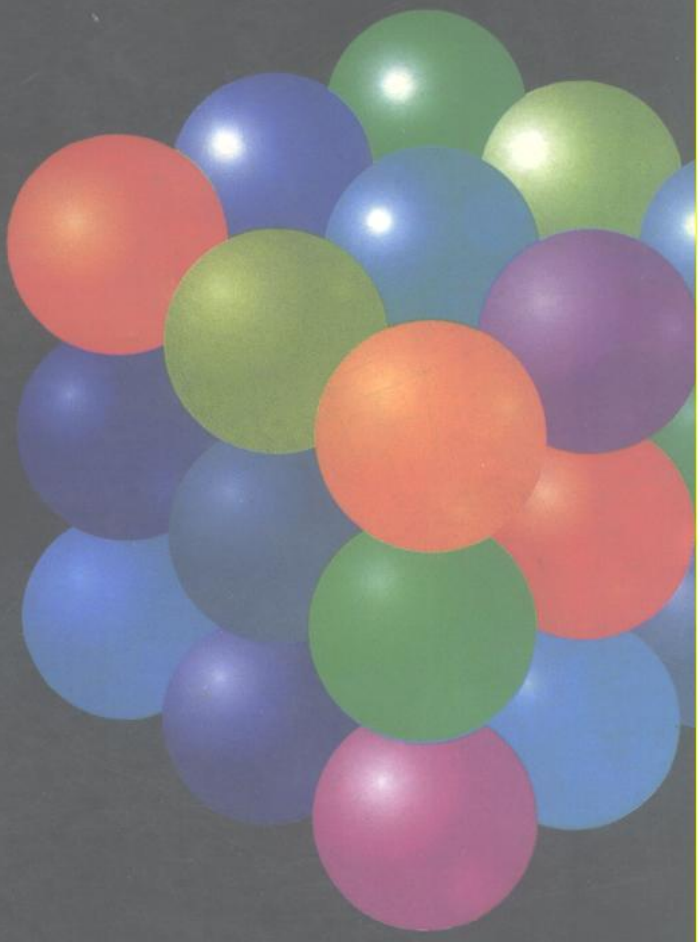
Brady
北京科海培训中心

Borland C++ for Windows 程序设计

[美] Steven Holzner 著
董春雷 赵 军 译
成 昊 审校



清华大学出版社

Brady　　北京科海培训中心

Borland C++ for Windows 程序设计

[美] Steven Holzner　著
董春雷　赵军　译
成昊　审校

清华大学出版社

Borland C++ for Windows 程序设计

Borland C++ for Windows Programming

(京)新登字 158 号

Borland C++ for Windows 程序设计

[美] Steven Holzner 著
董雷雷 赵军 译
放 昊 审校

清华大学出版社出版
北京 清华园
门头沟胶印厂印刷
新华书店总店科技发行所发行
☆
开本:787×1092 1/16 印张:22.75 字数:553 千字
1994 年 10 月第 1 版 1994 年 10 月第 1 次印刷
印数:0001-8000 册
ISBN 7-302-01679-8/TP.725
定价:50.00 元

导 言

编写小规模的程序是一种令人愉快的实践，小规模的程序通常编译和运行都很快——很少有程序员没有从小规模编程中得到愉悦和满足的。事实上，计算机特别是微型计算机的本来目的就是为小规模程序设计的。很少有几个计算机爱好者的机器中有超过几千 K 字节的内存。

但是时代在发展，尤其是计算机领域的发展更为迅速。随着技术进步，设备的费用开始降低，磁盘和内存的容量越来越大，CPU 的速度也越来越快。微型计算机从一种爱好变成了一种工业——一种用户对硬件、软件功能要求越来越高的工业。另外，在过去的 10 年中程序的规模也有了惊人的增加，而且编写规模大 10 倍的程序花费的努力绝不止 10 倍。

对于标准编程语言来说似乎存在一种反常的现象：随着程序规模的增加，越来越难于控制、难于管理，错误会扩散和影响到程序的所有部分，一部分代码会和另一部分冲突。调试大程序不再是一种练习，而是一种职业。

这是由于标准编程语言是为小规模项目（以今天的标准衡量）设计的。程序的某些部分通常与程序的其他部分相关联，特别是当有许多全局变量的时候。编写任何一部分代码时，对整个程序中的存储和使用数据的方式都必须很清楚。在开始调试时，会发现要调试 10000 行相关的代码和数据——整个程序——而不是一个易于管理的程序片段。

事实上，随着程序变大，主要的问题是如何将它分成易于管理的几部分。例如，在汇编语言中，文件中的一切都是全局的，也就是可以由程序的其他部分访问（当然也不会经常用汇编语言编写大程序）。在早期的 BASIC 中，可以用 GOSUB 子程序把代码分开——但是所有的数据在子程序和程序主体之间都是全局的，类似 FORTRAN 这样的语言开始支持纯局部数据的函数和子程序，并且允许显式传递或用 COMMON 语句传递数据。C 语言以其强数据类型而著称（并因之称为数据结构），可以确保发送到或从程序其他部分接收的就是你所需要的。随着程序变大，程序员的策略是分而治之，现在称之为“封装”。

为什么需要 C++？

数据局限于某一特定函数仍是不够的，随着程序复杂性的增加，程序员会发现其他函数也需要访问这些数据。需要把程序的整个部分（代码和数据）放在一起——不是一个简单的函数，现在称之为“对象”（object）。

如果把一个大程序看作一个工作间，则程序员希望它具有“自治”工具，即他们不需要考虑一台电视机内部的数据和功能，而只需要有一个电视机可供使用。在一个实际电视机中，内部数据和函数是完全私有的——也就是我们不知道内部是如何实现的，我们所看到的是有一台电视机，可以看到优美的图象。

编程的这一阶段已经可以实现，能够将数据和函数包装在一起，这样将程序内部的复杂细节隐藏起来（其思想是，眼不见心不烦）。要实现这种思想只有将数据和维护、操作数据的函数包装在一起，这样，就可以将大得难于控制的程序分成类似于工具和资源的部分（这也

是看问题的一般方法)。基于这种思想,随着程序增大,我们选择了C++语言。

对今天的PC程序员来说,用C++在Windows下的编程更是如此。如果有Windows编程的经验,就会知道是多么的困难——在屏幕上显示一个窗口可能需要几十行复杂的代码和5个或更多文件。

现在一切都不同了。C++不仅允许将程序的各部分分装成简单的对象,而且有许多类库,或能够产生许多不同类型对象的完整库。Borland为Windows编程创建了ObjectWindows Library(OWL),这个库也是本书的基础。如果要建立一个窗口,要从Borland类库中派生一个封装在一起的对象。要建立一个对话框也是一样。在许多已创建对象的基础上,用C++编写Windows应用程序要比用C快得多。并且由于提供了软件设计工具,Borland C++更为方便。现在,编写Windows程序甚至比C程序员过渡到C++更为简单。

我们的目标

这本书是为具有C或C++编程经验,并想转到Borland C++ for Windows的程序员编写的。也就是说,假设你已有C编程经验(大部分程序员都是从丰富的C基础上转到C++),请阅读第1章C++概述,并根据需要进一步了解。如果你已经熟悉C++,可以跳过这一基础章节,直接进入Windows编程。

由于Borland C++是为程序员提供的一种语言,而且这本书以"可能是"程序员为对象,所以我们打算以程序的效果为基础。换句话说,我们想知道这种语言能为我们做什么,而不是其他什么。这里我们并不打算讨论冗长的、学术的抽象概念——这些讨论不在本书范围之内。这本书是为程序员写的,并希望展现出Borland C++在Windows环境下的全部功能。

但是应该清楚,在使用Borland C++,甚至C++语言之前,有许多东西要学习。这里并不讨论类与对象、重载函数与覆盖函数、使用Windows与编写Windows程序之间的区别,而是主要学习:虚函数、内部I/O类、构造函数、析构函数、继承性等等。也就是说,将重点学习Borland C++中许多要学习的东西。

由于这个原因,本书中有大量的例子,都可以运行。没有比用例子来学习更好的方法了,而且这些例子都很不错。另外,我们在一行一行地开发这些较长的例子中常会有插图、注释和提示,特别是有些提示用于提供一些附加信息。提示中有些能使程序运行快两倍,也有些是关于Borland C++的其他部分特殊处理的方法,甚至可能是Borland C++内部的具体实现。无论是什么,提示都是从专业程序员的观点来看待Borland C++,因此可以给我们额外的功能和控制。

我们的目标是:通过实践学习Borland C++,从零开始,一步步向前发展。从最基本的Windows面向对象编程技术开始到编写功能强大的Windows程序,从Borland C++中最简单技术的到掌握最复杂的技术。

本书内容

C++语言本身具有很高的速度和精确性——我们将使之付诸实践。开始花一些时间学习C++本身,然后再根据需要进一步学习C++。下面是一些我们将要掌握的C++概念:

- 类
- 对象
- 函数重载
- 在C++中使用内存
- 继承性
- 缺省参数
- I/O 类库
- 虚函数

有了C++的基本概念之后，就会明白C++和 Windows 的结合是多么的自然。要充分利用 Borland C++，必须掌握下列 Windows 主题：

- 创建窗口
- 菜单
- Windows 中的文件处理
- 对话框
- 滚动条
- 列表框
- 鼠标
- Windows 图形
- 在 Windows 中读入击键
- 多窗口程序
- Windows 消息
- Borland C++中的调试

在明白 Windows 编程是多么简单的同时，要掌握这些及其他主题。我们将创建大量有用的例子，包括一个弹出式记事本、绘图程序、数据库程序、完整的文件编辑器等等。

预备知识

正如前面提到过的，使用本书需要一些 C 方面的知识，Borland C++建立在 Borland C 之上，在进入 C++之前，应该具有良好的 C 基础。随着软件的更新，我们使用 Borland C++ V4.0 for Windows 和 Windows 3.1 或更高版本。

从现在开始，充分展现 Borland C++的全部功能。之所以立即付之于实现是因为对编程而言没有比看到它实际效果更好的方法了。在第 1 章，将快速浏览 C++，然后开始我们的 Windows 编程指南。

目 录

第1章 C++概述

Borland C++是目前 Windows 下最为激动人心的编程工具，它使 Windows 编程发生了一次革命。这个功能强大的软件包使 Windows 编程从来没有过如此简易。要编写有价值的 Windows 应用程序，不再需要大量的耐心、经验和昂贵的软件。用 Borland C++（以及类似的程序）开发 Windows 应用程序要比过去容易得多。在这一章将建立第一个 C++程序——尽管它不是一个真正的 Windows 程序，但是 Borland C++可以在 Windows 3+下运行它。我们将会看到编程过程比预想的要容易，因为 Borland C++为我们做了大量的具体工作。

如果你曾用 C 为 Windows 编程，那么会发现 C++的真正优点。Windows 的编程接口丰富而且复杂，大多数还有多项选择项（对程序员而言），比如窗口式样、运行程序的方式、窗口尺寸、内存分配等等。

但是，Windows 程序的主要部分并不真正需要这么多选择——这也是为什么 C++能够完善工作的原因。所有这些选择项可以归为标准的“类”，我们可以从中派生自己的对象。用这种方式，许多细节被隐藏了起来，并且 Windows 编程接口的“表面部分”迅速减少到可管理的程度。这就是 C++所做的。将大的程序分成可管理的、自我包容的对象。这些对象能够完成它自己需要的所有操作——比如将它们自己写到磁盘文件中。我们会看到这种模块化方法在全书中是非常有用的。

要完成这些，必须熟悉 C++和一些 C++术语：类、对象和函数重载。本章，我们将熟悉这些概念，在下一章中，将学习把这些 Windows 的概念用于实践。但是，并不意味着必须等到程序在 Borland C++下运行。然而在开始做任何事情之前，首先应熟悉主机环境——Windows 本身。

1.1 关于 Windows

许多人认为图形用户接口（GUIs）是微机的未来潮流，这很可能是对的。很明显 Windows 3.0 曾是历史上卖得最快的软件包（在前 6 周 500000 份，前 9 个月 3000000 份），就许多出色的方面来说，Windows 自己就是一个完整的操作系统。

Windows 与 DOS 有很大的不同，主要差别在于 Windows 是图形用户接口（GUI），它引进了许多新的概念。总的思想是大部分可用选项都以屏幕上的对象形式呈现给用户，就象随时可用的工具。这种方式的用途是惊人的——用户不用记住复杂的技术和关键字，只需选择当前任务的合适工具即可。这样，图形接口使计算机可提供无限可用的工具。让我们看一看该操作系统的背景。

1.1.1 Windows 的简单历史

Microsoft 在 1983 年就开始 Windows 方面的工作，这仅在 PC 机出现两年之后。但是，

最初的版本 Windows 1.01 到 1985 年才真正推出，该版本运行在当时的标准机器上：一台双 360K 软驱、256K RAM 和 8088 处理器的 IBM PC。显示画面自动平铺，也就是窗口是占满屏幕的，由于其平面特点(二维)，没有给人留下什么特别的印象。

下一个主要版本是 Windows 2 在两年后推出，窗口首次可以在屏幕上重叠，但是，Windows2 只能运行在 80x86 的实模式下，也就是说只能使用 1M 内存。有一段时间，Windows 曾经分成 Windows 286 和 Windows 386，以便充分利用新推出的 80386 芯片的功能。这时有了很大的进步，但很明显，仍有许多工作要做。

1990 年的 5 月，Microsoft 推出了 Windows 3.0。Windows 3.0 在功能和外观上比以前的版本有了很大的改善，它以比例字体而著称，比例字体使显示更加精细，版本 3.0 对 DOS 程序也有更好的支持。1992 年 4 月推出版本 3.1，它是对版本 3.0 的改进，特别是在文件管理方面。现在许多用户把 Windows 作为 PC 上的主要操作系统。

早期版本中的 MS-DOS Executive 被三个管理 Windows 的窗口代替：Program Manager，Task List 和 File Manager。从编程的观点来看，Windows 3+的主要特点在于它支持扩展内存(多达 16M RAM)，并且在 386 增强模式下，Windows 利用 80/3/4/586 的内部虚拟内存能力使程序员可以访问到实际安装内存的 4 倍。这样在一台 16M 内存的机器上，Windows 可以提供 64M 内存能力。取消内存限制曾是 OS/2 的主要优点之一，但是现在越来越多的程序员转向 Windows。Windows 3+才是真正的 Windows(窗口系统)！

1.1.2 窗口的组成

一个典型的 Windows 3+的窗口显示在图 1.1 中，在继续阅读后续章节之前，应该熟悉它的组成和各部分的名称。正如你知道的，在编程之前了解用户希望 Windows 应用程序干什么是非常重要的。让我们花点时间来重温一下 Windows 技术——这有助于本书后文的阅读。在图 1.1 的窗口中，左上角是一系统菜单框，选择它时显示一菜单让用户移动窗口、关闭窗口或最小化窗口。上边的中间部分是标题栏，它为用户标识应用程序提供了一种方法。

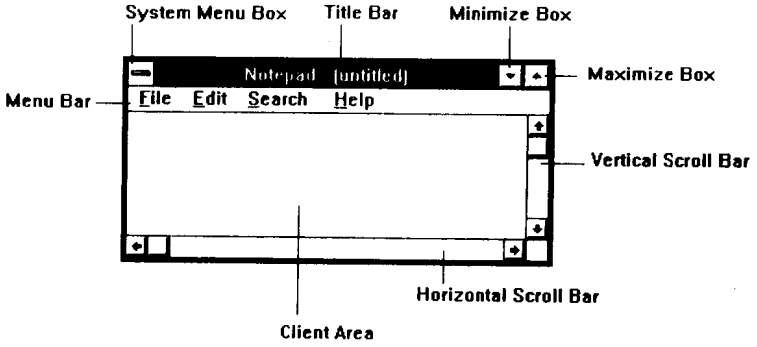


图 1.1 一个 Windows 3+窗口

标题栏的右边是 Minimize 和 Maximize 框，让用户把窗口缩成一个图标(称为应用程序的图标状态)，或者把窗口展开，通常是占满屏幕。标题栏下面通常是一个菜单栏，提供应用程序当前可用的菜单选项。绝大多数独立的应用程序都至少包含一个菜单：File 的菜单栏，该菜单中最下面的一个菜单项通常是 Exit 项，如图 1.2 所示：

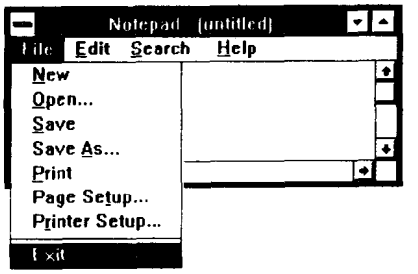


图 1.2　Exit 菜单项

总应有 Exit 项!

Exit 项是用户结束应用程序的一般方法，所以如果应用程序支持文件处理，则应在 File 菜单的底部包含 Exit 项(事实上，即使不使用文件，Windows 应用程序通常也有一个 File 菜单提供 Exit 项)。

菜单栏下面是客户区，如图 1.1 所示。实际上，窗口中菜单栏下面除了边界和滚动条之外都是客户区(该区是窗口用来显示的)，这是我们的绘图区，Borland C++将直接操作该区域，在这里可以放置按钮、列表框、文字框和程序的其他部分。在 Windows 下，这些可视对象称为控件(control)。

靠在客户区的右边是垂直滚动条，在显示文字的窗口中，这是常有的。如果文字在一个窗口之中放不下，滚动条移动文档以看到全文(也许你已知道，滚动条中可上下移动以确定当前位置的小方块称为姆指块(thumb)。

在窗口的底部是另一个滚动条，它水平移动客户区中的文字。窗口中除了客户区以外的一切称之为非客户区——也包括边界在内。Windows 负责管理窗口的非客户区，我们自己负责客户区。

1.1.3　保留 Windows 风格

正如前文所述，用 Borland C++编程时，应该熟悉用户希望的 Windows 程序的工作方式和风格。特别是，要非常熟悉鼠标的单击、双击语言，以及用户希望应用程序完成的操作。

例如，File 菜单中通常有一个 Exit 项，该项(如果有的话)总是最后一项，它是你应该编程序的 Windows 接口的一部分。还有许多其他方面，用户希望 Windows 应用程序按某种方式工作，这在你开始编程之前就应该熟悉，换句话说，即应掌握大量的 Windows 约定。尽管在涉及到这些主题时要讨论这些约定，但是仍应该熟悉用已有的 Windows 应用程序来掌握 Windows 接口的风格。

有些约定是自动实现的，例如在文件列表框中(程序在其中列出供打开的文件)，单击鼠标表示加亮显示文件名(称为选择)，双击表示打开文件(称为选中)。另一方面，也可以不用鼠标，只用键盘操作 Windows——则应同时提供键盘支持(这时，用户使用 Tab 键移到正确的框，用箭头键加亮显示文件名，用 Enter 键选中它)。

在本书的程序设计中，我们假设你有一个鼠标。尽管也可以不用鼠标操作 Windows 应用程序，但是会极大地降低 Windows 程序员(甚至有经验的 Windows 用户)的工

作效率。

还有一些其他 Windows 用户希望的约定，如果有一个对象可以在屏幕上移动，用户希望能够用鼠标拖动它。还希望在菜单、系统菜单中有加速键来关闭窗口、移动窗口、重置窗口大小或最小化窗口。掌握这些约定的最好方法是使用已有的 Windows 应用程序(参见附录 A——Windows 程序设计)。

1.2 第一个程序

现在启动 Borland C++，如图 1.3 所示。你看到的是 Borland C++的集成开发环境(IDE)及其组成部分，本书中我们应非常熟悉它。

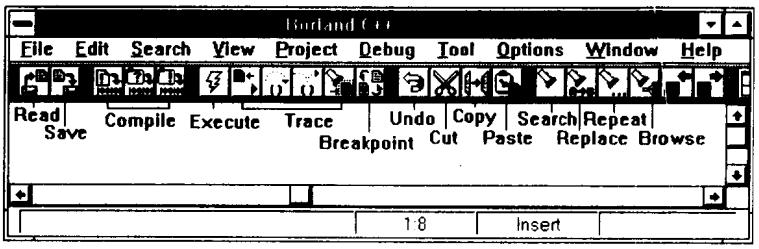


图 1.3 Borland C++的集成开发环境

我们从运行一个 Borland C++的 DOS 风格的 C 程序入手，比如，使用 C 中传统的第一个程序：

```
#include <stdio.h>
void main()
{
    printf("Hello,world.");
}
```

尽管在下一章中会发现，Windows 编程和 DOS 下的编程有极大的不同，但是 Borland C++ for Windows 仍能处理该程序。要建立程序 hello.c，从 Borland C++的 File 菜单中选择 New，出现一个新的窗口，如图 1.4 所示。

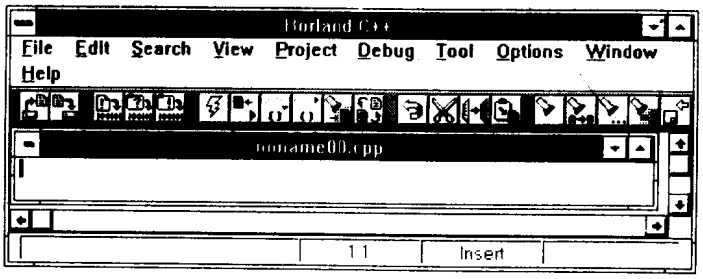


图 1.4 一个新窗口

现在键入上面的程序，如图 1.5 所示。

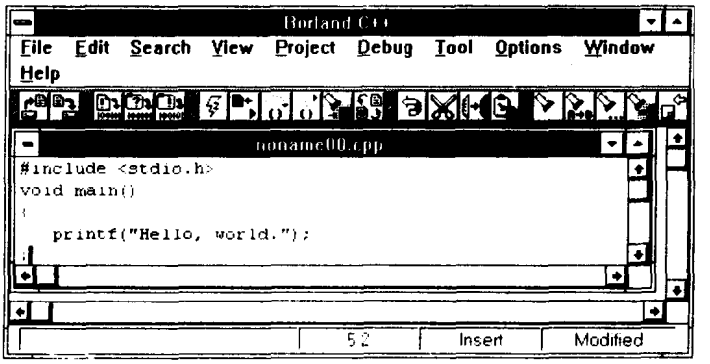


图 1.5　第一个 C 程序

使用 Borland C++中 File 菜单的 Save As…项，将该文件保存为 hello.c(也就是单击 File 菜单中的 Save As…，键，hello.c 为文件名并单击 OK)，这样就在磁盘上建立了文件 hello.c。

这时离运行该程序就只差一步之遥了。Borland C++把你开发的程序看成一个项目(project)，一个项目涉及到许多个文件，但是开发该程序时只有两个文件：hello.c 和项目文件本身 hello.ide。Borland C++为每个项目保留一个项目文件(.ide 文件——代表集成开发环境)。项目文件中保存与该项目有关的文件清单，所以在创建并运行 hello.exe 之前，要建立 hello.ide。

要建立它，从 Borland C++的 Project 菜单中选择 New Project…，则打开 New Project 窗口，如图 1.6 所示，分别在 Project Path and Name 框和 Target Name 框中键入 hello.ide 和 hello。

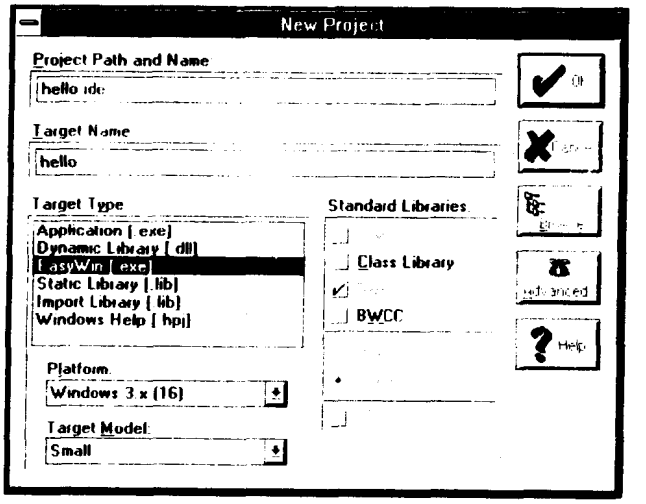


图 1.6　建立一个项目

最后，选择选项 EasyWin，如图 1.6 所示(下一步就能看到这是什么意思)，单击 OK 按钮建立 hello.ide，这样就建立了一个名为 hello，Borland C++在项目窗口中跟踪的项目(参见图 1.7)。

一些与 Borland C++ .exe 文件相关的普通文件将显示于项目窗口：一个.CPP 文件(C++)，一个.rc 文件(源程序)和一个 def 文件(每一个 Windows 应用程序均需要一个.def 文件)。然而，我们只有一个 hello_c 文件。如图 1.7 所示。

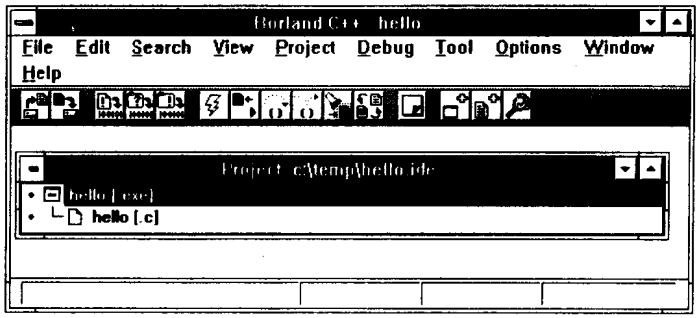


图 1.7 Project 窗口

每一个 Windows 程序都需要一个.def 文件，因为它可以定义一些基本的特性，比如.exe 目标类型(这里是一个 Windows.exe 文件)，程序堆栈的大小，程序是否可以交换出内存等。这是我们的项目需要的另一个唯一文件，可以使用 Borland C++提供的一个通用.def 文件，它在 Borland C++的 examples\owl 目录下。其内容如下：

```
EXETYPE WINDOWS
CODE PRELOAD MOVEABLE DISCARDABLE
DATA PRELOAD MOVEABLE MULTIPLE
HEAPSIZE 4096
STACKSIZE 8192
```

要加入该文件到我们的项目，加亮显示项目窗口的第一行——hello[.exe]并按 Ins 键，这样引出 Add to Project List 框，键入路径和名称：OWL.DEF，则这一行就加到了项目之中，如图 1.8 所示。

我们并不是建立一个真正的 Windows 程序，只是学习 C++，hello.c 是一个 DOS 风格的程序。Borland C++从程序中的 main()过程可以检测出来(Windows 程序则使用 WinMain())，Borland C++称之为 EasyWin 程序，而不是 Windows 程序，在建立它时已经说明了这一点。

要建立并运行 hello.exe，从 Project 菜单选择 Build all，Borland C++则建立 hello.exe，完成后，它在屏幕上显示一个 Messages 的窗口，并显示编译 hello.c 和建立 hello.exe 的结果。如果没有错误，在 Messages 窗口中选择 OK。

要运行 hello.exe，从 Debug 菜单中选择 Run，选择后，程序即投入运行，如图 1.9 所示，从中可看到 Hello，world。就是这样，我们的第一个程序成功了。要结束，则选择 Hello 的系统菜单中的 Close 菜单项(左上角的框)。

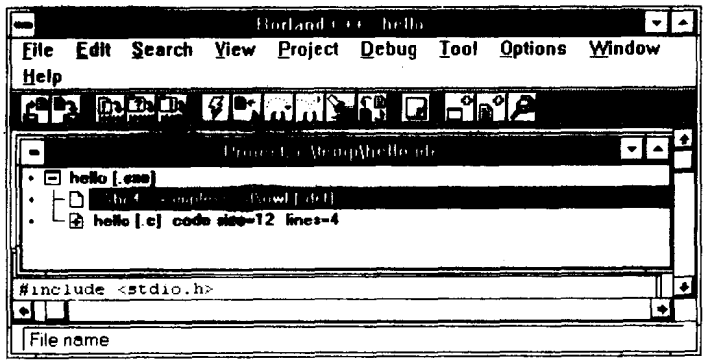


图 1.8　完成了的 Project 窗口

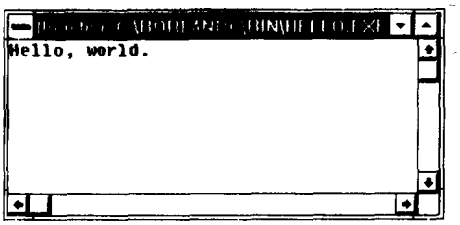


图 1.9　运行 hello.exe

1.3　欢迎进入 C++世界

现在我们来看一看到底是什么使C++优于C。首先看一些C++的独特的但很普遍的特性，然后再讨论真正使C++出名的原因——具有操作对象的能力。

对象背后的推动力是模块化(Modularity)，在本章的前面已简单地提了一下。正如前面提到的，C++的产生是为了解决C程序过长的问题(尽管成千上万的程序员喜欢它的许多灵活特性)。在长程序中，很难记住所有部分的所有细节——如果能把数据和相关的函数结合到一个概念性的对象中，把它作为单一的实体，这样就可以把整个对象作为单位来使用，而不用记住内部数据处理的细节。事实上，对象和一种新的结构非常相似，但只有一点不同，它既有数据又有函数。例如，最常用的对象之一是堆栈。堆栈是一片特别划定的内存区域，以特殊的方式保存数据，保存数据是压进堆栈，访问数据则是弹出堆栈。弹出数据时，数据的顺序正好与压入的相反——如果按 1，2，3，4 和 5 的顺序压入，而连续弹出的顺序则为 5，4，3，2 和 1。

要建立自己的堆栈，必须要两块内存空间，一块用于保存数据，另一块用于实现压入、弹出操作的函数。堆栈还可能有一些内部函数用于监视堆栈。如果要记住这许多事情是很困难的，如果把它们合成一个逻辑概念，(C++的术语叫作封装 encapsulate)堆栈则会容易些，有了这样一个例子，现在开始使用C++。