高等院校计算机应用系列教材

程序设计基础

——C 语言

(第 3 版)(微课版)

金　兰　主　编
梁　洁　田新春　副主编

清華大學出版社
北　京

内 容 简 介

C 语言是国内外广泛使用的编程语言，已被大多数高等学校作为典型的计算机教学语言。本书共分10 章，内容包括：C 语言概述，基本数据类型，运算符和表达式、输入输出，控制结构，数组，函数，指针，结构体与共用体，文件，综合应用案例——学生学籍管理系统，以及 4 个附录。

本书内容介绍深入浅出，例题丰富，侧重程序设计思维的构建和程序算法的分析与设计。本书采用“问题提出→问题分析→算法分析→程序实现→说明归纳”的步骤组织教材内容，符合读者的认知规律，强化了算法的分析和设计，有助于帮助读者建立良好的思维模式，培养读者分析问题和解决问题的能力，掌握软件开发的工作原理和系统方法。书中的典型程序一题多解，有助于新旧知识对比学习，融会贯通，启迪思维，拓展读者的程序设计能力和灵活运用能力。

本书可作为高等院校计算机相关专业“程序设计基础”“C 语言程序设计”课程的教材，也可作为程序开发人员的学习用书，还可作为全国计算机等级考试、编程爱好者的参考书。

图书在版编目(CIP)数据

程序设计基础 : C 语言 : 微课版 / 金兰主编.3 版.
北京 : 清华大学出版社, 2025. 2. -- (高等院校计算机应用系列教材).
ISBN 978-7-302-68048-2

Ⅰ. TP312.8

中国国家版本馆 CIP 数据核字第 2025VK5399 号

责任编辑：刘金喜
封面设计：高娟妮
版式设计：恒复文化
责任校对：成凤进
责任印制：曹婉颖

出版发行：清华大学出版社
网　　址：https://www.tup.com.cn，https://www.wqxuetang.com
地　　址：北京清华大学学研大厦 A 座　　**邮　　编：**100084
社 总 机：010-83470000　　**邮　　购：**010-62786544
投稿与读者服务：010-62776969, c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015, zhiliang@tup.tsinghua.edu.cn
印 装 者：三河市天利华印刷装订有限公司
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张：**23.5　　**字　　数：**601 千字
版　　次：2016 年 2 月第 1 版　　2025 年 3 月第 3 版　　**印　　次：**2025 年 3 月第 1 次印刷
定　　价：78.00 元

产品编号：109519-01

前　言

本书为《程序设计基础——C 语言》(ISBN 978-7-302-42444-4)的第 3 版。第 3 版在第 2 版的基础上，删除了第 2 章中数制的内容，修订了书中的部分错误，同时在章节中增加了“练一练”环节，有助于读者在学习的过程中及时消化、吸收和巩固所学知识。第 2 版在第 1 版的基础上，将 C 语言的编译环境从 Visual C++ 6.0 改为 CodeBlocks，修正了书中的差错，同时在章节中增加了二维码，读者可以通过扫描二维码查看对应章节的详细视频讲解，以便进一步学习和掌握书中的内容。

程序设计基础的入门课程——C 语言是目前广泛应用的程序设计语言之一，它具有功能强大、使用灵活、可移植性好的特点，同时兼备低级语言和高级语言的优点，可用于编写系统软件和应用软件。另外，C 语言的语法规则清晰，便于掌握和记忆，因此适合作为大多数人学习计算机程序设计的入门语言。通过本书的学习，可以加深学生对计算机系统的认识，建立良好的计算机思维模式，培养学生模块化、结构化编程方法与技巧，训练学生运用计算机分析问题和解决问题的实践能力，熟练使用 CodeBlocks 开发环境进行 C 语言编程、调试、运行等各个环节的基本操作，为今后进一步学习打下坚实的基础。

本书是作者在多年 C 语言教学、研究和实践积累的基础上，吸收国内外 C 语言程序设计课程的教学理念和方法，依据 C 语言程序设计课程教学大纲的要求编写而成的。

本书每章都配备了大量的例题讲解，所有程序例题均在 CodeBlocks 平台中调试通过。程序例题采用了“问题提出→问题分析→算法分析→程序实现→说明归纳”的步骤讲解，符合读者的认知规律，对例题的重难点位置强化算法的分析和设计，有助于读者建立良好的思维模式，培养读者分析问题和解决问题的能力。本书最后通过一个综合应用案例——学生学籍管理系统，按照软件工程的思想，沿着“需求分析→总体设计→详细设计→编码实现”的软件开发流程，完整地开展系统的分析设计与实现，有助于读者掌握软件开发的工作原理和系统方法。

全书共分为 10 章，具体内容如下。

第 1 章：讲述计算机编程语言的发展过程、在 CodeBlocks 集成开发环境中编写第一个程序的步骤和方法。

第 2、3 章：讲解数据类型、运算符和表达式的使用方法、基本输入输出函数的应用。

第 4 章：讲述运用三种基本的控制结构(顺序、选择和循环)进行编程的方法。

第 5、6 章：讲解数组和字符串的运用、函数的使用、变量的作用域与生存期、编译预处理命令。

第 7、8 章：讲解指针、结构体、共用体的使用方法和链表的相关操作。

第 9 章：讲解文件操作的标准库函数的应用。

第 10 章：完整讲解一个综合应用案例——学生学籍管理系统的分析设计与实现的全过程。

本书中加*的章节为有一定深度和开放性的选学内容，可以有选择性地讲授或留给学生自学。

本书具有以下特色。

1. 实例丰富

本书不仅理论完备，还通过100多个实例夯实基础，以及100多个课堂练一练、课后习题巩固练习，并通过分布在本书第6、8和10章的3个综合应用案例——学生成绩统计程序、学生成绩查询系统、学生学籍管理系统全面提升实战开发能力。

2. 一题多解

典型实例可采用多种算法来设计和实现，有助于新旧知识对比学习，融会贯通，启迪思维，拓展读者的程序设计能力和灵活运用能力。

3. 贴心提示

为了便于读者阅读，书中还穿插了一些说明、注意和思考等小贴士，约定如下。

- “说明”：进一步阐述相关知识点的应用，力求规范、全面。
- “注意”：指出在学习过程中需要特别注意的一些知识点和内容，让读者加深印象。同时，还为读者提供建议及解决问题的方法。
- “思考”：读者可利用课余时间独立思考、解决提出的问题，进一步深入学习训练。

4. 习题丰富

本书每章最后提供了大量习题，涵盖了每章知识的重难点内容，题型灵活多样，包括选择题、填空题、阅读程序填空题及编程题，方便读者课后巩固练习。

本书可作为高等院校计算机相关专业“程序设计基础”“C语言程序设计”课程的教材，也可作为程序开发人员的学习用书，还可作为全国计算机等级考试、编程爱好者的参考书。

本书还特别为任课教师免费提供教学视频资源、电子课件、全部程序源代码和习题参考答案等教学资源，可通过扫描右侧二维码获取。本书还配有相关上机环节指导书《程序设计基础上机指导——C语言》(ISBN 978-7-302-42445-1)，建议与本书配套使用。

教学资源

本书的统稿工作由金兰负责，第1、2、3、4、5、7、9、10章及附录由金兰编写，第6、8章由梁洁编写，田新春老师参与了第2～5章“练一练”习题的编写工作。在本书的编写过程中，武昌首义学院的领导给予了诸多的鼓励和关心。同时，本书的编写工作得到了许多同行的帮助，并参考了大量相关资料，在此深表谢意。因编者水平有限，书中难免会有疏漏和欠妥之处，恳请广大读者给予指正。

服务邮箱：476371891@qq.com。

编　者

2024年7月

目　录

第 1 章

C语言概述

计算机是 20 世纪最重要的发明之一，它深刻地影响了我们的生活，改变了我们的社会。例如，它控制着我们的交通、通信及其他系统。那么，什么是计算机？计算机又是如何工作的呢？

计算机(computer)是基于一串指令进行数据操作的机器，而这串指令被称为程序(program)。比起人的大脑，计算机具有更快的信息处理速度和数值计算能力，而安装在计算机内的程序告诉计算机什么时候获取信息，如何处理信息并实施计算，产生什么样的结果。程序指令由中央处理器(central processing unit，CPU)执行，CPU 通常也被称为微处理器(microprocessor)。目前，常见的台式计算机(desktop computer)、笔记本式计算机(notebook computer)或平板计算机(tablet computer)都是通用计算机，用来为不同的任务运行不同的程序。例如，运行在计算机上的游戏程序能够处理用户的输入并根据程序指令触发动作；运行在计算机上的 Internet 网页浏览器可以根据用户输入的网页地址，经一系列的处理后与运行在该网页地址上的主机的另一个程序进行通信而获得信息，最终将结果显示在网页浏览器上。

嵌入式系统(embedded system)，是为了完成一个或多个特殊功能的专用计算机系统。例如，一个机器人通常含有一个单板嵌入式计算机系统。根据机器人程序，机器人能够做出智能决策并根据一些外部传感信息(如位置、受力、视觉等)采取特定的动作。一台计算机或一个机器人仅服从写入其内的程序命令。

C 语言是用来编写能够与硬件交互的亿万嵌入式系统程序的首选。C 和它的扩展 C++也是编写游戏、网页浏览器、文字处理软件和运行在计算机上的大部分程序的首选语言。本书将教我们如何编写程序，让计算机来帮我们完成想做的事。

1.1 计算机编程语言

计算机编程语言可以分为低级语言和高级语言。低级语言包括机器语言和汇编语言。高级语言比低级语言更容易使用，也更容易移植。下面将介绍各种语言的工作原理。

1.1.1 机器语言

机器语言(machine language)是由二进制编码指令构成的唯一可被计算机直接识别的计算机

语言。每种处理器都有自己专用的机器指令集合。处理器的设计者用不同的二进制编码表示不同的机器指令，每条机器指令只能完成非常低级的处理任务。例如，下列程序是用某种处理器的机器语言编写的，该程序的功能是在屏幕上显示字符串 Hello。

```
11100000   01001000  ;输出字符 H
11100000   01100101  ;输出字符 e
11100000   01101100  ;输出字符 l
11100000   01101100  ;输出字符 l
11100000   01101111  ;输出字符 o
00000000
```

其中，二进制串 11100000 表示屏幕字符输出指令，该指令带一个操作数，表示输出字符的 ASCII 码。二进制串 00000000 表示停止指令，该指令没有操作数。由此可见，一条机器指令是由一个操作码(operator)和 0 到多个操作数(operand)构成的。其中，操作码规定了指令的功能，操作数指明了操作的对象。

尽管用机器语言编写的程序能够被计算机直接理解和执行，但用二进制编码进行编程不仅效率极低，而且所编写的程序含义不直观，难以理解和记忆，错误也难以查找。因此，使用机器语言根本无法高效地编写出高质量的复杂程序，现在已经没有人再用机器语言编写程序了。

1.1.2 汇编语言

为了缓解使用机器语言编程的困难，人们进行了一些改进，即用一些简洁的英文字母、符号串来替代一个特定指令的二进制编码，如用 ADD 代表加法、MOV 代表数据传递等。这样，人们很容易读懂并理解程序的含义，查找和纠正错误也变得更加方便，这种编程语言就是汇编语言(assembly language)。汇编语言为每条机器指令分配了一个助记符号，人们可以使用这些助记符号代替二进制串来编写程序。例如，在屏幕上输出 Hello 的程序，用某种机器的汇编语言可以写为：

```
Write H
Write e
Write l
Write l
Write o
Stop
```

在上述程序中，用 Write 代替了二进制的操作码，用字符代替了其对应的二进制的 ASCII 码。可见，汇编语言是用助记符号表示机器指令的计算机语言。汇编语言指令与机器指令基本上具有一一对应关系。采用汇编语言编程，程序的可理解性、编写效率及质量都有所提高，但是计算机不能直接理解和执行汇编语言程序，必须将其翻译成机器语言，程序才能被机器理解和执行，这个翻译过程称为“汇编”。目前，汇编语言主要用于资源受限的嵌入式系统和对实时性要求非常高的系统编程中。尽管汇编语言程序非常简洁且高效，但它高度依赖于机器，且编写过程烦琐，可读性差，难以进行修改。

1.1.3　高级语言

从最初与计算机交流的困难经历中，人们意识到应该设计一种既接近于数学语言或人的自然语言，又不依赖于计算机硬件，编出的程序能在所有机器上通用的语言，这样的语言被称为高级语言(high-level language)。高级语言与计算机类型无关，在某一机器上完成的程序可以在另一台机器上运行，而且它们易读、易写、易维护。例如，下面是一条用 C 语言书写的在屏幕上输出 Hello 的语句。

```
printf("Hello");
```

由上例可以看出，与汇编语言相比，高级语言将许多相关的机器指令合成单条指令，因此，高级语言指令能完成较复杂的任务。由于屏蔽了与硬件操作有关的细节，编程人员不需要掌握太多的计算机硬件的专业知识，可以集中精力于确定问题求解的算法上，因此，编码相对简单，所编写的程序也更加容易理解和维护。

工程人员和科学研究人员通常使用的高级语言有 FORTRAN、C、C++、Java 和 C#。表 1-1 给出了各种语言的应用领域及其发明者。

表 1-1　各种语言的应用领域及其发明者

语言	应用领域	发明者
FORTRAN	数值和科学计算编程	约翰 • 巴科斯(John W. Backus)
C	系统编程和嵌入式系统	丹尼斯 • 里奇(Dennis M. Ritchie)
C++	面向对象系统编程	本贾尼 • 斯特劳斯特卢普(Bjarne Stroustrup)
Java	网络与系统编程	詹姆斯 • 高斯林(James Gosling)
C#	网络与系统编程	安德斯 • 海尔斯伯格(Anders Hejlsberg)

FORTRAN 是首个通用高级语言，由 IBM 的约翰 • 巴科斯于 20 世纪 50 年代后期开发。它的主要目的是进行公式转换计算。尽管很多人认为它相当落伍，但它至今还是一种进行高性能数值计算的主要有效语言。

C 语言是在 20 世纪 70 年代为了写 UNIX 操作系统及相关应用程序由 AT&T 公司的丹尼斯 • 里奇开发的。之所以命名为 C，是因为它是从早期 B 语言(BCPL 语言的简化版)的基础上发展而来的。布莱恩 • 克尼汉(Brian Kernighan)和丹尼斯 • 里奇 1978 年出版的《C 程序设计语言》在相当长的时间里都是 C 语言的默认标准。1983 年，美国国家标准协会(ANSI)成立了 X3J11 委员会，为 C 制定了标准规范，该标准在 1989 年被认可，通常称为 ANSI C 或 C89 标准。1990 年，ANSI C 标准做了一些小的修改后，被国际标准化组织(ISO)吸纳为国际标准，该版本也被称为 C90，其与 C89 指的是同一语言版本。最先的 C 标准起始于 1991 年，在 1999 年完成，2000 年被认可，该标准称为 C99 标准。

C被广泛用来进行系统编程，如编写操作系统、应用程序、编译器、解释器及函数库。C 语言由于能够像汇编语言一样精确控制机器，所以通常被认为是一种中级语言(mid-level language)，其也是嵌入式系统开发的首选语言，广泛用于开发终端用户的应用程序。

1979 年，贝尔实验室的本贾尼 • 斯特劳斯特卢普为了增加 C 的面向对象编程能力开发了 C++。该语言最初命名为具有类功能的 C(C with classes)，1983 年，它才被命名为 C++(++是 C 和 C++中的增量操作)。面向对象编程是一种用对象及其交互接口设计应用程序的编程方法，它

适用于大规模的软件工程。C++比 C 复杂得多，因此，在没有较好的 C 基础上掌握 C++中的面向对象特性是不太可能的。在最新标准 C99 出现以前，C++一直是 C 的超集。C++不支持 C99 中的可变长数组等特性，然而在其下一个标准中很有可能包含 C99 的这些新特性。

就像 C++一样，很多所谓的现代计算机语言，如 Sun 公司开发的 Java、微软的 C#等都是基于 C 开发的，它们是面向对象的编程语言。其中，Java 能够用来方便地开发具有图形用户界面的网络计算应用。

计算机的 CPU 唯一能够执行的代码是机器码。编译器(compiler)就是一个用来把高级语言程序翻译成低级语言或汇编语言或机器码的计算机程序。源文本称为源代码(source code)或源文件(source file)，而通过编译器输出的代码称为目标码(object code)或目标文件(object file)。一般情况下，编译器将源代码直接翻译成机器码，例如，编译器能够自动将一些源代码格式的头文件(header file)包含到另一个 C 源文件中。链接器(linker)是一种能够将一个或多个由编译器生成的目标文件装配成单个可执行程序的应用程序，其还可以接纳库(library)，即一组编译后的目标文件的集合。图 1-1 给出了 Windows 下用 C 语言开发一个可执行程序的处理过程。首先，利用 Windows 中的文本编辑器，如 Notepad，编辑好文件名为 hello.c 的 C 源代码。该源代码用编译器程序 compile.exe 编译，源文件中的一些头文件在编译过程中会自动包含进来。编译器将源文件编译为目标文件 hello.obj。其次，链接器程序 link.exe 将目标代码和一些库链接生成一个可执行程序 hello.exe 并存放在外存上。如果不运行该文件，则什么都不会发生。该程序可以通过在命令行界面中输入程序名运行，或者通过 GUI 启动，将它所需的实时运行库装载到内存中。程序 hello.exe 的运行过程如图 1-2 所示，根据程序设计的要求，需要用户输入一些数据后，才能输出结果。

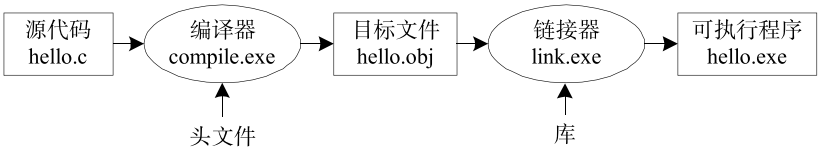

图 1-1　Windows 下用 C 语言开发一个可执行程序的处理过程

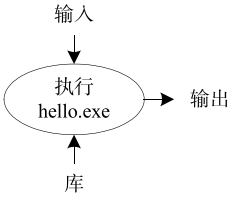

图 1-2　程序 hello.exe 的运行过程

由于几乎任何计算机平台中都有 C 编译器，所以一些高级语言首先可能被翻译成 C 语言，然后通过 C 编译器编译 C 代码。C++早期实现就是利用这种模式，但现在 C++程序已经可以跳过中间代码过程直接编译成机器码。

有些高级语言的源代码被编译成一种称作字节码(bytecode)的中间代码，这种字节码是可以在程序虚拟机(virtual machine)上运行的指令集，Java 就是利用这种运行模式。为了提高执行效率，如今的 Java 也可以直接编译成机器码。

用高级语言编写的程序也可以通过一个解释器直接读取并执行，不需要编译和链接的过程。

1.2 第一个 C 程序

我们从一个简单的程序开始，该程序运行后将在屏幕上输出如下信息。

```
Hello World
```

【程序 1-1】第一个 C 程序如下所示，该程序运行后的输出结果如上。

```
/*  文件：hello.c
程序功能：在屏幕上打印输出信息"Hello World" */
#include <stdio.h>
int main()
{
    printf("Hello World\n");
    return 0;
}
```

在这里，源代码保存在一个名为 hello.c 的文件中，然后由编译器处理。编译器能处理 C 源代码并生成一个可执行程序，运行可执行程序时，就可以产生基于源代码的期望结果。C 程序可以通过一个集成开发环境来编辑执行，这样用户就可以在同一个界面上完成程序的编辑运行。下一节将介绍如何利用集成开发环境编辑运行 C 程序。

下面详细介绍程序 1-1 中每一行的内容和含义。

(1) 用/*符号起始并用*/符号结尾的部分是注释(comment)，用来注解程序，使代码易读易懂。当编译器处理注释时，这些注释内容将被忽略，不产生任何动作。程序 1-1 中的注释行如下。

```
/*  文件: hello.c
程序功能：在屏幕上打印输出信息"Hello World" */
```

注释行注解该程序文件名称为 hello.c，程序的功能是在屏幕上打印输出信息 Hello World。C 程序文件通常都是以.c(称为扩展名)结尾。

注释内容可以跨越多行，但两个注释定界符号不允许嵌套出现。例如：

```
/*注释开始/*嵌套在一起的注释是不正确的*/注释结束*/
```

在 C99 标准中加入了有符号//的注释方式，这种方式下，将符号//后出现的文本视为注释内容，该注释到当前行末为止。例如：

```
printf("Hello World\n");            //这部分是注释
```

(2) 程序 1-1 中的行：

```
#include <stdio.h>
```

是预处理命令。用#开始的行称为预处理命令，其通常被 C 编译器的预处理程序处理。include 命令通知编译器要在该程序中包含文件 stdio.h 的所有内容。通过#include 预处理命令而包含的文件被称为头文件。头文件 stdio.h 包含了与标准输入输出库相关的函数声明等信息。头文件通常以.h 作为其扩展名，如 stdio.h 是标准 C 头文件，通常都是编译器自带的，用户不需要关心这

些标准头文件所包含的内容。它们的实现与编译器相关，不同的编译器提供的标准头文件的内容会有所不同，每一个编译器都会有各自的头文件集。程序 1-1 中之所以要包含头文件 stdio.h，是因为后面要用到标准输出函数 printf()。

(3) 函数(function)是 C 程序中基本的可执行模块。一个 C 程序含有一个或多个函数，其中必须包含的函数是 main()，它将返回一个类型为 int 的值，关于 int 类型的详述请参见第 2 章。下面这一行：

```
int main()
```

是每个 C 程序必须包含的部分。符号 main 后面的圆括号表示它是一个函数。C 程序都是从函数 main()开始执行的。

一组类似的、由 C 语言预先定义、具有特定含义的标识符被称为关键字。附录 A 给出了 C 语言所有完整的关键字列表，符号 int 就是其中的一个。int 用来声明函数 main()的返回值类型为整型，这意味着 main()函数返回值的类型是整数。返回到哪里呢？返回给操作系统。

如果浏览老版本的 C 代码，你将发现程序常以：

```
main()
```

形式开始。C90 标准勉强允许这种形式，但是 C99 标准不允许。

我们还将看到：

```
void main()
```

这种有些编译器允许的形式，但是还没有任何标准考虑接受它。因而，编译器不必接受这种形式，并且许多编译器也不这样做。如果坚持使用标准形式，当把程序从一个编译器移到另一个编译器时也不会有问题。

一个函数通常包含很多语句，函数中的所有语句用一对大括号表示起止，左大括号“{”是函数体的开始，与之匹配的右大括号“}”表示函数定义的结束。这对大括号及其内的所有语句被称为程序块(block)。

程序 1-1 中的函数 main()包含两条语句。其中：

```
printf("Hello World\n");
```

调用函数 printf()输出 Hello World。函数 printf()是输入/输出库中的一个标准函数，可以有多个参数。在本例中函数 printf()的参数是一个字符串，当一个字符串被传给函数 printf()时，通常将出现在程序中的字符串完整地显示在计算机屏幕上。

引号中的字符\n 表示开始新的一行，但并没有输出它们，那么发生了什么事情呢？\n 组合代表一个称为“换行符”的字符，它意味着“在下一行的最左边开始新的一行”，换句话说，打印换行字符的效果与在普通键盘上按 Enter 键一样。换行符是转义字符(escape sequence)的一个例子，转义字符通常用于难以表达的或无法输入的字符，完整的转义字符列表请参考第 2 章。

(4) 程序中的每一条语句必须用分号结尾。为了软件的可读性和可维护性，在函数中的语句要进行恰当的缩进(indent)，缩进时采用固定长度的空格，建议采用程序 1-1 中四个空格的缩进法。与 int 一样，符号 return 也是 C 语言的关键字。

```
return 0;
```

该语句出现在函数 main()的最后，一旦被执行，表示程序成功结束并返回值 0。根据程序

被执行时的方式，该程序返回值将会传递给执行环境。对于 main()函数来说，如果漏掉 return 语句，则大多数编译器将对该疏忽提出警告，但仍将编译该程序。此时，我们可以暂时把 main()中的 return 语句看作保持逻辑连贯性所需的内容，但对于某些操作系统(包括 DOS 和 UNIX)而言，它有实际的用途。

1.3　C 程序的上机步骤

CodeBlocks 是一个开源、免费、跨平台的集成开发环境，在 Windows、Linux 和 Mac 等多个平台中都可以使用。CodeBlocks 没有内置的编译器和调试器，但支持多种编译器和调试器，如 GCC 编译器和 GDB 调试器，常见组合为 Code::Blocks+GCC+GDB。我们选择安装的是捆绑了 MinGW 的 GCC 编译器的版本，选择该版本的好处是无须再去单独安装编译器。对于学习 C 语言来说，CodeBlocks 是完全足够的，下面开始安装。

1.3.1　CodeBlocks 的安装

(1) 登录 CodeBlocks 官方网站下载并安装文件到本机，双击文件 codeblocks-16.01mingw-setup 图标，如图 1-3 所示，运行该文件，打开如图 1-4 所示的安装对话框。

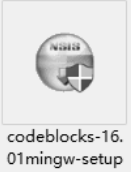

图 1-3　文件 codeblocks-16.01mingw-setup 图标

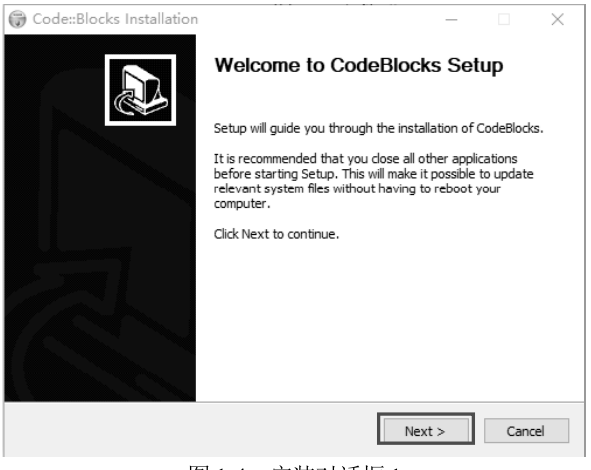

图 1-4　安装对话框 1

(2) 单击图 1-4 安装对话框中的 Next 按钮，弹出如图 1-5 所示的安装对话框。

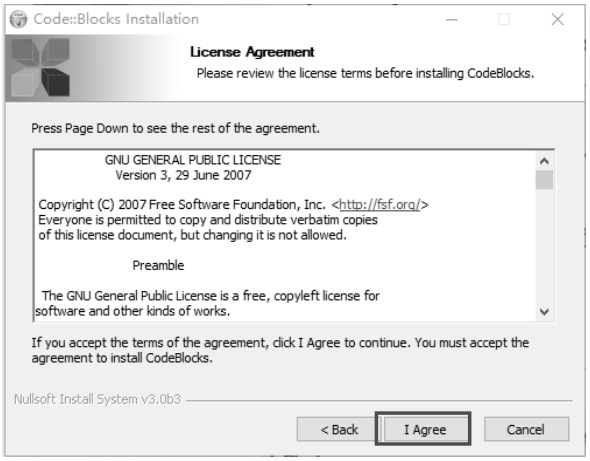

图 1-5　安装对话框 2

(3) 单击图 1-5 安装对话框中的 I Agree 按钮，弹出如图 1-6 所示的安装对话框。

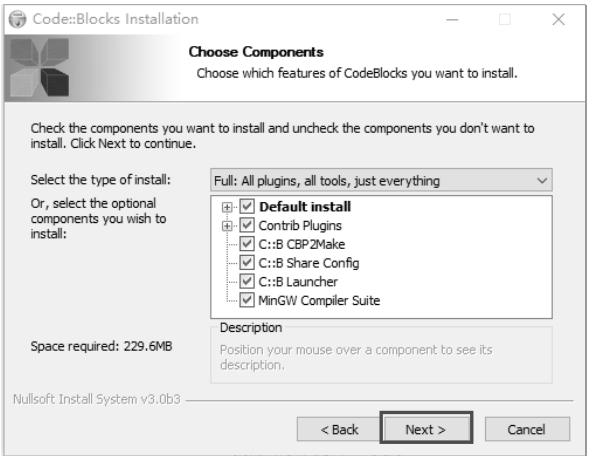

图 1-6　安装对话框 3

(4) 单击图 1-6 安装对话框中的 Next 按钮，弹出如图 1-7 所示的安装对话框。

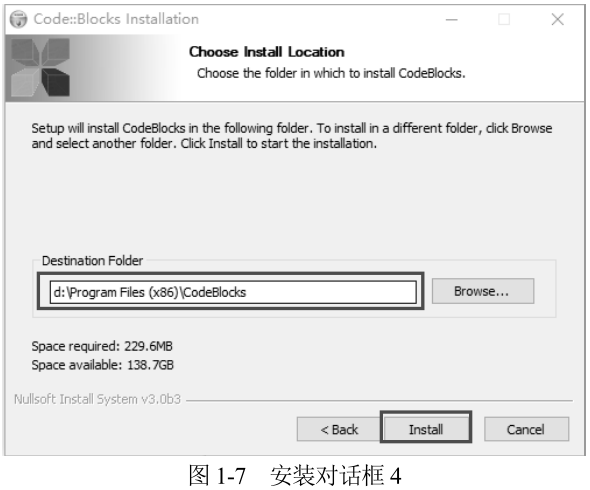

图 1-7　安装对话框 4

(5) 选择安装路径 d:\Program Files(x86)\CodeBlocks，单击 Install 按钮。安装完后，单击图 1-8 安装对话框中的“是”按钮，即可运行 CodeBlocks。

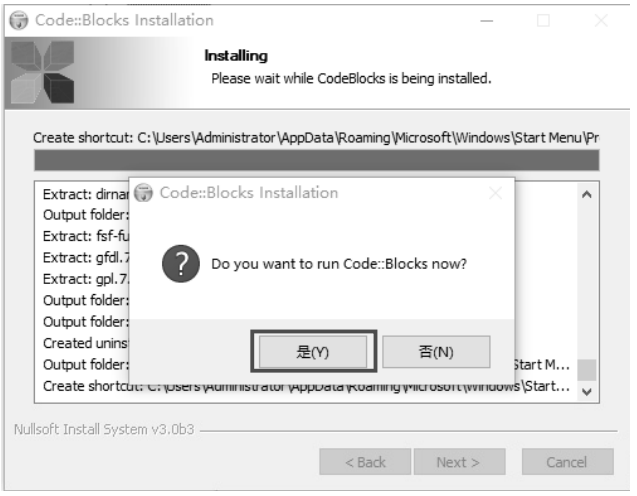

图 1-8　安装对话框 5

(6) 进入如图 1-9 所示的 CodeBlocks 开始界面，安装成功。

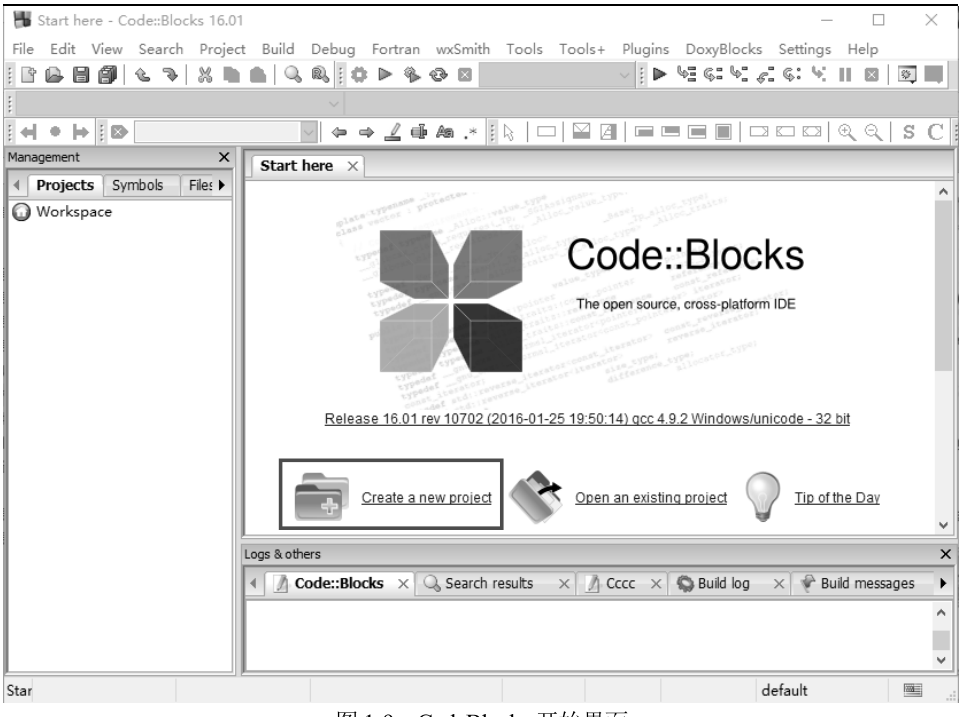

图 1-9　CodeBlocks 开始界面

1.3.2　新建工程

(1) 进入图 1-9 中的开始界面后，选择 Create a new project，或者单击如图 1-10 所示的 File→New→Project... 菜单项。

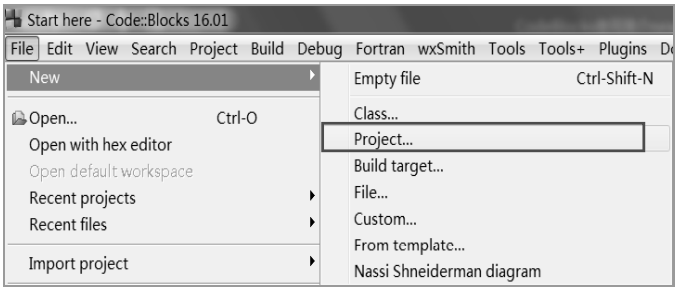

图 1-10　创建新工程

(2) 在弹出的如图 1-11 所示的对话框中选择 Console application 选项，单击 Go 按钮。

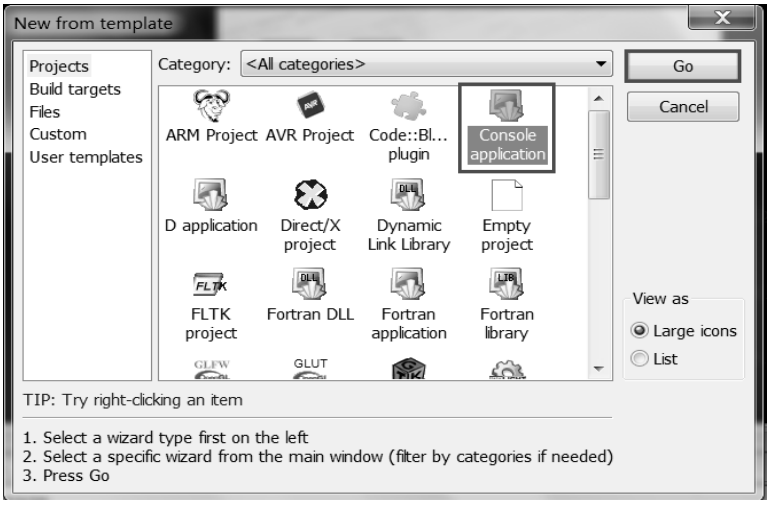

图 1-11　选择 Console application

(3) 在弹出的如图 1-12 所示的对话框中单击 Next 按钮。

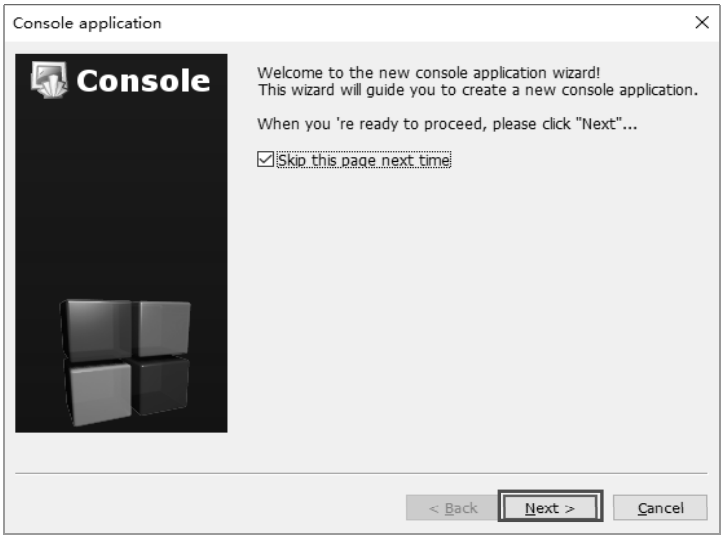

图 1-12　弹出的对话框 1

(4) 在弹出的如图 1-13 所示的对话框中选择 C，单击 Next 按钮。

图 1-13　弹出的对话框 2

(5) 在弹出的如图 1-14 所示的窗口中，一定要先在 Folder to create project in:文本框中选择一个已经在磁盘里建立好的文件夹，如 E:\CProject，然后在 Project title:文本框中输入工程名称，如 prj1，单击 Next 按钮。

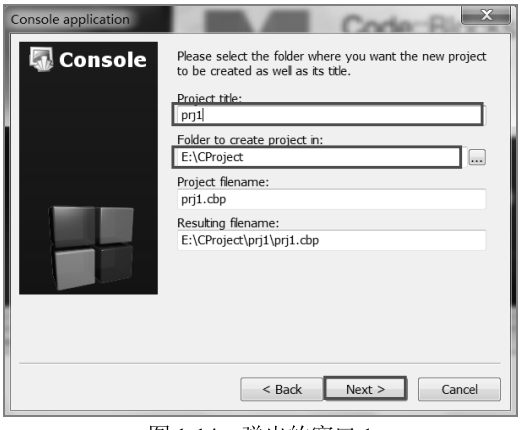

图 1-14　弹出的窗口 1

(6) 在弹出的如图 1-15 所示的窗口中单击 Finish 按钮，即可建立一个工程。

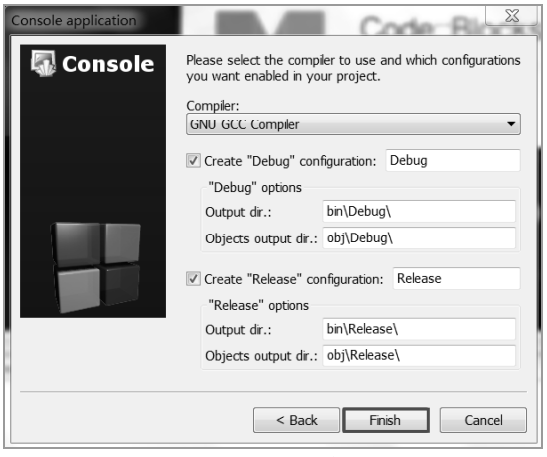

图 1-15　弹出的窗口 2

(7) 在图 1-16 所示的主窗口界面左侧的 Management 窗口中选择 Projects 选项卡，在其中的 Workspace 中可以看到刚建立的工程 prj1，选择 Sources 下的 main.c，双击，在右侧打开程序编辑界面。

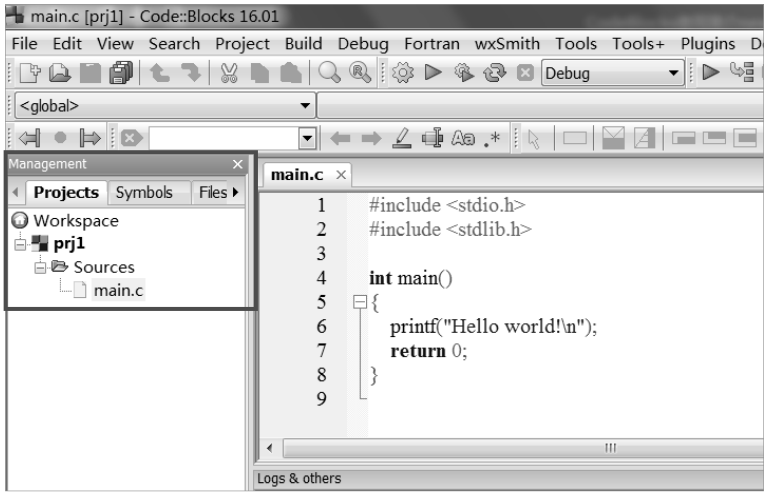

图 1-16　主窗口界面

(8) 选择图 1-17 所示的 Build 菜单栏中的 Build and run 选项，或者按快捷键 F9，即可运行程序。

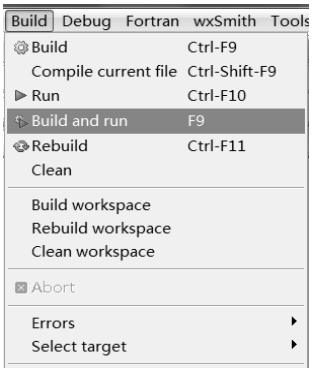

图 1-17　编译菜单栏

(9) 运行结果如图 1-18 所示。请注意，每次修改完程序后，一定要使用这一项才会运行最新的程序。

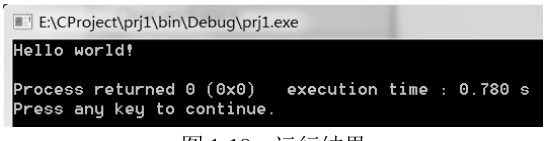

图 1-18　运行结果

(10) 完成程序后，关闭当前工程，如图 1-19 所示，选择 File 菜单中的 Close project 选项，关闭当前工程 prj1。

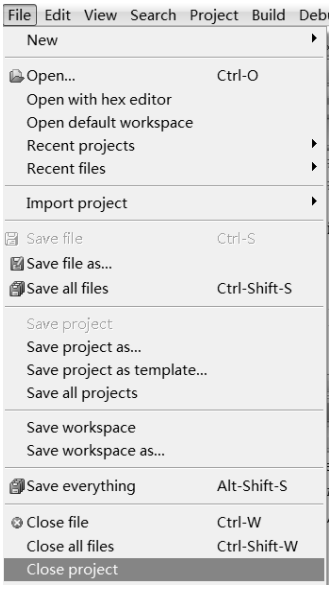

图 1-19　文件菜单栏

(11) 当需要再次打开之前创建的程序时，选择菜单栏中的 File→Open 选项，打开 Open file 对话框，如图 1-20 所示，选择前面创建的工程 E:\CProject\prj1\prj1.cbp 文件，单击“打开”按钮，打开工程 prj1。

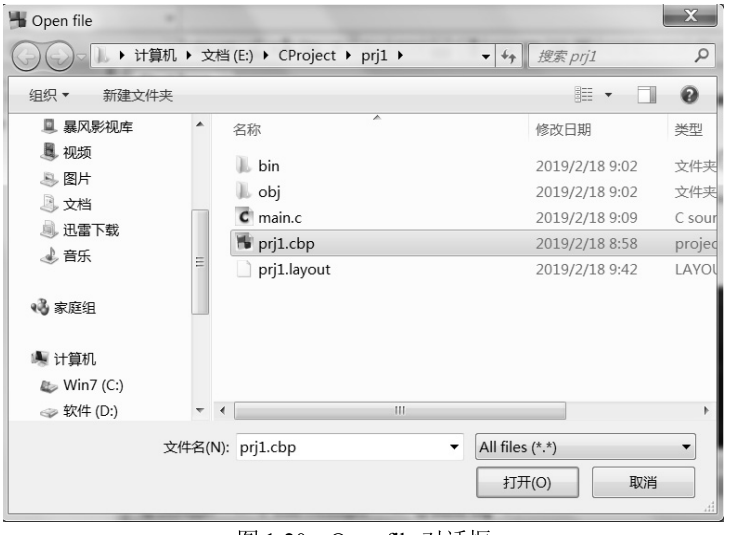

图 1-20　Open file 对话框

1.3.3　多工程切换

虽然每个 C 程序有且仅有一个 main()函数，但 CodeBlocks 中可以通过“新建工程”支持在同一个 CodeBlocks 主窗口中同时创建多个工程。如图 1-21 所示，在主窗口界面左侧的 Management 窗口的 Projects 选项卡中显示了多个工程，其中工程名称加粗显示的(prj2)是当前正在运行的工程，此时如果按快捷键 F9，将会运行此工程。

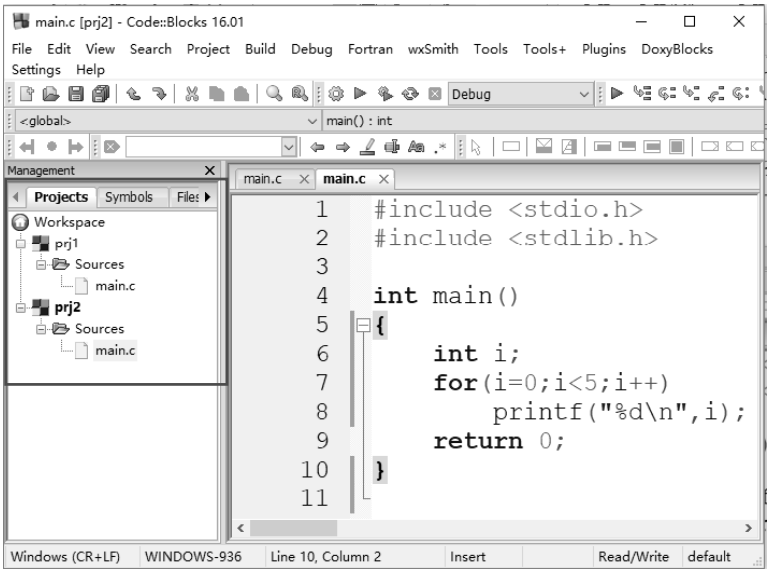

图 1-21　CodeBlocks 中的多个工程

如果想在多个工程中切换当前运行工程，为避免错误可以先选中想要运行的工程(prj1)，然后右击，在弹出的快捷菜单中选中 Activate project 来激活该工程，如图 1-22 所示。

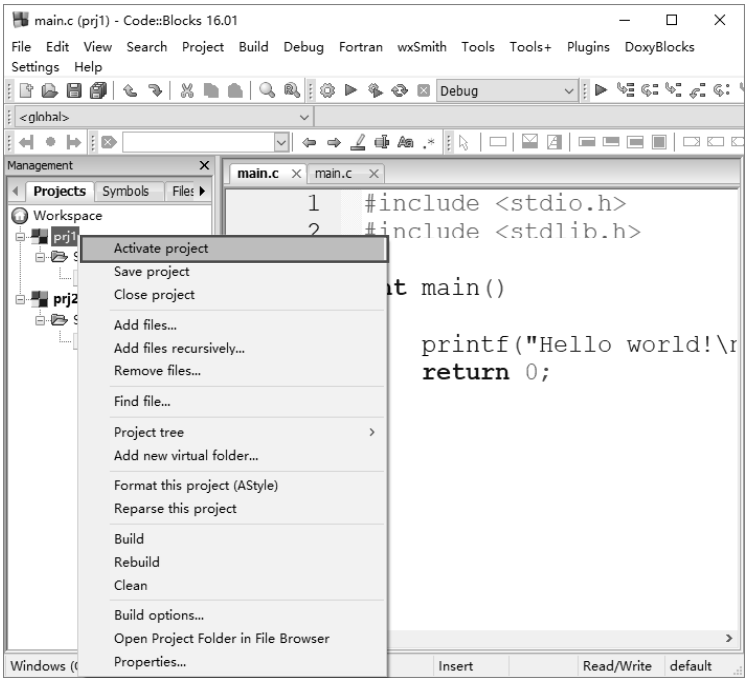

图 1-22　激活工程

1.3.4　单步调试程序

(1) 单步调试时，应先确定程序处于 Debug 状态。选取 Build target 组合框中的 Debug 项目，如图 1-23 所示。

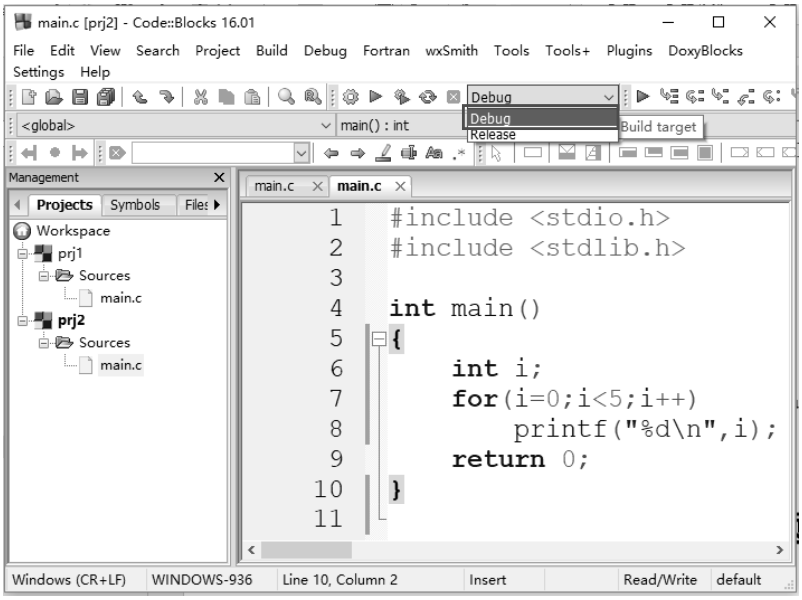

图 1-23　设置 Debug 状态

(2) 设置“断点(Breakpoint)”时，程序在调试状态执行到断点会自动暂停。在需要设置断点的行的左侧单击，会产生一个小红点，说明断点设置成功，如图 1-24 所示。可通过再次单击取消断点，或者按快捷键 F5 来设置或取消断点。

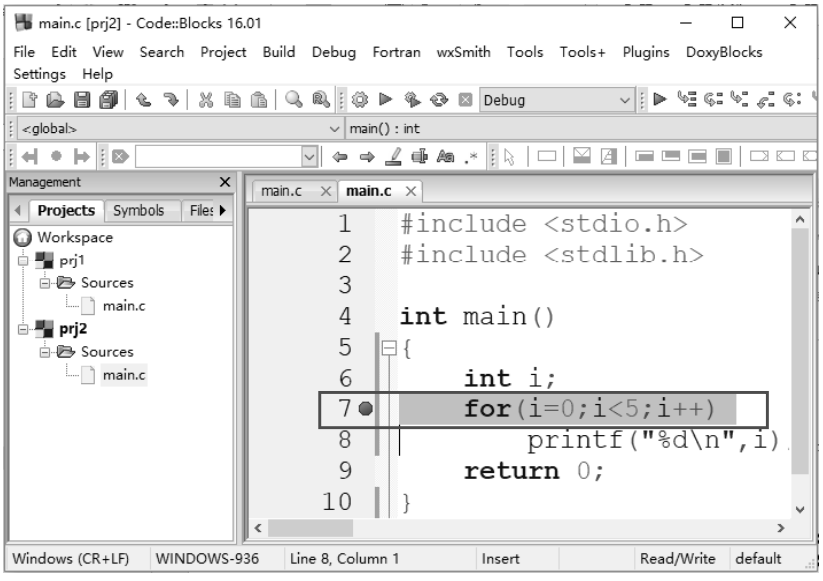

图 1-24　设置断点

(3) 单击快捷工具栏中的红色右三角 Debug / Continue 按钮或按快捷键 F8，开始调试。如图 1-25 所示，当行标识处出现黄色小箭头时，说明程序暂停执行，它所指示的是下一行要执行的代码。

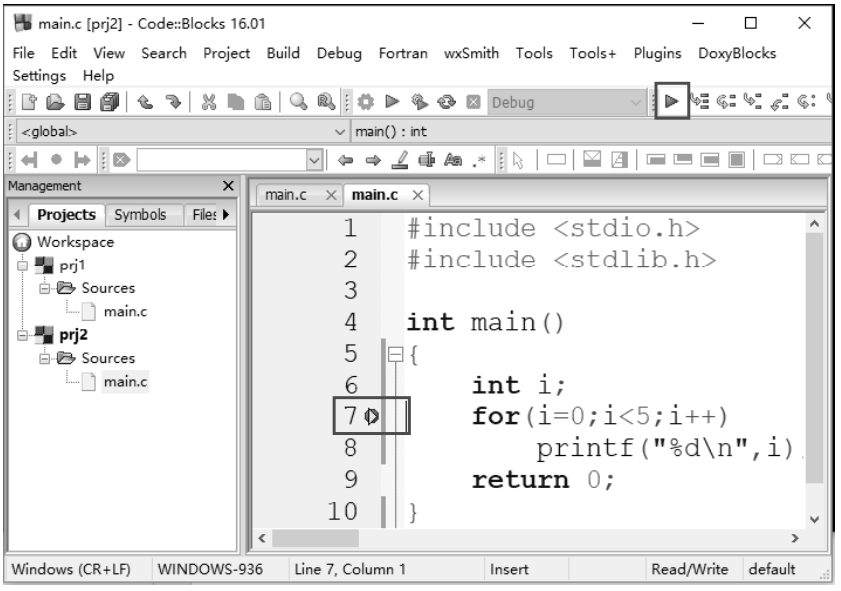

图 1-25　开始调试

(4) 单击 Step into 按钮，开始单步调试，可以看到黄色箭头向下移动，如图 1-26 所示。

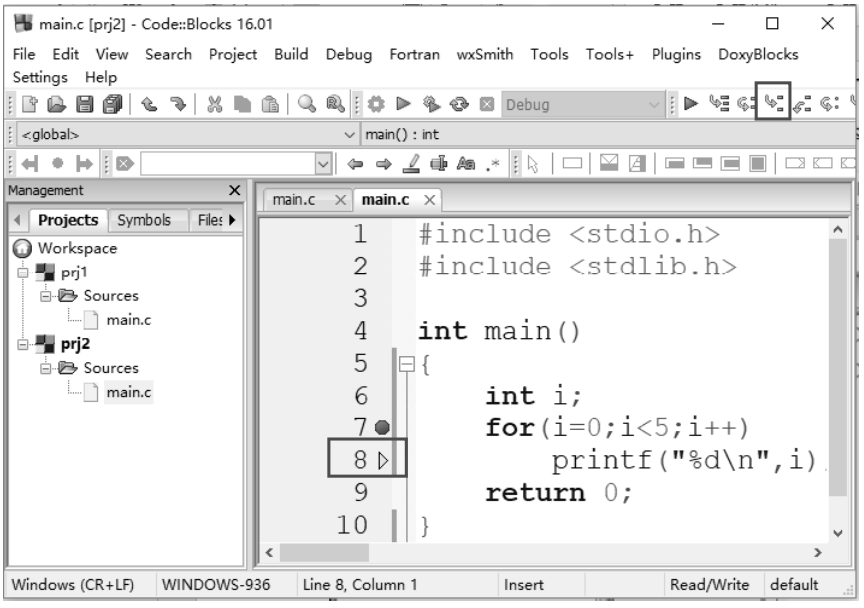

图 1-26　进入单步调试

(5) 调试过程中，可以监视每个变量的变化。如图 1-27 所示，单击快捷工具栏中的 Debugging Windows 按钮，在弹出的菜单中选择 Watches 子菜单。

