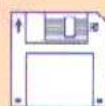


C++Builder

编程实例详解

段米毅 李罡 丁莉 编著



随书附实例软盘

全面介绍了使用 C++Builder 开发应用程序的方法

- IDE 集成开发环境
- 开发入门
- 菜单设计
- 事件处理编程
- 建立和管理工程
- 制作用户报表
- 文本编辑
- Internet 应用
- 设计窗体
- 代码编辑器
- 面向对象与 VCL 编程
- 设置工程参数和编译
- 构建 Internet 应用程序
- 图形处理
- 趣味程序精选

所附软盘包括书中所有实例的源程序



电子工业出版社

PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

TP312C
D8.5

C++ Builder 编程实例详解

段米毅 李 罡 丁 莉 编著



电子工业出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

内容提要

Borland 公司的 C++ Builder 是深受广大微型计算机软件开发人员欢迎的一种 C++ 语言的开发工具,其特点是功能强大,操作界面友好。使用 C++ Builder 开发 Windows 应用程序可以成倍地提高工作效率。

本书全面介绍了如何使用 C++ Builder 开发 Windows 应用程序。书中既介绍了有关 C++ Builder 的基本知识,又通过若干实例说明了使用 C++ Builder 开发不同范围应用程序的过程及方法。特别介绍了有关 Internet、图形处理以及游戏方面的多个实用的程序。

通过阅读本书,读者可以很快掌握 C++ Builder 的使用,并开发成功所需要的应用程序。

本书提供了大量实例,所有实例源程序均在本书随附的软盘中,供读者学习时使用,也可直接或稍加修改后使用。

书 名: **C++ Builder 编程实例详解**

编 著 者: 段米毅 李 罡 丁 莉

责任编辑: 陆伯雄 寇国华

印 刷 者: 北京牛山世兴印刷厂

出版发行: 电子工业出版社出版、发行 URL: <http://www.phei.com.cn>

北京市海淀区万寿路 173 信箱 邮编 100036 发行部电话 68214070

经 销: 各地新华书店经销

开 本: 787×1092 1/16 印张: 20 字数: 470 千字

版 次: 1999 年 1 月第 1 版 1999 年 1 月第 1 次印刷

定 价: 40.00 元

印 数: 5000 册

书 号: ISBN 7-5053-4939-2/TP.2422

凡购买电子工业出版社的图书,如有缺页、倒页、脱页者,本社发行部负责调换

版权所有·翻印必究

前 言

Borland 公司的程序设计工具以高效实用见长, C++Builder 是其精典之一。

C++Builder 是一种可视化、面向对象的 C++ 程序设计语言开发工具, 具有简单直观和功能强大的特点。C++Builder 的“所见即所得”的开发界面形式和面向对象的编程方法为编程人员开发高水平的应用软件提供了极大的方便。增强的数据库访问能力是 C++Builder 的主要特点, 其所具备的专用于多种类型数据库访问的控件, 直接提供了数据查询等规范功能, 这些原本需要编程人员编写大量代码才能实现的功能, 在 C++Builder 中可以少写甚至不写代码实现。此外, C++Builder 为数据库应用专门提供了生成复杂数据报表的功能。

为适应不同层次读者的需求, 本书循序渐进、由浅及深地介绍 C++Builder 的全貌。本书共分 15 章: 1~11 章主要介绍 C++Builder 的基本概念和集成开发环境 IDE 的使用方法; 12~15 章主要介绍 C++Builder 的编程方法和实用技术, 并提供了许多示范 C++Builder 功能的源程序。为方便读者阅读, 源程序中加有详细的文字注释。

在本书中:

第 1 章“基本概念”, 介绍 C++Builder 的特点和基本概念, 以及使用 IDE 开发应用程序的基础知识。

第 2 章“开发入门”, 介绍开发 C++Builder 应用程序的一般过程, 涉及如设置对象属性、在窗体中添加对象和使用数据控件访问数据库等内容。

第 3 章“设计窗体(Form)”, 介绍如何设计、使用窗体, 设置窗体和窗体中的控件, 查看窗体和控件及其属性, 在工程中管理窗体等内容。

第 4 章“菜单设计”, 介绍如何在 C++Builder 中制作与调试菜单, 包括建立、查看菜单, 使用菜单模板和动态增加菜单项等内容。

第 5 章“代码编辑器”, 介绍如何使用 C++Builder 的代码编辑器查看和编写代码。

第 6 章“事件处理程序”, 介绍编写各种不同类型事件处理程序的一般方法与技巧。

第 7 章“面向对象与 VCL 编程”, 介绍 VCL 概念及使用。涉及对象与实例, 继承数据和代码, 对象作用范围, 公有成员与私有成员, 访问对象数据成员, 建立非可视对象等内容。

第 8 章“建立和管理工程”, 介绍在 C++Builder 中管理工程, 工程文件类型, 查看工程文件和使用对象库等内容。

第 9 章“设置工程参数和编译”, 介绍 C++Builder 中的代码优化和编译方法, 涉及设置工程参数, 编译和连接及运行, 管理多个工程和协调小组开发等内容。

