AutoCAD
技术丛书

李世国 编著

AutoCAD 高级开发技术

ARX编程及应用

机械工业出版社
China Machine Press

AutoCAD技术丛书

AutoCAD高级开发技术

ARX 编程及应用

李世国　编著

机械工业出版社
China Machine Press

本书全面阐述了基于AutoCAD 的第三代开发环境ARX的应用程序设计技术。通过大量实例深入浅出地介绍了利用面向对象的C++语言对AutoCAD进行二次开发的基本方法和步骤、二维和三维实体的生成和编辑算法以及使用MFC实现交互界面可视化设计的关键技术。全书共分9章，第1章和第2章重点介绍了ARX应用程序的设计基础和编译链接方法；第3章和第4章介绍了在工程中非常实用的图块设计和图库管理技术；第5章和第6章全面介绍了AutoCAD数据库容器对象的操作使用技术；第7章介绍了在ARX应用程序中使用MFC的高级编程技术；第8章主要介绍二维参数化程序设计中常用的尺寸标注、图案填充等实用技术；第9章主要介绍了生成基本三维实体、通过挤压和旋转生成三维实体、利用布尔运算生成复杂三维实体的程序设计技术。书中的实例紧密结合工程实际，全部经过上机验证，并已在CAD课题中应用。

本书文字流畅，易于阅读理解，具有很好的实用价值，可供机械、电子等行业的工程技术人员和工科院校的师生使用。

图书在版编目(CIP)数据

AutoCAD高级开发技术：ARX编程及应用/李世国编著．北京：机械工业出版社，1999.9

(AutoCAD技术丛书)

ISBN 7-111-07348-7

Ⅰ.A… Ⅱ.李… Ⅲ.计算机辅助设计-；软件包，AutoCAD Ⅳ.TP391.72

中国版本图书馆CIP数据核字(1999)第27021号

出 版 人：马九荣(北京市百万庄大街22号 邮政编码100037)

责任编辑：陈剑匾

北京市密云县印刷厂印刷·新华书店北京发行所发行

1999年 9 月第1版第1次印刷

787mm × 1092mm 1/16·13.5印张

印 数：0 001 - 5000册

定 价：25.00元

前　言

ARX(实时运行扩展)作为继AutoLISP、ADS后的第三代开发工具，采用全新的面向对象编程技术，使对AutoCAD的开发从形式到内容均发生了巨大的变化，受到了AutoCAD用户和程序开发人员的一致欢迎。ARX环境下的开发技术代表了以PC机为硬件平台的CAD应用软件最先进的开发技术，掌握和应用该技术开发基于AutoCAD的专业软件对于众多科研人员和工科学生来说，无疑是十分必要和重要的。然而，由于目前有关ARX编程的参考书籍实在少见，加之涉及到AutoCAD的深入开发、面向对象的C++编程以及二者的有机组合等综合技术，使掌握ARX编程技术变得困难。因而本书的出版具有重要的实用价值。

作者在认真钻研ObjectARX开发工具包、深入编程实践和结合CAD课题研究开发的基础上，完成了书稿的撰写工作。本书全面地阐述了ARX应用程序的体系结构、AutoCAD数据库的管理机理及数据库对象的生成、检索和编辑的程序算法，详尽介绍了利用微软基础类库MFC和ARX编程的结合实现不同类型交互界面的可视化技术、程序结构、设计步骤和具体实现方法，给出了在工程中实用的专业开发技术和三维实体生成及编辑的ARX编程技术。同时，通过典型实例分析说明了ARX开发环境提供的常用类及其成员函数的用法和在CAD软件二次开发中的具体应用以及ARX程序编译链接的方式和操作步骤。

本书的主要特色是论述层次清晰、分析深入浅出、解释通俗易懂、举例简明实用。书中的所有实例具有广泛的代表性，涉及许多工程技术人员共同或各自关心的问题。所有实例经过精心设计，并紧密结合工程实际。在源程序的主要代码中均以注释的方式给出了简要的说明，非常方便阅读和理解。所有程序均通过上机调试和运行，读者可以直接在编程实践中引用或借鉴。

