程序设计实践教程（C语言）

CHENGXU SHEJI SHIJIAN JIAOCHENG

主　编◎黄同成　黄　磊

湖南人民出版社

程序设计实践教程（C语言）

CHENGXU SHEJI SHIJIAN JIAOCHENG

主　编◎黄同成　黄　磊

副主编◎夏太武　陈　智　成娅辉
周红波　刘金祥

湖南人民出版社

图书在版编目（CIP）数据

程序设计实践教程：C语言 / 黄同成、黄磊主编. —长沙 ： 湖南人民出版社，2011.12

ISBN 978-7-5438-8090-0
I. ①程… II. ①黄… III. ①C语言－程序设计－教材 IV. ①TP312

中国版本图书馆CIP数据核字（2011）第275257号

程序设计实践教程（C语言）

编 著 者　黄同成　黄　磊
责任编辑　赵颖峰　董静静
装帧设计　谌　茜

出版发行　湖南人民出版社［http://www.hnppp.com］
地　　址　长沙市营盘东路3号
邮　　编　410005
经　　销　湖南省新华书店

印　　刷　湖南天闻新华印务邵阳公司
版　　次　2012年1月第1版
　　2012年1月第1次印刷
开　　本　889×1194 1/16
印　　张　9.5
字　　数　150千字
书　　号　ISBN978-7-5438-8090-0
定　　价　22.00元

营销电话：0731-82226732　（如发现印装质量问题请与出版社调换）

内容提要

本书是结合本科生C语言程序设计课程教学大纲和全国计算机等级考试C语言上机考试的内容而编写的，旨在由浅入深、循序渐进地帮助读者提高C语言程序设计能力。

本书是《程序设计基础教程（C语言）》的配套实验教材，分为实践基础篇、实验指导篇、实践提高篇和等级考试指导篇四大块，共6章。实践基础篇介绍C程序开发过程、Visual C++6.0开发环境及C语言程序常见错误的分析与调试，实验指导篇给出了15个基本实验项目，分验证题、排错题、填空题与编程题等题型，实践提高篇以项目训练的方式，给出典型的课程设计实例与参考题目，等级考试指导篇较详细地分析了全国二级计算机等级考试内容，给出了主要知识点和历届相关试题分析等。

本书内容全面，结构合理，实例丰富，实用性强，所提供程序代码均在Visual C++6.0环境下调试通过。是学习C语言的理想实验教材，适合高等院校计算机及其他工科类相关专业的本科生使用，也适合爱好C语言的自学者和程序设计人员使用。

前 言

C语言是从事计算机软硬件开发工作的技术人员必备的语言工具。在学习C语言语法规则知识的基础上，基础知识的综合运用能力与程序设计能力的提高，需要在实践中反复训练才能达到。本书结合了本科生C语言程序设计课程教学大纲和全国计算机等级考试C语言上机考试的内容而编写，在编写中力求通过实验与实践两个环节，分层次循序渐进地帮助读者提高C语言的实际编程能力。

本书是《程序设计基础教程（C语言）》的配套实验教材，分实践基础篇、实验指导篇、实践提高篇和等级考试指导篇四大块，共6章。实践基础篇介绍C程序开发过程、Visual C++开发环境及C语言程序常见错误的分析与调试，实验指导篇给出了15个基本实验项目，分验证题、排错题、填空题与编程题等题型，实践提高篇以项目训练的方式，给出典型的课程设计实例与参考题目，等级考试指导篇较详细地分析了全国二级计算机等级考试内容，给出了主要知识点和历届相关试题分析等。全书内容全面，结构合理，实例丰富，实用性强，所提供程序代码均在Visual C++6.0环境下调试通过。本书有下面四个方面的特色：

⑴强调基础知识的实验，题型种类多。

该书较全面地以C语言语法规则为基础，通过实验环节强化基础知识的运用。每个实验内容通过验证型、排错型、填空型与编程型等多种题型给出。

⑵强化基础知识的综合运用。

在实验环节的基础之上，需要进一步提高基础知识的综合运用能力和分析问题与解决问题的能力，这需要在实践中加以提高。实践提高篇采用项目训练方式，给出了典型的实践实例的分析与设计实现的过程，循序渐进地提高知识综合运用能力与解决实际问题的能力。

⑶强调反复练习。

知识的掌握需要大量反复的练习加以巩固才能获得。本书在等级考试指导篇较详细地分析了全国二级计算机等级考试内容，给出了主要知识点和历届相关试题分析等。

⑷本书适合作为C语言实验指导教材和课程设计指导教材，同时适合报考全国二级计算机等级考试的读者。

本书第1、2章由陈智编写，第3章由黄同成编写，第6章由黄同成与夏太武共同编写，第4、5章由黄同成、周红波、陈智、黄磊、成娅辉、刘金祥、夏太武共同编写。全书的框架设计、统稿、改稿和定稿工作由黄同成教授完成。

