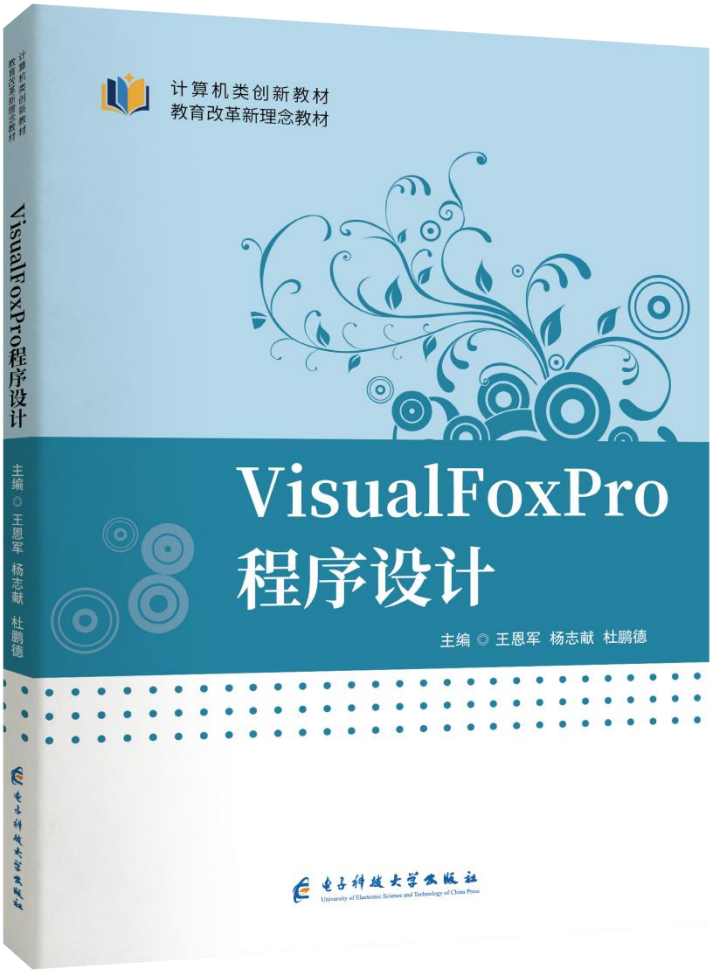
# Visual FoxPro 程序设计




**类目：**计算机类
**书名：**Visual FoxPro 程序设计
**主编：**王恩军 杨志献 杜鹏德
**出版社：**电子科技大学出版社
**开本：**大 16 开
**书号：**978-7-5770-1228-5
**使用层次：**通用
**出版时间：**2024 年 12 月
**定价：**49.80 元
**印刷方式：**单色
**是否有资源：**否

计算机类创新教材
教育改革新理念教材

# Visual FoxPro 程序设计

主编 ◎ 王恩军 杨志献 杜鹏德

策划编辑：万晓桐
责任编辑：魏 彬
封面设计：旗语书装

ISBN 978-7-5770-1228-5
定价：49.80元

电子科技大学出版社
University of Electronic Science and Technology of China Press

计算机类创新教材
教育改革新理念教材
Visual FoxPro程序设计
主编 ◎ 王恩军 杨志献 杜鹏德
电子科技大学出版社

计算机类创新教材
教育改革新理念教材

# Visual FoxPro 程序设计

主　审 ◎ 张红霞
主　编 ◎ 王恩军　杨志献　杜鹏德
副主编 ◎ 罗雨滋　黄　璟　钟前锐

电子科技大学出版社
University of Electronic Science and Technology of China Press
·成都·

**图书在版编目（CIP）数据**

Visual FoxPro 程序设计 / 王恩军，杨志献，杜鹏德主编. --成都 ：成都电子科大出版社，2024. 12.
ISBN 978-7-5770-1228-5

Ⅰ. TP311. 138

中国国家版本馆 CIP 数据核字第 2024GW3179 号

Visual FoxPro **程序设计**
Visual FoxPro CHENGXU SHEJI
**王恩军　杨志献　杜鹏德　主编**

策划编辑　万晓桐
责任编辑　魏　彬
责任校对　万晓桐
责任印制　梁　硕

出版发行　电子科技大学出版社
　　成都市一环路东一段 159 号电子信息产业大厦九楼　邮编 610051
主　　页　www. uestcp. com. cn
服务电话　028-83203399
邮购电话　028-83201495

印　　刷　涿州汇美亿浓印刷有限公司
成品尺寸　210 mm×285 mm
印　　张　13. 5
字　　数　441 千字
版　　次　2024 年 12 月第 1 版
印　　次　2024 年 12 月第 1 次印刷
书　　号　ISBN 978-7-5770-1228-5
定　　价　49. 80 元

# 前 言

QIAN YAN

党的二十大报告提出："加快发展数字经济，促进数字经济和实体经济深度融合，打造具有国际竞争力的数字产业集群。"从世界范围看，全球数字经济正在飞速发展，围绕数字经济的国际竞争不断加剧，产业的集中度越来越高。因此，打造具有国际竞争力的数字产业集群，既是拥抱新一轮信息革命浪潮的客观抉择，也是提振内需、推动高质量发展的现实需要。

随着计算机技术的发展和普及，各行各业的管理机构需要通过计算机处理大量的数据，而处理大量数据的最好方法是使用数据库进行管理。Visual FoxPro（简称"VFP"）是 Microsoft 公司推出的一款小型数据库管理系统，是用户收集信息、查询数据、创建集成数据库系统、进行实用系统开发的较为理想的工具软件。目前，尽管该软件的最新版本是 Visual FoxPro 9.0，但在学校教学和全国计算机等级考试中还依然沿用 Visual FoxPro 6.0 版本。

Visual FoxPro 6.0 使用了面向对象的编程技术。它不仅适用于单机，而且具有很好的安全性和较强的网络功能，能够实现数据的远程访问和存储加工。它可以直接访问 Access、Paradox、dBASE 等源文件中的数据而不需要任何转换，同时，它还为使用者提供了强大的向导工具。作为一种完整的编程语言，Visual FoxPro 6.0 既提供交互式的运行环境，又支持编译运行环境，是数据库应用领域使用得较为广泛的软件。

本书以 Visual FoxPro 6.0 为平台，按照"做中学、学中做"的要求，编者精心编写了这本项目-任务式 Visual FoxPro 数据库应用教材。本书分九个项目，内容涵盖：Visual FoxPro 数据库基础、Visual FoxPro 基本元素、数据库与数据表的操作、关系数据库语言 SQL、查询和视图设计、结构化程序设计、表单设计、报表设计、菜单设计。本书在内容编排上融理论学习、实例分析和实训于一体，以知识讲解为引导，兼顾能力培养，解决学生理论学习和实践训练所需。除讲授知识和技能外，本书还引导学生牢固树立法治观念，自觉践行职业精神和职业规范，并采用润物无声的方式将"课程思政"元素内化于教学任务中，培养学生具备爱国情怀、奉献精神、科学精神、工匠精神、创新意识等职业素养。

本书内容翔实，语言精练，通俗易懂，图文并茂，既可作为各类培训机构的专用教材，也可供参加全国计算机等级考试（二级 Visual FoxPro）和从事数据库程序设计的相关人员自学或参考。

编　者

2024 年 11 月

# 目 录

MU LU

# 项目一　Visual FoxPro 数据库基础

## 项目概述

数据库技术是 20 世纪 60 年代末开始兴起的一门数据处理与信息管理的学科，是计算机科学的一个重要分支，在信息技术应用领域中占据重要位置。随着大数据、云计算、人工智能等新一代信息技术的发展和应用，数据技术的创新发展不断加快，特别是大数据系统与数据库的融合并存，催生出了一系列新的模式和应用，并推动了新兴产业的发展壮大，改变了传统产业的经济发展方式。本项目我们将学习数据库基础知识及 Visual FoxPro 概述。

## 学习目标

◎知识目标

1. 掌握数据、信息、数据处理的概念。
2. 了解数据管理技术的发展。
3. 熟悉数据库系统及数据模型。
4. 了解 Visual FoxPro 的用户界面与工作方式。

◎能力目标

1. 会安装 Visual FoxPro 软件。
2. 会对 Visual FoxPro 进行启动、退出等操作。

◎素质目标

1. 具备主动学习，积极探索的意识。
2. 培养客观、严谨的科学态度。

## 任务一　数据库基础知识

### 一、数据处理的概念

数据库系统技术是利用计算机技术对各种信息和数据进行处理的最实用、最简单的技术。因此，在学习数据库系统知识前首先要了解数据、信息和数据处理这些概念。

### （一）数据

在计算机系统中，各种汉字、字母、数字符号的组合，图形、图像、音频、视频等统称为数据，数据经过加工后成为信息。

### （二）信息

信息是经过处理的有用数据。人们从自然界可以获得各种数据，这些数据并不都是信息，只有经过提炼和抽象，加工处理成具有使用价值的数据才能成为信息。显然，信息是有用的数据，数据是信息的表现形式。

### （三）数据处理

数据处理也称“信息处理”，是指利用计算机对各种类型的数据进行收集、存储、加工、分类、维护、检索和传输等处理的过程。通过数据处理得到的新信息反映了事物的本质和特征，为人们在管理活动中的行为和决策提供理论依据。利用计算机对数据进行处理，一般来说分为如下五个基本环节：原始数据的收集、数据的编码、数据输入、数据处理、数据输出。

## 二、数据管理技术的发展

随着计算机软件、硬件技术的不断发展，以及计算机网络的推广普及，利用计算机处理数据的需求量日益扩大，从而推动数据管理技术也不断发展，以适应计算机高效管理海量数据的需求。数据管理技术从产生至今主要经历了人工管理、文件系统管理、数据库系统管理三个阶段。

### （一）人工管理阶段

20 世纪 50 年代以前，计算机主要用于科学计算。当时的计算机比较落后，硬件上，只有磁带、卡片、纸带等小容量、低速的外部存储器，没有硬盘等高速、大容量的存储设备；软件上，没有操作系统、没有专门的软件对数据进行管理，数据处理方式是批处理。数据管理只能由程序员自行设计，一组数据对应一个应用程序，如图 1-1 所示。这种数据管理方式导致数据独立性差，数据和应用程序不可分割；程序之间数据不能共用，没有共享性；数据也不能独立保存，是最原始的数据管理方式。

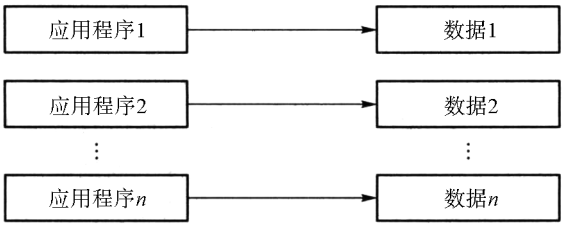



图 1-1　人工管理阶段应用程序和数据之间的关系

### （二）文件系统管理阶段

20 世纪 50 年代末至 20 世纪 60 年代中期，计算机不仅用于科学计算，还开始对大量的数据进行管理。此时的计算机，在硬件上已经有了磁盘、硬盘等大容量的外部存储设备；在软件上出现了操作系统，操作系统中的文件系统专门管理数据文件。数据按一定组织形式存储在数据文件中，应用程序通过文件系统从数据文件中读取数据，如图 1-2 所示。这种利用文件管理数据的方式使数据文件与应用程序分开，数据文件保存在存储器中可反复使用，数据与程序之间有了一定的独立性，但仍存在下面一些缺陷。

