跟着实例学 Visual C++ 6.0

访问数据库·绘图·制表

范晓平 编著

北京航空航天大学出版社
http://www.buaapress.com.cn

配光盘

跟着实例学
Visual C++ 6.0 访问数据库·绘图·制表

范晓平　编著

北京航空航天大学出版社
http://www.buaapress.com.cn

内 容 简 介

本书第1部分是Visual C++ 6.0访问数据库，介绍了DAO、MFC ODBC、ADO DLL以及结合使用ADO ActiveX与ADO DLL四种访问数据库的方法；第2部分是绘图，介绍了扇形、条形、折线及K线四种图形的编程方法；第3部分是制表，介绍了单页、分页、动态分页及禁用打印对话框四种报表的编程方法。全书以数据库数据作为线索，将三部分连接成一个整体。

贯通全书的四个实例分别与四种编程方法对应，各形成一个完整的应用程序。跟着实例，读者可以轻松地学习三部分的编程方法与编程技巧。

本书脉络清晰，语言流畅。凡对VC++ 6.0有基本了解的读者都可以阅读本书。本书可作为高等院校VC++ 6.0课程的补充教材或上机实习教材，或者作为相关内容的培训教材。

图书在版编目(CIP)数据

跟着实例学Visual C++6.0访问数据库、绘图、制表/
范晓平编著. —北京：北京航空航天大学出版社，
2003.1

ISBN 7-81077-260-0

Ⅰ.跟… Ⅱ.范… Ⅲ.C语言—程序设计
Ⅳ.TP312

中国版本图书馆CIP数据核字(2002)第097945号

跟着实例学Visual C++ 6.0访问数据库·绘图·制表

范晓平 编著

责任编辑 张光斌 范曼华

北京航空航天大学出版社出版发行

北京市海淀区学院路37号(100083) 发行部电话：82317024 发行部传真：82328026

http://www.buaapress.com.cn

E-mail:bhpress@263.net

北京市云西华都印刷厂印装 各地书店经销

开本：787×1 092 1/16 印张：28.25 字数：723千字

2003年1月第1版 2003年1月第1次印刷 印数：5 000册

ISBN 7-81077-260-0 定价：49.00元(含光盘)

前　言

写这本书，出于偶然。

我是搞软件开发的，不是专业作者。在我承担过的软件项目中，多次遇到要在应用程序中访问多种类型的数据库，以及利用数据库数据绘制多种图形或生成多类报表。最初，为图省事，我曾经四处寻觅，想得到一本有关内容的书作参考，但终未能如愿。

项目完成了，自然积累了一些经验。推己及人，我想也许还有许多读者像我当初一样，需要类似的书作参考。据我所知，数据库访问在 VC++开发应用中是相当普遍的，大约要占 40%。绘图与制表同数据库密切相关，图形是对数据库信息的直观表示，报表则是对数据库信息的格式化表示，两者都是数据库数据信息的终端输出。可以说，前者是后二者的“原料”，后二者是以前者为原料加工制成的“产品”，三者的结合往往构成一个计算机数据处理系统的主干。由此，我萌生了一个想法，把自己在这些方面的一些编程经验或编程技巧进行整理，变成文字，与同行交流、共享。

经过对原有素材去粗存精的筛选和由表及里的提炼，再加上必要的补充，终于写成此书。全书以四个实例作为载体，容纳了访问数据库、绘图与制表三个部分的内容。四个实例贯通全书，分别与各部分的四种编程方法对应，各形成一个有机的整体。实例不仅介绍了详细的设计方法与操作步骤，而且提供了完整的源程序代码。从本书中，既可以学习 Visual C++ 6.0 访问数据库、绘图与制表三部分的编程方法和编程技巧，同时还可以直接采用四个完整、实用的应用程序，或者略加修改以适合特定的需要。由于实例的设计是分段介绍、逐步完成的，各功能设计相对独立，因此读者还可以十分方便地根据需要将单项功能交叉组合，衍生出更多独特的应用程序。