在本书的写作过程中，得到了不少专家和任课教师的大力支持，他们为本书的编写提出了许多宝贵的意见和建议，在此表示衷心的感谢。由于编者水平有限，编写时间仓促，书中难免存在许多不足之处，恳请广大读者和同行批评指正。

编者

2011 年12 月

目　录

实践基础篇

实验指导篇

实践提高篇

等级考试指导篇

实践基础篇

第 1 章　C 程序开发过程

1.1 分析问题和建立求解模型

开发一个 C 语言程序的第一步是对待解决的问题进行分析，并运用相关知识为待解决的问题建立求解模型。这里的求解模型和数学模型是不一样的，虽然在很多时候，数学模型可以直接作为问题的求解模型来使用。问题的求解模型可以是数学模型，也可以是求解步骤的序列。

【例 1-1】求两个数字的乘积。

【分析】可以直接利用数学的基础知识将这个问题抽象表示为“A+B”。

一些问题中隐含了另外的问题。对这些问题不仅要分析原问题，而且要分析隐含问题。

【例 1-2】给定一个日期，输出这个日期是该年的第几天。

【分析】为这个问题找到求解模型很容易——只要将每个月的日期数加起来就可以了。如求 2011 年 3 月 15 日是这一年的第几天，只需要用 1 月的天数（31 天）加上 2 月的天数（28 天）再加上 15 天就可以。也可以很容易地想到其中隐含的一个问题，即 “2 月应该有多少天？”或者说“怎样判断闰年？”。

对于“怎样判断闰年？”的问题，其求解模型是闰年定义的描述，即“能被 4 整除但不能被 100 整除，或能被 400 整除的年份”。

还有一些问题很难找到求解模型，此时需要对问题进行一定的变换。为这些问题建立求解模型不仅需要相关的理论知识基础，而且需要一定的设计技巧和编程经验。

通俗地讲，建立问题的求解模型就是对特定的问题，我们怎样分析、处理和解决的过程，也可以认为是手工解决问题的过程。但是求解模型和手工解决问题的过程是不同的，其最大的区别就在于：问题的求解模型必须能够转化为 C 语言程序。

1.2 设计数据存储方式和流程

得到了问题的求解模型之后，还需要从程序设计语言的角度出发，就如何实现展开设计，设计的内容一般包括以下几点。

（1）规划程序的模块；

（2）设计程序中数据的表示和存储方式；

（3）设计程序运行的流程。

对于功能不太复杂的 C 程序而言，一个模块就是一个函数，完成一项独立的任务或实现一个独立的功能。规划模块的依据一般是程序的 I-P-O（Input-Porcess-Output，即输入-处理-输出）过程。基本上每个 C 程序都可以划分成输入、处理、输出和主模块四个模块。

模块划分的另一个思路是从程序需要处理的数据出发，分析数据之间的逻辑关系（一对一、一对多、多对多或者同属于一个集合），并根据数据逻辑关系的特点，确定数据在程序中的表示和存储方式，进而在已有的程序模块的基础上，得到程序运行的流程。《数据结构》为这种分析方式提供了理论和实践的指导，将程序中的数据映射为某种结构（一对一的线性

结构、一对多的树结构、多对多的图结构或者同属于一个集合的散列结构），在已有的各种结构的算法基础上，规划程序模块。这种分析方法有效地利用了已经得到公认的设计结果，模块划分更加合理，设计更加高效。

程序中的数据既包括程序执行所需要的数据，也包括运行过程中生成的数据，除此之外，程序运行需要的参考数据也应该在考虑的范畴。这些数据的表示和存储方式与语言息息相关，设计时需要考虑语言支持的数据类型，以及每种数据类型的取值范围。

对于逻辑复杂的程序，基本数据类型是难以满足要求的，因此需要数组、结构体、公用体、枚举等复合数据结构。

【例 1-3】已知学生信息包括学号、姓名、出生日期、数学、英语、C 语言程序设计的成绩，实现学生信息的输入和输出。

【分析与设计】

按照例题的要求，我们可以按照 I-P-O 的思想将 C 语言程序划分为两个模块，用两个函数实现：Input()函数实现学生信息的输入；Output()函数实现学生信息的输出。

在考虑数据存储的时候，我们认为：学号是一个流水号，用整型表示；姓名用字符串表示。出生日期比较麻烦，因为一个日期包括年、月、日三个整型的值；三门课的成绩都可以用浮点型表示。用多个变量来保存这些数据，会使程序变得冗长，程序代码难以理解。因此，我们考虑使用复合类型保存学生数据。

```
typedef struct
{
    int year;           /* 年 */
    int month;   /* 月 */
    int day;            /* 日 */
}DATE;    /* 日期的结构体 */
typedef struct
{
    int number;         /* 学号 */
    char name[20];      /* 姓名 */
    DATE birthday;      /* 出生日期 */
    float scores[3];    /* 数学、英语、C 语言程序设计的成绩 */
} STUDENT;    /* 学生的结构体 */
STUDENT   stu[100]; /* 多个学生数据的结构体数组 */
```