本书涉及到AutoCAD 的高级开发和Visual C++ 编程技术，读者在阅读本书之前应具备这方面的基础知识。另外，生成ARX应用程序需要32位Windows平台及ARX SDK2.02所提供的库函数和头文件的支持，可采用Visual C++ 4.2或Visual C++ 5.0进行编译和链接。生成的应用程序可在AutoCAD R14或MDT3.0 环境运行。书中的实例也可在AutoCAD 2000环境运行，但需要用ObjectARX 3.0开发环境提供的库函数和头文件的支持，采用Visual C++6.0编译和链接工具。

本书由南京航天航空大学CAD/CAM工程研究中心主任、博士生导师周儒荣教授主审，该中心的张丽艳副教授认真审读了全书。他们提出了许多宝贵意见，在此表示诚挚的感谢。

本书在编写过程中得到了所在学校有关领导和同事的大力支持，机械工业出版社的编辑为本书的顺利出版给予了热情帮助和具体指导，并做了大量的工作，在此谨向他们表示由衷的谢意。

ARX编程技术涉及的内容和知识非常广泛，由于作者水平所限，难免有谬误之处，欢迎读者批评指正。

编著者

1999年6月

目　录

第1章　ARX应用程序设计概述

1.1　AutoCAD开发环境的发展和ARX应用程序

1.1.1　概述

AutoCAD是美国Autodesk公司著名的CAD软件，它采用开放式的体系结构允许用户或二次开发商扩充新的功能和设计各种应用程序。随着系统功能的逐渐增强和版本的不断升级，提供了一系列开发环境和工具。1985年6月推出的AutoCAD2.17版本选用AutoLISP作为内嵌语言，向用户提供了用AutoLISP设计应用程序的二次开发环境。AutoLISP是一种解释型语言，主要用来修改和扩充AutoCAD命令及系统菜单、设计对话框驱动程序、实现对图形库的直接访问和修改。这是Autodesk公司提供的第一代开发环境。

AutCAD系统的第二代开发环境是R11版本提供的ADS(AutoCAD Development System)开发系统。该系统实际上是向用户提供了用C语言编写应用程序的设计环境。ADS应用程序用C语言编写，除了可以使用标准C函数以外，还可以使用对AutoCAD进行操作的ADS函数。对于AutoCAD来说，ADS应用程序等价于用AutoLISP编写的应用程序，不能单独执行，只能作为一组外部函数被AutoLISP装入和调用。在AutoCAD2000中不再支持ADS开发环境，ADS已完全被ObjectARX所取代。

AutCAD系统的第三代开发环境和工具包括ObjectARX、VBA(Microsoft Visual Basic for Applications)和Visual LISP等。在AutoCAD R13、R14和AutoCAD2000中，用户可以利用ObjectARX环境的支持，采用面向对象的C++语言开发ARX(AutoCAD Runtime eXtension)应用程序。在AutoCAD R14.01以上版本中VBA已作为标准安装组件。利用AutoCAD对VBA的支持，用户可开发VBA应用程序，使用ActiveX 对象。Visual LISP是AutoLISP的换代产品，其主体部分是一个ObjectARX应用程序。在AutoCAD2000中Visual LISP已内置于系统中，编译后的Visual LISP程序(扩展名为vlx)不再需要绑定LISP引擎。

1.1.2　AutoLISP和ADS应用程序与AutoCAD的通讯

用AutoLISP和ADS设计的应用程序以及AutoCAD系统本身实际上都是一个独立的进程(process，操作系统把系统中所运行的每个应用程序称为一个进程)，它们之间的通讯是通过IPC(InterProcess Communication，进程间通讯)机制来实现的。即AutoLISP应用程序作为一个独立的进程，与AutoCAD系统之间利用IPC机制通讯；而ADS应用程序作为一个独立的进程首先利用IPC机制与AutoLISP通讯，然后再通过AutoLISP实现与AutoCAD的通讯。AutoLISP和ADS应用程序都只能使用静态的AutoCAD命令集和系统提供的结构化函数库，因而在程序运行速度和功能上受到一定的限制。