实例与全书内容、有关章节的对应关系如下表：

实例名称	访问数据库			绘　图	制　表
	访问技术	数据库	采用数据		
Example_1	DAO （第 2 章）	MS Access 2000 （第 2 章）	某地区国有企事业单位专业技术人员数构成 （第 2 章）	扇形图 （第 7 章）	单页报表 （第 12 章）
Example_2	MFC ODBC （第 3 章）	MS Access 2000 （第 3 章）	某地区国内生产总值 （第 3 章）	条形图 （第 8 章）	分页报表 （第 13 章）
Example_3	ADO DLL （第 4 章）	SQL Server 7.0 （第 4 章）	某地区各种物价总指数 （第 4 章）	折线图 （第 9 章）	动态分页报表 （第 14 章）
Example_4	ADO ActiveX 与 ADO DLL 结合 （第 5 章）	SQL Server 7.0 （第 5 章）	上证指数 （第 5 章）	K 线图 （第 10 章）	禁用打印对话框 （第 15 章）

为了便于理解和比较，特意将四个实例设计成相同的用户界面，所包含的源程序文件用相同的命名方式命名，相应功能的函数及其某些变量也使用相同的名称。因此，尽管四个实例在编程原理及实现方法上各有区别，但程序代码看起来颇为相似。

实例采用的数据是某地区专业技术人员构成、某地区国内生产总值、某地区物价指数和上证指数，具有代表性和真实性。

本书提供的实例源程序清单，是在编程过程中手工修改过的类的头文件和实现文件，能够反映程序设计的全过程。为节省篇幅，其余由应用程序向导 AppWizard 自动生成而未经修改的文件，没有一一列出。在第 1、2 部分列出的，分别是在各章中手工修改过的类的头文件和实现文件；在第 3 部分列出的，则是在全书中所有手工修改过的类的头文件和实现文件。

书中包含的四个实例，不仅是编程原理与编程方法的载体，也是学习编程原理与编程方法的向导。通过实例来掌握有关概念、原理，即从感性到理性，这符合人的认识规律。

众所周知，读技术类的书，不如看小说那样轻松。更何况，VC＋＋访问数据库、绘图与制表是 VC＋＋开发应用中比较偏难的编程技术。正因为如此，我从一名软件开发人员的角度，在写作过程中力求将本书写得通俗一些。首先是避免空泛、晦涩的叙述，而将原理、概念等融会于实例之中。专门介绍概念、原理的书到处可见，没有必要重复。其次是对于避免不了的概念、原理的说明，在不失严密的前提下尽可能写得简明、通俗一些。作家魏明伦先生在谈到他的作品时说过："你们读起来轻松，那是因为我写得很苦。"我也希望本书能使读者读起来轻松一些，在写作过程中我一直在为此努力。如果您能轻松读完本书，并能获得您所需要的东西，那将是我莫大的慰藉。

限于水平，书中错漏在所难免，请读者不吝指正。

作 者

2002 年 10 月

目 录

第 1 部分　Visual C++ 6.0 访问数据库

数据库是计算机数据处理系统不可缺少的组成部分。没有数据库，计算机数据处理系统便成了无源之水。

这一部分的主要内容是 Visual C++ 6.0 访问数据库。第 1 章是概述，第 2 章～第 5 章将分别详细介绍 DAO、MFC ODBC、ADO DLL 以及结合使用 ADO ActiveX 与 ADO DLL 四种访问数据库的方法。实例 Example_1～Example_4 将分别在第 2 章～第 5 章中创建并完成访问数据库部分的设计。

读者也许在其他有关 Visual C++ 6.0 教程中见到过介绍访问数据库的例子，但可能视图多由 CFormView 派生。在这类例子中，用户修改数据的界面与视图窗口硬性结合在一起，这为程序继续增加其他任务将带来不便。笔者设计的四个实例，视图都由 CScrollView 派生，使用对话框作为用户修改数据的界面，这为在实例中后续增加绘图、制表功能留出了余地。

除此之外，本部分包含的其他一些编程技术或技巧可能会使读者感兴趣，例如：在对话框中加入工具条、在对话框中使用 UPDATE_COMMAND_UI 机制和从对话框访问文档对象等。

第 1 章　Visual C++ 6.0 访问数据库概述

本章是对 Visual C++6.0 访问数据库的综述，其主要内容包括：

- 数据库；
- 关系数据库；
- 结构化查询语言 SQL；
- Visual C++6.0 访问数据库的技术。

1.1　数据库

在开发应用程序时，需要对数据进行组织。在组织大量数据时，需要使用数据库技术对数据进行管理。例如 SQL Server、Oracle、Access、Visual Foxpro 等就是常见的数据库。什么是数据库？简单地说，数据库就是存放数据的基地，而数据是若干记录的集合。

