

丛书畅销  
30万册

励志改变人生  
编程改变命运

# 零基础学 C语言

第3版

11.5小时多媒体教学视频

康莉 李宽 等编著

## 本书特色

- ◎ 由浅入深，循序渐进，从零开始学C语言，一点都不难
- ◎ 编程基础、编程进阶、编程应用、项目实战、上机练习、面试指南
- ◎ 230个实例、114个练习题、24个面试题

## 超值、大容量DVD

- ◎ 本书教学视频
- ◎ 本书源代码
- ◎ 本书教学PPT
- ◎ 本书习题答案

## 本书技术支持

- ◎ 论坛: <http://www.rzchina.net>



机械工业出版社  
China Machine Press



---

本书仅提供部分阅读，如需完整版，请联系QQ: 461573687

提供各种书籍pdf下载，如有需要，请联系 QQ: 461573687

PDF制作说明：

本人可以提供各种PDF电子书资料，计算机类，文学，艺术，设计，医学，理学，经济，金融，等等。质量都很清晰，而且每本100%都带书签和目录，方便读者阅读观看，只要您提供给我书的相关信息，一般我都能找到，如果您有需求，请联系我 QQ: 461573687, 或者 QQ: 2404062482。

本人已经帮助了上万人找到了他们需要的PDF，其实网上有很多PDF,大家如果在网上不到的话，可以联系我QQ。因PDF电子书都有版权，请不要随意传播，最近pdf也越来越难做了，希望大家尊重下个人劳动，谢谢！

**备用QQ:2404062482**

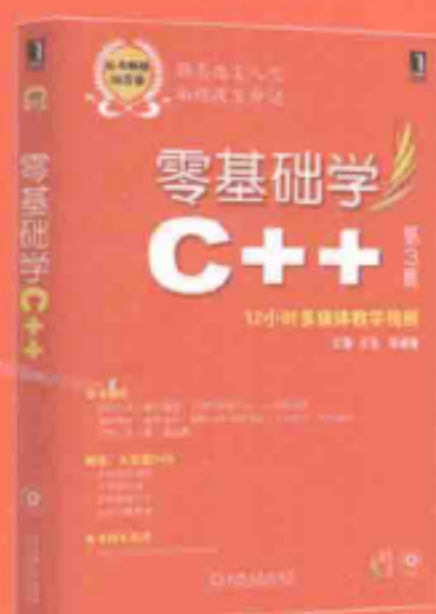




# 励志改变人生 编程改变命运

## 丛书特色

- ① 零门槛学习，没有任何基础便可轻松掌握
- ② 给出编程经验、编程陷阱、编程技巧和编程谬误
- ③ 源代码提供了详细的注释，阅读起来无障碍
- ④ 提供配套的多媒体教学视频，学习效果更好
- ⑤ 每个知识点都对应相应的实例讲解，容易理解
- ⑥ 提供上机实践练习题和常见面试题及解答
- ⑦ 提供典型案例帮助读者提高开发水平
- ⑧ 提供了专门的售后服务论坛：<http://www.rzchina.net>



9 787111 468592 >  
定价：79.00元（附光盘）



9 787111 461081 >  
定价：69.00元（附光盘）



9 787111 463672 >  
定价：69.00元（附光盘）



9 787111 466055 >  
定价：79.00元（附光盘）



9 787111 468721 >  
定价：69.00元（附光盘）



9 787111 467953 >  
定价：69.00元（附光盘）



9 787111 468608 >  
定价：79.00元（附光盘）



9 787111 449447 >  
定价：79.00元（附光盘）



9 787111 468615 >  
定价：79.00元（附光盘）



9 787111 461883 >  
定价：79.00元（附光盘）

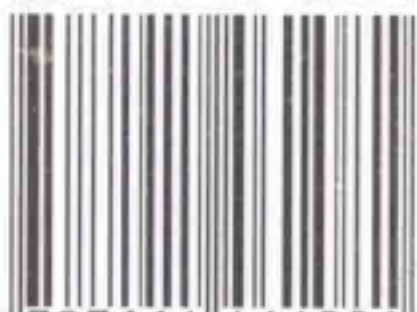


投稿热线：(010) 88379604  
客服热线：(010) 88378991 88361066  
购书热线：(010) 68326294 88379649 68995259

华章网站：[www.hzbook.com](http://www.hzbook.com)  
网上购书：[www.china-pub.com](http://www.china-pub.com)  
数字阅读：[www.hzmedia.com.cn](http://www.hzmedia.com.cn)

上架指导：计算机/C语言

ISBN 978-7-111-46108-1



9 787111 461081 >

定价：69.00元（附光盘）



# 零基础学 C语言

第3版

11.5小时多媒体教学视频

康莉 李宽 等编著



机械工业出版社  
China Machine Press





## 图书在版编目 (CIP) 数据

零基础学 C 语言 / 康莉, 李宽等编著. —3 版. —北京: 机械工业出版社, 2014.5  
(零基础学编程)

ISBN 978-7-111-46108-1

I. 零… II. ①康… ②李… III. C 语言—程序设计 IV. TP312

中国版本图书馆 CIP 数据核字 (2014) 第 047287 号

本书站在零基础学习的角度讲授 C 语言, 使初学者能尽快掌握 C 语言程序设计的精髓, 避免走弯路。在讲解知识点时, 笔者采用由浅入深、逐级递进的学习方式进行内容设置安排。本书一共分为 4 篇, 循序渐进地讲述了 C 语言的语法规则和编程思想, 从基本概念到具体实践、从入门知识到高阶主题、从语法语义到数据结构和算法都进行了详细的阐述。主要内容包括数据的存储和获取、屏幕的输入与输出、运算符、表达式、分支语句、循环语句、函数、数组、指针、字符串处理、结构体、共用体、枚举、位运算、文件处理、作用域、预处理、数据结构等。最后一章通过对一些常见的 C 语言面试题的解析, 为读者参加求职考试提供参考资料。

本书非常适合无 C 语言基础或基础薄弱的程序员阅读, 并可作为开发人员的参考手册。

## 零基础学C语言

康莉 等编著

出版发行: 机械工业出版社 (北京市西城区百万庄大街22号 邮政编码: 100037)

责任编辑: 陈佳媛

责任校对: 董纪丽

印 刷: 北京瑞德印刷有限公司

版 次: 2014年7月第3版第1次印刷

开 本: 185mm×260mm 1/16

印 张: 28

书 号: ISBN 978-7-111-46108-1

定 价: 69.00元 (附光盘)

ISBN 978-7-89405-409-8 (光盘)

凡购本书, 如有缺页、倒页、脱页, 由本社发行部调换

客服热线: (010) 88378991 88361066

投稿热线: (010) 88379604

购书热线: (010) 68326294 88379649 68995259

读者信箱: hzjsj@hzbook.com

版权所有·侵权必究

封底无防伪标均为盗版

本书法律顾问: 北京大成律师事务所 韩光/邹晓东

# 前 言

---

C 语言自 1972 年于贝尔实验室诞生以来，一直以其灵活和实用的特性得到了广大用户的喜爱，迅速发展成一种应用广泛的高级语言。不论是网站后台还是底层操作系统，也不论是多媒体应用还是大型网络游戏，均可使用 C 语言来开发。在工业领域，C 语言也是首选的系统语言。各种操作系统，如 UNIX、Linux 和 Windows 等的内核都是采用 C 语言和汇编语言来编写的。

创新推动着软件开发不断进步，在 C 语言之后，各种新的语言相继诞生，如 C++、Java、C# 等，但 C 语言的基础地位依然不可撼动。学好了 C 语言再去看上面几种语言，会发现其中的机理是相通的，所谓万变不离其宗，改变的只是语法的形式，编程思想却没有变化。而且，很多语言的编译器或者解释器就是用 C 语言编写出来的，比如风靡全球的 PHP、Ruby 等。

所以，C 语言是程序开发的基石。希望本书能像一盏明灯，照亮读者学习 C 语言之路。

## 本书特色

本书系统全面地介绍了 C 语言各个方面的知识，从最简单的“Hello World”程序写起，逐步深化、细化。书中对每个知识和技术要点都给出了翔实的示例及代码分析。和其他书籍中罗列代码的做法不同，本书中的代码力求短小精悍，直击要点，避免了细枝末节对读者思维的干扰。在讲解知识点的同时辅以笔者多年的 C 语言编程经验解析，可加深读者的理解。

本书的特点主要体现在以下几个方面：

- 编排采用密切结合、循序渐进的方式，每章主题鲜明，要点突出，适合初中级读者逐步掌握 C 语言的语法规则和编程思想。
- 示例丰富，关键知识点都辅以示例帮助读者理解。示例程序简洁，但并不是简单的代码罗列，而是采用短小精炼的代码紧扣所讲的技术细节，并配以详细的代码解释和说明，使读者印象深刻，对所学知识理解得更加透彻。
- 示例可移植性强，与编译环境和平台无关，读者可轻易地将代码复制到自己的机器上进行实验，自行实践和演练，直观体会所讲要点，感受 C 语言的无限魅力。本书的所有示例、源代码都附在随书光盘中，方便读者使用。
- 结构清晰，内容全面，几乎涉及了 C 语言的所有特性。
- 图文并茂，帮助读者对知识点建立直观印象。
- 结合笔者多年的 C 语言编程和系统开发经验，特别标注出易出错的技术点或初学者易误解的细节，使读者在学习少走弯路，加快学习进度。



- 很多 C 语言书籍只讲语法规则，不讲数据结构，读者即便掌握了语法理论，也无法写出大型的 C 语言程序。而本书介绍了数据结构和算法的知识，阐述了结构化程序设计思想，探讨了高质量编程的问题，为读者以后深入学习软件开发打下基础。
- 注重加强读者对技术点本质的理解，对诸如“编译器如何为程序实体分配内存”、“函数调用细节”等技术问题做了很多独创性的介绍。

## 本书内容

本书共分为 4 篇，23 章，第一篇从 C 语言的基础知识讲起，使读者初步了解 C 语言语法和编程机制。如果将编写 C 语言程序比作盖房子，那么基础知识就相当于砖瓦水泥。第二篇讲述如何将知识组织起来以构成完整的 C 语言程序。第三篇介绍了进阶内容，讨论一些深层次的技术细节，理解困难、易出错的要点。第四篇介绍了案例实践和面试技巧。

第一篇（第 1 章~第 9 章）C 语言基础。讲述了 C 语言的基础知识，包括 C 语言介绍、C 语言程序开发步骤、不同的开发环境、C 语言程序的组成、变量及数据类型、输入与输出、运算符和表达式、语句、分支、循环等。通过阅读本篇，读者可对 C 语言程序有个初步而全面的认识，了解 C 语言的由来及强大功能，明确开发环境如何通过文本形式的代码生成二进制形式的代码，熟悉 C 语言程序的结构，知道如何声明变量，如何组织语句。学完本篇，读者便可自行书写简单的 C 语言程序。这 9 章的知识是进一步学习的基础。

第二篇（第 10 章~第 15 章）一窥 C 语言门庭。C 语言博大精深，掌握了第一篇中的基础知识可以说只到了大门口。本篇从 C 语言的核心——函数讲起，介绍了与数组、指针、字符串和结构体相关的内容。指针是 C 语言的难点，也是 C 语言灵活性和实用性的直接体现。数组、字符串和结构体也是 C 语言初学者容易感觉头疼的地方。所以说，学完本篇才算迈进了 C 语言的大门。

第三篇（第 16 章~第 21 章）C 语言进阶主题。第二篇从较为独立的角度讲述了函数、数组、指针和结构体的知识，在实际应用中，这些要素彼此交叉，应用组合方式千变万化，这也是 C 语言灵活性的具体体现。本篇用两章的篇幅，分别介绍了指针和函数的技术细节，对初学者来说，理解起来可能略有难度，但这是通往高层次 C 语言学习的必经之路。此外，本篇还介绍了文件处理、编译及预处理、变量的生存期、作用域、可见域及数据结构方面的内容。本篇将使读者对 C 语言有更深入的理解。

第四篇（第 22 章~第 23 章）C 语言程序设计实例与面试题解析。本篇旨在让读者掌握如何用 C 语言开发案例和实践项目。本篇提供了几种常见游戏的开发，帮助读者进一步掌握 C 语言的语法和一些经典算法。最后一章通过一些常见的 C 语言面试题，为读者踏入职场、参加求职考试提供参考资料。

本书由浅入深，由理论到实践，尤其适合初级、中级读者逐步学习和完善自己的知识结构。

## 本书读者对象

本书作为 C 语言的基础教程，适合于以下人士：

- C 语言的初、中级读者
- 了解 C 语言，但所学不全面的人员

- ☐ 高等院校学习 C 语言课程的学生
- ☐ 使用 C 语言进行毕业设计的学生
- ☐ 使用 C 语言进行项目开发的人员
- ☐ 其他相关技术人员

## 本书作者

本书主要由康莉、李宽编写，其他参与编写和资料整理的人员有：冯华君、刘博、刘燕、叶青、张军、张立娟、张艺、彭涛、徐磊、戎伟、朱毅、李佳、李玉涵、杨利润、杨春娇、武鹏、潘中强、王丹、王宁、王西莉、石淑珍、程彩红、邵毅、郑丹丹、郑海平、顾旭光。

编 者

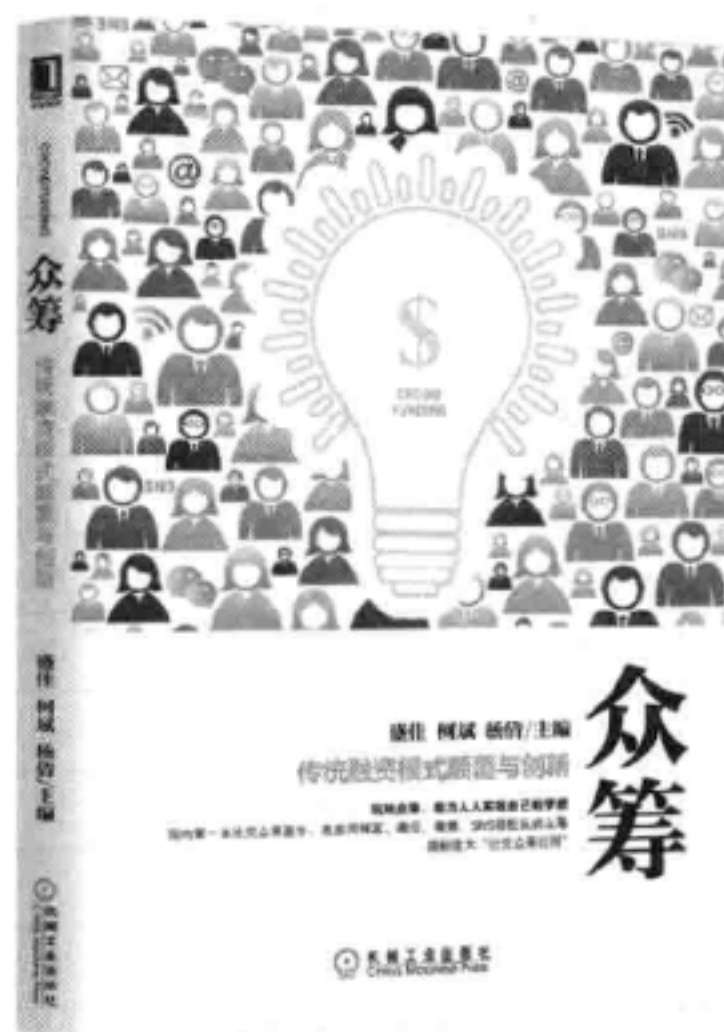
2014 年 2 月



---

## 推荐阅读

---



### 众筹

---

**国内第一本社交众筹著作：教你用博客、微信、微博、SNS轻松玩转众筹，揭秘庞大“社交众筹红利”**

未来属于众筹。十年内，众筹在全球将有3000亿美元的市场规模。

本书站在市场最前沿，回眸众筹历史，描述众筹的当下图景，理性分析众筹模式的革命性，勾勒出在社交网站上玩转众筹的模式，并深入解读中美众筹业不同的发展机遇与监管规则，解密推动众筹成为主流筹资方式的动力所在。

本书适合希望在互联网金融新浪潮中所斩获的读者，是低收入群体、初始创业者、梦想家及中小微企业通过互联网融资方式找到机遇、迅速成长的必备金融服务读本。

# 目 录

## 前言

## 第一篇 C语言基础

第 1 章 踏上征程前的思想动员 .....	1
1.1 为什么选择 C 语言 .....	1
1.2 如何学好 C 语言 .....	3
1.3 语言概述 .....	3
1.3.1 什么是语言 .....	3
1.3.2 什么是机器语言 .....	3
1.3.3 什么是汇编语言 .....	4
1.3.4 面向过程的语言 .....	5
1.3.5 什么是 C 语言 .....	5
1.4 程序的开发周期 .....	5
1.4.1 编辑 C 源代码 .....	6
1.4.2 编译 C 源代码 .....	6
1.4.3 连接目标文件 .....	7
1.4.4 编译连接过程示例 .....	7
1.4.5 运行程序 .....	9
1.5 VC++、C++、C 和 TC 的区别 .....	9
1.6 小结 .....	11
1.7 习题 .....	12
第 2 章 跟我写 Hello World .....	13
2.1 了解需求才能创建程序 .....	13
2.2 认识 LCC-Win32 开发环境 .....	13
2.2.1 为什么选择 LCC-Win32 .....	14
2.2.2 启动 LCC-Win32 .....	14
2.2.3 新建 Hello World 工程 .....	14



2.2.4	定义新工程 .....	15
2.2.5	添加源代码到工程 .....	16
2.2.6	编译器设置 .....	16
2.2.7	连接器设置 .....	16
2.2.8	调试器设置 .....	17
2.2.9	开始编辑代码 .....	17
2.3	编译运行 .....	18
2.4	调试排错 (debug) .....	19
2.4.1	debug 的由来 .....	19
2.4.2	设置断点 .....	20
2.4.3	调试运行 .....	20
2.5	第一个程序容易出现的错误 .....	21
2.5.1	包含命令 include .....	21
2.5.2	关键字 main .....	21
2.5.3	表示代码行结束的分号 .....	22
2.6	小结 .....	22
2.7	习题 .....	22
第 3 章	分解 Hello World——最简单 C 程序的组成 .....	24
3.1	C 程序的构成 .....	24
3.2	C 程序的注释 .....	25
3.3	C 程序必须有的 main 函数 .....	26
3.4	调用函数在屏幕上显示文字 .....	27
3.5	#include 预处理器指示符 .....	28
3.5.1	函数声明及其作用 .....	28
3.5.2	试验寻找#include 的作用 .....	29
3.5.3	#include 的作用 .....	30
3.6	计算 1+1 的程序实例 .....	31
3.7	数据从哪里来, 又到哪里去——变量 .....	32
3.7.1	变量在内存中的表现形式 .....	32
3.7.2	编译器使用变量符号表 .....	32
3.7.3	变量及其使用 .....	33
3.8	自己设计 C 函数 .....	35
3.8.1	在 main 函数中计算 3 个整数的平均数 .....	35
3.8.2	在 main 函数中分 3 次计算 3 个整数的平均数 .....	36
3.8.3	自编函数实现计算 3 个整数的平均数 .....	36
3.8.4	如何自编写函数 .....	37



3.8.5 试验观察总结函数声明和函数定义的意义 .....	38
3.9 语句构成程序 .....	39
3.10 优秀程序员的代码风格 .....	40
3.11 小结 .....	40
3.12 习题 .....	41
<b>第4章 常量、变量及数据类型 .....</b>	<b>42</b>
4.1 计算机是如何表示数据的 .....	42
4.1.1 二进制、八进制、十六进制 .....	42
4.1.2 表示数据的字节和位 .....	44
4.1.3 内存是存储数据的房间 .....	45
4.2 数据类型 .....	46
4.2.1 整数类型 .....	46
4.2.2 整数的有符号和无符号 .....	48
4.2.3 实数类型 .....	50
4.2.4 字符类型 .....	50
4.2.5 数据类型总结 .....	51
4.3 常量 .....	51
4.3.1 直接常量和符号常量 .....	52
4.3.2 符号常量的优点 .....	53
4.3.3 直接常量的书写格式 .....	54
4.3.4 转义字符 .....	55
4.3.5 字符串 .....	57
4.4 变量 .....	58
4.4.1 变量的声明 .....	58
4.4.2 变量声明的意义 .....	59
4.4.3 标识符和关键字 .....	60
4.4.4 变量在内存中占据的空间和变量的值 .....	62
4.4.5 为变量赋初值 .....	65
4.4.6 变量使用时常见的错误 .....	66
4.5 几个与变量相关的经典算法 .....	67
4.5.1 累加和累乘 .....	67
4.5.2 交换两个变量的值 .....	68
4.6 小结 .....	70
4.7 习题 .....	70
<b>第5章 用屏幕和键盘交互——简单的输出和输入 .....</b>	<b>71</b>
5.1 输入—处理—输出：这就是程序 .....	71



5.2 向屏幕输出——printf()函数详解	72
5.2.1 printf()函数的一般形式	72
5.2.2 printf()函数的输出原理	73
5.2.3 格式控制字符串的一般形式	74
5.2.4 输出带符号的整数%d	75
5.2.5 设置最小宽度的输出	75
5.2.6 长整型输出%ld	76
5.2.7 输出八进制形式	76
5.2.8 输出十六进制的形式	77
5.2.9 输出十进制的 unsigned 型数据	77
5.2.10 输出字符	78
5.2.11 输出字符串%s	79
5.2.12 输出实型数据%f	81
5.2.13 输出指数形式的实数	82
5.2.14 自动选择%f 或者%e 形式输出%g	83
5.2.15 printf()函数的几点说明	83
5.3 字符输出 putchar()和字符串输出 puts()	84
5.3.1 字符输出函数	84
5.3.2 字符串输出函数	85
5.4 接收键盘输入——scanf()函数详解	86
5.4.1 scanf()函数的一般形式	86
5.4.2 scanf()函数的输入原理	87
5.4.3 多数据输入分隔规则	89
5.4.4 控制输入的格式字符	90
5.4.5 scanf()函数的使用说明	91
5.4.6 使用 scanf()函数的注意事项	93
5.5 字符输入函数 getchar()	95
5.6 输入和输出程序举例	96
5.7 小结	98
5.8 习题	99
第 6 章 程序的基本构成——运算符和表达式	100
6.1 认识 C 中的运算符和表达式	100
6.1.1 运算符和表达式	100
6.1.2 操作数、运算符和表达式	101
6.1.3 C 运算符简介	102
6.2 算术运算符和算术表达式	102



6.2.1 基本的算术运算符 .....	103
6.2.2 ++自增、--自减运算符 .....	104
6.2.3 算术表达式和运算符的优先级及结合性 .....	106
6.3 逗号运算符和逗号表达式 .....	107
6.3.1 逗号表达式的一般形式 .....	107
6.3.2 逗号表达式的优先级和结合性 .....	108
6.4 关系运算符和关系表达式 .....	109
6.4.1 关系运算符的种类 .....	109
6.4.2 关系表达式的一般形式 .....	109
6.5 逻辑运算符和逻辑表达式 .....	110
6.5.1 逻辑运算符 .....	111
6.5.2 逻辑真值表 .....	111
6.6 赋值运算符和赋值表达式 .....	112
6.6.1 赋值表达式 .....	112
6.6.2 复合运算符 .....	113
6.7 强制类型转换和自动类型转换 .....	113
6.7.1 强制类型转换 .....	113
6.7.2 自动类型转换 .....	115
6.8 运算符的优先级 .....	116
6.8.1 优先级、结合性汇总 .....	116
6.8.2 sizeof 运算 .....	117
6.9 取地址运算符 .....	117
6.10 小结 .....	118
6.11 习题 .....	118
<b>第 7 章 程序的最小独立单元——语句 .....</b>	<b>120</b>
7.1 5 种语句类型 .....	120
7.1.1 表达式语句 .....	120
7.1.2 函数调用语句 .....	122
7.1.3 空语句 .....	122
7.1.4 复合语句（块语句） .....	122
7.1.5 流程控制语句 .....	123
7.2 结构化程序设计 .....	123
7.2.1 什么是算法 .....	123
7.2.2 算法的表示 .....	123
7.2.3 算法的伪代码表示 .....	124
7.2.4 算法的流程图表示 .....	124

7.2.5 3 种控制结构 .....	124
7.2.6 算法示例 .....	125
7.3 小结 .....	125
7.4 习题 .....	126
<b>第 8 章 条件判断——分支结构 .....</b>	<b>127</b>
8.1 if 语句 .....	127
8.1.1 判断表达式 .....	127
8.1.2 花括号和 if 结构体 .....	128
8.2 if...else 结构 .....	129
8.2.1 关键在 else .....	129
8.2.2 if...else 结构体 .....	130
8.3 多分支语句和分支语句嵌套 .....	131
8.3.1 多分支 .....	131
8.3.2 多分支 if 结构 .....	133
8.3.3 分支语句嵌套 .....	134
8.4 switch 结构 .....	135
8.4.1 一般形式 .....	135
8.4.2 为什么叫开关语句 .....	137
8.4.3 default 语句 .....	138
8.4.4 if 结构和 switch 结构之比较 .....	139
8.4.5 switch 结构的常见错误与解决方法 .....	139
8.5 小结 .....	140
8.6 习题 .....	141
<b>第 9 章 一遍又一遍——循环结构 .....</b>	<b>142</b>
9.1 构造循环 .....	142
9.1.1 循环的条件 .....	142
9.1.2 当型循环和直到型循环 .....	143
9.2 while 和 do...while 循环结构 .....	143
9.2.1 语法规则 .....	144
9.2.2 代码块 .....	145
9.2.3 while 语句的常见错误 .....	145
9.2.4 do...while 语句的常见错误 .....	146
9.3 for 循环结构 .....	147
9.3.1 基本形式 .....	147
9.3.2 表达式省略 .....	149
9.3.3 循环终止和步长 .....	150





9.3.4 for 语句的常见错误	150
9.4 循环嵌套	151
9.4.1 嵌套示例	151
9.4.2 嵌套的效率	152
9.4.3 循环嵌套程序的常见错误	153
9.5 与循环密切相关的流程转向控制语句	154
9.5.1 用 break 跳出循环	154
9.5.2 用 continue 重来一次	155
9.5.3 用 goto 实现跳转	156
9.6 小结	157
9.7 习题	157

## 第二篇 一窥C语言门庭

第 10 章 同一类型多个元素的集合——简单数组	159
10.1 什么是数组	159
10.1.1 数组是一大片连续内存空间	159
10.1.2 数组元素的访问	160
10.2 一维数组	160
10.2.1 一维数组的声明	160
10.2.2 一维数组元素的访问	161
10.2.3 一维数组的初始化	162
10.2.4 不合法的数组操作	163
10.3 二维数组	164
10.3.1 二维数组的声明	164
10.3.2 二维数组的初始化	164
10.3.3 二维数组应用举例	165
10.4 更高维的数组	166
10.4.1 高维数组的声明和元素访问	166
10.4.2 初始化	166
10.4.3 多维数组在内存中是如何排列元素的	167
10.5 小结	168
10.6 习题	168
第 11 章 写程序就是写函数——函数入门	170
11.1 什么是函数	170
11.1.1 函数的由来	170
11.1.2 C 语言中的函数	170

11.2 自定义函数 .....	171
11.2.1 定义的语法 .....	171
11.2.2 函数定义范例 .....	172
11.2.3 不要重复定义 .....	173
11.3 函数调用与返回 .....	173
11.3.1 形参和实参 .....	174
11.3.2 传址调用 .....	175
11.3.3 函数返回 .....	176
11.4 告诉编译器有这么一个函数 .....	177
11.4.1 函数声明的语法 .....	177
11.4.2 声明不同于定义 .....	179
11.4.3 标准库函数的声明 .....	180
11.5 函数的调用过程 .....	180
11.6 小结 .....	180
11.7 习题 .....	180
<b>第 12 章 C 语言难点——指针初探 .....</b>	<b>182</b>
12.1 计算机中的内存 .....	182
12.1.1 内存地址 .....	182
12.1.2 内存中保存的内容 .....	183
12.1.3 地址就是指针 .....	183
12.2 指针的定义 .....	183
12.2.1 指针变量的声明 .....	183
12.2.2 指针变量的初始化 .....	184
12.2.3 指针变量的值 .....	185
12.2.4 取地址操作符& .....	185
12.2.5 指针变量占据一定的内存空间 .....	185
12.2.6 指向指针的指针 .....	186
12.2.7 指针变量常见的错误分析与解决 .....	186
12.3 使用指针 .....	187
12.3.1 运算符* .....	188
12.3.2 指针的类型和指针所指向的类型 .....	189
12.3.3 同类型指针的赋值 .....	189
12.3.4 void 指针 .....	190
12.3.5 指针的类型和指针所指向的类型不同 .....	190
12.4 指针的运算 .....	192
12.4.1 算术运算之“指针+整数”或“指针-整数” .....	192



12.4.2 指针-指针 .....	193
12.4.3 指针的大小比较 .....	194
12.5 指针表达式与左值 .....	194
12.5.1 指针与整型 .....	194
12.5.2 指针与左值 .....	195
12.5.3 指针与 const .....	195
12.6 动态内存分配 .....	196
12.6.1 动态分配的好处 .....	196
12.6.2 malloc 与 free 函数 .....	197
12.6.3 calloc 与 free .....	198
12.6.4 free 函数与指针 .....	199
12.6.5 内存泄漏 .....	199
12.6.6 释放已经释放了的内存会出问题 .....	200
12.7 小结 .....	200
12.8 习题 .....	200
第 13 章 字符串及字符串操作 .....	202
13.1 C 风格字符串 .....	202
13.1.1 C 风格字符串的声明 .....	202
13.1.2 C 风格字符串在内存中的表示 .....	202
13.2 字符数组的输入输出 .....	203
13.2.1 字符数组的声明 .....	203
13.2.2 字符数组可以进行整体输入输出 .....	204
13.2.3 使用 gets() 函数读取整行 .....	205
13.2.4 访问字符数组中某个元素 .....	205
13.2.5 使用 puts() 函数实现字符串的输出 .....	206
13.2.6 使用字符数组的常见问题 .....	206
13.3 字符串处理函数 .....	207
13.3.1 理解: 数组名是常指针 .....	208
13.3.2 strlen() 函数与 size_t .....	210
13.3.3 字符串复制函数 strcpy() .....	211
13.3.4 字符串比较函数 strcmp() .....	212
13.3.5 字符串连接函数 strcat() .....	213
13.3.6 全转换为大写形式 .....	213
13.3.7 链式操作 .....	214
13.4 小结 .....	214
13.5 习题 .....	215



第 14 章 结构体、共用体、枚举和 typedef	216
14.1 结构体	216
14.1.1 结构体的定义	216
14.1.2 声明结构体变量	217
14.1.3 初始化结构变量	218
14.1.4 访问结构体成员	218
14.1.5 结构体定义的位置	219
14.1.6 结构体变量赋值	219
14.2 特殊结构体	220
14.2.1 结构体嵌套	220
14.2.2 匿名结构体	223
14.3 共用体	224
14.3.1 什么是共用体	224
14.3.2 共用体的定义	224
14.3.3 声明共用体变量	225
14.3.4 共用体变量的初始化	225
14.3.5 共用体成员访问	225
14.3.6 共用体赋值	226
14.4 结构体和共用体的内存差异	226
14.4.1 结构体变量和共用体变量内存形式的不同	227
14.4.2 结构体变量的内存大小	227
14.4.3 字节对齐	228
14.4.4 最宽基本类型	228
14.4.5 共用体的大小	229
14.5 枚举类型	230
14.5.1 什么是枚举类型	230
14.5.2 枚举类型的定义	230
14.5.3 声明枚举变量	231
14.5.4 枚举常量是什么	231
14.5.5 使用枚举时常见的错误	232
14.6 给类型取个别名——typedef	233
14.6.1 typedef 基本用法	233
14.6.2 #define 用法	233
14.7 小结	234
14.8 习题	234



第 15 章 如何节省内存——位运算 .....	235
15.1 什么是位运算 .....	235
15.1.1 开灯关灯 .....	235
15.1.2 改变状态 .....	235
15.1.3 计算机中的数据存储形式 .....	236
15.2 位逻辑运算符 .....	237
15.2.1 位取反操作 .....	237
15.2.2 位与运算 .....	237
15.2.3 位或运算 .....	237
15.2.4 位异或 .....	238
15.2.5 实例分析 .....	238
15.3 移位运算 .....	239
15.3.1 基本形式 .....	239
15.3.2 移位举例 .....	239
15.3.3 移位和乘以 2 .....	241
15.4 小结 .....	241
15.5 习题 .....	241

### 第三篇 C语言进阶主题

第 16 章 存储不仅仅局限于内存——文件 .....	243
16.1 什么是文件 .....	243
16.1.1 文件 .....	243
16.1.2 流 .....	244
16.1.3 重定向 .....	244
16.1.4 文件的处理形式——缓冲区和非缓冲区 .....	245
16.1.5 文件的存储形式——文本形式和二进制形式 .....	245
16.2 C 语言如何使用文件 .....	245
16.2.1 文件型指针 .....	245
16.2.2 文件操作的步骤 .....	246
16.3 文件的打开与关闭 .....	246
16.3.1 用于打开文件的 fopen()函数 .....	247
16.3.2 打开是否成功 .....	248
16.3.3 用于关闭文件的 fclose()函数 .....	249
16.4 文件的读写 .....	249
16.4.1 读写的相对参照 .....	249
16.4.2 如何判断文件已经结束 .....	249

16.4.3	字符读写函数 fgetc()和 fputc().....	250
16.4.4	字符串读写函数 fgets()和 fputs().....	251
16.4.5	块读写函数 fread()和 fwrite().....	253
16.4.6	格式化文件输入输出 fprintf()与 fscanf().....	256
16.5	文件的定位.....	258
16.5.1	移动指针到文件开头 rewind().....	258
16.5.2	移动指针到当前位置 ftell().....	258
16.5.3	移动指针 fseek().....	258
16.6	小结.....	260
16.7	习题.....	260
第 17 章	灵活却难以理解——指针进阶.....	262
17.1	指针与数组.....	262
17.1.1	数组名指针.....	262
17.1.2	使用数组名常指针表示数组元素.....	263
17.1.3	指向数组元素的指针变量.....	263
17.1.4	指向数组的指针变量.....	264
17.1.5	指针数组.....	265
17.1.6	指针与数组的常见问题.....	266
17.2	指针、结构体和结构体数组.....	266
17.2.1	两种访问形式.....	266
17.2.2	声明创建一个结构数组.....	267
17.2.3	结构数组的初始化.....	267
17.2.4	结构数组的使用.....	268
17.2.5	指向结构数组的指针.....	268
17.3	函数指针.....	269
17.3.1	函数名指针.....	269
17.3.2	指向函数的指针.....	270
17.3.3	函数指针类型.....	272
17.3.4	函数指针作函数参数.....	273
17.3.5	函数指针数组.....	274
17.3.6	指向函数指针的指针.....	275
17.4	小结.....	275
17.5	习题.....	276
第 18 章	更深入的理解——函数进阶.....	277
18.1	参数传递的副本机制.....	277
18.1.1	传值调用的副本.....	277





18.1.2 传址调用的副本机制 .....	278
18.2 函数返回值的副本机制 .....	281
18.2.1 return 局部变量为什么合法 .....	281
18.2.2 返回指针申请动态内存 .....	282
18.2.3 不要返回指向栈内存的指针 .....	282
18.2.4 返回指向只读存储区的指针 .....	283
18.3 函数与结构体 .....	283
18.3.1 结构体变量的传值和传址调用 .....	283
18.3.2 结构体变量的成员作为函数参数 .....	284
18.3.3 返回结构体的函数 .....	285
18.3.4 返回结构体指针的函数 .....	286
18.4 函数与数组 .....	286
18.4.1 数组元素作为函数参数 .....	286
18.4.2 数组名作为函数参数 .....	287
18.4.3 多维数组名作为函数参数 .....	289
18.4.4 数组名作函数参数时的退化 .....	290
18.5 递归 .....	290
18.5.1 递归流程 .....	290
18.5.2 递归两要素 .....	292
18.5.3 效率 VS 可读性 .....	293
18.6 带参数的主函数 .....	294
18.7 小结 .....	294
18.8 习题 .....	295
第 19 章 生存期、作用域与可见域 .....	297
19.1 内存分配 .....	297
19.1.1 内存分区 .....	297
19.1.2 变量的存储类别 .....	298
19.1.3 生存期 .....	299
19.1.4 作用域与可见域 .....	299
19.2 auto 变量 .....	299
19.2.1 定义格式 .....	300
19.2.2 作用域和生存期 .....	300
19.2.3 auto 变量的屏蔽准则 .....	301
19.2.4 重复定义 .....	302
19.2.5 auto 变量的初始化和特点 .....	302
19.3 register 变量 .....	303

19.3.1 定义格式 .....	303
19.3.2 使用举例 .....	303
19.4 extern 变量 .....	304
19.4.1 全局变量定义 .....	304
19.4.2 全局变量声明 .....	305
19.4.3 可见域 .....	306
19.4.4 全局变量的屏蔽准则 .....	307
19.4.5 全局变量的利与弊 .....	309
19.5 static 变量 .....	309
19.5.1 定义格式 .....	309
19.5.2 静态局部变量 .....	310
19.5.3 静态全局变量 .....	311
19.5.4 静态局部变量和静态全局变量的区别 .....	312
19.5.5 extern 变量和 static 变量的初始化 .....	312
19.6 函数的作用域与可见域 .....	312
19.6.1 内部函数 .....	313
19.6.2 外部函数 .....	314
19.7 结构体定义的作用域与可见域 .....	314
19.7.1 定义位置与可见域 .....	314
19.7.2 允许重复定义 .....	315
19.8 常见的有关变量存储的错误 .....	316
19.9 小结 .....	318
19.10 习题 .....	318
<b>第 20 章 编译及预处理 .....</b>	<b>320</b>
20.1 编译流程 .....	320
20.1.1 编辑 .....	321
20.1.2 预处理 .....	321
20.1.3 编译 .....	321
20.1.4 连接 .....	322
20.2 程序错误 .....	322
20.2.1 错误分类 .....	323
20.2.2 编译错误 .....	323
20.2.3 连接错误 .....	323
20.2.4 逻辑错误 .....	323
20.2.5 排错 .....	324
20.3 预处理命令之宏定义 .....	324



20.3.1	宏定义 .....	324
20.3.2	不带参数的宏定义 .....	324
20.3.3	带参数的宏定义 .....	326
20.3.4	#define 定义常量与 const 常量 .....	328
20.3.5	文件包含 .....	329
20.3.6	条件编译 .....	330
20.3.7	宏函数 .....	331
20.4	小结 .....	332
20.5	习题 .....	333
第 21 章	数据结构 .....	334
21.1	链表 .....	334
21.1.1	链表的结构 .....	334
21.1.2	创建链表并遍历输出 .....	335
21.1.3	链表的插入 .....	338
21.1.4	链表结点的删除 .....	341
21.1.5	链表的逆置 .....	342
21.1.6	链表的销毁 .....	344
21.1.7	链表的综合实例 .....	344
21.1.8	循环链表 .....	349
21.1.9	双链表 .....	349
21.2	栈和队列 .....	352
21.2.1	栈的定义 .....	353
21.2.2	栈的分类 .....	353
21.2.3	栈的操作 .....	353
21.2.4	函数与栈 .....	356
21.2.5	队列 .....	356
21.3	自定义类型 .....	359
21.4	小结 .....	361
21.5	习题 .....	361

## 第四篇 C语言程序设计实例与面试题解析

第 22 章	C 语言程序课程设计：游戏 .....	363
22.1	黑白棋 .....	363
22.1.1	程序功能要求 .....	363
22.1.2	输入输出样例 .....	363
22.1.3	程序分析 .....	364



22.1.4	程序初始化 .....	364
22.1.5	初始化图形设备 .....	365
22.1.6	绘制棋盘 .....	365
22.1.7	交替绘制黑白棋 .....	366
22.1.8	游戏（同时判断是否有一方胜利） .....	366
22.1.9	小结 .....	380
22.2	五子棋 .....	380
22.2.1	程序功能要求 .....	380
22.2.2	输入输出样例 .....	380
22.2.3	程序分析 .....	381
22.2.4	主函数程序设计 .....	382
22.2.5	系统初始化 .....	383
22.2.6	移动棋子模块 .....	383
22.2.7	程序胜负判定 .....	387
22.2.8	小结 .....	390
22.3	扫雷游戏 .....	391
22.3.1	程序功能要求 .....	391
22.3.2	输入输出样例 .....	391
22.3.3	程序分析 .....	392
22.3.4	程序设计 .....	393
22.3.5	初始化图形设备 .....	395
22.3.6	事件处理模块 .....	398
22.3.7	游戏处理部分 .....	402
22.3.8	小结 .....	405
22.4	速算 24 .....	405
22.4.1	程序功能要求 .....	406
22.4.2	输入输出样例 .....	406
22.4.3	程序分析 .....	406
22.4.4	程序设计 .....	407
22.4.5	扑克牌处理部分 .....	409
22.4.6	程序运算部分 .....	411
22.4.7	小结 .....	413
第 23 章	面试题解析 .....	414
23.1	基础知识 .....	414
23.1.1	指针自增自减有什么不同 .....	414
23.1.2	什么是递归 .....	414
23.1.3	宏定义与操作符的区别 .....	415



23.1.4	引用与值传递的区别 .....	415
23.1.5	指针和引用有什么区别 .....	415
23.1.6	什么是栈 .....	415
23.1.7	main 函数执行前还会执行什么代码 .....	415
23.1.8	static 有什么用途 .....	415
23.1.9	定义 <code>int **a[3][4]</code> , 则变量占用的内存空间为多少 .....	416
23.1.10	什么是预编译 .....	416
23.1.11	<code>int (*s[10])(int)</code> 表示什么意义 .....	416
23.1.12	结构体与共同体有何区别 .....	416
23.2	算法和思维逻辑知识 .....	417
23.2.1	100 美元哪里去了 .....	417
23.2.2	将 16 升水平均分给四个人 .....	417
23.2.3	算出小王买了几瓶啤酒、几瓶饮料 .....	418
23.2.4	找出不同的苹果 .....	418
23.2.5	找出不同的球 .....	419
23.2.6	猜自己的帽子颜色 .....	419
23.2.7	3 筐水果各是什么 .....	420
23.2.8	最后剩下的是谁 .....	421
23.2.9	聪明的商人 .....	422
23.2.10	红球和白球 .....	422
23.2.11	乌龟赛跑 .....	422
23.2.12	投硬币 .....	423
附录	ASCII 编码表 .....	424

# 第一篇

## C语言基础

---

### 第 1 章 踏上征程前的思想动员

C 语言是目前国内外广泛流行的高级程序设计语言，它是在 20 世纪 70 年代初问世的，是面向过程的较好的结构化程序设计语言。它不仅可以用来编写系统软件，也可以用来编写应用软件，同时也是面向对象程序设计技术的主要工具。C 语言是一门强大而灵活的语言，读者在学习时肯定会遇到很多困难，但恭喜读者选择了本书，因为笔者是十几年前自学 C 语言的，知道学习 C 语言的酸甜苦辣，因此笔者有信心带领读者轻松地学好 C 语言、用好 C 语言。

本章的主要知识点如下：

- ☐ 为什么选择 C 语言
- ☐ 如何学习 C 语言
- ☐ 机器语言进化史
- ☐ 认识 C 语言的代码
- ☐ 学习 C 语言的开发环境

#### 1.1 为什么选择 C 语言

为什么要选择 C 语言？这是每个读者应该问的问题。如果掌握了 C 语言之后，还是不能满足读者的需求，那么学习 C 语言就是一件浪费时间的事情。笔者在本节尝试回答这个问题，如果笔者的回答不能令读者满意，也许读者应该选择另外一门编程语言。

##### 1. 在计算机领域，C 语言“大小通吃”

C 语言的应用极其广泛，不论是网站后台还是底层操作系统，多媒体应用还是大型网络游戏，均可使用 C 语言来开发。

(1) C 语言可以写网站后台程序。用 C 语言编写 CGI (Common GateWay Interface，使浏览器能与用户交互的一种方法) 程序，然后在 HTML 页面中嵌入 CGI，即可完成强大的功能，至于连接数据库，查询、插入数据等常规操作，当然也不在话下。对于有大量连接的网站，比如大型论坛、社区、游戏，用 C 语言编写的 CGI，比起用其他语言编写的后台程序，速度更快、性能更优。



(2) C 语言可以写出绚丽的 GUI 界面。无论在 Windows 平台还是 Linux 平台上, 用 C 语言都可以写出绚丽华美的 GUI 窗口界面来。类似 QQ、MSN 等软件的 GUI 界面, 都可以通过 C 语言实现。

(3) C 语言可以专门针对某个主题写出功能强大的程序库, 然后供其他程序使用, 从而节省其他程序的开发时间。比如常用的压缩、解压缩软件, 就有专门的 zlib 库; mp3 解码软件, 有 libmad 库; 还有以前的 DOS 时代 Borland 公司提供的图形库等。有了各种各样的程序库后, 程序员开发软件时, 就可以把这些库拿来直接使用, 组装成自己所需的软件。而这些库一般都是用 C 语言写成的, 既高效又稳定。上面提到的很多库中都有 C 语言源代码可以供学习研究。

(4) 用 C 语言可以写出大型游戏的引擎。游戏中需要处理的事情繁多, 很多游戏对实时的要求比较高, C 语言运行高效、快捷, 能满足这些需求。

(5) 用 C 语言可以写出另一个语言。很多语言的编译器或者解释器就是用 C 语言编写出来的。比如风靡全球的 PHP, 常被用来写网站后台程序, 再如 Ruby 等。

(6) 用 C 语言可以写操作系统和驱动程序, 并且这些只能用 C 语言编写。Linux 操作系统的全部源代码都可以从网上得到。Windows 操作系统虽然无法获取到源代码, 但是一批开源运动者用 C 语言编写了一个 Windows 克隆版本的操作系统 ReactOS, 与 Windows 几乎一模一样, 它的代码也是开源的, 可以通过访问网站 [www.reactos.org](http://www.reactos.org) 获取相关信息。

(7) 任何设备只要配置了微处理器, 就都支持 C 语言。从微波炉到手机, 都是由 C 语言技术来推动发展的。

一句话, 没有 C 语言干不了的事情! 何况它同时干了这么多事情。

## 2. 掌握了 C 语言, 其他类似语言不学自通

当掌握了 C 语言后, 再去学习其他面向过程的语言, 最多一个星期就能学会。因为万变不离其宗, 其他语言只是在语法上有些许更改, 而思想却没有更改。

## 3. C 语言久经考验, 有现成的大量优秀代码和资料

因为 C 语言已经存在很多年了, 它有广泛的使用团体, 并且有大量的现成代码可以利用。这就使读者能在过去程序的基础上, 快速和高效地编写新的算法和函数。C 语言是一个开源组织的语言, 在全球著名的开源组织网站 [www.sourceforge.net](http://www.sourceforge.net) 上, 能找到任何想要的开源代码。C 语言使用者众多, 讨论者也就众多, 开发出了数不尽的资料可供学习。

## 4. 简洁、紧凑, 使用方便、灵活, 功能强大, 执行效率高

所有的优点都是基于 C 语言的简洁、紧凑, 使用方便、灵活, 功能强大, 执行效率高。C 语言仅有 32 个关键字, 9 种控制语句, 却能完成无数的功能。在某些方面 C 语言可能确实不如其他语言优秀, 比如在字符串处理方面就不如 Perl 语言; 在数值计算方面就不如 Fortran 语言; 在人工智能方面就不如 Lisp 语言。可是这些语言在其他方面却远远不及 C 语言。而且 C 语言其他的诸如表达力强、移植性好的特点, 也许现在读者还无法理解, 随着时间的推移, 将会慢慢了解到。

如果上面的回答还是不能满足读者的需求, 那么最后一条一定可以满足: 精通 C 语言, 工作不用愁!

## 1.2 如何学好 C 语言

无论出于什么目的，一旦下定决心准备学习 C 语言，就要端正思想，只是听说 C 语言难，所以觉得学不好，这是不可取的。只要读者掌握了一些方法，克服了畏难情绪，并且不轻言放弃，那么就完全可以学好。以下是一些基本方法：

(1) 多动手多求人。所有的问题都可以通过自己编写代码、观察结果解决。凡是可以通过编写代码观察到结果的问题，都不应该成为一个问题。不会的，也不要太固执，多问问有经验的人。

(2) 多学习优秀代码。C 语言灵活简洁，即使编写出不好的代码，也能编译出可以运行的程序来。但是还有更优秀的编程技巧，可以让程序更好地工作，这就要求读者多学习其他人编写的优秀代码。

(3) 多以人类的思考方法来类比计算机。计算机需要什么数据、如何获取这些数据、得到后如何存放、如何处理、处理后如何表现等，对这些问题要多问些为什么，一旦理解了计算机处理这些问题的过程，编程就是一件非常轻松的事情了。

(4) C 语言只是一个基本工具，要想编写强大的软件，必须学习相关操作系统的 API（应用程序编程接口），熟悉其他类库的使用方法，才能开发出满足用户需求的软件。

本书已经考虑到 C 语言难学的情况，将难点分散到各个章节，尽量以非计算机专业术语讲解，容易理解。同时尽量用图示和实例代码来帮助读者更快地学会 C 语言。

## 1.3 语言概述

一提到语言这个词，人们自然会想到像英语、汉语这样的自然语言，因为语言是人和人相互交流信息不可缺少的工具。而今天，计算机遍布了我们生活的每一个角落，除了人和人之间的相互交流之外，我们还必须和计算机交流。用什么样的方式才能和计算机做最直接的交流呢？人们自然想到的是最古老同时也是最方便的方式——语言。

### 1.3.1 什么是语言

类比人类的语言，如汉语、英语、法语等，可以总结出语言有如下特点：

(1) 语言是用来交流沟通的。有一方说，有另一方听，必须有两方参与，这是语言最重要的功能。语言就是用来表达意思、传递信息的。说的一方传递信息，听的一方接受信息；说的一方下达指令，听的一方遵从命令做事情。没有语言，双方就很难交流沟通。

(2) 语言有独特的语法规则，交流双方都必须了解并遵守这些规则。一个只会说汉语的中国人，和一个只会说法语的法国人，如果戴上面具，只通过嘴巴发出声音互相交流，结果一定是鸡同鸭讲，信息完全传递不出去。为什么？因为互相不知道对方的语法规则，当然听不懂了。为什么要戴面具？为什么只能通过嘴巴？因为人类的一些面部表情，身体动作，这些是相通的，不通过声音，而通过肢体语言也能多少表达出一些意思。

### 1.3.2 什么是机器语言

计算机是一个忠实的仆人，时刻等候着主人的命令。如何才能使计算机听话呢？当然是用计算





机听得懂的语言去命令它了。计算机的大脑或者说心脏就是 CPU，它控制着整个计算机的运作。每种 CPU 都有自己的指令系统。这个指令系统就是该 CPU 的机器语言。机器语言是一组由 0 和 1 系列组成的指令码，这些指令码是由 CPU 制作厂商规定出来的，然后发布出来请程序员遵守。如下面是某 CPU 指令系统中的两条指令：

1000000	加
1001000	减

要让计算机完成相应的任务，就得用这样的语言去命令它。这样的命令不是一条两条，而是上百条。由于不同型号的计算机的指令系统即机器语言是不相通的，按一种计算机的机器指令编制的程序，不能在另一种计算机上执行。

用机器语言编写程序，编程人员首先要熟记所用计算机的全部指令代码和代码的含义。在编写程序时，程序员得自己处理每条指令和每一数据的存储分配和输入输出，还得记住编程过程中每步所使用的工作单元处在何种状态。这是一件十分烦琐的工作，编写程序花费的时间往往是实际运行时间的几十倍或几百倍。而且，编出的程序全是由 0 和 1 组成的指令代码，不仅直观性差，还容易出错（读者可参考图 1-1 中所示的机器语言部分）。

### 1.3.3 什么是汇编语言

在用机器语言编程的实践中，一批顽强而聪明的先行者终于发明了汇编语言——一门人类可以比较轻松掌握的编程语言。只是这门语言计算机并不懂，人类还不能用这门语言命令计算机做事情。

所以，有一类专门的程序，既懂机器语言，又懂汇编语言，而且还很聪明，知道怎么把汇编语言翻译成机器语言。于是，人类和计算机间又有了一种新的交流方式，而且人类可以比较轻松地编写程序了。

上文提到过，不同的 CPU 有不同的指令系统，从而就有不同的机器语言与之一一对应。计算机硬件不同，机器语言就不同，汇编语言也不同。所以程序员用汇编语言编写程序，都要记住是在什么 CPU 上编写的。程序员不仅要考虑程序设计思路，还要熟记计算机内部结构，这种编程的劳动强度依旧很大。为了使读者对机器语言和汇编语言的表现形式有个感性认识，笔者截取了 Visual C++ .NET 在调试的时候所看到的汇编语言窗口，如图 1-1 所示（读者现在不必太在意它们的具体含义）。



图 1-1 在调试界面中的机器语言、汇编语言、C 语言



### 1.3.4 面向过程的语言

汇编语言和机器语言都是面向机器的，机器不同，语言也不同。既然有办法把汇编语言翻译成机器语言，难道就不能把其他更人性化的语言翻译成机器语言吗？1954年，Fortran语言出现了，其后相继出现了其他的类似语言。这批语言使程序员摆脱了计算机硬件的桎梏，把主要精力放在了程序设计上，不再关注底层的计算机硬件。这类语言称为高级语言。同样的，高级语言要被计算机执行，也需要由一个翻译程序将其翻译成机器语言，这就是编译程序。

这类高级语言解决问题的方法是，分析出解决问题所需要的步骤，把程序看成数据被加工的过程。基于这类方法的程序设计语言称为面向过程的语言。C语言就是一种面向过程的设计语言。

### 1.3.5 什么是C语言

如果读者对C语言的历史感兴趣，可以参考其他相关书籍。本书不再罗列众人皆知的事实了。一般来说，C语言可以简称C（注意，C是大写的）。至于什么是C语言，请读者自学完本书后，作一个定义吧。

## 1.4 程序的开发周期

在Windows下，利用“记事本”（Notepad.exe）这个小软件，可以输入并编辑文字，然后保存到计算机硬盘上，如图1-2所示。保存到硬盘上的数据以文件的形式存在，如要将文件保存到“d:\”目录，在保存的时候，“记事本”软件会提示用户输入文件名和保存的路径，例如我们可以用“C.txt”作为文件名，以“d:\”作为文件路径。保存后，通过Windows的文件浏览器定位到“d:\”就可以看到文件“C.txt”。同样的，用画图小软件可以信手涂鸦，也能保存一个扩展名为bmp的文件到硬盘上，如图1-2所示。

可以看到，“记事本”和“画图”都可以产生出文件来。这些文件被称作“文档”。这些文档都可以被应用软件打开，它们自身是无法运行也无法展现其内容的。如：.txt文本文件，要查看其内容，一种办法是用“记事本”软件打开；又如：.mp3音乐文件，想要听其声音，一种办法是用mp3播放软件打开。那么读者有没有想过，如何产生一个.exe的可执行文件呢？

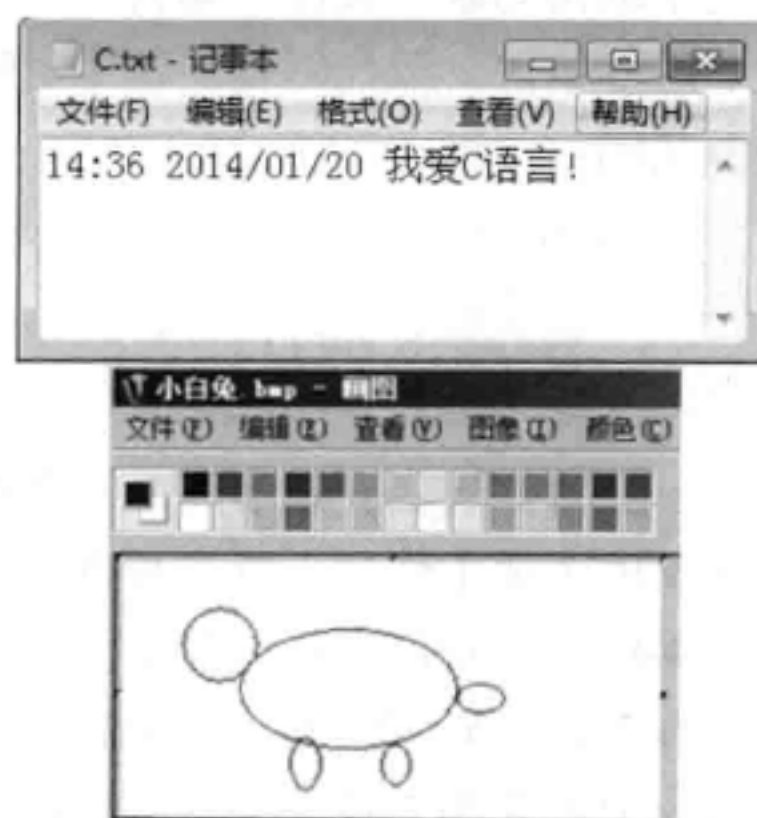


图 1-2 记事本和画图可以产生文件

聪明的读者一定知道了。是的，就是通过编写某种语言的源代码，编译成功通过后，再经过连接，成功后就出现了计算机可以执行的一个.exe文件了。这就是所谓的程序。是不是只有C语言才能编制出.exe的程序来呢？当然不是的。据不完全统计，全球现存的编程语言多达2500多种，这其中大部分都可以经过编译连接，最后产生出一个.exe可执行文件来。但是基本上，它们都遵循同样的流程：编辑源代码，编译源代码，连接目标程序，最后生成一个.exe可执行文件。用C语言开发程序的流程如图1-3所示。

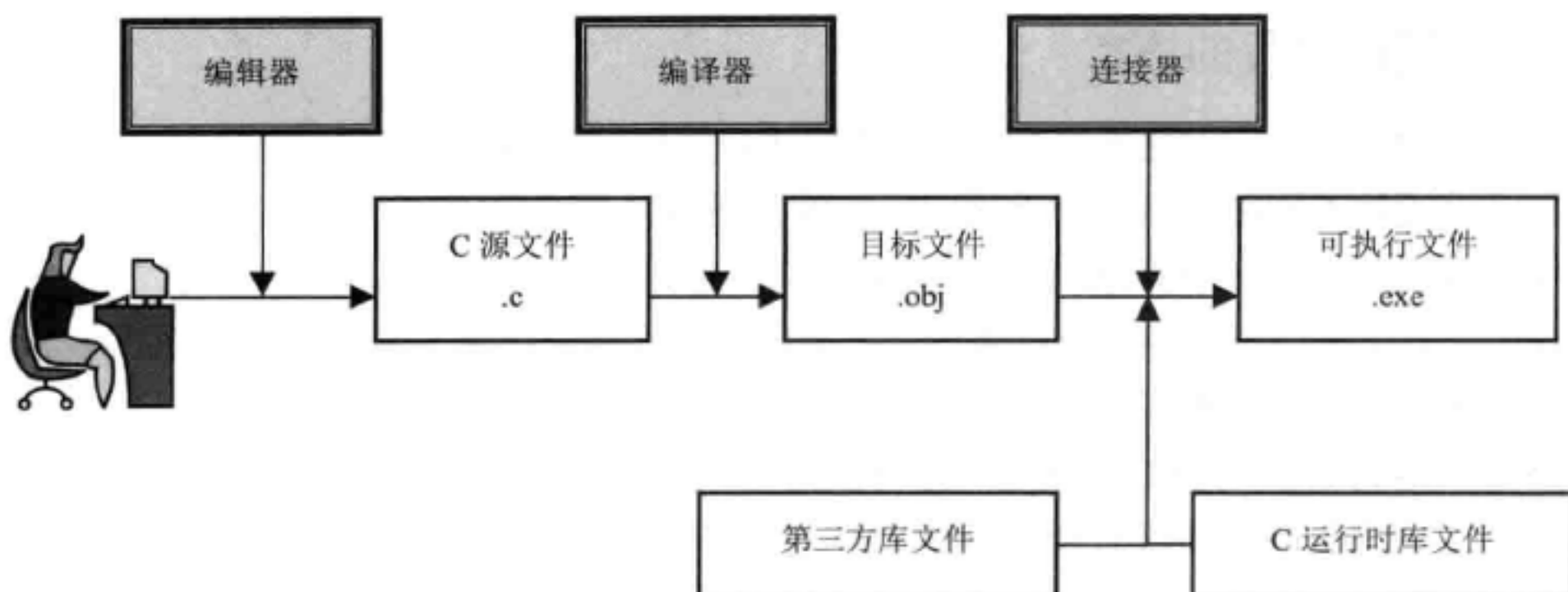


图 1-3 开发程序的流程

### 1.4.1 编辑 C 源代码

编辑 C 源代码就是做如下工作：

- (1) 逐个输入字符，如汉字、英文、标点符号或者其他可以用键盘输入的字符。
- (2) 通过插入、删除、移动、复制、粘贴等方法修改已经输入的字符。
- (3) 将输入、修改完毕的所有字符保存到硬盘上。

一篇由汉字、英文、标点符号或者其他可以用键盘输入的字符组合的内容被称作文本。能够进行文字编辑的软件被称作编辑器。

通俗地讲，源代码就是程序员输入编写的、符合 C 语言语法规则的文本。如下列片段就是一段源代码：

```
void main(void)
{
    printf("\nHello World!");
}
```

一般用扩展名.c 表示其为一个 C 源代码文件。源代码文件简称源文件，有时候也叫做源程序。程序员的主要工作之一就是根据需求编写源代码。

编辑器的功能在很大程度上能够帮助程序员提高工作效率。只要能输入文字的文本编辑软件都可以作为源代码编辑器。如记事本软件、字处理软件 Word、Ultra Edit、Edit Plus 等。但是专业程序员一般都采用专业的源代码编辑器。业界鼎鼎有名的编辑工具有 VI/VIM、Emacs/XEmacs 等。一个好的源代码编辑器，要求具备关键字着色功能（可以用不同的颜色表示代码的不同部分）、优秀的代码跳转功能、代码自动补全功能等。虽然用最普通的记事本软件也能编辑代码，但是却十分不方便。

### 1.4.2 编译 C 源代码

编译是把 C 语言源代码翻译成用二进制指令表示的目标文件。注意，这里的目标文件与机器语言还有一段距离，并不是真正的机器语言，所以不能被计算机直接运行。编译着重于“译”，就是翻译。



**声明**

读者要注意区分编辑和编译的概念。虽然一字之差，意义却大不相同。编辑，是指对文本的修改、插入、删除等操作；而编译却是将编辑好的源代码翻译成目标文件。

编译过程由 C 编译系统提供的编译程序完成。编译程序简称为编译器。编译程序运行后，自动对源程序进行句法和语法检查，当发现错误时，就将错误的类型和所在的位置显示出来，以帮助用户修改源程序中的错误。用户可以再利用编辑器对源程序进行修改。修改好后，重新进行编译，直到编译通过为止。如果未发现句法和语法方面的错误，就自动形成目标代码，并对目标代码进行优化后生成目标文件。

目标程序文件的扩展名为.obj，它是目标程序的文件类型标识。不同的编译系统，或者不同版本的编译程序，它们的启动命令不同，生成的目标文件也不相同，扩展名有时候也不一定相同，当然格式也不相同。但是其作用相同。

一个 C 源代码文件编译后就会产生一个目标文件与之对应。一般不会出现多个源代码文件对应一个目标文件的情况。进行软件开发涉及的源代码文件个数，不会像教学 C 语言这样简单到只有一个源文件，而是几十、上百甚至成千上万个。所以大型软件的开发一般通过工程文件的方式来管理源代码。

请读者思考一下，为什么软件开发不直接从源代码一步到位翻译成可执行文件，而要经过编译后，再经过连接这个步骤呢？这个问题，留在后续章节再解释。当读者明白了这个问题后，也就明白了目标文件存在的意义了。

### 1.4.3 连接目标文件

多个源代码文件经编译后产生了对应的多个目标文件，此时还没有将其组合装配成一个可以运行的整体，因此计算机还是不能执行。连接过程是用连接程序将目标文件、第三方目标文件、C 语言提供的运行时库文件连接装配成一个完整的可执行的目标程序。连接程序简称连接器。

可执行程序文件的扩展名为.exe，是可执行程序的文件类型标识。绝大部分系统生成的可执行文件的扩展名是.exe，但是在 UNIX 系统中，除非在编译时用户特别规定自己的文件名，否则生成的可执行文件自动确定为 a.out。有的 C 编译系统把编译和连接放在一个命令文件中，用一条命令即可完成编译和连接任务，减少了操作步骤。

程序员开发程序，除了要编写自己的代码外，有时候会使用其他人提供的库文件。如读者要编写一个 mp3 播放器软件，对于 mp3 解码部分，因为已经有现成的第三方代码库做好了这件事情，可以直接拿这个第三方库文件来使用。这个库提供的功能可供用户的播放器软件调用。为了方便开发，C 语言也有一批库函数，一般编译厂商都会提供给开发人员使用。

### 1.4.4 编译连接过程示例

有时候为了叙述简便，将编译连接这两个步骤，统一用“编译”一个词语代替，读者应该清楚实际是经历过了两步。在 VC.net 编程环境里，当用户下达 build（构建）命令后，开发环境就开始编译连接工作。本节的示例没有列出源文件，源文件名是 main.c，内容可以暂时不考虑，读者只关注编译、连接的步骤即可。



(1) 当源代码中没有错误时，其工作过程输出如下：

```
----- Build started: Project: 9.1, Configuration: Debug Win32 -----

Compiling...
main.c
Linking...

Build log was saved at "file:///e:/study/C Study/9.1/Debug/BuildLog.htm"
9.1 - 0 error(s), 0 warning(s)
```

从这个 Build 的过程中，明显看出经历了“Compiling...”（编译）、“Linking...”（连接）两步。最后结果是“0 error(s), 0 warning(s)”，即没有错误也没有警告。

(2) 如果源代码有错误，在编译过程就会提示用户。由于没有通过编译，也就没有目标文件，所以连接也就不进行了。一个源代码错误编译不通过的示例如下：

```
----- Build started: Project: 9.1, Configuration: Debug Win32 -----

Compiling...
main.c
e:/study/C Study/9.1/main.c(20) : error C2143: syntax error : missing ')' before '{'

Build log was saved at "file:///e:/study/C Study/9.1/Debug/BuildLog.htm"
9.1 - 1 error(s), 0 warning(s)

----- Done -----

Build: 0 succeeded, 1 failed, 0 skipped
```

#### 说明

读者现在可能还不明白错误提示信息的含义，不用担心，在后面的章节中会慢慢讲到。现在读者只需要关注如果源代码有错误，会出现什么情况即可。

“e:/study/C Study/9.1/main.c(20) : error C2143: syntax error : missing ')' before '{'”这行输出表示，代码第 20 行（“()”内表示出来的）出现错误。错误代码是 C2143，具体错误是语法错误，在“{”前缺少“)”。双击错误提示可将鼠标光标定位到错误行处，可以修改源代码。

(3) 有时候编译通过了，但是连接却不一定通过。如下：

```
----- Build started: Project: 9.1, Configuration: Debug Win32 -----

Compiling...
main.c
Linking...
LIBCD.lib(crt0.obj) : error LNK2019: unresolved external symbol _main referenced in
function _mainCRTStartup
Debug/9.1.exe : fatal error LNK1120: 1 unresolved externals

Build log was saved at "file:///e:/study/C Study/9.1/Debug/BuildLog.htm"
9.1 - 2 error(s), 0 warning(s)
```

```
----- Done -----
```

```
Build: 0 succeeded, 1 failed, 0 skipped
```

编译 main.c 后没有提示信息，表示通过，产生了.obj 文件。Linking 提示后，表示开始进行连接。但是在连接过程中出现错误，最终没有产生.exe 文件。同样，具体的连接错误原因，这里不再分析了。

### 1.4.5 运行程序

运行程序是指将可执行的目标程序投入运行，以获取程序处理的结果。如果程序运行结果不正确，可重新回到第一步，对程序进行编辑修改、编译和运行。与编译、连接不同的是，运行程序可以脱离语言开发环境。因为它是对一个可执行程序进行操作，与 C 语言本身已经没有联系，所以既可以在语言开发环境下运行，也可直接在操作系统下运行。

## 1.5 VC++、C++、C 和 TC 的区别

VC++、C++、C 和 TC，这几个语言名中都带有 C，可以说和 C 都有联系。

### 1. C

C 就是指 C 语言。C 语言的关键字少，而且拥有丰富的运算符和数据类型，可以解决大部分的“计算型”的问题或者“描述型”的问题。各大操作系统都提供了各种对 C 语言的集成化的调试编译环境，使用 C 语言编写的程序可以轻松地运行在各种平台上而不用做出任何修改，这也是 C 语言流行的原因。

### 2. VC++

VC++，一般是指微软公司的 Visual Studio 6 开发套件中的 Visual C++ 开发环境。Visual Studio 6 套件中包含了 Visual C++、Visual Basic、Visual FoxPro 等语言的开发环境。所谓开发环境，是集成了源代码编辑、编译、连接、调试等功能的一个综合程序。

Visual C++ 就是一个很好的 C 或者 C++ 开发环境。一般被简称为 VC 或者 VC++。该开发环境提供了优秀的代码编辑功能，同时提供了编译连接程序，在该开发环境中，输入完源代码，可立即编译运行，并且可以参照代码进行调试。

现在 Visual Studio 已经发展到了 Visual Studio .net 2012 版本，相对于 Visual Studio 6.0 又有比较大的改变。笔者平时工作时，一般使用 Visual C++ 开发工具，如图 1-4 所示就是该环境的一个快照。

图 1-4 是正在使用 Visual C++.net 进行调试的一个快照。从图中可以看到当前处于暂停 (break) 状态，程序运行到 “printf(“\nPlease input a number(0-100):”);” 这条语句，当前的变量 i\_score 的值是 0。函数的调用过程是：

```
9.1.exe!main() Line 12 C++
9.1.exe!mainCRTStartup() Line 259 + C
```

对上面提到的一些词语不理解没有关系，这里只是描述这个快照的情况，使读者对集成的编程环境有个感性认识。等读者有了一定的编程经验后，再回头来看就可以明白。如果读者没有安装



VC.net, 只需要明白编程环境的概念即可。



图 1-4 Visual C++编程环境

如图 1-1 所示也是 VC.net 的一个快照, 显示的是汇编语言窗口。从图 1-1 可以清楚地看到 C 语言被翻译成的汇编语言以及其对应的机器语言。从图 1-1 和图 1-4 中可以看出, VC.net 是一个比较方便的开发环境。笔者机器上还安装了 Visual Assist X 扩展工具, 这是一个扩展 VC.net 环境的一个小软件。从菜单上可以看到 Build、Debug 菜单项, 顾名思义, Build 就是编译相关的菜单项, 从中可以找到编译需要的一些命令; Debug 是调试的菜单项, 从中可以看到调试的相关命令。

**说明** 可以在VC的这个开发环境中进行编译、连接和运行。

对于编译有专门的编译程序, 同样, 连接也有专门的程序, 在 VC 的安装目录下面可以找到这些程序, 而通过开发环境编译连接的时候, 由开发环境在后台悄悄地去调用这些程序。从图 1-5 中可以发现这些程序的藏身之处。其中, cl.exe 就是微软的编译器, link.exe 就是微软的连接器, 它们都可以单独执行。cl.exe 执行后的情况如图 1-6 所示。

C 语言于 1987 年被标准化, 称为 ANSI C。由于不同软件厂商都可以开发出自己的 C 语言编译器, 在推出的编译器里, 多多少少会增加自己的特性, 这些特性被称作语言扩展。但是这些编译器都支持 ANSI C。如果使用了其中的语言扩展, 则在其他编译器上就不能被正确编译。为了不同编译器都能编译同一份源代码, 所以应尽量不使用各厂商的语言扩展功能。



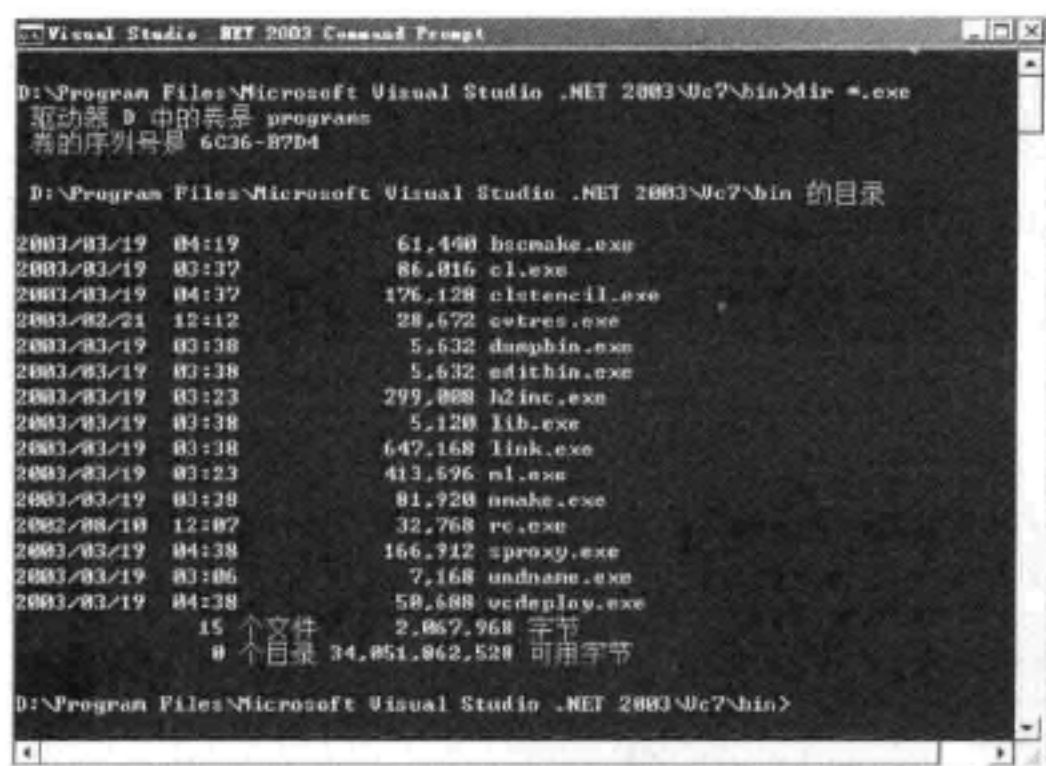


图 1-5 VC.net 的编译连接程序藏身之处



图 1-6 cl.exe 执行后的情况

所以说, VC++并不是一门语言, 而是开发环境。一般来说, 现在进行软件开发, 都是在集成的开发环境中进行的。当然, 如果愿意, 也可以单独编辑源代码, 然后用命令行的方法编译并连接程序。

### 3. C++

C++是另外一门有些类似 C 语言语法的面向对象的高级语言。虽然 C 语言不加修改就可以被 C++编译器编译, 但 C 和 C++是完全不同思想的两种语言, 不应将 C++看成 C 的超集。

### 4. TC

TC 是 Borland 公司早年在 16 位机器上开发的 C 编程环境, 是 Turbo C 的简称。最后版本是 2.0, 一般简称为 TC2。一般学习 C 语言, 都使用该编程环境, 只是这已经算是老古董了, 对于现在的操作系统, 使用 TC2 来编写程序已经很不合时宜了。如图 1-7 所示是 TC2 开发环境下输入完源代码后编译完毕的状态。

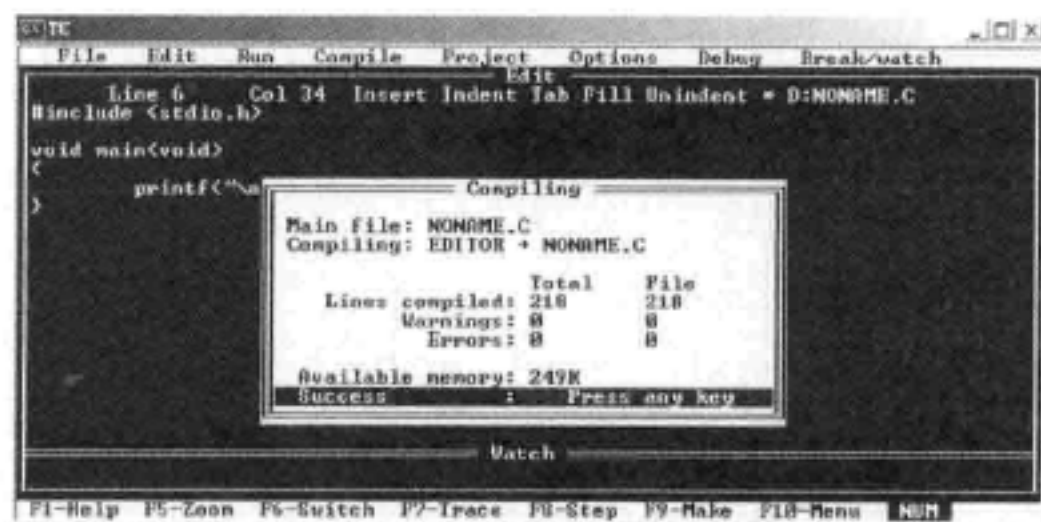


图 1-7 TC2 开发环境

### 5. 其他编译器及环境

Linux 下面开发软件主要使用 GCC (GNU Compiler Collection, GNU 编译器集合), 因为它免费。

Windows 下面除 VC++外, 也还有不少其他的 C 语言开发环境, Dev-C++是一个值得推荐的开发环境, 因为它开源。Code::Blocks 同样也是开源的。LCC-Win32 是免费的 C 小型编译器, TCC 是轻型 C 语言编译器。

**注意** 本书使用LCC-Win32作为开发环境。

## 1.6 小结

要想学好 C 语言程序设计, 为以后的学习打下一个好的基础, 那么从一开始就要有好的学习

态度和正确的学习方法。本章先是带领读者学习了 C 语言的一些特征，还了解了编程相关的一些背景知识。读者应该对什么是计算机语言有了一个大概的了解。通过本章的学习，读者还应该了解一个可运行的程序是如何被生产出来的，了解了编程的步骤及开发环境和语言的区别。这些知识看似与编程语言学习关系不大，但是却对学习编程语言大有帮助。

## 1.7 习题

### 一、填空题

1. C 语言开发程序所经过的 4 个步骤是\_\_\_\_\_、\_\_\_\_\_、\_\_\_\_\_和\_\_\_\_\_。
2. C 程序编译后，目标程序文件的扩展名是\_\_\_\_\_。

### 二、上机实践

1. 使用读者自己熟悉的编辑器，写一个输出“Hello World”的程序。

#### 【提示】

```
printf("\nHello World!");
```

2. 将上述代码进行编译连接等一系列程序后，找到生成的.exe 文件，复制到其他文件夹，如从 C 盘复制到 D 盘，并运行一下，测试是否能正确输出“Hello World”。

## 第 2 章 跟我写 Hello World

第 1 章的习题曾经让读者自己写一个“Hello World”程序，如果读者没有写出来或者说没有编译成功，那么不要着急。本章将通过编写一个“Hello world!”程序，来实践 C 语言最基本的语法特性。

本章包含的知识点如下：

- ☐ 在创建程序前要了解的内容
- ☐ 本书所用的 C 语言开发环境
- ☐ 新建程序并进行编译、连接等一系列完整操作
- ☐ 调试 C 语言源程序

### 2.1 了解需求才能创建程序

就像小学生解应用题一样，在答题之前，必须先了解题目给出的条件，然后明确题目的问题，最后才是解题。开发软件也需要这样的过程，必须先清楚用户的需求，根据需求来进行设计和开发，不遗漏需求，也不能有超出需求的功能。

本章的目标就是创建一个可以运行的程序，并输出一句话：“Hello World! ”。如果读者看见了这个要求后就准备开工写代码，则犯了软件开发的大忌。这个需求其实很不明确，并没有规定如何输出这句话。在实际的软件开发工作中，类似不明确的需求比比皆是，所以动手写代码前，一定要非常明确软件的需求。

很多曾经有过语言学习经历的读者看见上面的需求后，第一印象就是在控制台中打印出这句话来，然后就开始思索如何设计程序了。但是假设用户的需求是用人声读出这句话呢？声音也是输出啊！或者用户的需求是在一个窗口界面上输出这句话呢？更有甚者，还要求这句话的字体、颜色、大小符合规定，那么只是在控制台中打印一句“Hello World!”的程序，显然是不符合需求的。所以，在开发软件前，用大量的时间进行实际需求的调研是非常有必要的。否则，花费大量时间开发出来的程序可能是没有任何价值的。

更明确的需求是：在控制台程序中，输出一句简单的文字：“Hello World! ”。具体要求明确后，就可以开始进行设计程序了。本章只是引导读者认识 C 语言，所以这个程序简单到没有输入，没有处理，就只有一句话的输出。

### 2.2 认识 LCC-Win32 开发环境

使用 Linux 操作系统的读者，其编程能力已经不需要笔者去建议使用何种编程环境了。所以为



了照顾大多数读者，深思熟虑后，笔者最终选择了 Windows XP 操作系统下的 LCC-Win32 作为本书教学示例的编程环境。

### 2.2.1 为什么选择 LCC-Win32

笔者推荐使用 LCC-Win32 作为读者学习 C 语言的编程环境，是基于以下几个方面的原因：

(1) TC2 是 16 位机器上的开发环境，与现在常用的 32 位平台格格不入，除了学习之外，很少有人用它开发程序。并且在 Windows XP 平台上，其界面比较简单、丑陋，更重要的是调试运行时有些问题。如果读者的操作系统是 Windows 98 或者古老的 DOS 系统，则可以使用 TC2。

(2) VC6 或者 VC.net 都比较庞大，并且价格不菲，为了学习 C 语言购买它们，代价太大。虽然微软免费提供了命令行的 Toolkit，但是它的设置又比较麻烦。

(3) Dev-C++ 也是一个比较不错的选择，免费且开源。但是 Dev-C++ 主要还是用于开发 C++ 程序。虽然也可以编译 C 源代码，但是 C 毕竟不同于 C++，一不小心使用了 C++ 的语法，而又和 C 的意义不一样，将会迷惑使用者。

最终考虑使用 LCC-Win32，基于以下理由：

(1) 它是一个纯 C 语言编译器，虽具有一些扩展特性使它在某些方面具有 C++ 的部分特征，但它绝不是 C++。

(2) 具有简单而强大的编程环境。

(3) 使用和设置都比较简单。

(4) 它生成的程序和 TC2 生成的 DOS 程序格式完全不一样，而是完全的 Windows 格式，所以可以开发 Windows 窗口界面程序。

**注意** LCC-Win32 并不是一个免费软件，但是可以用来学习 C 语言。因为 LCC-Win32 对于非商业用途来说，可以自由使用。可以在网站 <http://www.cs.virginia.edu/~lcc-win32/> 下载最新版本。由于版权原因，本书光盘中并没有附带。本书采用了 3.0 版本的汉化版。

### 2.2.2 启动 LCC-Win32

安装 LCC-Win32 后，在【开始】菜单中就可以启动 LCC-Win32（后面简称为 LCC）。启动后的界面如图 2-1 所示。

### 2.2.3 新建 Hello World 工程

工程，简单来说是一个完整的应用程序。本节就介绍如何新建一个简单的工程。

依次单击【文件/新建/工程】菜单，在弹出的【请输入工程名称】对话框里输入工程名：“HelloWorld”，然后单击“确定”按钮关闭该对话框，如图 2-2 所示。

**注意** 在输入工程名之前，“确定”按钮是灰色的，表示不可以使用。

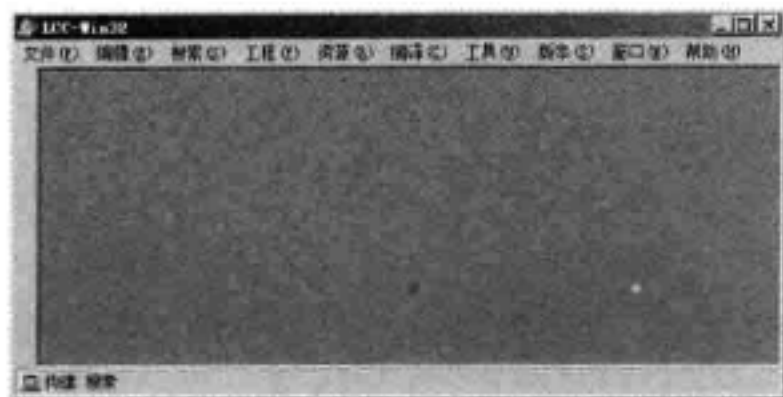


图 2-1 LCC-Win32 编程环境界面

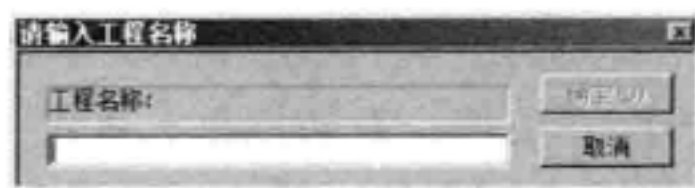


图 2-2 【请输入工程名称】对话框

## 2.2.4 定义新工程

关闭【请输入工程名称】对话框后，接着就弹出一个新的对话框，【定义新工程】对话框，如图 2-3 所示。

(1) 单击“浏览”按钮，选择工程存储路径。笔者选择的路径是“F:\CBook\src\2”，LCC 自动在【输出目录】文本框中填充了路径名：“F:\CBook\src\2\lcc”。编译后生成的.obj 文件和连接生成的.exe 就可以在这个目录下找到。如果读者将这个路径更改为其他路径，则输出文件就保存到更改后的路径中了。

**注意** 请读者记住这个路径，后面将会在这个路径下面查看生成的文件。

现在读者可以打开 Windows 的文件浏览器，定位到读者刚才输入的工程路径，其中应该只有一个 HelloWorld.prj 文件，因为此时没有输入代码，也没有编译连接，所以并没有输出文件。

(2) 【用户】选项栏中的选项可随意选择，这个是 LCC 对工程的管理形式，读者可以先不去管它。

(3) 在【工程类型】选项栏中选择【控制台应用程序】单选项。控制台应用程序就是类似 DOS 程序的一个具有黑乎乎窗口的程序，所有的输出都是文字，输入也靠键盘输入字符，但是实质并不是一个 DOS 程序。为了教学简单，以后的程序都选择【控制台程序】单选项。选择【Windows 可执行程序】单选项就是开发 Windows 图形用户界面的程序，这需要有了 C 语言基础后，再学习如何开发。本书不涉及如何开发 Windows 程序。另外，那两个单选项也要等到有了 C 语言基础后，才能开发相应的库程序。现在先不用理会。

(4) 单击“创建”按钮后，弹出【信息】对话框，提示是否使用向导生成应用程序框架，如图 2-4 所示。这里意思就是 LCC 能帮助程序员产生部分源代码，这些代码基本上是每个程序所应具备的源代码。为了读者能清楚编程的每个细节，这里选择“否”，由读者自行输入每一个代码字符。单击“否”按钮。

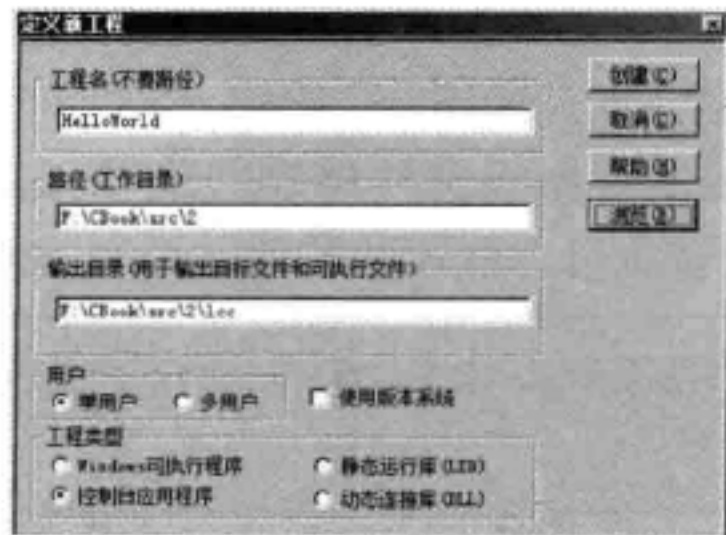


图 2-3 【定义新工程】对话框

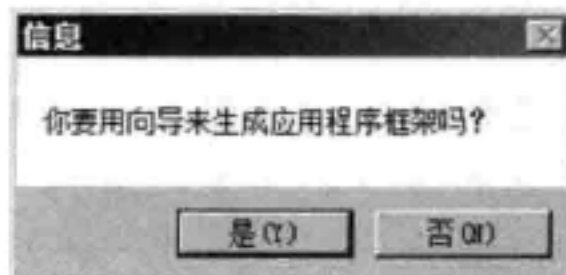


图 2-4 是否使用向导生成框架



### 2.2.5 添加源代码到工程

单击“否”按钮关闭【信息】对话框后，接着弹出【添加源代码到工程】对话框，这是一个标准的打开文件对话框。在文件名文本框中输入“2.1-HelloWorld.c”，单击“打开”按钮，然后弹出【源文件 - 工程：】对话框，如图 2-5 所示。其中列出了该工程所有已经添加的源代码文件。本示例只有一个源文件，单击“确认”按钮。

#### 注意

如有多个源文件，可以在该对话框中按“添加”按钮继续添加新的文件到工程列表中；也可以在以后需要的时候，通过菜单命令添加新文件。

### 2.2.6 编译器设置

关闭【源文件 - 工程】对话框后，接着弹出【编译器设置】对话框，如图 2-6 所示。



图 2-5 工程源文件列表确认



图 2-6 【编译器设置】对话框

在【编译器设置】对话框中，需要关注的设置如下：

- (1) 【语言扩展】选项栏。此处选择【只用 ANSIC】单选项，表示编译器只支持 ANSIC 语言语法，将 LCC 的语言扩展功能关闭。
- (2) 【杂项】选项栏。选择【生成中间文件】复选项，这样编译后将生成.obj 文件。该文件对于读者来说并无实际意义，但是为了观察开发流程，所以选择该项，以便在编译源程序后查看.obj 文件是一个什么样的文件。

设置完毕，单击“下一步”按钮，关闭【编译器设置】对话框。

### 2.2.7 连接器设置

关闭【编译器设置】对话框后，弹出【连接器设置】对话框，如图 2-7 所示。

- (1) 【输出文件名称】文本框：这里可以观察到生成的可执行文件保存的路径和文件名。“helloworld.exe”就是编译连接后生成的可执行文件名。读者也可以修改为其他路径和文件名。

#### 注意

这里的输出文件路径是根据读者前面选择的输出路径和工程名由 LCC 自动产生的。读者可以保持 LCC 产生的路径，也可以对其进行修改。

- (2) 【连接时要包含的额外文件】文本框：如果软件使用了第三方库，需要在这里指定库文件，



否则连接的时候会出现错误，找不到函数入口。

**注意** 如果读者不明白“函数入口”的概念，不必担心，在后续章节中将会详细讲解。

(3) **【输出类型】**选项栏：选择**【控制台应用程序】**单选项。

其他选项采用默认设置。设置完毕，单击“下一步”按钮，关闭**【连接器设置】**对话框。

## 2.2.8 调试器设置

关闭**【连接器设置】**对话框后，弹出**【调试器设置】**对话框，如图2-8所示。

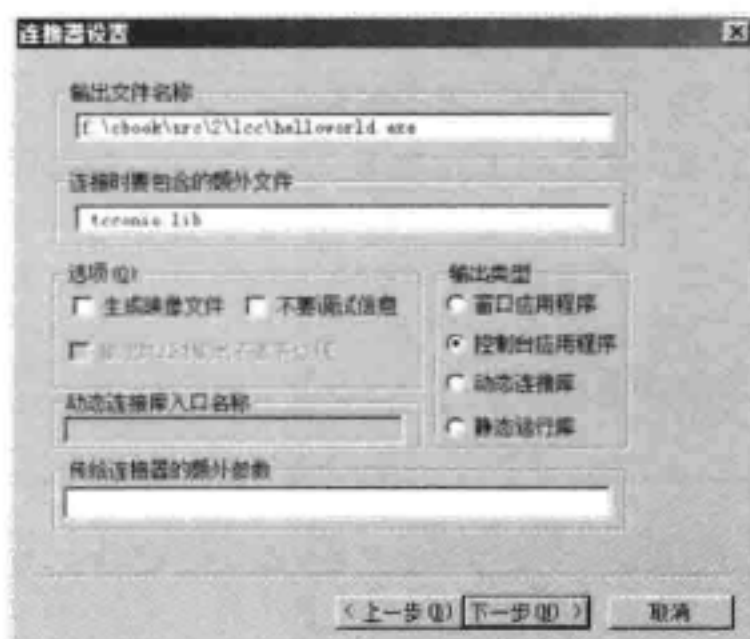


图 2-7 **【连接器设置】**对话框

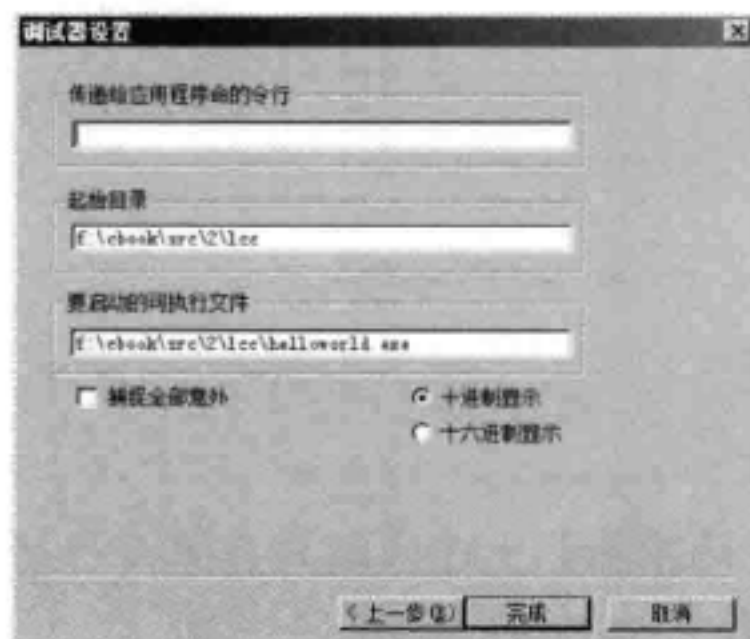


图 2-8 **【调试器设置】**对话框

调试器是用于调试源代码的程序，能够一步一步地按照源代码的顺序执行每一句代码，并且可以看见源程序中定义的变量的值。软件工程师将会耗费大量时间在调试器的跟踪调试上。

(1) **【传递给应用程序的命令行】**文本框：当程序需要参数的时候，通过这里的设置来传递给程序。参数就是类似DOS命令“dir /d /a”中的“/d /a”。本示例这里留空白，以后将会出现需要参数的情况。

(2) **【起始目录】**文本框：程序启动时的目录。当开发的程序比较大时，需要外部数据文件支持，就需要知道程序的启动目录，以便加载这些数据文件。这里也用不上，暂时不用管它。

(3) **【要启动的可执行文件】**文本框：这里就是需要调试的程序。

其他选项采用默认设置。设置完毕，单击“完成”按钮，关闭**【调试器设置】**对话框，继续其他操作。

## 2.2.9 开始编辑代码

设置完毕，就可以输入程序代码了。

**注意** 编辑代码时，涉及光标移动、复制与粘贴、块选择、添加代码注释、快速移动等功能，请读者自行熟悉LCC编辑器提供的功能。

程序代码输入完后，如图2-9所示。

在图2-9所示窗口中，有些字符是红色的，有些是灰色的，有些是黑色的。对于程序员来说，这些不同颜色的代码起到了提示的作用。HelloWorld的源代码如代码2-1所示。

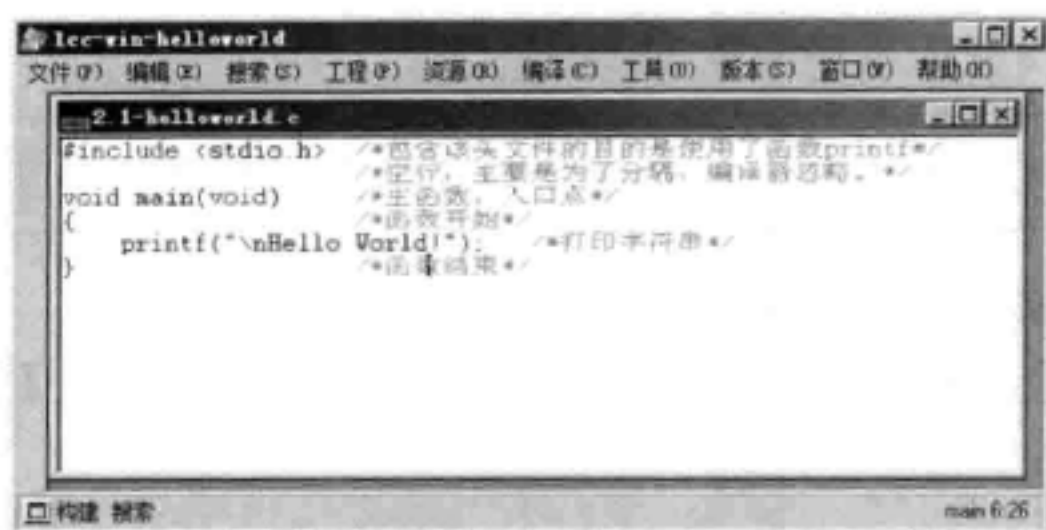


图 2-9 源代码编辑完成

代码 2-1 创建的第一个程序 HelloWorld

```

<-----文件名: HelloWorld.c ----->
01  #include <stdio.h>                                /*包含该头文件的目的是使用了函数printf()*/
02                                     /*空行, 主要是为了分隔, 编译器忽略*/
03  void main(void)                                    /*主函数, 入口点*/
04  {                                                  /*函数开始*/
05      printf("\nHello World!");                    /*打印字符串*/
06  }

```

**【代码解析】** 上述代码只有第 5 行是输出代码, 其他是构成 C 语言的一些必需代码, 这里不做详细解释, 读者可以参考下一章。

请读者将代码 2-1 自行一个字符一个字符地输入到计算机中。代码的具体含义将在第 3 章详细解释。

LCC对中文支持不太好, 删除的时候会出现半个汉字的问题。

**注意** 输入代码时不要漏掉任何字符, 也不要多输不需要的字符, 否则编译可能通不过。如果编译提示错误, 请对照代码 2-1 进行修改。

## 2.3 编译运行

代码输入完毕后, 依次单击 **【编译】** / **【构建】** 菜单命令, LCC 开始编译连接。成功后的窗口如图 2-10 所示。如果编译失败, 请对照代码 2-1 进行检查。注意不要遗漏分号“;”。

**注意** 程序中所有分号都是英文标点, 不是中文中的分号。

编译成功后, 依次单击 **【编译】** / **【执行】** 菜单命令。程序开始运行, 弹出一个控制台窗口, 如图 2-11 所示。

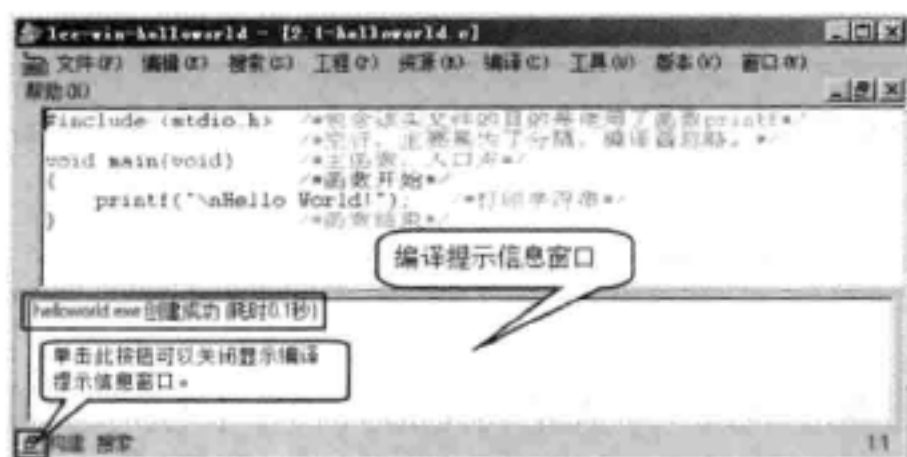


图 2-10 编译结果

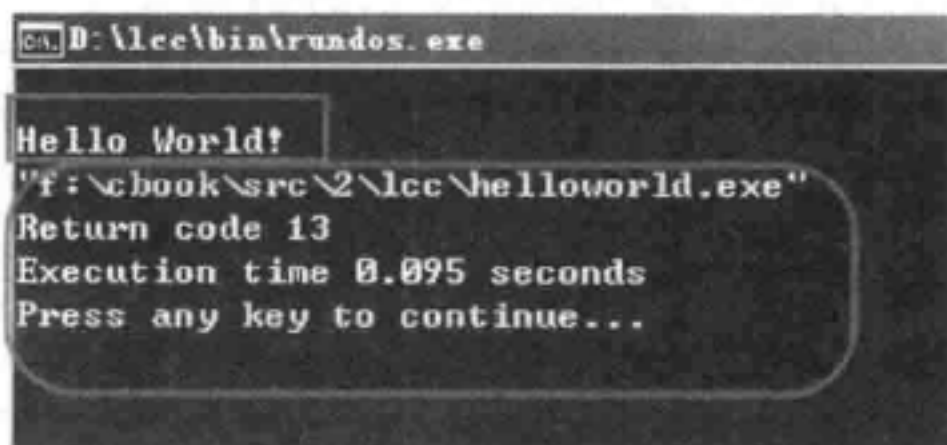


图 2-11 运行结果

---

本书仅提供部分阅读，如需完整版，请联系QQ: 461573687

提供各种书籍pdf下载，如有需要，请联系 QQ: 461573687

PDF制作说明：

本人可以提供各种PDF电子书资料，计算机类，文学，艺术，设计，医学，理学，经济，金融，等等。质量都很清晰，而且每本100%都带书签和目录，方便读者阅读观看，只要您提供给我书的相关信息，一般我都能找到，如果您有需求，请联系我 QQ: 461573687, 或者 QQ: 2404062482。

本人已经帮助了上万人找到了他们需要的PDF，其实网上有很多PDF,大家如果在网上不到的话，可以联系我QQ。因PDF电子书都有版权，请不要随意传播，最近pdf也越来越难做了，希望大家尊重下个人劳动，谢谢！

**备用QQ:2404062482**