1.1.3　ARX应用程序

在ObjectARX环境下开发的程序称为ARX应用程序(ARX application)。ARX应用程序不

再是一个独立的进程，而是一个动态链接库。因此，ARX程序与AutoCAD在同一地址空间运行，能够直接利用AutoCAD的内核代码，直接访问AutoCAD的数据库、图形系统及几何造型核心，在运行期间实时扩展AutoCAD具有的类及其功能，建立与AutoCAD本身的固有命令操作方式相同的新命令。ARX应用程序采用了与AutoLISP和ADS完全不同的运行机制，使运行速度大大提高和程序功能大大增强。此外，开发ARX应用程序还可以充分利用Windows的资源、微软的基本类库MFC(Microsoft Foundation Class)和先进的Visual C++可视化编程语言和工具，方便、高效地设计具有典型Windows 风格的CAD应用程序。

1.2 ARX 应用程序开发环境介绍

开发ARX应用程序需要ObjectARX提供的ARX SDK(Software Development Kit,软件开发工具包)。ARX SDK工具包主要提供了开发ARX应用程序所需的库文件、头文件、程序设计示例和有关使用说明。在AutoCAD不同版本运行的ARX应用程序需要不同的ObjectARX开发环境的支持和各自的编译链接工具，下面分别介绍。

1.2.1 AutoCAD R13 的ARX应用程序开发环境

开发在AutoCAD R13环境运行的ARX应用程序需要ARX SDK1.0、1.1(R13C4版本)开发工具包，采用Visual C++ 2.1、2.2或Visual C++ 4.X作为编译和链接工具。

1.2.2 AutoCAD R14 的ARX 应用程序开发环境

开发在AutoCAD R14 环境运行的ARX应用程序需要32位Windows平台及ARX SDK2.02所提供的库函数和头文件的支持。另外，还需配置Visual C++ 4.2或Visual C++ 5.0编译器。

在Windows 95或98环境下运行Obarxsdk.exe自解压文件，生成ARX SDK 2.02所提供的库函数、头文件、详细文档和程序实例。在缺省安装目录状态下，生成objectarx子目录和9个下级目录，其结构如下所示:

```
c:\objectarx\arxlabs
        \classmap
        \docs
        \docsamps
        \inc
        \lib
        \redistrib
        \samples
        \utils
```

arxlabs子目录包含lb01~lb10十个下级目录，每个下级目录包含一个ARX应用程序接口(API)的示例。

classmap子目录中有一个名为Classmap.dwg的AutoCAD图形文件，该图表示了ARX类的层次关系。

docs子目录存放ARX的在线帮助文件，包括ObjectsARX开发指南、ObjectsARX参考手册、ADSRX开发指南以及ObjectARX SDK Readme文件等。

docsamps子目录包含从ARX开发指南中摘录的每个程序实例(包括源程序代码和说明文件)构成的下级子目录。

inc目录包含了生成ARX应用程序所需的头文件。

lib目录包含了ARX应用程序所需的九个库文件。

redistrib子目录包含了运行ARX应用程序所需的一组动态链接库。开发者应将应用程序开发所需的动态链接库复制到AutoCAD能够查找的目录，在发布时将ARX应用程序与所需的动态链接库一起打包。

samples子目录的各下级目录中为ARX应用程序和实例，包括源程序代码和readme文件。其中最有意义的一组ARX应用程序实例放在polysamp 子目录中。

Utils目录包含ARX扩展的应用程序子目录，包括边界描述(brep)和组合文档的存储(istorage)。每个应用程序子目录中又包含inc、lib和实例三个下级子目录。

1.2.3 AutoCAD 2000 的ARX 应用程序开发环境

开发在AutoCAD 2000 环境运行的ARX应用程序需要Windows NT4.0、Windows 95、Windows 98或Windows 2000平台以及ObjectARX 3.0开发环境提供的库函数和头文件的支持，用Visual C++ 6.0作为编译和链接工具。