(6) 左侧 Watches(new)窗口可以看到正在执行的函数内所有局部变量的当前值，如图 1-28 所示。

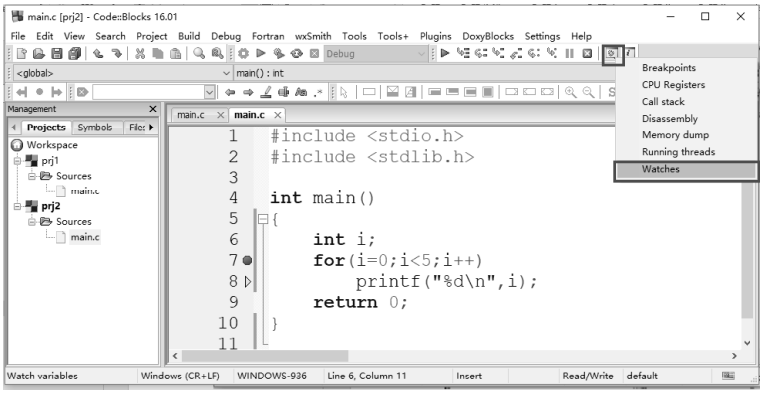

图 1-27　单击“Watches”子菜单

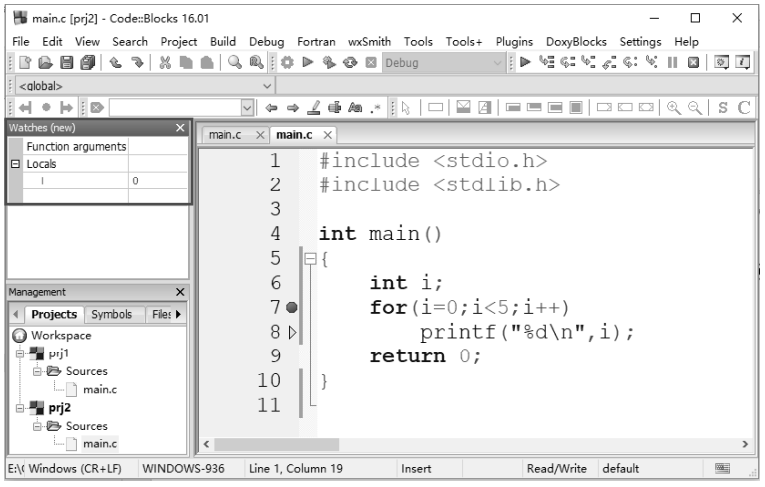

图 1-28　Watches(new)窗口

(7) 单击 Next line 按钮或按快捷键 F7 执行下一行代码，可以看到内存中变量值发生了变化，重复执行该步骤，变量值不断变化，黄色小箭头依次按顺序移动，如图 1-29 所示。

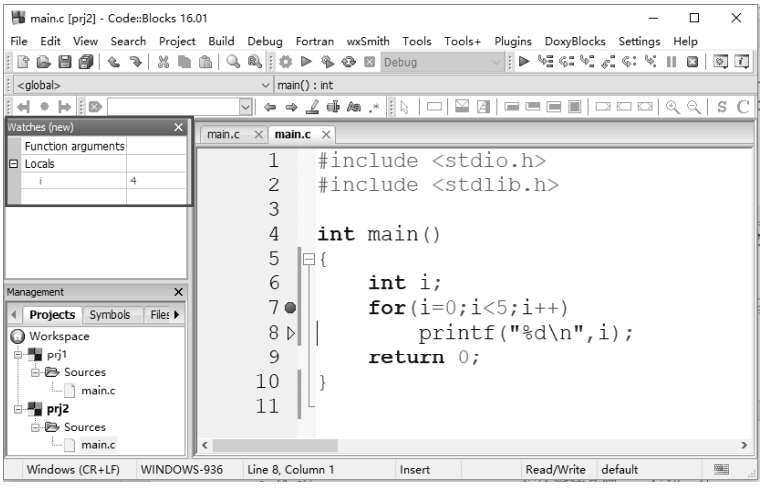

图 1-29　Watches(new)窗口值变化

(8) 如果想看到最后的结果，对于循环结构而言，这样一步一步单击速度太慢，可将光标移动到如图 1-30 所示的第 9 行，然后单击 Run to cursor 按钮或按快捷键 F4，程序会连续运行到光标所在位置再暂停。

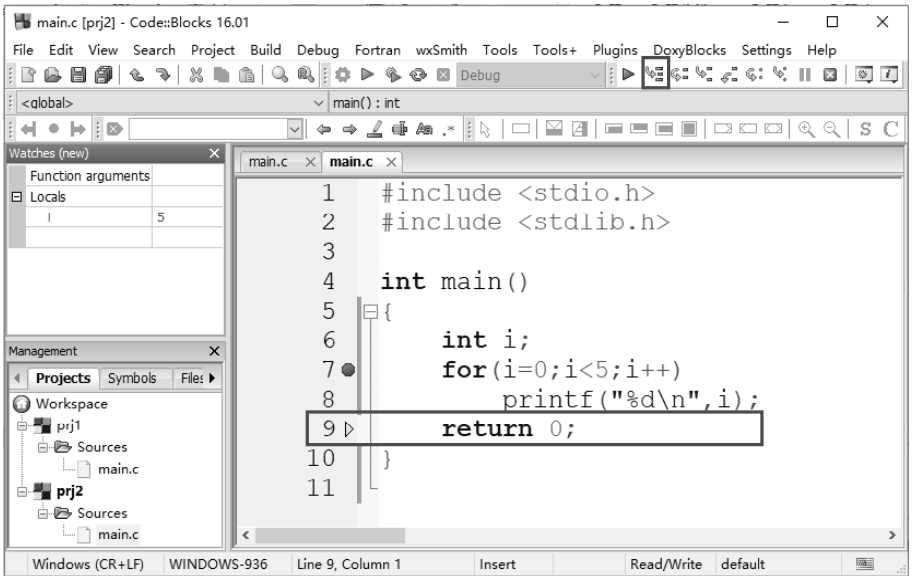

图 1-30　运行到光标

(9) 现在已执行到程序尾 return 0;，可以继续单击 Next line 按钮，直到退出。另外，如果不想让单步再进行下去，则可以单击 Stop debugger 按钮或按快捷键 Shift+F8 来结束单步操作，如图 1-31 所示。

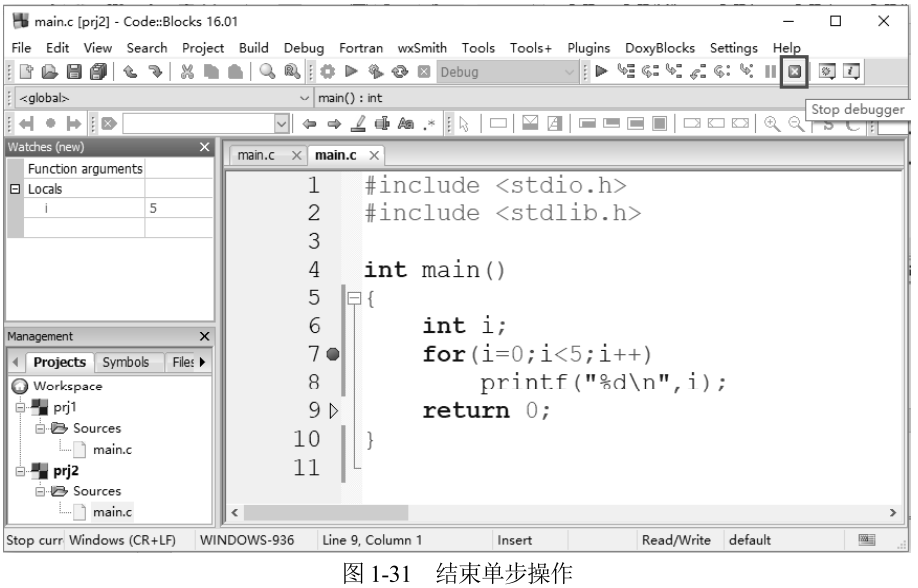

图 1-31　结束单步操作

课后习题 1

一、选择题

1. 一个 C 程序的执行是从(　　)。
   A. 本程序的 main()函数开始，到 main()函数结束
   B. 本程序的第一个函数开始，到本程序文件的最后一个函数结束
   C. 本程序的 main()函数开始，到本程序文件的最后一个函数结束
   D. 本程序的第一个函数开始，到本程序的 main()函数结束
2. 以下叙述不正确的是(　　)。
   A. 一个 C 源程序可由一个或多个函数组成
   B. 一个 C 源程序必须包含一个 main()函数
   C. C 程序的基本组成单位是函数
   D. 在 C 程序中，注释说明只能位于一条语句的后面
3. 以下叙述正确的是(　　)。
   A. C 程序的每行中只能写一条语句
   B. 对一个 C 程序进行编译的过程中，可以发现注释中的拼写错误
   C. C 程序中一行语句以“;”结束
   D. 在 C 程序中，main()函数必须位于程序的最前面

二、编程题

1. 请参照本章例题，编写一个 C 程序，输出以下信息。

```
**************************
*        Hello World!        *
**************************
```

2. 编写一个简单的 C 语言程序，使在屏幕上显示下列信息。

```
    *
   ***
  *****
*******
```