AutoCAD系统开发技术
——程序实现与实例

董玉德　谭建荣　赵　韩
魏修亭　张　燕　伊国栋　编　著

中国科学技术大学出版社

AutoCAD 系统开发技术

——程序实现与实例

董玉德　谭建荣　赵　韩
魏修亭　张　燕　伊国栋
编著

中国科学技术大学出版社
2001 · 合肥

内 容 简 介

本书以 AutoCAD 2000(包括 R14)为开发平台，以 Visual C++为编程工具，运用大量的专题实例，详细而又系统地介绍了用 ObjectArx 3.0(包括 ADS)进行二次开发的方法和技巧。

本书的一个重要特点是给出了如何运用 ObjectARX 和 Visual C++进行实用 CAD 系统的开发，而不仅仅是一本简单介绍 ObjectArx 的参考书籍。

本书的内容涉及到开发一个实用 CAD 系统的多个方面，其中包括系统开发的基础知识、绘图与设计环境程序设计、2D 参数化绘图、3D 参数化绘图、基于 ASE 的数据共享方法、图文管理系统开发方法，并就 AutoCAD 2000 的新特性，分别对多文档界面、事务处理、COM 等专项问题进行了探讨。

全书内容安排详略得当，语言易懂。各章专题例程相互独立，读者可从中学到 CAD 系统软件开发的方方面面。同时，本书还提供了部分 ADS 例程。

本书有鲜明的个性和极大的实用价值，可作为大专院校的师生学习 AutoCAD 二次开发的培训教材，也可供机械、电子、计算机、建筑、服装、广告等行业的 CAD 技术人员使用。

图书在版编目（CIP）数据

AutoCAD 系统开发技术——程序实现与实例/董玉德等编著.—合肥：中国科学技术大学出版社，2001.10

ISBN 7-312-01317-1

I. AutoCAD… II.①董… ②谭… ③赵… III.计算机辅助设计-应用软件，AutoCAD IV.TP391.72

中国版本图书馆 CIP 数据核字（2001）第 067138 号

中国科学技术大学出版社出版发行

（安徽省合肥市金寨路 96 号，邮编：230026）

中国科学技术大学印刷厂印刷

全国新华书店经销

开本：787×1092/16　　印张：18.5　　字数：480 千

2001 年 10 月第 1 版　　2001 年 10 月第 1 次印刷

印数：1—3000 册

ISBN 7-312-01317-1/TP・278　　定价：40.00 元

前　言

AutoCAD 作为平台上使用最为广泛的 CAD 系统，在机械、电子、建筑、服装等行业得到广泛的应用。二次开发是 AutoCAD 对外开放特性中最具魅力的一面旗帜，ObjectARX 是继 AutoLISP、ADS、ADSARX 后的第三代开发工具，采用全新的面向对象编程技术和 Visual C++ 集成化开发环境，在开发的内容和形式上都发生了变化，受到了 AutoCAD 使用者和广大程序开发人员的欢迎。ObjectARX 3.0 版针对 AutoCAD 2000 中的多文档界面、事务处理、COM 等最新特性提供了相应的开发手段。可以认为 ObjectARX 将会成为 AutoCAD 的主力开发工具。

目前，已出版的有关 ObjectARX 开发的参考书为数不少，对读者了解 Arx 的主要内容和应用程序开发方法起了重要作用，但由于时间等因素的限制，如何用 Arx(包括 ADS)开发一套实用的 CAD 系统，这一方面的书籍还比较少见。由于使用 Arx 编程不仅要求开发者对 AutoCAD 系统本身有深刻的认识，而且要能把面向对象的 C++编程和 Arx 有机的结合起来，这样对初学者，甚至是有一定编程经验的程序开发者来说，开发一套实用的 CAD 系统都会相当困难。