在ObjectARX 3.0开发环境中进一步扩充了ARX应用程序的编程接口功能：开发者可以从ObjectARX 的AcDb基类中派生AutoCAD的所有对象、支持对DXF文件图形预览的编程访问以及文件局部打开过程的编程控制。并提供多文档、创建和控制多重布局的开发和访问URL信息接口等功能。

1.3 ARX 应用程序的基本结构

ARX应用程序是一个动态链接库(DLL)，虽然程序代码可用C++或C编写，但是，它与ADS应用程序的一个显著区别是AutoCAD通过其动态链接库的入口函数acrxEntryPoint()调用ARX模块，而不再需要调用主函数main()。

1.3.1 ADSRX和ARX程序实例

下面通过两个简单的实例来分析ARX应用程序的基本结构。

例1-1 首先讨论在屏幕上生成一个矩形的参数绘图程序。源程序如下：

```
//************************************************
// ADSRX应用程序示例: arx1-1.cpp
//************************************************
//头文件包含部份
#include <stdio.h>
#include "rxdefs.h"
#include "adslib.h"
#include "adesk.h"
//函数说明部份
static int loadfuncs();
int arx11();
extern "C"
{
void initModule();
AcRx::AppRetCode acrxEntryPoint(AcRx::AppMsgCode msg, void* );
```

```
}
// 初始化函数
void initModule()
{
}
AcRx::AppRetCode acrxEntryPoint(AcRx::AppMsgCode msg, void * )
{switch(msg) {
   case AcRx::kInitAppMsg:
        initModule();
        break;
   case AcRx::kLoadADSMsg:
        loadfuncs();
        break;
   //.....
   default:
        break;
   }
   return AcRx::kRetOK;
}
// 定义外部函数
static int loadfuncs()
{if (ads_defun("C:arx11", 0) == RTNORM)
 {
     ads_regfunc(arx11, 0);
     return 1;
  }
 else
    return 0;
}
// 用户程序主体函数
int arx11()
{
        ads_point pt1,pt2,pt3,pt4;
        ads_getpoint(NULL,"\n输入第一点: ",pt1);
        ads_getpoint(pt1,"\n输入第二点: ",pt2);
        ads_getpoint(pt2,"\n输入第三点: ",pt3);
        ads_getpoint(pt3,"\n输入第四点: ",pt4);
        ads_command(RTSTR,"pline",RTPOINT,pt1,RTPOINT,pt2,
            RTPOINT,pt3,RTPOINT,pt4, RTSTR,"C",0);
        return AcRx::kRetOK;
}
```

例1-1的程序的结构与ADS应用程序基本相同，其主要差别是用acrxEntryPoint()函数取代ADS应用程序的main()函数。外部函数的定义(ads_defun())、注册(ads_regfucn())和用户程序的主体部分仍使用ADS函数。由于尚未用到ObjectARX的许多新特性，这类程序仅仅是作为ADS向ARX应用程序的过渡，故称为ADSRX应用程序。

头文件adesk.h 包含了ARX程序设计环境的标准定义，xdefs.h定义ARX应用程序通过acrxEntryPoint()函数与AutoCAD的交互，adslib.h包含了ADS的一般定义。

acrxEntryPoint()函数不仅作为ARX应用程序与AutoCAD之间通讯的入口函数，而且也是

ARX应用程序传递消息和向AutoCAD返回状态码的途径。

acrxEntryPoint() 函数说明如下:

```
extern "C"
AcRx::AppRetCode  acrxEntryPoint(AcRx::AppMsgCode  msg,void* pkt);
```