第 10 章“制作用户报表”, 介绍制作数据报表的一般方法。本章从制作简单报表开始, 深入介绍带有主从关系的复杂报表和增加页眉、页脚、标题、封面、表格线及预览报表等内容。

第 11 章“构建 Internet 应用程序”, 介绍如何使用 C++Builder 中的 Internet 控件

构建 Internet 应用程序。

第 12 章“文本编辑”，介绍如何在 C++Builder 中建立文本编辑器应用程序，并以实例示范文本编辑器应用程序开发技术。

第 13 章“图形处理”，介绍 C++Builder 的图形功能，包括 C++Builder 图形处理类的使用，编写绘制直线、椭圆、圆弧等图形的程序。作为示范，本章特别介绍了一个图形标绘程序。

第 14 章“Internet 应用”，通过几个不同类型的网络应用程序，介绍 Internet 网络应用程序开发技术，重点是 TCP 和 UDP 控件的应用。

第 15 章“趣味程序精选”，通过几个不同类型的 C++Builder 游戏程序，总结全书内容。为方便查阅，本书在附录中列出 C++Builder 常用函数、控件和窗体参考信息。

本书中，段米毅同志负责编写 1~8 章，丁莉同志负责编写 9~15 章，李罡同志负责全书统稿。

参加编写本书的同志还有：卢山、胡建华、王平、王铁敬、范庆年、官章全、舒志勇以及周庆彬等。

对书中不足之处，恳请读者批评指正。

关于本书所附软盘

本书所附的软盘包括书中所介绍的所有实例的源程序，程序的安装与使用详见软盘中的“Readme.txt”文件。

第1章 基本概念

Builder 是 Borland C++Builder 的简称。Borland 公司(现更名为 imprise 公司)是最著名的程序设计语言工具软件公司之一,其著名产品有: Turbo C++, Turbo Pascal, Delphi 和 C++Builder。

C++Builder 是一个 32 位 C++语言的 Windows 应用程序快速开发环境与工具的总称。C++Builder 提供集成的可视开发环境。集成开发环境包括了代码编辑、可视化用户界面设计、工程管理、编译和调试等多项应用程序开发所需要的功能。可视化指在用户界面开发过程中采用图形化方式,在空窗口框架中摆放窗口组成部件,其结果与应用程序运行时的显示完全一致。空窗口框架称窗体,窗口组成部件称控件。

C++Builder 的特长之一是数据库的连接与访问功能。可连接访问的数据库包括 Oracle、Sybase、Informix、dBASE、Paradox 等等。

C++Builder 的另一优点是提供多种类型的控件库,包括 Windows 系统控件库、与 Delphi 共用的 Borland 控件库、数据库专用控件库等等。

顾名思义, C++Builder 是一个 C++程序的构建器,用于生成 C++源程序。因此 C++Builder 程序的语法必须兼容于 C 和 C++。C++Builder 包含两个编译器,一个 C++编译器和一个对象 Pascal 编译器,并兼容于全部 Borland C++ 5.0x 的源程序代码和 Delphi 2.0 的对象 Pascal 源程序代码。

C++Builder 运行在 Windows 95 或 Windows NT 环境下,用其可开发普通应用程序或高级的客户/服务器应用程序。

1.1 IDE 集成开发环境

IDE 指 C++Builder 集成开发环境,主要包括:

- 屏幕显示的 **Main Menu**(主菜单);
- **Toolbar**(工具栏);
- **Form Designer**(窗体编辑器);
- **Component Palette**(工具箱);
- **Object Inspector**(对象检查器);
- **Code Editor**(源程序代码编辑器);
- **Project Manager**(工程管理器);
- **Debugger**(目标程序调试器)。

C++Builder 的主界面如图 1-1 所示。

IDE 中包括未显示的模块,例如: 32 位 C++和对象 Pascal 编译器, 32 位连接器等等。

由 IDE 模块组成可知,应用程序开发全过程中的每一步骤均可以用 IDE 模块完成,包括:

1. 用窗体编辑器、工具箱和对象检查器完成用户界面设计。
2. 用源程序代码编写器编写源程序。
3. 用工程管理器管理各类应用开发工程中产生的文件和编辑工程。
4. 用编辑器和连接器生成中间目标文件和最终的可执行文件。
5. IDE 总体以及各个显示模块的外形, 其中的内容可由用户根据需要调整设置, 如在工具栏上增添新的选项图标和扩充工具箱中的控件种类等。