本书的作者以多个 CAD 开发项目为研究背景，在认真总结 5 年多开发经验的基础上，完成了本书的编写。全书在内容安排上以较少的篇幅介绍了 Arx 的基础知识，相当部分的内容给出了在工程中开发实例，每个程序都具有极高的实用价值。考虑到 AutoCAD 2000 版本的新内容，在本书的最后对 Arx 的最新开发内容也做了一定的介绍。全书共分 7 章，内容安排如下：

第一章　介绍 ObjectArx 的基础知识和应用程序开发方法。

第二章　介绍绘图设计环境程序设计方法，内容包括基本绘图环境设置、工程设计标注、装配图基本要素、图元变换。

第三章　介绍 2D 参数化绘图，内容包括程序驱动、尺寸驱动、变量驱动等。

第四章　介绍 3D 参数化绘图，内容包括三维实体图元生成、如何遍历三维实体图元的拓扑结构、复杂零件的三维造型。

第五章　介绍基于 ASE 的数据共享方法，内容包括 ASE 类库、图形实体与数据库中表的关联实例。

第六章　介绍基于 AutoCAD 的图文管理系统的开发方法。

第七章　就 AutoCAD 2000 的新特性，分别对多文档界面、事务处理、COM 等专项问题进行了探讨。

本书由董玉德副教授策划和统稿。第三章中的程序源代码由魏修亭教授编写和提供，第四章中的标准件程序代码由张燕副教授编写，第六章中的内容和程序源代码由伊国栋博士编写和提供。其它章节由董玉德负责编写。

由于时间仓促，作者水平有限，书中一定存在很多错误，恳请广大读者批评指正。

作　者

2001 年 10 月于中国科大

目　录

第 1 章　ObjectArx 程序设计基础

1.1　ObjectArx 编程

1.1.1　为什么要用 ObjectArx

AutoCAD 有多种编程接口，AutoLISP 是它的第一个编程语言。它是一种解释性的编程语言，它提供了一个简单的扩充 AutoCAD 命令的机制，最初出现于 1985 年发行的 AutoCAD 2.5 版中。四年以后，在 R10 中增加了称为 ADS 的 C 语言编程能力，一个 ADS 程序实际上是由一组外部函数组成，它们由 AutoLISP 解释器来加载调用，ADS 程序本身并不能直接和 AutoCAD 进行通信。然后，在 R13 中又增添了 ARX(AutoCAD 运行扩展)编程接口，它是新一代的基于 C++的应用程序接口，可以为应用程序扩展 AutoCAD 的功能提供前所未有的能力。ARX 程序在很多方面都和 ADS 程序、AutoLISP 程序不同。最重要的一点是，ARX 程序实质上是一个动态链接库（DLL），它和 AutoCAD 共享地址空间并且直接和 AutoCAD 进行通信。对于经常需要和 AutoCAD 通信的应用程序来说，ARX 程序比 ADS 运行更块。除了速度上的提高之外，ARX 程序还可以创建新的类，这些类可以为其它程序共享，从而充分利用面向对象编程的优点。

ObjectArx 作为 AutoCAD 系统的第三代程序开发工具,和早期的 ADS，ADSARX，LISP 以及现在的 VBA，Visualize LISP 相比都有无可比拟的优越性，主要表现在：

- 全面支持面向对象的 C++编程，能充分利用 C++编程方法的一切优点；
- ObjectArx 应用程序本身就是一个动态链接库，它共享 AutoCAD 的地址空间，并可通过多种方式调用 AutoCAD 命令和函数，应用程序中的命令和 AutoCAD 的内部命令在形式上没有什么区别；
- ObjectArx 应用程序可以直接访问 AutoCAD 的数据结构和图形系统，可以这样说，在 AutoCAD 编辑环境下的所有动作，在应用程序中都可实现，在 AutoCAD 编辑环境下不能实现的行为，也可以用应用程序实现；
- 利用 ObjectArx,可充分利用 MFC 的网络编程功能，支持异地协作设计。

