Bianchengshiliyingyongzhinan

C++Builder 5.0

编程实例应用指南

系统程序员开发宝典

主编 / 施红芹

逐步掌握编程理论

快速提高编程技巧

航空工业出版社

C++Builder 5.0 编程实例应用指南

主 编 施红芹

编 委 彭 海 向文光

徐进明 郭秀英

航空工业出版社

内 容 提 要

本书是C++Builder 5.0编程的实例详解。全书共分十七章，每一章深入全面地介绍了一个高级编程实例以及与之相关的应用技巧。实例涵盖了网络、数据库、多媒体以及系统操作等各个方面的内容。其中网络方面介绍了聊天程序的实现、局域网下的实时语音传输；数据库方面介绍了多层数据库的设计、Web数据库服务程序；多媒体方面介绍了图像的平滑移动和演播、字幕效果的实现；系统方面介绍了进程通信、注册表操作等实例。丰富的内容，无论从深度还是实用上来说，可以说覆盖了C++ Builder 5.0编程的热点和难点。深刻的分析，翔实的注解将使读者无师自通，在较短的时间内迅速掌握C++ Builder 5.0编程的精髓。

图书在版编目（CIP）数据

C++Builder 5.0编程实例应用指南 / 施红芹主编.
北京：航空工业出版社，2000.5
ISBN 7-80134-617-3

Ⅰ.C… Ⅱ.施… Ⅲ.数据库系统-软件工具，Builder-程序设计 Ⅳ.TP311.56

中国版本图书馆CIP数据核字（2000）第02688号

JS470/30

航空工业出版社出版发行
（北京市安定门外小关东里14号 100029）
北京云浩印刷厂印刷　　全国各地新华书店经售
2000年7月第1版　　2000年7月第1次印刷
开本：787×1092　1/16　　印张：21.5　　字数：458千字
印数：1-6000　　定价：29.80元

前　言

C++Builder 是 Inprise 公司推出的一个强大的开发工具。虽然目前在与 VC 的竞争中还没能取得压倒性的优势，但是我们完全有理由相信，C++Builder 不久将取得和 Delphi 一样的成功，甚至会超出 Delphi 所创造的辉煌。Inprise 公司继 C++Builder 1.0 之后，在一年多的时间里先后推出了 4.0 版和 5.0 版，每版都有很大的改进，性能上也有了明显的提高。C++Builder 5.0 引入了如下的新特点和增进功能。

（1）ADOExpress

只有 C++Builder 5.0 企业版和专业版才提供此项组件。ADOExpress 采用 Microsoft ActiveX Data Objects（ADO）技术，提供和各种数据库格式的存储接口。ADOExpress 直接采用 ADO 接口，而不需要 Borland Database Engine（BDE）数据库引擎。新增的组件包括：TADODataSet、TADOTable、TADOQuery、TADOStoredProc、TADOConnection、TADOCommand 和 TRDSConnection 等。

（2）InterBase Express （企业版和专业版提供）

InterBase Express（IBX）通过运行 InterBase API 实现对 InterBase 数据库的本地存取。因为该组件直接引用 InterBase API，而不需要 BDE。IBX 是一组直接通过 Interbase API 存取 Interbase 数据库的组件类，和标准的 C++Builder 数据库组件在目的和用途上是并行的，因此对传统的数据库接口组件很熟悉的用户能够轻松地掌握新组件的使用。

（3）MIDAS enhancements （企业版提供）

MIDAS enhancements 不仅支持多层数据库（MIDAS），而且支持远程数据模块和新的 InternetExpress 组件，从而创建可以利用 MIDAS 程序服务器通过浏览器访问数据库的网络应用程序。

（4）CORBA 升级

CORBA 已经升级，可以运用 VisiBroker for C++ ORB version 4.0 工作。

（5）IDE 增强

IDE 集成开发环境得到增强。比如，自定义桌面设置，对象管理器的属性分类，将图片引入到对象管理器的下拉菜单中，代码编辑器功能增强，在代码编辑器中提供函数分隔线，窗体可作为文本保存，数据模块设计的增强等。

（6）工程制作和工程管理增强。

工程管理器的新增特性使文件、节点管理和设置简单化，这些新增性能包括：头文件属性增强，工程打包，新的工程文件格式等。

（7）编译器的增强（所有版本）

采用后台编译，使用户能够在编译的同时进行代码编写和窗体设计等工作。编译方面的新特性提高了 C++Builder 5.0 与微软的 Microsoft Visual C++的兼容性，在编译过程中增强了对扩展错误的报告和显示。