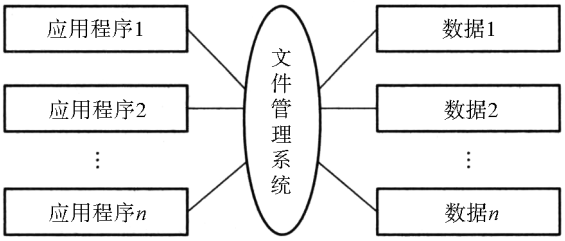



图 1-2　文件系统阶段应用程序与数据之间的关系

（1）数据冗余度大。相同的数据用于不同的应用程序中时，经常需要重复定义、重复存储。

（2）数据不一致性。由于相同数据要重复存储、单独管理，因而给数据的修改和维护带来难度，容易造成数据的不一致。

（3）数据联系差。数据文件中数据由一条条记录组成，每条记录由固定的几个字段构成，每个数据文件内部有一定的结构，但各数据文件之间是孤立的，数据之间的联系差，整体上不能反映出现实事务

之间的内在联系。

（4）数据独立性差。文件系统中的数据文件是为某一特定应用程序服务的，文件中的数据结构对该应用程序来说是最优的，如果应用程序增加一些新应用，原数据文件就适应不了，导致应用系统不易扩充；反之，一旦数据文件中的记录结构改变，也必须相应更改应用程序中数据结构的定义，因此数据与程序之间仍缺乏独立性。

### （三）数据库系统管理阶段

20 世纪 60 年代后期，随着计算机的迅速发展，计算机的应用深入社会的方方面面，需要利用计算机处理的数据类型也越来越广泛，数据量日益增多，采用文件系统管理数据已适应不了这种大批量、多类型的数据处理，数据管理进入了数据库系统阶段。

数据库系统采用专门的数据库管理软件来管理数据，数据按一定的组织形式存放在数据库中，应用程序与数据完全分开，应用程序对数据的访问通过数据库管理系统，由数据库管理系统根据应用程序的请求从数据库中读取数据，对数据的维护也由数据库管理系统专门管理，如图 1-3 所示。

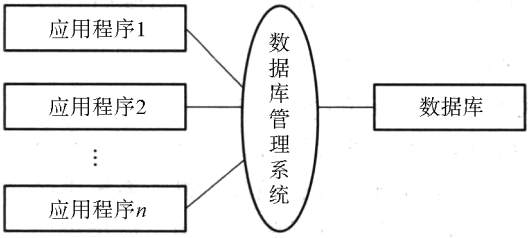



图 1-3　数据库系统管理数据

数据库系统管理数据有如下几个特点。

（1）数据结构化。数据库中每个数据库文件内的数据按一定的组织形式存放，数据项之间有结构，同时数据库中各数据库文件之间也有联系，整体也有结构。

（2）数据共享程度高。数据库系统中的数据库文件不是面向某个具体程序，而是面向整个数据库的系统，因此数据库文件可被多个应用程序共同使用，共享程度高。

（3）数据独立性强。数据库文件由专门的数据库管理系统管理，应用程序访问数据由数据库管理系统控制，不涉及数据库中具体数据结构，因此数据库中数据结构的改变对应用程序没有影响，数据独立性强，这是数据库系统的重要特点之一。

（4）冗余度小、易扩充。由于数据库系统中数据库文件之间有联系，整体结构化，数据冗余明显减少。

从 20 世纪 60 年代末美国 IBM 公司成功地研制出第一个商品化的数据库管理系统——IMS（information management system）系统起，相继出现了 dBASE、FoxBase、FoxPro、Visual FoxPro 等数据库管理系统。目前，广泛使用的数据库管理系统软件正是面向对象的、由事件驱动的数据库管理系统软件——Visual FoxPro。

## 三、数据库系统

### （一）数据库

数据库（database，DB）就是按一定组织形式把相关的数据组织在一起，长期保存在计算机存储设备上，由统一的数据库管理系统管理，可为多个用户共享的数据的集合。数据库的特点有：数据库中既有记录事物的数据，又含有事物之间的联系，整体结构化；数据库中的数据按一定的数学模型组织、存储，具有较小的冗余度；数据库中数据独立于应用程序，有较高的独立性和易扩展性，并为各种用户共享。

由于数据库中数据独立于使用数据的应用程序，因此对于数据库中的数据的输入、存储、检索、维护和使用都必须由专门的数据库管理系统软件进行统一控制。

### （二）数据库管理系统

为科学地组织和存储数据、高效地获取和维护数据而配置的系统软件被称为数据库管理系统（database management system，DBMS）。数据库管理系统是在操作系统的支持下运行的系统软件，Visual

FoxPro 就是一种在微机上运行的 32 位数据库管理系统软件。

数据库管理系统主要对数据库进行统一管理和控制，维护数据库的完整性和安全性，以保证应用程序高效的使用数据库，是数据库系统的核心软件。其主要包括数据定义、数据操作和数据控制三大功能。

### （三）数据库系统

数据库系统（database system，DBS）是指在数据库管理系统支持下，系统开发人员利用数据库资源开发出来的，面向某一类实际应用而建立的实用软件系统。

数据库系统主要由计算机硬件环境、支持数据库运行的软件环境、数据库以及开发、管理、使用数据库应用系统的人员组成。

硬件环境是数据库系统的物理支撑，包括计算机硬件系统的各种物理设备，硬件的配置能满足整个数据库系统的需求，比如要有足够大的内存空间保证数据库管理软件、应用程序的正常运行，很大的外存容量便于保存数据库、系统软件和应用软件等。

软件环境是支持数据库系统正常运行的系统软件和应用软件。其中，系统软件主要包括操作系统、数据库管理系统和开发应用系统所需的高级语言及编译系统、开发应用系统的工具软件等，这些为应用系统的开发提供了必要的软件环境，应用软件是软件研发人员在数据库管理系统的基础上，利用数据库资源，根据用户实际需要开发的应用程序。

数据库是数据库系统的核心和主体，主要为应用程序提供所需的数据源，由数据库管理系统统一管理和控制。

人员是管理、开发和使用数据库系统的全部人员，主要包括系统分析员、数据库设计员、应用程序员、数据库管理员和最终用户。系统分析员负责应用系统的需求分析和规范说明，他们和用户及数据库管理员一起确定系统的硬件配置，并参与数据库系统的概要设计；数据库设计员负责数据库中数据的确定、数据库各级模式的设计；应用程序员负责设计应用系统的程序模块，根据用户需求编写使用数据库的应用程序；数据库管理员负责全面管理、控制和维护数据库系统；最终用户是数据库系统的使用者，用户通过应用系统提供的用户接口界面访问和使用数据库。

## 四、数据模型

数据库中的数据不仅要记录客观事物本身，还要反映出事物之间的联系，因此数据库中的数据必须有一定的结构，这种结构用数据模型来表示，数据模型能正确地反映出数据之间存在的整体逻辑关系。它包含数据结构、数据操作和数据约束条件三方面的内容。

### （一）实体和联系

#### 1. 实体

在数据库中，客观存在且相互区别的事物被称为实体。实体可以是具体的人或事物，也可以是抽象的概念和事件，比如一个学生，一门课程，学生所选的课程。反映实体特征和性质的称为属性，比如一个学生的学号、姓名、性别、所在系、出生日期和照片等，这些都是用于描述学生特征的，是一个学生的所有属性。同类实体的所有属性的集合被称为实体型，比如全体学生的所有属性集合。

#### 2. 联系

两个或两个以上实体间的对应关系被称为联系。它反映现实世界事物间的相互关系，两个实体间的联系可以分为三种类型：一对一联系、一对多联系和多对多联系。

一对一联系指的是，如果实体型 A 中的一个实体只与实体型 B 中唯一的实体对应；反过来，实体型 B 中的一个实体也只与实体型 A 中的一个实体对应，则实体型 A 和实体型 B 之间的联系是一对一的联系。比如有实体型 A（班长），实体型 B（班级），实体型 A 中每个班长只能属于一个班级，实体型 B 中每个班级也只能有一个班长，则实体型班长和实体型班级之间是一对一联系，如图 1-4 所示。

一对多联系指的是，如果实体型 A 中的一个实体与实体型 B 中的多个实体对应，反过来，实体型 B

中的一个实体只与实体型 A 中的一个实体对应，则实体型 A 和实体型 B 之间的联系是一对多联系。比如有实体型 A（班级）与实体型 B（学生），实体型 A 中每个班级可以有多个学生，实体型 B 中每个学生只可能属于一个班级，则实体型班级和实体型学生之间是一对多联系，如图 1-5 所示。

多对多联系指的是，如果实体型 A 中的一个实体与实体型 B 中的多个实体对应，反过来，实体型 B 中的一个实体也与实体型 A 中的多个实体对应，则实体型 A 和实体型 B 之间的联系是多对多联系。比如有实体型 A（学生）与实体型 B（课程），实体型 A 中每个学生可选多门课，实体型 B 中每门课程也可以有多个学生选，则实体型学生和实体型课程之间是多对多联系，如图 1-6 所示。

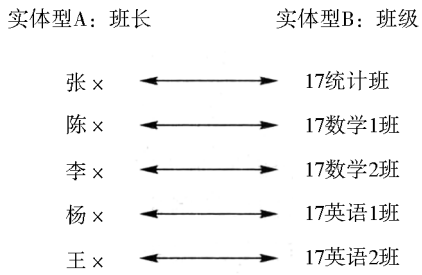



图 1-4　一对一联系

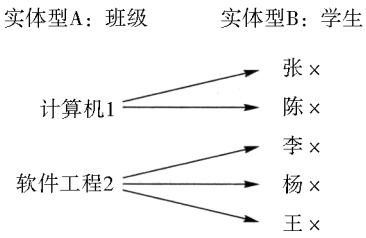



图 1-5　一对多联系

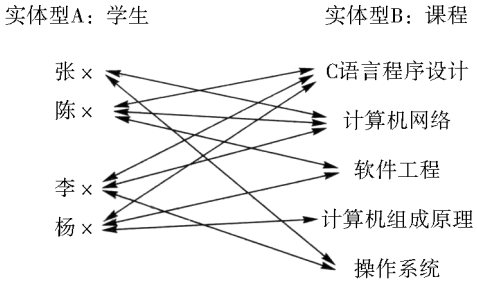



图 1-6　多对多联系

## （二）数据模型

反映实体及实体间联系的数据组织结构和形式被称为数据模型，数据模型是数据库系统中的一个关键概念，数据模型不同，相应的数据库系统就完全不同，任何一个数据库管理系统都是基于某种数据模型的。数据库管理系统常用的数据模型有三种：层次模型、网状模型和关系模型，因此，基于使用某种特定数据模型的数据库管理系统开发出来的数据库系统相应地被称为层次数据库系统、网状数据库系统、关系数据库系统。

### 1. 层次模型

层次模型（hierarchical model）是最早的数据库系统采用的数据模型，通过从属关系描述数据间的联系，用树形结构来表示。它只能表示数据之间的一对多联系，不能表示多对多联系。

在层次模型中每一个节点表示一个实体型，实体型之间的联系用节点之间的连线表示，比如用层次模型来表示学校、学院和系之间的关系，如图 1-7 所示。其数据结构的特点如下。

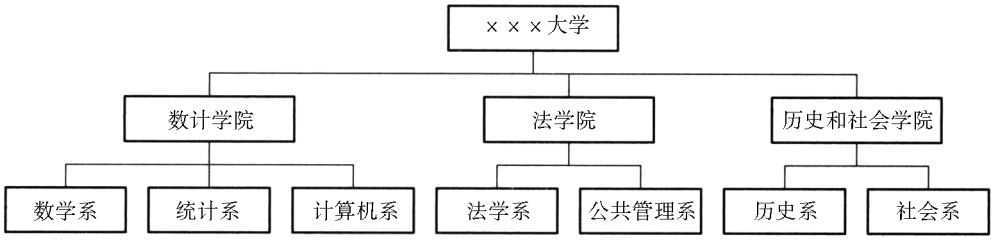



图 1-7　层次模型示例

- 有且仅有一个节点没有父节点，这个节点称为根节点。
- 除根节点之外的其他节点，有且只有一个父节点。其可以有多个子节点。
- 上一层实体型和下一层实体型之间的联系是一对多。