1.1.2　ObjectArx 的变化

1. 关于 ADS

在 1989 年，AutoCAD R10 在 OS/2 平台上引入了 ADS 作为一个新的开发应用程序的编程环境。在 1990 年，AutoCAD R11 也提供了在 DOS 及其它操作系统上对 ADS 的支持。AutoCAD 和 ADS 的应用程序是不同的可执行程序（.EXE），它们通过内部进程通信（IPC）

进行数据交换。

基于 ADS 的程序是用 C 语言编写的，然而，这些程序的运行情况和 AutoLISP 程序是一样的，因为作为外部编译过的 ADS 程序是由 AutoLISP 解释器加载的。而 AutoLISP 解释器是由 AutoCAD 加载并调用的。

ADS 程序是由 C 语言编制的，因此相对于 AutoLISP 具有一些编译程序的优点，例如这些优点可以表现在以下一些方面：源程序的安全性，内存操作的效率，文件尺寸及运行速度。

和标准的 C 语言函数库一样，ADS 系统也是由一些库文件和头文件组成的。ADS 函数库提供了所有必要的工具及函数用来与 AutoLISP 及 AutoCAD 进行数据通信。ADS 的函数名都是以“ads_”为前缀的，这就很容易和标准的 C 函数区别开。

AutoCAD R13 向开发商提供了能够直接存取的、面向对象的内核，最初被命名为 AutoCAD 运行扩展，或 ARX。这个面向对象的内核最终被命名为 ObjectARX，是 AutoCAD 的第一个 C++ 的编程接口。做为动态链接库（DLL），ObjectARX 应用程序可以在 AutoCAD 运行时存取其内核及内存地址空间，就像 AutoCAD 自己的特征做的一样。

ObjectARX 的应用程序能够存取 AutoCAD 中的符号表和系统变量、操作选择集、提示用户输入、进行坐标转换及查询（如捕捉和点取点）和控制图形显示。这些功能在 ADS 中也能做到，因此很自然地也被包含到了 ObjectARX 中。

把 ADS 功能集成进 ObjectARX 中，可以使 C++ 的应用程序兼容已有的 C 的应用程序。通过由动态链接库共享 AutoCAD 的内存地址空间，ADS 应用程序就不再需要通过 IPC 和 AutoCAD 进行通信了。

在 AutoCAD R14 中，ObjectARX 内核得到了进一步的改善，向开发商增加了扩展 AutoCAD 的能力。和 AutoCAD R14 一起提供了三套 ADS 库，包括：非 DLL 或 IPC 类型的 ADS，ADSRX（快速 ADS）和在 ObjectARX 中的 ADS 库。由于 ADSRX（包含在 AutoCAD 产品的光盘上）和在 ObjectARX 中包含的 ADS 库在内容上是完全相同的，同样的源程序可以和任意一套 ADS 库进行编译连接，并生成同样的 ARX 或动态链接库文件。

R14 提供了一个新的用 Windows 的系统注册表加载 ObjectARX 应用程序的方法，可以按需加载 ObjectARX 的应用程序。这种方法只适用于 ObjectARX 的应用程序，而不适用于老式的非 DLL 或 IPC 类型的 ADS 应用程序。AutoCAD 2000 提供了一个重大的改进就是把 ADS 集成进了 ObjectARX。

通过 ObjectARX 使 ADS 有了几个重大的改进，包括：

（1）不再提供慢速的、非 DLL 或 IPC 类型的 ADS，通过集成进 ObjectARX，所有的 ADS 应用程序都已是动态链接库（DLL）类型了。

（2）ADSRX 函数库不再被单独提供。在 R14 的 ADSRX 中的以“ads_”为前缀的函数限制了 ObjectARX API 的性能，通过 AutoCAD 2000 中的 ObjectARX 开发 ADS 应用程序，这种限制将不复存在。

（3）用 ObjectARX 开发的应用程序可以共享 AutoCAD 的内存地址空间，因此，应用 ObjectARX 函数库可以确保应用程序具有更高的效率及更好的性能。