通常所说的数据库指的是数据库系统。一个数据库系统包括应用程序、数据库管理系统、数据库和数据库管理员。数据库系统的结构如图 1.1 所示。

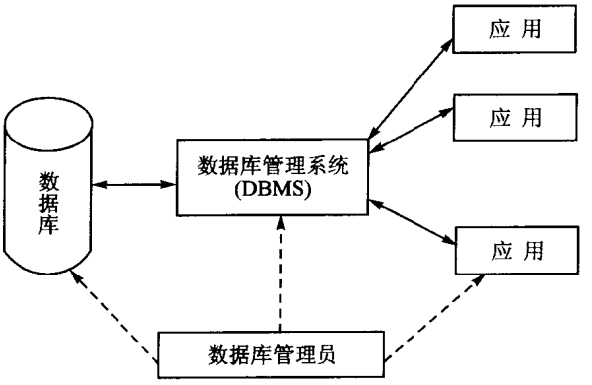

图 1.1　数据库系统的结构

数据库管理系统(DBMS)是管理数据库的软件，其实现数据库系统的各种功能。图 1.1 中的“应用”是指以数据库为基础的各种应用程序。应用程序必须通过数据库管理系统才能访问数据库。数据库管理员负责数据库的规划、设计、协调、维护和管理等工作。

数据库有 3 个特点：

(1) 涉及的数据量大。一般将数据存储在辅助存储器中，如硬盘、磁带等；

(2) 数据不随程序的结束而消失，可长期保留在计算机系统中；

(3) 数据为多个应用程序所共享。

数据库有 3 种类型：

(1) 层次型；

(2) 网络型；

(3) 关系型。

1.2　关系数据库

关系数据库是应用最多的一种数据库，本书实例中用到的数据库都是关系数据库。

在关系数据库中，数据被细分成多个单独的表(Table)，其通过惟一的标识(关键字)互相关联。

例如，要对一个商场的雇员进行简单记录。每个雇员的信息包括姓名、地址、邮政编码、部门及部门经理等。考虑到多个雇员可能在同一个部门工作，因此，不同的雇员所在的部门可能相同。为了减少数据存储的冗余，应当设计两个表来容纳这些信息。

第一个表叫做 Employee(雇员)表，如表 1.1 所示，其包含 EmpID、Name(雇员姓名)、Address(地址)、Postcode(邮政编码)及 DepID。第二个表叫做 Department(部门)表，如表 1.2 所列，其包含 DepID、Department(部门)及 Manager(部门经理)。

表 1.1　Employee 表

EmpID	Name	Address	Postcode	DepID
1	王建东	西都市新兴街 3 号	630038	1
2	李　强	西都市南云街 20 号	630025	2
3	张　华	西都市东风街 112 号	630046	1
4	赵　玲	西都市新兴街 8 号	630038	3

表 1.2　Department 表

DepID	Department	Manager
1	服务部	杜　鹏
2	家电部	蒋　丽
3	文玩部	罗　凯

表中的每一行叫做一条记录，每一列叫做一个字段。在 Employee 表中，第一个字段是 EmpID，这个 EmpID 叫做主关键字，是每个雇员独有的，对每个雇员具有惟一性。主关键字可以由表中的一个字段组成，也可以由两个以上的多字段组成，但必须保证其值在各记录中具有惟一性。主关键字并非所有的表都需要。在 Department 表中，DepID 是主关键字。

Employee 表与 Department 表是通过 DepID 联系的。DepID 既是 Employee 表的一个字段，也是 Department 表的一个字段。对于 Employee 表，它被叫做外来关键字，指向 Department 表中的主关键字。

当希望在应用程序的窗口显示雇员信息时，需要通过一种特殊的方式取出 Employee 表和 Department 表中的信息，这个操作叫做联合。所谓联合操作，就是把至少有一个字段相同的表中的数据关联在一起，也就是一个表中的外来关键字与另一个表中的主关键字相关联。通过联合 Employee 表与 Department 表，能够在 Employee 表中包含每个雇员的部门信息。

