最新Microsoft QuickC

2.50版

使用大全

周杉 王忠
晓夏 武仁 编译

（共四册）

语言参考手册

2

中国科学院希望高级电脑技术公司

最新Microsoft QuickC
使 用 大 全

（V2.50版）

■ 安装和运行·成套工具（之一）
■ 语言参考手册（之二）
■ 库函数集锦（之三）
■ 库函数集锦（之四）

中国科学院希望高级电脑技术公司
一九九〇年十一月

目　录

引 言

自从 Microsoft 在 1987 年推出 QuickC C Compiler1.0 版以来，QuickC 用户要求更多地了解 C 程序设计语言。《语言参考手册》满足了这些要求，特别是满足了那些已有一定编程经验但还未学过 C 人们的要求。

§0.1 本书内容

《语言参考手册》假定你已具有一些编程经验，但还不熟悉 C。因此，本书不解释诸如为什么程序循环是有用的这种基本编程思想，而是认为你已一般地了解了循环，只是现在想来学习怎样用 C 语言实现它们。本书内容如下:

- 第一部分“学习 C”：包括了诸如函数及数据类型这样的基本问题。本部分中的诸章应按顺序从头到尾阅读。
- 第二部分“使用 C”：包括了诸如输入/输出和图形学这样的实际编程问题。本部分是按各个问题而组织的，所以你不必按照任何特定顺序来进行阅读。
- 附录 A“C 语言指南”：总结了 C 语言的 QuickC 实现。你可在读完第一部分并对 C 有些了解之后将此附录作为一快速参考。
- 附录 B“C 库指南”：总结了 QuickC 运行库中最常使用的函数。该附录主要是当你未在使用 QuickC 时作为浏览而用的。当在 Quickb 环境中，可使用 Microsoft QuickC Advisor(联机帮助)来得到有关 C 语言性质及运行库函数的信息。

注意，以后用术语“OS/2”来表示 OS/2 系统——Microsoft 操作系统/2(Ms OS/2)及 IBM OS/2。类似地，术语“DOS”既表示 MS-DOS，也表示 IBM DOS。只是在说明为某系统所独特的性质时才给出具体系统名。

§0.2 使用示例程序

本书中的示例程序可通过联机帮助而得到。这就使你可在阅读时装入、运行及尝试示例程序。

为装入一例子，必须使用 QuickC 环境:在 Help 菜单中选择 Contents，然后选择本书的标题。找到所希望的程序，然后用 QuickC 的 Copy 和 Paste 函数将它拷贝至编辑器中。

在将一程序拷贝至 QuickC Editor 中后，就可以将它看成是任何一个 C 的源程序，从而可对之编译或编辑，及将它存至磁盘等等。

QuickC 联机帮助包括了本书所有有意义的例子(它不包括代码段及一些很短的程序)。在联机帮助中的每个程序都以如下形式的一行语句开始:

```
/* POINTER.C: Demonstrate pointer basics. */
```

该行包括了程序名及程序功能的简单介绍。含此行的程序在联机帮助中被命名为 POINTER.C。

联机帮助中的所有例子编译时都不会在第三警告层上出错，QuickC 在此层上进行最严格的出错检查。在此层上，有些例子会产生下面这样无害的警告:

C4103: 'main' : function definition used as prototype

C4035: 'main' : no return value

C4051: data conversion

你可在一较低的警告层上进行编译以取消这些警告。

§0.3 本手册所使用的编程风格

C语言源代码的格式具有很大的灵活性。本书中所用的风格是为了使程序可读性好，但在编写自己的程序时却不必用此风格。下面是本书中对示例程序所用风格的概要：

- 每个语句或函数都单独成行。
- 变量名及函数名都用小写。诸如 **TRUE** 或 **FALSE** 这样的符号常量名用大写。
- 如果一函数无参数，那么函数名后的一中间无空格的开括号及闭括号前各有一空格：`printf( "Number = %i", num_penguins );`
- 诸如加和减这样的二元操作符前后都是一空格：`3 + 5`
- 如果用括号来控制操作符的优先级，在开括号后及闭括号前就不用空格：`(3 + 5) * 2`。
- 开及闭花括号都位于控制关键字的第一个字符下方，其中的块缩进3个空格：

```
if( a == b )
  {
    c = 50;
    printf( "%i\n", a );
  }
```

第一章　C 程序解析

作为一个有经验的程序员，你可能会想立即投身于 C，但在此之前，你应该了解所有 C 程序的基本模型。本章剖析了一个 C 程序，但没有陷于规则的形式定义或例外情形中。

这里的讨论围绕一个不大的名为 VOLUME.C 的典型 C 程序。为习惯 C 程序的形式以及组成它们的基本成分，在阅读时你应常常参考 VOLUME.C。

§1.1　一个典型的 C 程序

VOLUME.C 是一个计算一球体体积并打印下列结果于屏幕上的简单程序：

```
Volume: 113.040001
```

象本书中所有的典型程序一样，VOLUME.C 也在 QuickC 的联机帮助中。“引言”介绍了怎样装入典型程序。图 1.1 说明了 VOLUME.C 程序。

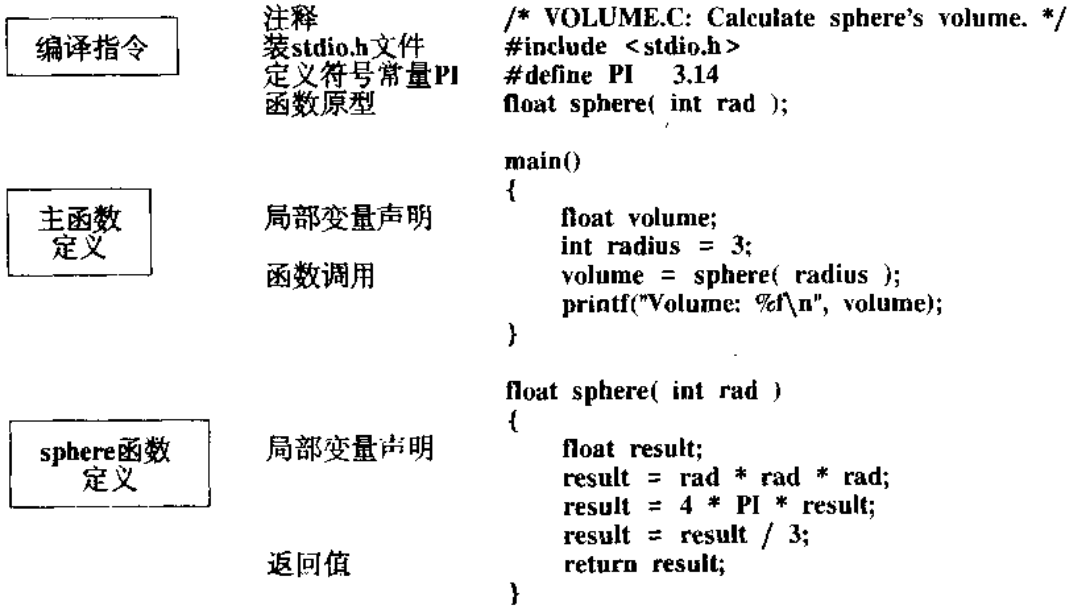

图 1.1　VOLUME.C 程序

§1.2　注释

VOLUME.C 的第一行是一注释：

```
/* VOLUME.C: Calculate sphere's volume. */
```

在 C 中，注释以/*开始，以*/结束。由于 C 只有很少的关键字，注释在使程序可读性

好这一方面就起了重要作用。

注释不可以嵌套(将一个注释放在另一个注释里面)。下面这行含有语法错误:

```
/* Error! /* You can't */ nest comments in C. */
```

§1.3 语句

C 语句总是以分号结束。例如 VOLUME.C 中的语句:

```
result = 4 * PI * result;
```

还可以用花括号将一组语句括起来以形成一个“语句块”。语句块含诸如一函数体内的各语句这样的相关语句。

C 语言忽略源程序中的空格，所以你几乎可以以任意风格来安排程序。但是通常，典型的 C 程序中是每行一个语句，花括号垂直对应，其中的语句是缩进的。引言中描述了本书所使用的编程风格。

§1.4 关键字和名字

C 是一种对大小写敏感的语言(即区别大写字母与小写字母)。所有 C 的关键字都用小写给出; 联机帮助中含有 C 关键字的一个完整集合。

你可以以大小写的任意组合来声明名字，但是许多程序员更喜欢用小写来表示变量和函数名，用大写来声明符号常量。(一个符号常量是一表示常数的描述名。在 VOLUME.C 中，PI 就是一符号常量)。

§1.5 预处理指令

一个程序中并非每行语句都是可执行的。程序可以包含“预处理指令”——用于 QuickC 编译器的命令。这样的指令以#开始，结尾处不加分号。

VOLUME.C 的第二、三行就是预处理指令。#include 指令告诉 QuickC 在编译 VOLUME.C 时插进文件 stdio.h:

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

STDIO.H 是 QuickC 提供的许多包含文件中的一个。包含文件含有为 C 库函数所需的声明及定义(“库函数”是和 QuickC 一起提供而非由你编写的)。在程序 VOLUME.C 中，printf 库函数需要来自 stdio.h 包含文件中的信息。

#define 指令定义一名为 PI 的符号常量:

```
#define PI 3.14
```

这样，每当 PI 在程序后面出现时，QuickC 就用 3.14 代替。符号常量可以是字母、数字或其它字符的任意组合。

§1.6 函数

函数是 C 的基本组块。每个 C 程序都至少有一个函数，每个可执行的 C 语句都出现在某函数内部。简单地说，一函数就是完成某特定任务并常常将一值返回至调用它的语句上的一组语句。

C 函数的目的与 Quick Pascal 的过程及函数，或者 BASIC 的 SUB 及 FUNCTION 过程相同。利用它们可以写出组织良好的程序。

C 还用函数来完成所有的输入与输出(I/O)。与其它高级语言不同，C 无诸如 PRINT 和 READ 这样的语句，所有的 I/O 都是通过调用 printf 这样的库函数而完成的。

VOLUME.C 程序含两个函数：main 和 sphere。每个 C 程序的主要执行部分本身就是一个名为 main 的函数，该函数标志着执行的开始与结束。当运行 VOLUME.C 时，执行从 main 函数的头部开始，在 main 的尾部结束。

§1.7 调用函数

函数可在一程序的任何地方被调用，它们既能接受值，也能返回值。传递给函数的一个值被称为一个变元。

程序 VOLUME.C 含两个函数调用，一个是调用库函数 printf，另一个是调用程序中定义的函数 sphere。下列语句调用 printf 函数。

```
printf( "Volume: %f\n", volume );
```

该语句将两个变元传递给 printf: "Volume: %f\n"提供了文本及格式化信息;volume 给出了一数值。

C 中一函数不必一定要返回一值，它可以象 QuickPascal 的函数那样返回一值，也可以象 QuickPascal 的过程那样什么也不返回。

当函数有返回值时，该值通常被赋给变量。VOLUME.C 中的下列语句调用函数 sphere 并将其返回值赋给变量 volume:

```
volume = sphere( radius );
```

函数用关键字 return 来返回一值。在 VOLUME.C 中，函数 sphere 的最后语句将变量 result 的值返回至调用 sphere 的语句中:

```
return result
```

§1.8 声明及初始化变量

对 C 程序中的每个变量都必须说明它的名字和类型从而加以“声明”。如果用到了一个未声明变量，QuickC 在编译程序时显示出一错误信息。

VOLUME.C 中的下列语句声明了一个 float(浮点数)类型的名为 volume 的变量。

```
float volume;
```

在声明了一变量后就应在使用它前将其初始化，即赋给它一值。未初始化的变量可有任意值，所以使用它们是危险的。VOLUME.C 程序通过赋给变量 volume 一个来自函数调用的返回值而初始化了 volume:

```
volume = sphere( radius );
```

还可以在声明变量的同时对其初始化。VOLUME.C 的下列语句声明了变量 radius 为 int(整数)类型变量，并将其初始化为 3:

```
int radius = 3;
```

§1.9 外部变量与局部变量

声明变量发生在什么位置就决定了该变量在何处是可见的。在所有函数之外声明的变量是“外部的”。你可在程序中任何地方引用它(外部变量在一些其它语言中也叫全局变量)。

在一函数花括号以内声明的变量就是“局部的”。你只可在该函数内引用它。在 VOLUME.C 中，变量 result 是在函数 sphere 内声明的：

```
float sphere( int rad )
{
  float result;
  .
  .
  .
}
```

由于它是局部于函数 sphere 的，所以变量 result 就不能在 VOLUME.C 中的其它地方加以使用。将变量置成局部的可以最大地降低变量值可能会被程序的其它部分不小心地改变的危险。

当函数接受变元时，这些变元就成为该函数内的局部变量。函数 sphere 需要一个变元，命名它为 rad。在该函数内，rad 就是一局部变量。

§ 1.10 函数原型

函数声明与变量声明相差无几。函数声明——通常被称为“原型”——可使 QuickC 进行类型检查。给出原型中的信息，QuickC 就能在以后每次使用函数时加以检查以确信你传递了正确的变元数目与类型以及使用了正确的返回类型。

一个函数原型给出了如下信息：

- 函数名
- 函数返回值的类型
- 函数所需的变元表

VOLUME.C 程序包含了函数 sphere 的原型：

```
float sphere( int rad );
```

该原型表明函数 sphere 返回一浮点(float)值，并且需要一整数(int)型变元。

§ 1.11 关于 printf

VOLUME.C 程序如同本书中的大多数例子，使用 printf 库函数来显示文本。为阅读本书的其余部分，你不必了解 printf 的所有细节，但是如果了解一些基本概念就能更容易明白例子。

printf 函数与 QuickBASIC 的 PRINT USING 语句和 QuickPascal 的 Writeln 过程相似，它能以不同格式显示(通常在屏幕上)串及数值数据。

传给 printf 一个串就能打印出一简单信息：

```
printf( "Hi,Mom!" );
```

该语句打印出

```
Hi,Mom!
```

printf 函数在行尾并不自动地加一新行符，所以语句：

```
printf( "Le Nozze di Figaro" );
printf( " by W. A. Mozart" )
```

在同一行上打印如下信息:

```
Le Nozze di Figaro by W. A. Mozart
```

为从一新行开始，使用转义序列\n:

```
printf( "Hi,\nMom!" );
```

该语句在两行上打印两个字:

```
Hi,
Mom!
```

printf 中的 f 代表格式化。为打印变量及其它项的值，可以提供给 printf 以格式代码。该代码作为第一个变元，用双引号括起。

下列语句使用%x 代码来以十六进制形式打印整数 553。它传递给 printf 两个变元:

```
printf( "%x", 553 );
```

第一个变元("%x")含有格式代码，第二个变元(553)是要格式化输出的项。该行显示为:

```
229
```

printf 函数还接受几种其它的格式代码。例如，VOLUME.C 程序使用%f 来打印一浮点数。后面章节中的一些程序用%d 来打印整数，用%ld 打印长整数。

传递给 printf 的第一个变元可以包含字符与格式代码的任意组合，其它变元则为要用 printf 进行格式化输出的项。语句:

```
printf( "%d times %d = %d\n", 2, 128, 2 * 128 );
```

打印出:

```
2 times 128 = 256
```

printf 函数以从左到右的次序将格式代码与项匹配。在上例中，第一个%d 与 2 对应，第二个%d 与 128 对应，第三个与表达式 2*128 对应。

关于 printf 和其它 I/O 函数还有非常多的内容，但对它们的了解可等到你开始学习第十一章“输入及输出”时再进行，在那里，我们将详细地描述 I/O。

既然你已经浏览了 C 的全貌，现在就可以从第二章“函数”开始对 C 程序设计的一些特殊问题加以详细考察了。