AutoCAD 2000 不再支持老式的 DOS 风格的加载 ADS 应用程序（非 DLL，IPC 类型）的方法。R14 中的（xload）和（xunload）函数已经不存在了。

（4）“ads_”函数已被改名为 acdbFuncName()，acedFuncName() 或 acutFuncName()，这些函数的名字显示了它们在 ObjectARX 系统里的功能。函数名的 4 个字母前缀分别代表了图形数据库、编辑器和工具函数。

（5）ADS 已成为 ObjectARX 的一部分，不再被描述为一个分离的，独立的开发系统。为了保持兼容性，AutoCAD 2000 里的 ObjectARX 已经定义（#define）了“ads_”类函数，指向新的 ADS 函数，使得原有的应用程序还能使用老的函数名(各种函数类型见附录二)。

2. ARX 与 ADS 的区别

AutoCAD 2000 的 ObjectARX 继续支持所有 ADS C 的库函数。为了使 AutoCAD 成为一个完全的 Windows 的应用程序，在 AutoCAD 2000 的 ObjectARX 中已不再提供对 ADS 中对老式的、非 DLL 接口的支持。当移植 ADS 应用程序到 AutoCAD 2000 上时，只要用 ObjectARX 函数库中的 ADS 接口重新编译源程序即可，而且还会把 ObjectARX 中的先进功能自动加入到 ADS 应用程序中。

ObjectARX 接口的功能十分强大，AutoCAD 自身的很大一部分就是用 ObjectARX 实现的。例如，处理光栅图象的子系统就几乎没有向 AutoCAD 核心系统增添什么新代码。其结果是，ObjectARX 使得 AutoCAD 成为了一个更加模块化的系统。需要了解的是，ObjectARX 并没有取代 LISP 和 ADS，在 AutoCAD 2000 中，LISP 和 ADS 仍然存在并有所扩展。

(1) ARX 程序是一个动态链接库(DLL)，它直接和 AutoCAD 进行通信。ADS 程序是一个可执行文件，它需要通过 AutoLISP 来和 AutoCAD 进行通信。

(2) AutoCAD 是不可重入的，因此 ADS 程序也是不可重入的。而在 ARX 中，每一个命令都有独立的入口。

(3) ARX 程序速度快，但更“脆弱”，ARX 程序和 AutoCAD 共享进程空间，ARX 程序本身是 AutoCAD 的一部分，ARX 程序的崩溃通常会导致 AutoCAD 系统的崩溃。而 AutoLISP 和 ADS 都是通过函数来间接访问 AutoCAD。ADS 程序速度慢，但更“绝缘”，ADS 程序崩溃并不一定导致 AutoCAD 系统崩溃。

(4) ADS 程序类似如宏（macro），ADS 中的函数（如 ads_command）以及和 AutoLISP 的通信使得 ADS 程序的工作类似于自动作用的宏。相比之下，ARX 程序则更基本，主程序（AutoCAD）调用每一个 ARX 程序注册的命令。

(5) ARX 程序具有 ADS 程序和 AutoLISP 程序所不具备的访问和控制 AutoCAD 的能力。

(6) ARX 提供了面向对象编程的技术。ARX 充分支持 C++，充分支持面向对象编程的技术，而 ADS 仍然只能使用传统的 C 语言编程。ARX 充分支持 C++类，通过五个主要类库来访问和控制 AutoCAD，另外，还可以利用 VC 中的 MFC 类库。

