应用型本科信息大类专业“十二五”规划教材

C++程序设计案例教程

主　编　吴　艳　冉　娟
副主编　殷　妍　朱　林　姜志明
　　赵凤怡　李　聪

華中科技大學出版社
http://www.hustp.com

应用型本科信息大类专业“十二五”规划教材

C++程序设计案例教程

主　编　吴　艳　冉　娟
副主编　殷　妍　姜志明　朱　林
赵凤怡　李　聪

华中科技大学出版社
中国·武汉

内容简介

“C++程序设计基础”是计算机专业的一门专业基础课程，同时，C++语言也是编程爱好者所需要了解和掌握的基础语言，因此，也就出现了种类繁多、各具特色及针对不同层次学生的C++程序设计教材。本书以C++程序设计的理论知识为基础，以案例为主线，采取由浅入深、逐步递进的方式阐述了C++程序设计的理论知识和具体应用。也就是说，本书既没有忽视理论的重要性，同时也注重学生实践能力、应用能力及创新能力的培养。

本书共9章，其中第1章为C++概述；第2章为C++程序设计基础；第3章为函数；第4章为指针和引用；第5章为类和对象；第6章为继承与派生；第7章为运算符重载；第8章为文件与流类库；第9章为模板与异常处理。

本书的特点是紧贴计算机专业教学的需求，图文并茂；以基本理论知识为基础，以实际应用为主线，通过形象的展示，将抽象的知识生动化，同时以吉祥航空公司货物运输实际项目为案例，将分散的知识点连接成串。

为了方便教学，本书还配有电子课件等教学资源包，任课教师和学生可以登录“我们爱读书”网(www.ibook4us.com)免费注册下载，也可以发邮件至hustpeiit@163.com免费索取。

本书适合作为普通本科院校、独立学院、高职高专等学校计算机专业及其他相关专业的程序设计基础教材，也可以作为广大编程爱好者及技术人员学习编程技巧、参加计算机专业技术资格考试以及从事计算机软件开发研究与应用的参考书。

图书在版编目(CIP)数据

C++程序设计案例教程/吴艳，冉娟主编. —武汉：华中科技大学出版社，2014.5
ISBN 978-7-5680-0118-2

Ⅰ.①C… Ⅱ.① 吴… ② 冉… Ⅲ.①C语言-程序设计-高等学校-教材 Ⅳ.①TP312

中国版本图书馆CIP数据核字(2014)第100482号

C++程序设计案例教程　　吴　艳　冉　娟　主编

策划编辑：康　序
责任编辑：康　序
封面设计：李　嫚
责任校对：周　娟
责任监印：张正林
出版发行：华中科技大学出版社(中国·武汉)
武昌喻家山　邮编：430074　电话：(027)81321915
录　　排：武汉正风天下文化发展有限公司
印　　刷：华中理工大学印刷厂
开　　本：787mm×1092mm　1/16
印　　张：16.5
字　　数：417千字
版　　次：2014年9月第1版第1次印刷
定　　价：35.00元

前言 PREFACE

20世纪90年代以来，面向对象程序设计(object oriented programming，OOP)成为计算机程序设计的主流，其设计思想逐步被越来越多的软件设计人员所接受。C++语言是在C语言的基础上发展起来的，它不仅集成了C语言灵活高效、功能强大、可移植性好等特点，而且引入了面向对象程序设计的思想和机制，可以在很大程度上提高编程能力，减少软件维护的开销，增强软件的可扩展性和可重用性。

C++是优秀的程序设计语言之一，它以其独特的语言机制在计算机科学领域中得到了广泛的应用，并逐步为广大程序设计人员所青睐。

本书以短小精悍的例题作为课内案例，针对每个章节的知识点进行详解及扩充，课内案例均是人们生活中喜闻乐见的问题，这样让读者更容易理解。此外，全书又是以吉祥航空公司货运费用计算案例为背景，分解C++语言中主要知识点，更形象地进行知识的应用，让读者在对知识点的掌握有侧重点的同时学会如何分析实际问题、如何解决实际问题，提高读者的实践能力。

