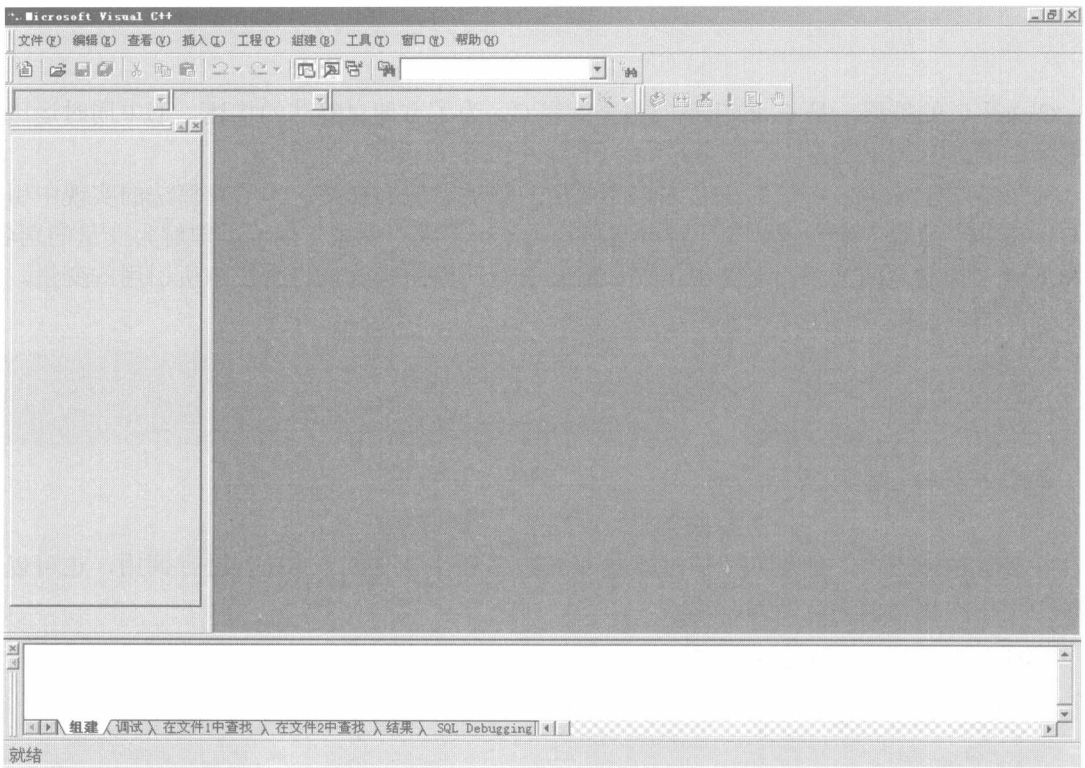

图 1.1 Visual C++ 6.0 的工作环境

Visual C++ 6.0 的工作环境可以划分为 3 块区域，最左边的区域是工作区，最下面的区域是输出区，最右面的区域是编辑区。

编辑区用来对源文件进行编辑，现在的编辑区是灰色的，表示还没有源文件在进行编辑。

输出区的作用是对程序进行编译和连接后，如果程序有错误或警告，则显示在输出区，可以对照错误或警告提示进行程序修改。

工作区的作用是用来管理各种源程序文件，在它的管理下，可以有条不紊地进行各种源文件的编辑。

可以单击工作区或输出区的关闭按钮关闭这两个区域，图 1.1 示意了输出区的关闭按钮。这两个区域关闭后能增加编辑区的面积，当对某个源程序进行编辑时，可以关闭这两个区域，需要时再打开。

