21世纪高等职业教育电子信息类规划教材

国家示范校重点专业建设成果教材

C程序设计实践教程

于京 吴振宇 编著

人民邮电出版社
POSTS & TELECOM PRESS

21世纪高等职业教育电子信息类规划教材

国家示范校重点专业建设成果教材

C程序设计实践教程

于京 吴振宇 编著

人民邮电出版社

北京

图书在版编目（CIP）数据

C程序设计实践教程 / 于京，吴振宇 编著. -- 北京 : 人民邮电出版社，2015.4
21世纪高等职业教育电子信息类规划教材
ISBN 978-7-115-35514-0

Ⅰ. ①C… Ⅱ. ①于… ②吴… Ⅲ. ①C语言－程序设计－高等职业教育－教材 Ⅳ. ①TP312

中国版本图书馆CIP数据核字(2014)第268065号

内 容 提 要

C 语言是一门经典的程序设计语言，具有简洁、高效、功能强大的特点，因此至今仍然被广泛应用。由于历史传承的原因，很多新的计算机语言也是类 C 的，所以无论从实际使用还是从学习计算机程序设计角度考虑，学习 C 语言都是很好的选择。

为了更好地引导读者学习 C 语言，本书作者总结多年的编程和教学经验，特别设计了一个贴近编程实际需求的全新组织结构，原创了大量有实际指导意义和说明编程技巧的案例，对 C 语言程序设计的方法进行了实例讲解，相信这样可以提高读者的学习效率。

本书可以用作 C 语言以及程序设计基础类教材，供职业技术院校师生和希望自学 C 语言程序设计的读者参考阅读。

◆ 编 著 于 京 吴振宇
责任编辑 王朝辉
责任印制 彭志环

◆ 人民邮电出版社出版发行 北京市丰台区成寿寺路 11 号
邮编 100164 电子邮件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
北京昌平百善印刷厂印刷

◆ 开本：787×1092 1/16
印张：12.25
字数：314 千字 2015 年 4 月第 1 版
印数：1-3 000 册 2015 年 4 月北京第 1 次印刷

定价：29.00 元

读者服务热线：(010)81055410 印装质量热线：(010)81055316
反盗版热线：(010)81055315
广告经营许可证： 京崇工商广字第 0021 号

前言

本书旨在用最短的篇幅引导读者学会 C 语言编程。虽然目前市面上已经有各种教材指导读者学习这门编程语言，但是本书的编写方式和学习路径安排与其他教材相比还是有鲜明特点的。第一，本书专为从零起步学习用 C 语言编程的读者设计，因为 C 语言的内涵太丰富了，如果全覆盖不太适合初学者，所以本书的讨论范畴并未覆盖 C 语言的全部细节。第二，本书对有些内容——例如程序的结构、指针与函数的应用方法（注意：是应用方法而不是使用格式）——讨论得比较深入，这有利于读者编写出一个高质量的程序。第三，本书的章节安排有特色，比如在第 1 章就教读者利用函数编写程序，在介绍数组的同时也介绍了结构体，这样编排的目的是促使读者从需求和程序架构的角度看待这些零散的知识，而不是从“字典”的角度。第四，如果参照“字典式”的教材的提法，本书更像“用法字典”而不是“释义字典”。第五，本书未采用说明语法问题的“宇宙通用型”语法例程（这会让读者摸不到头脑），而是希望通过一些原创例程和处理方法（比如本书专业的排序架构，虽然方法还是比较初级的“选择法”，但是架构非常实用），使读者从一开始编程时就适应和养成比较专业与正规的思路。

本书通过一系列有实际意义的案例将 C 语言的语法知识和技术要点变为解决问题的工具；并且为了使读者抓住学习的重点，不至于在纷杂的内容中失去方向，作者特别将一些知识的细节做了“屏蔽”，这样安排有利于让读者尽快掌握编程的主线。根据作者的经验，C 语言的所有细枝末节的语法应用就像一座仓库，读者应当先知道自己要什么（先会编程），再去仓库找物料（运用细节的语法），反其道而行之实乃舍本逐末。相信读者不会只用一本书去掌握 C 语言的全部细节，所以本书不涉及的内容大可从其他书里查找。本书的立意就像开篇所说的，“用最短的篇幅引导读者学会 C 语言编程”，再奢望一下，“希望用最短的篇幅让读者学会编程”。其实“学”会编程只有几步，这在本书的目录中也有体现。

“学”编程必须“学”的内容	本书的内容
输出	printf
输入	scanf
划分功能块并解决重复性劳动	函数和循环
决定程序流程	if 和 switch
处理大量数据	数组
开发自定义类型	结构体

只要学会这些，就应该说“初步学会编程了”。对其他编程语言也是类似。至于多出来的内容，是为了使读者能够更高效地解决问题，毕竟“会”和“好”之间还有很大差距。而本书保持较短篇

幅的用意就是让读者从零到比较好的路程尽量短。

本书由以下几部分组成。

正文描述：特点是用一些篇幅介绍 C 语言的某些语法到底要满足什么编程需求，希望读者一定读完。

例程：除经典例程外，本书增加了许多有实际意义的原创例程供大家参考。

随堂练习：检验知识点的掌握情况，一定要读到哪里测到哪里，这样可以保持自信地继续学习。

本章小结：总结了每章的知识点，但是这里的细节描述可能比正文描述还要复杂，因为这里不需要考虑这些细节是否打断了读者的编程思路。

练习：希望大家都做一做较切合编程实际的练习。

附录：并不是可有可无的内容，它提供了一个有用的查询表。

最后还有 3 个问题需要说明一下。

第一，本书可以作为职业技术院校 C 语言教材和学习 C 语言编程的读者的自学教材。

第二，本书介绍的是 C 语言，而不是某种 C 语言，故本书使用 Windows、Linux、Mac OS X 3 类系统环境进行程序调试，这么做是为了告诉读者 C 语言是基本独立于平台的。

第三，本书为国家示范性职业院校建设资助项目，其中，于京编写第 2 章至第 9 章，吴振宇编写第 1 章、第 4 章的 4.4 节、第 5 章的 5.1～5.3 节、第 8 章的 8.1 节、第 10 章，于京负责全书统稿。在此还要感谢祝智敏、陈明、胡亦、詹晓东、路远、安宁、曹艳芬、李佳睿、江为等在本书的编写过程中给予的帮助。

作者

目录

第5章 功能完善的月历

第6章 利用二维数组和结构体处理复杂的表格

第7章 函数与数组的综合运用

第8章 利用指针提高编程效率

第9章 利用链表处理复杂表格

第10章 文件操作

附录 基本语法总结

第1章 课题的提出：打印月历

学习一门新程序设计语言的快速途径就是使用它编写程序。对于所有语言的初学者来说，接触的第一个程序几乎都是相同的，这个程序就是“Hello world!”。本章首先通过这个众所周知的程序开始整本书的讲述，然后通过一系列输入/输出的例程让读者熟悉 C 语言中输入/输出的规则以及知道如何编写一个完整的程序。

1.1 从“Hello world!”开始

“Hello world!”程序首先需要创建一个源文件，读者可以通过文本编辑器创建。在该文件中，依次输入 C 语言代码，如例程 1-1 所示。最后，将该文件保存为以.c 为后缀的文件，比如 hello.c。包含 C 代码的源文件也称为文本文件。这样我们就编辑好了一个最简单的 C 语言程序。

```
1    /*第 1 个 C 程序*/
2    #include <stdio.h>
3    #include <stdlib.h>
4    int main()
5    {
6        printf("Hello world!\n");
7        return 0;
8    }
```

例程 1-1　第 1 个 C 程序

运行上面的例程 1-1 后，将在屏幕上出现一行文字：“Hello world!”。这个程序的功能十分简单，只是给用户打印一句话。通过阅读代码，我们能够猜出来程序中最核心的语句应该是位于第 6 行的：

```
printf("Hello world!\n");
```

那么，它到底应该如何使用，代码中的其他内容又是什么意思呢？下面我们就从第 1 行开始解析这个程序。

1.1.1 为程序作注释

程序的第 1 行：/*第 1 个 C 程序*/，是用一个“/*”和一个“*/”这两个对称的符号包括的一些文字，这是标准 C 语言的固定写法，表示本行文字是个注释。注释的内容不参与程序的运行，它

的主要作用是向代码的阅读者做出一些解释，比如程序的主要功能、代码的编辑日志等。另外，注释可以跨越多行，例如：

```
/*
这是
多行
注释
*/
```

在 C99 标准中有一种新的注释方式“//”。目前，有一些新的编译器支持 C99 标准，可以使用这种注释方式，例如：

```
//这是一行注释
```

这种用两条斜线开头的注释方法只用于单行注释，由于注释标记“//”位于注释文字的前面，使用起来比较简单。

评价一个程序好坏的标准很多，首先是程序的源代码是否清晰易懂，其次才是程序运行的结果如何。一个结果正确但是源代码不能被别人读懂的程序是“一次性”的，很难被再利用。相反，一个可以被别人读懂的程序不论结果如何总会为其他人提供一些借鉴，而让别人读懂程序的关键之一就是勤加注释。一般来说，每个源代码的开始部分都应该有一个注释，用来说明该程序的编写目标、编写日期、作者、大体结构等。程序中的注释占到整个程序总行数的 1/4～1/3。

1.1.2 include 的作用

自从 C 语言出现以后，很多优秀的开发者使用 C 语言开发了大量的程序。经过不断的测试、完善，程序变得越来越稳健。将成熟、稳健的程序作为“函数”放在“库”中，当一个开发者需要实现相同功能的时候，就不必再重新编写代码，可以直接使用已有的代码，这样就方便了代码重用。

“include”是 C 语言的关键字之一，它提供了一种机制，能够让编程者使用其他人的成果，或者说是“站在巨人的肩膀上”。读者也可以这样设想，一门计算机语言通常会提供一系列“工具”以方便大家编程，那么如何使用这些“工具”呢？在 C 语言中将这些“工具”划分为一系列“库”，每个“库”通过头文件的形式提供给编程者使用。使用方法就是通过“include”关键字将需要使用的头文件包含进编程者自己编写的程序，比如例程 1-1 中的第 2 行：

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

这一行的含义是，本程序需要使用输出功能，因此包含标准输入/输出头文件（stdio.h 文件中包含许多输入/输出工具）。注意：头文件的文件名用尖括号“<>”括起来。这里给一个小提示，有时候我们会看到类似这样的头文件包含：

```
#include "stdio.h"
```

与给定例程 1-1 中的 include 不同的是这个文件 include 使用了双引号，这个双引号与尖括号的区别是尖括号只在指定目录搜索文件包含的目标，而双引号先搜索当前目录再搜索指定目录，简而言之，双引号的搜索范围更大。

1.1.3 main、函数与函数的组成部分

显然，main 的含义是“主要的”，“()”括号的含义是“函数”（数学中就是这样使用和表示的），连起来 main( )就是主函数的意思［本书正文为叙述方便，省去“( )”］。顾名思义，这个函数非常重要，因为 C 语言的语法规定（语法也是法，即事先指定的一系列规则），如果一个程序需要运行（真的有不需要运行的程序），那么程序中必须有且只有一个主函数。

另外，主函数也是函数，读者可以查看例程 1-1 中的第 4 行：

```
int main( )
```

我们发现在 main 前面还有一个词：int。这是一个缩写词（全称是 integer），含义是“整型数”。如此看来 int main( )的含义是：定义一个主函数，主函数的类型是整型。函数的类型是指函数返回值的类型。在这里，这个值是整型的。为了加深理解，回想一下在数学中经常会遇到这样一个函数：

```
y=f(x);
```

上面的函数中，会有一个计算结果，这个结果是 *y*。在 C 语言中我们把这个结果 *y* 叫返回值。接下来看例程 1-1 中的第 7 行：

```
return 0;
```

第 7 行是一个语句。语句最明显的标志是后面的分号。在 C 语言中语句结束的标志就是分号，就如同在写文章的时候结束一句话需要使用句号一样。

“return 0;”的含义十分容易理解，返回一个值 0。在第 4 行定义了函数应该有一个整型返回值，第 7 行说明了这个返回值是 0，这是一个“应答关系”。一般情况下，函数都应该有一个返回值，当然，C 语言中允许出现无返回值的函数，例如：

```
void fun( );
```

上面的 fun 就是无返回值的函数，但是函数的返回值在错误检测等多方面都是有优势的，在以后的章节中我们会讨论关于函数返回值的问题。

第 5 行和第 8 行是一对大括号“{ }”，它们的意思很简单，即函数的边界标志，函数从“{”处开始至“}”处结束。

1.2 利用 printf 输出

之前已经提到，例程 1-1 中的第 6 行“printf("Hello world!\n");”是核心部分。这是一个格式化输出函数的调用语句，它使用（术语称为“调用”）了标准输出函数 printf，将一个字符串输出到标准设备（屏幕）上。它可以带一个字符串参数，其结果就是在屏幕上输出这个字符串，第 6 行的输出是：

```
Hello world!（换行符）
```

之所以换行是因为字符串的最后有一个“\n”，它的意思是“newline”（换行）。这种斜杠加一个字符的形式被称为“转义字符”。转义字符是不可见的字符，不可见字符不会显示于屏幕，但通常用来表示一些格式。表 1-1 列出了一些常见的转义字符。

表 1-1　　一些常见的转义字符

转 义 字 符	含　义
\n	换行符
\t	水平换行符
\b	回退符
\"	双引号被 C 语言用来作为字符串边界标志，所以如果字符串内部出现双引号，则用\"来表示
\'	单引号被 C 语言用来作为字符边界标志，所以如果需要单个单引号字符，则用\'来表示
\\	正常输出一个斜杠

另外，printf 是一个系统函数，专门用来进行格式化输出，但是我们并没有编写代码实现这个函数，那为什么能使用这个函数呢？原因在于之前介绍的“#include<stdio.h>”，包含头文件“<stdio.h>”中的一系列标准输入/输出函数，其中包括 printf。

现在利用 printf 进行说明。通过“Hello world!”的例子我们知道，printf 可以打印一个字符到屏幕上，如果需要换行我们可以使用 printf 打印一个换行字符“\n”。那么，如果想打印形如：“日　一　二　三　四　五　六”的月历标题应该怎么做呢？答案是一系列打印，如例程 1-2 所示。

```
1    /*打印月历*/
2    #include<stdio.h>
3    #include<stdlib.h>
4    int main() {
5        printf("日");
6        printf(" "); //打印一个空格
7        printf("一");
8        printf(" ");
9        printf("二");
10       printf(" ");
11       ……..
12       printf("六");
13       printf(" ");
14       printf("\n");
15       ……
16       return 0;
17   }
```

例程 1-2　打印月历的标题

这样打印，可以打印出来月历标题的效果，但这样使用 printf 很麻烦，应将所有 printf 合成为一个：printf（“日　一　二　三　四　五　六”）;。

1.3 利用函数可以简化编程

从上一节的例程 1-2 中可以发现，打印一行，就已经要 printf 十几次了。这样打印一个月历，就会打印几十次，整个源文件里全部是 printf 的重复了。这种程序不仅让人眼花缭乱，而且可读性差。那么有没有比较好的办法呢？

通过分析例程 1-2 我们发现，这个月历一共有 6 行。那么，我们可以一行行地进行打印，这样看起来结构更加清晰、简单，如例程 1-3 所示。

读者不仅可以调用 C 标准库提供的函数，也可以定义自己的函数。事实上定义自己的函数并不是新内容，比如例子中定义的 main 函数。main 函数的特殊之处在于执行程序时它自动被操作系统调用，操作系统就认准了 main 这个名字，除了名字特殊之外，main 函数和别的函数没有区别。

看看这次的 main 函数，它里面调用了我们定义的 6 个函数。其中，printTitle、printLine1、printLine2、printLine3、printLine4、printLine5 都是我们起的名字，这些名字都能反映出来这个函数要做的事情。比如 printTitle 就是打印一个标题，printLine1 就是打印第 1 行。而在源文件的其他部

分，需要有这 6 个函数的实现，为了节省空间我们只是示例了 printTitle。同时，我们定义了一个函数——printfSpace，用于打印空格，这样在函数中如果需要打印一个空格，就可以直接调用 printfSpace。

```
1    /*打印月历*/
2    #include<stdio.h>
3    #include<stdlib.h>
4    void printfSpace(){
5        printf(" "); //打印一个空格
6    }
7    void printTitle(){
8        printf("日");
9        printfSpace();
10       printf("一");
11       printfSpace();
12       printf("二");
13       printfSpace();
14       ……...
15       printf("六");
16       printfSpace();
17       printf("\n");
18   }
19   int main() {
20       printTitle();
21       printLine1();
22       printLine2();
23       printLine3();
24       printLine4();
25       printLine5();
26       return 0;
27   }
```

例程 1-3　使用函数简化编程

这里，读者可以体会一下函数的最主要作用，即组织代码，提高代码的可读性，就跟我们写文章时，通过分段来提高文章的可读性一样。函数如同文章中的段落，每个函数实现一个基本的功能。然后通过函数之间的调用关系实现一个有意义的大的功能。读代码和读文章不一样，按从上到下、从左到右的顺序读代码未必是最好的，代码也有它的阅读顺序。比如上面的例子，按源文件的顺序应该是先看 printfSpace，再看 printTitle，然后看 main。但是，这种阅读方法，不能让我们知道程序要实现的整个功能。如果换一个角度，按代码的执行顺序来读也许会更好：首先执行的是 main 函数中的语句，在 printTitle 之后调用了 printLine1、printLine2 等，知道这是在打印一个标题和 6 行。这时再去看 printTitle 的定义，其中又调用了 printfSpace，这时再去看 printfSpace 的定义，里面有一条 printf。这样就是一个比较有意义的阅读顺序。

有些读者或许会问，为什么在上面我们说定义了函数？在 C 语言中，有两个与函数有关的很重要的概念，即“声明”和“定义”。“声明”只是告诉编译器有这样一个函数，不一定有其具体实现。例如：

```
void printfTitle();
```

这种写法只能叫函数“声明”，而不能叫函数“定义”，只有带函数体的“声明”才叫“定义”。一般将“声明”放在头文件中，而将“定义”存放在.c 源文件中。

通过函数的方式，程序更具有可读性，同时更容易维护。比如，若要修改第 4 行打印的方法，那么直接在 printLine4 这个函数中进行修改。通过这个例子可以看到，函数可以让程序更加简洁。更具体的函数使用方法我们会在后面的章节中讲解。

1.4 程序的运行过程

前面在讲解“Hello world!”程序的时候提到，可以通过一个编译器将代码输入到一个以.c 为后缀的文件中。这只是刚刚完成了第 1 步。难道输入完这些代码之后，就可以得到我们需要的结果了吗？从文本代码到一个真正的应用程序之间是一个怎样的转换过程呢？本节我们作一个简单的介绍，如图 1-1 所示。

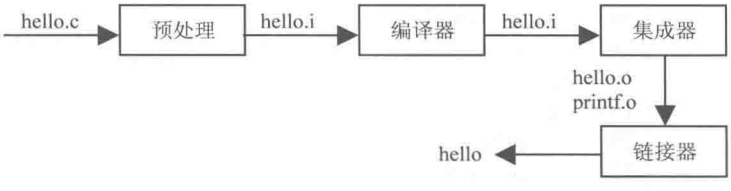

图 1-1　文本代码到应用程序的转换过程

一般来说，写完一个源文件之后要对其进行编译，编译器负责将一个源程序转换为可执行的程序。预处理负责处理源文件中的#include，比如在我们的例子中，会将两个头文件包含到程序文本中。然后进行编译，编译阶段会检查所写的代码中的语法错误。编译成功后生成一个目标文件，比如 hello.o。最后，因为应用程序里面使用的 printf 是在系统的库里的函数，编译器将这两个以.o 为后缀的文件链接之后生成可执行的程序“hello”。这里，“hello”就是我们最终得到的可执行程序。

许多厂商和组织为了让程序员的开发过程更便捷，提供了一系列开发工具。业界将把开发、编译、运行、源文件及项目管理功能组合为一个开发软件的开发工具称为“集成开发环境”，简称为 IDE。下面举一个应用 IDE 工具（DEV C++）的具体案例，来说明开发和运行一个 C 语言程序项目的过程。

步骤一：创建新项目。点击左上角菜单栏的“File→New→Project”，如图 1-2 所示。

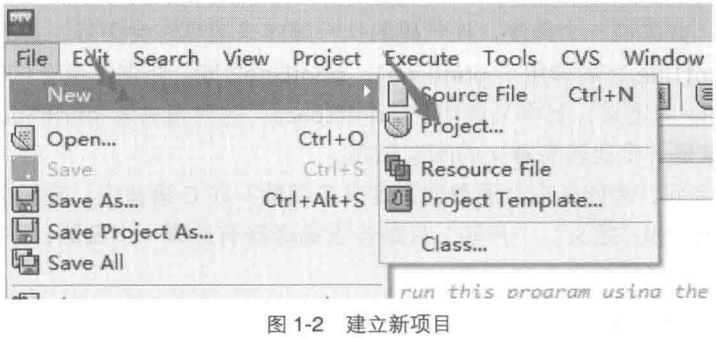

图 1-2　建立新项目

步骤二：编辑项目名称 HelloWorld，确定项目类型为 C 语言项目。这里以控制台输出为例，选择完后点击“OK”，如图 1-3 所示。

图 1-3　设定项目名称

步骤三：创建应用程序。右击项目名称，选择“New File”。右击创建好的新文件，选择“Rename file”。在弹出的对话框中编辑名称，如图 1-4 所示。

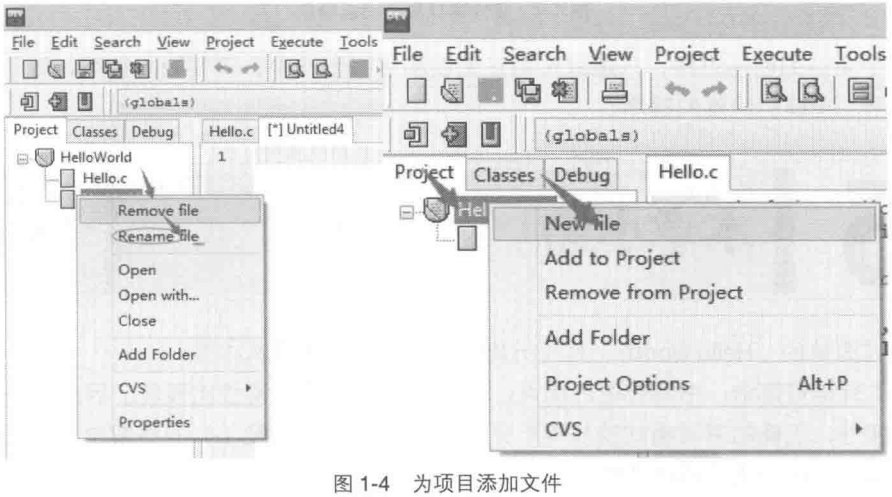

图 1-4　为项目添加文件

步骤四：打开文件，编辑源文件，如图 1-5 所示。

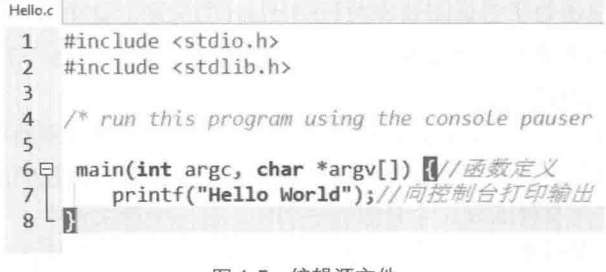

图 1-5　编辑源文件

步骤五：编译运行。先点击编译按钮编译程序，再点击运行按钮（具体位置见图 1-6）。