Microsoft C 5.0

语 言 说 明 书

(87 ANSI 标准 C)

译 校 者 序 言

翻 译：宗丽苹 吴 倩 邦继明 鲁 倩 王 越 徐砥祥 陈 林 彭怀宇
王淑平 何 骏 郭瑞阳 萧燕林

Microsoft C 5.0优化编译语言系统是美国87年国家标准，具有效率高、功能强的特点。为了迎合广大热心者的要求，配合国内用户高水平开发工作，由中国科学院386微机研究开发组直接组织编译了此书共九册。虽然译校者大多是软件专业的研究生及工作多年的高资历工作人员，但由于印排仓促，审校时间紧迫，难免会有这样或那样的错误。望读者能给以谅解与指正。

目　录

第一章 介 绍

1.1 C 语言综述

C 语言是一种通用程序设计语言。对于它的效率、经济性和可移植性均众所周知。正是这些优点几乎是任何其它语言所无法比拟的。特别在系统程序设计中，C 语言是非常有用的，因为程序员可以编写快速且紧凑的 C 代码，并与其它系统程序有良好的界面。多数情况下，一个优秀的 C 语言程序在速度上可与汇编程序相比。C 语言还具有易于维护和可读性强的优点。

C 以比较小的语言规模，实现了效率和功能上的统一。C 没有内部函数去作输入输出、存储分配、屏幕操作和过程控制的工作。但仍然可用 C 完成所需的任务，因为 C 程序员可依赖库程序来完成这些任务。

因为 C 语言没有用某种假定或某种强制的程序模式为前提，所以易于实现，语言简洁。利用库程序，对于用户的特殊需要提供了支持；如果需要，程序员可对程序库进行裁剪而作适当的使用。

这种设计有助于使语言的性质充分体现在各种特殊的机器上，具体的 C 实现中表现出来的特殊性质就相对地孤立出来，从而帮助程序员书写出可移植的程序。C 语言的严格定义使之不依赖于任何特殊的操作系统或机器，同时，程序员可方便地加入指定的系统子程序以提高对机器的使用效率。

注意：

Microsoft 保证与美国国家标准——程序设计语言 C（这以后简称为 ANSI C 标准）的草案提出的 C 语言扩展标准保持一致性。对 ANSI C 标准的 Microsoft 扩充已在正文中注明。由于这部分扩充不在 ANSI C 标准之内，它的使用可能会限制程序在不同机器间的移植，能否使用它，请参考你的系统手册（编译器指南）。

C 语言有下述重要的性质：

- C 语言提供了一整套循环、条件判断和转移语句，实现了对程序逻辑流的有效控制，是有利于结构化程序设计的。
- C 语言提出了一较大的操作符集合。大多数 C 的操作符是与一般机器指令相一致的，可直接翻成机器代码。其他操作符明确指定不同的操作，可产生最短的机器代码。
- C 语言的数据类型包括几种尺寸的整数，单、双精度浮点数。程序员可以设计更为复杂的数据类型，例如：数组和结构来适应特殊的程序需求。
- C 语言容许程序员说明变量和函数指针。一个指针即是与机器地址相关的说明项。正确地使用指针能提高程序的效率，因为指针是让程序员以相同于机器码的形式存取内存的数据。C 语言中还支持指针的运算，允许程序员直接访问和操纵内存地址。

● C语言的预处理也是一种正文处理，是编译之前对正文文件的再安排。其中，用的最多的是定义程序的常数、代替函数调用的宏（较快速运行）和条件编译。

● C语言是一种很灵活的语言，容许程序员作出各种决定。为了保持这种特性，C很少有强制性的限制，例如类型转换。这通常是很有益的，不过，C程序员必须了解语言特征，以便更好地理解C程序。

1.2 关于本手册

Microsoft C语言参考手册（本手册）定义了由Microsoft公司实现的C语言。它是提供给对C或其它程序设计语言上有一些经验的程序员的一本参考手册。这里假定了你已具备了程序设计的基本知识。

注意：

如果你打算快速地一览Microsoft C的明确定义，以及了解与Brian W.Kernighan和Dennis M.Ritchie所定义的C的差异，可参见本手册中的附录B和A。它们分别提供了本语言的文法及差异说明。

C库函数可用于Microsoft C程序中，对此的讨论在另一本《Microsoft C程序库参考手册》中专门介绍。

《Microsoft C编译器用户指南》解释在你的系统上如何进行编译和连接；还解释有关的特定信息。

本手册的组织如下：

第二章“C的成份”描述了字母、数字和符号出现在C程序中的各种用法，以及对编译器来说的特殊含义。

第三章“程序结构”描述C程序的组织结构，并解释怎样组织C的源程序文件。

第三章“说明”描述如何在C语言中对变量、函数和各种用户类型说明其属性。C语言中提供了一些预定义的数据类型，并允许程序员说明合成类型及指针。
此外，函数原型（Function prototypes）——C的一个相当新的特点在本章以及第七章“函数”中作了讨论。

第五章“表达式与赋值”描述C语言中表达式和赋值的各种操作数及操作符。本章还讨论了类型的转换及表达式运算可能出现的副作用。

第六章“语句”阐述C程序执行时，各种语句构成的控制流。

第七章“函数”讨论了C中的函数。特别是解释了函数原型(function prototypes)，形式参数和返回值以及怎样定义，说明和调用函数。

第八章“预处理命令和杂注”描述C预处理器所能识别的命令。C语言的预处理是在编译前自动完成的正文处理工作。本章还介绍了可以嵌入在源程序文件中对编译器的指示杂注。

附录A“差异”开列出Microsoft C与W.Kernighan & M.Ritchie所定义的C之间的差别。

附录B“语法总结”开列出Microsoft C的语言文法规则。

本章的余下部份是描述贯穿于整个手册的记法约定。

1.3 记法约定

本手册使用了下列记法约定：

约定	含义
● 关键字 (Keywords)	正文中的粗体字书写时须严格按要求，它们包括C语言关键字，如 goto 和 char，还有操作符，如加号（+）和乘号（*）。
● 位置标记 (place holders)	在语法描述或正文中可能会出现斜体字。这些斜体字是用作位置标记，在具体的C语言程序中可以用特定项或特定值替代。 例如在 goto name 中，这里 name 用的是斜体字，表明这只是 goto 语句的一般形式，你必须用一个具体标识符替代这个位置标记 name。间或，斜体字也用作强调正文中的特殊词。

● 范例 Examples	C语言程序和程序成份中的范例用一种专门字体，看来类似于列在屏幕上或者一般用的计算机打印输入出当中。
	`int x, y;`
	`.`
	`.`
	`.`
	`swap (&x, &y);`

● 输入：输出	某些范例中既有程序输出又有用户输入，这种情况下，输入用黑体字。

● 垂直省略点	垂直省略点用在程序例子或语法中以示程序的这部分被省略了。在下面的这个例子中，垂直省略法表明在两个花括号{ }之间有一些或没有说明项，然后是一个或多个语句：
	`{`
	`[`*declaration*`]`
	`.`
	`.`
	`.`
	statement
	`[`*statement*`]`
	`.`
	`.`
	`.`
	`}`
	下述垂直省略点的引用出现在程序两行之间，它表示可在此处插入一些程序行，但并没有具体写出来：
	`int x, y;`

```
.
.
.
swap (&x, &y);
```

● **水平省略点** …

水平省略点表示其后可跟与左边相同形式的若干项。例如

={expression[, expression]…}

这表明在花括号之间可出现一个或多个用逗号分隔的表达式 expression

● **[可选项]**

用[ ]括住的语法成份为可选项，即可有可无，例如：

return [expression]

这一语法描述表示 return 语句的关键字后面可跟可省略的表达式。

● **[ ]**

[ ]用在C语言数组说明和下标表达式中，例如 a[10]就是一个C语言下标式中用括号的例子。

● **"定义项"**

双引号标记在正文定义中起区分作用，如 "token"。某些C组成单元，如字符串要用双引号来标记。这里所说的双引号是\ "，而非 " "。

例如\"abc"是一个C的符号串。双引号偶尔也用在会话中。

KEY + NAMES 专用的组合键的名字，如：CTRL + z 是用小号的大写字母串表示。

第二章　C 的 成 份

2.1　简介

本章描述C程序设计语言的基本成份。所谓基本成份是指构成C程序的各种名字、数字和符号。在本章将要讨论的有下述内容：

- 符号集
- 常数
- 标识符
- 关键字
- 注释
- 标志

2.2　符号集

在C程序中，定义了两个符号集可供使用。它们是C符号集和可表示符号集。C符号集包括字母、数字和对C编译器有指定含义的标点符号。任何的C程序都必须由C符号集中的符号构成具有完整含义的语句，再组成程序的整体。

C符号集是可表示符号集的一个子集。可表示符号集包括所有的字母、数字以及用户用单字符可绘出的任何符号。可表示符号集的范围是依赖于终端、主控制台或符号发生器的类型。

C程序仅可选用C符号集。例外的是文字串、符号常数和注释，以及非include命令中的文件名它们可采用任何可表示字符。C符号集中的每个符号对C编译器来说都有明确的含义。当编译器碰到非法符号或不属于C符号集的符号时，即会产生出错误信息。

下面将介绍C符号集中的各种符号及给出相应解释和使用情况。

2.2.1　字母、数字和下划线

C符号集中包括英文符号表的所有大写、小写字母、十进制阿拉伯数字系统和下划线：

- 大写英文字母：
  ABCDEFGHIJKLMNOPQRSTUVWXYZ
- 小写英文字母：
  abcdefghijklmnopqrstuvwxyz
- 十进制数字：
  0 1 2 3 4 5 6 7 8 9
- 下划线（_）

这些字母和数字可用来构成各种常数、标识符和关键字，这在本章的后面还要阐述。

C编译器对待大写和小写不同的字母视为不同的符号。如果a被指定为某一项，则不能用A来表示该项，必须使用小写才有相同的含义。

2.2.2 空白符号

空格符、制表符 (tab)、垂直制表符 (vertical tab)、回车符 (carriage return)、换行符 (line feed) 和换页符 (form feed) 都称为空白符，因为它们在显示某正文页面时，字与字之间，行与行之间的空白间隔，就是用它们来表示的。这些符号的内部值虽不同，但都是对用户定义的项起分隔作用。如在一程序中，各种常数或标识符与其它项的分隔等。

CONTROL + Z 这一特殊功能键，用于表示文件终结。编译器对任何正文的末尾总是要识别这一标记。在它之后的字符将被忽略。

除了这些符号在程序中起分隔作用，或作为字符常数的一部分，或是文字串的内容，编译器总是忽略这些空白符号。这可以使你在程序中加上适量的空白符以增强其读性。注释的内容也视为空白符号。（见2.6部分）

2.2.3 标点符号和特殊符号

在C符号集中的标点符号和特殊符号可用于多种用途。从一程序正文的组织到定义各种任务都需要使用这些符号来通知编译器。被编译的程序使用了这些符号也会有明显的易读性。下表开列了这些符号：

表2.1：标点符号与特殊符号表

符　号	名　称	符　号	名　称
,	逗　号	>	右尖括号
.	圆　点	!	感叹号
;	分　号	|	竖　线
:	冒　号	/	斜　线
?	问　号	\	反斜线
'	单引号	~	波折号
"	双引号	#	井　号
(	左（圆）括号	%	百分号
)	右（圆）括号	&	and 符
[	左方括号	^	xor（不可兼"或"）
]	右方括号	*	乘　号
{	左花括号	-	减　号
}	右花括号	=	等于号
<	左尖括号	+	加　号

这些符号对C编译器来说都有固定的含义。本手册各书都将要使用这些符号。另外，仅能出现于文字串、符号常数、注释和#include命令的文件名中的可表示符号集并没有开列在

此表中。

2.2.4 转义序列

表2.2：转义序列表

符号序列	名称
\n	回车换行
\t	水平制表
\v	垂直制表
\b	左徹一格（退格）
\r	回车
\f	换页
\a	响铃报警
\'	明引号
\"	双引号
\\	反斜线
\ddd	八进制 ASCII 代码值
\xddd	十六进制 ASCII 代码值

转义序列又称为换码序列，是由确定的符号组合表示的一种空白或禁止打印的符号。常用于字符串和常数中，以表示某种特定的动作。例如，需要使用转义序列控制回车和制表跳格等在终端和打印机上的动作，以及下面的字符的文字表示：禁止打印的字符和通常具有特定意义的字符如双引号，转义序列由反斜线（\）开头，后面跟有字母或数字的组合。表2.2开列了C语言的转义序列。

如果反斜线后面的字符不包括在上表中，那么反斜线不起作用，例如\c仅表示某个正文串或字符常数中的字符C。尽管如此，ANSI C标准为了以后便于标准化，对转义序列中小写字母的使用有保留。因此，出现非定义的转义序列虽说目前无害处，但可能会对以后的移植性问题带来不便。

当使用\ddd和\xddd序列时，可表示ASCII码的任意字符，\后面的数字（d）或是三位八进制数或是三位十六进制数。例如ASCII码中的退格符可用通常的C转义序列（\b）表示，也可用\010（八进制）或\x008（十六进制）表示。八进制转义序列中只用到数字0到7。该序列中虽然不必一定要3位，但至少要出现一位数字。例如退格符可以表示成\10。十六进制转义序列也至少要有一位数字，前两位可省略，例如退格可以表示成\x08或\x8。

注意：

在串中使用八进制或十六进制转义序列时，最安全的还是使用全数字序列（即三位八进制或三位十六进制数序列。）如果你少用了一位或二位，而紧跟在这个转义序列后面的字符碰巧又是一个八进制或十六进制数字，那么编译器会将它当作前面的转义序列的一部分。例如串"\x07Bell"将是示成{ell，因为\x07B被编译器解释成ASCII码中的左花括号{。串

\x007 Bell（注意前面两个连续的0起作用）才是正确的，表示单词Bell跟在报警符号后面。串\x7Bell则会引起错误，产生编译诊断信息，因为7BE这个十六进制数太大不能放在一个字节中。

转义序列还能表示不可显示的控制符，送给显示设备。例如转义序列\033就是常用于向终端或打印机发送命令的有符号（ESCAPE）。某些转义序列是设备专用的。例如垂直制表符（Vertical tab）和换欠符（form feed）（\V和\f）就不能影响屏幕输出，但是它们使打印机产生正确动作。

重要点：

在C程序中，不可显示的控制符需要用转义序列表示，因为直接用该字符可能会出错，产生编译器诊断信息。

反斜线（\）也可用作连续符。当换行符紧跟在反斜线\后面时，编译器将忽略\和换行符，而将新的一行当作前一行的一部分，这主要是用在预处理程序定义比一行要长的情况。过去，这种方法也用作产生长于一行的串。尽管如此，现在产生长串文字，多用串连接运算（见2.3.4节）

2.2.5 操作符表

操作符是对数值进行处理的一些特殊符号（既有单个字符，也有字符的组合）。每个操作符被解释为相应的程序单元，称为一个“单词”（“单词”的定义见2.7节）。

表2.3列出了C语言中的一元运算操作符及其相应的名称。表2.4列出了二元和三元运算符及名称。在使用中必须按照表中开列的形式正确地书写，在两字符复合操作符中不得使用空格。注意有三个操作符（*，-，&）同时出现在两个表中，它们究竟被解释为二元还是一元运算符要视所在的上下文而定；另外，sizeof操作符不包括在这两个表中，因为它是以关键字的形式出现，而非特殊符号，所以被列在2.5节。

2.3 一元操作符

操作符	名　称	操作符	名　称
!	逻辑非	*	访问内容
~	按位求补	&	访问地址
—	算术负号	+	一元加*

* 一元加（+）的执行号语法上的，而非语义上的。

表2.4 二元和三元操作符

操作符	名称
!	逻辑非
~	按位求补
+	加
-	减
*	乘
/	除
%	取余
<<	左移
>>	右移
<	小于
<=	小于等于
>	大于
>=	大于等于
==	等于
!=	不等于
&	按位与 (and)，地址
|	按位或 (or)
^	不可兼按位或 (xor)
&&	逻辑与
||	逻辑或
,	顺序求值
?:	条件*
++	增量
--	减量
=	简单赋值
+=	加赋值
-=	减赋值
*=	乘赋值
/=	除赋值
%=	取余赋值
>>=	右移赋值
<<=	左移赋值
&=	按位与赋值
|=	逐按位或赋值
^=	逐位不可兼或赋值

* 条件操作符是一个三元操作符，而不是一个两字符在一起的操作符。在问号与冒号之间还有一条表达式。其形式如：　　表达式? 表达式: 表达式

对于上述每种操作的完整描述请见第五章。

2.3 常数

在程序中能作为常数值的有数字、字符串。常数值在执行的自始至终均不能改变。C语言中有四种常数可供使用，它们是：整数、浮点数、字符常数和字符串。下面将分别定义它们的每种形式和使用。

2.3.1 整数

整数可以是十进制、八进制、十六进制数字表示的整数值。十进制常数的形式为：

digits

这里 digits 可以是从 0 到 9 的一个或多个十进制数位，第一位不能是 0 。

八进制常数的形式为：

0digits

在此，数字可以是一位或多位八进制数（0 到 7 之间）。起始 0 是必须的引导符。

十六进制数字是下述形式：

0xhdigits

0Xhdigits

在此，hdigits 是一位多位十六进制数（从 0 到 9 的数字，并从“a”到“f”的字母）。引导符 0 是必须有，X 及字母可用大写或小写。

注意：空白字符不可出现在整常数数字之间。表2.5列出了整常数的形式：

表2.5 整常数的例子

十进制	八进制	十六进制
10	012	0xa或0xA
132	0204	0x84
32179	076663	0x7db3或0x7DB3

整常数在不加特别说明时总是正值。如果需要的是负值，则负号“-”必须放置于常数表达式的前面，来表示此负值。负号被看作为一个算术操作符。

每个常数依其值要给出一种类型。当整常数应用于一表达式时，或出现有负号时，常数类型的决定必须执行相应的转换，十进制常数可等价于带符号的整型或长整型，取决于所需常数的尺寸。

八进制和十六进制常数可对应整型，无符号整型，长整型或无符号长整型，也取决于常数的尺寸。如果常数可用整型来表示，则给它为整型。如果常数值大于一个整型所能表示的最大值，但是又小于整型位数所能表示的最大数，则给它为无符号整型。同理，如果一个常数比无符号整型所表示的值还大，则给它为长整型。如果需要，当然也可给之为无符号长整

型。

表2.5列出了八进制和十六进制不同类型所能表示的值域。在此是指十六位计算机上的常数值。

表2.6：八和十六进制常数的类型值域

十六进制域	八进制域	类型
0X0-0X7FFF	0-077777	int
0X8000-0XFFFF	0100000-0177777	unsigned int
0X10000-0X7FFFFFF	0200000-017777777777	long
080000000-0XFFFFFFFF	020000000000-030000000000	unsigned long

上述的类型规则指明了十六进制和八进制常数当它们转换成较高类型时符号是不被扩展至高位的。有关类型的转换请见第五章“表达式和赋值”。

程序员也可以强制一个常数为长型，这要使用字母“l”或“L”于常数的尾部。表2.6介绍了长整型常数的表示。

表2.7：长整型常数的示例

十进制	八进制	十六进制
10L	012L	0XaL或0XAL
79l	0115l	0X4fl或0X1Fl

在第四章有详细的类型的描述。有关转换的讨论请参见第五章

2.3.2 浮点常数

浮点常数是一个十进制表示的符号实数。符号实数的值可包括整数部分、尾数部分和指数部分。浮点常数的形式如下：

[digits] [.digits] [E|e [-|+] digits]

在此digits是一位或多位十进制数字（从0到9）。E（也可用e）是指数符号。小数点之前的是整数部分，小数点之后的是尾数部分，它们是可省略的。小数点在没有尾数时可省略。

指数部分用E或e开头，幂指数可以为负，当没有负号时视为正。在浮点常数中不得出现任何空白符号。

在不加说明情况下，浮点常数为正值。如果表示负值，需要常数之前使用负号。负号（-）必须放置于浮点常数表达式的前面。负号视为一个算术操作符。

下列例子中介绍了一些浮点常数和表达式的可能形式。

15.75

1.575E1

1575e-2

-0.0025

-2.5e-3

25E-4

所有的浮点常数视为双精类型。

浮点常数的整数部分可可省略，如以下形式是允许的：

.75

.0075e2

-.125

-.175E-2

2.3.3 字符常数

'Char'

单字符常数是由单引号（'）括住的可表示字符集中的单个字符。转义序列视为单个字符，因而也是有效的字符常数。注意转义字符必须用转义序列表示，否则会出错。一个字符常数的值就是该字符在字符集中的数值。

在上面的语法描述中，字符（Char）可以是可表示字符集（包括任意转义序列）中除单引号（'）、反斜线（\）和换行符之外的任意字符。在使用单引号或反斜线的常数时，在其前面必须再用反斜线，见表2.8的使法。要表示换引符，用转义序列\n。

表2.8：符号常数的例子

常　数	值
' '	单个空格
'a'	小写 a
'?'	问　号
'\b'	退　格
'\X1B'	ASCII 码的 escape 符
'\''	单引符
'\\'	反斜线

字符常数总是整型，在类型转换中不带符号（详见第五章的5.7节。“类型转换”）

2.3.4 字符串

语法 "Characters" ["Characters"]…

“字符串”是括在双引号（" "）之中的可表示字符集中字符的序列。下面是一个单字符串的例子：

"This is a string literal."

在一个字符串中，Characters 是位置标记，须用 0 个或多个可表文字符集（包括转义序列）中的字符代替。双引号（"）、反斜线（\）、换行符要用它们各自的转义序列（\"，\\，

\n) 表示。非打印字符也总要用相应的转义序列表示。每个转义序列视为单个字符。

为在一个字符串中造成换行，可以在你想换引的地方加上换引转义序列（\n），范例如下："Enter a number between 1 and 100 \n or press Return"

传统方法形成是于一行的字符串是写一个反斜线然后按 RETURN 键。这个反斜线使得编译器忽略其后的换行符。例如，字符串：

```
"Long strings can be bro\
ken into two or more pieces."
```

实际串是：

```
"Long strings can be broken into two or more pieces."
```

仅用空格隔开的两个或多个字符串将被连接成一个字符串。例如，长字符串作为 Printf 函数的参数输入现在可以从相继行的信息总列位置继续写而不影响输出结果，如果象以下输入：

```
Printf  ("This is the first half of the string,"
                    "this is the second half");
```

只要串的每一部分都用双引号括位，各部分将被连接起来输出为一个串：

```
"This is the first half of the string, this is the second half"
```

以前你也许用一个反斜线后面跟一个换行符插到比一行长的串中，现在你都可以用串连接操作代替。由于可以保证串从源程序的行总列开始而不影响它们的屏幕表示，所以各个串可以适当安排位置以增强源代码的不读性。例如下面的指针，开始时是作为两个反用空格分隔的字符串存贮在单个串中，当它作为函数参数正确引用时（如下面的例子所示）将产生和上面的例子相同的结果：

```
Char  *string = "This is the first half of the string."
                "this is the second half";
Printf ("%s", string);
```

为了将双引号或反斜线插到一个字符串当中，必须在（"）和（\）之前冠以\，如下例所示：

```
"First \\ Second"
"\"yes, I do, \"she said."
```

注意一个字符串中的转义序列（象\\或\"）当作一个单字符

字符串的字符按照从左到右的顺序依次存贮在连续的存贮单元中。空字符（用转义序列\0 表示）自动添加到串尾表示每个字符串的结束。程序中的每个字符串通常视为是彼此不同的，尽管如此，两个相同的字符串还是不能存贮在不同的单元中。因此，程序在执行当中是不允许修改字符串的。

字符串的类型是 char[]。这里的意思是一个字符串是一个元素是字符型的数组。数组元素的个数是串中的字符数加 1（最后的空字符也作为数组的元素被存储）。

2.4 标识符

标识符是提供给变量、函数和标识，在给定程序中的名字。你可以用说明的方式建立标