msg: 表示从ARX内核传递到应用程序的消息。

pkt: 回调数据信息指针。

AppRetCode : 返回AutoCAD的状态码。

在acrxEntryPoint()函数定义中, 用switch 语句处理来自AutoCAD的消息, 执行与消息有关的相应动作, 并返回一个整数的状态码。

例1-2 这是按ARX应用程序基本结构编写的一段程序, 其功能与例1-1完全相同。程序代码如下:

```
//************************************************
//  ARX应用程序示例: arx1-2.cpp
//************************************************
//头文件包含部分
#include <aced.h>
#include <adslib.h>
//函数说明部分
int arx12();
//调用ARX应用程序
void CallarxApp()
{
arx12();
}
// 初始化函数
void initApp()
{
   acedRegCmds->addCommand("ARX12",
                           "ARX12",
                           "ARX12",
                           ACRX_CMD_MODAL,
                           CallarxApp);
}
//卸载函数定义
void unloadApp()
{
  acedRegCmds->removeGroup("arx12");
}
//入口点函数定义
extern "C" AcRx::AppRetCode
acrxEntryPoint(AcRx::AppMsgCode msg, void* pkt)
{
      switch (msg) {
      case AcRx::kInitAppMsg:
      acrxDynamicLinker->unlockApplication(pkt);
            initApp();
            break;
```

```
        case AcRx::kUnloadAppMsg:
            unloadApp();
        }
        return AcRx::kRetOK;
}
// 用户程序主体函数定义
int arx12()
{
        ads_point pt1,pt2,pt3,pt4;
        ads_getpoint(NULL,"\n输入第一点: ",pt1);
        ads_getpoint(pt1,"\n输入第二点: ",pt2);
        ads_getpoint(pt2,"\n输入第三点: ",pt3);
        ads_getpoint(pt3,"\n输入第四点: ",pt4);
        ads_command(RTSTR,"pline",RTPOINT,pt1,RTPOINT,pt2,
          RTPOINT,pt3,RTPOINT,pt4, RTSTR,"C",0);
        return AcRx::kRetOK;
}
```

1.3.2 程序的基本结构分析

例1-2所示的程序由下述几部分组成:

1) 文件包含和函数说明。

2) AutoCAD调用ARX应用程序的入口函数acrxEntryPoint()定义。

3) 加载ARX程序时的初始化函数定义。

在AutoCAD环境加载ARX程序时通过入口函数调用initApp()函数，其功能为调用注册用户定义的AutoCAD命令函数addCommand()。该函数为ARX库中定义的acedRegCmds类成员函数，第一个参数表示AutoCAD的命令组名，组名由用户定义或用系统已定义的名字；第二个参数表示通用命令名，指容易被不同地区和国家接受的名字，一般用英语表示；第三个参数表示为本地命令名可用本国语言表示，如用中文表示。用这种形式为用户定义新命令提供了灵活性，但在多数情况下两种命令用相同的字符串表示；第四个参数表示用户定义的命令类型，ACRX_CMD_MODAL表示定义的命令不能作为透明命令使用，定义透明命令用ACRX_CMD_TRANSPARENT参数；第五个参数表示函数指针，指向执行命令时所调用的函数。如例1-2中表示注册了一个命令组名、通用命令名和本地命令名均为“ARX12”的应用程序，当用户在AutoCAD环境装入和注册成功后，输入“ARX12”命令，将执行arx12()函数.

4) 卸载ARX程序的函数定义。

卸载函数unloadApp()调用removeGroup()函数释放用addCommand()函数定义的命令组。在缺省时应用程序载入后为锁定状态，不能用卸载函数卸载。要使装入的应用程序能够卸载，在处理AcRx::kInitAppMsg消息时应设定为解锁状态，如例1-2中的语句: acrxDynamicLinker->unlockApplication(pkt);

5) 用户程序主体函数定义。

该段代码是用户编写的主要部分，本例为一种最简单的情况，尚未用到C++和ARX的新特性。

上面介绍的两个实例虽然功能完全相同，但反映了ADSRX和ARX应用程序在结构上的差异，前者的结构适用于ADS应用程序向ARX应用程序的移植，后者为ARX应用程序的典型结

构，本书主要介绍后一种结构形式。

1.4 ARX 应用程序的生成方法

由于ARX应用程序是一个动态链接库，因此要正确生成程序模块，必须有一个相应的模块定义文件。在该文件中必须定义AutoCAD调用的入口函数名，下面列出了例1-1的模块定义文件内容：

```
LIBRARY    arx11
DESCRIPTION 'Arxads Sample  Program'
EXPORTS
    acrxEntryPoint
   _SetacrxPtp
    acrxGetApiVersion
```