本书共9章，其各章节的内容大致如下。

第1章介绍了C++语言的发展和特点、C++程序的基本结构，以及完成C++程序开发的完整步骤。

第2章介绍了C++程序基础知识，主要包括程序中的基本元素，基本数据类型和用户定义数据类型及对应的表达式等，此外，详细介绍了三种基本程序设计结构。

第3章介绍了函数，主要包括函数的分类，函数的定义与声明，函数的调用以及常用的特殊函数。

第4章介绍了指针和引用，主要介绍指针的定义、初始化以及应用，引用的定义、初始化以及指针与引用的关系。

第5章介绍了类，主要介绍类的定义及应用，构造函数与析构函数的作用及定义，静态成员和友元函数的应用。

第6章介绍了继承与派生，主要介绍了继承的分类以及派生类的定义与应用。

第7章介绍了运算符的重载。

第8章介绍了文件与流类库。

第9章介绍了模板与异常处理。

本书的每一章后面均有小结以及习题，习题中的练习是为验证读者对章节中知识的消化、理解程度。同时，促进读者对章节知识侧重点的理解与应用。书后的实验内容是一个完整项目的分解，通过实验课上的练习，有助于提高读者的实际操作能力及运用能力。本书还配有教学大纲、实验大纲以及电子课件等相关教学资源。

本书由辽宁科技学院吴艳、天津大学仁爱学院冉娟担任主编，天津大学仁爱学院殷妍、大连科技学院姜志明、东南大学成贤学院朱林、华中师范大学武汉传媒学院赵凤怡、武汉科技大学城市学院李聪担任副主编。其中，第1～4章由吴艳编写，第5章由殷妍编写，第6章由朱林编写，第7章由姜志明编写，第8章由李聪编写，第9章由赵凤怡编写。另外，实验项目由冉娟编写，并且由冉娟进行习题答案整理以及对全书进行统稿。

为了方便教学，本书还配有电子课件等教学资源包，相关教师和学生可以登录“我们爱读书”网(www.ibook4us.com)免费注册下载，或者发邮件至 hustpeiit@163.com 免费索取。

由于编者水平有限，书中错误和疏漏之处在所难免，恳请广大读者批评指正。

编　者

2014年4月

目录

CONTENTS

第1章 C++概述

C++语言作为面向程序设计语言的基础，其鲜明特色和强大功能是其他语言所不能比拟的。通过了解C++语言的发展及特点、C++源程序的基本组成和C++源程序开发的基本过程，能够对面向对象程序设计的开发有初步了解，为后续程序设计的学习奠定坚实的基础，本章要求重点掌握C++源程序的基本构成以及C++源程序的实现方法与步骤。

1.1 C++发展和特点

20世纪90年代以来，面向对象程序设计(object oriented programming，OOP)成为计算机程序设计的主流，其设计思想逐步被越来越多的软件设计人员所接受。C++语言是在C语言的基础上发展起来的，它完全兼容了C语言，不仅集成了C语言灵活高效、功能强大、可移植性好等特点，而且引入了面向对象程序设计的思想和机制，可以在很大程度上提高编程能力，减少软件维护的开销，增强软件的可扩展性和可重用性。

C++是优秀的程序设计语言之一，由于C++语言在兼容C语言的基础上，添加了自身个性化的特性，因此比C语言更容易为人们所学习和掌握。C++语言以其独特的语言机制在计算机科学领域中得到了广泛的应用，并逐步为广大程序设计人员所青睐。

1.1.1 C++的发展历史

C语言是贝尔实验室的Dennis Ritchie在B语言的基础上研发出来的，于1972年实现了最初的C语言。由于C语言是与硬件无关的程序设计语言，因此，用C语言编写的程序可以移植到大多数计算机上。C语言被人们所熟知是它作为UNIX操作系统的开发语言身份的亮相。到20世纪70年代末，C语言开发程序的技术已经发展得相当成熟了。

C语言具有以下优点。

(1) 语言简洁、紧凑，使用方便、灵活。C语言只有32个关键字，程序书写形式自由。

(2) 丰富的运算符(如逗号运算符、赋值运算符等)和数据类型(基本数据类型和用户自定义数据类型)。

(3) C语言可以直接访问内存地址，并且能进行位操作，是较为适合开发操作系统的语言之一。

(4) 生成的目标代码质量高，程序运行效率高，可移植性好。

当然，C语言也有很多不足，体现在以下几个方面。

(1) C语言类型检查机制相对较弱，使得程序中的某些错误不能在编译时发现。

(2) C语言本身几乎不支持代码重用，因此，程序开发的效率不是很高。

(3) 用C语言不适合开发较为复杂、规模较大的软件。因为，当程序规模达到一定的程度时，程序员就很难控制其复杂性。

因此，1980年贝尔实验室的Bjarne Stroustrup开始对C语言进行改进和扩充。最初的成果称为“带类的C”，1983年正式命名为C++，在先后经历了3次对C++的修订后，于1994年制定了ANSI C++标准的草案。由此可见，C++语言是建立在C语言的基础上