程序运行的流程一般可以由流程图来描述，使用流程图的好处在于：流程图与程序代码是一一对应的，在设计了合理的流程图之后，程序各个模块的核心代码就已经明确了下来。在编辑源程序时，只要把流程图中每个逻辑框中的文字“翻译”成 C 程序语句，再补充完整 C 程序的语言细节就可以了。

设计程序中数据的存储方式和设计程序运行的流程是有一定联系的，对数据存储的不同会直接影响程序的流程。

【例 1-2 设计】根据分析，可以将程序分成 6 个模块：Input()函数用来实现输入，IsValidDate()函数用来检查用户输入是否合法；CountDays()函数用来统计“第几天”，IsLeapYear()函数用来判断闰年，Output()函数用来实现输出，main()函数提供运行入口。

在设计数据表示和存储时，可以有两种不同的方案：第一种方案使用一个一维数组将非闰年每个月的天数保存下来（假设数组名为 MonthDays[]）；第二种方案不使用额外的存储空间保存天数，只在程序中使用数值常量。不同的数据存储方案，IsValidDate()函数和 CountDays()函数的流程图也不同。

以 CountDays()函数为例，第一种方案的流程图如图 1-1。

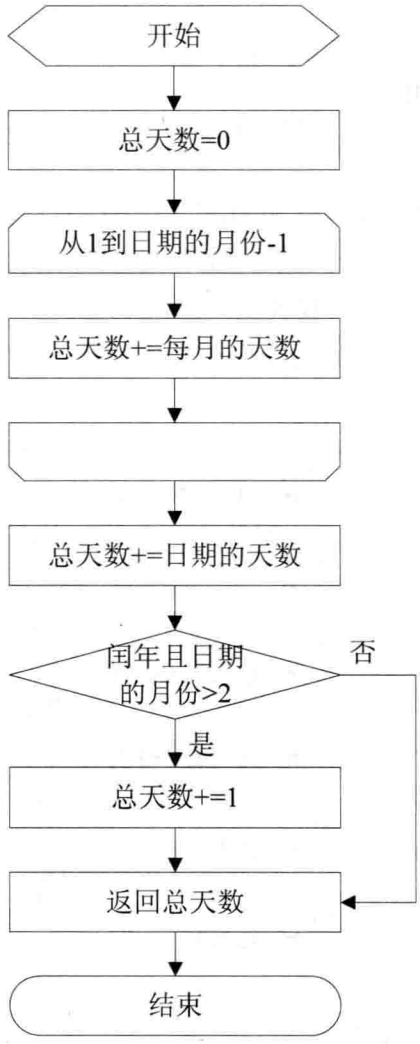

图 1-1 第一种方案的流程图

第二种方案的流程图如图 1-2 所示。

1.3 编辑源程序

编辑是指把源程序的全部或部分输入到计算机，并对其进行修改，最后以文件的形式保存在磁盘存储器上的过程。源程序的编辑操作，是在编辑程序提供的环境下，通过编辑命令来完成的。

源程序编写的依据是设计时生成的流程图，根据流程图描述，将其中每个逻辑框中的文字“翻译”成 C 程序语句，再补充完整 C 程序的语言细节。继续考虑**例 1-2** 的例子。

【例 1-2 部分源程序】

一个日期包括年、月、日三个部分，将这三个部分看作一个整体，需要在 C 源程序中定义结构体。

```
struct date {
    int year;       /* 年 */
    int month;      /* 月 */
```

```
    int day;        /* 日 */
};
```

这样，CountDays()函数的函数原型即为：

```
int CountDays( struct date d)
```

对于两种不同的设计方案，假设第一种方案将每个月的日期存放在一个一维数组MonthDays[]中，即：

```
int MonthDays[13]={0,31,28,31,30,31,30,31,31,30,31,30,31};
```

这里为了使程序更容易理解，每个月的天数从下标为 1 的数组元素开始存放，即 1 月的天数就是 MonthDays[1]，2 月的天数就是 MonthDays[2]，以此类推。另外，增加了MonthDays[12]保存 12 月的天数，虽然这个值永远不会用到。