数据库服务器能够容纳多个不同的表。正常情况下，这些表不是以单个实体的形式存在的，而是被收集在一起放在两个或多个表的数据库或数据源中。

在一个数据库中，表的集合在数据库服务器看来是一个实体。数据库服务器能为这个实体提供一些有趣的服务，例如，数据库服务器能够维护任何数据库中的引用完整性。这就是说，数据库服务器将保证外来关键字列的每一项数据在对应表的主关键字中都有一个真正的同伴。另外，服务器还能使用事务(transaction)提供多个表的数据条保护，例如，大型数据库中的单个复杂操作可能需要同时更新五个表，如果该操作开始后只更新了两个表就因电源故障而无法继续，那么可以把对五个表的操作放在一个事务中，通知数据库该事务的开始，执行五个操作，然后结束；如果在数据库服务器收到结束信号之前就发生电源故障，服务器将把数据库恢复到该事务开始之前的那个状态。

1.3 结构化查询语言 SQL

访问关系数据库需要某种数据库描述语言。最常用的数据库语言就是结构化查询语言 SQL(Structured Query Language)。SQL 使用起来非常简单，但功能却非常强大。SQL 不仅可以用于管理小型数据库，也可以用于管理大型数据库。

MFC 中的 CrecordSet 类(将在 3.1.2 节中介绍)主要处理查询。要完全掌握 MFC 中的 CrecordSet 类，必须理解 SQL 查询语法；但不必理解整个 SQL 语言，只需要理解其中的 CrecordSet 类使用 SQL 语言的一个子集。

SQL 语言包括数据定义语言、数据操作语言及数据控制语言。下面只重点介绍数据操作语言。在数据操作语言中，SELECT 是最重要的一条语句，可以说，SELECT 语句是 SQL 语言的核心，是 SQL 的灵魂所在。

1.3.1 基本 SELECT 语句

SELECT 语句的功能十分丰富，可以附加许多子句。它最基本的用法是选择字段名和选择从中选出这些字段的表名，其一般形式为：

SELECT <columns> FROM <table>

例如，要从 Employee 表中检索出所有的列，可以使用如下 SELECT 语句：

SELECT EmpID、Name、Address、Postcode、DepID FROM Eemployee

使用 SELECT 语句可以创建一个记录集合，这个记录集合可以包含数据库中任何表的一列、几列或所有的列。数据库服务器将检查确认所选择的表中是否包含所要求的字段。只要有一个字段名不正确，将返回一条错误信息。

当向数据库服务器发出一个命令时，该服务器检索出要求的数据并将其放进一个记录集合，然后使用网络上的客户应用程序得到这个记录集合。

1.3.2　WHERE 子句

WHERE 是 SELECT 语句中的一个子句。使用 WHERE 子句可以从一个表中选择符合某些条件的数据。例如，从 Employee 表中检索出邮政编码为 610038 的所有雇员的信息，可以使用下面的 SELECT 语句：

SELECT EmpID、Name、Address、Postcode、DepID FROM Employee WHERE(Postcode=610038)

在 WHERE 子句中，还可以使用 AND 和 OR 运算符来增强检索功能。例如，要查询所有邮政编码为 610038 并且 DepID 为 3 的雇员，则使用如下的 SELECT 语句：

SELECT EmpID、Name、Address、Postcode、DepID FROM Employee WHERE(Postcode=610038) AND (DepID=3)

1.3.3　ORDER BY 子句

ORDER BY 是 SELECT 语句中的又一个子句。ORDER BY 可以添加到 SELECT 语句中以强迫数据库服务器对一个记录集合中的数据进行排序。

例如，使用以下语句可以使服务器按 DepID 对记录集合进行排序：

SELECT EmpID、Name、Address、Postcode、DepID FROM Eemployee ORDER BY DepID

如果要按 Name 排序，可以使用下面的语句：

SELECT EmpID、Name、Address、Postcode、DepID FROM Employee ORDER BY Name

通过使用关键字 ASC 和 DESC，可以对由 ORDER BY 选择的字段设置升序排序或降序排序。下面的语句是使 DepID 按降序排序：

SELECT EmpID、Name、Address、Postcode、DepID FROM Eemployee ORDER BY DepID DESC

在 SELECT 语句中，可以组合使用 WHERE 和 ORDER BY 子句，以实现对数据库数据信息的非常精细的查询和排序。例如，下面的语句从 Employee 表中检索出邮政编码为 610038 的所有雇员的信息，并且按 DepID 对记录集合进行排序：