的，C++语言包含了C语言的全部特征、属性和优点，同时又增加了面向对象的程序支持。

1.1.2 C++的特点

C++语言包括过程性语言部分和类部分。过程性语言部分与C语言并无本质上的差别，只是在版本上有所提高，在功能上有所增强。类部分是C语言中所没有的，它是面向对象程序设计的主体。

目前，程序设计方法正在从结构化程序设计向面向对象程序设计过渡。从根本意义上说，C语言能够很好地支持结构化程序设计，而C++语言更能很好地支持面向对象程序设计。C语言程序设计开发锻炼了程序员进行抽象程序设计的能力，而C++语言则是C语言的扩展，它分享了C语言的许多技术风格，同时增加了自己的特色，其主要特点包括以下几点。

(1) 由于C++语言全面兼容C语言，因此许多C语言代码不经修改就可以在C++语言中使用。

(2) 用C++语言编写程序可读性更好，代码结构也更为合理，生成代码的质量较高。

(3) C++语言从开发时间、费用，软件的可重用性、可扩充性、可维护性和可靠性等方面相对C语言都有很大的提高。

(4) C++语言支持面向对象程序设计，比较符合人类解决实际问题的思维方式。

1.2 C++程序基本结构

1.2.1 简单的C++小程序

下面通过编制一个小的程序来认识一下C++程序的基本结构。

【例 1-1】 简单的C++小程序。

```
#include <iostream.h>←编译预处理命令
voidmain()
 {                          ←程序主体
   cout<<"It's a C++ Programming!"<<endl;   //在显示器上显示结果
 }                                          ↑注释部分
```

运行结果为：

```
It's a C++ Programming!
```

1.2.2 程序构成

上一小节介绍了用C++语言编写的小程序，本节则详细介绍C++程序的构成。C++程序可以理解为是使用C++语言给计算机写了一份工作指示书，要求计算机按照其要求来完成一系列的工作。这份指示书就像其他一些必要的文书一样需要有固定的格式。一般情况下，C++程序结构由三部分组成：注释、编译预处理和程序主体。

1. 注释

注释是程序员为某一条语句或是一段代码所做的必要说明，其目的是为了提高程序的可读性。一般可将注释分为两种类型：序言性注释和解释性注释。序言性注释是用在程序开头，说明程序或文件的一些概要信息，包括程序或文件的名字、用途、编写时间、编

写人等；解释性注释是用在程序中某一条语句的后面，是为了解释或介绍该语句的功能。注释内容是为阅读程序的读者提供相关的参考信息，在程序执行过程中会自动略过这些注释语句。

注释有两种格式："//"一般只能写一行注释，常放在程序的开头；"/*……*/"可以包括多行注释内容，常放在程序中的语句后面。

【例 1-2】 编写程序实现求一个矩形面积(要求：理解注释的应用)。

```
//完成求任意大小的矩形面积功能
#include <iostream.h>
void main()
{
   int a,b,s;/* 变量 a 和 b 用来存放矩形的长和宽的值，s 用来保存其面积。*/
   cin>>a>>b;
   s=a*b;     //求矩形面积
   cout<<"矩形面积是："<<s<<endl;
}
```

2. 预处理命令

编译预处理命令也称为预处理器，以"#"开头，它不属于 C++程序的语句部分，而是预处理命令行。"#"必须是该行除了任何空白字符外的第一个字符，"#"后是指令关键字，在关键字和#号之间允许存在任意多个的空白字符。该指令将在编译器进行编译之前对源代码进行某些转换。其格式如下。

```
#< 预处理命令> < 预处理信息>
```

常见的预处理命令包括 3 类：文件包含命令(include)、宏定义命令(define)和条件编译命令(undef、if、endif 等)。下面详细介绍几个常用的预处理命令。

1) 文件包含命令 include

该命令的作用是在指令处展开被包含的文件。包含可以是多重的，也就是说，一个被包含的文件中还可以包含其他文件。在程序中包含文件有以下两种格式。

```
#include  <文件名.扩展名>        //例如 #include <iostream.h>
#include  "文件名.扩展名"        //例如 #include "math.h"
```