在模块定义文件的LIBRARY段定义动态链接库的库名，该库名应与ARX应用程序名相同，不区分大小写。在EXPORTS段定义输出函数名acrxEntryPoint()、_SetacrxPtp()和acrxGetApiVersion()。第一个为AutoCAD调用ARX应用程序的入口函数名，后两个是在rxapi.lib库中生成的函数。对于不同的ARX应用程序，模块定义文件只需修改第一行“LIBRARY”后的库名和第二行中的说明文字，其余内容可以照写。

ARX的C++源程序需要经过编译和链接才能在AutoCAD环境运行。下面介绍两种方式。

1.4.1 命令行方式的编译和链接

这是一种最简单的编译和链接方式，主要用于生成AutoCAD R13的ADSRX程序。可选用Visual C++ 2.1以上版本的编译器，用其BIN子目录中的NMAKE.EXE完成编译和链接。

首先，假设存放C++源程序和生成的可执行程序的目录为d:\arx;Visual C++ 4.0系统在d:\MSDEV目录；AutoCAD 13 for Windows在d:\R13目录；

其次，将R13的\adsrx\sample子目录中生成ARX应用程序可执行文件的两个样本文件mkarx.bat和mkarx.mak复制到设定的用户目录d:\arx。然后，用文本编辑工具按当前的目录情况对mkarx.bat文件进行修改，如：

```
@echo off
if (%1) == () goto noname
:: 设置存放C++源程序名(不需要扩展名)的变量
set INFILE=%1
:: 保存原来的路径
set OLDPATH=%PATH%
:: 设置变量MSVCPATH=Visual C++目录名(此处按Visual C++的实际目录进行修改)
set MSVCPATH=d:\msdev
set PATH=%MSVCPATH%\bin
:: 设置变量ARXPATH=d:\arx目录名(根据arx文件的实际目录进行修改)
set ARXPATH=d:\arx
:: 设置变量ADSINC=d:\R13\ADS子目录名(根据ads头文件的实际目录进行修改)
set ADSINC=d:\r13\ads
set ADSRXINC=%arxpath%\inc
set ADSRXLIB=%arxpath%\lib
set INCLUDE=%msvcpath%\include;%adsinc%;%adsrxinc%
set LIB=%msvcpath%\lib;%adsrxlib%
```

```
nmake -nologo -s -k -f mkarx.mak
::.....
goto end
:noname
echo syntax is
echo mkarx [filename with no extension]
:end
```

若源程序扩展名为.cc，则mkarx.MAK文件不需要任何修改；若源程序扩展名为.cpp,则需进行修改(将 * ext=cpp前的 * 号删去，在ext=cc前加上 * 号)。

在Windows 95环境生成arx可执行文件的操作步骤为:

1) 启动Windows 95后，选MS-DOS方式，进入MS-DOS窗口。

2) 进入arx子目录。

3) 输入：mkarx C++源程序名(不含扩展名)，如：d:\arx>mkarx arx11

如果操作成功即可生成arx11.arx 文件。注意，编译和链接只能在Windows的MS-DOS窗口中进行，不能直接在DOS环境进行。

1.4.2 在Visual C++ IDE环境的编译和链接

在Visual C++ IDE(Integrated Develop Environment) 环境下可以完成ARX应用程序的编制、调试、编译和链接工作。与命令行编译和链接方法相比，这种方法更先进实用，多用于生成在AutoCAD R14和AutoCAD 2000环境运行的ADSRX或ARX应用程序。但是，采用这种方式要求用户必须正确设置编译和链接选项，否则即使编译链接成功也不能保证ARX程序在AutoCAD环境顺利运行。

假设C++源程序如例1-2所示，文件名为arx12.cpp，相应的模块定义文件名为arx12.def，生成的ARX应用程序名为arx12.arx。程序文件存放的路径为e:\arxbook\sample\arx12,ARX开发环境提供的头文件和库文件分别存放在e:\arxbook\inc和e:\arxbook\inc子目录，下面说明在Visual C++5.0 IDE环境下生成ARX应用程序的关键技术和操作步骤:

1) 创建项目文件。

进入Visual C++ 5.0 集成开发环境Microsoft Developer Studio，选菜单File/New…选项，进入图1-1所示的对话框。

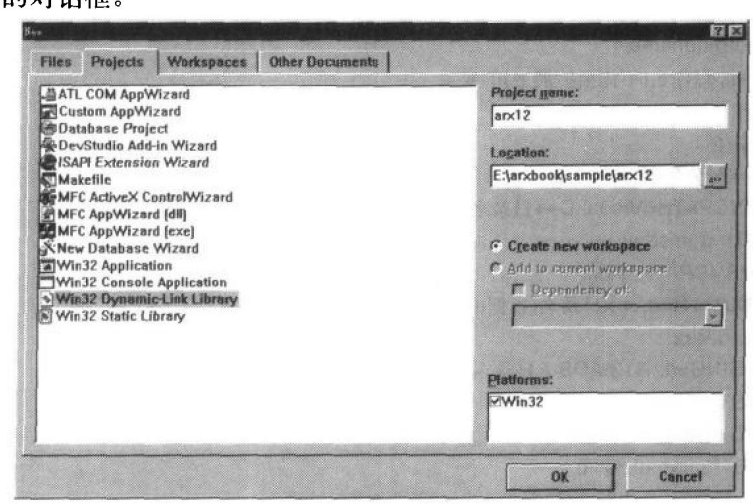

图1-1 New对话框

在New对话框中选“Projects”选项卡，选择创建动态链接库“Win32 Dynamic-Link Library”项。在Project name：输入框中输入项目文件名，在Location：输入框中输入路径。单击OK按钮，完成项目文件框架的建立。

选择Project/Add To Project/Files…,在FileView 中插入ARX源程序文件、模块定义文件。其结果如图1-2所示。

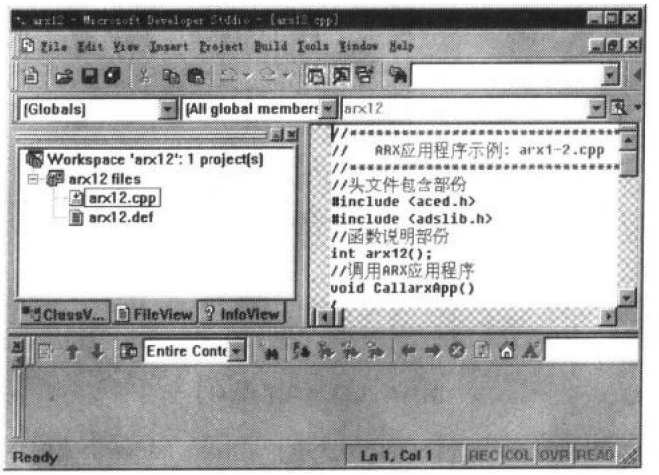

图1-2 Visual C++ 5.0 IDE

2) 设置编译和链接选项。

C/C++选项设置：选Project/Setting…，在类型(category)弹出式菜单中选择预处理器(Preprocessor)选项，在additional include directories：编辑框中输入ARX包含文件的子目录,如图1-3所示。

