全国计算机语言培训教材

自学C语言程序设计及实例分析

张忠智 审核
任干生 编写

陕西电子编辑部
西部电子行业经济技术协作网计算机组

自学C语言程序设计及实例分析

张忠智　审核

任干生　编写

陕西电子编辑部

西部电子行业经济技术协作网计算机组

前　言

一九七三年由美国 Bell 实验室的 K.Thompson 和 D.M.Rithie 采用 C 语言编写了 UNIX 操作系统以来，至今全世界研究 C 语言的人在爆炸般地迅猛增长着。这主要是由于 C 语言的优点所导致的。总括其特点是：表达能力强；语言简洁；编译程序小；所要的支持少，内存占用量小；生成的代码质量高；有数据型结构能力和强的控制流结构；程序的移植性比其它语言均好；软件工具丰富；程序设计的自由度大。所以灵活，紧凑，实用，高效。可以用 C 语言来编写操作系统，编译程序，数据库管理系统及各种各样的应用软件。目前已在八位到十六位的微型机上广泛使用。在各种操作系统上均可运行。在大，中型机和巨型机上使用极为方便。从而迎得了广大软件工作者的喜欢。形成了一定程度的 C 语言热。

为了普及 C 语言，以提高编程人员的业务水平，尤其对只具有高中文化程度，而又非计算机专业毕业的各类程序人员，没有机会参加专业学习班的同志，迫切需要一本自学方式的读本。鉴此，我们在 C 语言学习班教材的基础上特编写了这本实例分析较多，层次从简到繁，文字通俗的自学 C 语言材料，以便于广大编程人员参考。

任一凡，刘策，刘凯，任烨，陈敏健，韩浩等同志参加了本书的编写和校对工作。在编写过程中得到了张忠智，陈瑞泉，石补天三位高级工程师的帮助。

本书错误和不妥之处，诚恳地希望广大读者和专家批评指正。

目　录

第一章　C 语言概况

1-1　C 语言的来由

C 语言是 unix 或 xenix 操作系统的主力语言。C 语言是由 UNIX 开发的，是一种系统程序设计语言，C 语言与 UNIX 操作系统是相互依存的，早在 1971 年，美国贝尔实验室的 K.Thompson 在 PDP-11／20 机上用 b 语言写了 unix 操作系统。b 是源于 BCPL 语言的。于 1969 年由 M.Richards 发表的，但 b 没有发展起来，是因为：

第一，PDP-11 是字节编址，而 b 是面向字的，所以无法访问单字节。

第二，高级语言应有类型结构，而 b 却没有。公有机器字，若要访问其它类型数据，就要用特殊操作符，或调用函数。

第三，b 编译后产生的代码运行速度太慢。

鉴此，便产生了 C 的要求。于 1971 年由贝尔实验室的 D.M.Ritchie 用了一年的时间写了第一个 C 语言编译程序，1972 年投入使用。

在 1973 年 K.Thompson 和 D.M.Ritchie 用 C 语言将 unix 重写了一遍，就形成了以后的各种 unix 版本，致使在全世界取得巨大成功。它已成为 16 位机以上的微，小，中，大型机的主要操作系统。目前，C 语言已独立于 unix 系统，适用于 8 位以上的微机。

1-2　C 的特点

1. 表达能力强：它能处理字符，数字，地址。能代替由硬件来完成的算术和逻辑运算。可取代汇编语言来写各种系统软件和应用软件。如 unix 的三层(核心；外层接口 shell 命令解释程序；外层子程序)，除核心中的 1300 多行(约占 10%)以外，其余均由 C 语言写成。由于 C 语言有数值处理和字处理的功能，所以它已成为一种通用语言。

2. 有数据型结构能力和强的控制流结构：它能在字符，整数，浮点数等基本类型的基础上，按层次逐层构造各种构造类型，如数组，指针，结构，联合等。所以数据类型丰富。一个分层结构的推导数据类型结构语言。

c 的控制语句功能很强。如 if，do while，switch 等。所以能写出良好的程序。

c 的存储类有静态，外部等类。有助于数据隐藏的模块化结构程序设计。

3. 语言简洁：编译程序小，运行时所要求的支持少，所以占内存空间也少。因为它没有 I／O 设施，没有并行操作，没有同步或协同程序等复杂控制。例如，它用｛　｝代替 begin-end 的复合语句括号。

4. 所生成的代码质量高：所以内存开销少。据估计，其代码效率只比汇编语言低 10-20%，由于至今的系统软件仍多由汇编语言写成，但汇编语言要因机器而异，而 C 语言就不因机器而异，可移植。故综合而论，以 C 语言为优。

5. 可移植性好：对于不同的环境(如主机和外设)，可不加修改或稍加修改就可运行同一 C 程序。至少有 80% 的代码可公用。

6. C 的缺点:

(1) 运算符的优先级较多，不便于记忆。个别的还异于常规的约定。

(2) 类型检验太弱，转换随便，故不太安全。但可用予处理 lint 检验类型匹配后可以克服一些。

(3) 理论研究不如 Pascal 和 Algol68 等语言。

鉴此，与各种语言优劣之比。全世界研究 c 的人爆炸般的迅猛增加。也希望同志们成为研究 C 语言行列中的一位。

1–3 C 语句内容简介

1–3–1 词汇表

利用词汇表中的符号和关键字，按给定的语法规则，可构成其它的符号，语句，和程序等。构成 c 词汇表的元素有:

1. 数字: 0–9 共 10 个。

2. 字母: (1) 大写英文字母 A–Z 共 26 个。

(2) 小写英文字母 a–z 共 26 个。内部处理以小写字母为主。

3. 下画线: ( ) 做为一个字母对待，即起一个字母的作用。

4. 特殊字符。包括两种:

(1) 运算符: 如下表，该表同时介绍优先级，功用，适用类和结合性。

优先级	运算符	功　用	适 用 类	结 合 性
十五	( )	强制类参数表	表明函数参数用	→向右
十五	[ ]	下标	数组	→向右
十五	→	存取结构	指向结构的指针	→向右
十五	.	存取结构	结构成员	→向右
十四	!	求逆(不是)	逻辑运算	←向左
十四	~	求反	字位运算	←向左
十四	++	加 1	增 1 运算	←向左
十四	—	减 1	减 1 运算	←向左
十四	-	取负	算术运算	←向左
十四	&	取地址	指针	←向左
十四	*	取内容	间接运算	←向左

续表一

优先级	运算符	功　用	适 用 类	结 合 性
十四	(类型名)	强制类型	类型转换	←向左
十四	sizeof	长度计算	结构	←向左
十三	*	乘	算术运算取余数	→向右
十三	/	除取整	算术运算	→向右
十三	%	整数取模	算术运算	→向右
十二	+	加	算术运算	→向右
十二	−	减	算术运算	→向右
十一	<<	左移	字位	→向右
十一	>>	右移	字位	→向右
十	<	小于	关系运算	→向右
十	<=	小于等于	关系运算	→向右
十	>	大于	关系运算	→向右
十	>=	大于等于	关系运算	→向右
九	==	恒等	关系运算	→向右
九	!=	不等	关系运算	→向右
八	&	按位与	字位	→向右
七	^	按位加	字位	→向右
六	|	按位或	字位	→向右
五	&&	逻辑合取，与	逻辑运算	→向右
四	||	逻辑析取，或	逻辑运算	→向右
三	?:	条件表达式	条件	←向左
二	=	赋值	赋值	←向左
二	*=	运算并赋值	赋值	←向左
二	/=	运算并赋值	赋值	←向左
二	%=	运算并赋值	赋值	←向左
二	+=	运算并赋值	赋值	←向左
二	−=	运算并赋值	赋值	←向左
二	>>=	运算并赋值	赋值	←向左
二	<<=	运算并赋值	赋值	←向左
二	&=	运算并赋值	赋值	←向左
二	^=	运算并赋值	赋值	←向左
二	|=	运算并赋值	赋值	←向左
一	,	逗号运算符	次序	→向右

(2) 关键字：有固定含义。不可以做为一般标识符。共 29 个，它们是：

int	整型数；	char	字符，一个字节；
float	浮点数；	double	双精度浮点数；
struct	结构；	union	联合；
long	长整数；	short	短整数；
unsigned	无符号数；	auto	自动变量；
extern	外部变量；	register	寄存器变量；
static	静态变量；	typedef	类型定义；
goto	转向语句；	return	返回语句；
sizeof	操作数字节长度；	break	间断语句；
continue	接续语句；	if	if 条件语句；
else	if 条件否则；	do	do-while 语句的执行；
while	while 循环语句；	default	开关语句中的黑默认子句；
switch	开关语句；	case	开关语句中的情况子句；
entry	(保留暂未使用)；	fortran	(保留暂未使用)；
asm	(保留暂未使用)；		

(3) 不严格的关键字：

define	宏替换定义；	undef	去掉予处理定义；
include	包含文件；	ifdef	条件编译；
ifndef	检验是否未定义	endif	结束条件汇编
line	行控制；		

5. 使用词汇表的注意事项：

(1) 词汇表中没有的符号，不允许在任何情况下出现。如以 x 代替 * 。

(2) 不允许违犯语法规则来构造符号。

(3) 在不同的上下文中，个别符号的含意可能不同，如 * 号。

① 在变量类型说明中。它解释为一个变量的指针，即 * 号是指针类型的标志。

② 在赋值语句中的赋值号的右边，表示将某变量内容赋给左边的量，即 * 号是变量内容标志。

1-3-2 语法图

语法规则的表示法。

1. 基本表示符号。有三个：

(1) →< >—：表示保留关键字，或者是不要再定义的语句实体。如字母和数字。

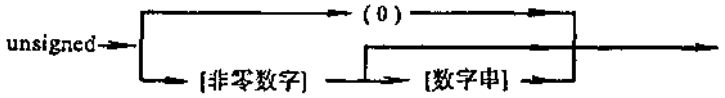
(2) →( )—：表示操作符和其它特殊符号。

(3) →[]—：表示由其它流程图定义的语法实体。

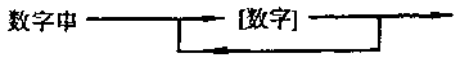
2. 语法图的作用：描述相应的语法规则。按箭头的指向，有一条或几条线路。每条线路所定义的符号、语句，程序。均为合法。

3. 语法图举例：

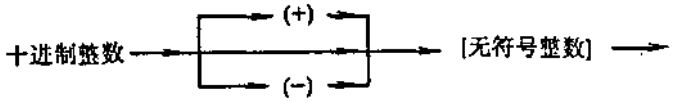
(1) 无符号数的语法图，(unsigned)。



其中：数字串的语法图如下：



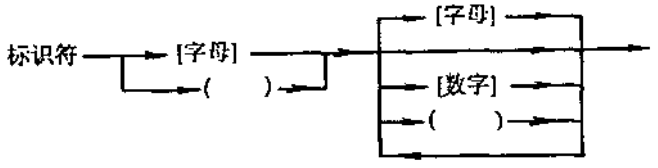
(2) 十进制整数的语法图，若无正负号时认为是正数。



该数的范围是：−2eb 到 2eb−1，其中 b 是机器的字长，如 16 位机的范围是 −32768 到 32767。

下列数是合法的：−32768，−20，0，+10，4096，+32767，

(3) 标识符语法图。标识符代表变量，常量，程序的名字等。



下列标识符是合法的：xyz，*text，sp15，global，NFILE。

1−3−3 C 的程序结构

1−3−3−1 C 语言程序组成

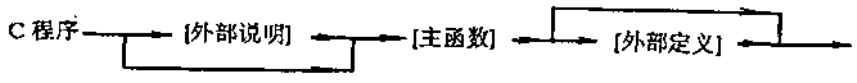
由两部分组成：

1. 操作符描述，即对要实现的算法的描述。它由一系列语句所组成。这些语句可以是复合语句，分程序，函数，过程等。

2. 操作数，即算法所要操作的对象。它们是数据。数据分为常量和变量。对变量要有类型说明。

1−3−3−2 C 语句的基本结构

1. C 程序结构语法图。



基本结构有二部分:

(1) 程序模块: 即 Include 模块. 其中:

①外部说明: 主要是函数说明和数据说明.

②外部定义: 主要是函数定义.

③主函数: main ( )

(2) 包含模块: 即上图中的下方线路.

2. C 程序由一个或多个函数所组成. 函数驻留在一个或几个源文件中. 源文件以 .c 结尾. 在多个函数中从主函数 main ( ) 开始执行.

3. 每个函数名后面一定要有一对圆括号 ( ), 它是函数的标志. 其中的参数可有可无.

4. 程序中的注释符是从 / * 开始, 到 * / 结束. / * —— * / 注释可放在程序中的任何地方. 首, 尾, 中间均可以. 程序在编译时会去掉所有的注释.

5. 程序体括号 { }, 其中可以放分程序, 复合语句, 单个语句, 或空语句. 在一个函数中至少要有一对花括号 { }.

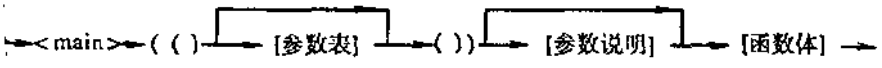
6. 例如:

```
                        / * small.c 最小的 C 程序 * /
main ( )                / * 没有可执行的代码 * /
{
}
```

1-3-3-3 主函数

其特定的标识符(函数名)为: main ( )

1. 主函数语法图.



2. 函数体的语法图.

函数体——→({)——→[内部数据说明]——→[语句表]——→(})——→

其中: (1) 内部数据说明: 主要是数据说明.

(2) 语句表: 是一组语句, 包括空语句和空.

3. 例如: 上例可以写成:

```
                    / * EX-1.C samll.c 在一行上最小的 C 程序, * /
main ( ){ }
```

相程序是正确的, 但什么也没有干.

4. 请注意养成良好的编程习惯. 即多用小函数来构成大程序. 例如, 由主函数来调用一个函数 doit, 即主函数有一个可执行语句, 就是调用函数 "doit ( );". 程序如下:

```
                    / * EX-2.C smallswb.c 调用子程序的最小 C 程序 * /
main ( )
{
```

```
    doit ( );
}
                    / * doit ( ) 什么也没有做 * /
doit (  )
{
}
```

1-3-3-4　C 中的函数

1. C 中的函数相当于 Fortran 语言中的子程序和函数. 也相当于 Pascal 和 PL/I 语言中的过程. 它把有独立意义的一部分程序封闭在一小段之内, 可任意调用, 所以它方便, 高效, 容易, 而且编程清晰.

2. 函数调用语句, 要列出函数名及其参数. 调用的目的有两点:

(1) 只控制返回, 而不要求返回的值. 可认为是一般过程.

(2) 控制返回也要求返回的值. 可认为是函数过程. 该调用语句出现在赋值号(=)的右边.

3. 函数出现的次序是任意的. 但主函数要在最前面. 函数可出现在一个或两个源文件中.

4. C 函数定义的语句图.

C 函数——→[函数头]——→[参数说明]——→[函数体]——→

其中: (1) 函数头: 说明函数返回值的类型和参数.

(2) 参数说明: 包括函数参数区分说明.

(3) 函数体: 包括数据说明和语句表. 见 1-3-4-3 节的 2.

1-3-3-5　语句终结符(;)

1. 在每个合法的可执行 c 语句之后, 必须要有分号(;)做为语句结束符. 尽管语句后有 (}), 也要有语句结束符(;).

2. 分号不是语句分隔符. 所以 main ( ) 之后没有分号.

3. 举例: 求三个整数之和.

```
                    / * EX-3.C sum.c 求 a, b, c 的和 * /
main (  )           / * 主函数, 用以来调用其它函数 * /
{
int a, b, c, sum;  / * a, b, c, sum 为整型数, 类型说明语句 * /
a = 1;              / * 给变量赋值. 此处三行可写在一行上, * /
 b = 2;
c = 3;
 sum = a+b+c;       / * 求和 * /
 printf ("sum is %d", sum);  / * printf 是通用格式转换函数, 显示输出 * /
 }
```

1-3-3-6　通用格式转换函数 printf

1. 常用的打印格式功能符如下表:

功能符	功　能	举　例	输　出
%d	十进制整数	printf("...%d\n", Q);	若 Q=6，就输出...6
%f	十进制浮点数	printf("...%f\n", Q);	若 Q=6.1，就输出...6.1
%c	单个字符	printf("...%c\n", Q);	若 Q=A，就输出...A
%s	整个字符串	printf("...%s\n", Q1);	若 Q1 是个字符数组，就显示字符数组的内容
%0	八进制数	printf("...%0\n", Q);	若 Q=177，就输出...177
%x	十六进制数	printf("...%x\n", H);	若 H=oxA，就输出...A
%%	% 本身	printf("...%%\n");	输出 ...%

2. 在 % 之前的字符串原样印出，在 % 之后的单个字符是功能符，见上表。

3. \n ---- 是换行符，碰到它就换行，没有 \n 就在一行上印出。

4. 参数替换的对应：每个 % 要依次地用后面的参数来替换。即依次地代表逗号后面的参数，将其印出。

5. 在 %d 中，若形式为 %md，表示打印的十进制数至少要有 m 位。如 %8d，就有 8 位十进制整数。

6. 在 %f 中，若形式为 %m.nf，表示打印的浮点数至少要有 m 位，其中小数点后有 n 位。如 %8.3f，则该数共有 8 位，小数点后有 3 位，小数点前有 5 位。又如 %.4f，表示小数点后占 4 位。

7. 转义序列字符:

(1) \t ---- 制表跳格符。一般跳八格。

(2) \b ---- 退格符。一般退一格。

(3) \" ---- 表示双引号。

(4) \\ ---- 表示反斜线本身\。

(5) \n ---- 换行符。

8. 举例:

(1) EX-4.C，打印字符串 "Hello, world!"

```
                / * hello / .c greet the world,  introduce output in c * /
  main ( )
  {
   printf ("Hello,  world! \n");
   }
```

(2) EX-5.C 与上例产生同样的结果。

```
                / * stream.c print "Hello,  world!" as stream output * /
  main ( )
  {
  printf ("\t");
  printf ("Hello");
```

```
printf(",");
printf(" ");
printf("world");
printf("!");
printf("\n");
}
```

(3) 对应替换参数的例子。EX-6.C

```
main( )
{
int n;
n=511;
printf("what is value of %d in octal?", n);
printf("%s%d decimal is %o octal\n","right", n, n);
}
```

本程序执行后显示的结果是:

What is the value of 511 in octal? right 511 decimal is 777 octal

注释:①第一个 printf 后无 \n,故不换行,与下面的连着显示。

②第二个 printf 中,依次把 right 替换 %s,之后的 n 替换 %d 为 511,最后一个 n 替换 %o 并把 511 转换成八进制为 777。

1-3-4 参数说明

函数带有若干参数,对其参数要加以说明。说明其类型,该说明放在参数表的后面,即( )的后面,函数体之前,即{ }之前。

1-3-5 函数返回值类型

函数一般均有返回值。调用者对其返回的值要加以类型说明。类型有多种多样,若类型位置空着,就默认为是整数型。

1-4 C 语言的编辑、编译和运行

1. 编辑:在小型机的 unix 系统和微型机的 xenix 系统上,用编辑程序来编写。如 xenix 操作系统的 ed 编辑程序。

第一,向系统管理员申请帐号。

第二,终端空时,按回车,屏幕提示 login:

第三,键入登录名和个人口令,即可进入系统。

第四,键入 ed xxx.c,按回车 <cr>,打开文件进行编辑。若该文件名 xxx.c 原已有内容,就显示 ddd...d 字符数。若该文件名为新文件,则系统在内存缓冲区建立这个文件。

第五,用追加方式时,键入命令 a <cr>,这时可输入正文。

第六，键入 <cr>，就退出追加方式，回到编辑状态。

第七，若要查看输入的内容，可键入 lp <cr>，显示一行，或键入 1，lp <cr>，就显示全部内容。

第八，随后存盘 (w 命令)保留。若此时在 shell 状态，则按 Ctrl-d，终端显示 login:

关于 xenix 的几种编辑程序，请看如何使用 xenix 操作系统一书。

2. 编译:

(1) 编辑成功的 C 程序是源程序。要经过编译才成为可执行的目标程序，即机器语言程序。

(2) 编译命令在 unix 和 xenix 中是 cc (有许多任选项)。如编译的程序是前例 EX-3.C 的 sum.c，则键入:

```
cc  sum.c
```

编译期间出现提示符￥。完成后的目标文件名是 a.out

(3) 还可以采用下面的办法来编译。

```
cc-o  sum1  sum.c <cr>
```

则编译后的目标文件为 sum1

3. 运行:

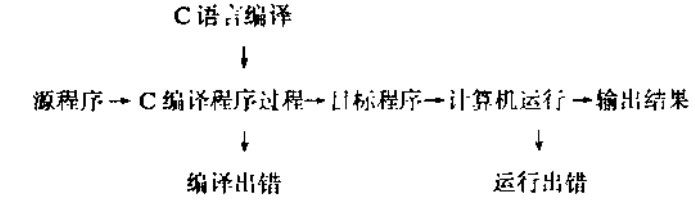
键入目标文件名再按回车就开始执行。如:

```
a.out <cr>
```

或 sum1 <cr>

就在端终上显示 sum is 6。最后给出提示符 ￥。

4. 程序编译和运行过程示意如下。对于微型机上使用 CI-C86，请见 8-4 节。



1-5　C 语言的出错信息

C 语言的出错情况，详细情况请见第九章。

1. 语法错：在编译过程中产生，由于违犯了语法规则所致。如变量名不对，语句不匹对等。通过编辑修改之。

2. 运行出错：如变量赋值不对，求了根号 0，变量超出范围等。一般程序暂停待修改。

3. 逻辑性错：在编译和运行中均无法发现。如将 100 写成了 10。可在程序中插入一些中间测试语句来观察之。

第二章　数据、赋值语句和表达式

2–1　标识符

用来标记，常量，变量，数据类型，函数和程序名。标识符由字符序列表示。

1. 标识符语法规则。

(1) 以字母 A–Z 或 a–z，和下画线 ( ) 打头。

(2) 首字符后由任意序列的字母，下画线，数字，或空串所组成。

2. 标识符的分类。共分三类。

(1) 关键字：说明一个有固定含义的字。共 27 个，用小写。不允许再赋予其它含义。如前面 1–3–1 中 4 之 (2) 所列。

(2) 特殊字：有特殊含义的字。共 7 个。用于预处理程序中。如前面的 1–3–1 节中 4 之 (3) 所列。

(3) 一般标识符：由用户定义使用。按语法规则构成。一般是前面八个字符有意义。

2–2　变量

计算机处理的对象是数据，而数据是由常量和变量所构成。程序执行过程中可以被替换的值是变量。

1. 变量说明：指出变量类型，通常是类型后跟变量名来表示。如：

```
int  lower,  upper,  step;        定义下限，上限，步长是整型。
char  c,  line[1000];             定义 c，数组 line[1000] 的元素是字符型。
```

2. 变量可按任意方式分布在若干说明中。如上例：

```
int lower;          下限是整型。
int upper;          上限是整型。
int step;           步长是整型。
char c;             c 是字符型。
char line[1000];    数组 line[1000] 的元素是字符型。
```

3. 变量与当前值的区别。

(1) 在变量说明中，如 int step，在编译时，给 step 分配一个字长的存贮单元。也可认为 step 是该存贮单元的名字。step 代表该存贮单元。

(2) 在程序中有赋值语句，如 step＝1，在执行后，则在名为 step 的存贮单元中存放了整数 1。故 1 成为 step 的当前值。

(3) 当前值是在不断变化的。如执行 step++，则每执行一次，step 的当前值就加 1。

(4) 变量的当前值只能在数据类型给定的范围内取值。

(5) 变量的标识符不能二次说明。一个变量只能有唯一的类型。如下面的错误：

int i;　　i 是整型。
float i;　　同时又说明 i 是浮点型。

2-3　常量

程序在执行之前就已知其具体值。该值可用一个文字量表示。给文字量的命名就是常量。

1. 在 c 语言中，可用宏替换来定义常量。其文字量的宏定义如下：

```
        # define 名字 值                用值来替换名字。
如:     # define  PI  311415929535      圆周率 3.11415929535
        # define CYCLELTIMIT 100        循环次数 100
        # define PAGESIZE 66            页长是 66 行
        # define ENDSTRING '\o'         字符串结束符为 '\o'
```

2. 好的 c 语言要尽量用宏定义。如改变页长时只需要修改宏定义就可以了。不必在程序有关页长的各个地方去修改常量的值。

3. 在 c 语言中的常量有：整常量，浮点常量，字符常量，和字符串常量。

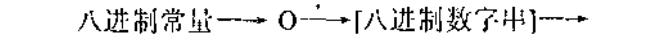
2-3-1　整常量

在计算机上占一个字长。是带正负符号的数。

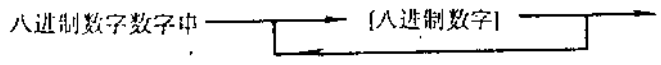
1. 十进制常量：即通常的十进制数，其语法图如 1-3-3 节中的 3 之 (1)。

2. 八进制常量：逢八进一的数制，以 o 打头。三位二进制数可组成一位八进制数。

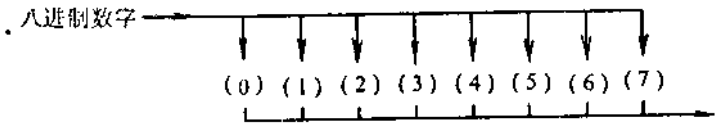
(1) 八进制常量语法图：



(2) 八进制数字数字串语法图：



(3) 八进制数字语法图：



3. 十六进制数常量。逢十六进一的数制。以 ox 或 OX 打头。

(1) 十六进制常量语法图：