第一种格式是用尖括号把文件括起来，这种格式告诉预处理程序在编译器自带的或外部库的文件中搜索被包含的文件。第二种方法是用双引号把文件括起来，这种格式告诉预处理程序在当前被编译的应用程序的源代码文件中搜索被包含的文件，如果找不到，再搜索编译器自带的文件。两种格式没有绝对的区别，可以互换使用，主要是对存放在不同的位置的文件进行查找时，在程序的执行速度上的区别。

2) 宏定义命令 define

该命令的作用是定义一个代表特定内容的标识符。预处理过程会把源代码中出现的宏标识符替换成宏定义时的值。宏定义分为两种情况：带参数的宏定义和不带参数的宏定义。

(1) 不带参数的宏定义，其定义格式如下。

```
#define  标识符  标识符代表的代码
```

其功能是，在其定义之后的源代码中，遇到该标识符就用相应的代码来替代。这种宏把程序中要用到的一些全局变量提取出来，用一些容易记忆的标识符代替。例如：

```
#define MAX  10
int  array[MAX];          //定义了一个一维数组 array,数组里含有 10 个元素
for(int i=0;i<MAX;i++)    /*循环控制变量 i 的结束条件是 i 大于等于 10 时*/
```

上述例子中,第一行定义了标识符 MAX,它有特定的含义:代表一个具体的整数值 10。第二行中,利用标识符 MAX 给出了数组所能容纳的最大数组元素个数为 10。第三行中,利用标识符 MAX 控制着循环条件,循环控制变量 i 从 0 开始循环,到 MAX－1 结束,共循环 MAX 次。

注意:(1) 在 C++语言的程序中,允许多次使用定义过的宏定义。

(2) 作为一种约定,习惯上总是用大写字母来定义宏名,这样易于把程序中的宏标识符和一般变量标识符区别开来。

(3) 宏表示的值可以是一个常量,也可以是一个常量表达式,其中还允许包括前面已经定义过的宏标识符。例如:

```
#define ONE 1
#define TWO 2
#define THREE (ONE+TWO)
```

上面的宏定义使用了括号,尽管它们并不是必需的,但基于某些特殊情况的考虑,应该加上圆括号,这样更安全。例如:

```
int six=THREE*TWO;
```

预处理过程把上面的一行代码转换成式子:six=(ONE+TWO)*TWO;若上述的宏定义替换语句中没有加圆括号的话,则该式子就转换成:six=ONE+TWO*TWO;可见,这样的转换并非用户的本意。所以一般当宏替换的文本是表达式的时候应尽可能地用圆括号括起来。

(4) 宏还可以代表一个字符串常量。例如:

```
#define INSTRUCTION  "C++ Version 6.0"
```

(2) 带参数的宏定义,其定义格式如下。

```
#define 标识符(参数表)  标识符代表的代码
```

带参数的宏和函数调用看起来有些相似。例如:

```
#define  area(x)   ((x)*(x))
```

可以用任何数值表达式甚至函数调用来代替参数 x,需要读者注意的是圆括号的使用。宏展开后完全包含在一对圆括号中,而且参数也包含在圆括号中,这样就保证了宏和参数的完整性。例如:

```
int num=5+6;
result=area(num);
```

展开后为((5+6)*(5+6)),如果没有圆括号,则展开后就变为 5+6*6+6 了。

3) 条件编译指令

条件编译指令将决定哪些代码被编译,而哪些代码不被编译。可以根据表达式的值或某个特定的宏是否被定义来确定编译条件。

(1) #if 指令。#if 指令用于检测跟在指令关键字后的常量表达式。如果表达式为真,则编译后面的代码,直到出现#else、#elif 或#endif 指令为止,否则就不编译。

(2) #endif 指令。#endif 用于终止#if 预处理指令。

【例 1-3】 条件编译指令#if 与#endif 的应用案例。

```
#include <iostream.h>
#define  DEBUG  0
void main( )
{
  #if DEBUG                    //DEBUG值为0
    cout<<"Debugging"<<endl;
  #endif
    cout<<"Running"<<endl;
}
```

运行结果为：

```
Running
```

说明：由于程序定义 DEBUG 宏代表 0，所以 #if 条件为假，不编译后面的代码直到 #endif，所以程序直接输出 Running。读者自行思考，若在定义宏 DEBUG 时，将其代表的代码变为 1，则该程序的运行结果是什么呢？