（8）新的调试特点。

包括新的调试显示，提供 FPU 窗口显示 CPU 浮点运算的内容，增强了断点设置功能和断点组，增加了一些调试命令和设置，增强了调试的可用性。

（9）可视化控件库（VCL）的增强

根据开发者的需求，集成了很多新的技术，提供大量新的对象、属性和事件。包括：通用组件的升级、网络组件增强、可自定义控件拖曳事件、控制面板向导组件、新增帧的功能，能够分割和嵌套在别的窗体或帧中。

（10）COM 和 COM+的增强

C++Builder 5.0 开始支持 COM+，COM 服务器可作为可视化组件被导入执行，而且提供新的 COM 对象向导，简化编程工作。

（11）新增的工具和向导

增加了两个新工具、三个新向导和一个新的工具宏定义。比如应用程序向导和新的工具宏等。

（12）其他的变化和增强

比如 ActiveX 的增强和性能调整。

C++Builder 5.0 的快速开发能力，优异的性能，得到进一步改进的中文支持环境，对最新技术的强大支持，以及开放式的的编程体系结构等特性具有无可比拟的吸引力。在中国市场，C++Builder 5.0 同样受到重视，对它进行讨论和交流的各种网站也发展迅速，许多程序员纷纷从其他开发工具转到 C++Builder 5.0 上来。

由于 C++Builder 5.0 的推出时间比较短，所以目前资料相对短缺，特别是好的参考书更是少之又少。为了帮助广大 C++Builder 5.0 爱好者快速、深入地掌握其编程的方法和技巧，编者编写了本书，希望与大家分享编程的心得和乐趣。

本书紧扣 C++Builder 5.0 编程的根本特点，围绕开发中的热点和难点进行介绍，力图向读者展示 C++Builder 5.0 编程的最精彩世界。全书共分十七章，以实例的形式详细介绍了用 C++Builder 5.0 进行网络、数据库、多媒体以及系统等各个方面编程的高级技巧。其中网络方面介绍了聊天程序的实现、局域网下的实时语音传输、浏览器的编写；数据库方面介绍了多层数据库的设计、Web 数据库服务程序；多媒体方面介绍了图像的平滑移动和演播、字幕效果的实现；系统方面介绍了多线程、进程通信、注册表操作等实例。书中所有实例均是编者多年开发经验的总结和升华，加上详实的注解，无论对初学者还是有一定经验的朋友都是非常实用的。特别是对于有一定经验的朋友，在学习过本书之后，将会有一个质的飞跃，并迅速而快捷地掌握用 C++Builder 5.0 进行开发的精髓。由于每个实例都有一定的深度，为了方便读者参考，在每章的结尾列出了主要程序清单。

有人说，聪明的程序员用 Delphi，真正的程序员用 C++。编者认为，真正聪明的程序员用 C++Builder，因为它同时具备两者的优点。

愿读者在 C++Builder 的世界里尽情游翔！

本书由施红芹主编。参加本书编写和制作的人员还有彭海、向文光、徐进明、郭秀英等。由于编者水平有限，加之时间仓促，书中难免有错误与不妥之处，恳请广大读者批评指正。

编 者

2000 年 5 月

目 录

第 1 章　在程序中动态创建 VCL 组件

VCL 是 Visual Component Library 的缩写，是 C++Builder 的一个非常重要的组成部分，可以说是 C++Builder 的精华之一。正是通过 VCL 组件，C++Builder 才实现了真正的可视化编程。从本质上来说，VCL 是一个完整的、开放式的类库，它与微软的 MFC 和原 Borland 的 OWL 相似。但是 VCL 比前两者都要优秀，这不仅在于它的可视化，更重要的是它的开放性。

1.1　动态创建 VCL 组件

用 C++Builder 编程，可以使很多常见的工作都能通过 VCL 来完成，这样就可以大大节省时间，提高开发效率。在通常情况下，用户在 C++Builder 的集成开发环境 IDE 中用拖放的方式生成 VCL 组件，随后设置它们的属性和编写方法等。但是在一些特殊情况下，用户可能需要在程序运行的时候生成某些 VCL 组件，这就需要动态创建 VCL 组件。

动态创建 VCL 组件的有一个好处，就是可以高效利用系统资源。因为如果是在 IDE 中静态创建的话，那么 VCL 组件在程序运行时就要装入内存，直到程序结束才被释放；而动态创建就可以在需要时生成，用完以后马上删除，可以节省内存。下面就通过实例详细阐述 VCL 组件的动态创建。

1.1.1　构造与析构

前面已经提及，VCL 的本质是一个类库，因此它有类的基本特性。要动态创建 VCL 组件，就要用到 VCL 的构造函数。VCL 的构造函数与普通的 C++类没有什么区别，只是一般它都有 AOwner 这个参数。正是这个 AOwner 参数决定了 VCL 组件的析构与普通的 C++类有些不同，那就是 VCL 组件具有自动释放的功能。也就是说，用户在用 New 创建了一个 VCL 组件之后，不必对它进行释放，在 AOwner 参数指向实例析构时，会自动析构它所指向的所有子实例。当然，如果用户想自己来释放一个 VCL 实例，也是可以的，用 delete 关键字即可，与普通的 C++语法没有任何区别。