打开方法：选择菜单命令“查看”→“工作空间”或“查看”→“输出”，如图 1.2 所示，分别打开这两个区域。

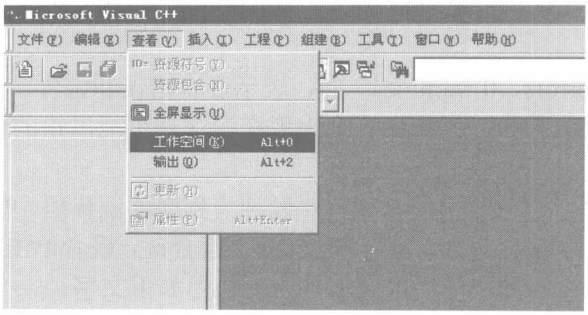

图 1.2　工作区和输出区的打开

2. 源程序的建立与编译、连接

由于本教材只是借助 Visual C++的开发环境来运行基于 C 语言编写的程序，所以不对其他功能做详细的介绍。下面用一个简单的 C 语言程序为例，来介绍如何在 Visual C++的开发环境下建立文件、编辑、编译和运行程序。例如，对于下面程序段：

```
#include <stdio.h>
void main()
{
    printf("使用 VC++开发环境！");
}
```

(1) 建立 C++源程序文件。建立方法：选择菜单命令“文件”→“新建”，显示“新建”对话框，选择“文件”选项→“C++ Source File”，输入文件名及存储路径，如图 1.3 所示。

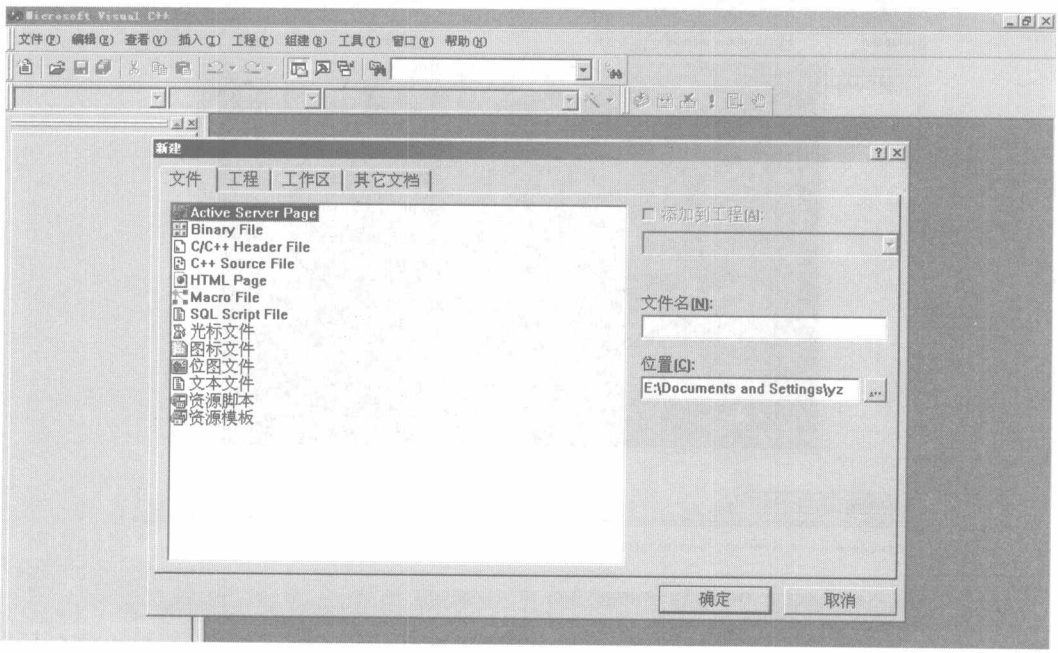

图 1.3　建立 C++源程序文件

(2) 程序的编辑与编译。编辑完成后，选择菜单命令“组建”→“编译”对源程序进行编译，编译完成后会生成以.bj 为后缀的目标文件，然后选择菜单命令“组建”→“组建”对源程序进行连接，连接完成后生成可执行文件。也可使用工具栏上的 build 工具，一次性完成编译和连接。

当输出区显示“0 errors，0 warnings”时，表示没有错误和警告，反之，则会按序号列出错误和警告。双击错误或警告，编辑标志会出现在源文件可能出错的位置，当然有时提示位置不一定很准确。

(3) 程序的执行。如图 1.4 所示的 5 行程序段称为程序的源代码，单击工具栏上的按钮可将源代码存储到文件中保存起来。然后按 Ctrl+F5 组合键，或者单击工具栏上的“红色感叹号”按钮，即可执行刚编写的程序。这时屏幕上会显示如下文字：

