美国当代计算机职业培训系列教程

C An Introduction to Programming

C语言程序设计轻松入门

Jim keogh
(美) Peter Aitken 著 冯博琴 等译
Bradley L. Jones

机械工业出版社
西蒙与舒斯特国际出版公司

美国当代计算机职业培训系列教程

C 语言程序设计轻松入门

Jim Keogh
（美） Peter Aitken 著
Bradley L. Jones

冯博琴 等译

机 械 工 业 出 版 社
西蒙与舒斯特国际出版公司

本书讲述了怎样利用C语言进行程序设计。通过对C语言的语法结构、函数调用、数据结构、指针、文件操作、函数库及C语言的高级应用的讲解，使读者对C语言程序设计有全面性的了解。书中每章后都附有练习，使读者对相应内容加深认识。

本书可作为大专院校电子与计算机专业程序设计课程教材，也适合一般计算机用户使用。

本书版权登记号：图字：01-96-0759

图书在版编目（CIP）数据

C语言程序设计轻松入门/（美）凯奇（Keogh，J.）等著；冯博琴等译.
-北京：机械工业出版社，1996.11
美国当代计算机职业培训系列教程
ISBN 7-111-05303-6

Ⅰ.C… Ⅱ.①凯… ②冯… Ⅲ.C语言-程序设计-普及读物 Ⅳ TP312C

中国版本图书馆 CIP 数据核字（96）第 12477 号

出 版 人：马九荣（北京市百万庄南街1号 邮政编码100037）
责任编辑：傅豫皮
三河永和印刷有限公司印刷·新华书店北京发行所发行
1996年11月第1版第1次印刷
787mm×1092mm 1/16·23.25印张·580千字
0001-8000册
定价：39.00元

前　言

1. 循序渐进学习C语言

“C语言程序设计轻松入门”是一本供读者循序渐进地学习C语言的大学水平课程的教材。本书简要提出了最重要的主题并全面地讲解，使读者能够很快地成为成果丰硕的程序员。本书具备以下明显的特色：

- 每章以本章主题表开始。
- 为读者学习提供大量的例子。
- 前几章致力于介绍能使读者快速地独立进行程序设计所必需的“知识主体”。
- 章末的问题指出本章的重要概念并检验读者对内容的掌握程度。
- 每章以与本章主题有关的实习题结束，其中的一些练习是“故障诊断”，给出包含常见错误（故障）的代码列表，老师可要求完成全部或某些练习。
- 章末的词汇部分提供本章所用的关键术语的定义。

2. 本书体例

每章采用相同的格式，都以“本章主题”开始，以小结结束，小结中回顾本章中最重要的知识。小结之后是练习，检查读者掌握的程度。

前几章注重培养读者的信心和理解而且不使读者过多的失败，当然，失败在编程时是不可避免的。讲解C的更高级的知识时加快了速度。

第1章：C语言程序设计介绍

讲解编写C程序、即程序开发过程中所包含的步骤。读者要清楚地掌握编辑—编译—连接—测试的周期，以及每一步所用的工具。

第2章：C程序的结构

介绍C程序的主要组成部分，学习第一个程序的组成以及怎样用注释行说明程序。

第3章：数值型变量和常量

分析两类数值型变量，整型和浮点型。本章还讲解了字面常量和符号常量。本章中，读者将学习各种数据类型和常量说明。

第4章：语句、表达式和运算符

提供C的语句定义，学习使用复合语句（块）以及使用算术、关系和逻辑运算符来处理C表达式。

第5章：函数

讨论函数，这是C程序设计的一个重要组成部分。学习结构化程序设计，这是一种强调自上而下的逐步进行的程序设计方法。

第6章：变量的作用域

讨论C的变量存储方式，这是结构化程序设计的一个重要方面。学习外部（全局）变量和局部变量的使用。

第7章：程序控制

讲解控制程序执行的三种循环语句：for、while和do... while，同时也通过一些有意义的例子解释了嵌套的概念。

第8章：高级程序控制

讲解各种有关程序控制的主题，包括 goto 语句、break 和 continue 语句、switch 语句以及 exit（）函数。

第 9 章：基本输入和输出

讲解如何在程序中有效地使用 printf（）、puts（）和 scanf（）函数的方法。

第 10 章：数值型数组

介绍了一种强有力的数据存储方法。学习怎样使用数值型数组组织同一类数据。

第 11 章：指针

帮助用户了解指针，这是 C 程序设计的核心部分。学习怎样使用取地址运算符（&）和指向运算符（*）以及怎样用指针访问数组元素，怎样通过传送一个数组的指针而将数组作为实参传送到函数。

第 12 章：字符与字符串

讲解 C 的 char 数据类型。本章中，学习定义字符串和基本的存储器管理。

第 13 章：结构与共用体

说明怎样使用结构，这是一种程序员为满足程序需要而设计的数据类型。本章解释结构和共用体的差别以及 typedef 关键字的应用。

第 14 章：使用屏幕、打印机和键盘

说明 C 怎样使用流并讨论 5 种预定义流。

第 15 章：高级指针

讲述指针的高级应用。学习怎样使用指针的指针，了解怎样用指针数组来处理字符串。也学习 C 怎样处理多维数组以及怎样通过指针使用这些数组。

第 16 章：文件

说明用户需要了解的所有有关磁盘文件的知识，用户使用的各种输入输出函数，讲解一些有关删除和重命名磁盘文件的基本文件管理函数。

第 17 章：高级字符串操作

讨论程序操作字符串的各种方法。讲解比较、转换和字符串连接函数。

第 18 章：函数的高级应用

分析 C 程序能执行的一些另外的操作。学习由值和引用来传送参数的区别，以及怎样用 void 类型来建立能指向任何 C 数据对象的通用指针。

第 19 章：探索函数库

讲解一些 C 函数库提供的更有用的函数。其中的一些用于处理时间，另一些帮助程序进行出错处理，其余的用来排序和查找数据。

第 20 章：存储器管理

说明在程序运行时如何分配、重分配以及释放存储器。学习如何以及何时对变量和指针使用强制类型。

第 21 章：预处理器、预处理器命令、宏和命令行参数

说明怎样用预处理命令来建立函数宏，并分析由编译器提供的函数宏。

第 22 章：位处理

讲解怎样用移位和位逻辑运算符操作整型变量中的单个位，学习怎样用结构中的位字段来把数据存储的有效性提高到最大程度。

目　录

第1章　C语言程序设计介绍

本章主题
- C简史
- 为什么C是程序设计语言中最好的选择
- 解决问题的技巧
- 程序开发周期中的步骤
- 怎样编写、编译并运行第一个C程序
- 编译器和连接器所产生的错误信息

1.1　C简史

C是Dennis Ritchie于1972年在贝尔电话实验室创建的，采用C来开发应用软件比采用当时可用的其他计算机语言更为有效。当时可用的一种语言是B，也是在贝尔实验室创建的，创建人为Ken Thompson。因此，用C来命名这种新的语言。

C的强大功能与灵活性使贝尔实验室的开发人员创建了诸如UNIX操作系统这样的复杂软件。这种新的语言迅速地从贝尔实验室传播开来，到处都有程序员在尝试用C编写简单的或复杂的程序。

然而，C的广泛应用在各开发团体之间出现了问题。各种机构都修饰Ritchie的成果而创建自己的C版本。某些执行过程的细微差别不时地引起C程序之间的不兼容。为解决这个问题，美国国家标准化协会（ANSI）在1983年形成了一个建立C的标准定义的协议，即众所周知的ANSI标准C。几乎无例外地，每一种现代C编译器都遵守这个标准，再强化语言的功能。

1.2　为什么C是程序设计语言中最好的选择

今天的程序员可以在各种不同的高级语言，包括C、Pascal、BASIC以及Modular中进行选择。这些优秀的语言适用于大多数程序设计任务，但计算机专业人员认为C具备其他语言所不具备的优点：

- C是一种功能强而且灵活的语言。C语言能完成用户想得到的任何任务；语言本身不对用户施加限制。C语言经常用于创建各种操作系统、字处理器、图形软件、电子表格，甚至其他语言的编译器。
- C是一种流行的、专业程序员优先选择的语言。正因为如此，可以选用的编译器和有用附件的种类也十分繁多。
- C是一种可移植的语言。“可移植”意味着为一个计算机系统（如IBM PC）所写的程序，能不加修改或略加修改就可以在另一个系统（如DEC VAX系统）上编译并运行。C的ANSI标准强化了C的可移植性，这个标准公布了一个C编译器的公共规则集。

·C 只包括少数单词，称为关键字，语言的函数机制就是由这些关键字构成的。一个语言即使有再多的关键字（有时称为保留字）也未必能比 C 功能更强，相对来讲，C 用较少的关键字，可以编程序来完成任何任务。

·C 是模块化的。C 代码能够也应该写成称为函数的例行程序，这些函数能在其他的程序或应用软件中重复使用。

这些性能使 C 成为程序设计语言的最佳首选。另外，学习 C 也为掌握 C++（读做 C 加加）打下基础。C++是一种包含 C 语言的面向对象程序设计语言。C 中所用的机制可直接用于 C++。

1.3 解决问题的技巧

编写计算机程序是为了解决问题。因此，解决任何问题的步骤与写程序的步骤是相同的。

·定义问题。

·制定解决问题的方案。

·实现这个方案。

·测试执行结果，以判断问题是否解决。

若以 C 来建立一个程序，应遵循以下步骤：

1. 确定程序的目标。

2. 确定程序中所用的解决问题的方法。

3. 用 C 语言将这种算法转换成计算机程序。

4. 运行并测试程序。

例如，编写一个数据库程序就是一个目标。更简单的一个目标的例子是将一个名字在屏幕上显示出来。

在写程序时，要确定所采用的方法，要搞清楚是否要编写程序来解决这个问题？需要跟踪什么信息？会用到哪些公式？在这一步中，必须确定该方法中各组成部分的实现顺序。例 1-1 是这种方法的一个说明。

例 1-1

1. 假定要编写程序来求一个圆的面积，这是该程序的目标。

2. 确定怎样计算圆的面积以及用户应提供哪些信息来帮助程序解决问题。本例中，用户应按提问输入圆的半径。

3. 给出这些信息后，即可将公式 πR 转换成 C 语言，建立一个计算圆面积的程序。

1.4 程序开发周期中的步骤

程序开发周期有其自身的步骤。第一步，用一种编辑器建立一个包含源代码的磁盘文件；第二步，编译源代码生成一个目标文件；第三步，连接目标文件生成一个可执行文件；第四步，运行程序以确定该程序是否能按方案工作。

1.4.1 用编辑器输入源代码

源代码是用于指示计算机完成特定任务的语句或命令的序列。如上所述，程序开发周期中的第一步是将源代码输入编辑器。例 1-2 是一行已输入编辑器的 C 源代码，该行指示计算机在屏幕上显示信息 Hello，Mom!。

例1-2　用于在屏幕上显示文本的一个语句。

```
printf('Hello,Mom!");
```

注：本章及其他各章可能会出现还未解释过的语法，但这些语句的作用在例子中应该能够理解，如果还不明白，可参看书末的索引。

将源代码文件存盘时要给文件取一个名字。源代码文件的名字与扩展名都可任取，但名字最好能描述程序的目的。文件扩展名用 .C，.C 是推荐的最合适的扩展名。

有些编译器带有编辑器，可用来输入源代码，有些则不能。查阅编译器手册即可知是否带有编辑器。如果没有，可用其他的编辑器。

大多数计算机系统都带有一个可用做编辑器的程序。例如，UNIX 系统有各种可用的编辑器，如 ed、ex、edit、emacs 和 vi。Microsoft Windows 提供了 Notepad。MS-DOS 5.0 及后续版本有 edit 可用。5.0 以前的 DOS 版本中，可以用 edlin。PC/DOS 6.0 及后续版本带有 E 编辑器。在 OS/2 中，可在 E 与 EPM 编辑器中选择一种。

大多数字处理器用专门的代码使文档格式化。这些代码不能被其他程序正确读入。美国标准信息交换码（ASCII）规定了一种几乎可为任何程序包括 C，能够使用的标准文本格式，许多字处理器，如 WordPerfect、AmiPro、Word 、WordStar 都能将源文件存作 ASCII 格式，即作为文本文件而不是文档文件。这种文件要用 ASCII 或 text 选择项来存储。

注：本书包容 ANSI 标准 C。用户可使用任意一种遵循 ANSI 标准的 C 编译器。

1.4.2　使用编译器

C 源代码用类似于英文单词的形式书写，便于阅读和理解。可惜的是，计算机不能辨认这些单词。计算机需要的是机器语言形式的数字式（或二进制）指令。C 程序在计算机上运行之前，必须从源代码翻译成机器语言。这种翻译，即程序设计的第二步，是由一种称为编译器的程序来实现的。

编译器将源代码文件作为输入并产生一个包含与源代码语句相对应的机器语言指令的磁盘文件。编译器所生成的机器语言指令称为目标代码，包含目标代码的磁盘文件称为目标文件。

每一种编译器都需要为其提供生成目标代码的命令。编译时，用这个命令来运行编译器，后跟源文件名。表 1-1 是用各种不同的 DOS 编译器来编译一个名为 radius. c 的源程序的例子。

表 1-1　编译一个 C 程序的典型命令

编 译 器	命　令	编 译 器	命　令
Microsoft C	cl radius. c	Borland C	bcc radius. c
Borland's Turbo C	tcc radius. c	Zortec C	ztc radius. c

如果在 UNIX 机器上编译 radius. c，则用例 1-3 中的命令。可查阅编译手册来确定该编译器的正确命令。

例 1-3　在 UNIX 中编译一个 C 程序的正确命令

```
cc radius.c
```

编译之后，生成一个与源文件同名而扩展名为.obj(不是.c)的目标文件。扩展名为.obj 表

示是一个目标文件，供连接器使用。在 UNIX 系统中，编译器生成带有扩展名 .o 而不是.obj 的目标文件。

1.4.3 生成可执行文件

在程序可执行之前还需要一个步骤。C 语言要包括一个函数库，函数库中含有预定义函数的目标代码，即已经编译的代码。预定义函数包含已编写好并由编译软件包提供的备用形式的 C 代码。例 1-2 中的 printf（）函数就是一个库函数。

库函数完成那些调用频繁的任务，例如在屏幕上显示信息、从磁盘文件中读数据等等。由源代码经编译产生的目标文件与函数库中的库函数目标代码相结合才产生最终的可执行程序，即可上机运行的程序。这个过程称为连接，由称为连接器的程序来完成。

图 1-1 是从源代码到目标代码再到可执行程序的过程。

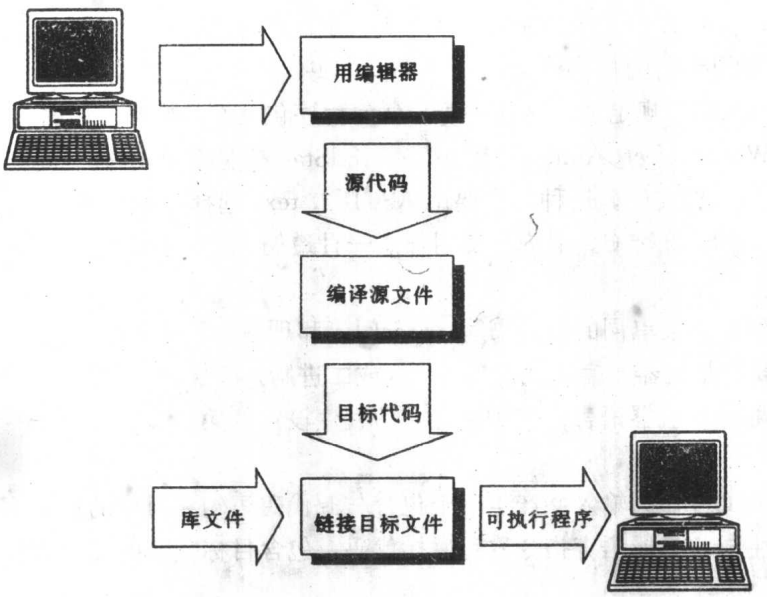

图 1-1 C 程序开发周期

第一步：用编辑器写出源代码。按惯例，C源代码的扩展名为c，例如，可以取名为myprog. c。

第二步：用编译器编译该程序。如果编译器未在程序中发现错误，便产生一个与源代码文件同名但扩展名为. obj 的目标文件，如由 myprog. c 产生 myprog. obj。如果编译器发现了错误，将报告错误，这些错误在源代码中更正。

第三步：用连接器连接该程序。如果不出现错误，连接器将产生一个位于磁盘文件的可执行程序，该文件与目标文件同名但扩展名为. exe。如 myprog. obj 连接生成 myprog. exe。

第四步：运行并测试该程序，确定程序功能是否正确。如果不正确，再从第一步开始，并对源代码进行修改或添加内容。

1.4.4 运行程序

程序经编译与连接而生成一个可执行文件后，便可在系统提示符处输入该程序名使其运行。如果该程序功能有误，说明源代码有问题。在这种情况下，更正源代码后再编译与连接该程序产生一个校正了的可执行文件。重复这个过程直到程序能正确执行为止。

即使是很熟练的程序员也很难一次写出没有错误的程序！专业人员常要将编辑－编译－

连接—测试这样的过程重复多次。

注：这里将编译与连接描述为独立的步骤，但许多编译器，如前述的 DOS 编译器，是在一步中完成这两种动作的。无论编译与连接以什么方法来完成，即使是用一个命令来执行，也要理解为两种独立的动作。

1.5 怎样编写、编译并运行第一个 C 程序

本节中，将通过构造一个样板程序来学会编写、编译与连接 C 程序。在本节中，只要求将样板源代码复制到编辑器中并产生可执行程序，而不必理解程序中包含的所有内容。

样板程序名为 hello.c，它将 Hello，World! 字样显示在屏幕上。源代码如列表 1-1 所示。

列表 1-1 hello.c

```
1: #include <stdio.h>
2: void main()
3: {
4:     printf("Hello, World!");
5: }
```

安装好编译器。确定是用 UNIX、DOS 还是其他操作系统，这要看操作者了解哪种编译器和编辑器。一旦编译器和编辑器准备好，按照输入、编译与执行 hello. c 这样的步骤来进行操作。

输入和编译 hello.c 程序的步骤如下：

1. 使存放 C 程序的目录成为当前目录并启动编辑器。如上所述，可使用任何一种文本编辑器，但大多数 C 编辑器，如 Borland 的 Turbo C++和 Microsoft 的 Visual C/C++，都带有集成开发环境（IDE），能够以一种方便的方式来输入、编译与连接。查看编译器手册看所用编译器是否具备 IDE。

2. 在键盘上敲入列表 1-1 所示的 hello.c 源代码，代码的每一行结束时按回车键。

注：输入本书中的程序行时，左边不要打行号。这些行号是为叙述方便而加的，不是程序的组成部分。

3. 保存源代码。文件名取为 hello.c。

4. 列出该目录中的文件，查看盘上有无 hello.c。

5. 编译并连接 hello.c。执行编译手册中指定的适当的命令。编译器将显示一条说明无错误或警告的信息。

6. 检查编译器信息。如果无错误或警告信息显示，程序将完成编译。注意，编译器会捕捉输入时的错误并在屏幕上显示相应错误信息。假定单词 printf 错拼为 prntf，则会出现如下信息：

Error: undefined symbols: _prntf in hello.c (hello. obj)

7. 如果出现错误信息，返回到第 2 步。在 editor 中打开 hello. c 文件。仔细地对照列表 1-1 中的文件内容，进行必要的更正，然后继续第三步。

8. 至此，这个 C 程序可以编译并准备运行。显示目录并列出名为 hello（扩展名任意）的所有文件。屏幕上将显示如下信息：

hello. c

hello. obj or hello.

hello. exe

9. 要运行 hello. exe，输入 hello 即可。Hello，World！将在屏幕上显示。

1.6 编译器和连接器所产生的错误信息

编译性错误是当编译器在源代码中遇到不能编译的情况时出现的。拼写错误或其他的印刷错误都可能引起编译性错误。了解这一点便于发现并更正源代码中的错误。

为了理解怎样发现并更正源代码中的错误，可故意在 hello. c 中引入一个错误：在 editor 中，将光标移到包含 printf()调用的行尾，删去行尾的分号，则改过的 hello. c 的代码如列表 1-2 所示。

列表 1-2 有错误的 hello. c

```
1: #include <stdio. h>
2: void main()
3: {
4:     printf("Hello, World!")
5: }
```

保存这个文件并输入相应编译器的编译命令来编译该程序。由于引入了错误，编译不能完成，编译器将在屏幕上显示如下信息：

hello. c(5) : Error: ';' expected

由这一行可看出，这条错误信息有三部分：

①hello. c　　错误所在的文件名

②(5) :　　错误所在的行号

③Error: ';' expected　　错误描述

这条信息是很有用处的。它表明了在 hello. c 源代码文件的第 5 行，编译器要找的分号没找到。注意分号是第 4 行遗漏的而不是编译器所报告的第 5 行。出现这种混乱是因为编译器无法顾及诸如分行这样的问题。属于 printf () 语句的分号也可放在下一行上。只有当找到第 5 行的大括号后，编译器才确认遗漏了分号，因此，报告第 5 行有错。

虽然 C 编译器检测与定位错误是很灵敏的，但也无法准确地分辨源代码中的每一种错误。因此，程序员要分析编译器的信息并确定所报告错误的实际位置。通常可在编译器所报告的行找到错误，如果不能定位，一般可在前一行中找到。

注：给出的错误信息可因编译器不同而不同。大多数情况下，错误信息指出是什么问题或问题出在什么地方。

结束本章之前，再看一下另一种编译错误。将 hello. c 再调入编辑器并作如下变动：

①将分号放回到第 4 行末。

②删去 Hello 之前的双引号。

将该文件存盘并再次编译。这一次，编译器将显示如下错误信息：

hello. c(4) : Error: undefined identifier 'Hello'

hello.c(5) : Lexical error: unterminated string

Lexical error: unterminated string

Lexical error: unterminated string

Fatal error: premature end of source file

第 1 条信息准确地发现了错误,定位于第 4 行的 Hello。错误信息"undefined identifier"的出现是由于单词 Hello 未在引号中而使编译器不能确定其作用。那么,所报告的其他 4 种错误因何而产生呢?这 4 种错误的存在表明 C 程序中的一个错误可能会导致产生多条错误信息。

由本例可知,如果编译器报告了多个错误,可先更正第一个错误并重新编译。更正一处错误可能会解决所有问题,使程序可无错误地编译。

连接器错误相对较少一些,通常是由拼错 C 的库函数名而引起的。这时,屏幕上将显示 Error: undefined symbols:,后跟一条下划线和错拼的名字。更正了拼写错误之后,问题就解决了。

1.7 小结

C 集中了功能强、普及性与可移植性等无与伦比的优点。这些因素加上 C 与新的面向对象语言 C++的密切关系,使 C 立于不败之地。

本章解释了编写一个 C 程序,即一般所说的程序开发的各个步骤。应清楚地理解编辑—编译—连接—测试这样一个过程以及每一步所用的工具。

出错在程序开发中是不可避免的。C 编译器测试源代码中的错误,显示错误信息,并给出错误性质与位置。可利用这种信息来编辑源代码以更正错误。应该了解,编译器并不总能准确地报告错误的性质与位置,有时需要用户以自己的 C 知识来查出引发错误的原因。

1.8 练习

1. 复习题

(1) 如果要与另一个程序员共享一个程序,应该把哪个文件送给他呢?

(2) 生成一个可执行文件后,还有必要保存源文件 (.c) 或目标文件 (.obj) 吗?

(3) 必须使用编译器自带的编辑器吗?

(4) 警告信息可以忽略吗?

答案

(1)有幸的是 C 属于编译型语言。这表明在源代码编译之后将产生一个可执行程序。这是一种独立的程序,要把程序分给他人,只要把这个可执行程序,如 hello.exe 给他即可。而源文件(hello.c)与目标文件(hello.obj)不必给。

(2) 如果源文件被毁坏,将来就没有办法对该程序进行修改了。因此,应保存源文件。而目标文件却不同,尽管它也有许多保存的理由,但这超出了本章的范围。至此,在生成可执行文件之后,即可删去目标文件。必要时,编译源文件即可再生成目标文件。

(3) 不一定。任何能以文本格式存储源代码的编辑器都可使用。应尽量使用编译器自带的编辑器。

(4) 有些警告信息不影响程序运行,有些会影响。如果编译器显示一条警告信息,只是发出一种提示某些内容并不十分准确的警告信号。大多数编译器允许用户设置警告级,以决

定是只显示最严重的警告还是连最细小的警告也一起显示。有些编译器给出这些警告级的选项。用户应检查每一个警告，确定是否需要更改源代码。当然，最好还是尽力写出既无警告又无错误的程序。如果程序有错误，编译器不会产生可执行文件。

2. 思考题

思考题中提出一些问题帮助理解讲过的内容，应在学习下一章之前回答这些问题。

(1) 为什么C是程序设计语言中最好的选择？给出三种理由。

(2) 编译器的作用是什么？

(3) 程序开发周期中的步骤有哪些？

(4) 用某种编译器编译一个名为program1. c的程序，要键入什么命令？

(5) 编译器可否以一条命令来执行编译与连接两种操作，还是必须分别键入命令？

(6) C源文件用什么扩展名？

(7) filename. txt是一个有效的C源文件名吗？

(8) 如果执行一个已编译的程序但不能像预期的那样工作，该怎么办？

(9) 什么是机器语言？

(10) 连接器的作用是什么？

3. 实习题

本习题让读者练习在编辑器中输入源程序然后编译源程序。所出现的某些语法可能使读者感到陌生，但在以后各章中会作解释。

(1) 用文本编辑器查看由列表1-1建立的目标文件。目标文件与源文件相像吗？(退出编辑器后不要保存这个文件)

(2) 输入并编译以下程序，该程序的作用是什么？

```
1: #include <stdio.h>
2: int main()
3: {
4:     int radius,area;
5:     printf("Enter radius (i.e. 10):");
6:     scanf( "%d", &radius );
7:     area = 3.14159 * radius * radius;
8:     printf( "\n\nArea = %d",area );
9:     return 0;
10: }
```

(3) 输入并编译以下程序，该程序的作用是什么？

```
1: #include <stdio.h>
2: int main()
3: {
4:     int x,y;
5:     for ( x=0; x<10; x++, printf( "\n" ))
6:     for ( y=0; y<10; y++ )
7:         printf( "x" );
```