### 2. 网状数据模型

网状模型（network model）是采用网状结构来表示数据间联系的数据模型。它是层次模型的扩展，相当于多个从属关系的综合，如学生和课程之间、教师和课程之间的关系用网状模型表示，如图 1-8 所示。网中的每个节点表示一个实体型。它能够表示实体型间的多种复杂联系和实体型之间的多对多的联系。其数据结构的特点如下。

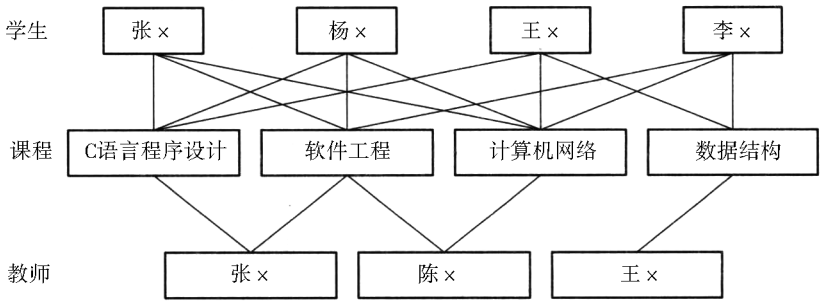



图 1-8　网状模型示例

· 容许有一个以上的节点没有父节点。
· 一个节点可以有多个父节点，也可以有多个子节点。
· 上一层实体型和下一层实体型之间的联系是多对多。

### 3. 关系模型

关系模型（relational model）是用二维表格结构来表示数据间联系的数据模型。关系是由若干个二维表组成的集合。关系模型是当今最流行的数据库模型，Visual FoxPro 就是一种典型的关系型数据库管理系统。在关系型数据库中，一个关系就是一张二维表，每个关系有一个关系名。

## 任务二　Visual FoxPro 概述

Visual FoxPro 既是关系型数据库管理系统，又是可视化的面向对象的集成开发工具。它具有强大的数据库管理功能，支持自含型语言和结构化查询语言对数据的操作，拥有 500 条命令和 200 余种函数。

Visual FoxPro 既支持传统的面向过程的程序设计，又支持面向对象的程序设计方式。

使用 Visual FoxPro 提供的项目管理工具和向导、生成器、设计器等可视化开发工具，用户可以简便、快速地开发应用程序。

### 一、Visual FoxPro 的发展历程

在 20 世纪 70 年代末期，美国的 Ashton-Tate 公司研制的 dBASE 是最流行的微机关系数据库管理系统。1986 年，美国 FOX 软件公司发布了与 dBASE 兼容的 FoxBase。它功能更强大，运行速度更快，很快成为 20 世纪 80 年代中期主导的微机数据库管理系统。

1989 年，FOX 软件公司开发了 FoxBase 的后继产品——FoxPro 1.0 版，1991 年推出FoxPro 2.0版。FoxPro 2.0是一个 32 位软件产品，使用了 Rushmore 查询优化技术、先进的关系查询与报表技术，以及第四代语言工具，性能大幅提高。

1992 年，微软公司收购了 Fox 公司。它利用自身的技术优势和巨大的资源，在不长的时间里开发出 FoxPro 2.5、FoxPro 2.6 等大约 20 个软件产品及其相关产品，支持 DOS、Windows、Mac 和 UNIX 4 个操作系统平台。

1995 年，微软公司发布了 FoxPro 的新版本 Visual FoxPro 3.0，它全面支持面向对象技术和可视化编程技术。1998 年，微软公司推出了可视化编程语言集成包 Visual Studio 6.0。Visual FoxPro 6.0 是其中的一个产品。

进入 21 世纪以来，微软公司又相继公布了 Visual FoxPro 7.0（2001 年）、Visual FoxPro 8.0（2003 年）和 Visual FoxPro 9.0（2004 年）。这些版本都没有发布中文版，目前，在学校教学和全国计算机等级考试中还依然沿用 Visual FoxPro 6.0 版本，本书仍将以其为例进行阐述。

## 二、Visual FoxPro 用户界面与工作方式

### （一）Visual FoxPro 界面

当 Visual FoxPro 启动后，进入如图 1-9 所示的 Visual FoxPro 6.0 系统主界面，主界面由标题栏、菜单栏、工具栏、工作区、状态栏和命令窗口等组成。


图 1-9　Visual FoxPro 系统主界面

#### 1. 标题栏

标题栏位于主界面的第一行，显示 Visual FoxPro 的有关信息，包括程序图标、标题、最大化、最小化以及关闭按钮。

#### 2. 菜单栏

菜单栏位于标题栏下方，是各种操作命令的分类组合。单击其中任一个菜单，可弹出相应的下拉式菜单，在下拉式菜单里会显示若干菜单项，用户可以通过单击的方式执行菜单项操作。

在 Visual FoxPro 菜单中菜单栏的菜单会随着使用环境的变化而发生改变，也就是说，当前运行的程序不同，所显示的横向主菜单和其子菜单也不尽相同。

#### 3. 工具栏

工具栏位于菜单栏下方，由若干个工具按钮组成，每个按钮对应一个特定的功能，通过这些快捷按钮用户可以快速地访问相应的菜单命令，用户也可以根据自己的需要用鼠标将工具栏拖曳到窗口的任意位置。

用户可以通过选择“显示”→“工具栏”命令，弹出“工具栏”对话框，如图 1-10 所示。通过“工具栏”对话框显示或隐藏工具栏。


图 1-10　“工具栏”对话框

#### 4. 命令窗口

命令窗口是 Visual FoxPro 命令编辑和执行的窗口，用户可以通过在命令窗口中输入相应的命令实现数据库的管理，也可以对命令进行修改、删除、剪切、复制、粘贴等操作，还可以在命令窗口建立和运行命令文件。用户可以通过“窗口”菜单来控制显示或隐藏命令窗口。

#### 5. 工作区

在工具栏与状态栏之间的一大块空白区域是系统工作区，各种工作窗口在这里显示。

#### 6. 状态栏

状态栏位于屏幕的最底端，用于显示某一时刻管理数据的工作状态。

如果当前没有打开的表文件，状态栏的内容是空白的；如果当前已打开表文件，状态栏显示表名、表所在的数据库名、表当前记录的记录号、表中的记录总数、表中的当前记录的共享状态等内容。

### （二）Visual FoxPro 的工作方式

Visual FoxPro 提供了两种常见的工作方式：一种是命令方式，另一种是程序操作方式。

### 1. 命令方式

命令方式即命令执行方式或者界面操作方式。用户先在 Visual FoxPro 的命令窗口输入一条指令，然后按 Enter 键可以立即执行该指令，同时从主界面中立即可以看到该程序运行的结果，使用户的操作更加直观，可以说这是字符操作的一种方式；如果用户不习惯用键盘，也可以采用鼠标单击菜单和窗口的方式来实现同样的功能。

### 2. 程序操作方式

Visual FoxPro 用户可以将一系列命令存储到程序文件中，再通过特定的命令（如 Do 命令）来调用这个程序文件，此时，Visual FoxPro 就可以自动逐条执行程序文件中的每一条命令，而不需用户介入。

程序方式不仅运行效率高而且程序文件可以重复执行，提高了代码的利用率。

## 三、Visual FoxPro 的语法约定

Visual FoxPro 系统提供了一系列命令和函数，书写这些命令和函数必须遵循一定的规则。为了便于说明，本书在语法书写的过程中采用如下符号规定。

[ ] 表示方括号内的项是可选的。

<> 表示必须提供一个特定类型的值，以满足尖括号内项的要求。

| 表示在其中多个选项中选择一项。

**注意：**

*在实际使用中，这些符号无须出现在命令行中。输入时，所有的命令、函数、运算符和标点符号必须在英文输入状态下输入。*

## 四、Visual FoxPro 的命令格式

Visual FoxPro 命令一般由命令名及通过子句所指定的操作对象、操作范围、操作条件等若干选择项所构成。

【格式】<命令名>[<表达式表>][<范围>][FOR<条件>][WHILE<条件>][TO FILE<文件名>| TO PRINTER | TO ARRAY<数组表>| TO<内存变量>][ALL [LIKE | EXCEPT<通配符>] ][IN<别名>]

【说明】

（1）命令名：所有命令都以命令名开头，它是一个表示完成何种操作的英文单词。

（2）表达式表：是一个或多个由逗号隔开的表达式，用来表示该命令的操作对象。

（3）范围：表示该命令对数据库文件记录的操作范围，有以下四种选择。

RECORD N：对第 *N* 条记录进行操作。

NEXT N：对从当前记录开始的 *N* 条记录进行操作。

ALL：对所有记录进行操作。

REST：对从当前记录开始到文件结束的所有记录进行操作。

（4）FOR<条件>：对所有满足“条件”的记录进行操作。

（5）WHILE<条件>：对直到不满足条件记录为止的所有已满足条件的记录进行操作。

（6）TO 子句：用来控制操作结果输出，可以输出到文件、打印机、数值、内存变量。

（7）ALL [LIKE | EXCEPT<通配符>]：用来指明包括或不包括与通配符相匹配的内存变量或字段变量。通配符指“?”号和“*”号：“?”号代表任何一个字符，“*”号代表任意一串字符。

（8）IN<别名>：该子句允许在当前工作区下操作其他工作区中的数据库文件。

# 实训任务：安装 Visual FoxPro

## 任务说明

Visual FoxPro 的安装一般有两种方式：通过软件光盘安装或从硬盘安装（可以通过网络搜索并下载 Visual FoxPro 的安装程序）。本实训以使用软件光盘安装 Visual FoxPro 为例介绍其具体操作。

## 任务实施

步骤 1　将 Visual FoxPro 软件光盘放入光驱，这时 Visual FoxPro 安装程序会自动启动并打开如图 1-11 所示的“Visual FoxPro 6.0 安装向导”对话框。

**注意：**

在安装 Visual FoxPro 之前最好先关闭杀毒软件及其他应用程序。除可用光盘安装软件外，还可通过硬盘安装，即：首先打开 Visual FoxPro 安装程序所在的文件夹，然后双击 Visual FoxPro 的安装程序 SETUP.EXE，打开“Visual FoxPro 6.0 安装向导”对话框。

步骤 2　单击“下一步”按钮，打开如图 1-12 所示的“最终用户许可协议”界面，先选中“接受协议”单选钮，然后单击“下一步”按钮打开“产品号和用户 ID”界面。


图 1-11　“Visual FoxPro 6.0 安装向导”对话框

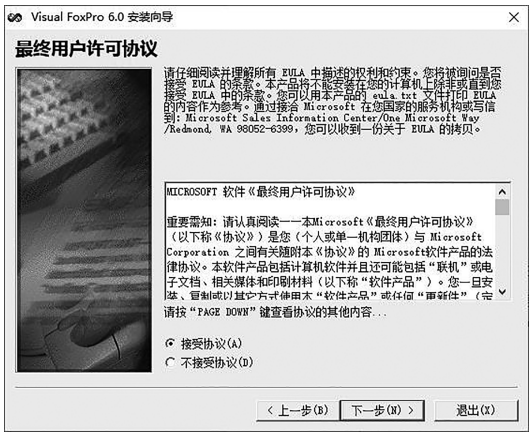



图 1-12　“最终用户许可协议”界面

步骤 3　在如图 1-13 所示的“产品号和用户 ID”界面中先输入产品的 ID 号（通常会写在光盘封面上，如果是通过网络下载的安装程序则通常会在软件安装文件夹中的“说明.txt”或“sn.txt”文档内），然后单击“下一步”按钮打开“选择公用安装文件夹”界面。

步骤 4　在如图 1-14 所示的“选择公用安装文件夹”界面中可以通过单击“浏览”按钮重新选择程序的安装位置，不过通常我们保持默认设置，直接单击“下一步”按钮即可。