图 1.4　对源程序进行编译

使用 VC++开发环境！ Press any Key to continue

如图 1.5 所示为执行编写的程序。

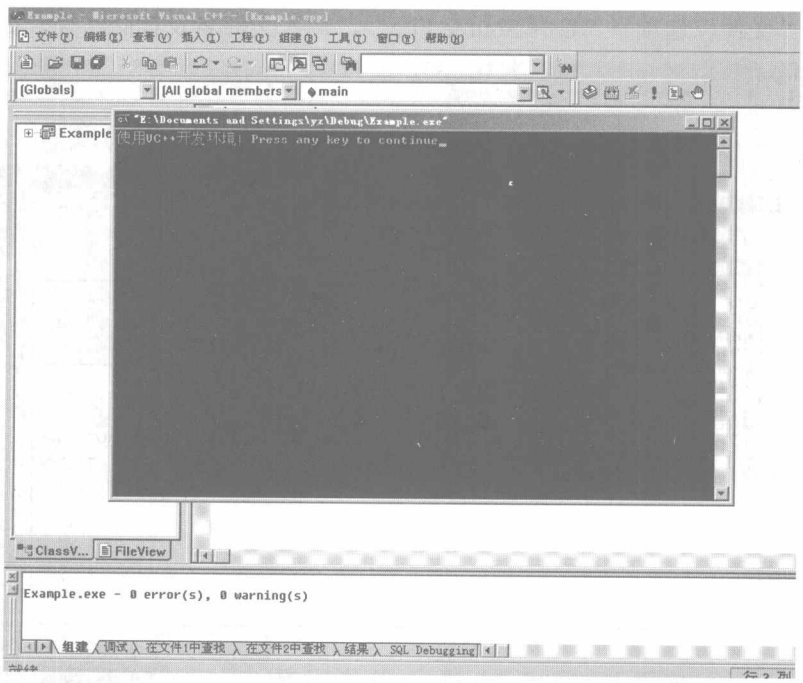

图 1.5　执行编写的程序

此时，按任何一个键都可以从输出界面返回到编辑界面。

1.4 算法的评价

通常，对于同一种问题可以有求解它的多种不同的算法，这就产生了如何评价这些算法的问题。通过对算法的评价，一方面可以从解决同一问题的不同种算法中区分相对优劣，选出较为适用的一种，另一方面也有助于设计人员考虑对现有算法进行改进或设计出新的算法。

1.4.1 评价算法的一般原则

一般来说，设计一个“好”的算法主要应考虑以下几个方面。

(1) 正确性：算法应能正确地实现处理要求，即该算法在合理的输入数据下，能在有限的时间内得出正确的结果。分析算法的正确性，对于较复杂的问题，可以将其算法分成一些局部的过程来分析，只有每个局部过程都是正确的，才能保证整个算法的正确性。

(2) 易读性：算法易读性有助于对算法的理解，便于纠正和扩充。

(3) 简单性：算法的简单性使得证明其正确性比较容易，对算法进行修改也比较方便。

(4) 高效率：达到所需的时、空性能。一个算法的时、空性能是指该算法的时间性能和空间性能。解决同一问题如果有多个算法，执行时间短的算法时间效率高，而占用存储容量小的算法空间效率高。

1.4.2 算法复杂性的分析

算法复杂性是算法运行所需要的计算机资源的量，在评价算法性能时，复杂性是一个重要的依据。计算机的资源，最重要的是运算所需的时间和存储程序和数据所需的空间资源。由于不同的计算机千差万别，运算速度和字长可以相差很大，因此，不可能以算法在某一台计算机上计算所需要的实际时间和存储单元(空间)去衡量这个算法，而一般以算法的时间复杂性与空间复杂性来评价算法的优劣。