SELECT EmpID、Name、Address、Postcode、DepID FROM Employee WHERE(Postcode=610038) ORDER BY DepID

1.3.4　联　合

在 1.2 节中，已经提到过联合。联合是数据库最重要的一种操作，可以返回从几个不同的表中集合起来的记录，这些不同的表是在查询过程中相互联合的。下面举例介绍如何实现将两个表联合起来。

例如，要检索 Employee 表中所有的信息，同时还要将其联合到 Department 表并显示部门和部门经理，可以使用下面的 SELECT 语句：

SELECT Employee. EmpID、Employee. Name、Employee. Address、Employee. Postcode、Employee. DepID、Department. Department、Department. Manager FROM Employee、Department WHERE Employee. DepID =Department. DepID

在联合中，是通过使用 WHERE 子句来实现的。上例中的 WHERE 子句要求数据库服务器通过字段 DepID 将 Employee 表与 Department 表联合起来，创建一个包含两个表中的字

段的记录的集合。FROM 语句告诉服务器要使用的表，字段列表规定了要显示的字段。为了将两个表中可能包含的相同字段名区别开来，所以每个字段名前面都加上了其所在的表名，并在表名与字段名之间加上“.”。

1.3.5 SQL 其他的数据操作语言

除 SELECT 外，SQL 还包括其他一些数据操作语句，简要介绍如下：

- INSERT 语句：将行(记录)插入表中。
- DELETE 语句：删除表中的行(记录)。
- UPDATE 语句：修改数据库中现有的数据。
- CALL 语句：使用 CALL 能够调用数据库中的过程。许多数据库系统有自己定义的过程，这些过程是事先编写好的 SQL 代码，用于封装和保护某个数据库。

1.4 Visual C++ 6.0 访问数据库的技术

程序设计语言用于开发应用程序。在众多的程序设计语言中，Visual C++ 6.0 以其功能丰富、表达能力强、目标程序效率高等优势得到广泛的应用。

程序设计语言与数据库是两种不同的软件开发平台。Visual C++ 6.0 本身并不包含数据库。如果 Visual C++ 6.0 要修改或使用其他数据库的数据，必然涉及两种不同开发平台之间的数据交流，即 Visual C++ 6.0 访问数据库。Visual C++ 6.0 访问数据库，是指 Visual C++ 6.0 应用程序对数据库存入或取出数据的操作。

在 Visual C++ 6.0 中，已经提供了对数据库应用的全面支持。Visual C++ 6.0 访问数据库的技术概括起来，主要有以下五种：

- ODBC API (Open DataBase Connectivity)；
- MFC ODBC (Microsoft Foundation Classes ODBC)；
- DAO (Data Access Object)；
- OLE DB (Object Link and Embedding DataBase)；
- ADO (ActiveX Data Object)。

在后面四章中，将详细介绍其中的 MFC ODBC、DAO 与 ADO 三种技术。

下面简要介绍这些技术的应用范围及特点。

1.4.1 ODBC

ODBC 是关系数据库的客户端应用程序访问数据库的一个统一接口。对不同的数据库，ODBC 提供了一套统一的 API。任何应用程序，只要系统包含有某数据库的 ODBD 驱动程序，就可以应用 API 来访问该数据库。由于 ODBC 已经成为一种标准，所以目前的关系数据库一般都提供了驱动程序，这使得 ODBC 的应用范围十分广泛，基本上可用于所有的关系数据库。

ODBC 只能用于关系数据库，这是由 ODBC 的本质决定的。使用关系数据库很难访问对象数据库以及其他非关系数据库、顺序文件系统等。ODBC 是一种底层访问技术，ODBC API 可以使客户端应用程序能够从底层设置和控制数据库，完成一些高层数据库技术无法完成的功能。

1.4.2　MFC ODBC

虽然 ODBC 提供了统一的访问数据库的接口，但是直接使用 ODBC API 访问数据库需要编制大量的代码。为了简化编程，Visual C++又提供了 MFC ODBC 类。在 MFC ODBC 类中封装了 ODBC API，提供了面向对象的数据库类。

由此可见，MFC ODBC 简化了程序设计。但是，MFC ODBC 技术没有提供对数据库的底层操作，只是一种高层访问技术，不能解决某些需要底层访问技术才能解决的问题。