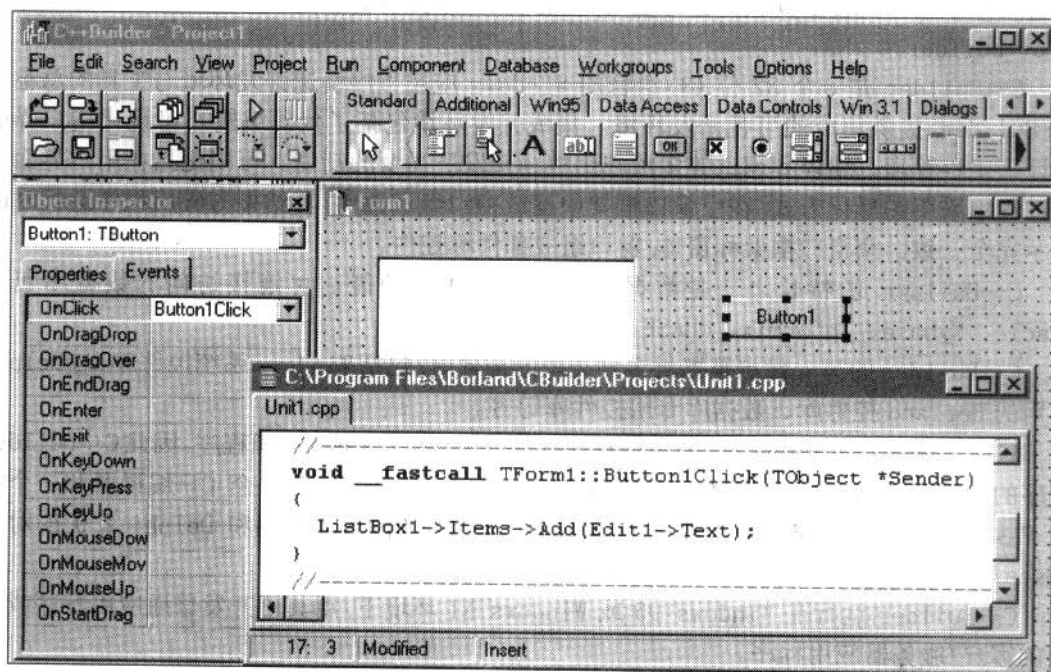


图 1-1 C++ Builder 的主界面

此外, 在主菜单中还可以控制调用数据库开发界面和图像编辑器等等开发工作需要的模块。

因此, 在 IDE 中几乎可以完成全部应用程序的开发工作。简言之, 开发从打开 IDE 开始, 在关闭 IDE 前得到全部开发出的应用程序文件, 包括最终的完成调试的可执行程序文件。

1.1.1 编写源程序代码

使用 C++Builder 集成开发环境, 在可视的窗口设计环境中以图形方式“画”出用户操作窗口界面, 系统自动产生对应的 C++源程序代码, 包括常规的头文件和窗口类文件及窗口过程文件等等。在代码编辑器中, 以系统自动产生的代码为基础, 用 C++语言编写用户处理程序。

事实上 Windows 系统中的 C++语言相当复杂, 直接编写全部 C++程序需要掌握 Windows 程序设计原理。使用 C++Builder 的自动源代码产生功能, 在很大程度上简化了源程序的编写过程, 但是 C++源程序本身并未有任何简化。在 IDE 的代码编辑框可以看到全部源程序代码。

1.1.2 对象

C++Builder 中的对象一般指设计中的窗体和控件。在 Windows 系统中,对象往往指窗口,大部分控件的本质也是窗口。设计窗口包括给窗体选择控件、定义从属关系、显示状态和针对窗口事件编写窗口过程程序。

从程序运行与操作的角度看,对象是输入数据和处理数据的模块。用户面对众多可选择使用的操作对象,可随意地选择某一个完成某种处理。这种操作方式打破了结构化操作必须按步骤分层次使用操作选项的限制,使用户能够比较主动地操作使用应用程序。相对于用户操作形式的变化,程序结构也从面向固定模块关系结构体系过渡到面向对象随机选择模块体系。这种程序设计语言由此得名为“面向对象的程序设计语言”。

面向对象的程序设计需要有相应的环境,包括设计环境和运行环境。Windows 系统的对象指窗口,它支持面向标准的和非标准的窗口的程序的运行。同样,面向对象的语言也要适应 Windows 系统,也就是要把窗口当作对象。

1.1.3 Windows 系统中的窗口

在 Windows 系统中,窗口被当成显示、操作、编辑和程序设计的主体结构。屏幕称为桌面窗口,每一个应用程序一般至少具有一个窗口。运行的应用程序的窗口是桌面窗口上的顶级窗口,不运行,但可直接启动运行的程序以图标形式显示在桌面窗口上或文件夹窗口中。应用窗口通常是可覆盖窗口,多任务系统中同时处于运行状态的多个窗口可以相互重叠显示。其中的一个是键盘输入的对象,称为具有键盘输入焦点的窗口,又称为激活状态窗口,其他窗口都是非激活状态窗口。非激活状态窗口并非不能运行或输入数据,它仍然能够接受和处理系统或其他窗口发来的消息数据。对于运行状态的应用程序,它总是以可覆盖或弹出窗口形式显示在屏幕上。不过,窗口也有可视窗口和不可视窗口之分。将窗口设置为不可视状态,窗口就从屏幕上消失并在任务栏上表现为一个未被按下的按钮,它代表最小化窗口。而任务栏上的按下状态按钮代表可视窗口,即代表屏幕上正在显示的窗口,无论它是全部显示来、部分显示或完全被覆盖在后面。可视窗口又有两种,即最大化窗口和还原状态窗口。

