第3章 数据存储类型与相关运算

程序运算的对象是数据，程序运行时数据需要存储调用和运算。数据是描述和表达客观事物的符号表达形式，是信息的载体，计算机在识别和处理信息数据时，按数据类型占用系统资源。数据类型定义说明了数据运算性质，占据存储空间多少及内部存储模式等。C语言可以使用多种数据类型，每个数据都属于某一种数据类型，每种类型又有常量或变量两种数据类型分类，不同类型的数据取值范围、表达形式及其在计算机存储器中的存放格式是不相同的，对于程序中用到的所有数据都必须定义指定其数据类型。本章主要内容有：

- 数据存储方式与数据类型；
- 存储地址与占用空间；
- 数据常量与变量定义；
- 数据存储的正负数问题；
- 数据变量取值范围；
- 各种存储类型的混合运算；
- 运算符优先级与数据类型转换；
- 各类运算符与运算表达式。

3.1 数据存储方式

计算机中所有信息数据和指令都是以二进制形式存放与运行的，应用C语言编程必须要了解数据在计算机中的存储与定义。

3.1.1 数据存储与数制转换

计算机应用的信息数据表示是多样而复杂的，不仅有能够进行数学运算的数值型数据，还有表达自然语言的字符数据，以及图形图像数据、音频视频数据等，最终都要转换成计算机可以识别和运行的二进制形式代码。由于具备2态稳定的物理状态在自然界比较容易实现，如电路电压电平的高与低、电容的充电与放电、磁场磁畴的变换、开关接通与断开、晶体管导通和截止等，使用二进制数编码，技术上容易实现，通常用0和1这两个数字表示，数据存储最终以二进制数码表示与存储。

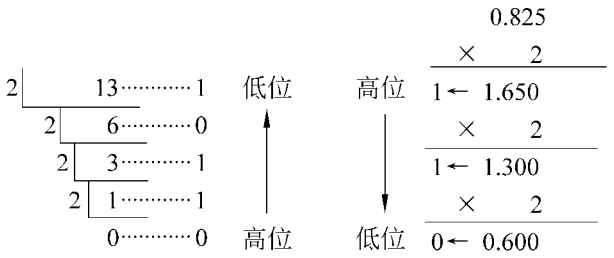
机器语言中的二进制数只有 0 和 1 两个数码，分别代表逻辑代数中的“假”和“真”两种状态。现实生活中人们习惯使用十进制，这样就存在着机器与人之间数据交流与转换问题，因此，一般有二进制、十进制、八进制、十六进制几种进位记数制之间的转换与应用，正确使用时通常由计算机自动实现数制之间的转换。常用的几种进位记数制之间的转换关系如表 3-1 所示。

表 3-1　常用进位记数制相互关系对照表

十进制	二进制	八进制	十六进制	十进制	二进制	八进制	十六进制
0	0	0	0	9	1001	11	9
1	1	1	1	10	1010	12	A
2	10	2	2	11	1011	13	B
3	11	3	3	12	1100	14	C
4	100	4	4	13	1101	15	D
5	101	5	5	14	1110	16	E
6	110	6	6	15	1111	17	F
7	111	7	7	16	10000	20	10
8	1000	10	8				

学习语言程序设计需要熟悉并掌握各种数据类型的存储性质与使用。十进制整数转换成二进制数，用除 2 取余方法，小数部分用乘 2 取整方法。

例如，求$(13.825)_{10}$的二进制形式：



求得：$(13.825)_{10}=(1101.11)_2$。

其中精度问题由计算机根据数据类型定义自动取舍。

任何进位记数制的数值都可以展开成为一个多项式，其中每个数据项是所在位权与位系数的乘积，这个多项式求和结果是该数所对应的十进制数。

例如，求$(1101.11)_2$ 的十进制形式：

$$(11001.01)_2=1\times2^3+1\times2^2+0\times2^1+1\times2^0+1\times2^{-1}+1\times2^{-2}$$
$$=8+4+1+0.25=25.25=(13.75)_{10}$$

求得：$(11001.01)_2=(13.75)_{10}$。

二进制转换为八进制，整数部分从低位到高位，小数部分则从小数点开始往右划分，每 3 位为一组，不够补 0，每组二进制数对应 1 位八进制数。

例如，求$(100111101.11011)_2$ 的八进制形式：

将$(100111101)_2$分组为100 111 101 . 110 110

对应八进制数　　4　7　5　.　6　6

求得：$(100111101.11011)_2=(475.66)_8$。

二进制转换为十六进制数，则每4位为一组，不足4位用0补齐。

例如，求$(1010110.11101)_2$的十六进制形式：

二进制数　　101 0110 . 1110 1000

对应十六进制数　5　6　.　E　8

求得：$(1010110.11101)_2=(56.E8)_{16}$。

3.1.2　数据存储类型与定义

程序设计一般包括数据描述和流程控制描述。数据描述主要是指数据结构，在C语言中，数据结构以数据类型的形式表现，定义其存储空间的大小及其数值运算范围。

标准C语言并没有规定各种数据类型占有多少字节，只要求int类型普通整型长度应大于或等于short类型短整型，并且小于或等于long类型长整型。不同的编译系统各种存储类型各有差异，因此数据存储类型的最大值和最小值限制使用就会有所不同，应用时可以使用sizeof(类型名)字节长度测试函数，在实际使用的编译系统环境中测试一下实际存储类型的字节长度，了解在定义的数据存储类型限定数值范围内，正常使用的最大数据值范围。

C语言提供了丰富的数据类型，这些数据类型及定义关键字归纳如图3-1所示。

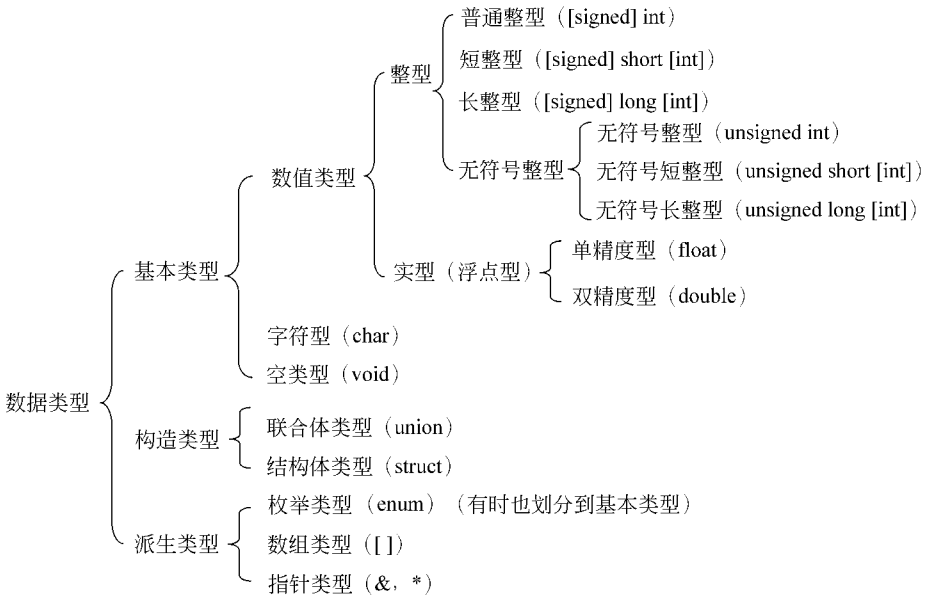


图3-1　数据类型及定义关键字

由于ANSI C并没有规定每一种数据类型的长度、精度和数值取值范围，不同的C编

译系统对于数据类型存储长度有较大的差别，应用时注意测定不会影响正常使用。

例 3-1 编程在 Microsoft Visual C++ 6.0 和 Turbo C2.0 集成环境分别测试常用数据存储类型的实际字节长度。

程序源代码：

```
/* L3_1.C */
#include "stdio.h"
main()
{
    printf("the char is%d\n", sizeof(char));
    printf("the int is%d\n", sizeof(int));
    printf("the short int is%d\n", sizeof(short));
    printf("the long int is%d\n", sizeof(long));
    printf("the float is%d\n", sizeof(float));
    printf("the double is%d\n", sizeof(double));
    printf("the long double is%d\n", sizeof(long double));
    printf("the void is%d\n", sizeof(void));
}
```

该程序在 Microsoft Visual C++ 6.0 集成环境下运行后，显示 sizeof(类型名关键字)测试各种常用数据存储类型的实际字节长度，结果如图 3-2 所示。

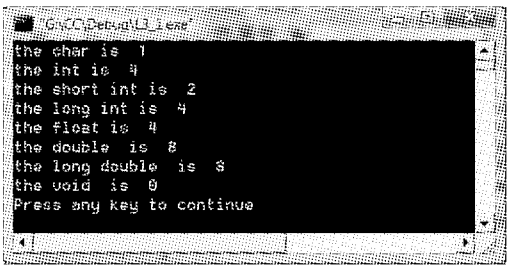


图 3-2 Visual C++ 6.0 数据存储类型字节长度

由这个案例程序的运行结果可见，Microsoft Visual C++ 6.0 集成环境下，char 字符数据类型长度为 1 个字节；int 普通整型数据类型的长度为 4 个字节，即 32 个二进制位；short int 短整型数据类型长度为 2 个字节；long int 长整型数据类型长度为 4 个字节；float 实数浮点数据类型长度为 4 个字节；double 实数浮点双精度数据类型长度则为 8 个字节；long double 长实数浮点双精度数据类型长度也为 8 个字节；void 空类型输出 0 字节。

该程序在 Turbo C2.0 集成环境下运行后，显示 sizeof(类型名关键字)测试各常用数据存储类型的实际字节长度，结果如图 3-3 所示。

由这个案例程序的运行结果可见，Turbo C2.0 集成环境下，char 字符数据类型长度为 1 个字节；int 普通整型数据类型的长度为 2 个字节，即 16 个二进制位；short int 短整型数据类型长度也为 2 个字节；long int 长整型数据类型长度为 4 个字节；float 实数浮点

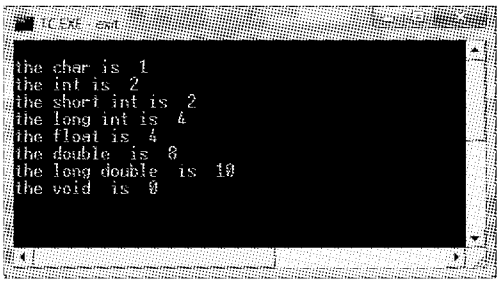


图 3-3　Turbo C2.0 数据存储类型字节长度

数据类型长度为 4 个字节；double 实数浮点双精度数据类型长度则为 8 个字节；long double 长实数浮点双精度数据类型长度则为 10 个字节；void 空类型无输出。

常用 C 语言编译系统，整型数据在内存中占 2 个或 4 个字节(Byte)，即 16 个二进制位或 32 个二进制位(bit)。

通常占 4 个字节(32 位)的单精度数据类型的数值范围为 10^{-38}～10^{38}，有效位数为 7 位；占 8 个字节(32 位)的双精度数据类型的数值范围为 10^{-38}～10^{38}，有效位数为 15～16 位，需要时注意使用就可以。

基本数据类型为常用类型，表 3-2 列出 C 语言在 Microsoft Visual C++ 6.0 和 Turbo C2.0 集成环境下对各种基本数据类型分配的字节位数和有效取值范围。

表 3-2　数据类型及取值范围

类型定义标识符	Microsoft Visual C++ 6.0		Turbo C2.0	
	长度/位	数值表达范围	长度/位	数值表达范围
[signed] int	32	−2 147 483 648～2 147 483 647	16	−32 768～32 767
[signed] short [int]	16	−32 768～32 767	16	−32 768～32 767
[signed] long [int]	32	−2 147 483 648～2 147 483 647	32	−2 147 483 648～2 147 483 647
unsigned [int]	32	0～4 294 967 295(即 $2^{32}-1$)	16	0～65 535
unsigned short [int]	16	0～65 535	16	0～65 535
unsigned long [int]	32	0～4 294 967 295(即 $2^{32}-1$)	32	0～4 294 967 295
float	32	3.4E−38～3.4E+38	32	3.4E−38～3.4E+38
double	64	1.7E−308～1.7E+308	64	1.7E−308～1.7E+308
[signed] char	8	−127～127	8	−127～127
unsigned char	8	0～255	8	0～255
void	0	无值	0	无值

C 语言中的数据有常量与变量之分，分别属于以上数据类型，这些数据类型还可以构成更为复杂的数据结构，例如利用指针和结构体类型可以构成链表结构、结构树和栈等复杂实用的数据结构。

在程序运行过程中其值保持不变的量，简称常量；而存放数据常量的是数据变量，简称变量，变量是在程序运行过程中其值能够被改变的量。常量和变量都表现或属于某一数据类型。C 语言中，常量不需要类型说明，变量则需要类型定义与说明，需要“先定义

再使用”。

3.1.3 存储地址与占用空间

计算机程序运行时需要调入内存工作区，程序中定义的数据在编译后也占有各自的内存区域，数据所占有的存储单元个数是由其类型决定的。内存空间通常划分为一个个存储单元，每个实际存储单元最小为一个字节(Byte,B)，即 8 个二进制位(bit,b)；每个存储单元均有一个唯一的编号，就是内存地址，计算机系统访问内存时是按地址操作的，而每一种数据类型所占用的内存地址空间也是不一样的，如图 3-4 所示。

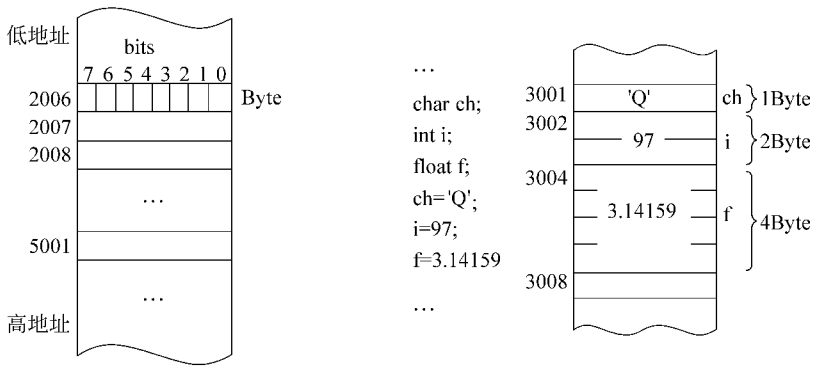


图 3-4　内存单元与地址

内存单元的地址与存储在单元里的数据是有区别的，单元地址如同房间号码，单元数据则是存放的内容，不同类型数据占有的内存长度不同，字符类型占一个字节，整形占 2 个字节，浮点类型则占用 4 个连续存储单元。各种数据类型的第一个单元地址称为“首地址”，是计算机系统寻找数据存储单元的起始地址。

例 3-2　编写一个程序，定义不同数据类型的变量并赋值，分别输出各种数据类型的变量值和所占内存空间大小，即输出各数据类型存放内容及占有的字节长度。

程序源代码：

```
/* L3_2.C */
#include "stdio.h"
main()
{
    char c;                          /* 创建字符类型变量 c */
    int i;                           /* 创建整型数值类型变量 i */
    short int s_i;
    long int l_i;
    float f;                         /* 创建浮点数值类型变量 f */
    double d;
    long double l_d;
    /* 分别对变量赋值 */
```

```
    c='B';s_i=123;i=456;l_i=789;f=123.5;d=678.9;l_d=1.7e302;
    /*按类型分别输出各变量值,并输出各类型所占用存储字节*/
    printf("the variable c is %c, save long as%d\n", c,sizeof(c));
    printf("the variable s_i is %d, save long as%d\n",s_i, sizeof(i));
    printf("the variable i is %d, save long as%d\n",i, sizeof(s_i));
    printf("the variable l_i is %ld, save long as%d\n",l_i, sizeof(l_i));
    printf("the variable f is %f, save long as%d\n",f, sizeof(f));
    printf("the variable d is %lf, save long as%d\n", d,sizeof(d));
    printf("the variable l_d is %e, save long as%d\n",l_d, sizeof(l_d));
}
```

编译运行后,显示输出各变量值和各种数据类型变量所占内存空间字节,输出结果如图 3-5 所示。

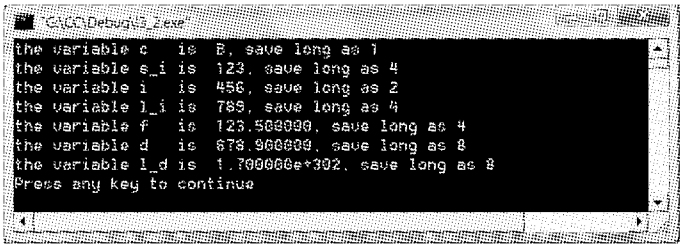


图 3-5 各种数据类型变量值及所占字节

计算机操作的最小存储单位是一个二进制位,简称位;计算机数据操作的最小存储单位是字节(Byte),8 个二进制位称作一个字节,字节是能表示信息操作的最小存储单位;存储单元则是存放指令和数据的基本单位,每一类数据的存储空间可以由一个或若干个字节存储单元组成。

存储容量通常以字节为单位来表示,如 1KB 表示 1K 字节,是 1×2^{10} 个字节,等于 1×1024 个字节长度;相应 1MB 是 1×2^{20} 个字节,等于 1×1024KB;1GB 是 1×2^{30} 个字节;1TB 是 1×2^{40} 个字节;1PB 则是 1×2^{50} 个字节,约为千万亿个字节。

3.1.4 数据常量分类

C 语言常量按数据类型分类,其实就是各种数据类型的常量数据,分为数值型常量、字符型常量和字符串常量;数值常量又分为整型常量和实型常量。这些常量无须说明就可以使用,一般从字面形式就可以判别它们的类型,以下分别举例说明。

1. 整型常量

整型常量可以用十进制、八进制和十六进制三种形式表示。

- 十进制整数:由数字 0～9 和正负号表示,如 0、－123、456 等。
- 八进制整数:由数字 0 开头,后随数字 0～7 表示,如 0123、075 等。八进制转换为十进制数符合进位记数制运算规则。例如,将 0123 转换为十进制数,即

$$1\times8^2+2\times8^1+3\times8^0=(83)_{10}$$

- 十六进制整数：由 0x 开头，后随 0～9，a～f，A～F 表示，如 0x1B3、0Xdf 等。如－7、45、6789u、05706、0x67f9 和 6789L 等。其中－7，45，6789u 为整型十进制数，05706 表示的是整型八进制数，0x67f9 表示的是十六进制整型数，6789L 则表示的是长整型十进制数。

一个长整型数常量后面加上小写字母 l 或大写字母 L，则表示为长整型常量(long int)，数据占用存储单元为长整型。

一个长整型数常量后面加上小写字母 u 或大写字母 U，则表示为无符号整型常量(unsigned int)，数据占用存储单元的最高位不作为符号位，而用来表示数据。

八进制或十六进制转换为十进制数，运算方法符合进位记数制运算规则。例如，将十六进制常数 0x123 转换为十进制数，即

$$1\times16^2+2\times16^1+3\times16^0=(291)_{10}$$

注意实际应用中，八进制和十六进制的表示没有负数，这是因为所有数据均以二进制数码物理存放，八进制数和十六进制数是由二进制表示转换而来，十进制负数转换成二进制数码存放在计算机系统中是以补码的形式表示和体现的。例如，八进制数“07602”表示正确，八进制数“－0357”则表示错误。另外，八进制数“－0687”也表示错误，因为八进制数不可能出现“8”这个数码。

程序设计中各种类型的数据在使用过程中一定要匹配，比如变量的赋值操作、函数的参数传递、输入与输出类型控制等。

2. 实型常量

实型常量也称为浮点型常量、实数或浮点数。计算机程序设计除了整数运算外还有小数运算，如 1234.567。小数是不能用整型数据方式存储的，只能用实数方法表示和使用，实数只采用十进制数表示。

实型常量有十进制小数形式和指数形式两种表示方法。

- 十进制小数形式：十进制小数表示法由数码 0～9 和小数点组成，实数表示必须要有小数点，例如 0.16、.5、2.6、9.、2.34、－678.921 等均为十进制实数表示。
- 指数形式：指数形式表示也称科学计数法，科学计数法以指数形式表示浮点类型数据，如 2.3E－5、6.542e6 等。在计算机系统中用来表示很大的数值或非常小的数值，如 2.34E－12，实际表示数值 2.34×10^{-12}。

指数形式表示由三部分组成，中间字母“e”或“E”代表数字 10，可以用大写字母“E”，也可以用小写字母“e”表示，字母“e”或“E”之前必须有数字，字母之后指数部分必须为整数。例如，7.1E5、－2.8E－2 为合法有效实数，而 2.7E、59、e7、－257、46.－E4 均为非法无效实型常量。

一个实数可以有多种指数表示形式，通常用标准规范化指数形式表示，即在字母“e”或“E”之前的小数部分中，小数点左边保留 1 位非零数字，例如 1.3562e6、3.0254E5 都属于标准规范化的指数形式。实数在以科学计数法输出时，是按标准规范化指数形式输出的。

3. 字符常量

字符类型常量包括键盘上使用的大小写字母(A～Z、a～z)、数字字符(0～9)、专用字符(!、@、#、$、&、*、|)等，每一个字符均按国际通用标准代码 ASCII(American National Standard for Information Interchange)码编码规则转换后存储，例如根据附录 A 查得大写字母“Q”的 ASCII 码值为十进制 81，转换成二进制表示为 1010001；小写字母“q”的 ASCII 码值为十进制 113，转换成二进制表示为 1110001。大写字母“Q”的存储方式为

0	1	1	1	0	0	0	1

即使用一个字节存储，不足高位补 0。实际上，ASCII 码表只用低 7 位就可以表示各种常用字符及控制字符，最多可以表示 $2^7-1=127$ 个不同的 ASCII 代码值。

字符常量在使用时还要注意一些细节上的区分，分别说明如下。

(1) 单个字符常量。

单个字符常量是一对用单引号引起来的单个字符，如'a'、'$'、'3'、'9'等，每一个字符均对应有自己的 ASCII 码，或是 ASCII 码转移控制符，如'\n'表示回车换行。注意键盘上的数字字符均有自己的 ASCII 码，而非其字符本身，例如数字字符“8”的字符常量表示为'8'，其二进制 ASCII 代码为 00001000，字符常量必须是单引号引起来。

每一个 ASCII 字符都有唯一的代码值，可以用十进制表示，例如，字符'A' 的十进制 ASCII 值为 65，字符'a'的十进制 ASCII 值为 97，字符'0'的十进制 ASCII 值为 48，字符'\n' 的十进制 ASCII 值为 10 等。

(2) 字符串常量。

字符串常量是一对用双引号引起来的单个字符或字符串，如"a"、"123"、"test"等。字符串常量中的每一个字符均有自己的 ASCII 代码值，必须用双引号引起来。双引号是字符串常量的标识，如常量"9"是字符串，而不是字符常量'9'，各自占用存储空间不同。

字符串常量'9'占 2 个存储单元：

9	\0

字符常量'9' 占 1 个存储单元：

9

可见，字符串常量和字符常量是不同的量，字符串常量使用双引号括起来，而字符常量使用单引号括起来。字符串常量如"abc"在存储器内部存储的字符序列为：

a	b	c	\0

其中 ASCII 字符'\0'(NULL)为字符串结束标志，是字符串操作的结束符。

(3) 控制字符常量。

在 ASCII 表字符集中，码值在十进制 0～255(十六进制 0x00～0xFF)之间的 ASCII

字符，其 ASCII 码值不能用字符符号表示，也是不能通过键盘操作输入的，这些字符代表着特定的意义，需用反斜杠“\”和特定字符组合表示，使用这类表示方法的字符叫做转义字符，C 语言中的转义字符如表 3-3 所示。

表 3-3　转义控制字符常量表

字符形式	ASCII 码值	功　能
\0	0x00	NULL
\a	0x07	响铃
\b	0x08	退格
\t	0x09	水平制表(tab)
\f	0x0c	走纸换页
\n	0x0n	回车换行
\v	0x0b	垂直制表
\r	0x0d	回车(不换行)
\\	0x5c	反斜杠
\'	0x27	单引号
\"	0x22	双引号
\?	0x3f	问号
\nnn	0nnn	用 1～3 位八进制数表示 ASCII 字符
\xmm	0xhh	用 1～2 位十六进制数表示 ASCII 字符

实际上，C 语言 ASCII 码表字符集中的任何一个字符均可用转义字符来表示。表中“\nnn”是用八进制数表示 ASCII 字符，“\xhh”是用十六进制数表示 ASCII 字符。“nnn”是八进制数代表的具体 ASCII 代码值，“hh”是十六进制数代表的具体 ASCII 代码。例如，'\102'表示字母“B”，'\134'表示反斜线等。以字符常量“A”为例，可以有以下几种表示形式：

- 'A'为字符“A”的字符常量；
- '\101'为字符“A”的八进制数常量；
- '\x41'为字符“A”的十进制数常量；
- 65 为字符“A”的十六进制数常量。

4 种方式均可以表示字母“A”的常量值。

(4) 符号常量。

为了增加 C 程序设计实现过程的可维护性，可以使用一个标识符来代表一个常量，称为符号常量。符号常量是在一个程序中指定一个标识符，用以代表一个常量。符号常量的定义形式为：

```
#define 符号常量标识符 常量字串
```

其中 #define 是一条预处理命令，预处理命令均以“#”开头，也称为宏定义命令，该命令的功能是把标识符定义为随后的常量字串，程序编译时，在程序中所有出现符号常量的位置均代换以其后的常量字串。C 语言程序设计习惯上符号常量名用大写标识，普通变量名用小写标识。符号常量一经定义，程序编译执行时，在程序中所有出现该标识符的地方