(3) #ifdef 和 #ifndef 指令。#ifdef 表示如果宏定义存在，则执行后面的代码直到 #endif；#ifndef 与 #ifdef 含义正好相反。

【例 1-4】 #ifdef 和 #ifndef 的应用案例。

```
#include <iostream.h>
#define DEBUG
void main()
{
  #ifdef DEBUG
    cout<<"yes\n";
  #endif
  #ifndef DEBUG
    cout<<"no\n";
  #endif
}
```

运行结果为：

```
yes
```

说明：上述例题中在第二行定义了宏名 DEBUG，所以在 #ifdef 中为真，执行后续语句直到 #endif。读者自行思考，若取消上述例题中第二行的宏名 DEBUG 的定义，则该程序的执行结果是什么呢？

注意：#ifdefined 等价于 #ifdef，#if! defined 等价于 #ifndef。

(4) #error 指令。#error 指令将使编译器显示一条错误信息，然后停止编译。编译程序时，只要遇到 #error 就会跳出一个编译错误，其目的就是保证程序是按照用户所设想的那样进行编译的。

【例 1-5】 #error 指令的应用案例。

```
#include"iostream.h"
void main( )
{
   #define CONST_NAME"CONST_NAME"
   cout<<CONST_NAME<<endl;
   #undef CONST_NAME
   #ifndef CONST_NAME
   #error No defined Constant Symbol CONST_NAME
   #endif
   ⋮
}
```

在编译的时候输出如下编译信息：

```
fatal error C1189:#error:No defined Constant Symbol CONST_NAME
```

1.3 VC++6.0集成开发环境

Microsoft Visual C++ 6.0(以下简称 VC++ 6.0)是微软公司出品的高级可视化计算机程序开发工具，界面友好、使用方便，可以识别 C/C++程序。VC++ 6.0 可以在“独立文件模式”和“项目管理模式”两种模式下使用。当只有一个文件时，可以使用独立文件模式；当程序比较大，一个程序需要由多个源文件组成时，使用项目管理模式，这时所有源程序文件合在一起共同构成一个程序，在 C++中称为一个“项目”。下面简单介绍独立文件模式下该开发环境的使用。

1.3.1 VC++ 6.0界面

VC++ 6.0 界面如图 1-1 所示。

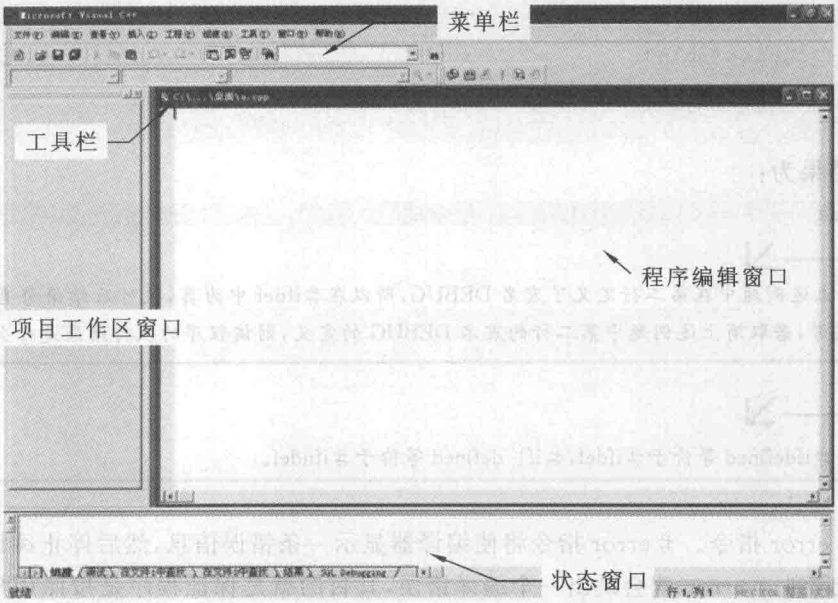

图 1-1 VC++ 6.0 运行环境界面

1.3.2 C++源程序的实现过程

1. 进入 VC++ 6.0 集成环境

选择“开始”→“程序”→“Microsoft Visual Studio 6.0”→“Microsoft Visual C++ 6.0”命令，或者双击桌面上 Visual C++ 6.0 图标，就能进入 Visual C++ 6.0 集成环境。此时，屏幕上弹出 VC++ 6.0 的主窗口，如图 1-2 所示。