图 1-13　“产品和用户 ID”界面

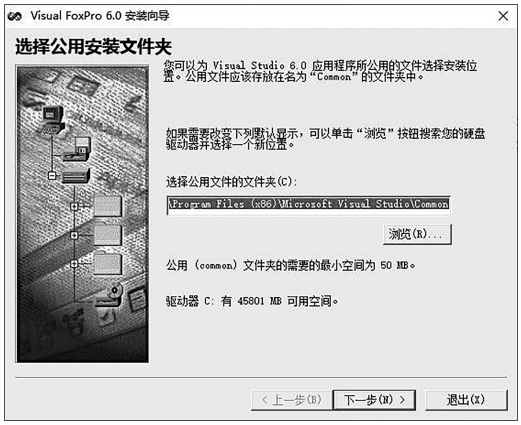



图 1-14　“选择公用安装文件夹”界面

步骤 5　在打开如图 1-15 所示的“Visual FoxPro 6.0 安装程序”对话框中单击“继续”按钮。

**注意：**

假如用户使用的是 Windows 7 及以上版本的操作系统，当弹出类似如图 1-16 所示的对话框时直接单击“运行程序”按钮即可。

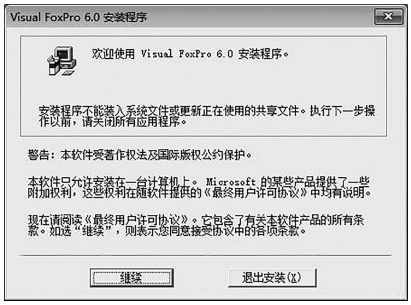



图 1-15　“Visual FoxPro 6.0 安装程序”对话框

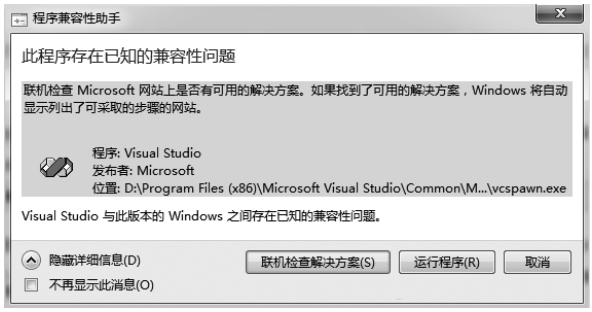



图 1-16　提示对话框

步骤 6　在打开如图 1-17 所示的“Product ID”界面中显示了用户的产品 ID，直接单击“确定”按钮即可。

步骤 7　在打开如图 1-18 所示的“选择安装类型”界面中选择 Visual FoxPro 的安装类型，本例单击“典型安装”按钮，按照程序默认设置开始安装 Visual FoxPro（如果要修改 Visual FoxPro 的安装位置或安装组件，可先单击“自定义安装”按钮，然后根据提示进行操作即可）。


图 1-17　“Product ID”界面


图 1-18　选择安装类型

步骤 8　Visual FoxPro 安装结束后会弹出如图 1-19 所示的“安装完成”界面，单击“确定”按钮即可。

步骤 9　这时 Visual FoxPro 6.0 安装向导会打开如图 1-20 所示的界面。如果用户的安装光盘（或程序）中带有 MSDN（Visual FoxPro 的帮助文档），则可以勾选“安装 MSDN（I）”复选框以安装帮助文档，否则请取消“安装 MSDN（I）”复选框，不安装帮助文档。本例选择不安装帮助文档，单击“下一步”按钮。


图 1-19　“安装完成”界面

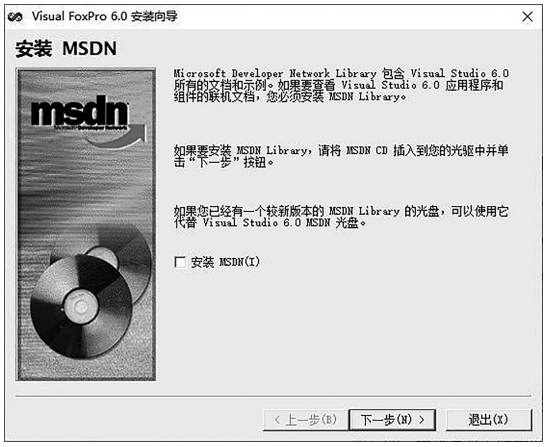



图 1-20　选择是否安装帮助文档

步骤 10　在打开如图 1-21 所示的“不安装 MSDN”对话框中单击“是”按钮不安装帮助文档，这时会打开如图 1-22 所示的 Web 注册界面，本例取消“现在注册”复选框，先选择不进行网上注册，然后单击“完成”按钮完成 Visual FoxPro 的安装。


图 1-21　“不安装 MSDN”对话框


图 1-22　完成 Visual FoxPro 的安装

## 思考练习

一、选择题

1. 数据库系统中对数据库进行管理的核心软件是_____。

A. DBMS　　B. DB　　C. OS　　D. DBS

2. Visual FoxPro 是一种关系型数据库管理系统，这里的关系通常是指_____。

A. 数据库文件（.dbc 文件）　　B. 一个数据库中的两个表之间有一定的关系

C. 表文件（.dbf 文件）　　D. 一个表文件中的两条记录之间有一定的关系

3. 如果一个班中只能有一个班长，而且一个班的班长不能同时担任其他班的班长，则班长和班级两个实体之间的关系属于_____。

A. 一对一联系　　B. 一对二联系　　C. 多对多联系　　D. 一对多联系

4. 采用二维表结构来表示实体及实体之间联系的模型被称为_____。

A. 层次模型　　B. 关系模型　　C. 网状模型　　D. 链状模型

5. 数据库系统与文件系统的主要区别是_____。

A. 数据库系统复杂，而文件系统简单

B. 文件系统不能解决数据冗余和数据独立性问题，而数据库系统可以

C. 文件系统只能管理程序文件，而数据库系统能够管理各种类型的文件

D. 文件系统管理的数据量较少，而数据库系统可以管理庞大的数据量

6. 在 Visual FoxPro 6.0 中，以下的_____菜单用于设置工具栏的相关功能。

A. 显示　　B. 文件　　C. 工具　　D. 窗口

二、填空题

1. 对关系进行选择、投影或连接运算之后，运算的结果仍然是_____。

2. 在关系数据模型中，属性的取值范围称为属性的_____。

3. 在奥运会游泳比赛中，一个游泳运动员可以参加多项比赛，一个游泳比赛项目可以有多个运动员参加，游泳运动员与游泳比赛项目两个实体之间的联系是_____联系。

4. 自然联接是去掉重复属性的_____联接。

# 项目二 Visual FoxPro 基本元素

## 项目概述

学习数据库应用技术是为了今后应用和开发数据库系统，除了需要表中的数据之外，还经常要处理其他数据。Visual FoxPro 包含常量、变量、表达式和函数四种形式的数据，常量和变量是数据处理的基本对象，表达式和函数则体现了 Visual FoxPro 对数据的处理能力。本项目我们将学习 Visual FoxPro 基本元素，这些都是编写程序时要用到的基本内容。

## 学习目标

◎知识目标

1. 了解 Visual FoxPro 数据类型。
2. 掌握常量、变量、运算符和函数的使用方法。

◎能力目标

1. 能对变量进行定义、赋值、访问、显示、释放等操作。
2. 可以利用函数进行数据计算、各种类型数据的转换和数据库的处理。

◎素质目标

1. 提升观察能力，提高专业素质。
2. 培养专注、专心、负责的工作态度。

## 任务一　数据类型

数据是反映客观事物属性的记录，每一个数据都有自己的类型，数据类型决定了数据的存储方式和运算方式。在 Visual FoxPro 中，数据类型一旦被定义，就确定了其存储方式和使用方法。Visual FoxPro 系统为了使用户建立和使用数据库更加方便，将数据划分为以下六种类型。

### 一、数值型

数值型数据是描述数量的常用数据类型，用来表示一个数量的大小，通常可分为数值型、浮点型、双精度型、整型和货币型五种类型。

数值型（numeric）是由数字（0~9）、小数点和正负号组成的，最大长度为 20 位（包括正负号和小数点在内）。其允许用户进行各种数学运算。

浮点型（float）和数值型数据完全等价，只是在存储形式上采取浮点格式。

双精度型（double）是更高精度的数值型数据，仅用于表中的数值型字段定义。双精度型数据的长度为 8 位。

整型（integer）是不包含小数部分的数值型数据。它只用于数据表中的字段类型定义，整型数据的长度为 4 位。

货币型（currency）数据用来表示货币值，用来存储与货币有关的数据。在货币表达式中，如果指定值的小数部分超过 4 位，Visual FoxPro 将对其进行四舍五入，使其保持在 4 位。货币型数据通常用 $ 表示。

### 二、字符型

字符型（character）数据是由任意字符（字母、数字、空格、符号等）组成的一个字符串，长度范围是 0~254 个字符，使用时必须用定界符双引号“""”,单引号“''”或者中括号“[ ]”括起来。

**注意：**

这里的单引号、双引号、方括号都是西文的标点符号。

### 三、日期型和日期时间型

日期型（date）数据是用来表示日期的数据类型，其默认的显示格式为：mm/dd/yy。mm 表示月，dd 表示日，yy 表示年。日期型数据长度固定为 8 位。日期型的使用格式为 {^yyyy-mm-dd}，该格式在书写时要注意，“^” 不可省略；年份必须用 4 位（如 1998、2017 等）；年月日的次序不能颠倒、不能省略。

日期时间型（datetime）数据是用来表示日期和时间的数据类型，默认格式为 mm/dd/yy hh：mm：ss。其中，前半部分表示日期，与日期型相同；后半部分表示时间，hh 表示小时，mm 表示分，ss 表示秒。日期时间型数据类型长度为 8 位。

### 四、逻辑型

逻辑型（logical）数据用于描述真与假，用来表示逻辑判断的结果。逻辑型数值只包含两个值：真（.T.）、假（.F.）。逻辑型数据类型长度为 1 字节。

### 五、备注型

备注型（memo）数据用于创建表时定义字段类型，用来存放长度不等的文本数据。这些文本数据既可以是短小的句子，也可以是较长的段落。其实，备注型字段并不直接存放在文本本身，而是存放指向这些文本的指针。文本的内容则存放在由系统自动生成的与数据库同名、扩展名为 .dct 的备注文件中，系统规定备注型数据长度固定为 4 位。

### 六、通用型

通用型（general）数据仅用于表中的字段类型定义，它实际上是一种特殊的备注型数据。系统规定它的固定长度为 4 位。通用型数据用于存储指向一个 OLE 对象的指针，如图像、声音和电子表格等，这些对象一般由其他应用程序创建。

## 任务二　常量和变量

数据在加工处理当中是需要保存的，否则所做的处理将会丢失。在 Visual FoxPro 中，数据一般保存

在常量、变量及数组中。除此之外，Visual FoxPro 系统还可将数据保存在字段、记录和对象中。

## 一、常量

常量是指在程序运行过程中其值不能被改变的量。在 Visual FoxPro 中，常量分为数值型、货币型、字符型、逻辑型、日期型和日期时间型等几种类型。

### （一）数值型常量

数值型常量也称为“常数”，用来表示一个数的大小，数值型常量由数字（0~9）、小数点和正负号组成，如 3，-12.3 等，为了方便表示很大或很小的数，可以用科学记数法形式表示，如 1.6E -8 表示 $1.6\times10^{-8}$。数值型常量的长度为 8 个字节。

### （二）货币型常量

货币型常量用来表示货币，以符号“$”开头，并四舍五入到小数点后 4 位。例如，货币型常量 $1 387.342 19，计算结果为 $1 387.342 2。

### （三）字符型常量

字符型常量是用定界符括起来的字符串，定界符为西文的单引号、双引号和方括号。定界符不作为字符串本身的内容，如“Hello”，“241000”。

字符型常量的定界符必须成对匹配。如果某种定界符本身是字符串的内容，则需要用另一种定界符为该字符串定界。

### （四）逻辑型常量

逻辑型常量只有真和假两个值，分别用 .T.、.t.、.Y.、.y. 和 .F.、.f.、.N.、.n. 来表示。

### （五）日期型常量和日期时间型常量

日期型常量必须用花括号括起来，花括号里包括年、月、日三部分内容，各部分之间用分隔符分隔，系统默认分隔符为斜杠“/”，其他的日期分隔符有连字号“-”、句点“.”和空格。其中，月、日为两位数，年份可以为两位数也可以为四位数。空白的日期可以表示为 {} 或者 {/}。

