第 3 章　基本数据类型、运算符与表达式

✧ 学习意义

学习 C 语言的最终目的就是能编写程序来解决实际问题，那么什么是程序呢？从功能上来讲，程序是解决某种问题的一组指令的有序集合；从结构上来讲，一个程序应包括对数据的描述和对数据处理的描述。著名计算机科学家沃思（Nikiklaus Wirth）提出一个公式：**程序＝数据结构＋算法**，这里的数据结构其实就是指对数据的描述，数据结构是计算机学科的核心课程之一，有许多专门著作论述，本书不再赘述。在 C 语言中，系统提供的数据结构是以数据类型的形式体现出来的，而算法其实就是对数据处理的描述，算法是为解决一个问题而采取的方法和步骤，是程序的灵魂。

因此，要学好 C 语言并用 C 语言来编写程序首先必须十分了解和熟练掌握 C 语言中的数据类型描述以及运算符与表达式，这是学习 C 语言的重要基础，后续章节的内容都是在此基础上而展开的。

✧ 学习目标

（1）掌握变量和常量的概念；

（2）理解各种类型的数据在内存中的存放形式；

（3）掌握各种类型数据的常量的使用方法；

（4）掌握各种整型、字符型、浮点型变量的定义和引用方法；

（5）了解调用 printf 函数输出各种类型数据的方法；

（6）掌握数据类型转换的规则以及强制数据类型转换的方法；

（7）掌握赋值运算符、算术运算符、位运算符、逗号运算符及 sizeof 的使用方法；

（8）理解运算符的优先级和结合性的概念，记住所学的各种运算符的优先级关系和结合性。

✧ 难点提示

（1）数据在内存中的表示；

（2）有符号数与无符号数；

（3）数据类型的自动转换与强制类型转换。

3.1 C 语言的数据类型

C 语言程序在处理数据之前，要求数据必须具有明确的数据类型。因此，C 语言是一种强类型语言。所谓数据类型是按被说明量的性质、表示形式、占据存储空间的多少及构造特点来划分的。在 C 语言中，数据类型可分为基本数据类型、构造数据类型、指针类型、空类型四大类。

1. 基本数据类型

基本数据类型最主要的特点是其值不可以再分解为其他类型。也就是说，基本数据类型是自我说明的。在 C 语言中，基本数据类型主要有整型（短整型、长整型）、字符型、实型（单精度实型、双精度实型）三种。这是本章讨论的重点。

2. 构造数据类型

构造数据类型也叫复杂数据类型，是根据已定义的一个或多个数据类型用构造的方法来定义的。也就是说，一个构造数据类型的值可以分解成若干个“成员”或“元素”。每个“成员”都是一个基本数据类型或是一个构造数据类型。在 C 语言中，构造数据类型主要有数组类型、结构类型、联合体类型（共用体类型）、位域、枚举类型五种。关于构造数据类型将在第 11 章详细讨论。

3. 指针类型

指针是一种特殊的同时又是具有重要作用的数据类型。其值用来表示某个量在内存中的地址。虽然指针变量的取值类似于整型量，但这是两个类型完全不同的量，因此不能混为一谈。关于指针类型将在第 9 章详细讨论。

4. 空类型

空类型是从语法完整性的角度给出的一种数据类型，表示该处不需要具体的数据值，因而没有数据类型。其类型说明符为 void。关于空类型将在第 8 章详细介绍。

图 3-1 为 C 语言数据类型层次图。

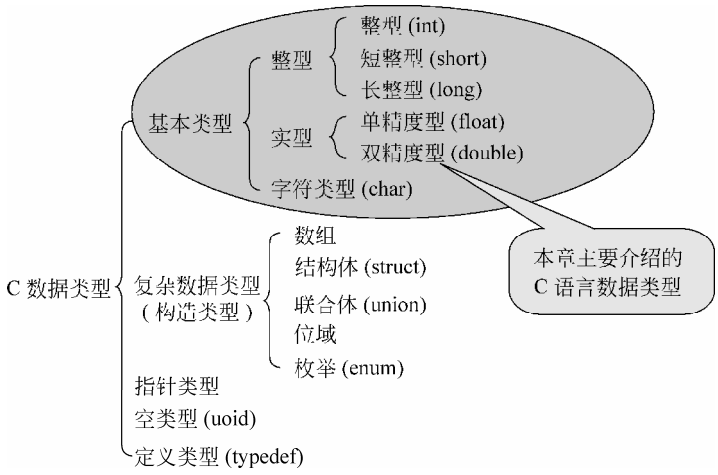

图 3-1　C 语言数据类型层次图

3.2 常量、变量和标识符

C 语言中存在着两种表征数据的形式：常量和变量。常量用来表示数据的值，变量不仅可以用来表示数据的值，而且可以用来存放数据，因为变量对应着一定的内存单元。但变量甚至常量以及后面要介绍的函数常常需要一个名字（即标识符）来表示才能使用，所以首先介绍标识符及其命名规则。

1. 标识符

1）标识符的定义

用来标识变量、常量、函数等的字符序列。

2）标识符的命名规则

（1）有效字符：只能由字母、数字、下划线组成，且第一个字母必须是字母或下划线。

（2）有效长度：最长 32 个字符。随系统而异，但至少前 8 个字符有效。如果超长，则超长部分被舍弃。

例如，由于 student_name 和 student_number 的前 8 个字符相同，有的系统认为这两个标识符是同一个标识符而不加区别。在 TC2.0 及 BC3.1 中，变量名（标识符）的有效长度为 1 ～32 个字符，默认值为 32，但在 VC 中其长度可达到 255。

（3）C 语言的关键字（又叫保留字）不能用作变量名（关键字见附录 C）。

（4）C 语言对英文字母的大小写敏感，即同一字母的大小写，被认为是两个不同的字符。

【思考题 3-1】

在 C 语言中，变量名 total 与变量名 TOTAL、ToTaL、tOtAl 等是同一个变量吗？

3）标识符的命名习惯

（1）变量名和函数名中的英文字母一般用小写，以增加可读性。

（2）见名知义，即通过变量名就知道变量值的含义。通常应选择能表示数据含义的英文单词（或缩写）作变量名，或汉语拼音字头作变量名。

例如，name/xm（姓名）、sex/xb（性别）、age/nl（年龄）、salary/gz（工资）。

（3）字符不易混淆。如数字 1 与字母 l、数字 0 与字母 O 等。

【思考题 3-2】

判断下列标识符号的合法性。

sum　Sum　M.D.John　day　Date　3days　student_name

#33　lotus_1_2_3　char　a＞b　_above　$123

2. 常量

1）常量的定义

程序运行时其值不能改变的量（即常数）。

2）常量的分类

（1）直接常量：又称值常量，主要包括整型常量（如 10、15、－10、－30）、实型常

量（如 12.5、30.0、－1.5）、字符常量（如'A'、'b'、'c'）、字符串常量（如"sum"、"A"、"123"）。

（2）符号常量：用标识符来代表常量。定义一个符号常量需要使用一条预处理命令#define，其定义格式为：

```
#define       符号常量          常量
```

例如：

```
#define   NUM   20
#define   PI    3.1415926
```

有了上面的两行文本，NUM 的值就是 20，PI 的值就是 3.1415926。定义一个符号常量，实际上就是为一个值常量起个名字。关于预处理命令#define 的详细用法见第 10 章。

【例 3-1】 符号常量举例。

```
1     #include  <stdio.h>
2     #define   PRICE   30
3
4     void main ( )
5     {
6       int num, total;
7
8       num = 10;
9       total = num * PRICE;     //PRICE 是符号常量，它代表常量 30
10      printf ("total = %d", total);
11    }
```

运行结果

```
total=300
```

注意：

- 在用#define 定义符号常量时，行尾一般不能有分号，除非特意这样做；
- define 前面的#标志着 define 是一个预处理命令而不是 C 语句；
- 符号常量名最好使用大写字符来命名；
- 符号常量名最好有意义，这样可提高程序的可读性。

使用符号常量的好处如下：

- 含义清楚。在例 3-1 的程序中，从字面上就可知道 PRICE 代表价格。因此定义符号常量名时应考虑“见名知义”。
- 在需要改变一个常量时能做到“一改全改”。如果在一个程序中多处用到物品的价格，若价格用常数表示，则在价格调整时，就需要在程序中作多处修改，很不方便，且易出错。若用符号常量 PRICE 代表价格，只需改动一处即可。如例 3-1 中的第 2 行，

如果将 30 改为 50，则在程序中所有以 PRICE 代表的价格就会一律自动改为 50。

3. 变量

1）变量的定义

在程序运行过程中，其值可以被改变的量称为变量。

2）变量的两个要素

（1）变量名：变量的名字，变量命名遵循标识符命名规则。

（2）变量值：变量存储在内存中的值。在程序中，通过变量名来引用变量的值。

3）变量的定义与初始化

在 C 语言中，要求对所有用到的变量，必须先定义后使用；在定义变量的同时进行赋初值的操作称为变量初始化。

（1）变量定义的一般格式：

```
[存储类型]    数据类型      变量名 1[，变量名 2，变量名 2，…，变量名 n]；
```

例如：

```
float  radius,  length,  area;
```

（2）变量初始化的一般格式：

```
[存储类型]    数据类型      变量名 1[=初值 1][，变量名 2[=初值 2] …]；
```

例如：

```
float radius=2.5,length,area;
```

注意：

- 变量和符号常量必须先定义后引用，否则就会出错；
- 变量定义以后，计算机会在程序运行时给该变量分配一定大小的内存单元，具体大小随变量定义的数据类型而定，后面的内容中会详细给出；
- 变量定义的位置一般放在函数开头；
- 对变量进行赋值实际上就是将数据写到变量所对应的内存单元中。

3.3 简单数据类型与表示范围

通过 3.2 节的学习，已经知道对于基本数据类型量来说，根据其取值是否可改变分为常量和变量两种。但如果与数据类型结合起来分类，则又可分为整型常量、整型变量、浮点常量、浮点变量、字符常量、字符变量等。本节将重点讨论 C 语言中的简单数据类型所表示的常量、变量以及变量在内存中的表示和数据表示范围。

3.3.1 整型数据

1. 整型常量

整型常量即整常数，在 C 语言中整型常量可用三种形式表示：

1）十进制整数

由数字 0～9 和正负号表示。以下各数是合法的十进制整常数：237，－568，65535，1627；以下各数不是合法的十进制整常数：023（不能有前导 0），23D（含有非十进制数码）。

2）八进制整数

由数字 0 开头，后跟数字 0～7 来表示。以下各数是合法的八进制数：015（十进制为 13），0101（十进制为 65），－012（十进制为－10）；以下各数不是合法的八进制数：256（无前缀 0），03A2（包含了非八进制数码），O127（前缀不能是字母 O）。

3）十六进制整数

由 0x 或 0X 开头，后跟 0～9，a～f 或 A～F 来表示。以下各数是合法的十六进制整常数：0X2A（十进制为 42），－0xA0（十进制为－160），0XFFFF（十进制为 65535）；以下各数不是合法的十六进制整常数：5A（无前缀 0X 或 0x），0X3H（含有非十六进制数码）。

有了上面三种整数表示方法，我们可以这样定义整数的符号常量：

```
#define    NUM1     20       //十进制数
#define    NUM2     020      //八进制数
#define    NUM3     0x2a     //十六进制数
```

其中常量 NUM1 的值是 20，常量 NUM2 的值是 16，常量 NUM3 的值是 42。

【思考题 3-3】

下列哪些整型常量是合法的？

012，oX7A，00，078，0x5Ac，－0xFFFF，0034，7B。

注意：

- 八进制整数和十六进制整数都是以数字 0 打头，不要写成字母 O 或 o;
- 定义十六进制整数时，0x 或 0X 后面的 a～f 或 A～F 大小写可以交错。例如，0x6Ac，0XFFbC 都是合法的。

2. 整数在内存中的表示

通过第 1 章的学习，已经了解到一个实际的数（即真值）在计算机中以二进制形式存放，其机内表示形式（即机器数）通常有三种：原码、补码和反码。根据它们的运算规则，补码运算最为简单，可连同符号位一起参与运算，也就是说对于补码运算，符号位和数据位一起只看待成一个二进制数值。所以计算机系统有这样的规定：整数的数值在内存中以补码的形式存放。

如何求得某个整数所对应的补码的方法在第 1 章中已经介绍过，下面用一种比较简单的办法来求一个整数的补码表示（假设用 *n* 个二进制位的内存单元来表示它）：

- 如果是正整数，采用符号-绝对值表示，即最高有效位（符号位）为 0 表示正，数的其余部分则表示数的绝对值；

- 如果是负整数，则先写出与该负数相对应的正数的补码表示，然后将其按位求反，最后在末位（最低位）加 1；
- 然后将上述求得的补码的低 *n* 位存放于内存单元之中，就得到了该整数在内存中的表示，内存单元的最高位是符号位（0 表示正，1 表示负）。

在 TC2.0 或 BC3.1 下，一个整数默认情况下需要两个字节（16 位）的内存单元存放；而在 VC6.0 下，则需要 4 个字节（32 位）。下面来具体看一看几种不同整数在内存中的存放形式。

1）＋14

对于 16 位的内存单元来说，$(14)_{补}$＝**0**000 0000 0000 1110，最前面的 **0** 为符号位，表示是正数。其表示形式如图 3-2 所示。

其在内存中的实际存放形式如图 3-3 所示。

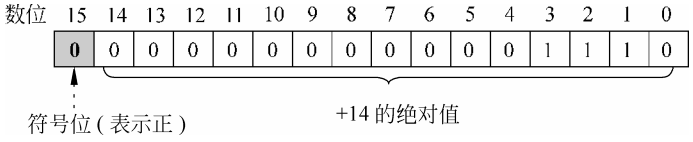

图 3-2　十进制数＋14 两个字节的补码表示形式

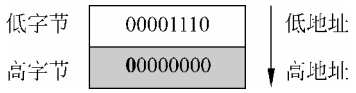

图 3-3　十进制数＋14 两个字节的内存实际存放形式

对于 32 位的内存单元来说，$(14)_{补}$＝**0**000 0000 0000 0000 0000 0000 0000 1110。其表示形式如图 3-4 所示。

其在内存中的实际存放形式如图 3-5 所示。

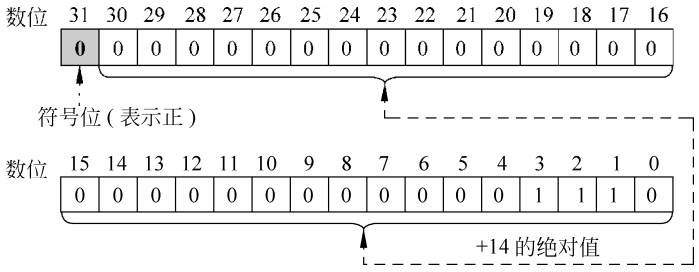

图 3-4　十进制数＋14 四个字节的补码表示形式

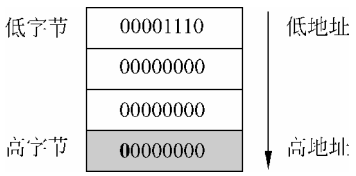

图 3-5　十进制数＋14 四个字节的内存实际存放形式

记住：数据在内存中的存放位置是高字节放在高地址的存储单元中，低字节放在低地址的存储单元中。

2）－14

对于 16 位的内存单元来说，先计算(＋14)$_{补}$＝**0**000 0000 0000 1110，然后按位求反，末位加 1 得(－14)$_{补}$＝**1**111 1111 1111 0010，最前面的 **1** 为符号位，表示是负数。其表示形式如图 3-6 所示。

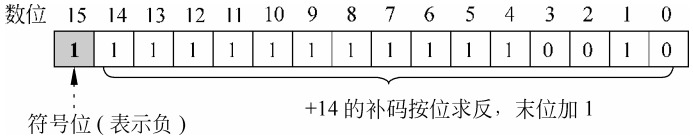

图 3-6　十进制数－14 两个字节的补码表示形式

其在内存中的实际存放形式可参考图 3-3。

对于 32 位的内存单元来说，同样先计算(＋14)$_{补}$＝**0**000 0000 0000 0000 0000 0000 0000 1110，然后按位求反，末位加 1 得(－14)$_{补}$＝**1**111 1111 1111 1111 1111 1111 1111 0010。其表示形式如图 3-7 所示。其在内存中的实际存放形式同样可参考图 3-5。

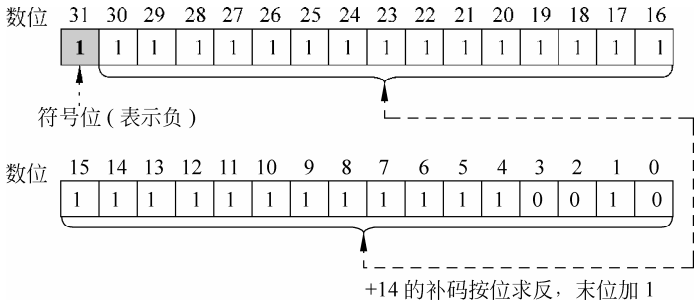

图 3-7　十进制数－14 四个字节的补码表示形式

3）－65537

对于 16 位的内存单元来说，先计算(＋65537)$_{补}$＝**0**1 0000 0000 0000 0001，然后按位求反，末位加 1 得(－65537)$_{补}$＝**1**0 1111 1111 1111 1111。再将(－65537)$_{补}$ 的低 16 位存放于内存之中，所以－65537 在内存中的实际存放的值为：**1**111 1111 1111 1111，最前面的 **1** 为符号位，表示是负数，其真值为－1，而不是－65537，也就是说，－65537 在内存中的值与－1 在内存中的值是一样的，这一点读者务必注意！－65537 在内存中的实际存放形式如图 3-8 所示。

对于 32 位的内存单元来说，同样先计算(＋65537)$_{补}$＝ 0000 0000 0000 0001 0000 0000 0000 0001，然后按位求反，末位加 1 得(－65537)$_{补}$＝**1**111 1111 1111 1110 1111 1111 1111 1111。最前面的 **1** 为符号位，表示是负数，其真值为－65537，其在内存中的实际存放形式如图 3-9 所示。

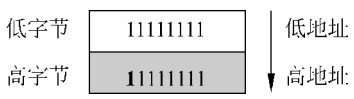

图 3-8　十进制数－65537 两个字节的内存实际存放形式

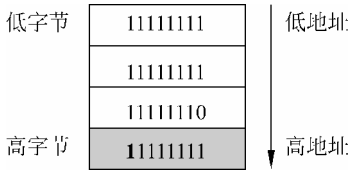

图 3-9　十进制数－65537 四个字节的内存实际存放形式

为什么－65537 这个数在 16 位内存单元中的表示与在 32 位内存单元中的表示不相同呢？这主要是因为－65537 这个数超出了 16 位内存单元表示数的范围，也就是发生了数据溢出，所以实际存储的值（－1）与要表示的值（－65537）不同，但－32767 并没有超出 32 位内存单元表示数的范围，所以实际存储的值就是其本身。因此，在 C 语言中对数据处理时必须要注意数据的表示范围，以免引起不必要的错误。

【思考题 3-4】

请问－0 和 0 在内存中如何表示？如果一个整数用 2 个字节的存储单元来存放，问－1 和 65535 在内存中表示的是同一个数吗？－65535 在内存中又如何表示呢？

3. 整型变量

1）整型变量的定义

定义变量的基本格式是“数据类型标识符　　变量名;”。整型数据类型标识符是 int（integer 整数），因此，定义整型变量的基本格式为：

```
int    变量名[，变量名 2，…，变量名 n];
```

格式说明如下：

- 整型类型名 int 必须小写；
- int 与变量名之间至少要用一个空格分开；
- int 后面可以一次定义多个变量，但变量名之间必须以逗号(,)隔开；
- 可以在变量定义时就对变量赋初值，具体方法是在变量名后面增加“＝数值”；
- 最后必须以分号(;)结尾；

例如：

```
int  a;
int  x, y, z;
int  m=2, n=-3;
```

当程序中定义了一个变量时，计算机会为这个变量分配一个相应大小的内存单元。因此，这个变量是有值的，它的值就是对应内存单元的值。但这个值程序员是无法预知的。

2）整型变量的分类

整型变量的基本类型符是 int。C 语言允许程序员在定义整型变量时，在 int 的前面增加两类修饰符：一类控制变量是否有符号，这类修饰符包括 signed（有符号）和 unsigned（无符号）；另一类控制整型变量的值域范围，这类修饰符包括 short 和 long。比如，可以这样定义一个无符号的长整型变量 a：

```
unsigned  long  int  a;
```

unsigned 和 long 都是数据类型修饰符。如果定义变量时，不指定 signed，也不指定 unsigned，则默认为 signed（有符号）。实际上，signed 修饰符完全可以不写。这样一来就形成了六种整型变量。

（1）有符号基本型（int）。

数据类型符为 int，占用字节数随机器不同而不同，一般占一个机器字，在 TC 2.0 或 BC 3.1 下，这种变量占用 2 个字节（16 位）的内存单元，在 VC 6.0 下，这种变量则占用 4 个字节（32 位）的内存单元。当该变量所对应的内存表示形式的最高位（即符号位）为 **1** 时，变量的值是负数，否则就是正数。

例如：int a＝－2；（定义一个有符号整型变量 a，并赋初值－2），其内存存放形式如图 3-10 所示（假设是在 TC 2.0 或 BC 3.1 下定义的）。

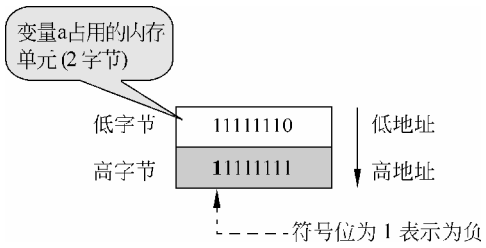

图 3-10 有符号整型变量在内存中的表示形式

（2）无符号基本型（unsigned int 或 unsigned）。

数据类型符为 unsigned int，也可写成 unsigned，占用字节数同 int 类型。但该变量所对应的内存表示形式的最高位不是符号位，而是数据位，参与数值计算。因此，unsigned int 型变量的值是非负数的。

例如：unsigned int b＝2；（定义一个无符号整型变量 b，并赋初值 2）它与 int b＝2；在内存中的表示形式是完全相同的，其值均为 2。但对于 unsigned int b＝－2；来说，其在内存中的表示形式与图 3-10 是相同的，可 b 的值不是负数－2（因为最高位 1 不是符号位，是数据位），而是 65534，是很大的一个数。

注意：有符号数也好还是无符号数也好，其实在计算机内存中表示是不加区分的（但运算时有区别，见 3.6 节），都是以其补码形式表示，只是我们怎样看待最高二进制位的问题，如果把最高位当成符号位看待，则为有符号数；如果把最高位当成数据位看待，则变为无符号数。例如：

```
unsigned int a=-2;
printf("%d", a);    //有符号输出，为-2
printf("%u", a);    //无符号输出，则为 65534（在 BC 下），4294967294（在 VC 下）
```

（3）有符号短整型（short int 或 short）。

数据类型符为 short int，也可写成 short，占用 2 个字节（16 位）的内存单元，最高位为符号位。在 TC 和 BC 中，short 类型与 int 类型是等价的，但在 VC 中 int 类型占 4 个字节，而 short 类型占 2 个字节。

（4）无符号短整型（unsigned short int 或 unsigned short）。

数据类型符为 unsigned short int，也可写成 unsigned short，占用的字节数同 short 类型，最高位为数据位。在 TC 和 BC 中，unsigned short 类型与 unsigned int 类型是等价的。

（5）有符号长整型（long int 或 long）。

数据类型符为 long int，也可写成 long，占用 4 个字节（32 位）的内存单元，最高位为符号位。在 VC 中 long 与 int 类型基本相同，但在 TC 或 BC 中，long 类型与 int 类型除所占字节不同外，其他数据处理方法是一样的。

下面给出 long int 类型变量定义的范例：

```
long  int  x=0xFF00FFCD;
long  y=215678;
```

（6）无符号长整型（unsigned long int 或 unsigned long）。

数据类型符为 unsigned long int，也可写成 unsigned long，占用 4 个字节（32 位）的内存单元。最高位不是符号位，而是数据位。unsigned long 类型与 long 类型的区别可参考 unsigned int 与 int 类型区别。

下面给出 unsigned long int 类型变量定义的范例：

```
unsigned  long  int  x=4389828;
unsigned  long  y=0XA5CD99AB;
```

在标准 C 语言程序中，变量的定义必须放在函数的声明部分，下面的例子说明了实际程序中如何来定义整型变量（行号不属于程序）。读者只需看懂有阴影的部分即可。

【例 3-2】 各种整型变量的定义。

```
1     include  <stdio.h>              //文件包含,头文件说明
2
3     #define   SUM   65535           //定义符号常量 SUM，值为 65535
4
5     void main ( )
6     {
7       int  a, b=20;                 //定义两个 int 型变量 a 和 b，b 赋初值 20
8       unsigned  int  c=0xff;        //定义无符号整型变量 c，并赋初值 0xff
9       long  D;                      //定义长整型变量 D
10
11      a=SUM;                        //对 a 赋值为 SUM，这时 a 的值是 65535
12      D=301;                        //对 D 赋值为 301
13
14      printf("a=%d\n", a);          //以有符号十进制形式（"%d"）显示 a 的值
15      printf("b=%d\n", b);          //以有符号十进制形式（"%d"）显示 b 的值
16      printf("c=%d\n", c);          //以有符号十进制形式（"%d"）显示 c 的值
17      printf("D=%d\n", D);          //以有符号十进制形式（"%d"）显示 D 的值
18    }
```

运行结果：

在 BC 3.1 下运行

```
a =   1
b = 20
c = 255
D = 301
```

在 VC 6.0 下运行

```
a = 65535
b = 20
c = 255
D = 301
```

对于16位的有符号整型变量a来说，因65535在内存中的形式为1111111111111111，最高位为1表示负，则其所对应的十进制数就为－1。