1.1.3 ObjectArx 编程第一步

随 AutoCAD 2000 同时发行的 ObjectArx 嵌入工具使得 ObjectArx 应用程序的开发和一般 C++应用程序的开发过程一样简单。

1. 嵌入工具的安装与配置

（1）运行 WizardSetup.exe,将应用程序向导安装到 Miscrosoft VisualStdio 6.0 中；

（2）在 Visual C++集成开发环境中，按以下顺序打开工具条：

Tools→Customize→Add-ins and Marcro Files →ObjectArx2000 Add-In→Close。

2. 创建第一个应用程序

（1）进入 Visual C++集成开发环境；

（2）File→New→Project→ObjectArx 2000 Appwizard→设置一工程文件名称；

（3）指定该工程文件的类型（ObjectDBX、ObjectArx、COM）和使用的库（MFC、ATL）；

（4）完成。 这时的应用程序还只是一个框架，没有任何功能。

3. 利用嵌入工具，可以完成的工作

（1）可以包含在工程中要使用的类，这要根据应用程序的需要而选择；

（2）可以在应用程序中追加命令，这些命令可以向 AutoCAD 的内部命令一样使用；

（3）增加对 ObjectArx 入口点信息的处理；

（4）可以对 ObjectArx 的反应器进行管理，如添加或删除反应器类、从一个已定义的反应器中添加或删除函数；

（5）可以自定义或修改一个类中的函数、成员变量；

（6）可以选择 MFC 支持功能，如加载应用程序时使用屏幕背景、获取 Windows 传递给 AutoCAD 的消息、创建消息处理器窗口、支持 MFC 工具条等；

（7）使用 ObjectArx 入口点 API,创建 AcEdInputPointMonitor 和 AcEdInputPointFilter 派生的类；

（8）设置应用程序的加载特性；

（9）加入 ATL 部件；

（10）存储一些常用的函数和源程序的代码；

（11）可以启动在线帮助文件。

1.1.4 ObjectArx 应用程序的能力

ObjectArx 程序设计环境，为程序员使用、用户化和扩充 AutoCAD, 提供了一个面向对象的 C++应用程序开发接口。使用 ObjectArx 可以完成下列任务：

1. 访问 AutoCAD 数据库

AutoCAD 图形实质上使存储在图形数据库中对象的集合，包括实体（点、线、面、体尺寸、文本）和非实体对象（图层、线型、各种视图等）。ObjectArx 提供的类和函数可以直接访问这些对象（在 AutoCAD 环境或其它环境），另外，根据实际需要还可以在应用程序中创建各种数据库对象。

2. 与 AutoCAD 编辑器进行交互

ObjectArx 提供了与 AutoCAD 编辑器通信的类和成员函数，ObjectArx 应用程序可以向 AutoCAD 注册自定义命令，这些命令的运行方式将和 AutoCAD 内部命令一样，应用程序可以接受和回应发生在 AutoCAD 内的各种事件。

3. 使用 MFC 创建用户界面

ObjectArx 应用程序在编译时可以链接 MFC 类库，使用 VC 中的各种控件，从而可以在 AutoCAD 中使用标准的 Windows 用户界面。

4. 支持多文档接口

ObjectArx 应用程序可以支持 AutoCAD 2000 的多文档接口。

5. 自定义用户类

可以用 ObjectArx 类库中的各种类作为基类，创建用户自定义类。

6. 创建复杂的应用程序

ObjectArx 应用程序可以完成接受事件通告、进行事务处理、进行深层克隆复制、引用编辑、协议扩充和代理对象支持。

7. 与其它编程接口通信

ObjectArx 应用程序可以和其它的编程接口如 Visual Lisp、ActiveX、COM 等进行通信，另外，还可以通过 Internet 和其它对象进行通信。

1.2 AutoCAD 数据库内容

1.2.1 符号表

AutoCAD 数据库（AcDb）是按一定结构组织的 AutoCAD 图形全部有关数据的结合。用一组符号表和有名对象字典来组织和管理数据库对象。

◆ 块表（AcDbBlockTable）

块表类从代码表类（AcDbSymbolTable）继承而来,用以表示在图形数据库中对块的定义。

构造函数 AcDbBlockTable::AcDbBlockTable

在应用程序中不需要使用该构造函数，而是由 AcDbDatabase 创建；

查询函数

方法 1

Acad::ErrorStatus getAt(const char* entryName,AcDbObjectId& recordId, bool getErasedRecord = false) const;