日期时间型常量的写法与日期常量类似，空白的日期时间型可以表示为 {/:}。

严格的日期时间型格式为：{^yyyy-mm-dd [,] [hh [: mm [: ss] ] [a | p] ] }。其中，符号“^”表明该日期格式是严格的，并按照 YMD 的格式来解释日期和日期时间；其中的“-”可用斜杠代替。

在 Visual FoxPro 中，如果要使用通常的日期格式，必须先执行“SET STRICTDATE TO 0”命令；如果要设置严格的日期格式，可执行“SET STRICTDATE TO 1”命令。

## 二、变量

变量是指在命令操作和程序运行过程中其值允许发生变化的量。变量在内存中占据一定的存储单元，在该存储单元中存放着变量的值，每一个变量都有一个名称，称为变量名，Visual FoxPro 通过变量名来使用变量。在 Visual FoxPro 中，变量包括内存变量、字段变量。

### （一）内存变量

#### 1. 内存变量的命名规则

在 Visual FoxPro 中，变量命名须遵守下列规则：变量名以字母、数字、下画线组成，且第一个字符必须是字母或下画线。其长度为 0~128 个字符。变量名不能与系统关键字重名。变量名中字母不区分大小写。

#### 2. 内存变量赋值命令

【格式 1】<内存变量>=<表达式>

【功能】先计算<表达式>的值，然后将该值赋给内存变量。

【格式 2】STORE<表达式>TO<内存变量表>

【功能】先计算<表达式>的值，然后将该值赋给内存变量。

【例 2-1】使用“=”给内存变量赋值

```
value=156                    &&value 为一个值为 156 的整型变量
str1="Hello"                 &&str1 为一个值为 Hello 的字符串
flag=.T.                     &&flag 为一个值为 .T. 的逻辑型变量
```

【例 2-2】使用 STORE 命令给内存变量赋值

```
B=9                          && 定义变量 B 并赋值
store B+3 to C               && 定义变量 C 并赋值
store"Visual FoxPro"to A, B, C   && 同时定义变量 A, B, C 并为其赋相同值
```

【说明】

（1）命令后的“&&”符号表示该符号后跟随的是对该命令的注解，起说明的作用，与命令的执行无关。

（2）内存变量在赋值时定义了它的值和类型，其类型与所附的值的类型相同。

（3）STORE 命令的“内存变量表”可以包含多个内存变量，但必须用逗号分隔。

（4）STORE 命令可以将同一个值一次性赋给多个内存变量，而“=”命令每次只能为一个内存变量赋值。

### 3. 内存变量的输出

【格式】? | ?? <表达式表>

【功能】计算表达式的值并将其输出到屏幕上。

【例 2-3】使用? | ?? 输出内存变量。

```
B="Visual  FoxPro"           && 定义变量 B 并赋值。
? B                          && 在窗口中输出“Visual FoxPro”
??"数据库"                   && 紧接在“Visual FoxPro”后面输出字符串“数据库”
?"S=", B                     && 换一行显示“S=Visual FoxPro”
```

【说明】

“?”表示换行显示结果；“??”表示从当前行开始显示结果。“表达式表”可以用逗号来隔开多个表达式，命令执行时遇逗号就空一格。

### 4. 内存变量的显示

【格式】LIST | DISPLAY MEMORY [LIKE<通配符>] [TO PRINTER [PROMPT] [TO FILE<文件名>]

【功能】显示当前已定义的内存变量名、作用范围、类型和值。

【例 2-4】显示内存变量中以 A 开头的所有变量。

```
A1="数据库"
Store 20 to A1000
listmemory like A*
```

结果显示：

```
A        Pub    C    "数据库"
A100     Pub    N    20
```

【说明】

（1）LIKE 子句表示显示与通配符匹配的内存变量。

（2）默认 LIKE 选项则显示全部内存变量，并同时显示当前内存变量的总数、字节数等。

（3）选项 TO PRINTER 能将屏幕显示内容输出到打印机，[PROMPT] 选项可以提供是否打印的提示

窗口，选项 TO FILE<文件名>能将显示内容存入文件。

### 5. 内存变量的释放

【格式】RELEASE［<内存变量表>］［ALL［ LIKE ｜ EXCEPT<通配符>］］

【功能】从内存中清除指定的内存变量。

【格式】CLEAR ALL

【功能】将内存中所有变量都释放。

【例 2-5】清除指定内存变量。

```
release a，b                    && 释放变量 a，b
releaseall like abc?           && 释放所有变量名为 4 个字符且前 3 个字符为 abc 的所有变量
release all except abc *       && 释放所有除了变量名前 3 个字符为 abc 以外的所有变量
```

### （二）字段变量

字段变量是数据库中定义的变量，每个数据表中都包含若干个字段变量。字段变量在建立表结构时定义。在数据表中，同一个字段名下的数据值是不同的。

在数据表中，对字段必须先定义再使用。对字段变量的定义主要给出变量名、变量类型、变量宽度以及数值型数据的小数位数等。变量名由汉字、字母、数字和下画线组成，而且必须以汉字或字母开头。字段变量的类型有字符型、数值型、浮点型、双精度型、整型、逻辑型、日期型、日期时间型、备注型和通用型等。

## 三、数组

数组是一组有序数据的集合，其中的每个数据称为数组元素，为了区分不同的数组元素，每个数组元素可以通过数组名和相应的下标来表示。每个数组元素相当于一个简单内存变量，可以给每个元素单独赋值，对每个元素分别访问。

### （一）数组的定义

数组在使用之前必须使用 DIMENSION 或者 DECLARE 进行定义。

【格式】DIMENSION ｜ DECLARE<数组名>（<下标上限 1）>［，<下标上限 2>）］［，<数组名>（<下标上限 1）>［，<下标上限 2>）］］…

【说明】这两条命令功能相同，用户可按自己习惯选用。

【例 2-6】一维数组的定义。

```
declare A(7)                   && 定义了一个包含 7 个元素的一维数组
```

【例 2-7】二维数组的定义。

```
dimension B(2,3)               && 定义了一个包含 6 个元素的二维数组
```

在例 2-7 中，定义了一个包含 6 个元素的二维数组，数组元素分别为：B(1,1)，B(1,2)，B(1,3)，B(2,1)，B(2,2)，B(2,3)，数组 B 是一个二维数组。

### （二）数组元素的赋值和使用

数组实际上就是一组带下标的变量，使用时与一般的变量的使用方法一样，可以对其赋值。如 A（4）＝6 即表示将 A（4）数组元素变量赋值为 6。

【例 2-8】定义一个包含 6 个元素的一维数组 B，给所有数组元素赋值并输出。

```
Dimension B(6)
B(1)= 2
B(2)= 7
store 6 to B(3),B(4),B(5),B(6)
? B(1),B(2),B(3),B(4),B(5),B(6)
```

输出结果：

```
2  7  6  6  6  6
```

【说明】

（1）在 Visual FoxPro 中，同一数组中的各个数组元素类型可以相同，也可以不同，而且还可以通过重新赋值方式来改变数组元素的类型。在定义数组时，若没有给数组元素赋值，系统默认数组元素为 . F. 。

（2）用赋值命令可以对单个数组元素赋值，如例 2-8，也可以利用直接给数组名赋值的方法一次性对所有数组元素赋值。如“B=9”，表示将数组 B 中所有元素的值都赋为 9。

（3）在同一运行环境下，数组名不能和变量名重复。

（4）二维数组在内存中按行的顺序存储，但也可以按一维数组的形式来表示其数组元素，如对于例 2-7中定义的数组 B，B(2，2）排在第 2 行第 2 列，由于每行 3 个元素，所以该数组元素也可用B（5）来表示。

# 任务三　运算符和表达式

运算是对数据进行加工的过程，描述不同运算的符号被称为运算符，参与运算的数据被称为操作数。表达式用来表示某个求值规则，它是由常量、变量和函数通过特定运算符连接起来的公式。

Visual FoxPro 的运算符包括算术运算符、字符运算符、日期时间运算符、关系运算符、逻辑运算符。根据运算符的类型，表达式可分为数值表达式、字符表达式、日期时间表达式、关系表达式和逻辑表达式。

## 一、算术运算符和算术表达式

算术表达式由算术运算符、数值型常量、变量、函数和圆括号组成，算术表达式的运算结果是一个数值。如 2^3 的值为 8。34+9 * 5-14%4 的值为 77。Visual FoxPro 的算术运算符及优先级见表 2-1 所列。

表 2-1　Visual FoxPro 的算术运算符及优先级

| 优先级 | 运算符 | 运算符说明 |
|---|---|---|
| 1 | ( ) | 圆括号括起来的表达式优先级最高，若有圆括号嵌套，按照先内后外顺序运算 |
| 2 | ^或 * * | 同数学中的乘方，如 2^3 表示 $2^3$ |
| 3 | * 、/、% | 依次为乘运算、除运算、取余数运算 |
| 4 | +、- | 加运算、减运算 |

## 二、字符运算符和字符表达式

字符表达式由字符运算符将字符数据连接起来组成，字符表达式的运算结果是字符型数据。如“hello” + “world” 的结果是“hello world”。在 Visual FoxPro 中字符运算符有两个，它们的运算优先级相同。Visual FoxPro 的字符运算符及优先级见表 2-2 所列。

表 2-2　Visual FoxPro 的字符运算符及优先级

| 运算符 | 说明 |
|---|---|
| + | 将前后两个字符串首尾连接成一个新字符串 |
| - | 连接前后两个字符串，并将前字符串的尾部空格移到合并后的新字符串尾部 |

【例 2-9】字符串运算示例。

```
X="Hello      "
Y="World      "
? X+Y, X-Y
```

输出结果：

```
Hello      World      HelloWorld
```

用户在字符表达式中还可以使用宏代换函数。

【格式】&<字符型内存变量>[. 字符表达式]

【功能】替换一个字符型内存变量的内容。若字符型内存变量与后面的字符表达式无空格分隔，则 & 函数后的“.”不能省略。例如：

```
m="4*3^2+5"
? m
? &m
```

第一个输出结果为：4*3^2+5（即字符串 m）。第二个的输出结果为：41.00（即字符串 4*3^2+5 的值）。

可使用宏替换实现内存变量换名，这是宏替换函数中的“.”字符表达式的一种用法。在使用宏替换时，将连字符“&”放在变量前，告诉 Visual FoxPro 将此变量值当作名称使用，并使用“.”来结束这个宏替换表达式。

例如，下面的语句：

```
x="Fox"
"&x. Pro"
```

输出结果为"FoxPro"

## 三、日期时间运算符和日期时间表达式

日期时间表达式由日期时间运算符（+和-）、数值表达式、日期型常量、日期型变量和函数组成。只能对日期型数据进行加减运算，其中，“+”表示一个日期型数据和一个整数之间的运算，“-”表示日期与日期、日期与整数间的运算。在 Visual FoxPro 中，日期型数据不能相加，若出现无效日期，Visual FoxPro 会将其处理成空日期。常见的日期时间表达式见表 2-3 所列。

表 2-3　日期时间表达式

| 格式 | 结果及类型 |
|---|---|
| <日期>+<天数> | 指定日期若干天后的日期。日期型 |
| <天数>+<日期> | 指定日期若干天后的日期。日期型 |
| <日期>-<天数> | 指定日期若干天前的日期。日期型 |
| <日期>-<日期> | 两个指定日期相差的天数。数值型 |
| <日期时间>+<秒数> | 指定日期时间若干秒后的日期时间。日期时间型 |
| <日期时间>-<秒数> | 指定日期时间若干秒前的日期时间。日期时间型 |
| <日期时间>-<日期时间> | 两个指定日期时间相差的秒数。数值型 |