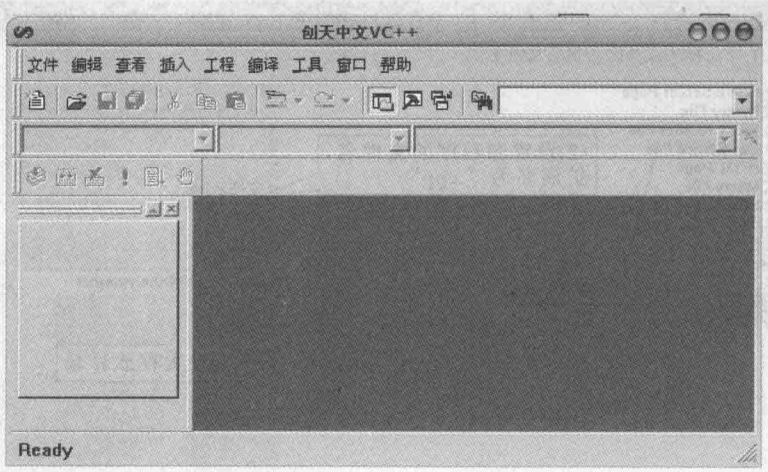

图 1-2 VC++ 6.0 主窗口

在 VC++ 6.0 主窗口的顶部是 VC++ 6.0 的主菜单栏。其中包含 9 个菜单项，即文件、编辑、查看、插入、工程、编译、工具、窗口和帮助。

主窗口的左侧是项目工作区窗口，右侧是程序编辑窗口。工作区窗口用来显示所设定的工作区的信息，程序编辑窗口用来输入和编辑源程序。

2. 输入和编辑源程序

1) 新建一个源程序

在 VC++ 6.0 主窗口中选择“文件”→“新建”命令，如图 1-3 所示。

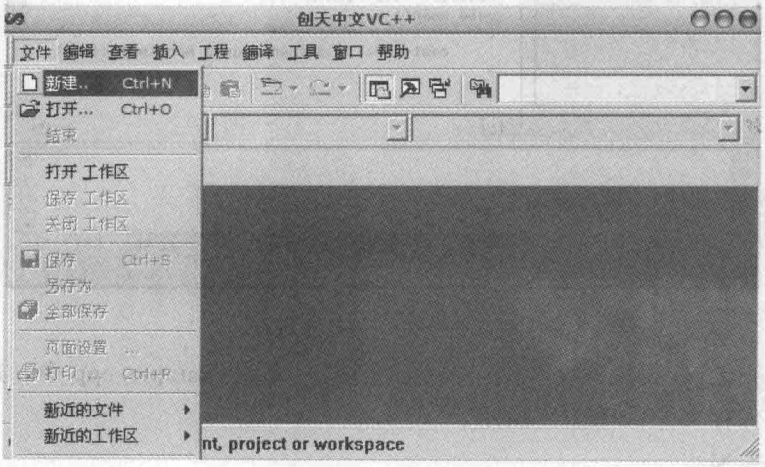

图 1-3 新建源程序文件步骤

在弹出的新建对话框中设置好相应的内容。单击此对话框上方的“文件”标签页，在其

菜单中选择“C++ Source File”项，表示要建立新的C++源程序文件，然后在对话框右半部分的“位置(C)”文本框中输入准备编辑的源程序文件的存储路径(假设为D:\C++_LIANXI)，表示准备编辑的源程序文件将存放在“D:\C++_LIANXI”子目录下。在其上方的“文件名(N)”文本框中输入准备编辑的源程序文件的名字“example”，这样，即将进行输入和编辑的源程序就以example.cpp为文件名存放在D盘的C++_LIANXI目录下。设置完成后，单击“确定”按钮，就进入VC++ 6.0的编辑窗口，如图1-4所示。