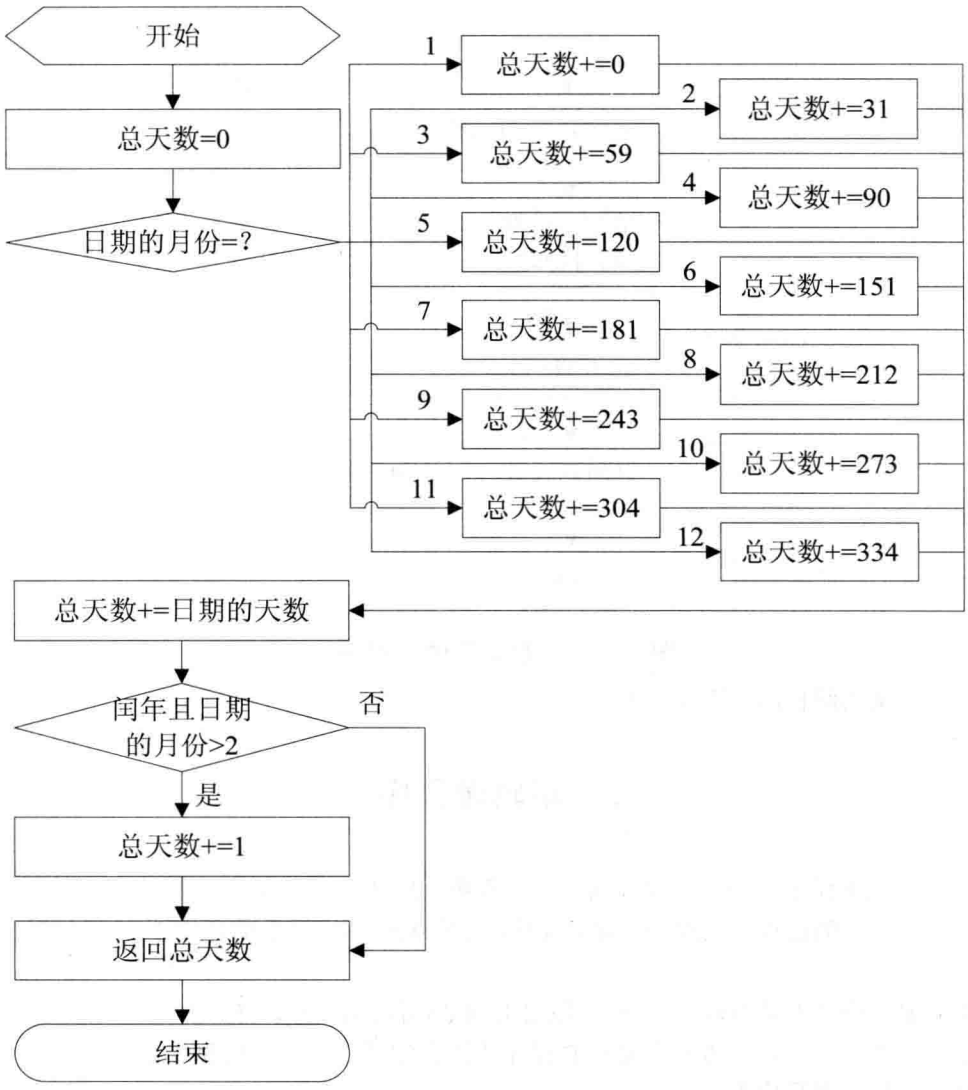

图 1-2 第二种方案的流程图

第二种方案使用数值常量，则流程图中不同的部分对应的源程序为：

```
for (i=1; i<d.month; i++)
    daysSum+=MonthDays[i];
```

和

```
switch (d.month) {
```

```
        case 1: daysSum+=0; break;
        case 2: daysSum+=31; break;
        case 3: daysSum+=59; break;
        case 4: daysSum+=90; break;
        case 5: daysSum+=120; break;
        case 6: daysSum+=151; break;
        case 7: daysSum+=181; break;
        case 8: daysSum+=212; break;
        case 9: daysSum+=243; break;
        case 10: daysSum+=273; break;
        case 11: daysSum+=304; break;
        case 12: daysSum+=334; break;
    };
```

1.4 编译、运行与调试

编辑好源程序并存盘后，应当对源程序进行编译和调试。编译是把源程序文件翻译成相应的目标文件的过程。一个规模较大的程序，一般划分成几个独立的模块分别进行编辑和编译，产生一个个独立的目标模块。这些目标模块是不能够直接执行的。只有将这些目标模块通过连接程序按一定的方式装配起来，才能称为解决问题的可执行程序。

如果在编译、连接过程中未发现错误，系统就会生成目标文件.OBJ 和可执行文件.EXE，这时就可以运行了。运行一个程序，可以采用以下两种方法：

第一种方法是找到.EXE 文件并执行。对于 Visual C++ 6.0 而言，编译生成目标文件和可执行文件保存在工程目录的 Debug 目录下，可以通过双击.EXE 文件或在“命令提示符”中输入命令运行。

第二种方法是在集成开发环境中使用相关菜单、图标或快捷键运行.EXE 文件。如在 Visual C++ 6.0 中，使用菜单“组建|运行”。在编译和运行的过程中，出现错误是在所难免的。程序中错误包括编译错误和逻辑错误两类。编译错误一般包括错误（error）和警告（warning），可以根据集成开发环境的提示加以处理。逻辑错误在运行时发生，此时需要对源程序进行调试。调试常用的方法包括单步执行和设置断点两种，在程序的执行过程中，观察主要变量的值，以此判断错误所在。