【例 2-10】日期时间运算实例。

```
? {^2023-5-20}  +12
```

结果为 06/01/13，表示 2023 年 5 月 20 日后 12 天为 2023 年 6 月 1 日。

```
? {^2023-5-20 10:12:12 am} - {^2023-5-20 9:12:12 am}
```

结果为 3 600，表示从 2023 年 5 月 20 日 10：12：12am 到 2023 年 5 月 20 日 9：12：12 am 是 3 600 秒。

## 四、关系运算符和关系表达式

关系运算符由关系运算符和字符表达式、算术表达式、日期时间表达式组成，用于相同类型数据之间的比较，其运算结果为逻辑型常量。当关系成立时，表达式的结果为 . T. ；当关系不成立时，表达式的结果为 . F. 。关系运算符的含义及实例说明见表 2-4 所列。

【格式】<表达式 1><关系运算符><表达式 2>。

表 2-4　关系运算符

| 运算符 | 含义 | 实例 | 说明 |
|---|---|---|---|
| > | 大于 | $a>5$ | $a$ 大于 5 结果为真，否则为假 |
| >= | 大于或等于 | $a>=12$ | $a$ 大于或等于 12 结果为真，否则为假 |
| < | 小于 | $a<4$ | $a<4$ 结果为真，否则为假 |
| <= | 小于或等于 | $a<=20$ | $a<=20$ 结果为真，否则为假 |
| = | 等于 | $a=b$ | $a$ 等于 $b$ 结果为真，否则为假 |
| == | 绝对等于 | $a==b$ | $a$ 绝对等于 $b$ 结果为真，否则为假 |
| <>、#或！= | 不等于 | $a<>b$ | $a$ 不等于 $b$ 结果为真，否则为假 |
| $ | 子串包含测试 | $a \$ b$ | $a$ 是 $b$ 的子串结果为真，否则为假 |

在进行比较运算时，应注意以下几点。

· 除了日期和日期时间型数据可以比较外，前、后两个运算对象的数据类型必须一致。

· 数值型数据和货币型数据比较是按数值大小比较的，包括符号。例如，3>5 结果为假，10>-12 结果为真，$10<=$20 结果为真。

· 日期或日期时间型数据在比较时，越晚的时间越大。如 {^2013-5-12} > {^2010-12-9} 结果为真。

· 逻辑真大于逻辑假。如 . T. < . F. 结果为假。

· 运算符 $ 和==仅适用于字符型数据，其他运算符适用于任何类型的数据。

在用“==”比较两个字符串时，只有当两个字符串完全相同（包括空格以及各字符的位置）时，运算结果才为真，否则为假。

· 字符串精确比较和 EXACT 设置。

在用“=”比较两个字符串时，结果与 SET EXACT ON/OFF 设置有关，该命令是设置精确匹配与否的开关。系统默认 OFF 状态，当处于 OFF 状态时，只要右边的字符串与左边的字符串的前面部分匹配即为逻辑真。也就是说，字符串的比较以右边的字符为目标，右字符串结束即终止比较。

当处于 ON 状态时，比较到两个字符串全部结束，先在较短字符串的尾部加上空格，使两个字符串长度相等；然后进行比较。

【例 2-11】在 EXACT OFF 状态下的比较。

```
set exact off
store"Visual FoxPro"to S1
store"Visual FoxPro"to S2
store "Visual FoxPro 程序设计"to S3
? S1=S3 , S3=S1, S1=S2 , S2=S1, S2==S1
```

主屏幕显示 . F. . T. . F. . T. . F.

【例 2-12】在 EXACT ON 状态下的比较。

```
set exact on
store"Visual FoxPro"to S1
store"Visual FoxPro"to S2
store"Visual FoxPro 程序设计"to S3
? S1=S3 , S3=S1, S1=S2, S2=S1, S2==S1
```

主屏幕显示 . F. . F. . T. . T. . F.

除了可以用 SET EXACT ON | OFF 设置精确匹配以外，用户还可以通过选择“工具”→“选项”命令，利用对话框中的“数据”选项卡来设置精确匹配。

## 五、逻辑运算符和逻辑表达式

逻辑运算包括逻辑与（. AND. ）、逻辑或（. OR. ）、和逻辑非（. NOT. ）三种运算符。在书写时，前后两个点可以省略。

逻辑表达式由逻辑运算符和逻辑型常量、逻辑型变量、返回逻辑型数据的函数以及关系表达式组成。其结果仍为逻辑型常量。

设 A、B 为两个逻辑型变量，它们之间的运算规则见表 2-5 所列。

表 2-5　逻辑运算符运算规则

| A | B | NOT A | A AND B | A OR B |
|---|---|---|---|---|
| . T. | . T. | . F. | . T. | . T. |
| . T. | . F. | . F. | . F. | . T. |
| . F. | . T. | . T. | . F. | . T. |
| . F. | . F. | . T. | . F. | . F. |

【例 2-13】逻辑运算表达式示例。

假设有“教师信息表”数据表（见表 2-6 所列）如下。

表 2-6　例 2-13 数据表

| 工号 | 姓名 | 职称 | 所在系 | 工资/元 | 照片 | 备注 |
|---|---|---|---|---|---|---|
| 10010 | 杨× | 副教授 | 计算机系 | 9 500 | | |
| 10011 | 朱× | 副教授 | 计算机系 | 9 600 | | |
| 10031 | 刘× | 教授 | 软件工程系 | 9 300 | | |
| 10040 | 王×勇 | 讲师 | 通信工程系 | 8 000 | | |
| 10041 | 王×荣 | 副教授 | 通信工程系 | 8 400 | | |
| 10032 | 许× | 讲师 | 软件工程系 | 8 100 | | |
| 10033 | 林× | 副教授 | 软件工程系 | 8 300 | | |
| 10012 | 沈× | 讲师 | 计算机系 | 8 800 | | |

在表 2-6 中查找职称为“副教授”，所在系为“计算机系”或者“软件工程系”，工资在 8 200 元以上的记录，则条件表达式应写成：

（职称="副教授"）AND（所在系="计算机系"OR 所在系="软件工程系"）AND 工资>8 200

## 六、运算符的优先级

表达式的运算不一定都是从左到右进行的，不同的运算在表达式中有不同的优先运算顺序。表 2-7 按先后顺序列出了各类运算符的优先级。另外，在表达式中可以使用圆括号（可嵌套使用）来改变表达式的优先运算顺序，并提高程序的可读性。

表 2-7　运算符的优先级

| 类别 | 运算符 |
| --- | --- |
| 算术运算符 | ^或 * * |
| | * 、/、% |
| | +、- |
| 关系运算符 | <、<=、>、>=、=、==、<>或#或！=、$ |
| 逻辑运算符 | NOT |
| | AND |
| | OR |

在一个表达式中有不同的运算符时，表达式遵循以下运算规则。

· 优先级高的运算符先运算。

· 相同级别的运算符遵循从左至右的顺序进行运算。

· 字符串连接运算符和算术加、减运算符优先级相同。

· 可以用括号改变运算符的优先顺序。使用括号后遵循先内后外的原则。

# 任务四　常用函数

所谓函数就是具有特定功能的子程序，每个函数都有特定的功能。函数往往需要若干自变量，即运算对象，但只能有一个运算结果，称为函数值或返回值。函数类型就是返回值类型。在使用函数时，用户必须了解函数的返回值类型，以免发生数据类型不匹配的错误。函数可以用函数名加一对圆括号加以调用，自变量放在圆括号里。

【格式】函数名（<自变量列表>）

其中，函数名是系统规定的，自变量表可以是一个变量或多个变量，也可以为空，即：函数名（），此时函数名后面的括号不可省略。

函数有函数名、参数和函数值三个要素。

· 函数名起标识作用。

· 参数是自变量，写在括号内，有的函数参数可以默认。

· 函数运算后返回的值称为函数值。

Visual FoxPro 提供了大量函数供用户使用。这些函数大致可以分为数值函数、字符函数、日期时间函数、数据类型转换函数、测试函数等。

## 一、数值函数

数值函数是指函数值为数值的一类函数，它们的自变量和返回值往往都是数值型的数据。

### （一）圆周率函数

【格式】PI（）

【功能】返回圆周率的数值（数值型）。

**注意：**

PI 函数无参数。

### （二）取整函数

【格式】INT（<数值表达式>）

【功能】返回值等数值表达式的整数部分。例如：

```
store 35.667 to A
? int (A), int (-A)
```

结果为：

```
35-35
```

### （三）求绝对值函数

【格式】ABS（<数值表达式>）

【功能】返回指定数值表达式的绝对值。例如：

```
? abs (-9)
```

结果为：

```
9
```

### （四）四舍五入函数

【格式】ROUND（<数值表达式>，<保留小数位>）

【功能】按保留小数位指定的位数对数值表达式数值进行四舍五入运算。例如：

```
? round (3.14159, 1), round (3.14159, 2), round (3.14159, 3)
```

结果为：

```
3.1      3.14      3.142
```

### （五）求余函数

【格式】MOD（<数值表达式 1>，<数值表达式 2>）

【功能】取数值表达式 1 除以数值表达式 2 的余数。例如：

```
? mod (18, 4), mod (18, -4), mod (-18, 4), mod (-18, -4)
```

结果为：

```
2      -2      2      -2
```

### （六）求平方根函数

【格式】SQRT（<数值表达式>）

【功能】返回数值表达式的平方根。例如：

```
? sqrt (16)
```

结果为：

```
4
```

### （七）求最大、最小值函数

【格式】MAX | MIN（<表达式 1>，<表达式 2>，<表达式 3>，…，<表达式 *n*>）

【功能】求<表达式 1>，<表达式 2>，<表达式 3>，…，<表达式 *n*>中的最大值 | 最小值。例如：

```
? max (3, 12, 8, 15), min (3, 12, 8, 15)
```

结果为：

```
15      3
```

## 二、字符函数

字符处理函数是实现文字处理的重要方法，其自变量一般是字符型数据。

### （一）求字符串长度函数

【格式】LEN（<字符串表达式>）

【功能】求字符串的长度，即字符串所包含的字符个数。若是空串，则长度为 0。函数值为数值型。例如：

```
? len（“Visual FoxPro 数据库管理系统”）
```

结果为：

```
27
```

### （二）删除字符串前后空格函数

【格式】RTRIM | LTRIM | ALLTRIM（<字符串表达式>）

【功能】RTRIM（）删除字符串的尾部空格，LTRIM（）删除字符串的前导空格，ALLTRIM（）删除字符串中的前导和尾部空格。例如：

```
S1="Hello"
S2="World"
? len（S1），len（ltrim（S1））， len（S2），len（ltrim（S2）），len（alltrim（S2））
```

结果为：

```
8  7  13  7  5
```

### （三）生成空格函数

【格式】SPACE（<数值型表达式>）

【功能】生成若干个空格，空格的个数由数值型表达式的值决定。例如：

```
? len（space（8）-space（3））
```

结果为：

```
11
```

### （四）求子串位置函数

【格式】AT（<字符串表达式 1>，<字符串表达式 2>［，<数值表达式>］）

【功能】如果字符串表达式 1 是字符串表达式 2 的子串，则返回字符串表达式 1 在字符串表达式 2 中的起始位置；若不是子串，则返回 0。AT（）的函数值为数值型，ATC（）与 AT（）功能类似，但在子串比较时不区分字母大小写。“数值表达式”用于指明搜索字符串表达式 1 在字符串表达式 2 中是第几次出现的，其默认值是 1。例如：

```
store"This is Visual FoxPro"tox
? at（"FOX"，x），at（"is"，x，3），at（"xo"，x）
```

结果为：

```
0  10  0
```

### （五）取子串函数

【格式】LEFT | RIGHT（<字符串表达式>，<数值表达式>）

SUBSTR（<字符串表达式>，<起始位置>［，<数值表达式>］）

【功能】LEFT（）从字符串表达式左边第一个字符开始截取子串。RIGHT（）从字符串表达式右边第一个字符开始截取子串，其子串的长度由“数值表达式”的值所决定。若数值表达式的值大于字符串的长度，则给出整个字符串。SUBSTR（）是从字符串表达式中取出从指定“起始位置”开始截取的子串，子串的长度由数值表达式的值所决定。若省略数值表达式，则截取的子串从“起始位置”开始到字符串表达式的最后一个字符；若“起始位置”或数值表达式为 0，则函数值为空串。显然，SUBSTR 函数可以代替 LEFT 函数和 RIGHT 函数的功能。例如：

```
store"GOOD BYE !"to X
    ? left（X，2），substr（X，6，2）+substr（X，6），right（X，3）
```

结果为：

```
GO   BYBYE!      YE!
```

### （六）大小写转换函数

【格式】LOWER | UPPER（<字符表达式>）

【功能】LOWER（）将字符表达式值中的大写字母转换成小写字母；UPPER（）将字符表达式值中的小写字母转换成大写字母。例如：

```
? lower ("HELLO WORLD"), upper ("hello world")
```

结果为：

```
hello world   HELLO WORLD
```

### （七）宏替换函数

【格式】&<字符型内存变量>[.]