窗口的位置可以一层套一层,最外层是父窗口,其他是子窗口或后代窗口。子窗口只能出现在父窗口的工作区中。同级子窗口可以相互覆盖,也可以拥有下层子窗口。子窗口不能具有某些控件,如菜单条。

窗口的打开也可以是一层套一层的,这与父子窗口的位置关系有所不同。最先打开的是主子窗口,从主子窗口中再打开的窗口是从属窗口。从属窗口大多是临时性的弹出窗口,操作完成后要关闭它才能返回到上层主子窗口中。从属窗口大小不受主子窗口大小限制,从属窗口的显示总处于主子窗口之前。当主子窗口最小化时从属窗口也随之退出屏幕,如果主子窗口被废止,从属窗口也随之被废止。

比如,一个应用程序中的“打开文件”对话框,它是应用程序窗口的从属窗口,而对话框中的一些控件,如下拉列表框、按钮等等又是对话框窗口的子窗口。

应用窗口通常是顶级窗口、覆盖窗口、父窗口和主子窗口,应用窗口的组成“零件”称为窗口控件,包括:菜单、标题条、菜单条、窗口菜单(即窗口顶部标题条左端应用图标被单击后显示的菜单)、标题条右边的最小化按钮、最大化按钮或还原状态按钮、关闭按

钮。其他的控件还有窗口的边框线,窗口中部的用户区,水平滚动条和垂直滚动条,等等。通常应用程序窗口均包括上述部件,而一个用户区中的子窗口不含菜单。

从原理上讲,窗口控件都是应用窗口的特殊子窗口,它们有固定的位置和形态,占居着应用窗口的非用户区(用户区以外的区域)。

窗体是在可视程序开发环境中,编辑中的程序所包含的用户操作界面,它通常是一个标准窗口的框架,或称为空窗口,用户根据设计需要可在其中放置各种控件,如按钮、检查框和组合框等等。

1.1.4 控件

使用 C++Builder 开发应用程序,在建立窗体之后,需要在窗体内设置某些控件,承担针对具体数据的处理、显示和控制等任务。C++Builder 提供一个称为对象检查器的模块,供程序员在设置窗体和控件时观察和设置其参数。

C++Builder 提供的控制多达数十种,按其功能分类主要包括:

文本输入和操作控件,如文本框,运行时可在其中输入和编辑文本。

特种输入控件,如滚动条,运行时操作滚动条可使某个数据改变大小。

按钮和选项控件,包括文字按钮和图形按钮,检查框和单选按钮等。

列表显示和操作控件,如列表框和组合列表框,运行时可在其中列表显示数据,支持用户在列表中中选择数据或输入新数据。

分组控件,如分组框。分组控件可将多个控件组合成为一个控件。

可视反馈控件,如进度条,在运行时显示某项处理的完成情况。

图形显示控件,包括多种显示图像和图形的控件,可用其在窗体中贴图或画线。

Windows 公用对话框控件。C++Builder 支持使用 Windows 公用对话框,包括文件操作、颜色、字体、打印等对话框。

C++Builder 中的控件分为两类,即:“不可视的”和“可视的”。同时,可视的控件又分两类,即:Windows 的和非 Windows 的。从窗口概念出发,各类控件的形式是一致的,在应用开发时使用哪一种并没有明显的区别。但是在运行时,可视控件能够响应用户输入事件并具有固定的显示形态,而不可视控件则不具有这些特点。因此,在应用开发时,因为不存在运行时的用户输入,可视控件和不可视控件都是“不可操作”且具有“显示形态”。

不可视控件,可理解为在运行时不受用户操作直接控制的控件(如计时器或数据库连接器),或不直接显示的控件(如系统公用对话框)。在应用开发时,不可视控件以小图标形式显示在窗体或数据模块中,程序员选择图标后可用对象检查器对其属性进行设置。在运行时,某些不可视控件实际上是具有显示形态的,但是这种显示是间接地产生的。通常,当程序可能需要使用 Windows 系统共用控件时,可在应用开发时设置一个对应的不可视控件;运行时,该控件只在调用函数和方法后启动非响应用户操作而启动的功能。

1.1.5 启动 IDE

IDE 是 C++Builder 的可视开发环境,它由多个模块组成,包括主菜单、窗体、工具箱、代码编辑器、对象检查器以及 C++ Builder 编译器和连接器等。在 IDE 环境中可以完成全部应用程序的开发工作。

启动 IDE 与启动其他 Windows 应用程序一样,有三种方法:

1. 双击 C++Builder 图标。
2. 启动系统资源管理器或文件管理器,在...\Borland\CBuilder\Bin 路径下查找到 BCB.EXE 文件后,双击该文件。
3. 在 Windows“开始”菜单中选择“运行”选项,在“运行”对话框中指定路径到 BCB.EXE,然后单击“确定”按钮运行。