查询记录的输入名称 recordId，返回实体的标识 ID，getErasedRecord 表示该实体是否存在或被删除

方法 2

Acad::ErrorStatus getAt(const char* entryName,AcDbBlockTableRecord*& pRecord, AcDb::OpenMode openMode, bool openErasedRecord = false) const;

EntryName 查询记录的输入名称 ，pRecord 返回打开的记录

openMode 记录的打开方式（AcDb::kForRead、AcDb::kForWrite、AcDb::kForNotify ）

getErasedRecord 表示该实体是否存在或被删除

编辑函数

Acad::ErrorStatus add(AcDbBlockTableRecord* pRecord);

Acad::ErrorStatus add(AcDbObjectId& recordId,AcDbBlockTableRecord* pRecord);

◆ 尺寸标注样式表（AcDbDimStyleTable）

尺寸标注样式表类从代码表类（AcDbSymbolTable）继承而来,用以表示在图形数据库中对尺寸类型的定义。

查询函数

方法 1

Acad::ErrorStatus getAt(const char* entryName, AcDbObjectId& recordId,bool getErasedRecord = false) const;

EntryName　查询记录的输入名称，recordId　　返回查询对象的 ID

getErasedRecord　表示该对象是否存在或是否被删除

方法 2

Acad::ErrorStatus getAt(const char* entryName,AcDbDimStyleTableRecord*& pRecord,AcDb::OpenMode openMode, bool openErasedRecord = false) const;

entryName　查询记录的输入名称，pRecord　　返回打开的记录

openMode　输入打开记录的模式（AcDb::kForRead、AcDb::kForWrite、AcDb::kForNotify）

getErasedRecord　输入布尔值标识是否或没有发现一个删除的记录

方法 3

bool has(const char* name) const;　name　查询尺寸的名称

结果：如存在该尺寸实体则返回 TRUE，否则返回 FALSE

方法 4

bool has(const AcDbObjectId & id) const; id　查询尺寸的 ID

结果：如存在该尺寸实体则返回 TRUE，否则返回 FALSE

方法 5

Acad::ErrorStatus newIterator(AcDbDimStyleTableIterator*& pIterator,bool atBeginning = true, bool skipDeleted = true) const;

pIterator　返回指向最新生成的反应器指针

atBeginning　输入布尔值约定是在表的开始或结尾扫瞄

skipDeleted　输入布尔值约定在扫瞄时是否忽略已删除记录

编辑函数

追加一个尺寸记录

Acad::ErrorStatus add(AcDbDimStyleTableRecord* pRecord);

pRecord　指向记录的指针

结果：Acad::eOk, Acad::eOutOfMemory, Acad::eDuplicateRecordName, or Acad::eNoDatabase (如果尺寸不在数据库中).

Acad::ErrorStatus add(AcDbObjectId& recordId, AcDbDimStyleTableRecord* pRecord);

recordId　返回该记录的对象标识符 ID，pRecord　输入指向记录的指针

◆ 层表（AcDbLayerTable）

记录 AutoCAD 数据库中图层信息。

追加一个图层

Acad::ErrorStatus add(AcDbLayerTableRecord* pRecord); pRecord　指向层表记录的指针

Acad::ErrorStatus add(AcDbObjectId& recordId, AcDbLayerTableRecord* pRecord);

recordId　返回该记录的对象标识符 ID ，pRecord　指向层表记录的指针

查询图层

Acad::ErrorStatus getAt(const char* entryName, AcDbObjectId& recordId, bool getErasedRecord = false) const;

entryName 查询记录的名称， recordId 返回该记录的对象标识符 ID

getErasedRecord 输入一个布尔值表示是否或者没有发现删除的记录

Acad::ErrorStatus getAt(const char* entryName, AcDbLayerTableRecord*& pRecord, AcDb::OpenMode openMode, bool openErasedRec = false) const;