【功能】替换出字符型变量的内容，即 & 的值是变量中的字符串。若字符型内存变量与后面的字符无空格分隔，则 & 函数后的”.”不能省略。宏替换可以嵌套使用。例如：

```
m="32 * sqrt (4)"
? Int (34.15+&m)
```

结果为：

```
98
i='1'
j='2'
x12="made in China"
? x&i.&j
```

结果为：

```
made in China
```

### （八）子串替换函数

【格式】STUFF（<字符表达式 1>，<起始位置>，<长度>，<字符表达式 2>）

【功能】用字符表达式 2 的值替换字符表达式 1 中自“起始位置”起由“长度”确定的一个子串，替换和被替换的字符个数不一定相等。如果“长度”值为 0，字符串表达式 2 则插在由“起始位置”指定的字符前面。如果字符表达式 2 的值是空串，那么字符表达式 1 中由“起始位置”和“长度”指明的子串将被删去。例如：

```
store"HE IS SO CLEVER"to S1
store"SHE"to S2
store"BRIGHT"to S3
? stuff (S1, 1, 3, S2), stuff (S1, 10, 6, S3)
```

结果为：

```
SHEIS SO CLEVER   HE IS SO BRIGHT
```

## 三、日期时间函数

日期时间函数的自变量一般是日期型数据或日期时间型数据。

### （一）系统日期和系统时间函数

【格式】DATE（）
　　TIME（）
　　DATETIME（）

【功能】DATE（）返回当前系统日期，函数值为日期型。TIME（）以 24 小时制返回当前系统时间，函数值为字符型。DATETIME（）返回当前系统日期时间，函数值为日期时间型。

### （二）年份、月份和天数函数

【格式】YEAR（<日期表达式>| <日期时间表达式>）
MONTH（<日期表达式>| <日期时间表达式>）
DAY（<日期表达式>| <日期时间表达式>）

【功能】YEAR（）从指定的日期表达式或日期时间表达式中返回年份，以 4 位数值型数据表示（如 2017 年）。MONTH（）从指定的日期表达式或日期时间表达式中返回月份。DAY（）从指定的日期表达式或日期时间表达式中返回月里面的天数。这 3 个函数的返回值都为数值型。例如：

```
store {^2023-10-6} to d
? year (d), month (d), day (d)
```

结果为：

```
2023  10  6
```

### （三）时分秒函数

【格式】HOUR（<日期时间表达式>）
MINUTE（<日期时间表达式>）
SEC（<日期时间表达式>）

【功能】HOUR（）从指定的时间日期表达式中返回小时部分（24 小时制）。
MINUTE（）从指定的时间日期表达式中返回分钟部分。
SEC（）从指定的时间日期表达式中返回秒数部分。例如：

```
store {^2023-10-5 10: 25: 20 } to x
? hour (x), minute (x), sec (x)
```

结果为：

```
10  25  20
```

常用的日期时间函数见表 2-8 所列。

表 2-8　常用的日期时间函数一览表

| 函数名、格式及说明 | 输出类型 | 例子 |
|---|---|---|
| TIME（）<br>返回当前系统时间 | C | ? time（） |
| DATE（）<br>返回当前系统日期 | D | ? date（） |
| YEAR（<日期表达式>）<br>返回日期表达式的年份的数值 | N | ? year（date（））<br>输出：2024 |
| MONTH（<日期表达式>）<br>返回日期表达式的月份的数值 | N | ? month（date（））<br>输出：4 |
| DAY（<日期表达式>）<br>返回日期表达式的日期的数值 | N | ? day（date（））<br>输出：28 |
| HOUR（<日期时间表达式>）<br>返回日期时间表达式的小时的数值（24 小时制） | N | ? hour（{^2023-10-06 12：14：34}）<br>输出：12 |
| MINUTE（<日期时间表达式>）<br>返回日期时间表达式的分钟的数值 | N | ? minute（{^2023-10-06 12：14：34}）<br>输出：14 |
| SEC（<日期时间表达式>）<br>返回日期时间表达式的秒钟的数值 | N | ? sec（{-2023-10-06 12：14：34}）<br>输出：34 |

## 四、数据类型转换函数

数据类型转换函数的功能是将某一种类型的数据转换成另一种类型的数据。

### （一）字符串转换为数值函数

【格式】VAL（<数字字符串>）

【功能】将数字字符串（包括正、负号，小数点）转换为对应的数值型数据。若字符串内出现非数字，字符就停止转换；若字符串的首字符为非数字符，则返回值为 0，但忽略前导空格。例如：

```
? val ("X12.8"), val ("-35.875"), val ("3A.79")
```

结果为：

```
0.00      -35.88      3.00
```

### （二）数值转换成字符串函数

【格式】STR（<数值型表达式>［，<长度>，［，<小数位数>］］）

【功能】将<数值型表达式>的值转换成字符串，转换时根据需要自动四舍五入。转换后字符串的长度 *L* 应该是数值型表达式值的整数部分的位数加上小数位数值，再加上一位小数点。如果“长度”值大于 *L*，则字符串加前导空格以满足规定的长度要求；如果“长度”值大于、等于数值型表达式值的整数部分位数（包括负号）但又小于 L，则优先满足整数部分而自动调整小数位数；如果“长度”值小于数值型表达式值的整数部分位数，则返回一串星号（*）。“小数位数”的默认值为 0，“长度”的默认值为 10。例如：

```
store -348.387 TO P
?"P="+str (P, 8, 3)
```

结果为：

```
P=-348.387
? str (P, 9, 2), str (P, 6, 2), str (P, 3), str (P, 6), str (P), str (P, 10, 4)
```

结果为：

```
□□-348.39   -348.4 * * *   □□-348□□□□□□-348
□-348.3870   &&□表示空格
```

### （三）字符串转换成日期或日期时间函数

【格式】CTOD（<字符型表达式>）

　　CTOT（<字符型表达式>）

【功能】CTOD（）将字符型表达式值转换成日期型数据。CTOT（）将字符型表达式值转换成日期时间型数据。

### （四）日期或日期时间转换成字符串

【格式】DTOC（<日期表达式>| <日期时间表达式>［，1］）

　　TTOC（<日期时间表达式>［，1］）

【功能】DTOC（）将日期表达式或日期时间表达式的日期部分转换成字符串。TTOC（）将日期时间表达式转换成字符串。

字符串中日期部分的格式与 SET DATE TO 语句的设置和 SET CENTURY ON | OFF（ON 为 4 位数年份，OFF 为 2 位数年份）语句的设置有关。时间部分的格式受 SET HOURS TO 12 | 24 语句的设置影响。

对 DTOC（）来说，如果使用选项“1”，则字符串的格式总是为 YYYYMMDD，共 8 个字符；对 TTOC（）来说，如果使用选项“1”，则字符串的格式总是为 YYYYMMDDHHMMSS，采用 24 小时制，共 14 个字符。例如：

```
storedatetime ( ) to t
? t
```

结果为：

```
10/06/2023 10：07：09pm
? dtoc（t），dtoc（t，1），ttoc（t），ttoc（t，1）
```

结果为：

```
10/06/13   20231006   10/06/13 10：07：09pm   20231006220709
```

### （五）字符与 ASCII 之间的转换函数

【格式】ASC（<字符型表达式>）

CHR（<数值型表达式>）

【功能】ASC（）函数给出指定字符表达式最左边的一个字符的 ASCII 码值。函数值为数值型。CHR（）函数将数值表达式的值作为 ASCII 码，转换为对应的字符。函数值为字符型。例如：

```
? asc（"cat"），chr（65）
```

结果为：

```
99A
```

常用数值类型转换函数见表 2-9 所列。

表 2-9　常用数值类型转换函数

| 函数名、格式及说明 | 输出类型 | 例子 |
|---|---|---|
| CTOD（<字符表达式>）<br>把字符表达式转换成日期型日期 | D | ? ctod（"10/18/23"）<br>输出：10/18/23 |
| DTOC（<日期表达式>［，1］）<br>把日期表达式转换成字符型日期 | C | ? dtoc（｛^2024-06-09｝，1）<br>输出：20240609 |
| CTOT（<字符型表达式>）<br>将字符型表达式值转换成日期时间型数据 | D | Set century to on<br>Set century on<br>t="2023-10-6"+""+time（）<br>? ctot（t）<br>输出：2023/10/06 10：27：32 PM |
| TTOC（<日期时间表达式>）<br>将日期时间表达式转换成字符串 | C | ? ttoc（datetime（））<br>输出：2023/10/06 10：34：48 PM |
| CHR（<数值表达式>）<br>把数值表达式的值转换为相应的 ASCII 码字符 | C | ? chr（98）<br>输出：b |
| STR（<数值表达式 1>［，<数值表达式 2>［，<数值表达式 3>］］）<br>将数值表达式 1 转换为字符型数据，数值表达式 2 是转换的长度，数值表达式 3 指定小数位数 | C | ? str（156.73，6，1）<br>输出：156.7<br>A=1<br>? len（str（A））<br>输出：10 |
| VAL（<字符表达式>）<br>将字符表达式转换为数值型数据 | N | ? val（"-563.4375"）<br>输出：-563.44 |

## 五、测试函数

在数据处理过程中，有时需要了解操作对象的状态，比如文件是否存在，检索是否成功，数据库当前记录号是否到达文件尾等。此时，往往需要根据测试函数的返回值来决定下一步的处理方法或程序走向。

### （一）条件测试函数

【格式】IIF（<逻辑型表达式>，<表达式 1>，<表达式 2>）

【功能】测试<逻辑表达式>的值，若为逻辑真 . T. ，函数返回表达式 1 的值；若为逻辑假 . F. ，函数返回表达式 2 的值。表达式 1 和表达式 2 的类型不要求相同。例如：

```
x=2
y=9
? iif（x>8，x-1，x+1），iif（y>2，y-1，y+1）
```

结果为：

```
3      8
```

### （二）数据类型测试函数

【格式】VARTYPE（<表达式>［，<逻辑表达式>］）

【功能】测试表达式的类型，返回一个大写字母，函数值为字符型。函数值的含义见表 2-10 所列。

表 2-10　函数值的含义

| 返回的字母 | 数据类型 | 返回的字母 | 数据类型 |
|---|---|---|---|
| C | 字符型或备注型 | G | 通用型 |
| N | 数值型、整型、浮点型或双精度型 | D | 日期型 |
| Y | 货币型 | T | 日期时间型 |
| L | 逻辑型 | X | NULL 值 |
| O | 对象型 | U | 未定义 |

