

高等教育质量工程信息技术系列示范教材

新概念

C程序设计

大学教程

张基温 编著



清华大学出版社

## 内 容 简 介

本书是一本基于能力培养体系的 C 语言程序设计教材。本书按照作者提出的“提出问题、分析问题—编写程序、语法说明—程序测试、结果分析”的思路,并按照“前期以培养解题思路为主,语法知识够用就行;后期补充必要的语法细节”的原则编写,旨在引导读者在逻辑思维能力、语法应用能力和程序测试能力 3 个方面同步提高。

全书分为 3 篇。第 1 篇从几个经典问题入手,将读者带入穷举、迭代、递归、随机模拟、时间步长、事件步长等基本逻辑思维训练之中,并相对集中地融入基本语法,为初学者奠定程序设计的基本知识和能力。第 2 篇通过数组、结构体和指针 3 种构造数据类型以及常用算法设计策略的介绍,使读者的程序设计能力上升到“数据结构 + 算法 = 程序”的水平。第 3 篇对 C 语言重点语法进行总结、提升和拓展,使读者在发挥 C 语言优势方面得到提升。

本书结构新颖、概念准确,鱼渔并重、和木皆宜,例题经典、习题丰富、题型全面,适应面宽、注重效果,适合教学、兼顾自学,与教育部计算机科学与技术教学指导委员会推荐的《高等学校计算机科学与技术专业:公共核心知识体系与课程》中关于程序设计课程的要求一致,可以作为高等学校各专业的新一代程序设计课程教材,也可供从事程序设计相关领域的人员自学或参考。

本书封面贴有清华大学出版社防伪标签,无标签者不得销售。

版权所有,侵权必究。侵权举报电话:010-62782989 13701121933

### 图书在版编目(CIP)数据

新概念 C 程序设计大学教程 / 张基温编著. —北京:清华大学出版社, 2012. 6

(高等教育质量工程信息技术系列示范教材)

ISBN 978-7-302-28312-6

I. ①新… II. ①张… 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(2012)第 044163 号

责任编辑: 白立军 李玮琪

封面设计: 常雪影

责任校对: 白 蕾

责任印制: 王静怡

出版发行: 清华大学出版社

网 址: <http://www.tup.com.cn>, <http://www.wqbook.com>

地 址: 北京清华大学学研大厦 A 座 邮 编: 100084

社 总 机: 010-62770175 邮 购: 010-62786544

投稿与读者服务: 010-62776969, [c-service@tup.tsinghua.edu.cn](mailto:c-service@tup.tsinghua.edu.cn)

质 量 反 馈: 010-62772015, [zhiliang@tup.tsinghua.edu.cn](mailto:zhiliang@tup.tsinghua.edu.cn)

印 刷 者: 北京密云胶印厂

装 订 者: 三河市溧源装订厂

经 销: 全国新华书店

开 本: 185mm×260mm 印 张: 21.75 字 数: 513 千字

版 次: 2012 年 6 月第 1 版 印 次: 2012 年 6 月第 1 次印刷

印 数: 1~3000

定 价: 35.00 元

---

产品编号: 045868-01

# 前 言

## (一)

在信息时代,程序设计会被人称为计算机以及相关专业的看家本领,然而不仅如此,在程序设计中所蕴涵的逻辑,是解决所有领域复杂问题的根基,同时在程序设计课程中所进行的逻辑思维训练,所有人都可从中获益。但是,尽管程序设计课程已经开设几十年了,教学效果却很不尽如人意。

程序设计涉及逻辑思维、语言和方法 3 方面内容。然而,早期的程序设计课程仅仅是语言手册的改编。因此,从 20 世纪 80 年代,就开始着手改变这种状况,于 1985 年出版的我的第一本著作——《BASIC 程序设计》中就在原来的语法体系中引入典型算法和软件工程思想。以后受 CIT 考试的启发,在为 NIT(国家信息技术考试)编写的《程序设计(C 语言)》(清华大学出版社,1999)一书中,把程序测试引入到程序设计教材中。

之后,几家出版社先后约稿,出版了《新概念 C 语言程序设计》(中国铁道出版社,2003)、《C 语言程序设计案例教程》(清华大学出版社,2004)、《新概念 C 程序设计教程》(南京大学出版社,2007)、《新概念 C 语言教程》(中国电力出版社,2011)。在这些教材中,逐步形成并完善了按照内容体系的程序设计教材。所以将它们以“新概念”命名,是想表明这种全新理念的程序设计体系。让我欣慰的是,随着这几本书的不断改进,类似的书也陆续问世,品种不断增加,说明面向问题,按照“问题分析—设计代码—语法说明”线索组织程序设计教学的理念日益被广泛接受。

## (二)

本书应清华大学出版社之邀而撰写,它是对前几本教材的进一步完善与改进。全书分为 3 篇。第 1 篇由 5 个单元组成,在这 5 个单元中,以几个经典问题为载体,以穷举、迭代、递归、随机模拟、时间步长、事件步长等基本算法为主线,将设计思路、程序测试方法和 C 语言基本语法知识融于其中。第 2 篇由 4 个单元组成。第 6~8 单元分别介绍数组、结构体和指针这 3 种支持程序数据结构的重要类型,使读者可以初步领略数据结构对于程序设计的重要性,第 9 单元介绍几种常用的算法设计策略。通过这 4 个单元,使“数据结构 + 算法 = 程序”的思想在读者心中扎根。前面两篇,按照“训练解题思路为主,语法够用就行”的原则编写。第 3 篇用 6 个单元补充了一些重要的语法细节,使读者能在前两篇初步掌握了程序设计的基本方法的基础上,将 C 语言程序设计的学习引向深入。这样不仅建立了一种全新的内容体系,将应试教育向能力培养方面做了较大幅度的转变,同时也与教育部计算机科学与技术教学指导委员会推荐的《高等学校计算机科学与技术专业:公共核心知识体系与课程》中关于程序设计课程的要求一致,读者可以按照自己的专业定位选择其中的有关内容。

## (三)

为了能有有的放矢地进行训练,本书以二级节为单位给出习题。习题分为 4 个栏目:概念辨析、代码解析、探索验证和开发练习。

“概念辨析”主要提供了一些选择题和判断题,旨在提高读者对基础语法知识的认知。

“代码解析”包括指出程序(或代码段)执行结果、改错和填空,旨在提高读者的代码阅读能力。因为读程序也是程序设计的一种基本训练。

“探索验证”主要是用于提示或者指导学习者如何通过自己上机验证来提高掌握语法细节的能力。除了这个栏目中的习题外,学习者最好也能通过设计程序验证自己对于概念辨析栏目中的习题的判断是否正确。

“开发练习”是一种综合练习,应当要求学习者写出开发文档。内容主要包括问题(算法)分析、代码设计、测试用例设计、测试及调试结果分析等几个部分,重点应当放在问题分析、代码设计和测试用例的设计上。要把这些工作都做好,再上机调试、测试,不要什么还没有设计出来就上机。

#### (四)

在本书即将出版之际,由衷地感谢在本书写作过程中参与收集资料、程序调试以及校阅工作的姚威、张秋菊、文明瑶、杜勇、丁群、朱莎、史林娟、张展为、张有明。同时,也殷切地期待着广大读者和同仁的批评和建议。让我们共同把程序设计课程的改革做得更有实效。

张基温

2012年2月

# 目 录

## 第 1 篇 C 语言程序设计初步

<b>第 1 单元 简单的 C 语言程序</b> .....	3
1.1 两个整数相加 .....	3
1.1.1 两个整数常数相加的 C 语言程序 .....	3
1.1.2 C 语言程序的编译与连接 .....	3
1.1.3 带有输出操作的 C 程序 .....	4
习题 1.1 .....	6
1.2 变量初步 .....	8
1.2.1 使用变量的两整数相加程序 .....	8
1.2.2 从键盘给变量输入值 .....	10
习题 1.2 .....	12
1.3 用浮点数进行除运算 .....	14
1.3.1 整数相除的问题 .....	14
1.3.2 两个浮点数相除的 C 程序 .....	15
习题 1.3 .....	16
<b>第 2 单元 选择结构</b> .....	18
2.1 将从键盘输入的任意两个数按升序输出 .....	18
2.1.1 问题分析与参考代码 .....	18
2.1.2 关系运算符与关系表达式 .....	19
2.1.3 if-else 二分支选择结构 .....	20
2.1.4 程序测试 .....	21
2.1.5 程序的书写风格 .....	22
习题 2.1 .....	23
2.2 三中取大 .....	25
2.2.1 算法分析与参考代码 .....	25
2.2.2 逻辑运算符与逻辑表达式 .....	26
2.2.3 多分支选择结构中 if 与 else 的配对规则 .....	27
2.2.4 测试用例设计：语句覆盖与分支覆盖 .....	28
2.2.5 else if 结构 .....	29
2.2.6 条件运算符 .....	30
习题 2.2 .....	31

2.3	字符分类	34
2.3.1	字符类型	35
2.3.2	基于整型值匹配的 switch 结构	37
2.3.3	算法分析与参考代码	37
2.3.4	程序测试用例设计的等价分类法	40
2.3.5	switch 结构与 if-else 结构的比较	41
	习题 2.3	42
<b>第 3 单元</b>	<b>重复结构</b>	<b>46</b>
3.1	C 语言重复结构基础	46
3.1.1	C 语言的三种重复结构	46
3.1.2	累加器程序	47
3.1.3	打印九九乘法表	53
	习题 3.1	56
3.2	穷举	65
3.2.1	求素数	65
3.2.2	搬砖问题	67
3.2.3	推断名次	69
	习题 3.2	73
3.3	迭代与递推	78
3.3.1	用辗转相除法求两个正整数的最大公因子	78
3.3.2	Fibonacci 数列	81
3.3.3	猴子吃桃子	83
3.3.4	用二分迭代法求解一元二次方程	85
	习题 3.3	88
<b>第 4 单元</b>	<b>用函数组织 C 程序</b>	<b>92</b>
4.1	函数基础	92
4.1.1	函数定义	92
4.1.2	函数调用	94
4.1.3	函数原型声明	95
4.1.4	局部变量与全局变量	96
4.1.5	模块化程序设计	99
	习题 4.1	102
4.2	递归	105
4.2.1	阶乘的递归计算	105
4.2.2	汉诺塔	106
	习题 4.2	109

<b>第 5 单元 计算机模拟</b> .....	111
5.1 随机问题模拟 .....	111
5.1.1 产品随机抽样.....	111
5.1.2 用蒙特卡洛法求 $\pi$ 的近似值 .....	114
习题 5.1 .....	115
5.2 基于步长的模拟 .....	116
5.2.1 事件步长法——中子扩散问题.....	116
5.2.2 时间步长法——盐水池问题.....	118
习题 5.2 .....	122

## 第 2 篇 数据结构十算法

<b>第 6 单元 顺序地组织同类型数据——数组类型</b> .....	127
6.1 数组基础 .....	127
6.1.1 扑克牌的表示与数组定义.....	127
6.1.2 扑克牌查找：数组元素引用与数组名参数 .....	129
6.1.3 扑克洗牌的随机模拟.....	132
6.1.4 扑克牌整理：数组元素排序 .....	133
6.1.5 扑克发牌：二维数组应用 .....	135
习题 6.1 .....	139
6.2 字符串 .....	143
6.2.1 字符串与字符数组.....	143
6.2.2 字符串输入输出.....	144
6.2.3 字符串的其他操作.....	147
习题 6.2 .....	150
<b>第 7 单元 描述一类对象的属性——结构体类型和共用体类型</b> .....	153
7.1 结构体类型基础 .....	153
7.1.1 结构体类型的定义.....	153
7.1.2 结构体类型的实例化.....	154
7.1.3 结构体变量的引用.....	155
习题 7.1 .....	157
7.2 结构体数组 .....	159
7.2.1 结构体数组的定义与初始化.....	159
7.2.2 结构体数组元素的引用.....	161
习题 7.2 .....	163
7.3 union 类型 .....	165
7.3.1 共用体类型的定制与共用体变量的定义.....	165
7.3.2 共用体类型与结构体类型的比较.....	165

7.3.3	共用体变量的应用	167
	习题 7.3	168
<b>第 8 单元</b>	<b>指针类型</b>	<b>171</b>
8.1	指针的概念	171
8.1.1	指针=基类型+地址	171
8.1.2	悬空指针、空指针与 void 指针	173
8.1.3	多级指针	174
8.1.4	指针的操作	174
	习题 8.1	177
8.2	数组的指针形式	180
8.2.1	数组名与指向数组的指针	180
8.2.2	二维数组的指针形式	182
8.2.3	指针与 C 字符串	184
	习题 8.2	186
8.3	指针参数	188
8.3.1	变量地址传送	189
8.3.2	数组地址传送	190
8.3.3	字符指针参数	194
8.3.4	带参主函数	197
	习题 8.3	199
<b>第 9 单元</b>	<b>常用算法设计策略*</b>	<b>202</b>
9.1	分治策略	202
9.1.1	二分查找	202
9.1.2	快速排序	204
9.1.3	自行车带人问题	207
	习题 9.1	209
9.2	回溯策略	211
9.2.1	迷宫问题	211
9.2.2	使用堆栈组织搜索过程	215
	习题 9.2	220
9.3	贪心策略	221
9.3.1	旅行费用问题	222
9.3.2	删数问题	224
	习题 9.3	227
9.4	动态规划	228
	习题 9.4	232



## 第 3 篇 深入学习 C 语言

<b>第 10 单元 C 语言中常量的表示</b> .....	237
10.1 字面常量 .....	237
10.1.1 整型字面常量的表示和辨识 .....	237
10.1.2 浮点类型字面常量的表示和辨识 .....	238
习题 10.1 .....	239
10.2 宏 .....	241
10.2.1 宏定义 .....	241
10.2.2 使用宏应当注意的几点 .....	241
10.2.3 带参宏定义 .....	243
习题 10.2 .....	245
10.3 const 修饰符 .....	248
10.3.1 用 const“固化”变量 .....	248
10.3.2 用 const 修饰指针 .....	249
习题 10.3 .....	252
10.4 枚举类型 .....	253
10.4.1 枚举类型及其定义 .....	253
10.4.2 枚举变量的定义 .....	254
10.4.3 对枚举变量和枚举元素的操作 .....	254
习题 10.4 .....	255
<b>第 11 单元 数据类型</b> .....	257
11.1 基本数据类型特性 .....	258
11.1.1 整型数据类型的主要特性 .....	258
11.1.2 浮点数据类型的主要特性 .....	260
习题 11.1 .....	261
11.2 数据类型转换 .....	262
11.2.1 数据类型转换的一般规则 .....	262
11.2.2 数据类型转换的副作用与注意事项 .....	265
习题 11.2 .....	269
11.3 typedef 和 sizeof 操作符 .....	270
11.3.1 typedef .....	270
11.3.2 sizeof 运算符 .....	272
习题 11.3 .....	272
<b>第 12 单元 C 程序中变量的访问属性</b> .....	274
12.1 变量访问属性的概念 .....	274
12.1.1 变量的存储类型与生存期 .....	274
12.1.2 标识符的作用域 .....	275

12.1.3	标识符的链接属性	276
习题 12.1		276
12.2	C 语言程序实体的存储类型	277
12.2.1	C 程序中的局部变量	277
12.2.2	C 程序中的外部变量	279
习题 12.2		283
12.3	C 程序中的动态内存分配	287
12.3.1	申请存储空间	288
12.3.2	释放一个指针指向的存储空间	289
12.3.3	修改一个指针指向的存储空间大小	290
习题 12.3		291
<b>第 13 单元</b>	<b>格式化输入输出函数详解</b>	292
13.1	格式化输出函数 printf ()	292
13.1.1	格式参数结构	292
13.1.2	基本格式符	292
13.1.3	长度修饰符	293
13.1.4	域宽与精度说明	293
13.1.5	前缀修饰符	294
习题 13.1		296
13.2	格式化输入函数 scanf ()	297
13.2.1	地址参数	297
13.2.2	格式参数结构与工作机制	297
13.2.3	数值数据的输入控制	300
13.2.4	scanf ()与输入缓冲区	301
13.2.5	字符型数据的输入控制	302
13.2.6	scanf ()的停止与返回	306
习题 13.2		306
<b>第 14 单元</b>	<b>文件</b>	308
14.1	C 文件的基本概念	308
14.1.1	I/O 流与缓冲	308
14.1.2	文件及其分类	309
14.1.3	FILE 类型及其指针	309
习题 14.1		310
14.2	C 文件操作的一般过程	310
14.2.1	文件打开	311
14.2.2	文件读写定位与读写操作	312
14.2.3	文件关闭	313
习题 14.2		314

14.3	文件操作程序示例	317
14.3.1	写若干行字符串到文本文件	317
14.3.2	文件复制	317
	习题 14.3	318
<b>第 15 单元</b>	<b>位运算与位段</b>	<b>320</b>
15.1	位运算	320
15.1.1	按位逻辑运算	320
15.1.2	移位运算	322
15.2	位段	322
	习题 15	325
<b>附录</b>		<b>327</b>
附录 A	C 语言的关键字及其用途	327
附录 B	C 语言运算符的优先级和结合方向	328
附录 C	编译预处理命令	328
	C.1 宏定义	328
	C.2 文件包含	329
	C.3 条件编译	329
附录 D	C 语言常用标准库函数	329
	D.1 数学函数	330
	D.2 字符函数和字符串函数	331
	D.3 输入输出函数	331
	D.4 动态内存分配函数	333
	D.5 退出程序函数	333
	D.6 数值转换函数	334
	D.7 时间和日期函数	334
<b>参考文献</b>		<b>335</b>

# 第 1 篇 C 语言程序设计初步

计算机是人类历史上迄今为止最为神奇的一种机器。它几乎“无所不能”地在人类社会的每个领域大显身手。这种神奇之所在主要来自它的程序。因为计算机的工作完全是由程序控制的,程序安排计算机做什么,计算机才能做什么。离开了程序,计算机就只剩下一堆金属和塑料。

程序是由人编写的。为了编写程序,人要先理清自己的解题思路,建立一个问题的模型,并用计算机能直接或间接理解的符号系统描述出来。这个过程可以简单地记为

程序=模型+表现

计算机能直接或间接理解的、用于描述解题思路的符号系统称为计算机程序设计语言,简称计算机语言或程序设计语言。本书介绍的 C 语言,是目前在计算机语言市场上占据统治地位的一种计算机程序设计语言。

学习程序设计,主要涉及两个方面:培养解题思路和掌握某种计算机程序设计语言的语法规则。其中解题思路的培养尤为重要。在计算机科学中,把问题的求解思路称为算法(algorithms)。算法是程序的灵魂。相对于算法的培养,程序设计语言语法的学习就简单多了。这一篇通过几个实例,介绍几种程序中常用的算法和用 C 语言表现这些算法的基本方法,为进一步学习奠定基础。

## C 语言的来历

C 语言的来历要从 ALGOL(也称 A 语言)说起,ALGOL 是首批高级程序设计语言之一。1963 年剑桥大学基于 ALGOL 60 提出了 CPL(Combined Programming Language)。1967 年剑桥大学的 Martin Richards 对 CPL 语言进行了简化,形成了 BCPL 语言。1970 年贝尔实验室的 Ken Thompson 将 BCPL“煮”了一下,提炼出其精华,将之命名为 B 语言。他还用 B 语言写了第一个 UNIX 操作系统。1973 年贝尔实验室的 Dennis M. Ritchie 又把 B 语言“煮”了一次,在此基础上设计出了一种新的语言,将之称为 C 语言。1977 年 Ritchie 发表了不依赖于具体机器系统的 C 语言编译文本——可移植的 C 语言编译程序。1978 年 Brian W. Kernighian 和 Dennis M. Ritchie(如图 0-1 所示)出版了名著《The C Programming Language》,把 C 语言推向流行最广泛的高级程序设计语言宝座。

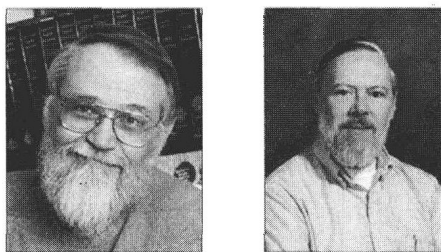


图 0-1 Brian W. Kernighian 和 Dennis M. Ritchie

随着 C 语言的流行,特别是在 1988 年后,随着微型计算机的日益普及,出现了许多 C 语言版本,导致了应用上的困难。为了改变这种情况,美国国家标准学会(American National Standards Institute,ANSI)为 C 语言制定了一套 ANSI 标准,成为现行的 C 语言标准。但是,任何标准都不可能包罗万象,总有考虑不周之处。而且 C 语言的标准也在不断调整。所以,现在实际应用的 C 语言系统,因版本不同,还在细微处存在一些区别。



# 第1单元 简单的C语言程序

本单元通过一个简单的实例,介绍C语言程序的基本结构。

## 1.1 两个整数相加

### 1.1.1 两个整数常数相加的C语言程序

**【代码 1-1】** 一个用于计算 2+3 的 C 语言程序。

```
int main (void) {  
    2+3;  
    return 0;  
}
```

说明:本例是一个最简单的C语言程序。图1-1所示为这种最简单的C语言程序的结构——它由一个名为main的函数组成。

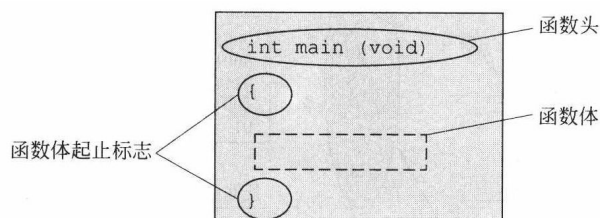


图 1-1 C 语言主函数的基本框架

main 函数由一个函数头和一个函数体组成。除非有特殊用途,否则就认为 main 函数的函数头是一个固定的形式: int main(void)。函数体是用一对花括号括起来的一组声明(declaration)或语句。声明是向编译器所作的关于数据的说明。语句用于向计算机发出操作指令。C语言规定,每个声明和语句都以分号结束。在代码 1-1 中,包含了两条语句:“2+3;”和“return 0;”。

2+3 称为一个 C 语言表达式。C 语言表达式是用操作数和操作符组成的合乎 C 语言语法的式子。在 2+3 这个表达式中,2 和 3 称为操作数;“+”称为操作符,准确地说,是一个算术操作符。C 语言的操作符有算术操作符、关系操作符、逻辑操作符、赋值操作符等十几种类型。其中,算术操作符有+(加)、-(减)、\*(乘)、/(除)、%(模,即整除取余)。例如,19%7 的结果是 5。

### 1.1.2 C 语言程序的编译与连接

C 语言不是计算机 CPU(中央处理器)可以直接理解的程序设计语言。因为 C 语言是

一种靠近英语的程序设计语言,使用这样的语言编写程序,可以使程序员把精力集中在解题思路的设计上。而计算机 CPU 所能直接理解的是用 0、1 码描述的指令系统。为此,要计算机 CPU 理解并执行一个 C 语言程序,就要将这个 C 语言程序转换——翻译成所在计算机 CPU 机器语言程序。这个过程称为编译(compile)。为了区别程序员编写的程序和经过编译的程序,将前者称为源程序代码或源程序,将后者称为目标程序代码或目标程序。实现编译功能的程序称为编译器(compiler)。

但是,目标程序还不是可以执行的程序。因为,随着计算机所求解的问题越来越复杂,计算机程序的规模也越来越庞大。为了便于发现错误和纠正程序中的错误,采用了分块编写程序的方法。一个程序在编写过程中形成了多个模块,并且一些是程序员自己编写的,有些可以使用别人或开发商提供的。要让系统正确地执行一个程序,必须将这些经过编译的程序资源同主函数连接起来,这个过程称为链接(或连接)。执行这个任务的程序称为连接器。只有经过链接的程序才可以成为可执行程序。

图 1-2 描述了一个 C 语言程序的编译和链接过程。

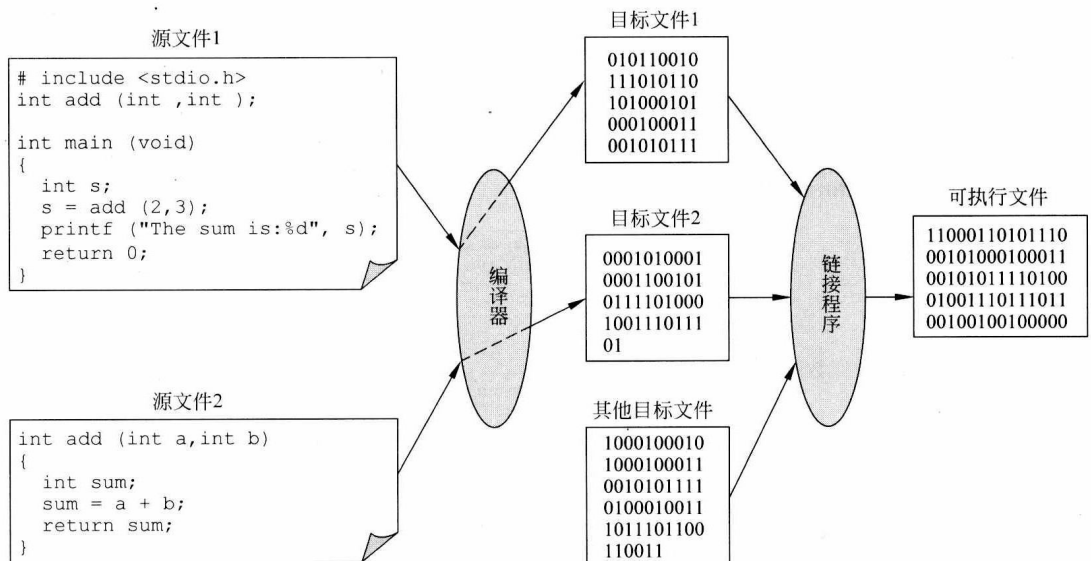


图 1-2 C 语言程序的编译和链接过程示意图

一个完整的高级语言程序的开发过程,就是从问题出发,编写出高级语言程序,再编译链接得到可执行程序的过程。图 1-3 显示了这个过程。

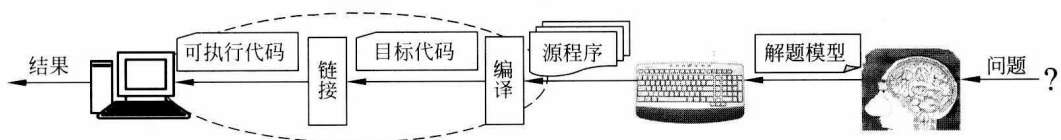


图 1-3 高级语言程序开发的过程

### 1.1.3 带有输出操作的 C 程序

如果在计算机上运行代码 1-1,计算机上什么也不会出现,即得不到任何结果。什么原

因呢？不是计算机没有进行计算，而是这个程序没有输出功能，犹如“哑巴吃饺子——心中有数”却没有表达出来。因此，一个程序必须有输出功能，把运行的状况和结果告诉用户。

**【代码 1-2】** 带有输出操作的计算 2+3 的 C 语言程序。

```
#include <stdio.h>
int main (void) {
    printf ("%d",2+3);
    return 0;
}
```

执行这个程序，计算机屏幕上显示如下：

```
5
```

有以下几点说明。

(1) 代码 1-2 的程序结构如图 1-4 所示。它由两个函数组成：main()函数和 printf()函数。在函数 main()中，用语句“printf(“%d”,2+3);”调用函数 printf()的功能。当操作系统接受人的命令开始执行这个程序时，首先从函数 main()开始执行，当需要 printf()函数的功能时，就调用函数 printf()，把执行流程转交到函数 printf()中，执行完函数 printf()，再返回到函数 main()中。

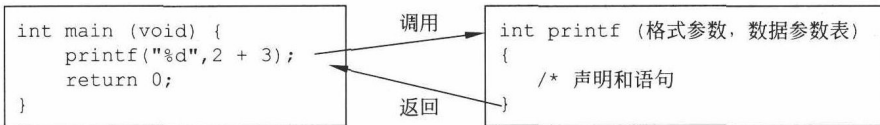


图 1-4 代码 1-2 的程序结构

一个 C 语言程序的一般情况如图 1-5 所示。即由一个或多个函数组成，但必须有且只能有一个函数 main()。其他函数都是由函数 main()直接或间接调用的。所以，把函数 main()称为主函数。或者说，执行一个 C 语言程序，就是从主函数开始到主函数结束的执行过程。

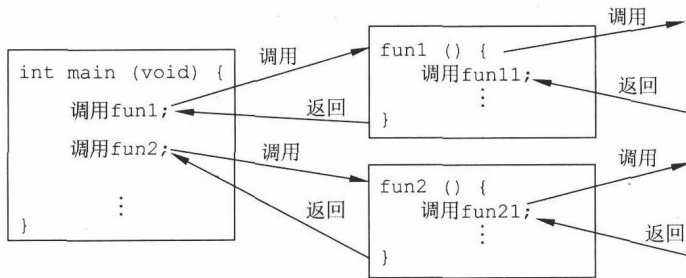


图 1-5 C 语言程序的一般结构

(2) 代码 1-2 有两个函数，可是为什么看不见函数 printf()的定义的呢？这正是 C 语言的一个优点。C 语言追求精干、高效和灵活，它没有设置专门的输入和输出语句，输入和输出采用一些库函数实现。printf()就是最常用的一个用于格式化输出的库函数。即这个函数定义在系统的函数库中。程序员要使用库函数，有两个要求：第一是要正确地写出函数



printf()的调用语句。第二是要让编译器能根据函数的基本信息(称函数原型)来检查这个函数的调用语句是否正确并找到这个函数的定义。这就是在代码 1-2 的开头写了一行“#include <stdio.h>”的原因。因为在头文件 stdio.h 中有 printf()的函数原型信息。注意, #include 是一条编译预处理命令(不是语句,不用分号结尾),用于将一个文件包含到当前程序中。图 1-6 所示为对一个源代码文件进行文件包含预处理的示意图。

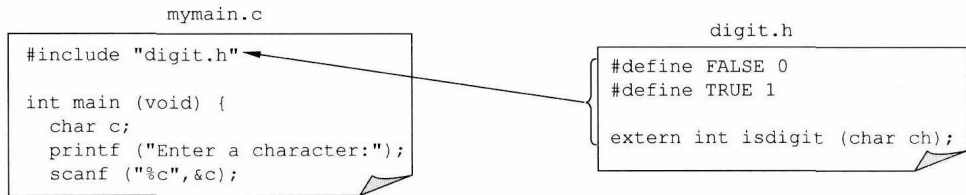


图 1-6 一个源代码文件中的文件包含预处理示意图

(3) 如图 1-7 所示,printf()的参数分为两部分:前面一部分被括在一对双撇号中,称为格式参数。其中的 %d 称为格式字段,%表明后面的字符 d 不是普通的字符,而是一个用来说明后面的数据项要用十进制整型格式输出的符号,称为整型格式字符。后面部分参数称为数据参数,如本例中的 2+3。两部分合起来就表示:把表达式 2+3 的值用整型格式显示出来。

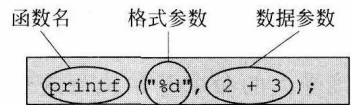


图 1-7 函数 printf()的调用表达式

用一个 printf()函数可以输出多个内容表达式的值。

**【代码 1-3】** 用 printf()函数输出多个内容表达式的值。

```
#include <stdio.h>
int main (void) {
    printf ("%d+%d=%d", 1+1, 1+2, 2+3);
    return 0;
}
```

执行这个程序,会在计算机屏幕上显示如下:

```
2+3=5
```

计算机执行这个 printf()函数时,首先分别计算后面的 3 个内容表达式的值,再分别按照顺序用计算出的 3 个值代替格式参数中的格式字段“%d”,最后将形成的一串字符显示出来。可以看出,原来格式字符串中的 +、= 都原样显示出来了,只有格式字段被用后面表达式的值替换了。显然,需要几个输出项,就需要在格式字符串中使用几个用 % 开始的格式字段。

## 习 题 1.1

### 一、概念辨析

选择正确的答案。

(1) 下面叙述中,错误的是( )。

A. 一个 C 语言程序至少要有有一个主函数