1. 时间复杂性

算法需要时间资源的量称为时间复杂性。一个算法所需的运算时间通常与所解决问题的规模大小有关。我们用 n 作为表示问题规模的量，例如，树的问题中 n 是树的顶点数；排序问题中 n 为需排序元素的个数等。我们经常把算法运行所需的时间 T 表示为 n 的函数，记为 $T(n)$。评价一个算法时，增长率的概念是非常重要的，不同的 $T(n)$算法，当 n 增长时，运算时间增长的快慢很不相同。凡是 $T(n)$为 n 的对数函数、线性函数或多项式的(包括幂函数，幂函数是多项式的特例)，我们称这种算法为“好”的算法，而 $T(n)$为指数函数或阶乘函数的算法，因 $T(n)$随 n 的增长而增长得太快，当 n 较大时是不适用的，故称为“坏”的算法。

一个算法所消耗的时间，应该是该算法中每条语句的执行时间之和，而每条语句的执行时间是该语句的执行次数(也称为频度 Frequency count)与该语句执行一次所需时间的乘积。但是，当算法转换为程序之后，每条语句执行一次所需的时间取决于机器的指令性能、速度以及编译所产生代码的质量，这是很难确定的。为此，我们在研究时间复杂性时，假设每条语句执行一次所需的时间均是单位时间，一个算法所需的执行时间就是该算法中所有语句执行次数之和。

讨论时间复杂性时，我们往往研究所谓的“渐进时间复杂性”，即当 n 逐渐增大时 $T(n)$ 的极限情况。一般把算法的渐进时间复杂性简称为时间复杂性。为了分析方便，时间复杂性常用数量级的形式来表示，记为：$T(n)=O(f(n))$，其中，大写字母 O 为 Order(数量级)的字头，$f(n)$为函数形式，如 $T(n)=O(n^2)$。用数量级的形式表示 $T(n)$，当 $T(n)$为多项式时,可只取其最高次幂项，且它的系数也可略去不写，因为当 n 大到一定程度时，总是最高次幂占 $T(n)$的主要部分。例如：

$$T(n)=8n^3+150n+32$$

则 $T(n)=(n^3)$。

一般来说，对于足够大的 n，常用的时间复杂性存在以下顺序：

$$O(1)<O(\log_2 n)<O(n)<O(n\log_2 n)<O(n^2)<O(n^3)\ldots<O(2^n)<O(3^n)<\ldots<O(n!)$$

其中，$O(1)$为常数数量级,即算法的时间复杂性与输入规模 n 无关。

一个算法的运行时间除与问题的规模 n 有关外，往往还与具体输入的数据有关。例如，在一个一维数组中欲查找某一数据，若采用从头到尾顺序查找的算法，当碰巧待查找的数据在数组的第一个单元时，只需比较一次即可查找到，而最坏的情况，当待查找的数据在数组的最后一个单元时，则比较 n 次才可查找到。因此，在分析算法的时间复杂性时，通常用两种方法来确定一个算法的运算时间：一种是平均时间复杂性，即研究同样的 n 值时各种可能的输入，取它们运算时间的平均值；另一种是最坏时间复杂性，即研究各种输入中运算最慢的一种情况下的运算时间。采用取平均的办法似乎较好，但在很多情况下，由于各种输入数据出现的概率很难确定，分析起来比较困难，有时甚至是不可能做到的，所以一般情况下，我们是研究最坏情况下的时间复杂性。在本书中讨论的时间复杂性，除特别指明外，均指最坏情况下的时间复杂性。

2. 空间复杂性

算法需要的空间资源的量称为空间复杂性。记为

$$S(n)=O(f(n))$$

其中 n 是问题的规模(输入大小)。根据算法执行过程中对存储空间的使用方式，可以把对算法的空间代价分析分成两种：静态空间和动态空间。

(1) 静态空间。一个算法静态使用的存储空间，称为静态空间。静态空间分析的方法比较容易，只要求出算法中使用的所有变量的空间，再折合成多少空间存储单位即可。

(2) 动态空间。一个算法在执行过程中，必须以动态方式分配的存储空间是指在算法执行过程中才能分配的空间。

随着计算机硬件技术的飞速发展，扩展内存与外存的容量已并不困难，所以现在研究的算法的复杂性，重点指的是时间复杂性。

1.5 应用示例及分析

【例 1.1】 计算下面交换 i 和 j 内容程序段的时间复杂性。

```
temp=i;
i=j;
j=temp;
```


此为试读,需要完整PDF请访问: www.ertongbook.com