近十年来对计算机程序设计
实践的最重要的贡献之一

C语言问题及习题解答

刘乃琦 编

电子科技大学出版社

高等学校教学用书

C语言问题及习题解答

刘乃琦

电子科技大学出版社

·1990·

内 容 简 介

本书提供了B. W. Kernighan和D. M. Ritchie所著著名的《C语言》（1988年新版）的全部全新的习题及若干疑难问题解答，所有习题均按照新的ANSI C国际标准进行解答编程，给出了在C语言学习、编程和应用中涉及到的疑难问题的解释，列出了ANSI C和原标准C之间的差别和主要改动之处，给出了C语言标准库参照，向读者提供了一本难题解答，程序对照，程序分析，编程技巧和编程艺术方面的极好参考书。

该书材料充实，内容新颖，程序实用有效，可以作为广大C语言编程人员的必备书籍，也是广大计算机工作者和高等院校教师和学生的实用参考书。

高等学校教学用书

C语言问题及习题解答

刘乃琦

电子科技大学出版社出版
（中国成都建设北路二段四号）
四川省青神县印刷厂印刷
四川省新华书店经销

开本 787×1092 1/16 印张 12.75 字数 300千字
版次 1990年1月第一版 印次 1990年1月第一次印刷
印数 1－7500册
中国标准书号 ISBN 7-81016-211-X/TP·16
（15452·92） 定价：4.00元

前　言

自从B. W. Kernighan和D. M. Ritchie（合称K&R）在十年前（1978年）写出了他们的传世之作“The C Programming Language”以来，计算机世界经历了革命性的发展，在机器系统结构和性能上，无论是大型机还是微型机都产生了惊人的突破，语言本身也经历了很大变化，C语言也几乎在任何一种计算机系统上安了家。虽然，C语言早期是伴随着著名的UNIX操作系统发展起来的，而现在C语言却已远远超出了UNIX系统的范围，产生了各种C语言编译程序、编译环境及软件环境，产生了功能各异的各种C语言支持系统和应用系统，成为目前最受人欢迎的一种通用编程语言。在这期间，美国国家标准化组织（ANSI）产生制定了C语言的另一个国际标准草案，称为ANSI C标准，而C语言的两位开发者（K&R）也在十年后（1988年）修改了他们的经典著作，按照新的ANSI标准修定了他们的《C语言》。

本书内容既取自于最新的C语言书籍，又取自我们在C语言应用中的实际经验。其中，习题部分主要参考C. L. Tondo和S. E. Gimpel 1989年所著“The C Answer Book”。修改后的《C语言》增加、删除了部分习题，对几乎全部的程序都按照ANSI C标准进行了修改和重写，反应了十年来C语言编程中新的结构、技巧和程序艺术。

读者通过C语言的学习，可以先针对各章习题进行编程，然后再仔细阅读本书，分析所给的程序，并上机实践，把用户自己的程序与之作一比较，这将大大有助于对C语言的理解和掌握，并获得良好的C语言编程技巧。或者，读者也可以把新旧习题及其解答作一对比，这将是很有意思、很有启发的。读者还可能碰到C语言使用中的一些问题，其中一部分将从第一部分中找到答案。

本书分为两部分，第一部分是问题的解答，对C语言的概念理解，系统掌握，环境熟悉，C语言语法结构，数据结构，运算过程，以及函数使用、调用过程，C语言应用等方面出现的问题一一作了回答。第二部分是习题解答，保持与《C语言》原书对应的章节顺序，对所有习题按ANSI C语言标准进行了解答，全书习题共111题，均经过了上机验证。此外，在附录中，总结了C语言的符号使用，简介了C语言的标准库，列出了ANSI C新标准和原标准C语言的区别，列出了其主要修改之处，也给出了目前使用的若干C编译系统的名目参考。

利用这本书，读者可以更好地学习和理解C语言，而且能将C语言用得更好。对另一些读者，他们可以在自己的系统上验证和执行这些程序，也可参照这些程序利用自己的知识编写出更好的解答。

本书的写作得到电子科技大学计算机系刘心松教授、佟远祯副教授的热情支持，邹玲声老师在成书过程中给予了极大的支持，对他们的宝贵意见和建议，在此表示衷心的感谢。此外，编者真诚希望读者在使用本书后把有关C语言的疑难问题和使用问题反馈给我

们，以便进一步解决；书中存在的不足之处也恳请批评指正。

另外，《C语言》新旧版本中的所有程序以及习题解答的全部C语言程序（约200余道）已汇集在5.25″软盘上，若需要者请与电子科技大学计算机系联系。

愿此书能对C语言的学习者有所帮助！

编　者

一九八九年七月于四川·成都

电子科技大学

目　　录

第一部分　C语言问题解答

第二部分　C语言习题解答

第一部分 C语言问题解答

在C语言学习和编程中，编程者常常遇到许多语法上，程序结构上，程序分析过程以及执行过程中的问题，有时因为未能对C语言进行深刻理解，有时因为缺少必要的参考书，这些问题往往得不到解决。为了帮助C语言学习者和应用人员了解这些问题并解答这些问题，这一部分汇集了许多与C语言编程密切相关的重要问题，它们来自各类应用实践，大部分涉及到C语言语法的使用，编程结构，数据结构等方面。

解答中所举的例子都按照ANSI C语言标准，并力求对问题解答击中目的，这些问题分别归纳为三个方面。

1） 概念及C语言系统

2） 数据结构、类型与运算

3） C语言的编程与应用

我们真诚地希望读者在C语言的学习和应用过程中，如果遇到什么问题，可以告诉我们，我们一起来解答和攻克这些问题和难关，并希望不断地提出新的问题。

第一章　概念及C语言系统

1-1　C语言与标准库是什么关系？

C语言本身并不包括标准库，换句话说，标准库并不是“C语言”的一部分，它只属于这个C语言系统（或者C语言环境），库中的所有函数只是支持C语言形成一个系统编程环境，并通过标题文件（Header）提供了标准库的函数定义、说明、类型及宏定义，提供了基本的输入输出。例如，C语言本身没有输入输出语句，它的I/O操作不作为语言本身的一部分，但并不是说C语言程序不能完成I/O操作，这些输入输出操作可以通过库或用户定义的函数来实现，这与其他高级语言相比是不同的，其他语言都把I/O作为其语言的一部分。此外，C语言也没有提供如Pascal语言中New函数那样的存储器管理，不过，编程者可以很容易编写这类程序，这就必须借助函数库所提供的功能程序块。

1-2　C语言系统的库包含了些什么？

C语言的库是一个语言环境，它支持标准C语言并提供若干功能性操作，库的组成和大小可以随系统及C编译系统不同而异，但附属C的标准库是基本一致的。用户可以通过把对应库中功能模块的标题文件在自己的程序中进行包含说明，如#include＜××××.h＞，就可以直接使用库中的对应程序。

一般来说，人们常用的是标准库。此外，有人把与语言所在机器系统（或操作系统，文件系统）有关的库函数集合又称为系统库，在这里，C程序中所用的函数的名称是与该系统对C的补充定义有关的。例如，在Intel 86/300系列的C语言中，C语言系统就含有一个称为系统接口库，这里，在C程序中对系统进行的操作，都与PL/M-86语言一样冠以dq$的前缀，如

```
void  dq$exit(  )
void  dq$open(…)
void  dq$close(…)
void  dq$read(…)
      ……
```

等，而不同于基于UNIX上C语言中的低级文件操作，如open，close，read，……等。

此外，C语言系统还可以有自己的用户库（或应用库），即用户可以根据需要生成自己的库，如图形库，数学计算库，过程处理库等，库中所含内容均面向一个领域，或者形成一个所谓“工具库”，包含若干实用程序。这样，由于库的支持，C语言就变成了一种编程灵活，支持程序雄厚，扩展开发容易的语言了，也相当于对C语言本身进行了扩充。

1-3　如何使用库函数？

库函数的使用与在C语言中进行的直接函数调用一样，只不过在使用前，在程序始端应当把对这些函数进行说明和类型定义的标题文件包括进去，否则就不能正常使用。比如，在C语言程序中要用到函数getchar( )，此时就必须在程序始端把包含对getchar( )说明的标题文件＜stdio.h＞包含进去，即

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

因为getchar被定义成getc(stdin)，否则，就不能正常使用。

除标准库外，系统库与用户库的内容是不同的，可以与机器系统有关。对标准库来说，它应当按照ANSI C的标准设置并包含如下标题文件：

assert.h	float.h	math.h	stdarg.h
stdlib.h	ctype.h	limits.h	setjmp.h
stddef.h	string.h	errno.h	locale.h
signal.h	stdio.h	time.h	

不过，在实际实现中，各种C语言版本，标准库中的定义，函数的多少等仍有不同。标准库的详细情况可参考附录B。

1-4 什么是面向目标的C语言？

面向目标的语言(object-oriented)也称为面向对象的语言，它是基于把相似的和有关的命令归并在一起的概念，这些命令都作为“目标”或者“类别”。在这种环境下，程序员只需处理目标，而不是程序语句。一个目标可以包含多行程序或者整个程序，目标能重复使用，也易于修改。而且，对于所有面向目标的语言来说，种类(或称类别，class)和目标(object)的概念是其核心。

面向目标的C语言是让C语言具有针对和支持“目标”编程的环境，主要是数据抽象和面向目标的编程技术。最早的一种面向目标的语言是简单的Smalltalk-80，但后来的Smalltalk则已大大发展了。面向目标的C语言目前著名的有Objective公司的Objective C，以及AT&T公司的C++语言。

1-5 C++是一种什么样的语言，它也是C语言吗？

C++语言是AT&T公司于1985年推出的一种面向目标的C语言编程环境。开始是由贝尔实验室的B. Stroustrup编写的“C with Class”的C语言新版本，1983年后经改进并更名为C++。顾名思义，C++(++是C语言中的增量运算符)是一种在C语言基础上的比C语言功能更强的语言，它本身作为一种通用编程语言，但最重要的是它在C语言的基础上支持数据抽象(data abstraction)和面向目标的编程技术。

C++语言是在C语言基础上的一大成功，它克服了C语言的许多局限性，提供了面向目标的编程环境。它改善了C语言的记数法，提供了较高的类型安全性，并补偿了C的弱点，它让程序员看清楚每个模块的内部情况，看清它是如何工作的。C++语言的编程效率很高，有人曾统计，没有一个用C语言编译的程序在运行时的效率能达到用C++语言写的程序的效率。

C++语言可以用于许多应用分支，如事务系统，CAD系统，图形图象处理，数据库管理，编译构造，语音合成，联网通信，编程环境，人工智能及机器人，科学计算，模拟仿真以及VLSI和ULSI设计等。C++提供了一个非常灵活的类型系统，这样，就使得在不违反C++的类型系统规则的情况下，应用混合程序设计风格成为可能，并使用户可以选择与应用领域紧密结合的程序设计风格。

虽然C语言并不是设计上最规则的语言，也不是最容易应用的语言，然而，它所具有的灵活性，高效性，可扩展和可移植性是它得以广泛应用的关键。而C++语言保留了C的

强大功能，C也吸取了C++的优点，两者各自地走着自己的生命之路却又相互影响着对方。C++是在兼容性、内部一致性和高效性的严格制约下被设计出来的，它与C语言和其他高级语言的最重要区别是对数据抽象和面向目标的编程技术的支持。面向目标的程序设计技术不依赖于任何已知的程序设计语言，C++的出现在一定程度上完善了C语言的功能，它的优越性能也正吸引着广大的编程工作人员。

目前，C++语言的微机版本有 Lattice 公司的 Lattice C++，Zortech 公司的 Zortech C++，Oasys公司的Designer C++，Guidelines 公司的Guidelines C++ 以及Lifeboat公司的Advanced C++。

1-6 C语言与PASCAL语言有什么区别，哪一种语言好？

C语言与PASGAL语言在编程结构和语法结构上都有一些区别。但必须清楚：比较每一种语言的标准性能并决断出语言谁"好"谁"坏"之举是不明智的。每一种语言都具有其生存环境、特点和局限性，它们在整个社会、经济及应用发展中各自走完自己的生命历程。当然，用户从感情出发议论所选用的语言的优点则情有可原，但一个聪明的用户是不会因选用一种语言而摒弃另一种语言的。如果用户关心的是时间、金钱乃至效率，最好的决断不是挑选一种语言，而是一个语言的编译器（或编译系统）。

具体说来，PASCAL语言采用了较多的单词，如begin，end等，可读性较好；C语言则采用了较多的符号（如icon等），若采用较多的复杂说明与结构时，可读性稍差。在编程上PASCAL比C更严格，而C则更加灵活，例如，在PASCAL程序中字符运算是无意义的：

```
VAR x : char;
x : = x - 30;      是无效的
```

可以用

```
x : = chr ( ord ( x ) - 30 );
```

但在C语言中这是可以的：

```
char x ;
x = x - 30;
```

自然，这只是编程风格与品味的不同而已，并非涉及到语言本身的能力与本质。

此外，PASCAL的输入输出操作是以语句方式存在于语言中的，而C语言却没有输入输出语句，是利用标准库中的函数或者用户定义的函数实现的。

1-7 C语言与PASCAL语言在程序结构上有什么区别？

从推荐的语言标准结构来看，它们可以对比如下，读者可以看出同一问题和操作的处理与实现的差异。

PASCAL程序	C语言程序
PROGRAM myprog;	
VAR	
I, J, K : integer;	int i, j, k;
function max (A, B : integer) : integer;	int max (int a, int b)
begin	{

```
        if  A>B
        then max :=A
        else  max :=B
    end; { End of func max }
procedure swap(var A,B:integer);
    var
      Temp  : integer;
    begin
      Temp :=A; A :=B; B :=Temp
    end; { End of proc swap }
begin  {Main body of myprog }

  I :=10; J :=15;
  K :=max(I, J);
  swap(I, K);
write('I=', I:2, 'J=', J:2);
writeln('K=', K:2)
end. {End of program myprog }
```

```
    if(a>b)return(a);
    else  return(b);
}/* End of max() */
void swap(int *a, int *b)

{
   int  temp;
   temp=*a; *a=*b; *b=temp;
}/*End of swap() */
main()
{
  i=10; j=15;
   k=max(i, j);
   swap(&i, &k);
   printf("i=%2d, j=%2d",i,j);
   printf("k=%2d\n", k);
}/*End of main() */
```

注意程序中如果考虑安全和优化，参数i，j，k是可以不作为全局变量而只作为局部变量放到主程序模块中去。读者可以一一对应看出它们的写法和异同点。

1-8 什么是标准C和ANSI C，两者有什么不同？

通常，标准C是指由Brian W. Kernighan和Dennis M. Ritchie在1978年所著的“The C Programming Language”一书的基础上确立的C语言版本，以前的C语言版本都按此标准而建立。1983年ANSI C标准推出后，就把原C语言的版本称做标准C，而ANSI提出的标准称为ANSI C，所以K&R在1988年（十年之后）再次修改了他们的著作《C语言》，完全按ANSI C的标准来介绍和编程了，新的ANSI标准问世后，各C语言编译系统都向此标准靠拢，原来的所谓标准C也就不再“标准”了。

ANSI C是由美国国家标准局（ANSI）语言标准化委员会对C语言问世以来的若干发展、补充和修改后公布的一个C语言国际标准草案。ANSI C比起原C语言有了很大发展，引入了许多新的结构、类型和功能，最重要的是在数据表示、数据结构、数据运算上的变化，如说明和定义函数变量的描述，异常错误处理，类型说明与扩展，运算符增删等。此外，第二个优点是定义了标准库，规定了诸如访问操作系统，格式化I/O，存储器分配，字符串处理，数字计算等函数，并形成了若干对应的标题文件（即.H文件），通过对这些标题文件的修改达到与主系统的密切配合和程序兼容。

两者的详细不同点请见附录C，在K&R原书中曾归结成37个主要不同点。

1-9 为什么说C语言是一种较“低级”的高级语言，它与汇编语言有什么区别？

C语言是一种紧凑、高效的语言，实际应用中，由于C语言具有对机器系统本身，输入输出操作，硬件端口，存储区地址内容甚至寄存器，中断矢量等进行直接操作的功能，所以，C语言在许多系统中可以取代汇编编程，并且比汇编语言更清晰易懂。有人称其为与

其他高级语言相比而“低级”的语言，这并不很客观。不过，C语言可以对字符、数字的地址等进行逐位操作，实现算术和逻辑运算，能够完成许多只有汇编语言才能完成的事，它有与汇编语言相类似的许多功能但又比汇编程序易于阅读和编程，而在执行速度上几乎可以与汇编语言媲美，并且具有丰富的操作种类和数据类型，可见，C语言并不低级。

1-10 有人常把C语言编译器分为两种形式，一种为命令行方式，一种为集成方式，两种方式有何不同？

这两种方式只是在使用上和编译步骤的选择上不同，编译功能是一样的。随着近年集成软件的发展，C语言的编译器上实现了一种集成编译环境，这种环境集屏幕编辑、语言编译、模块连接、动态调试和程序运行于一体；并配之以窗口、图形、高效率交互方式（如鼠标）等支持的一个功能强大的C语言工具环境，一旦进入此环境就可以自始至终完成程序的编制调试（除非程序运行死机或使系统崩溃需要重启），这一种方式称为集成语言环境或者集成编程方式。Boland公司的Turbo C和Turbo系列，Microsoft公司的Quick系列，就是一种集成方式的语言编程系统。

然而，为了种种应用需要，比如，只进行预处理，只编译形成目标文件，在编译中加入种种选择（这些选择有的在集成环境中可以用目录菜单选择，有的功能，集成环境中并不具有）。所以除集成软件工具外，系统仍然提供了另一种分散型的语言编译方式，由于此方式与C语言命令行类似，故称为命令行编译方式，也称为与UNIX（操作系统）兼容方式，其选择项多以在“-”号后跟若干字符组成。例如：

```
cc -c file ⟨CR⟩
```

以这种方式编译的文件，还必须经过连接程序把目标模块或库模块连接到一起后形成可执行文件。两种方式各有优点和长处，编程者可以选择使用。

第二章　数据结构、类型与运算

2-1　有的C语言程序中使用了许多复杂的指针形式，用户常感到困惑，读不懂，怎么办？

指针的使用是C语言的一个重要特点，也是它的迷人之处，指针的使用可以非常灵活，甚至到了“太灵活以致难以理解了”。ANSI C标准允许C语言中有复杂说明，这种类型对于用户（尤其是初学者）是很难理解的，不过，我们可以从最简单的形式开始，逐步向复杂形式过渡，其中，既要考虑算符的优先级，又需考虑（ ）的结合特性。

下面是一些类型名，注意其中没有加上所定义和说明的名称：

int	一个整型值
int *	一个指向整型值的指针
int *[3]	一个含有3个指向整型值的指针的数组
int (*)[ ]	一个指向未定数目整型值数组的指针
int *( )	一个具有未定参数的返回指向整型值的指针的函数
int (*[ ])(void)	一个具有不定大小的数组，数组的每一个元素都是一个指针，每一个指针都指向一个无参数的并返回整型值的函数

采用上述类型去定义和说明变量或函数，就产生相应的操作。例如：

int args	变量args为整型
int *addr	addr为整型指针
int *pt[4]	pt是一个指针数组，每个数组元素都是指针
int *fnc( )	函数fnc送回一个整型指针

2-2　如何区别容易混淆的指针类型说明和指针作用于函数的说明？

在C语言应用中，使用户感到不便和受到非议最多的是它的说明的语法格式，特别是针对函数的指针的引入。在这里，语法本身是为了使函数说明和使用之间达成一种协调，这种说明方式对于简单的情况是很清楚的，无可非议的，但是，对于较复杂的情况则会产生混淆，并使用户感到困惑，这是因为对一个说明过程和格式，不能完全按照从左到右的顺序进行阅读和理解，再加上过多的括号，就可能增加理解和使用上的困难。这时，只有从里到外采用层层剥皮的方式分析。在分析中，需要牢记运算符*是一个前缀操作符，它的优先级比（ ）低，所以，为了解决适当的配合，就不得不再加上括号。尽管在实际应用中，复杂的说明并非经常出现，但有必要理解和懂得这些形式，并且能够创建和使用。当然，熟练的C语言编程者还可以用一种实用程序（称为dcl或undcl）来对复杂的C说明进行分析（见习题5-19，5-20）。下面是一些复杂说明的例子，也是极易混淆的情况。

例1

```
int *f( )    返回一个指向整型值的指针的函数
int (*f)( )  指向一个返回整型值的函数的指针
```

例2

char **argv 指向字符指针的**指针**

int *daytab[13] 一个具有13个指向整型值的指针的**数组**

int (*daytab)[13] 指向一个具有13个整型值元素的数组的**指针**

例3

void *comp() 返回一个指向void的指针的**函数**

void (*comp)() 一个指向返回void的函数的**指针**

例4

char(*(*x())[])() x是一个函数，它返回一个指针指向一个指针数组，数组的元素指向返回char型的函数

char(*(*x[3])())[5] x是一个指针数组，数组元素指向一个返回指针的函数，该指针指向字符数组

在例4中可以这样理解，对char(*(*x())[])()

x是一个函数：x()

它返回一个指针：(*x())

该指针指向一个数组：(*x())[]

该数组是指针数组：(*(*x())[])

这个指针指向一个函数：(*(*x())[])()

函数返回一个字符型值：char(*(*x())[])()

对char(*(*x[3])())[5]可以这样理解：

x是一个数组：x[3]

而且是一个指针数组：(*x[3])

指针指向一个函数：(*x[3])()

该函数返回一个指针：(*(*x[3])())

该指针指向一个数组：(*(*x[3])())[5]

该数组元素为字符型：char(*(*x[3])())[5]

2-3 复杂类型应当怎样应用，有无特殊要求？

复杂类型的应用与其他简单数据类型的使用一样，只需把它们放到该类型和结构对应的地方，这样，函数就如同正常函数调用一样使用。如果有一个说明

```
int (*comp)(void *, void *);
```

它表示comp是一个指针并指向一个函数，这个函数带有两个void*型的参数变量，并且返回一个整型值。如果要在程序中使用此函数，比如：

```
if((*comp)(v[i], v[j])<0)
```

这里，(*comp)(v[i], v[j])是对comp函数的调用，这个调用送回一个整型值，比如ival，那么条件式就是

```
if(ival<0)
```

就完成了条件判断。但如果在使用时不小心掉了一个括号，如

```
int *comp(void *, void *)
```

这样，comp就成为一个返回一个指向整型值的指针的函数，当然就完全不同了。

2-4 C语言的指针数组的表示有几种，如何区别应用？

在C语言程序中应用指针数组时要掌握指针与数组的关系，只要熟悉这种关系，就可以放心地使用，例如，有如下定义的一个指针数组

```
#define  MAXWORDS  100
char  *p[MAXWORDS];
```

那么

```
char *p[100]   也等效于  char *(p[100])
```

即p是一个指向char的数组。此外

```
char (*s)[8]   也等效于  char s[][8]
```

即s是一个指针，指向一个含有8个字符型元素的数组。

2-5 void是一个什么类型，如何应用？

void是ANSI C标准引入的一个新类型，它表示函数不返回任何值，在程序中，就可以用void来说明无返回值的函数。一旦函数说明成void，由此函数的赋值将是非法的。例如：

```
main()
{  int  k;
   void  prt_char();
     ……
   k=prt_char('A');
     ……
}
```

将产生错误，赋值操作为非法，因为函数prt_char被说明成void，不送回任何值。

但是，如果不要void，即

```
main()
{  int  k;
   ……
   k=prt_char('A');
   ……
}
```

编译时不会产生错误，但将给出运行错误或打出错误的值。

2-6 什么是enum类型，如何使用？

enum也是ANSI C引入的新类型——枚举类型，它类似于C语言结构格式，定义一组常量，这些常量称为枚举元素，常量一般从0开始，按每次增1的顺序排列(也可对其赋值)，而且不必对每一元素再赋值，使用时，用enum再说明某一个具有这种结构的变量就可以了。

例1

```
enum day {sun, mon, tue, wed, thu, fri, sat};
```

这里说明了一个枚举常量集，从星期天(sunday)开始，sun=0，以后每一常量依次增

值，形成7天的数值。上述说明只是这种类型结构的说明，并不分配内存，使用时要用此结构说明某一变量，才可以分配内存并进行操作。

例2

```
enum day{sun, mon, tue, wed, thu, fri, sat} d1, d2;
```

或者 `enum day d1, d2;`

说明了变量d1，d2都具有与day同样的类型结构，并分配存储空间。但第二个式子必须已经对day作了说明后才可用。

enum类型还可以分段给值，以适应不同需要。

例3

```
enum {apple=7, pear, orange=3, lemon, peach} fruit;
```

说明了一个常量集fruit并分配内存，其枚举元素中，apple=7，因此，pear=8，因orange=3，所以，lemon=4，peach=5。

2-7 在C语言数值表示中'a'与"a"相同吗？

'a'与"a"是不同的，'a'是字符常数，而"a"是一个只有一个字符的字符串常数，它表示一个具有类型char的数组。在字符常数和数字本身的值之间并没有什么特别的关系，例如，字符常数'3'就不是数值3，所以在应用中决不能混淆，比如，要从键盘上输入数字或者字母，所接收到的字符用switch语句进行开关转换，如果写成

```
c=getchar( );
switch(c){
    case 1: ……
    case 2: ……
    case a: ……
    case y: ……
}
```

是不起作用的，因为从键盘输入的值对应于字符常数如'1'，'2'，'a'，'y'，其值是ASCII码的对应值，如果要使上式正确，可以将switch改成switch(c-48)，或者重写为

```
switch(c){
  case '1': ……
  case '2': ……
  case 'a': ……
  case 'y': ……
}
```

而且，更不能写成"1"，"2"，"a"和"y"。

2-8 C语言中运算符&和|与运算符&&和||经常容易混淆，如何区别使用？

在这里，还应包括运算符～和！的区别。首先要明确，一个是字位逻辑运算，另一个是真值逻辑运算，前者相当于变量间的逐位逻辑操作，后者相当于一种逻辑连接操作并意味着从左到右的真值运算。

例如：x＝1，y＝2，那么

x&y结果为0，x&&y结果为1

x|y结果为3，x||y结果为1

～x　结果为FE（十六进制），即－2

!x　结果为0

2-9　目前有的C语言程序中很多名称、关键字和类型说明看不懂，与原来C语言写法有不同，是什么原因？

原因之一是该程序可能按照新的ANSI C的标准进行编程，原因之二是用户定义了他自己的应用程序库，许多函数来自应用库。解决问题的办法可以有以下途径。

1）参考ANSI C语言标准，了解新的改进对C语言程序的影响。

2）阅读标准库的对应的标题文件（.h文件），查阅所用的函数、类型、宏定义等。

3）查阅用户程序自己的标题文件（.h文件），或者全局变量与数据结构。

4）查阅所连接的C语言库（除标准库外的其他库）。

2-10　C语言中结构类型的用法也很灵活，它们有各种定义说明以及使用，请总结一下这些用法。

C语言的结构类型功能很强，类似于PASCAL语言的记录类型，可以组成大型的数据结构，也可以模拟其他高级语言相应的结构，从而完成不同语言间数据的交换。要用好结构，首先必须了解其构成和说明。下面则是一些实例。

1）
```
struct date {
        int day;            /*日*/
        int month;          /*月*/
        int year;           /*年*/
    };
```

这是结构的说明，它只有结构名date，而无结构变量名，对此不分配存储空间。

2）
```
struct date d;
```

这是在上述基础上对结构变量d的定义，即d是一个具有date型的结构。当然，也可以直接在1）中{ }后加上d，此时要分配内存空间但未初始化。

3）
```
struct date d={4, 7, 1988};
```

形成了结构的初始化，使用时用“.”对结构成员进行访问，如d.month。

4）
```
struct person {
        char name[NAMESIZ];
        struct date birthday;
    };
```

这是结构的嵌套说明，即结构person中又嵌套了一个结构，可以嵌套下去，只要编程者自己清楚。

5）
```
struct tnode {
```