下面开始具体的工作。新建一个名为 DyMenu 的工程，Form1 的文件存为 DMenuMain.cpp。

将 Form1 的 Caption 改为 Dynamic Menu Demo。在 Form1 上放置两个 Button，一个命名为 Simple，Caption 设置为 Simple Menu；另一个命名为 Complex，Caption 设置为 Complex Menu，保存所做的工作。整个 Form1 如图 1-1 所示。

这个例子将动态创建两个菜单，一个较简单，一个较复杂。通过这个例子，读者将掌握动态创建 VCL 组件的全部技巧。

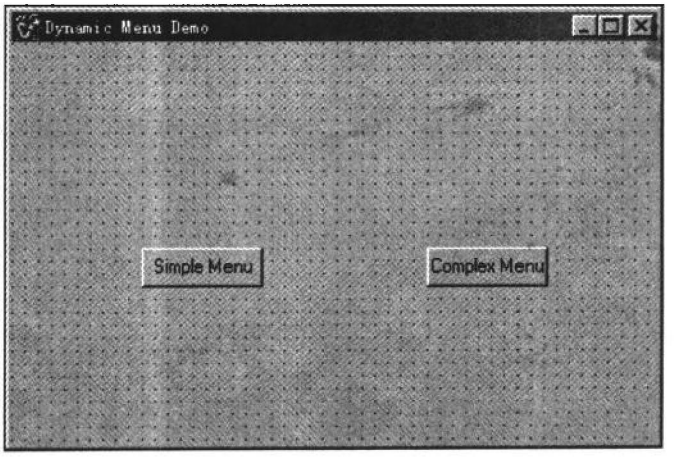

图 1-1 表单 Form1

在 DMenuMain.cpp 的头文件中包括菜单声明的头文件：

```
#include <Menus.hpp>
```

在 TForm1 内声明私有成员：

```
private:    // 用户声明
    Menus::TMainMenu *MainMenu;
```

此处的 MainMenu 就是后面将要动态创建的主菜单。

下面在按钮 Simple 的 OnClick 事件中添加如下代码：

```
//---------------------------------------------------------------------------
void __fastcall TForm1::SimpleClick(TObject *Sender)
{
    //删除原来的菜单
    delete MainMenu;
    //动态生成一个新的菜单
    MainMenu=new TMainMenu(this);
    //向新菜单添加菜单项

    TMenuItem* MenuItemF=NewItem("File",NULL,false,true,NULL,0," MenuItemF ");

        TMenuItem* MenuItemO=NewItem("Open...",
            ShortCut(Word('O'),TShiftState() << ssCtrl),
            false,
            true,
            OpenFile,
            0,
            "MenuItemO");
        MenuItemF->Add(MenuItemO);
```

```
    TMenuItem* MenuItemS=NewItem("Save As...",
        ShortCut(Word('S'),TShiftState() << ssCtrl),
        false,
        true,
        SaveFile,
        0,
        "MenuItemS");
    MenuItemF->Add(MenuItemS);

    TMenuItem* MenuItemNull=NewItem("-",
        0,
        false,
        true,
        NULL,
        0,
        "MenuItemNull");
    MenuItemF->Add(MenuItemNull);

    TMenuItem* MenuItemE=NewItem("Exit",
        ShortCut(Word('C'),TShiftState() << ssCtrl),
        false,
        true,
        ExitApp,
        0,
        "MenuItemE");
    MenuItemF->Add(MenuItemE);

MainMenu->Items->Add(MenuItemF);

TMenuItem* MenuItemHelp=NewItem("Help",NULL,false,true,NULL,0,"MenuItemHelp");

    TMenuItem* MenuItemAbout=NewItem("About...",
        0,
        false,
        true,
        About,
        0,
        "MenuItemAbout");
    MenuItemHelp->Add(MenuItemAbout);
```

```
    MainMenu->Items->Add(MenuItemHelp);

    //将新建的菜单指定为 Form 的菜单
    Menu=MainMenu;
}
//---------------------------------------------------------------------------
```

代码中首先删除原来的菜单，随后动态地新建一个菜单。在此读者也许要问，开始时 MainMenu 还没有创建，删除会不会出错。是的，删除一个没有赋值的指针是会出错的，因此应该在 Form1 的构造函数中增加一行：

```
MainMenu=NULL;
```