若表达式是一个数组，则根据第一个数组元素的类型返回字符串。若表达式的运算结果是 NULL 值，则根据逻辑表达式的值决定是否返回表达式的类型；如果逻辑表达式值为 . T. ，就返回表达式的原数据类型；如果逻辑表达式值为 . F. 或默认，则返回 X，以表明表达式的运算结果是 NULL 值。例如：

```
X="AAA"
store 10 to y
store. null. to x
store $ 12. 22 to z
vartype（x），vartype（X，. T. ），vartype（y），vartype（z）
```

结果为：

```
X      C      N      Y
```

### （三）表文件首测试函数

【格式】BOF（［<工作区号>| <表别名>］）

【功能】测试当前表文件（当默认自变量时）或指定表文件中的记录指针是否指向文件首（也即第一条记录的前面位置）。若是，则返回 . T. ；若否，则返回 . F. 。

**注意：**

若指定工作区上没有打开的表文件，函数返回 . F. 。若表文件中不包含任何记录，函数返回 . T. 。

### （四）表文件尾测试函数

【格式】EOF（［<工作区号>| <表别名>］）

【功能】测试当前表文件（默认自变量时）或指定表文件中的记录指针是否指向文件尾（也即最后一条记录的后面位置）。若是，则返回 . T. ；若否，则返回 . F. 。

**注意：**

若指定工作区上没有打开的表文件，函数返回 . F. 。若表文件中不包含任何记录，函数返回 . T. 。

### （五）记录号测试函数

【格式】RECNO（［<工作区号>| <表别名>］）

【功能】返回当前表文件（默认自变量时）或指定表文件中当前记录（记录指针所指记录）的记录号。

**注意：**

如果指定工作区中没有表文件打开，函数值为 0。如果记录指针指向文件尾，函数值为表记录总数加 1。若记录指针指向文件首，函数值为首记录的记录号。

### （六）记录个数测试函数

【格式】RECCOUNT（［<工作区号>| <表别名>］）

【功能】返回当前表文件（默认自变量时）或指定表文件中记录的个数。

**注意：**

如果指定工作区上没有表文件打开，函数值为 0。

### （七）记录删除测试函数

【格式】DELETED（［<工作区号>| <表别名>］）

【功能】测试在指定的表或指定工作区打开的表中，记录指针所指的当前记录是否有删除标记“ * ”。若有，则返回 . T. ；若否，则返回 . F. 。

**注意：**

若默认自变量，则默认测试当前工作区中打开的表。

### （八）值域测试函数

【格式】BETWEEN（<被测试表达式 T>，<下限表达式 L>，<上限表达式 H>）

【功能】判断被测试表达式的值是否介于另外两个表达式的值之间。当表达式 T 值大于或等于表达式 L 且小于或等于表达式 H 时，函数值为逻辑 . T. ，否则函数值为逻辑 . F. 。如果表达式 L 或表达式 H 有一个是 NULL 值，那么函数值也是 NULL 值。该函数的自变量类型既可以是数值型，也可以是字符型、日期型、日期时间型、浮点型、整型、双精度型或货币型。但三个变量的数据类型要一致。例如：

```
store 100 to a
between（150，a，a+100），between（15，a，a+100）
```

结果为：

```
. T.      . F.
```

常用测试函数见表 2-11 所列。

表 2-11　常用测试函数

| 函数名、格式及说明 | 输出类型 | 例子 |
|---|---|---|
| BOF(［<工作区号>\|<表别名>］)<br>测试表文件开始状态 | L | use_teacher<br>gobottom<br>? recno( ) ,eof ( )<br>10. F.<br>skip<br>? recno( ) ,eof( )<br>11　　. T. |
| EOF(［<工作区号>\|<表别名>］)<br>测试表文件结束状态 | L | |
| RECCOUNT( )<br>返回表文件记录总数(包括已作删除标记的记录) | N | use_teacher<br>? recount( )<br>&& 测试教师数据表的记录总数<br>7 |

续表

| 函数名、格式及说明 | 输出类型 | 例子 |
|---|---|---|
| RECNO([<工作区号>|<表别名>])<br>返回指定表中当前记录号 | N | ? recno( ) |
| IIF(<逻辑表达式>,<表达式 1>,<表达式 2>)<br>在逻辑表达式的值为 .T. 时,返回表达式 1 的值;<br>在逻辑表达式的值为 .F. 时,返回表达式 2 的值 | | num=10<br>? iif(num>9,num+1,num-1)<br>11 |

## 六、其他常用函数

【格式】MESSAGEBOX（<字符串>［，<对话框类型>［，<对话框标题字符串>］］）

【功能】显示提示对话框。提示对话框类型及定义见表 2-12 所列，函数返回值见表 2-13 所列。

表 2-12　对话框类型及定义

| 设置按钮类型值 | 图标类型值和对应图标 | 设置隐含按钮（默认按钮） |
|---|---|---|
| 0：“确定” | 16：“停止”图标 | 0：第一个按钮 |
| 1：“确定”“取消” | 32：问号 | 256：第二个按钮 |
| 2：“放弃”“重试”“忽略” | 48：惊叹号 | 512：第三个按钮 |
| 3：“是”“否”“取消” | 64：信息图标 | |
| 4：“是”“否” | | |
| 5：“重试”“取消” | | |

表 2-13　MESSAGEBOX 函数返回值

| 返回值 | 按钮 | 返回值 | 按钮 |
|---|---|---|---|
| 1 | 确定 | 5 | 忽略 |
| 2 | 取消 | 6 | 是 |
| 3 | 放弃 | 7 | 否 |
| 4 | 重试 | | |

【例 2-14】MESSAGEBOX 对话框示例。例如，执行以下命令将弹出如图 2-1 所示的对话框。

```
MESSAGEBOX（"是否要退出系统?"，2+32+256,"退出"）
```

【说明】

（1）对话框提示信息文本为“是否要退出系统?”。

（2）对话框标题栏文本为“退出”。

（3）2 表示：按钮排列为“中止”“重试”“忽略”。

（4）32 表示：消息框显示图标为问号。

（5）256 表示：默认按钮为第二个按钮“重试”。

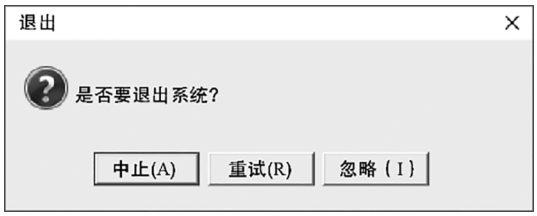



图 2-1　使用 MESSAGEBOX 函数定义的“退出”对话框

# 实训任务：Visual FoxPro 表达式求值与 BMI 指数判断

## 任务说明

### 一、表达式求值

小李想通过 Visual FoxPro 计算数学表达式 $\dfrac{6+\dfrac{y}{x}}{13-\dfrac{y}{x^2}}$ 的值。现已知 $x=2$，$y=36$，写出符合 Visual FoxPro 要求的表达式并求出表达式的最终结果。

### 二、BMI 指数判断

BMI 指数是用体重（单位：kg）除以身高（单位：m）的平方得出的数字，其值处于 18~24 范围可视为身高、体重指数正常。已知小李身高 1.78 m，体重 70 kg，判断小李的身高、体重指数是否正常。

## 任务实施

### 一、表达式求值

步骤 1　将数学表达式转换为 VFP 表达式：(6+y/x) / (13-y/ (x^2) )。

步骤 2　在命令窗口执行以下命令：

```
x=2
y=36
? (6+y/x) / (13-y/ (x^2) )
```

步骤 3　查看 Visual FoxPro 的工作区，可以看到表达式的结果为 6.0000。

### 二、BMI 指数判断

步骤 1　设身高为 H(m),体重为 W(kg)，则 BMI 指数的计算公式为 BMI=W/H^2。

步骤 2　在命令窗口执行以下命令：

```
H=1.78
W=70
? W/H^2>=18 AND W/H^2<=24
```

步骤 3　查看 Visual FoxPro 的工作区，可以看到表达式的结果为 .T.，说明小李的 BMI 指数正常。

## 思考练习

一、选择题

1. 在 Visual FoxPro 中，下列 4 个关于日期或日期时间的表达式中，错误的是_____。

A. {^2022.09.01 11：10：10：AM} - {^2021.09.01 11：10：10AM}

B. {^01/01/2022} +20

C. {^2022.02.01} + {^2021.02.01}

D. {^2022/02/01} - {^2021/02/01}

2. 在默认状态下，表达式"WinWord"="Win"结果为_____。

A. 1　　B. .T.　　C. .F.　　D. 0

3. 已知 D=5>6，则函数 VARTYPE（D）的结果是_____。

A. L　　B. C　　C. N　　D. D

4. 下列 4 个表达式中，运算结果为数值的是_____。

A. 30+28=58　　B. "30"+"28"

C. {^2012/12/20} -10　　D. LEN（SPACE（3））-1

5. 函数 SUBSTR（"Visual FoxPro 6.0"，8，6）的返回值是_____。

A. VisualFoxPro　　B. FoxPro

C. FoxPro6　　D. FoxPro 6.0

6. 设 A=6，执行命令？A=A+1 后，结果是_____。

A. 7　　B. 6　　C. .T.　　D. .F.

7. 已知字符串 S1="ABCD "，S2="EF GH"，则下列表达式的运算结果是 .T. 的为_____。

A. S1-S2=ALLTRIM（S1）+S2　　B. S1+S2=S1-S2

C. ALLTRIM（S1）+S2=S2　　D. S1-S2="ABCDEFGH"

8. 函数 ROUND（125 623.122 3，-3）的运算结果为_____。

A. 125 623　　B. 126 000

C. 125 623.122　　D. -125 623.122 3

9. 在下列 Visual FoxPro 6.0 表达式中，不正确的是_____。

A. 04/05/13-2　　B. CTOD（'04/11/13'）-DATE（）

C. YEAR（CTOD（'04/05/13'））-3　　D. DATE（）+"04/05/13"

10. 函数 MAX（MOD（SQRT（36），INT（-3.5）），ABS（-1））的运算结果为_____。

A. 1　　B. 2　　C. -4　　D. 36

二、填空题

1. 在 Visual FoxPro 6.0 中，函数可分为_____、日期型函数、_____、逻辑型函数和_____。

2. 函数 BETWEEN（40，30，50）的运算结果是_____。

3. LEFT（"123456789"，LEN（"数据库"））的计算结果是_____。

4. 执行命令 A=2005/4/2 之后，内存变量 A 的数据类型是_____型。

5. 在 Visual FoxPro 6.0 中，函数由_____、参数、_____构成。其中，_____起标识作用；_____用于在程序之间传递信息，_____是函数运算结束后的返回值。

6. Visual FoxPro 6.0 的变量有_____和_____两种。

# 项目三　数据库与数据表的操作

## 项目概述

开发一个数据库应用系统，首先要建立该系统所管理的对象——数据库。在 Visual FoxPro 中，数据库用来组织和管理相互联系的多个数据表及相关的数据库对象。每个数据表都是由表结构和数据记录组成的二维表，独立地存放在磁盘上。

本项目将介绍如何设计数据库，以及建立、打开、关闭数据库的操作。重点讲解建立和维护数据表的方法，索引的建立和使用。此外，还将介绍如何在数据表之间建立关联和设置参照完整性。

## 学习目标

◎知识目标

1. 掌握数据库和数据表的创建方法。
2. 掌握数据库和数据表的基本操作。
3. 掌握索引的概念和使用方法。
4. 掌握使用多个表中数据的方法。
5. 了解永久联系及参照完整性。

◎能力目标

1. 能创建和修改数据库。
2. 能创建和修改数据表。

◎素质目标

1. 培养客观、严谨的科学态度。
2. 以信息化推进中国式现代化，为中华民族伟大复兴贡献力量。

## 任务一　设计数据库

要成功地建立一个符合用户需要、满足用户要求的数据库，就一定先要进行数据库的设计。一般来说，数据库设计的基本步骤如图 3-1 所示。