第 2 章　Visual C++ 6.0 开发环境介绍

Visual C++ 6.0，简称 VC6.0，是微软推出的一款 C/C++的可视化软件开发工具。Visual C++6.0 不仅是一个 C/C++编译器，而且是一个基于 Windows 操作系统的可视化集成开发环境（integrated development environment，IDE）。Visual C++6.0 由许多组件组成，包括编辑器、调试器以及程序向导 AppWizard、类向导 Class Wizard 等。这些组件通过一个名为 Developer Studio 的组件集成为和谐的整体。

2.1 主要组成

2.1.1 Developer Studio

Developer Studio 是一个集成开发环境，程序设计 99%以上的工作都是在它上面完成的，再加上它的标题赫然写着“Microsoft Visual C++”，所以很多人理所当然的认为，那就是 Visual C++了。其实不然，虽然 Developer Studio 提供了一个很好的编辑器和很多 Wizard，但实际上它没有任何编译和链接程序的功能。

2.1.2 MFC

从理论上来讲，MFC 并不是专用于 Visual C++的。Borland C++，C++Builder 和 Symantec C++同样可以处理 MFC。同时，用 Visual C++编写代码也并不意味着一定要用 MFC，只要愿意，用 Visual C++来编写 SDK 程序，或者使用 STL，ATL，一样没有限制。不过，Visual C++本来就是为 MFC 打造的，Visual C++中的许多特征和语言扩展也是为 MFC 而设计的，所以用 Visual C++而不用 MFC 就等于抛弃了 Visual C++中很大的一部分功能。但是，Visual C++也不等于 MFC。

2.1.3 Platform SDK

Platform SDK 以 Microsoft C/C++编译器为核心，包括了 MASM 和其他一些工具及文档资料。Visual C++的 Developer Studio 没有编译程序的功能，对 C/C++程序的编译是通过 Platform SDK 提供的 CL、NMAKE 和其他许许多多命令行程序实现的，这些平时看不到的程序才是构成 Visual Studio 的基石。

2.2 基本使用

以下以“Hello World！”为例，介绍使用 Visual C++ 6.0 开发一个简单的 C 语言程序。

1．进入 Visual C++ 6.0 集成开发环境

启动并进入 Visual C++集成开发环境至少有 3 种方法：

（1）选择“开始”菜单中的“程序”，然后选择 Microsoft Visual Studio 6.0 级联菜单，再选择 Microsoft Visual C++6.0。

（2）在桌面上创建 Microsoft Visual C++6.0 的快捷方式，直接双击该图标；

（3）如果已经创建了某个 Visual C++工程，双击该工程的 dsw（Develop Studio Workshop）

文件图标，也可进入集成开发环境，并打开该工程。

2．创建一个“控制台应用程序”工程

（1）进入 Visual C++集成开发环境后，选择“文件|新建”菜单，在对话框单击“工程”标签，打开其选项卡，在其左边的列表框中选择“Win32 Console Application”工程类型，在“工程名称”文本框输入工程名，在“位置”文本框输入工程路径，单击“确定”按钮。如图 2-1 所示。

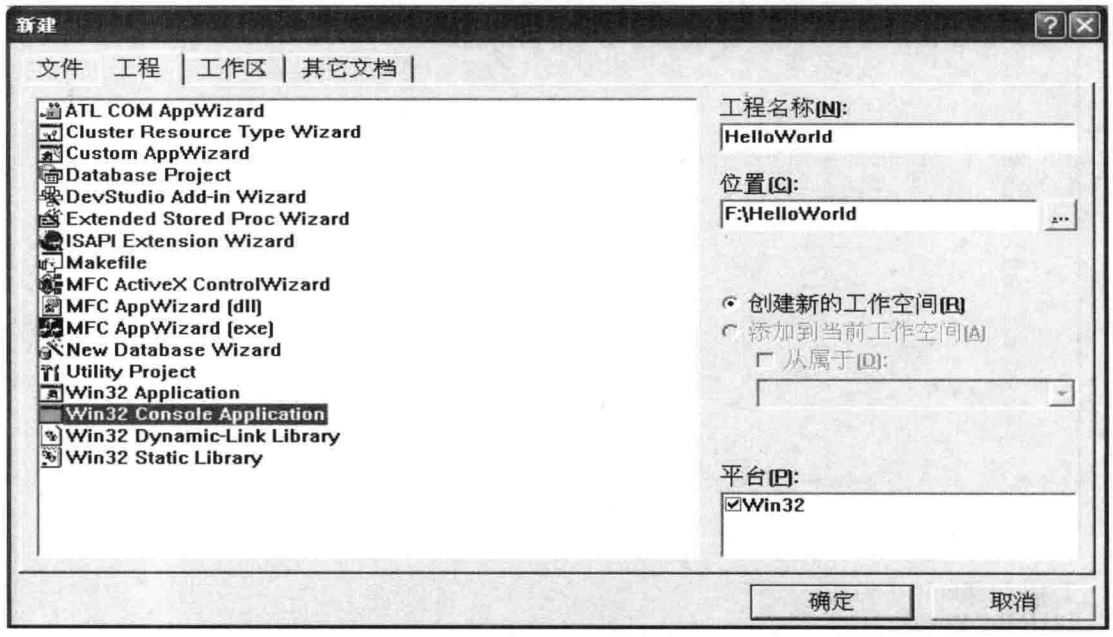

图 2-1 创建工程

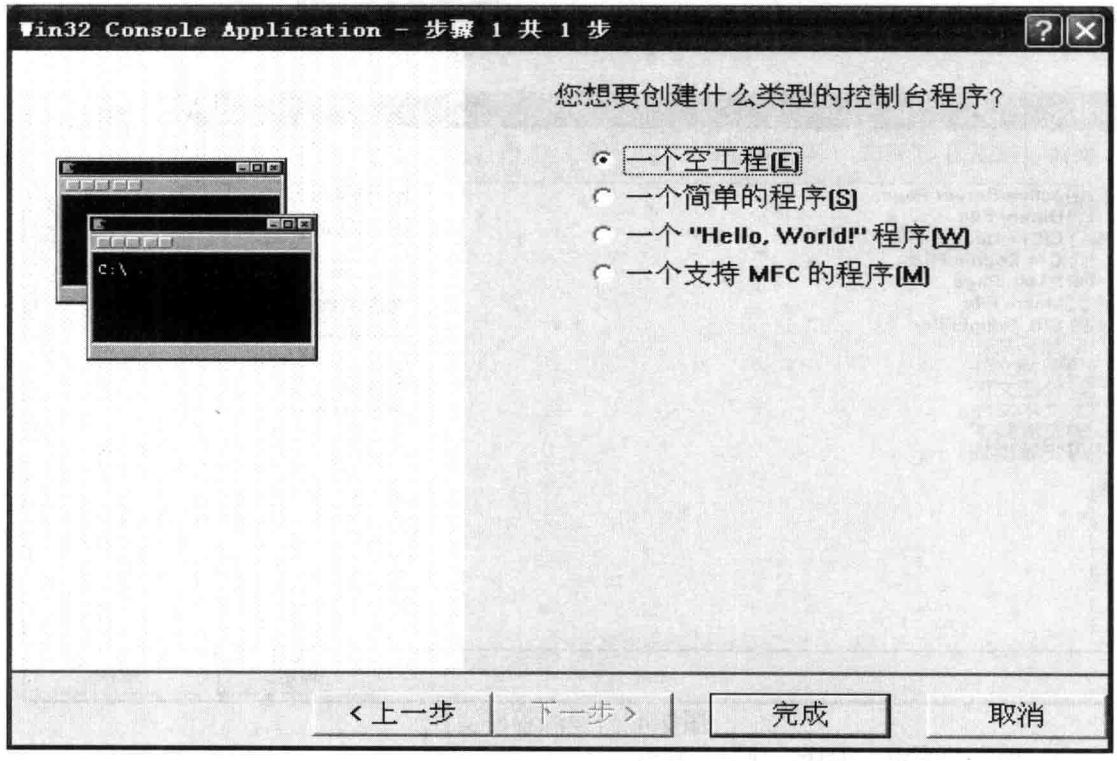

图 2-2 选择工程类型

（2）在弹出的对话框中，选择“一个空工程(E)”，单击“完成” 按钮。如图 2-2 所示。

（3）此时出现“新建工程信息”对话框，此对话框提示用户创建了一个空的控制台应

用程序，并且没有任何文件被添加到新工程中，单击“确定”按钮，工程创建完成。如图 2-3 所示。

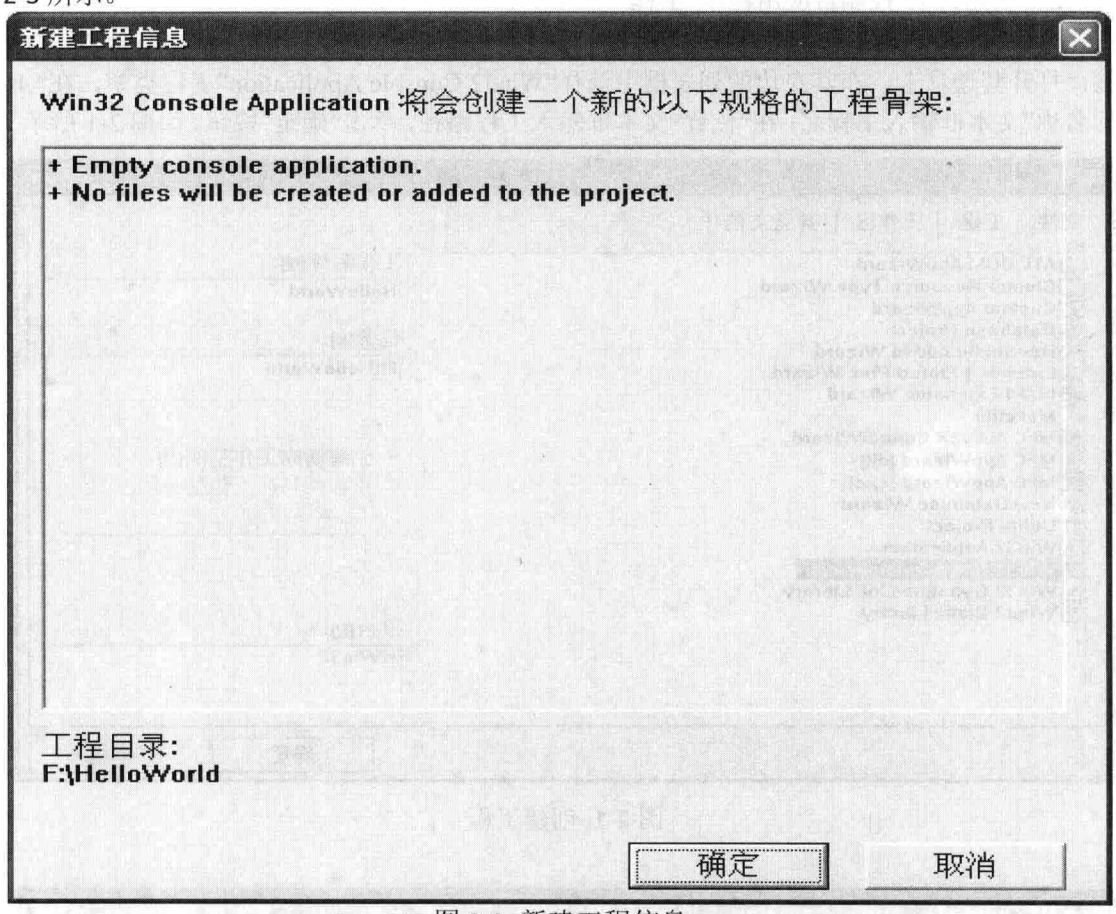

图 2-3 新建工程信息

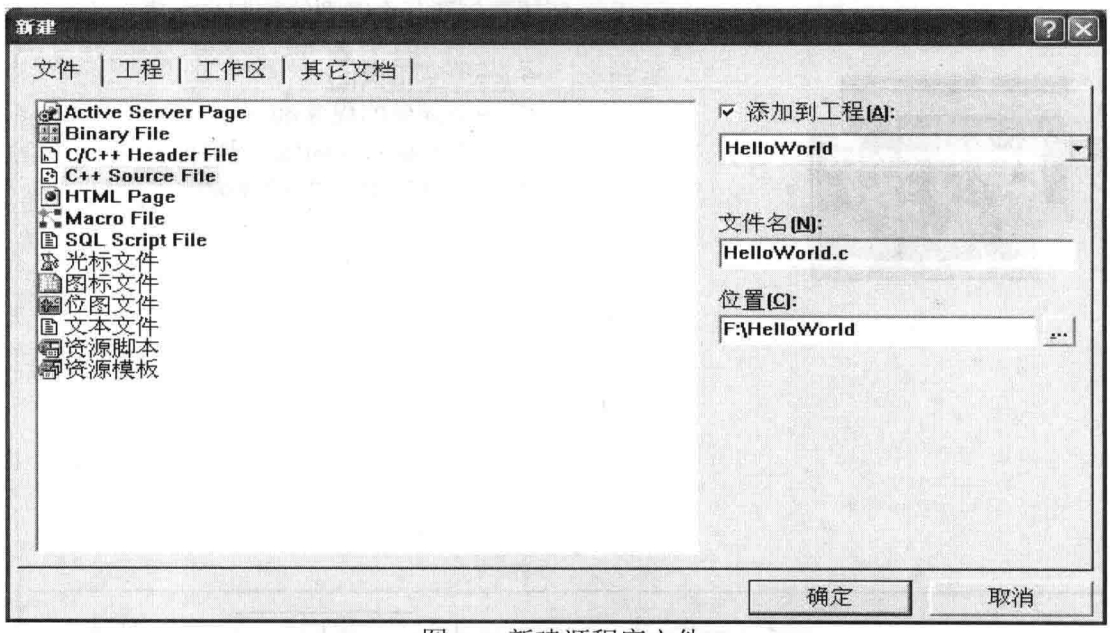

图 2-4 新建源程序文件

3、程序的编辑、编译、建立、执行。

（1）选择“文件|新建”菜单，出现“新建”对话框，打开“文件”选项卡，在列表框中选择 C++ Source File，在“文件名”文本框中输入文件名，选中“添加到工程”复选框，然后单击“确定”按钮，打开源文件编辑窗口。如图 2-4 所示。

对于已经存在的源文件，选择“工程|添加到工程|文件.....”菜单项，在随后打开的插入文件对话框中选择待添加文件，按“确定”将文件添加进工程。

（2）在源文件编辑窗口输入 C 语言程序。如图 2-5 所示。

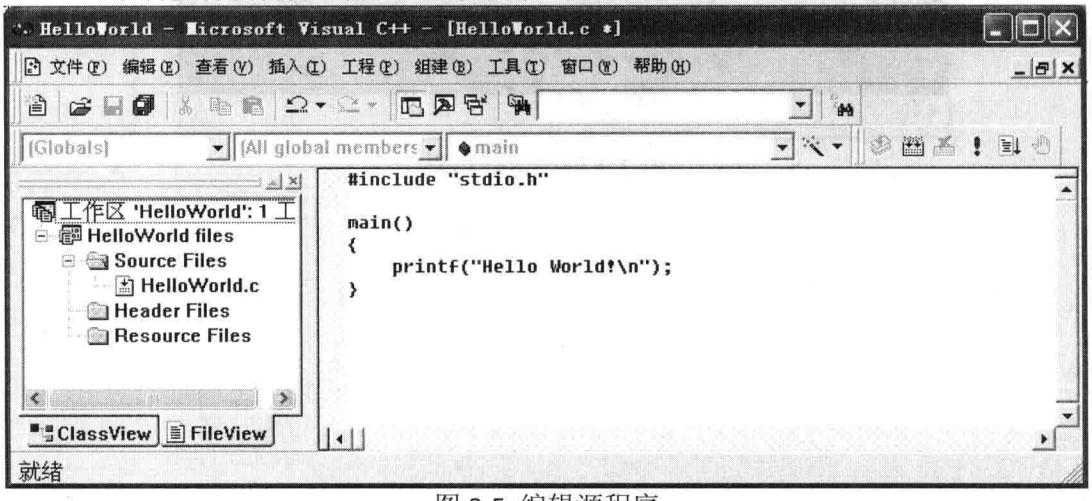

图 2-5 编辑源程序

（3）选择“组建|编译”菜单项，即可编译源文件，系统会在输出窗口显出错误 (Error) 信息以及警告 (Warning)信息。当所有错误改正后，可得到目标文件（.obj）。

（4）选择“组建|组建”菜单项，连接并建立工程的 EXE 文件，得到可执行文件（.exe）。这时编译器可能会给出连接错误(Linking Error)。产生连接错误的原因可能是缺少所需要的库文件或目标文件，或程序中调用的外部函数没有定义等，只要补充相应文档再重新建立即可。如图 2-6 所示。

（5）选择“组建|运行”菜单项，执行工程文件，会出现一个类似 DOS 操作系统的窗口，此时可以进行数据的输入和输出。如图 2-7 所示。

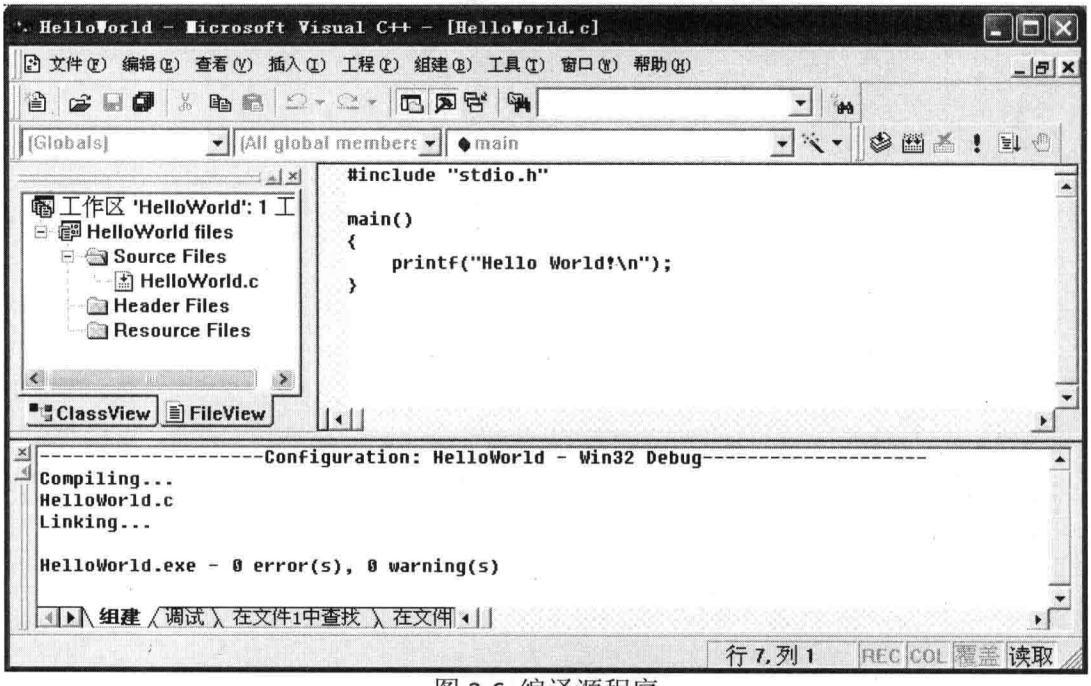

图 2-6 编译源程序