1.4.3　DAO

DAO 通过 Microsoft Jet 数据库引擎来访问数据库中的数据和数据库的结构定义。Microsoft Jet 数据库引擎并不专门用于对 Microsoft Jet 数据库文件(*.mdb)的访问。通过 DAO 技术，还可以访问从文本文件(*.txt)到大型后台数据库的多种数据格式。

DAO 是一套简化数据库编程的 OLE 对象，多个 DAO 对象构成一个体系结构。

虽然 DAO 是通过 OLE 对象来实现的，但是用户并不需要去直接处理这些对象。MFC 对 DAO 进行了完整的封装，向用户提供了 DAO 丰富的数据库访问和数据库操纵手段，即 MFC DAO 类。用户只需通过 MFC DAO 类的成员变量和成员函数就可以访问数据库。MFC DAO 是微软公司推出的用在 Visual C++中访问 Microsoft Jet 数据库文件(*.mdb)的强有力的数据库开发工具。

DAO 包括了 ODBC 的绝大部分功能，可以说，DAO 在许多方面都超越了 ODBC。在一个 ODBC 程序中，只需改变类名就可以将应用程序转变成 DAO 应用程序。但是，DAO 没有 ODBC 效率高。

1.4.4　OLE DB

OLE DB 是 Visual C++开发数据库应用中所提供的新技术。与传统的数据库访问技术相比，OLE DB 技术有两点重要的改进：一是 OLE DB 技术基于 COM 接口；二是 OLE DB 对所有的文件系统包括关系数据库和非关系数据库都提供一种统一的接口。这些特性使 OLE DB 技术比传统的数据库访问技术应用更加广泛。

在传统的数据库访问技术中，只能提供对关系数据库的访问，而不能访问非关系数据库或其他的文件系统。OLE DB 可以访问任何关系数据库和非关系数据库，以及其他电子表格和各种文件系统，甚至可以访问用户自己定义的文件系统。

与 ODBC API 相似，OLE DB 也属于访问数据库技术中的底层接口。

1.4.5　ADO

ADO 对 OLE DB 的接口进行了封装，定义了 ADO 对象。ADO 技术仍然基于 OLE DB 的访问接口，是面向对象的 OLE DB 技术。

由于 OLE DB 技术属于访问数据库技术中的底层接口，使用 OLE DB 技术开发应用程序需要编制大量代码。ADO 技术则继承了 OLE DBO 技术的优点，同时又使程序开发得到简化。

第 2 章　使用 DAO

本章将结合实例 Example_1 详细介绍使用 ADO 技术访问数据库的编程方法，其主要内容包括：

- MFC DAO 类；
- 创建 MS Access 数据库 Exa_1 和数据表 Exat_1；
- 创建应用程序框架；
- 完成程序设计；
- Example_1 源代码。

2.1　MFC DAO 类

在使用 DAO 技术访问数据库之前，应该首先对 MFC DAO 类有一个基本的了解。

在 1.4.3 节已经介绍过，DAO 对象与 MFC DAO 类是两个不同的概念。DAO 是一组访问数据库的 OLE 对象，而 MFC DAO 类则是 MFC 对 DAO 进行封装后在 VC++中提供的与 DAO 的接口。MFC DAO 类与 DAO 对象的对应关系如表 2.1 所列。

表 2.1　MFC DAO 类与 DAO 对象的对应关系

MFC DAO 类	DAO 对象
CDaoWorkspace	Workspace
CDaoDatabase	Database
CDaoTableDef	TableDef
CDaoQueryDef	QueryDef
CDaoRecordset	Recordset
CDaoFieldExchange	(None)
CDaoException	Error

使用 DAO 访问数据库，不需要直接和这些对象打交道，而是通过 MFC DAO 类访问数据库。下面对 MFC DAO 类作一简单介绍。

1. CDaoWorkspace 类

CDaoWorkspace 对象代表一个 DAOWorkspace 对象。CDaoWorkspace 对象是在利用 MFC AppWizard 生成程序框架时自动产生的。一般情况下，一个数据库应用程序只需要一个 CDaoWorkspace 对象。如果用户还需要额外的 CDaoWorkspace 对象，可以调用 CDaoWorkspace 类的成员函数 Append 自行创建。