教育部人才培养模式改革和开放教育试点教材

计算机应用专业系列教材

C++语言程序设计实验

李 宁　徐孝凯　张纪勇　编

12C-43
a

中央广播电视大学出版社

计算机应用专业系列教材

C++语言程序设计实验

李　宁　徐孝凯　张纪勇　编

中央广播电视大学出版社

前　言

本书是依据中央电大教学大纲为“C++语言程序设计”课程编写的指导上机实践的实验教材，应与主教材《C++语言程序设计》一书配合使用。

上机实践是学习程序设计的重要教学环节。只有通过上机实践，才能真正领会主教材中介绍的知识，才能正确灵活地利用C++语言中的各种要素，才能熟练地掌握作为集成化开发环境的编译系统，才有可能获得用程序设计解决实际问题的经验和技巧。本教材安排了11个实验，全面覆盖了主教材的教学内容，但又有所侧重。本教材系统地将那些需要通过实践环节掌握的知识和技能安排为实验的具体内容，以便学生通过实验能够领会和巩固所学知识，能够获得用程序设计解决实际问题的基本技能和初步经验。

当然，要想学好C++语言程序设计，仅靠这11次实验是不够的，这些实验只是对学生在上机实践方面的最低要求。有条件的读者应在阅读主教材和完成自测题、习题的过程中，随时上机，随时验证。特别是程序设计方面的习题，不上机就无法获得必要的反馈，就无法判断是否正确，因而就很难达到完成该类习题应达到的效果。

参加本书审定工作的学科专家有陈明、孙天正、晋良颖和李书通等，陈明教授担任主审。诸位学科专家对本书的初稿提出了宝贵的意见，特此致谢。

由于学识水平和时间的限制，不妥之处在所难免，欢迎批评指正。

编　者

2000年4月

实验一　C++程序的编辑、编译、连接和运行

一、实验目的

1. 熟悉编译系统的操作环境。
2. 掌握编辑、编译、连接和运行一个C++程序的基本过程。

二、预备知识

这里将简要介绍

① Visual C++

② C++ Builder

③ Borland C++

三种C++开发环境，读者应根据具体上机条件，选择阅读。

1. Visual C++环境下应用项目的建立与运行

Visual C++是Microsoft公司的产品，是一个使用广泛的C++集成化开发环境，最新版本是6.0。Visual C++不但是一个功能强大的C++编译器，而且还是一个对应用项目进行统一管理的工具软件。每一个开发过程中的应用系统就是一个应用项目(简称项目)。Visual C++既可用于管理基于Windows的应用项目，也可用于管理基于DOS的应用项目。基于DOS的应用系统也称为控制台应用系统，主教材中的程序实例以及本书实验中的程序都属于控制台应用系统，因此这里主要结合控制台应用系统(DOS应用系统)的开发过程介绍Visual C++开发环境。

(1) 应用项目的建立

一个应用项目(Project)是由若干编译单元(简称单元)组成的，而每个编译单元由一个程序文件(扩展名是CPP)及与之相关的头文件(扩展名是H)组成。在组成项目的所有单元中，必须有一个(也只能有一个)单元包含主函数main()的定义，这个单元称为主单元，相应的程序文件称为主程序文件。一个简单的控制台应用系统可以只有一个单元，即主单元。通过编译，每个单元生成一个浮动程序文件(也称为目标程序文件，扩展名是OBJ)。通过链接这些浮动程序文件，整个系统生成一个惟一的可执行文件，扩展名是EXE，而主名与项目名称相同。

若干个关系密切的项目构成一个工作区(有的编译系统称之为项目组)。例如，一个应用软件的演示版和正式版就可以作为两个项目通过一个工作区进行管理，这样，这两个软件可以很方便地共享某些资源(如函数定义)，并维持它们在功能、界面等多方面的一致性。工作区在建立时自动生成扩展名为DSW的工作区文件，以及扩展名为DSP，NCB，OPT等的其他文件，用来保存工作区信息和项目信息；这些文件都由编译系统自动维护。每个工作区在建立时必

须为之命名，同时生成一个以该名称为目录名的文件目录，所有与该工作区相关的文件都将保存在该目录中，其中包括与工作区名称相同的、扩展名为 DSW 的工作区文件。

对于本教材的使用者而言，只需考虑一个工作区中只有一个项目的情况。在这种情况下，不必专门地去建立工作区，在建立项目时相应的工作区就自动建立了，我们几乎感觉不到工作区的存在。以这种方式建立的工作区与所建项目同名。

现以主教材例 1.4 为例，说明一个简单应用系统项目的构成。假定清单中的程序保存在文件 area.cpp 中。作为一个简单的控制台应用系统，它只需一个编译单元，即主单元，项目中的文件包括：

area.cpp：主单元的程序文件
area.obj：主单元的浮动文件(目标文件)
area.exe：项目的可执行文件

这里没有列出主单元所用到的头文件 iostream.h，因为它是系统提供的头文件，是不能修改的，因此一般不把它们看成是项目的组成部分。

Visual C++ 能够识别项目中的各种文件之间的依赖关系，自动维持这些文件的一致性。例如，若某个头文件被修改了，则所有用 # include 命令插入该头文件的程序文件都将重新编译，更新原来的 OBJ 文件，并且重新进行链接，生成新的 EXE 文件。