entryName 输入查询记录的名称 ，pRecord 返回打开记录的指针

openMode 输入打开记录的模式（AcDb::kForRead、AcDb::kForWrite、AcDb::kForNotify）

openErasedRec 输入一个布尔值表示是否打开一个已经被删除的记录

bool has(const char* name) const; name 查询记录的名称

boolean has(AcDbObjectId id) const; id 查询记录的对象标识符 ID

Acad::ErrorStatus newIterator(AcDbLayerTableIterator*& pIterator, bool atBeginning = true, bool skipDeleted = true) const;

pIterator 返回指向新生成反应器的指针，atBeginning 输入布尔值约定是在表的开始或结尾扫瞄

skipDeleted 输入布尔值约定是否跳读已删除的记录

◆ 线型表（AcDbLinetypeTable）

用以描述图形数据库中的线型。

查询函数

Acad::ErrorStatus getAt(const char* entryName, AcDbObjectId& recordId, bool getErasedRec = false) const;

entryName 输入查询记录的名称，recordId 返回该记录的对象标识符 ID

getErasedRec 输入布尔值表示是否不查找已删除的记录

Acad::ErrorStatus getAt(const char* entryName, AcDbLinetypeTableRecord*& pRec, AcDb::OpenMode openMode, bool openErasedRec = false) const;

entryName 输入查询记录的名称，pRecord 返回已打开记录的指针

openMode 输入打开记录的模式（AcDb::kForRead、AcDb::kForWrite、AcDb::kForNotify）

openErasedRec 输入布尔值表示是否不查找已删除的记录

bool has(const char* name) const; name 输入查询记录的名称

bool has(AcDbObjectId id) const; id 输入查询记录的对象标识符 ID

Acad::ErrorStatus newIterator(AcDbLinetypeTableIterator*& pIterator,

bool atBeginning = tue, bool skipDeleted = true) const;

pIterator 返回指向新生成反应器的指针，atBeginning 输入布尔值标识是在表的开始或结尾开始扫瞄

skipDeleted 输入布尔值表示是否跳读已删除的记录

◆ 应用程序注册表（AcDbRegAppTable）

表示已寄存的应用名称。

Acad::ErrorStatus getAt(const char* entryName, AcDbObjectId& recordId, bool getErasedRec = false) const;

entryName 输入查询记录的名称，recordId 返回该记录的对象标识符 Id

getErasedRec 输入布尔值表示是否不查找已删除的记录

Acad::ErrorStatus getAt(const char* entryName, AcDbRegAppTableRecord*& pRec,

AcDb::OpenMode openMode, bool openErasedRec = false) const;

entryName　输入查询记录的名称，pRecord　返回已打开记录的指针

openMode　输入打开记录的模式（AcDb::kForRead、AcDb::kForWrite、AcDb::kForNotify）

openErasedRec　输入布尔值表示是否不查找已删除的记录

bool has(const char* name) const;　name　输入查询记录的名称

bool has(AcDbObjectId id) const;　id　输入查询记录的对象标识符 ID

Acad::ErrorStatus newIterator(AcDbRegAppTableIterator*& pIterator, bool atBeginning = tue,　bool skipDeleted = true) const;

pIterator　返回指向新生成反应器的指针，atBeginning　输入布尔值标识是在表的开始或结尾开始扫瞄

skipDeleted　输入布尔值表示是否跳读已删除的记录

Acad::ErrorStatus add(AcDbRegAppTableRecord* pRecord);　pRecord　指向被追加记录的指针

Acad::ErrorStatus add(AcDbObjectId& recordId, AcDbRegAppTableRecord* pRecord)

recordId　返回记录的对象标识符，pRecord　指向被追加记录的指针

◆　文字样式表（AcDbTextStyleTable）

记录文字的样式。

Acad::ErrorStatus getAt(const char* entryName,　AcDbObjectId& recordId, bool getErasedRec = false) const;