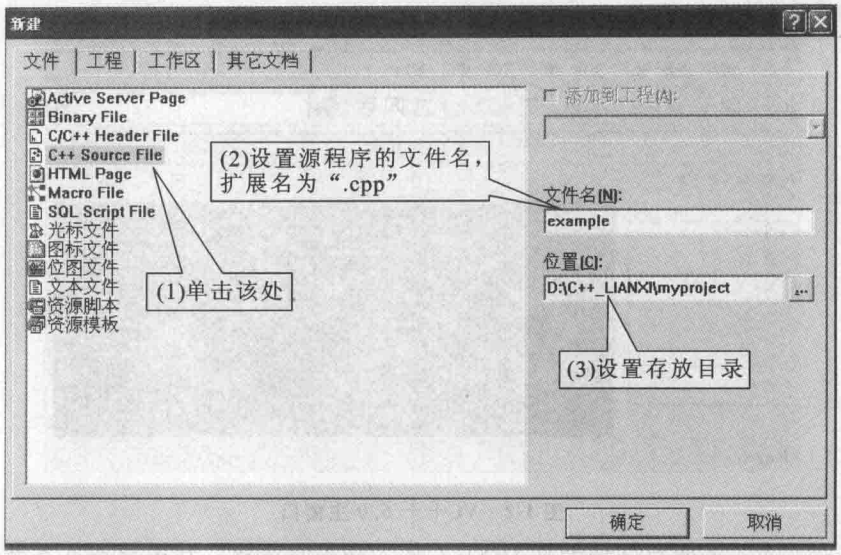

图1-4 新建对话框的设置

在图1-5所示的程序编辑窗口中，可以输入源程序。

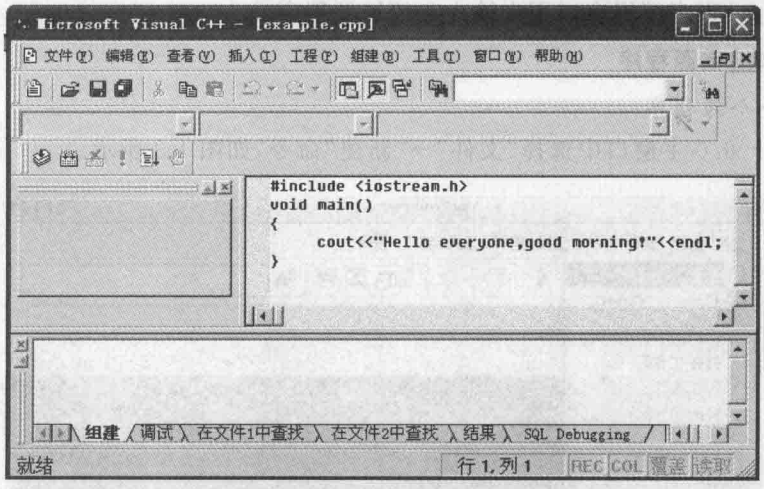

图1-5 输入、编辑源程序

如果经检查无误，则将源程序保存在前面指定的文件example.cpp中。选择“文件”→“保存”命令来保存文件，如图1-6所示。

说明：VC++ 6.0可以编译后缀为.cpp的C++源程序，也可以编译后缀为.c的C源程序。

图 1-6 保存源程序

2) 打开一个已有的程序

如果用户已经编辑并保存了一些 C++源程序,而希望打开其中一个用户所需要的源程序文件,并对它进行修改,具体步骤如下。

(1) 在"我的电脑"中按路径查找已有的 C++程序,如 example.cpp。

(2) 双击此文件,则进入了 VC++ 6.0 集成环境,同时程序也显示在程序编辑窗口中。

(3) 修改后选择"文件"→"保存"命令,将其保存在原来的文件中。

3) 通过已有的程序建立一个新程序

如果用户已经编辑并保存过一些 C++源程序(不论其是否在 VC++ 6.0 集成环境中处理过),则可以通过一个已有的程序来建立一个新程序,这样做比重新输入一个新文件简单得多,还可以省去不少步骤,而且可以利用原有程序中的部分内容。具体方法如下。

(1) 打开任何一个已有的源文件,例如 example.cpp。

(2) 将源文件 example.cpp 中的程序修改后,选择"文件"→"另存为"命令,将它以文件名 example1.cpp 另存在其他路径下,这样就生成了一个新文件 example1.cpp。

用这种新建文件的操作方法很方便,但应注意:另存新文件时,不要选择"文件"→"保存"命令,否则原有文件 example.cpp 的内容将被修改。

在编译新文件前,应先选择"文件"→"关闭工作区"命令,将原有的工作区关闭,以免新文件在原有的工作区进行编译。

3. 程序的编译

在编辑和保存了源文件 example.cpp 以后,则可以对该源文件进行编译。选择"组建""编译[example.cpp]"命令,或者单击工具栏中的"编译"按钮,如图 1-7 所示。