这样删除一个指向 NULL 的指针就不会出错了。

下面再来看 OnClick 事件中的代码：

```
MainMenu=new TMainMenu(this);
```

这一行生成一个主菜单，可以看到也是用 new 关键字，在参数中用了 this 指针（这里，this 指针就是指向 Form1）。因此如果不主动删除 MainMenu 指针，在 Form1 析构时，将自动释放 MainMenu。至此可得到一个空的主菜单。下面添加菜单项：

```
TMenuItem* MenuItemF=NewItem("File",NULL,false,true,NULL,0," MenuItemF ");
```

这一行用 NewItem 生成了一个菜单项。

专家指导：NewItem 的声明如下：

extern PACKAGE TMenuItem * __fastcall NewItem(const AnsiString ACaption, TShortCut AShortCut, bool AChecked, bool AEnabled, Classes::TNotifyEvent &AOnClick, Word hCtx, const AnsiString AName);

在 Menus.hpp 中可以找到此函数的定义。

所生成的菜单项标题为 File，没有定义快捷方式，AChecked 项为 false，菜单有效项 AEnabled 为 true；没有定义 OnClick 事件，菜单名字为 MenuItemF。

下面再向菜单项 MenuItemF 中添加子菜单项：

```
TMenuItem* MenuItemO=NewItem("Open...",
        ShortCut(Word('O'),TShiftState() << ssCtrl),
        false,
        true,
        OpenFile,
        0,
        "MenuItemO");
    MenuItemF->Add(MenuItemO);
    TMenuItem* MenuItemS=NewItem("Save As...",
        ShortCut(Word('S'),TShiftState() << ssCtrl),
```

```
        false,
        true,
        SaveFile,
        0,
        "MenuItemS");
    MenuItemF->Add(MenuItemS);

    TMenuItem* MenuItemNull=
        NewItem("-",
        0,
        false,
        true,
        NULL,
        0,
        "MenuItemNull");
    MenuItemF->Add(MenuItemNull);

    TMenuItem* MenuItemE=
        NewItem("Exit",
        ShortCut(Word('C'),TShiftState() << ssCtrl),
        false,
        true,
        ExitApp,
        0,
        "MenuItemE");
    MenuItemF->Add(MenuItemE);
```

这里向菜单项 MenuItemF 添加了 4 个子菜单项：MenuItemO、MenuItemS、MenuItemNull 和 MenuItemE。每一个子菜单都定义了自己的 OnClick 事件。

注意事项：所谓的菜单项和子菜单项其本质都是 TMenuItem，只是为了区分它们的层次而使用了不同的名字。因此，子菜单项还可以有自己的子菜单项，菜单就是这样一级一级向下延续的。

这里有必要解释一个函数：

```
extern PACKAGE TShortCut __fastcall ShortCut(Word Key, Classes::TShiftState Shift);
```

这个函数专门用来生成菜单项的快捷方式，在 Menus.hpp 中可以找到它的声明。上面 MenuItemO、MenuItemS 和 MenuItemE 等 3 个子菜单项都定义了快捷方式，依次为：【Ctrl＋O】、【Ctrl＋S】和【Ctrl＋C】组合键。

生成子菜单最关键的部分是定义该菜单项的 OnClick 事件。MenuItemO、MenuItemS、MenuItemE 的 OnClick 事件处理函数依次为：OpenFile()、SaveFile()、ExitApp()。这 3 个

函数的定义如下：

头文件声明：

```
void __fastcall OpenFile(TObject *Sender);
void __fastcall SaveFile(TObject *Sender);
void __fastcall ExitApp(TObject *Sender);
```

上述 3 个函数均为私有成员，函数体如下：

```
//---------------------------------------------------------------------------
void __fastcall TForm1::OpenFile(TObject *Sender)
{
    TOpenDialog *OpenDialog=new TOpenDialog(this);

    OpenDialog->Execute();
    delete OpenDialog;
    OpenDialog=NULL;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::SaveFile(TObject *Sender)
{
    TSaveDialog *SaveDialog=new TSaveDialog(this);

    SaveDialog->Execute();
    delete SaveDialog;
    SaveDialog=NULL;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::ExitApp(TObject *Sender)
{
    Close();
}
//---------------------------------------------------------------------------
```

这样就为子菜单项定义了完整的 OnClick 事件。

不难发现，在 OpenFile()和 SaveFile()两个函数中，又动态生成了两个对话框，分别是 OpenDialog 和 SaveDialog。其创建方法与动态创建菜单一样。TOpenDialog 和 TSaveDialog 的声明在 Dialogs.hpp 中，因此应该在 Form1 的头文件中加入一行：

```
#include <Dialogs.hpp>
```

接着向下看 SimpleClick 里的代码：

```
MainMenu->Items->Add(MenuItemF);
```