entryName　输入查询记录的名称，recordId　返回该记录的对象标识符 Id

getErasedRec　输入布尔值表示是否不查找已删除的记录

Acad::ErrorStatus　getAt(　const　char*　entryName,　AcDbTextStyleTableRecord*&　pRecord,　AcDb::OpenMode openMode, bool openErasedRec = false) const;

entryName　输入查询记录的名称，pRecord　返回已打开记录的指针

openMode　输入打开记录的模式（AcDb::kForRead、AcDb::kForWrite、AcDb::kForNotify）

openErasedRec　输入布尔值表示是否不查找已删除的记录

bool has(const char* name) const;　name　输入查询记录的名称

bool has(AcDbObjectId id) const;　id　输入查询记录的对象标识符 ID

Acad::ErrorStatus newIterator(AcDbTextStyleTableIterator*& pIterator, bool atBeginning = true,　bool skipDeleted = true) const;

pIterator　返回指向新生成反应器的指针，atBeginning　输入布尔值标识是在表的开始或结尾开始扫瞄

skipDeleted　输入布尔值表示是否跳读已删除的记录

Acad::ErrorStatus add(AcDbTextStyleTableRecord* pRecord);　pRecord　指向被追加记录的指针

Acad::ErrorStatus add(AcDbObjectId& recordId,AcDbTextStyleTableRecord* pRecord);

recordId　返回记录的对象标识符，pRecord　指向被追加记录的指针

◆　用户坐标系表（AcDbUCSTable）记录用户坐标系，函数同上。

◆　视口表（AcDbViewPortTable）记录视口的配制，函数同上。

◆　视窗表（AcDbViewTable）记录存储的视窗信息，函数同上。

其它相关类：

AcDb2dPolyline // 表示二维多义线实体

AcDb2dVertex // 表示二维多义线的顶点

AcDb2LineAngularDimension // 表示用两条线定义的角度标注

AcDb3dSolid // 表示三维实体

AcDbAttribute // 表示属性实体

AcDbAttributeDefinition // 表示 ATTDEF 实体

AcDbBlockBegin // 表示块定义的开头部分，该类对象由 AutoCAD 自动生成，并维护

AcDbBlockReference // 表示 INSERT 实体

AcDbBlockTableIterator // 块表的迭代器

AcDbBlockTableRecord // 类对象用来保存图形数据库对象

AcDbBlockTableRecordIterator // AcDbBlockTableRecord 类的迭代器

AcDbCircle // 表示在 AutoCAD 中的圆实体

AcDbDatabase // 表示 AutoCAD 图形数据库，每个类对象包含了不同的系统变量、符号表、符号表记录、实体和所有图形对象

AcDbDictonary // 与数据库相关的对象词典类

AcDbDimension // 所有尺寸标注实体类的基类

AcDbDimStyleTableIterator // AcDbDimStyleTable 类的迭代器

AcDbDimStyleTableRecord // AcDbDimStyleTable 类的记录类

AcDbEllipse // 表示在 AutoCAD 中的椭圆实体类

AcDbEntity // 所有具有图形属性的数据库对象的基类

AcDbFace // 表示三维面实体

AcDbGroup // 表示一个指定名字的实体集合

AcDbGroupIterator // AcDbGroup 类的迭代器

AcDbHyperlink // 包含超链接的路径、该链接的一个下层目录和该链接的描述文本

AcDbLayerTableIterator // AcDbLayerTable 迭代器

AcDbLayerTableRecord // AcDbLayerTable 类的记录类

AcDbLine // 表示线实体

AcDbLinetypeTableIterator // AcDbLinetypeTable 迭代器

AcDbMline // 表示 MLINE 实体

AcDbMlineStyle // 该类对象保存 AcDbMline 实体中线条的数目、偏移量和线型数据

AcDbMText // 表示 AutoCAD 中的 MTEXT（多行文本）实体

AcDbObject // 图形数据库中所有对象的基类

AcDbPoint // 表示点实体

AcDbPolyline //表示多义线实体

AcDbProxyEntity // 一个抽象类，该类提供了图形数据库中代理实体描述数据的访问接口