1.1.6 使用窗体建立应用程序操作界面

任何一次启动 IDE 系统都将自动给出一个窗体,该窗体将作为应用程序的主窗口使用,它也就是应用程序的第一个操作界面。可视化的操作界面设计工作在窗体中完成。当启动 C++Builder 建立一个新工程时,窗体就显示在屏幕上,设计者按设计需求在窗体中放置和对齐控件,如按钮或列表框等。如果需要,还可以增设子窗口或从属窗口,调出可用的其他控件,最终完成全部应用程序操作界面的设计工作。

可以在窗体中设置不可视控件,在从数据库获取信息、执行内部计算和管理或间接使用系统公用对话框时需要使用不可视控件。

在 IDE 的可视窗体设计环境中,用户可先为应用程序快速设计用户界面的基本形态,然后逐步按开发进度需要进行修改与扩展。

1.1.7 添加控件

可以把窗体想像成为能够包容其他控件的对象。程序开发需要的大部分控件集中于 IDE 的工具箱中,可以从工具箱中用光标选择控件,然后在窗体中以光标拖动方法设置在预定区域中。设置控件到窗体中后,还可进一步用拖动方法调整它的位置和大小。

C++Builder 工具箱是分页式的,每一页面上集中同类型的多个控件。工具箱及页面都是可滚动的,因此能够对齐的控件种类与数量原则上是无限的。例如,常用的控件像菜单、编辑框或按钮等放在工具箱的标准页中。另一些控件,如计时器、绘画板、媒体播放器和 OLE 容器等放在系统页中。

每一个窗体和控件都具有多种属性,这些属性的具体数值、状态或参量决定了窗体或控件的全部特征,所有的属性数据最终将写入源程序以描述窗口或控件。

1.1.8 增加、修改和删除可用控件

C++Builder 的可用控件保存在可视化控件库(VCL)中,有关 VCL 的使用管理可参阅系统 Help 中的 VCL 参考手册。如果需要修改和删除处理控件库中的控件,或者编写新的控件加入控件库。增加控件方法参见系统 Help 中的“控件编写指南”。

1.1.9 设置控件属性和编写事件程序代码

程序员可以通过对象检查器(Object Inspector)窗口观察和调整指定控件的属性和事件。每当通过单击操作选择窗体区中一个控件时,控件的属性以及可能的事件就全部显示在对象检查器的页面中。当程序员未对属性和事件做任何设置或修改时,页面上显示的内容全部为系统定义的缺省值。

对象检查器由三部分组成。上部的列表框中给出全部可选择的控件,选择某项与在窗体中选择控件效果相同。对象检查器包含一个属性页和一个事件页,分别用于属性与事件的显示和设置。针对某一设置项,如果存在系统给定的可选参量,在用鼠标选择该项后,工程栏上显示下拉列表列出可选参量,用鼠标选择某项即可完成设置。对于某些工程,系统不能确定基内容,只能由程序员输入参量,或者要进一步调用系统对话框选择参量。前者的例子是控件设置高度或宽度,后者的例子对图形控件,要双击控件,使系统打开“Load Picture”对话框输入图形文件。

如果双击对事件页上的某一事件参量,系统自动为选定的控件事件产生事件处理代码的结构,程序员在代码编辑器上看到该结构,并可直接在其中编写代码。

1.1.10 设计菜单条

菜单条通常横向显示出多个菜单名,每一菜单名下对应一个菜单,每一菜单中纵向显示多个选项名。通常在 IDE 中设计菜单使用 **MainMenu**(主菜单)控件。它是工具箱标准控件页的第一项。首先,单击它,再单击窗体的任何位置设置该控件到窗体中。然后双击窗体中的主菜单控件,屏幕显示一个临时窗体,其中存在一个显示第一菜单名的空文本框。输入第一菜单名称后按回车键,此时临时窗体中显示第一菜单名,同时在其下面和右边各显示输入第一选项名和第二菜单名的两个空框,单击其一可输入相应名称。连续地采用选择空文本框并输入名称的步骤,可完成全部菜单名和菜单选项的输入。单击应用程序窗体,可以看到新设计的菜单,该菜单也响应光标的操作,当光标移到菜单名上,相应的菜单就自动打开。

上述的设计只设置了一个可显示的菜单,其中并未包括每一选项名对应的处理程序。要输入程序代码,可选择菜单中的选项,系统自动显示代码编辑框,而且输入文字的位置恰好在输入开始处。

注意,菜单控件是一个整体,以一个控件名和一个控件图标为标志,控件图标并不是实际的设计和对象。

1.2 开发应用程序

在 IDE 中开发应用程序,大致分为设计界面、编写代码、管理工程、编译连接、调试运行等步骤。

1.2.1 设计界面

使用窗体和工具箱上的控件在窗体中设置控件,并在对象检查器中选择修改窗体或控件的属性,完成应用程序操作界面的设计。如果要设计访问数据库的应用程序,须使用数据库专用界面。如果要设计 Internet 网络应用程序,须使用网络控件。