在选中编译命令后,界面中弹出了如图 1-8 所示的对话框。单击"是"按钮,表示同意由系统建立默认的项目工作区,然后开始编译。单击"否"按钮,表示不同意由系统建立默认的项目工作区,将取消编译。也可以使用快捷键 Ctrl+F7 来启动编译。

在进行编译时,编译系统检查源程序中有无语法错误,然后在主窗口下部的调试信息窗口中输出编译的信息。如果无错误,则生成目标文件 example.obj;如果有错误,则会指出错误的位置和性质,提示用户改正错误。用户对源程序进行修改后,再重新编译,直到没有错误为止。编译完成后的界面如图 1-9 所示。

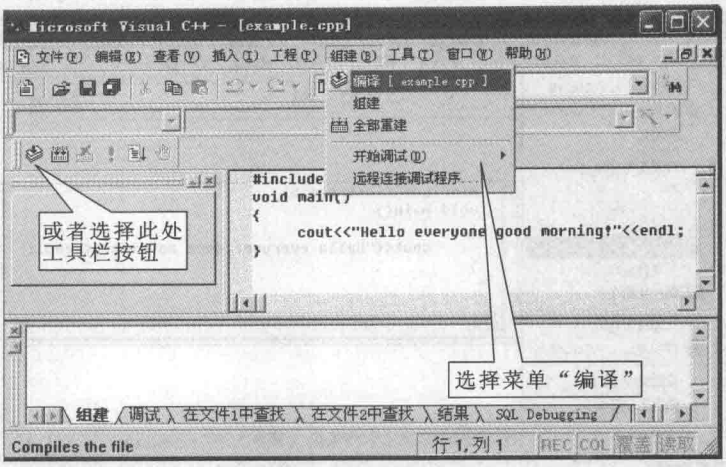

图 1-7 编译源程序

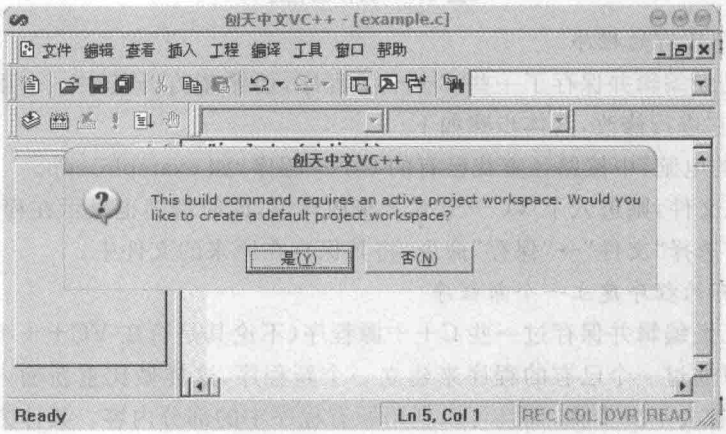

图 1-8 要求创建一个工作区

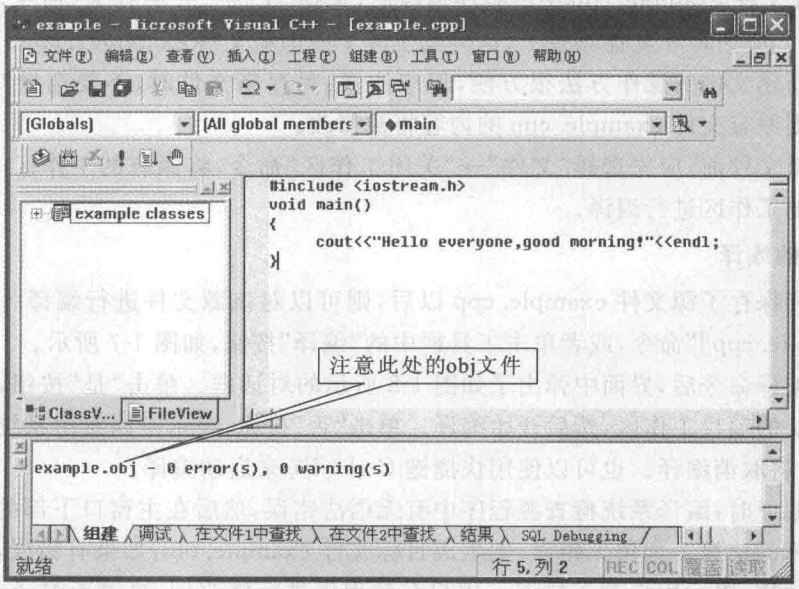

图 1-9 编译成功