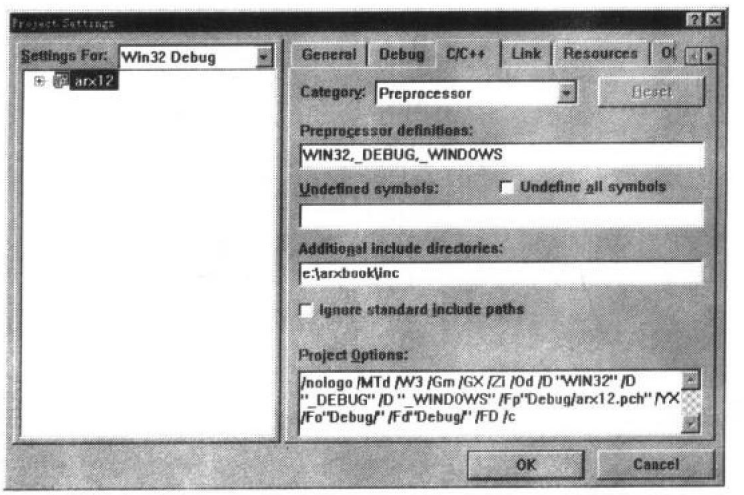

图1-3 C/C++选项设置

link选项设置：在选项设置窗口选Link选项卡，在类型(category)弹出式菜单中选择常规(General)选项；在Output file name:编辑框中输入ARX应用程序的文件名——arx12.arx(扩展名arx)；在Object/library modules：编辑框中插入链接器包含的ARX库文件——acad.lib(数据库访问库)、acedapi.lib(命令扩展库)、rxapi.lib(运行扩展库)和libacge.lib(通用几何库)等需要的

ARX库(由Microsoft Developer Studio自动生成的库名不用更改)。每个库名应包含路径，各库名之间用空格分开，见图1-4。

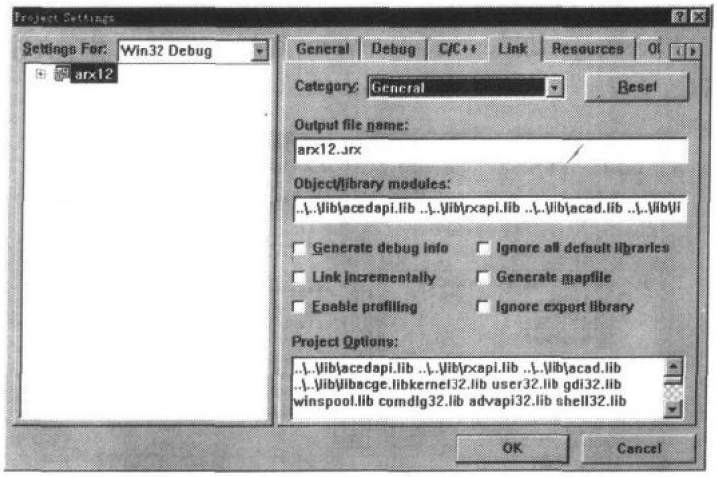

图1-4 link选项设置(1)

在类型(category)弹出式菜单中选择Output选项，在base address:编辑框中输入0x1c000000; Entry-point Symbol：编辑框中输入DllEntryPoint@12，如图1-5所示。

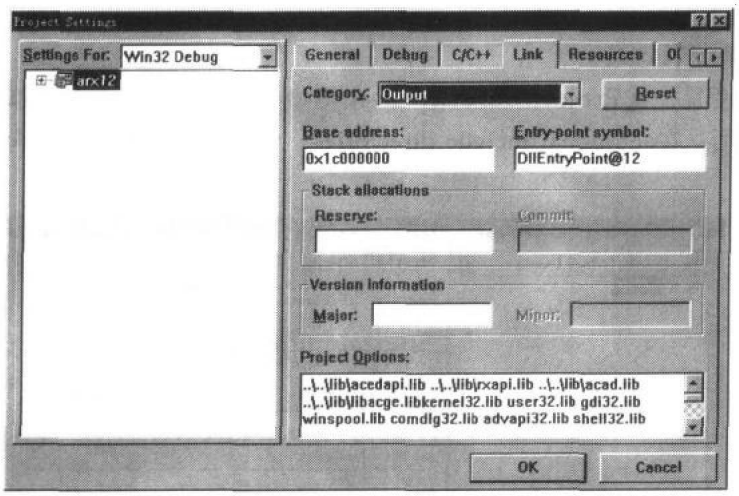

图1-5 link选项设置(2)

3) 生成ARX程序。

选择Build/Build arx12.arx或Build All生成扩展名为ARX的程序文件：arx12.arx

1.5 ARX 应用程序的装入和运行

ARX应用程序在运行之前必须进入AutoCAD环境，然后在该环境装入内存。本节介绍四种基本装入方式。

1. 用AutoLISP的arxload函数装入

格式：(arxload <文件名>)