在设计应用程序操作界面的同时,C++Builder 自动产生 C++源程序代码支持这个应用。当程序员选择和修改控件或窗体属性时,相应改变的结果自动以源代码形式显示,这些程序通常只是一个窗口过程的完整外壳,它支持显示控件窗口的显示,但无任何数据处理功能。数据处理功能需要由程序员以 C++语言程序代码形式在给定的窗口过程外壳中编写。由于 C++Builder 的窗体和控件功能较强,某些形式的应用程序甚至可以不编写一程序即

可实现。

1.2.2 编辑源程序代码

编写 C++Builder 应用程序可以使用 IDE 的源程序代码编辑器,也可以用其他的文本编辑器在 C++Builder 系统以外编写源程序文件。

使用 IDE 的代码编辑器,对于窗体或控件代码所做的改动可以立即反映到窗体中。C++Builder 代码编辑器是一个完全的 ASCII 代码编辑器,使用它可以浏览、编写、编辑和删节除源程序的各种 C++语言的可编辑单位,如各种类型的数据、表达式、方法或函数,语句,子程序等等。

1.2.3 管理工程文件

使用 C++Builder 开发应用程序的过程中,系统将产生多个与开发相关的文件。使用 C++Builder 的工程管理器,程序员可以统一管理全部与开发相关的文件。当从 **View** 菜单中选择 **Projects Manager** 选项时,系统在“文件”对话框中列出相关的每个文件,并可方便地在其中选择打开某一个文件。

1.2.4 存储应用程序的对象

C++Builder 使用对象仓库(Object Repository)文件存储应用程序用到的对象,如窗体、**Data Modules**(数据组)、**Wizard** 或 **Experts**(数据导航器)和 **DLLs**(动态连接库)。从对象仓库中可以方便地选用和重复使用保存的对象,这对使用自制的对象特别有利。重复使用对象有利于开发统一界面和功能的系列化应用程序。此外,还可以在异地分组开发大型应用程序时统一提供作为核心标准的应用程序开发工具,供各开发组成员在网络上获取使用。因此,使用 C++Builder 可开发一些不仅限于某一个应用程序使用的窗体或其他对象,并把它们存入对象库,以备他用。比如在一个大环境中,专业开发人员开发一套标准窗体,供其他程序员在自己开发的应用程序中使用。

1.2.5 使用联机帮助

开发应用程序需要足够的开发知识与技术,这对边学边开发的初学者来说相当不容易,即使是熟练的开发人员也难以记忆全部的技术细节,如一个函数名的正确拼写或一个系统属性的特殊设置参量。因此,开发应用程序总是希望开发工具具有内容全面且组织合理的联机 Help 系统。

C++Builder 的联机帮助系统可帮助程序员及时准确地了解与某一问题相关的详细说明信息。C++Builder 的 Help 是了解语言、程序设计、数据库、编译控制和开发环境等信息的有力工具,它方便实用而且内容全面。对于众多的可视控件库、标准 C++函数库、运行时刻库,Win32 和 API 函数及头文件库,C++Builder 都有设有专门的 Help 详细地说明其内容和使用方法。同样,对于 C++Builder 的辅加功能,如 PVCS 版本识别控件等,也都有详细的 Help 说明。

使用联机 Help,通常采用下列方式之一:

1. 在主菜单中选择 **Help** 菜单的 **Contents** 选项。

2. 按 F1 键。
3. 在对话框中单击 Help 按钮。

在 C++Builder 中允许采用其他更方便的方式, 直接显示需要了解的 Help 信息段。

在代码编辑器中, 使用拖动方法选择一段文字, 比如一个函数名, 然后按 F1 键, C++Builder 立即显示该函数的 Help 说明。如果没有与选择文字完全一致的信息, 则显示最相似的信息说明。**Help** 菜单的 **Contents** 选项打开“Help”窗口, 从“Book Shelf”架处显示信息。“Book Shelf”架的内容是各种 Help 的目录, 选择某个目录可以进一步打开选定的 Help 内容。

在窗体中或在工具箱上选择控件, 然后按 F1 键, 显示该控件的说明信息。这种方式称显示上下文敏感信息。同理, 在对象检查器中选择属性项或事件项, 然后按 F1 键, 可显示该项目的上下文敏感信息。如果系统处理中出现错误, 选择错误信息后按 F1 键, 可得到关于错误的详细说明。

C++Builder 支持用户重组 Help 内容的功能, 该功能通过一个称为 **OpenHelp** 的辅助工具实现, 该工具的文件名为 OpenHelp.exe。

1.2.6 编译和运行应用程序

所有 C++Builder 工程的最终目标都是要得到相应的 .EXE 可执行文件或 .DLL 动态连接库文件, 即把开发得到的源程序经编译器编译和连接器连接, 最后产生 .EXE 或 .DLL 文件。

不论在编译、连接还是运行阶段, 程序员均可观察并调试自己的应用程序。

程序员可通过“工程参数”对话框或选项行来按需要设置选项。C++Builder 提供了完备的编译工具, 包括编译器、连接器、**MAKE**(工程构成器)和其他一些实用工具。