下面以主教材例 1.4 为例，说明建立一个控制台应用项目的过程。

① 在 D 盘根目录下建立名为 area 项目(及工作区)。

A. 启动 Visual C++ 后，选择菜单命令 File|New，屏幕上出现 New 对话框，其中包括 Files(文件)、Projects(项目)、Workspace(工作区)和 Other Documents(其他文档)四个卡片，一般当前卡片是 Projects，如果不是，点击标有 Projects 的标签，使之成为当前卡片，如图 1-1 所示。

B. 在 Location(位置)下输入一个全路径目录名“D:\”(或点击旁边的...按钮，浏览选择该目录)，作为工作区目录；在 Project name(项目名称)下输入项目名称“area”；单击选中左边清单中的 Win32 Console Application(Win32 控制台应用系统)；最后点击 OK 按钮，屏幕出现 Win32 Console Application - Step 1 of 1(Win32 控制台应用系统-1 个步骤中的第 1 步)窗口。

C. 点击 Win32 Console Application - Step 1 of 1 窗口中的 Finish(完成)按钮，屏幕出现 New Project Information(新项目信息)窗口。

D. 点击 New Project Information 窗口中的 OK 按钮，项目建立完毕，在 D 盘根目录下多了一个名为 area 的目录。此时的屏幕如图 1-2 所示。

② 建立主程序文件 area.cpp。

A. 如图 1-2 所示，屏幕左边的窗口显示的是工作区及项目信息，其中包括 ClassView 和 FileView 两个卡片，如果当前卡片不是 FileView，点击标有 FileView 的标签，使之成为当前卡片。FileView 以文件夹的形式显示项目中已有的文件，其中的 area files 为项目 area 的文件夹。

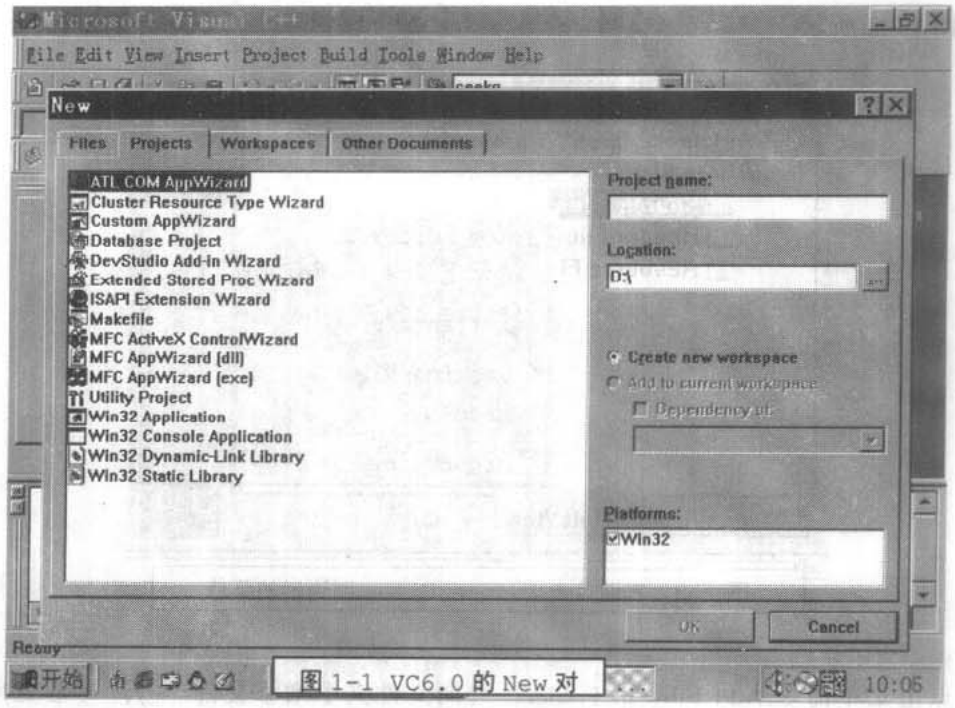

图 1-1 VC6.0 的 New 对

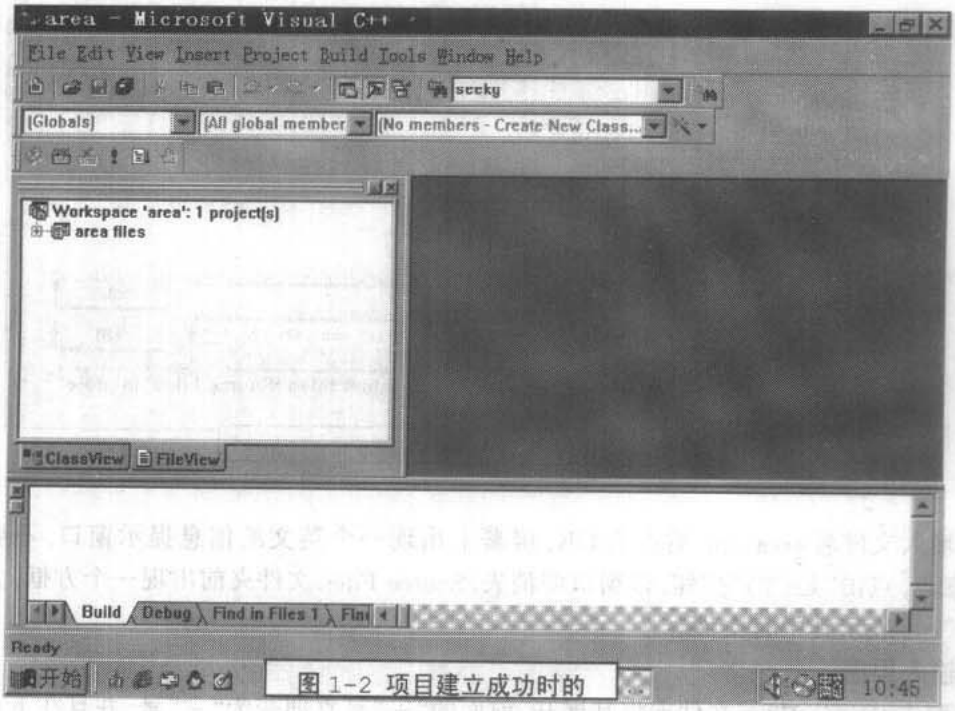

图 1-2 项目建立成功时的

B. 双击 area files 文件夹将其展开，在其下面显示出 Source Files(源程序文件)、Header Files(头文件)和 Resource Files(资源文件")三个子文件夹，右键单击 Source Files，弹出一个的菜单，如图 1－3 所示。

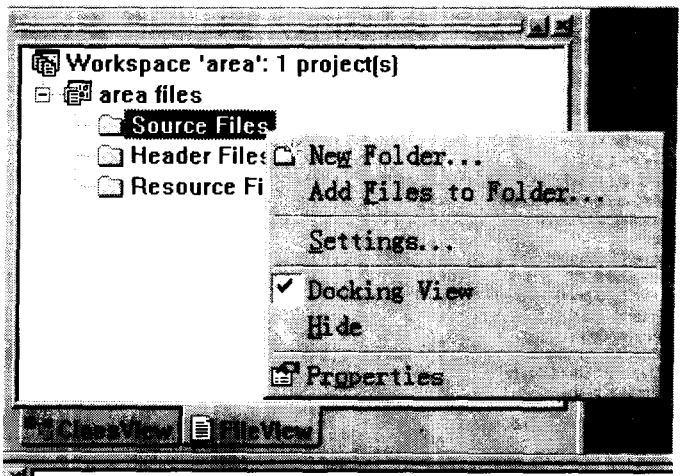

图 1-3　右键点击 Source Files 时弹出的菜单

C. 点击菜单命令 Add Files to Folder...(向文件夹中增添文件...)，屏幕上出现一个 Insert Files into Project(将文件插入到项目中)对话框，如图 1－4。

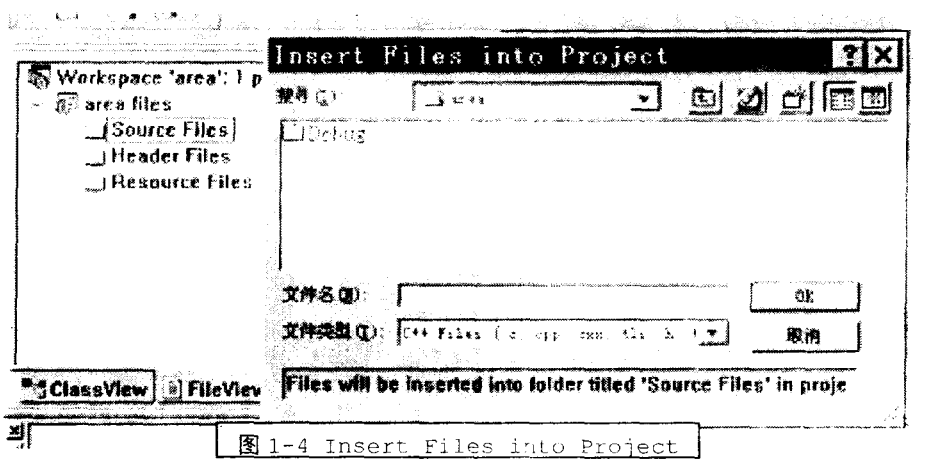

图 1-4 Insert Files into Project

D. 输入文件名 area.cpp 后点击 OK，屏幕上出现一个英文的信息提示窗口，不要去管它是什么意思，点击"是(Y)"按钮，该窗口即消失，Source Files 文件夹前出现一个方框，方框中有一个"＋"号，表示其中有文件。

③ 输入程序

A. 双击 Source Files 文件夹将其展开，前面的"＋"号立即变为"－"号，并且在下面显示出文件 area.cpp 的图标，如图 1－5 所示。

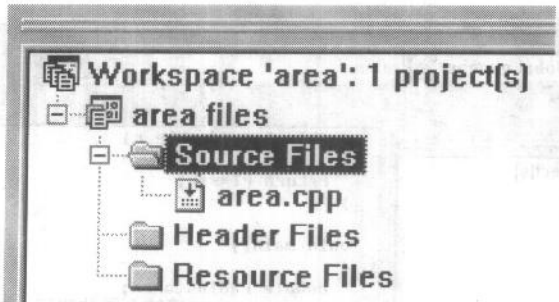

图 1-5 展开的 Source Files 文件夹

B. 双击文件 area.cpp 的图标，屏幕上出现一个英文的信息提示窗口，不要去管它是什么意思，点击“是(Y)”按钮，该窗口即消失，屏幕右侧出现一个针对文件 area.cpp 的文本编辑器窗口。

C. 在文本编辑器窗口中即可输入所有程序语句，如下面这样：

```
#include<iostream.h>

#define PI 3.1416

double Area(double r)
{
return PI * r * r;
}

void main()
{
  double radius, area;
  cout<<endl<<"请输入圆的半径:";
  cin>>radius;
  area = Area(radius);
  cout<<endl<<"圆的面积:"<<area;
}
```

程序输入后的屏幕画面(局部)如图 1-6 所示。

④其他有关的操作

A. 存盘：为了防止意外，应及时地保存有改动的文件。选择菜单命令 File | Save Workspace，将所有有改动的文件存入磁盘。

B. 关闭工作区：选择菜单命令 File | Close Workspace，关闭工作区中所有的文件。如果其中包含已改动而未存盘的文件，则将在存盘后关闭。

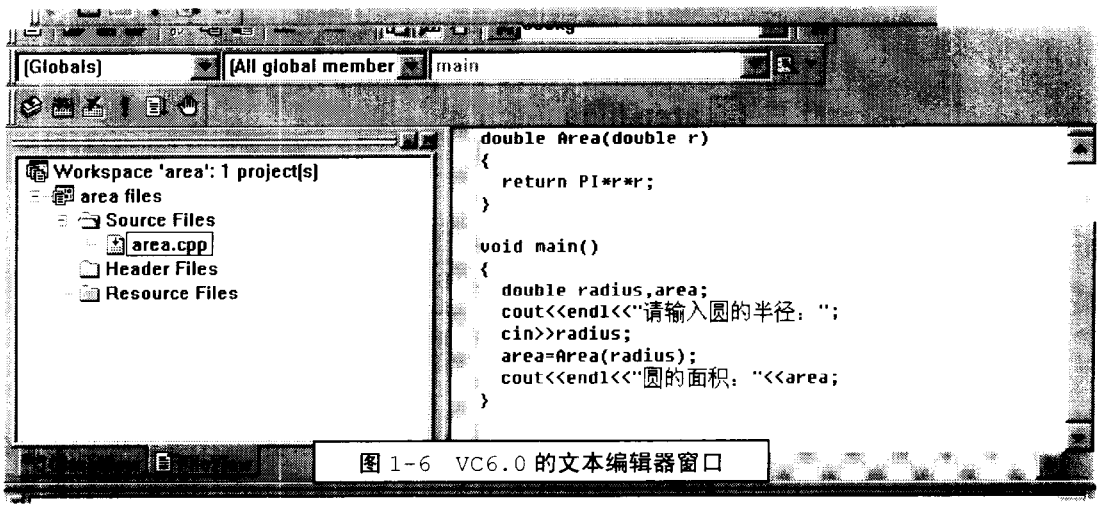

图 1-6 VC6.0 的文本编辑器窗口

C. 打开工作区:选择菜单命令 File|Open Workspace ,在弹出的 Open Workspace(打开工作区)对话框中选择要打开的工作区文件,然后点击"打开(O)"即可。例如要打开前面建立的 area 工作区,在 Open Workspace 对话框中应先选择 D 盘根目录下的 area 子目录,在列出的子目录清单中再选择工作区文件 area.dsw,该工作区即被打开。

D. 打开程序文件:在打开了工作区的情况下,双击项目文件夹将之展开,在显示出的文件图标中双击要打开文件的图标,该文件中的内容即出现在屏幕右侧的文本编辑器中。

(2) 程序的运行

Visual C++ 是 C++ 应用系统的集成开发环境,不但可以建立项目,对源程序进行管理,还可对源程序进行编译、链接,从而生成可执行文件,进而还可以运行和调试生成的可执行文件。

①程序的编译和链接

选择菜单命令 Build|Build 项目名.exe (或按 F7 键)或 Build|Rebuild All 即可对当前项目进行编译、链接,最后生成项目的可执行文件。注意,*项目名* 是指当前项目的名称,如果当前项目是 area,对应于菜单命令 Build*项目名*,实际显现的是 Build area.exe。Build 命令依据文件间的依赖关系,进行一致性检查,从而只编译那些必须编译的程序文件。例如,若一个头文件被修改了,则只有那些用到该头文件的程序文件才被重新编译一遍,其他文件并不重新处理。如果连续两次执行 Build 命令,则第二次将不做任何事情。由于 Build 的这一个不做无效劳动的特性,它的处理效率是很高的,这对于包含大量文件的大系统尤其重要。

也可以用选择菜单命令 Rebuild All 对当前项目进行编译连接。Rebuild 与 Build 不同的是,它不进行任何一致性检查,无条件地对项目中的程序文件统统重新处理。在未对源程序进行修改,但改变了项目某些设置(Settings)的情况下,就可以选择 Rebuild 命令强制性地把项目中的所有程序文件重新编译一遍,并重新链接,生成符合设置要求的可执行文件。

②程序的运行

选择菜单命令 Build|Start Debug|Go,或按 F5 键,即运行当前项目的可执行文件。如果项目中的某个文件已被修改,但又没有使用 Build 或 Rebuild 这样的菜单命令重新生成可执行程序,则有关文件将被重新编译链接,Go 执行的是新生成的可执行文件。也就是说,Go 在执行

程序前，首先执行 Build，因此在程序开发过程中很少直接使用菜单命令 Build Build 项目名.exe。

前面一直提到的"运行程序"，是指运行控制台应用程序。程序运行时会弹出一个 DOS 窗口，程序的输入输出均在这个窗口中进行。注意，程序运行一旦结束，这个 DOS 窗口就消失了，因此往往来不及观察运行的效果。为了使 DOS 窗口能够停留在屏幕上，可在 main( )函数后的末尾加上一个临时的键盘输入语句行，例如：

```
void main( )
{
   ...
   //下行是为了停留 DOS 窗口，以便于观察程序输出，不需要时应删除。
   cout<<endl<<endl<<"按回车键继续...";cin.get( );
}
```

运行程序的另一种方法就是直接在 DOS 界面下运行，即在 DOS 提示符＞后键入程序名。如果当前界面是 Windows，则应先启动 MS－DOS 方式。

2. C++ Builder 环境下应用项目的建立与运行

C++ Builder 是 Inprise 公司(原 Borland 公司)的产品，是另一个使用广泛的 C++ 集成化开发环境，最新版本是 4.0. 其显著特点是：

① 采用的 C++ 标准是最新的和最规范的。能在 Visual C++ 中通过的程序通常在 C++ Builder 中也能通过，但反过来就可能出现问题。可以通过设置使之与 Visual C++ 兼容，方法是：打开应用项目(见下面的解释)，选择菜单命令 Projrct | Options，在弹出的 Projrct Options 窗口中选择卡片 Advanced Compiler，点击右下角的 MFC compatibility 选中它，最后点击 OK 关闭 Projrct Options 窗口结束设置。

② 更加简化了 Windows 下应用程序的开发过程，降低了开发的复杂程度。

(1) 应用项目的建立

一个控制台应用项目是由若干编译单元(unit，简称单元)组成的，而每个编译单元由一个程序文件(扩展名是 CPP)及与之相关的头文件(扩展名是 H)组成。在组成项目的所有单元中，必须有一个(也只能有一个)单元包含主函数 main( )的定义，这个单元称为主单元，相应的程序文件称为主程序文件。一个简单的控制台应用系统可以只有一个单元，即主单元。通过编译，每个单元生成一个浮动程序文件(也成为目标程序文件)，扩展名是 OBJ。通过链接，整个系统生成一个惟一的可执行文件，扩展名是 EXE。最后，还必须有一个文件保存项目信息(项目的组成及项目可选项设置等信息)，这个文件称为项目文件，扩展名是 BPR。注意，项目文件、主程序文件以及可执行文件的文件名的主名部分相同，不同的只是扩展名。

现以主教材例 1.4 为例，说明一个简单应用系统项目的构成。假定清单中的程序保存在文件 area.cpp 中。作为一个简单的控制台应用系统，它只需一个编译单元，即主单元，项目中的文件包括：

area.bpr：项目文件
area.cpp：主单元的程序文件
area.obj：主单元的浮动文件(目标文件)

area.exe：项目的可执行文件

这里没有列出主单元所用到的头文件 iostream.h，因为它们都是系统提供的头文件，是不能修改的，因此一般不把它们看成是项目的组成部分。

C++ Builder 能够识别项目中的各种文件之间的依赖关系，自动维持这些文件的一致性。例如，若某个头文件被修改了，则所有用 #include 命令插入该头文件的程序文件都将重新编译，更新原来的 OBJ 文件，并且重新进行链接，生成新的 EXE 文件。

下面以主教材例 1.4 为例，说明建立一个控制台应用项目的过程。

①建立项目 area，生成项目文件 area.bpr 及其主程序文件 area.cpp。

A. 启动 C++ Builder 后，选择菜单命令 File|New，屏幕上出现 New Items 对话框，如图 1-7 所示。

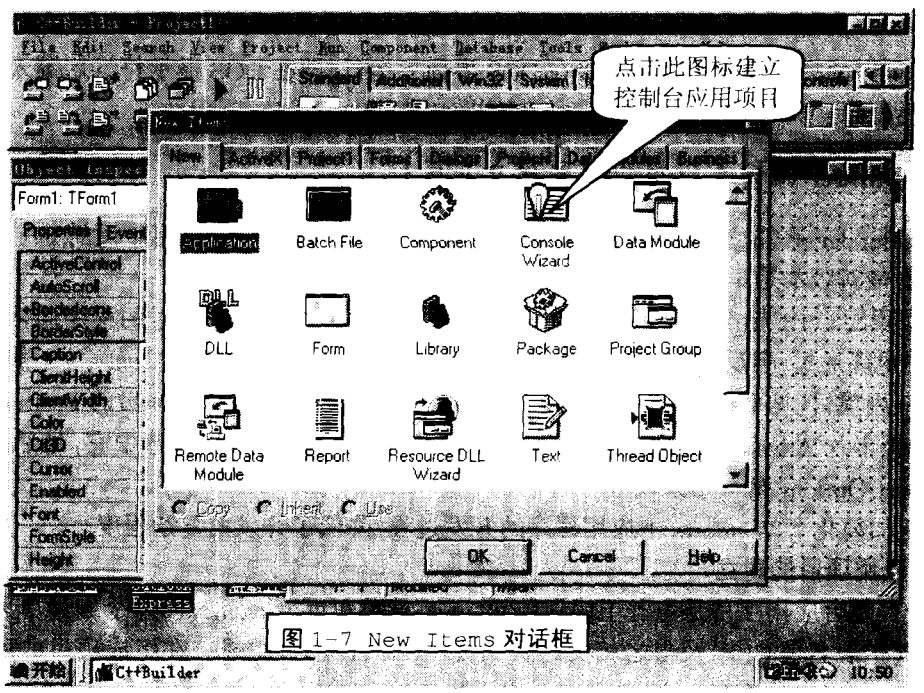

图 1-7 New Items 对话框

B. 双击 New Items 对话框的 Console Wizard 图标，屏幕上出现 Console Aplication Wizard（控制台应用程序向导）对话框，点击按钮 Finish，即建立了一个控制台应用项目，此时的屏幕画面如图 1-8。对于控制台应用系统，左边的 Object Inspector（对象观察器）窗口一般不用，可以关掉。右边是代码编辑器窗口，项目中的所有源文件（头文件和程序文件）都可以利用这个窗口进行输入、修改和调试。一个刚建立的项目只包含两个文件：项目文件和主程序文件，其临时文件名为 Project2.bpr 和 Project2.cpp。注意，代码编辑器可同时编辑若干个文件，每个文件占用一个卡片，卡片的标签登记着相应的文件名，可通过卡片标签选择当前要编辑的文件。在已生成的两个文件中，项目文件 Project2.bpr 是由系统自动维护的，不能直接对其进行编辑修改；因此一开始编辑器中只有一个卡片，其标签上的文件名是 Project2.cpp，是当前惟一的可以进行编辑修改的文件。Project2.cpp 中的内容是系统自动生成的程序框架，程序设计者应在此基础上补充输入自己的程序代码。

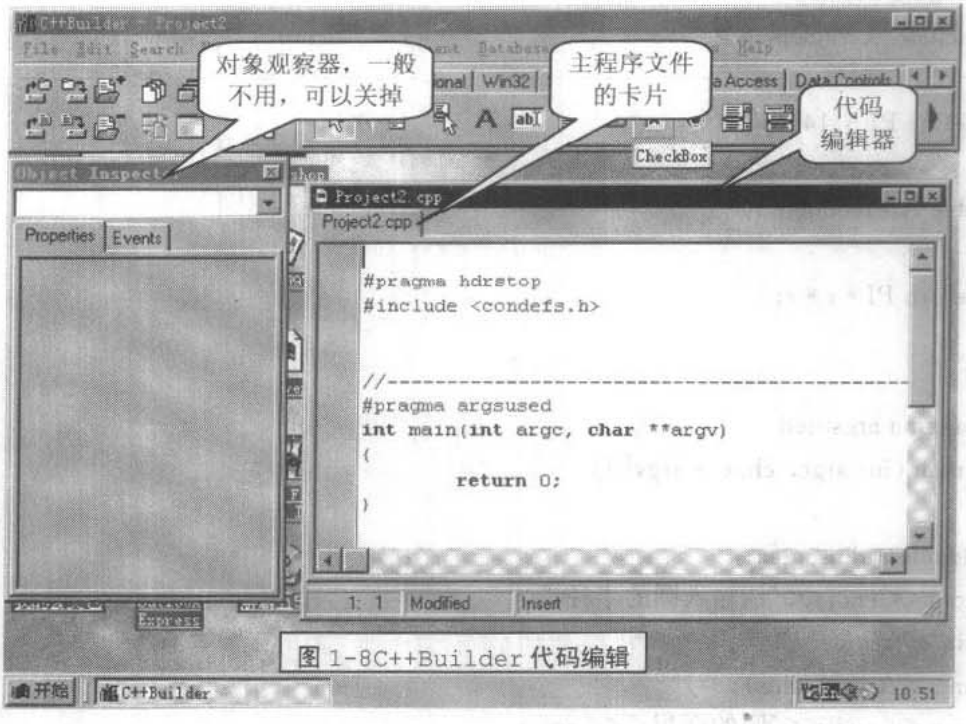

图 1-8C++Builder 代码编辑

C. 为项目命名：选择菜单命令 File|Save Project As...，屏幕出现保存文件对话框。选择一个文件夹以便保存项目中的各种文件，并在文件名处输入项目文件名 area.bpr，点击保存按钮。这样，项目文件和主程序文件就以 area.bpr 和 area.cpp 这两个正式名称保存在选定目录中，编辑器窗口标题栏及卡片标签上的 Project2.cpp 也相应地改为 area.cpp。

②输入程序

主单元程序文件 area.cpp 的程序框架如下：

```
#pragma hdrstop
#include <condefs.h>
//---
#pragma argsused
int main(int argc, char *argv[])
{
        return 0;
}
```

在#pragma hdrstop 的下面应输入一个或多个#include 命令行(也可以输入在该预处理命令之前，这涉及到预编译头文件的问题，这里不讨论)，插入有关的头文件。主函数的代码应输入在语句行 return 0;的前面。如果还需要定义其他函数或定义全局变量，应输入在命令行#pragma argsused 的前面。输入后的效果如下：

```
#include<iostream.h>
```

```
#pragma hdrstop
#include <condefs.h>
//---------------------------------------------------------------------
#define PI 3.1416

double Area(double r)
{
  return PI * r * r;
}

#pragma argsused
int main(int argc, char * argv[])
{
  double radius, area;
  cout<<endl<<"请输入圆的半径:";
  cin>>radius;
  area = Area(radius);
  cout<<endl<<"圆的面积:"<<area;
  return 0;
}
```

对于控制台应用系统,也可以删除程序框架,自己输入所有程序语句,如下面这样:

```
#include<iostream.h>

#define PI 3.1416

double Area(double r)
{
  return PI * r * r;
}

void main()
{
  double radius, area;
  cout<<endl<<"请输入圆的半径:";
  cin>>radius;
  area = Area(radius);
  cout<<endl<<"圆的面积:"<<area;
```

}

③其他有关的操作

A. 存盘：为了防止意外，应及时地保存有改动的文件。选择菜单命令 File|Save All，将所有有改动的文件存入磁盘。

B. 关闭项目：选择菜单命令 File|Close All，关闭项目中所有的文件。如果其中包含已改动而未存盘的文件，则将在存盘后关闭。

C. 打开项目和主程序文件：选择菜单命令 File|Open Project...，在 Open Project（打开项目文件）对话框中选择要打开项目的项目文件，例如 area.bpr，该项目即被打开，屏幕上出现该项目的代码编辑器。主程序文件随同项目一起打开，因此项目刚打开时，编辑器中已有对应于主程序文件（例如 area.cpp）的卡片。

（2）程序的运行

C++ Builder 是 C++ 应用系统的集成开发环境，不但可以建立项目，对源程序进行管理，还可对源程序进行编译、链接，从而生成可执行文件，进而还可以运行和调试生成的可执行文件。

①程序的编译和链接

选择菜单命令 Porject|Make 项目名 或 Porject|Build 项目名均可对当前项目进行编译、链接，最后生成项目的可执行文件。注意，项目名是指当前项目的名称，如当前打开的是项目 area，对应于菜单项 Make 项目名，实际显现的是 Make area；对应于菜单项 Build 项目名，实际显现的是 Build area。

Make 依据文件间的依赖关系，进行一致性检查，从而只编译那些必须编译的程序文件。例如，若一个头文件被修改了，则只有那些用到该头文件的程序文件才被重新编译一遍，其他文件并不重新处理。如果连续两次执行 Make，则第二次将不做任何事情。由于 Make 的这个不做无效劳动的特性，它的处理效率是很高的，这对于包含大量文件的大系统尤其重要。

Build 与 Make 不同的是，它不进行任何一致性检查，无条件地对项目中的程序文件统统重新处理。在未对源程序进行修改，但改变了项目某些设置（Options）的情况下，就可以选择 Build 命令强制性地把项目中的所有程序文件重新编译一遍，并重新链接，生成符合设置要求的可执行文件。

②程序的运行

选择菜单命令 Run|Run，或按 F9 键，即运行当前项目的可执行文件（见图 1－9）。如果项目中的某个文件已被修改，但又没有使用 Make 或 Build 这样的菜单命令重新生成可执行程序，则有关文件将被重新编译链接，Run 执行的是新生成的可执行文件。也就是说，Run 在执行程序前，首先执行 Make，因此在程序开发过程中很少直接使用菜单命令 Make。

前面一直提到的“运行程序”，是指运行控制台应用程序。程序运行时会弹出一个 DOS 窗口，程序的输入输出均在这个窗口中进行。注意，程序运行一旦结束，这个 DOS 窗口就消失了，因此往往来不及观察运行的效果。为了使 DOS 窗口能够停留在屏幕上，可在 main()函数后的末尾处加上一个临时的键盘输入语句行，例如：

```
int main(int argc, char * argv[])
{
```

```
...
//下行是为了停留 DOS 窗口,以便于观察程序输出,不需要时应删除。
cout<<endl<<endl<<"按回车键继续...";cin.get();

return 0;
}
```

运行程序的另一种方法就是直接在 DOS 界面下运行,即在 DOS 提示符>后键入程序名。如果当前界面是 Windows,则应先启动 MS-DOS 方式。

3. Borland C++ 环境下应用项目的建立与运行

Borland C++ 是 Borland 公司的早期产品,是一个在 DOS 环境下使用广泛的 C++ 集成化开发环境,最后的版本是 3.1。另外,Turbo C++ 也是 Borland 公司的产品,与 Borland C++ 非常类似,因此使用 Turbo C++ 也可参考下面的说明。

Borland C++ 在 DOS 下运行,因此对硬件条件要求较低,例如在 286 电脑上就能很好地运行。Borland C++ 采用的是较为早期的 C++ 标准,但对于本课程的学习影响不算太大。

(1) 应用项目的建立

一个控制台应用项目是由若干编译单元(简称单元)组成的,而每个编译单元由一个程序文件(扩展名是 CPP)及与之相关的头文件(扩展名是 H)组成。在组成项目的所有单元中,必须有一个(也只能有一个)单元包含主函数 main()的定义,这个单元称为主单元,相应的程序文件称为主程序文件。一个简单的控制台应用系统可以只有一个单元,即主单元。通过编译,每个单元生成一个浮动程序文件(也称为目标程序文件),扩展名是 OBJ。通过链接,整个系统生成一个惟一的可执行文件,扩展名是 EXE。最后,还必须有一个文件保存项目信息(项目的组成及项目可选项设置等信息),这个文件称为项目文件,扩展名是 BPR。注意,项目文件、主程序文件以及可执行文件的文件名的主名部分相同,不同的只是扩展名。

现以主教材例 1.4 为例,说明一个简单应用系统项目的构成。假定清单中的程序保存在文件 area.cpp 中。作为一个简单的控制台应用系统,它只需一个编译单元,即主单元,项目中的文件包括:

area.prj:项目文件
area.cpp:主单元的程序文件
area.obj:主单元的浮动文件(目标文件)
area.exe:项目的可执行文件

这里没有列出主单元所用到的头文件 iostream.h,因为它们都是系统提供的头文件,是不能修改的,因此一般不把它们看成是项目的组成部分。

Borland C++ 能够识别项目中的各种文件之间的依赖关系,自动维持这些文件的一致性。例如,若某个头文件被修改了,则所有用 #include 命令插入该头文件的程序文件都将重新编译,更新原来的 OBJ 文件,并且重新进行链接,生成新的 EXE 文件。

下面以主教材例 1.4 为例,说明建立一个控制台应用项目的过程。

① 在 D 盘根目录下的 area 目录下建立项目 area,生成项目文件 area.prj。

A. 如果 D 盘根目录下尚未建立目录 area，先建立该目录。

B. 启动 Borland C++ 后，选择菜单命令 Project | Open Project，屏幕上出现 Open Project File(打开项目文件)对话框。

C. 输入全路径项目文件名 D: \ AREA \ AREA.PRJ，然后点击 OK 即可。注意，文件 AREA.PRJ 必须不存在，否则就不是建立项目，而是打开项目了。若项目建立成功，屏幕下方出现一个 Project: AREA 窗口(项目窗口)。从项目窗口可以看出，刚建立的项目是空的，没有任何文件。

② 建立主程序文件 area.cpp。

A. 选择菜单命令 Project | Add Item...，屏幕上出现 Add to Project List(添加到项目清单中...)对话框。

B. 输入文件名 area.cpp，然后点击 Add 按钮，该文件名即出现在项目窗口中。由于不需要添加更多的文件，点击 Done 按钮关闭 Add to Project List 对话框。

C. 双击项目窗口中的文件名 AREA.CPP，该文件项目窗口的上方即出现一个针对文件 area.cpp 的文本编辑器窗口

③ 输入程序

文本编辑器窗口中输入所有程序语句，如下面这样：

```
#include<iostream.h>

#define PI 3.1416

double Area(double r)
{
  return PI * r * r;
}
void main()
{
  double radius, area;
  cout<<endl<<"请输入圆的半径:";
  cin>>radius;
  area = Area(radius);
  cout<<endl<<"圆的面积:"<<area;
}
```

④ 其他有关的操作

A. 存盘：为了防止意外，应及时地保存有改动的文件。选择菜单命令 File | Save(或按 F2 键)将已改动的当前文件存入磁盘，选择菜单命令 File | Save All 将所有有改动的文件存入磁盘。

B. 关闭文件：把光标移到该文件的文本编辑器窗口，使之成为当前文件，然后选择菜单命令 Window | Close 或按 ALT + F3 键，编辑器即消失，表示文件已关闭。选择菜单命令 Window

|Close All 将关闭所有打开的文件。如果文件被改动过，而又没有存盘，关闭文件时会出现一个提示信息对话框，由操作者决定是否存盘。

C. 关闭项目：上面介绍的关闭文件的方法也可以使项目窗口消失，但那只是把项目窗口隐藏起来，并没有关闭它。隐藏的项目窗口可以通过执行菜单命令 Window|Project 重新显现出来。真正要关闭项目，必须选择菜单命令 Project|Close，它不但关闭项目文件，属于该项目的已打开的任何文件也同时关闭。

D. 打开项目和主程序文件：选择菜单命令 Project|Open Project，屏幕上出现 Open Project File(打开项目文件)对话框，输入项目文件的全路径文件名，点击 OK 按钮，项目文件即被打开。如果此时没有出现项目窗口，说明该窗口处于隐藏状态，只须选择执行菜单命令 Window|Project，即可使项目窗口显现出来。点击项目窗口中的主程序文件名，即可打开该程序文件。也可以选择菜单命令 File|Open(或按 F3 键)来打开一个文件，这种方法常用于打开非当前项目中的文件。

(2) 程序的运行

Borland C++ 是 C++ 应用系统的集成开发环境，不但可以建立项目，对源程序进行管理，还可对源程序进行编译、链接，从而生成可执行文件，进而还可以运行和调试生成的可执行文件。

①程序的编译和链接

选择菜单命令 Compile|Make(或按 F9 键)，或选择菜单命令 Project|Build All 均可对当前项目进行编译、链接，最后生成项目的可执行文件。

Make 依据文件间的依赖关系，进行一致性检查，从而只编译那些必须编译的程序文件。例如，若一个头文件被修改了，则只有那些用到该头文件的程序文件才被重新编译一遍，其他文件并不重新处理。如果连续两次执行 Make，则第二次将不做任何事情。由于 Make 的这个不做无效劳动的特性，它的处理效率是很高的，这对于包含大量文件的大系统尤其重要。

Build All 与 Make 不同的是，它不进行任何一致性检查，无条件地对项目中的程序文件统统重新处理。在未对源程序进行修改，但改变了项目某些设置(Options)的情况下，就可以选择 Build All 命令强制性地把项目中的所有程序文件重新编译一遍，并重新链接，生成符合设置要求的可执行文件。

②程序的运行

选择菜单命令 Run|Run，或按 Ctrl + F9 键，即运行当前项目的可执行文件。如果项目中的某个文件已被修改，但又没有使用 Make 或 Build All 这样的菜单命令重新生成可执行程序，则有关文件将被重新编译链接，Run 执行的是新生成的可执行文件。也就是说，Run 在执行程序前，首先执行 Make，因此在程序开发过程中很少直接使用菜单命令 Make。

前面一直提到的“运行程序”，是指运行控制台应用程序。程序运行时会生成一个称为用户屏幕(user screen)的全屏幕窗口，程序的输入输出均在这个窗口中进行。注意，程序运行一旦结束，这个窗口就消失了，往往来不及观察运行的效果。选择菜单命令 Window|User screen(或按 Alt + F5 键)可显现用户屏幕，观察完毕后按任意键又可回到编译系统主窗口。

运行程序的另一种方法就是直接在 DOS 界面下运行，即在 DOS 提示符>后键入程序名。

注意，如果计算机没有安装鼠标，可用等价的键盘操作达到同样的目的。应注意屏幕下面的一行关于功能键的提示，特别是 F10——光标移到顶行主菜单，Esc——光标从主菜单回到