1.2.7 调试应用程序

IDE 提供集成调试器, 它帮助程序员查找并修正应用程序中的错误、控制程序执行、监视可变参数值和数据结构项内容以及在调试中修改数据值。

为准备调试, 应启动带有调试信息的编译连接工程文件, 然后在调试器控制下运行程序并进行调试。用户能利用调试器的特性去检查程序的当前状态, 可专门调试一个特定的程序段, 或直接运行, 也可以控制在一个时间点或一个程序断点处停止执行程序以检查运行状态。如观察变量值、堆栈、函数和程序的输出, 最终检查出源程序代码中的错误, 或未达到设计要求的错误因素。

1.3 编写数据库应用程序

C++Builder 的强大功能之一就是支持建立高级数据库应用程序。它包括一组嵌入工具, 允许用户连接 Oracle、Sybase、Informix、dBASE、Paradox 或别的数据库服务器, 实现应用程序之间的透明化的数据共享。Borland 的 **BDE**(数据库引擎)支持从桌面系统到客户/服务器方式的各档次应用程序的开发。一些工具, 如 **Database Explorer**(数据库浏览器), 可以简化数据库应用程序编写工作。数据库浏览器是一个层次型的浏览器, 可以审查并修改服务器特性的数据库结构对象, 如表、字段、存储过程定义、触发器、参考和索引描述。

在数据库的永久性连接下, 数据库浏览器能够完成如下功能:

1. 建立和维护数据库别名。
2. 观察数据库的 **Schema data** (结构数据), 如表、存储过程和触发器等。
3. 观察表对象, 如字段和索引。
4. 建立、观察和修改表中的数据。
5. 键入 SQL 结构化查询语言选项, 直接查询任何数据库。
6. 建立和维护数据词典, 保存属性集。

第 2 章 开发入门

实践是学习编程的最好方法。

实践前需要设计程序的基本流程。

比如,要编写一个查阅数据库字段的 C++Builder 应用程序,其大致的流程为:首先为数据库表建立访问关系,然后处理数据,最后产生一个文件保存结果。

2.1 开发新的应用程序

在开始开发一个新应用程序之前,应先建立一个文件夹,保存应用程序的源文件。这种方法使程序员不容易将该应用程序的源文件与其他类型文件混淆。一个唯一的文件夹也便于查找和维护应用程序相关的各种文件。

比如:

1. 建立名为 **MySource** 的文件夹,位于当前驱动器的根目录下,作为开发的工作目录保存即将建立的应用程序工程文件。

2. 启动 IDE,用 **New Project** 选项打开一个新工程。C++Builder 把新应用程序视为一个工程,当启动 C++Builder 时,它用缺省工程文件作为空工程。如果 IDE 已经打开了一个工程,可选择 **File** 菜单的 **New Application** 选项建立一个全新的工程。

无论何时打开一个新工程,C++Builder 都自动建立如下两个文件:

- (1) **Unit1.CPP**: 与工程主窗体相关的源文件。

- (2) **Project1.CPP**: 在工程中保持与各窗体关联的源文件。

3. 选择 **File** 菜单的 **Save All** 选项保存工程文件到硬盘。在“Save”对话框中选择 **MySource** 文件夹,以缺省文件名保存上述文件。

此外 C++Builder 还要建立与工程有关的其他文件,它们可在 **MySource** 文件夹中看到。

当打开新工程时,C++Builder 显示一个缺省名为“Form1”的工程主窗体。

在 C++Builder 中,总是使用窗体为应用程序设计用户界面,窗体可以包含菜单和上下文菜单,它们放在一起又可以产生应用程序的对话框,而对话框又可以进一步用作父窗口或子窗口使用。

根本上说,窗体如同一块可以在上面建立应用系统的画布。

采用设置对象到窗体的方法建立用户界面是最常用的工作方式。对象可以是标准接口控件,如 **checkbox boxes**(检查框)或 **drop-down lists**(下拉列表),也许需要将其完全包容在另一个控件中,如 **data grids**(数据线段)、**bar charts**(条状控件)和 **editors**(编辑类控件)。

在新工程打开后,系统除显示新窗体之外,还显示 **Object Inspector**(对象检查器),它显示所有选用对象的各种属性和事件设置。当工程窗体被选中时,它显示窗体的属性和事件设置。此时,对象检查器上方的下拉列表中显示当前对象名,对于工程主窗体,一般情况下该对象是 **Form1:TForm1**。

2.2 设置对象的属性值

为设计用户应用程序界面的式样和细节, 需要使用对象检查器设置对象的相关属性。比如要为 Form1 设置 **Color**(颜色)属性“clAqua”, 即海蓝色, 则在对象检查器中找到窗体的 **Color** 属性, 然后单击右端, 打开出现的下拉列表框, 选择下拉列表框中预定义颜色名中的“clAqua”。此时窗体 Form1 的工作区立即变为海蓝色。这种设置窗体或控件的方法相当于初始化对象。

采用上述的初始化设置方法与 C++编程的初始化方法有明显不同。在 C++中初始化窗口用户区颜色, 要将“clAqua”写到源程序中。

2.3 在窗体中增加对象

系统的工具箱上列出了所有可使用的控件, 这些控件在不同页面上分组对齐, 便于取用。使用工具箱上的控件在窗体上建立应用程序接口, 简言之就是把对象加到窗体上。其操作如下例所示:

1. 增加一个 **Panel** (面板)控件窗体。在工具箱的 **Standard** (标准)控件页面上, 双击 **Panel** 控件图标, 则该控件自动加到窗体中央。另一种方法是先单击控件图标, 然后单击窗体任意位置, 于是该对象就加到了单击位置上。为方便识别控件图标, 可将光标置于控件图标之上, 片刻后将显示相应控件名提示。一旦将一个控件加到窗体上, 系统将自动产生相应的应用程序代码, 代码内容与格式都不必由程序员处理, 完全由 C++Builder 自动生成。

2. 将 **Panel** 控件的 **Align** 属性设为“alBottom”。在窗体上选择 **Panel** 对象, 然后转到对象检查器的属性页, 选择 **Align** 属性并修改设置为“alBottom”。参数“alBottom”的作用是使面板平铺显示在窗体的下部。

3. 改变窗体大小。可直接拖动窗体边框实现。

4. 改变 **Panel** 控件大小。先选择控件使控件外围框线出现, 然后直接拖动框线改变控件大小。

5. 增加 **Table**(表)控件到窗体。在工具箱上找到 **data Access** (数据访问)控件页面, 再从中找到 **Table** 控件。单击该图标, 然后单击窗体区任意位置。此时, 表控件的名子是系统指定的缺省名“Table1”。

6. 将 **Table** 控件的 **Data base Name** (数据库名)属性设置为“BCDEMOS”, 该设置使控件成为一个访问数据库表 BCDEMOS 的数据集。

2.4 使用数据控件访问数据库

设置 **Table** 控件及 **Data base Name** (数据库名)属性以后, 可以进一步为数据库访问建立一个数据源。

从工具箱的数据访问页面上选择 **Data source** (数据源)控件到窗体区。将 **Data set** (数据集)属性设为“Table1”。

在工具箱的 **Data Controls** (数据控制)页面上有多种控件,用于以不同方式观察数据库的数据。为了显示数据库表中所有字段,此处使用 **DBGrid** (数据栅格)控件。

2. 从工具箱上选择 **DBGrid** (数据栅格)控件,将其设置到面板控件“Panel1”上。具体方法是:首先单击数据栅格图标,然后单击窗体中的面板控件“Panel1”的区域,使 **DBGrid** 控件加到面板控件。之后,设置有关控件的属性:

(1) 设置 **DBGrid** 控件的 **align** (对齐)属性为“alLeft”,使栅格区域位于面板的左部,然后拖动栅格区右边线调整该区占操纵板区域约四分之三大小。

(2) 设置 **DBGrid** 控件的 **DataSource** (数据源)属性为“DataSource1”。

(3) 设置表控件“Table1”的相关属性:

在窗体中选择“Table1”控件,在对象检查器中设置属性如下:

- 设置 **TableName** (表名)属性为“BIOLIFE.DB”。
- 设置 **ReadOnly** (只读)属性为“true”。
- 设置 **Active** (激活状态)属性为“true”。

此时,在窗体数据栅格区中显示数据,该数据来源于数据库“BCDEMOS”的“BIOLIFE.DB”表。在 C++Builder 开发环境中,类似于 **DBGrid** 控件的对象具有“数据感知”功能。“数据感知”使程序员能够检查开发中的应用程序是否与数据库连接。但这时并不意味着可以滚动画面观察全部数据,或者立即着手编辑当前数据,执行这些操作必须要在应用程序运行之后。

(4) 按 F9 键,编译并运行工程。

当运行应用程序时,系统显示应用程序窗口,它与设计的窗体样式相同。应用程序窗口可用最小化和最大化按钮改变大小,窗口左上角也有控制菜单按钮。在应用程序窗口中,用户可以在数据栅格区域看到数据库表中的数据,用户此时可以做滚动操作以得到表中其他数据的显示。因为开发时对该栅格控件设置了只读属性,故此用户不能改变数据库表中的字段内容。

3. 使用数据库导航器和数据库图像控件。首先,从工具箱的数据控制页面上选择 **DBNavigator** (数据库导航器)控件到窗体,设置其 **Data Source** 属性为“DataSource1”。数据库导航器由多个按钮对齐组成。

一般,缺省属性参数的数据库导航器具有 10 个按钮,其中的一部分此时并不需要。在此只用到 4 个,即:

- First(第一个);
- Prior(前一个);
- Next(下一个);
- Last(最后一个)。

选择按钮须设置 **Visible Buttons** (可视按钮)属性。

注意,选择导航器按钮的设置方式与前述不同。首先,选择 **DBNavigator** 控件,然后在对象检查器属性页面上双击 **VisibleButtons** (可视按钮)属性栏左部的属性名,系统展开该属性为 10 项,各对应不同的按钮。针对每一项,在栏右部选择其逻辑值为“True”或“False”,以确定对应按钮是显示还是不显示。

设计时需将如下 6 项属性设置为“False”: