

第2章 数据类型、运算符和表达式

为了提高程序的运行速度，有效地利用存储空间，各种程序设计语言都将要处理的数据分成不同的数据类型。本章将介绍程序中可以使用的数据类型、常量、变量、运算符、表达式等基本元素的规定与使用。

2.1 C语言的数据类型

程序的执行过程实际上是对数据进行处理以得到正确结果的过程，而这些需要处理的数据以及处理数据过程中产生的中间数据，往往需要保存在内存中的某段存储单元内。对这些存储单元的使用，要求“先定义，后使用”。在定义中，规定了该量的名字、数据类型、存储类型以及作用域。系统会为不同数据类型的量分配不同大小的存储空间，按名称使用它们，并通过存储类型及作用域进一步规定了该量占据存储空间的时间与范围。

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，如图2-1所示。

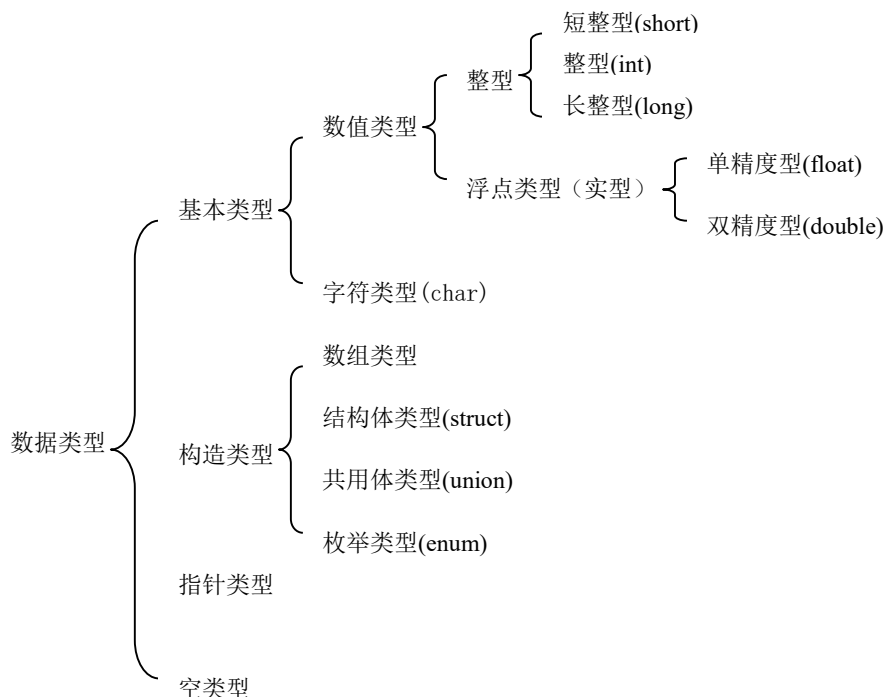


图 2-1 数据类型分类

1. 基本数据类型

是值不可再分的最基本的数据类型，包括整型、浮点型（实型）、字符型等。一个基本类型的值就是一个单个数据。

2. 构造类型

由已知的基本类型通过一定的构造方法构造出来的类型，包括数组、结构、联合、枚举等。一个构造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又是一个构造类型。

3. 指针类型

指针是存储单元地址的形象表示，指针类型的值用来表示某个数据在内存中的地址。通

过指针访问内存取得数据，访问效率更高，可以直接操作硬件。

4. 空类型

也叫无值类型，通常用来描述函数无返回值。调用后不需要向调用者返回函数值的函数，可以定义为“空类型”，其类型说明符为 `void`。

本书的前 5 章只涉及基本数据类型，构造数据类型将在第 6、9 章详细介绍，指针类型在第 8 章详细介绍。

2.2 标识符、常量和变量

对于基本数据类型量，按其取值是否可改变分为常量和变量两种。在程序执行过程中，其值不发生改变的量称为常量，其值可以改变的量称为变量。它们可与数据类型结合起来分类。例如，可分为整型常量、整型变量、浮点型常量、浮点型变量、字符型常量、字符型变量等。

在 C 程序中，直接常量是可以不经说明而直接引用的，而变量则必须先定义后使用。定义变量规定了变量的名称及类型，则编译时不仅能为其分配相应的存储单元，还能够检查出变量名是否正确使用、对该变量的运算是否合法。

1. 标识符

在 C 语言中，有许多需要命名的对象，如符号常量名、变量名、数组名、函数名、文件名、类型名等。标识符就是名字，用以区分不同的对象。

C 语言规定，标识符只能由英文字母、下划线、数字三种字符组成，且只能用字母和下划线开头。以下划线开头的标识符往往是系统变量或系统函数，由大写字母构成的标识符往往是符号常量，用户自定义的标识符尽量避免采用上述形式。

选择标识符时，应注意以下规则。

(1) 严格区分大小写，即大小写字母表示不同意义。例如：`sum` 和 `Sum` 代表不同的标识符。

(2) 标识符长度最好不要超过 8 个字符。因为有的 C 编译系统只识别 8 个字符，即前 8 个字符有效。例如：变量名 `information1` 和 `information2` 的前 8 个字符相同，C 编译系统认为是同一个变量。

(3) 尽量做到“见名知意”，即选择有含义的英文单词或其缩写词作标识符，以增加程序的可读性，如用 `count`、`score`、`total`、`age` 分别命名存储数目、分数、总计、年龄值的变量。

(4) 标识符不能使用 C 的关键字，这些关键字已被 C 系统使用，用户不能自定义使用。

由 ANSI 标准定义的关键字共 32 个，如下。

<code>auto</code>	<code>double</code>	<code>int</code>	<code>struct</code>	<code>break</code>	<code>else</code>
<code>long</code>	<code>switch</code>	<code>case</code>	<code>enum</code>	<code>register</code>	<code>typedef</code>
<code>char</code>	<code>extern</code>	<code>return</code>	<code>union</code>	<code>const</code>	<code>float</code>
<code>short</code>	<code>unsigned</code>	<code>continue</code>	<code>for</code>	<code>signed</code>	<code>void</code>
<code>default</code>	<code>goto</code>	<code>sizeof</code>	<code>volatile</code>	<code>do</code>	<code>if</code>
<code>while</code>	<code>static</code>				

下面举出几个正确和不正确的标识符。

正确	不正确
<code>price5</code>	<code>5price</code>
<code>_decision</code>	<code>bomb?</code>
<code>key_board</code>	<code>key.board</code>
<code>FLOAT</code>	<code>float</code>

2. 直接常量和符号常量

在C语言中，常量分为直接常量和符号常量两种。

(1) 直接常量

直接常量也叫字面常量，包括如下几种。

- ✧ 整型常量：0、23、-3。
- ✧ 浮点型常量：1.2、-3.4。
- ✧ 字符常量：'a'、'1'、'\n'、'\101'。
- ✧ 字符串常量："I am a student."、"2220131234"。

(2) 符号常量

可以用一个标识符代表一个常量，该标识符称为符号常量。

符号常量在使用之前必须先定义。有如下两种形式。

- ✧ #define 标识符 字符串

这是一条预处理命令——宏命令，功能是把标识符定义为其后的字符串。例如：

```
#define PI 3.14159
```

经过上面定义，以后程序中所有出现符号常量PI的地方，在预处理阶段都会被3.14159取代。

- ✧ const 类型 标识符=常数

定义指定类型的标识符（符号常量）的值，且该值在程序运行过程中不可改变，例如：

```
const double PI=3.14159;
```

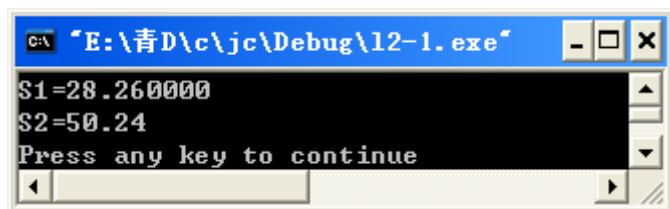
习惯上，符号常量的标识符用大写字母，以区别于一般变量。使用符号常量可以将复杂的长字符串用简洁的标识符代替，而且能够做到“一改全改”。

【例 2.1】符号常量的使用。

程序如下：

```
#define PAI 3.14
#include<stdio.h>
void main()
{
    float r1=3,r2=4,s1,s2;
    s1=r1*r1*PAI;
    s2=r2*r2*PAI;
    printf("S1=%f\nS2=%.2f\n",s1,s2);
}
```

运行结果为：



本例功能为求半径为3和4的圆面积。因为面积值为实型数，所以定义为float型，对应的输出格式符为“%f”，默认数据保留6位小数，在“%”和“f”之间加“.2”，则输出时数据保留两位小数。

3. 变量

变量的值在程序运行中是可以改变的，即可以多次赋值。C语言严格要求变量“先定义，后使用”。

变量定义语句格式:

<数据类型><变量名>[, <变量名>[, <变量名>……]];

例如: int count;
 float x, y;
 double result;
 char cc;

变量定义规定了被定义量的名称、性质、表示形式、占据存储空间的多少以及构造特点。编译系统为不同类型变量分配规定大小的存储空间, 例如, 在 Visual C++6.0 中, 为每一个 int 型变量分配 4 字节存储空间, 以后使用变量名访问该段存储单元, 该变量的值只能是整型数据, 不能超出-2147483648~2147483647。

变量定义后, 其值不确定, 需要为其赋一个确定的值后才能使用, 例如:

```
int a;  
a=100;
```

也可以在定义变量的同时为其赋初值——初始化, 变量初始化的格式如下:

<数据类型><变量名>=<常量表达式>[, <变量名>=<常量表达式>[, <变量名>……]];

例如: int a=100;

在定义变量 a 的同时为其赋初值 100。

也可以部分赋初值, 例如:

```
int a=100, b, c=500;
```

变量定义语句一般放在函数体的开头部分。

2.3 整型数据

在计算机中, 整数是与其二进制的补码形式保存的。

整型数据包括整型常量和整型变量。

1. 整型数据的分类

整型变量的基本类型符为 int。按照数值的表示范围, 分为如下 3 类。

- ✧ 基本类型: 类型符为 int。
- ✧ 短整型: 类型符为 short int, 可简写成 short。
- ✧ 长整型: 类型符为 long int, 可简写成 long。

实际应用中, 变量的值常常是正的, 如学号、成绩、年龄等。为充分利用存储空间, 可省掉符号位的存储, 分为:

- ✧ 有符号型: 类型符为 signed。signed 可省略, 隐含为有符号型。
- ✧ 无符号型: 类型符为 unsigned。

因此, 归纳起来, 整型数据有如下 6 种形式。

- (1) 有符号基本整型: [signed] int。
- (2) 无符号基本整型: unsigned int。
- (3) 有符号短整型: [signed] short [int]。
- (4) 无符号短整型: unsigned short [int]。
- (5) 有符号长整型: [signed] long [int]。
- (6) 无符号长整型: unsigned long [int]。

上述的方括号表示其中内容可选, 选与不选含义一样。例如:

```
int a;                               /* a 被定义为有符号整型变量 */  
unsigned long c;                    /* c 被定义为无符号长整型变量 */
```

编译系统不同，各类整型数据长度的规定也不同。在 V++6.0C 中，短整型（short）为两个字节，整型（int）和长整型（long）为四个字节；在 Turbo C 中，短整型（short）和整型（int）为两个字节，长整型（long）为四个字节。C 语言对于各类数据的长度没有明确规定，只要长整型的长度大于或等于整型，整型的长度大于或等于短整型即可。VC++6.0 中，整型类型标识符、数据的长度和数值范围如表 2-1 所示。

使用无符号整数可以在长度不变的情况下扩大数值的表示范围。如果用两个字节存储数据，则+13 和-13 的表示形式为：

+13	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1
-13	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	

对于有符号数，最高位是数的符号，“0”代表“正”、“1”代表“负”，其余 15 位是尾数，用来存放数值，尾数最大可以表示的值为 $2^{15}-1$ 。对于无符号数，全部 16 位都用来表示数值，则尾数最大可以表示的值为 $2^{16}-1$ 。

表 2-1 VC++6.0 整型数据的长度、类型标识符和数值范围

	类型标识符	数据长度	数值范围
有符号整数	short	16 位	-32768~32767
	int	32 位	-2147483648~2147483647
	long	32 位	-2147483648~2147483647
无符号整数	unsigned short	16 位	0~65535
	unsigned int	32 位	0~4292967295
	unsigned long	32 位	0~4292967295

2. 整型常量的表示

整型常量就是整常数。

（1）整型常量的表示形式

在 C 语言中，整型常量有 3 种表示形式。

- ✧ 十进制数：如 220、-560、4590。
- ✧ 八进制数：以 0 作为前缀，如 06、0105、05777。
- ✧ 十六进制数：以 0X 或 0x 作为前缀，如 0X0D、0XFF、0x4e。

（2）整型常量的后缀

可以在整型常数的后面添加后缀来限定其类型。

- （1）加一个“L”或“l”字母表示该数为 long int 型数，如 22L、0773L、0Xae4l。
- （2）加一个“U”或“u”字母表示该数为 unsigned int 型数。

3. 整型量的定义与使用

在程序中，一个变量应该先定义，再为其赋初值，然后才能参与运算，而且在使用过程中其值不能超过该类型量的表示范围。

【例 2.2】整型量的定义与混合运算。

程序如下：

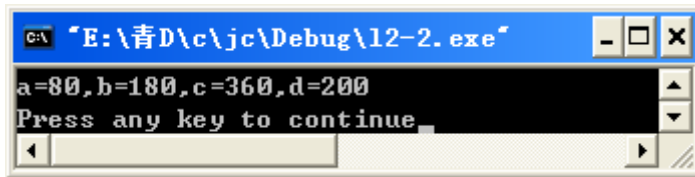
```
#include<stdio.h>
void main()
{
    int a,b;
```

```

    signed short int c;
    short d=100;
    a=d-20;
    b=a+d;
    c=a+b+d;
    d=d-a+c-b;
    printf("a=%d,b=%d,c=%d,d=%d\n",a,b,c,d);
}

```

运行结果为：



本例中，变量 c 和 d 都是有符号短整型，在相应的定义语句中省略了 signed 和 int。

【例 2.3】整型数据的溢出。

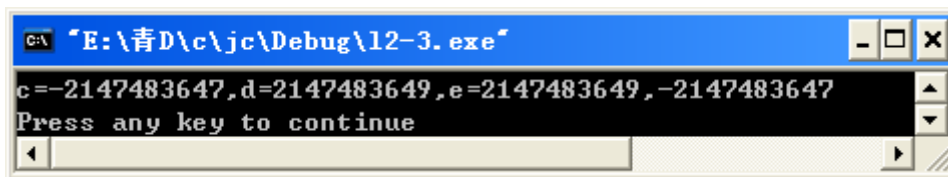
程序如下：

```

#include<stdio.h>
void main()
{
    int a=2147483647;
    unsigned int b=2,c,d;
    int e;
    c=a+b;
    d=a+b;
    e=a+b;
    printf("c=%d,d=%u,e=%u,%d\n",c,d,e,e);
}

```

运行结果为：



为什么程序中 c 的输出结果是 -2147483647 呢？因为整数是以二进制的补码形式存储的，有符号整型量的数值范围是 -2147483648 ~ 2147483647。

变量 a 的值是 +2147483647。在计算机中的存储形式是：

0	1	1	1	1	1	1	1	1	1	1	1	1
---	---	---	-----	-----	-----	---	---	---	---	---	---	---	---	---	---

变量 b 的值是 2。在计算机中的存储形式是：

0	0	0	0	0	0	0	0	0	1	0
---	---	---	-----	-----	-----	---	---	---	---	---	---	---	---

a+b 的值为 2147483649，对于有符号整型数来说，超出了其表示范围，发生溢出错误。但是系统并不报告错误，继续按其内部规定操作，将 a+b 的值赋给 c，则 c 的值是：

1	0	0	0	0	0	0	0	0	0	0	1
---	---	---	-----	-----	-----	---	---	---	---	---	---	---	---	---

变量 c 的输出格式为 “%d”，规定按有符号十进制整型数输出。最高位是数的符号位，1 表示负数。对负数的补码再求补得到其原码，c 的原码是：

1	1	1	1	1	1	1	1	1	1	1	1
---	---	---	-----	-----	-----	---	---	---	---	---	---	---	---	---

所以，c 的输出结果是 -2147483647。
 变量 d 的输出格式为 “%u”，规定按无符号十进制整型数输出，所以系统认为没有符号位，所有二进制位都是数值，表示为：

1	0	0	0	0	0	0	0	0	1
---	---	---	-----	-----	-----	-----	-----	---	---	---	---	---	---	---

对应的十进制数的值是 2147483649。
 实际上，系统在进行整型数运算时，不论是什么类型都用补码进行运算，至于结果如何表示，取决于输出语句中的格式规定。

变量 e 以 “%u”、“%d” 两种格式输出。第一个 e 值的输出格式为 “%u”，所以 e 的输出结果与 d 一样。第二个 e 值的输出格式为 “%d”，所以输出结果也是 -2147483647。

- 由上面例子可以看出：
- (1) C 对数据类型检测不严格，不同类型的量可以参与运算并相互赋值。其中的类型转换是由编译系统自动完成的。有关类型转换的规则将在 2.6.5 中介绍。
 - (2) 当数值超过数据类型所能容纳的数值范围（溢出）时，系统不报错。

2.4 实型数据

实型也称浮点型。在 C 语言中，实型数只采用十进制形式表示，所有实型数据均为有符号数，没有无符号的实型数据。

与整型数不同，浮点型数据是按照指数形式存储的。系统把一个浮点型数分成小数部分和指数部分分别存放。如 3.14159 在内存中的十进制表示形式如图 2-2，即 $+0.314159 \times 10^1$ 。

+	0.314159	1
数符	小数部分	指数

图 2-2 实型数在内存中的存储形式

1. 实型数的分类
- 实型数分为单精度 (float)、双精度 (double)、长双精度 (long double) 3 类。VC++6.0 中，浮点型量标识符、数据的长度和数值范围如表 2-2 所示。

表 2-2 VC++6.0 实型数据的长度、类型标识符和数值范围

类型标识符	数据长度	有效数字	数值范围
float	32 位	6~7 位	$-3.4 \times 10^{-38} \sim 3.4 \times 10^{38}$
double	64 位	15~16 位	$-1.7 \times 10^{-308} \sim 1.7 \times 10^{308}$
Long double	64 位	15~16 位	$-1.7 \times 10^{-308} \sim 1.7 \times 10^{308}$

2. 实型常量
- 浮点型常量只采用十进制表示，有十进制数形式和指数形式两种形式。
- (1) 十进制形式
- 由数码 0~9 和小数点组成，小数点不可省略。例如 +29.56、-56.33、-6.8、.034、30.。

(2) 指数形式

由十进制数加阶码标志“e”或“E”以及阶码（只能为整数，可以带符号）组成，其一般形式为：

$a E n$

表示 $a \times 10^n$ 。其中，a 为十进制数，不可省略；n 为十进制整数。例如：2.1E5 相当于 2.1×10^5 、3.7E-2 相当于 3.7×10^{-2} 、0.5E7、-2.8E-2、1e1、-.0015E-3 等。

这些浮点常量不合法：345（无小数点）、E7（阶码标志 E 前无数字）、-5（无阶码标志）、53.-E3（负号位置不对）、2.7E（无阶码）、1e2.3（阶码为小数）。

系统输出指数形式数据时，会自动转换为规范化形式输出，即小数点前保留一位有效数字，例如 12.34e2 规范化后写成 1.234e3。

注意，所有浮点常量都被默认为 double 型，要表示 float 型常量，可以在常量的末尾加后缀“f”或“F”。例如 34.5 为 double 型，34.5f 为 float 型。

3. 实型变量

实型变量定义的格式和书写规则与整型相同。例如：

```
float a, f;          /* a, f 被定义为单精度型变量 */
double b;            /* b 被定义为双精度型变量 */
```

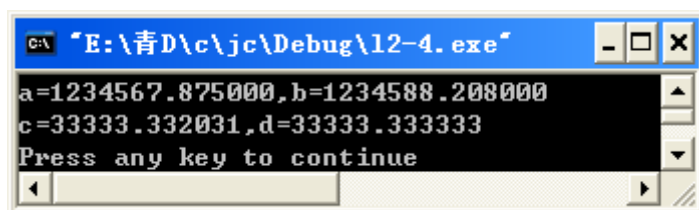
实型变量使用时要注意有效数字的位数，超过有效位数的值系统不能保证其准确性。

【例 2.4】浮点型变量数值的有效位数与舍入误差。

程序如下：

```
#include<stdio.h>
Void main()
{
    float a, b;
    float c=33333.33333;
    double d=33333.3333333333;
    a=1234567.814;
    b=a+20.333;
    printf("a=%f, b=%f\n", a, b);
    printf("c=%f, d=%f\n", c, d);
}
```

运行结果为：



单精度型量的有效数字为 7 位，第 8 位会有效参与运算，所以，a、b、c 变量的输出结果只能保证前 7 位是准确的。双精度型量的有效数字为 16 位，但 VC++6.0 规定小数最多保留 6 位，其余部分四舍五入，所以，d 变量的输出结果中小数部分只有 6 位。

2.5 字符型数据

字符型数据是指由中文字符、英文字符、数字字符或其他 ASCII 码字符构成的，通常不

需要计算的数据，字符长度为 1 个字符。

字符型数据是以字符的 ASCII 码值的二进制形式在计算机中保存的，二进制长度为一个字节。例如，大写字母 A 的 ASCII 码值为 65，保存在计算机中的形式是 01000001。

1. 字符常量

字符常量是用单引号括起来的一个字符。例如，'a'、'A'、' '、'?、'9'、'\n'等都是不同的合法字符常量。

字符常量也可以用转义字符表示。转义符必须以“\”开头，后面跟一个字符或该字符的 ASCII 码值的八进制或十六进制形式。转义字符具有特定的含义，不同于字符原有的意义，故称“转义”字符。例如，'\0102'（八进制）和'\X42'（十六进制）都表示大写字母'B'、'\\"'表示一个双引号、'\n'表示换行。常用转义字符及其含义如表 2-3 所示。

表 2-3 常用转义字符及其含义

转义符	等价形式	含义
\n	\012	换行，将光标当前位置移到下一行的开头
\r	\015	回车，将光标当前位置移到本行的开头
\t	\011	制表键，将光标当前位置跳到下一个制表位
\f	\014	换页，将光标当前位置移到下页开头
\b	\010	退格，将光标当前位置移到前一列
\\	\0134	一个反斜杠字符“\”
\'	\047	一个单引号字符“'”
\"	\042	一个双引号字符“””。
\ddd		1~3 位八进制数作为 ASCII 值所代表的字符
\xhh		x 开头的 1~2 位十六进制数作为 ASCII 值所代表的字符

C 语言字符集中的任何一个字符均可用转义字符来表示，只要知道该字符的 ASCII 码值，利用表 2-3 中的\ddd 和\xhh 格式就可以表达出来。例如，大写字母 A 的转义符表示形式为'\101' 或'\x41'。

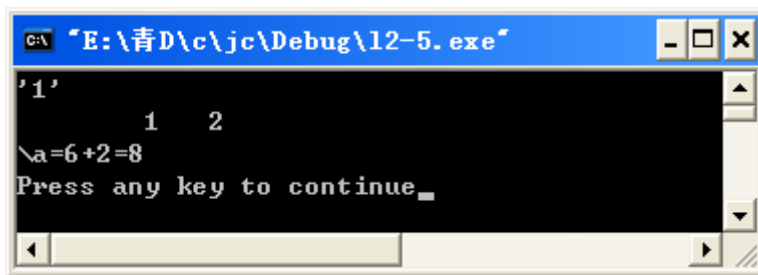
转义字符一般用来表示那些在代码中不便于表示的控制字符。

【例 2.5】 转义字符的使用。

程序如下：

```
#include<stdio.h>
void main()
{
    int a,b;
    a=1; b=2;
    printf("\' %d\' \n\t%d    %d\n",a,a,b);
    a=6+2;
    printf("\\\\141=6+2=\t\b%d\n",a);
}
```

运行结果为：



程序运行后,首先在第二列输出一个单引号,然后输出 a 的值 1,紧接着再输出一个“ ”,碰到“\n”换行;接下来的“\t”控制光标跳到下一制表位置,再输出 a 和 b 的值,碰到“\n”又换行;第二条 printf 语句先输出一个反斜杠 “\”, “\141” 是 a 的转义符,遇到“\t”跳到下一制表位(与上一行的 1 对齐),但下一转义字符“\b”又使其退回一格。

2. 字符变量

字符变量的取值是字符常量,即单个字符。字符变量定义的格式和书写规则都与整型变量相同。例如:

```
char a;
```

字符在计算机中以其 ASCII 码方式存储,所以也可以把它们看成是整型量。C 语言允许对整型变量赋以字符值,也允许对字符变量赋以整型值。在输出时,允许把字符变量按整型量输出,也允许把整型量按字符量输出。整型量占 4 个字节,字符型量占 1 个字节,当整型量按字符型量处理时,只有低 8 位参与处理。

字符型量在参与数值运算时,使用的是其 ASCII 码值。例如:有整型变量 a,执行语句 a='5'+5 后,变量 a 的值是字符'5'的 ASCII 值 53 加上 5,即 58。

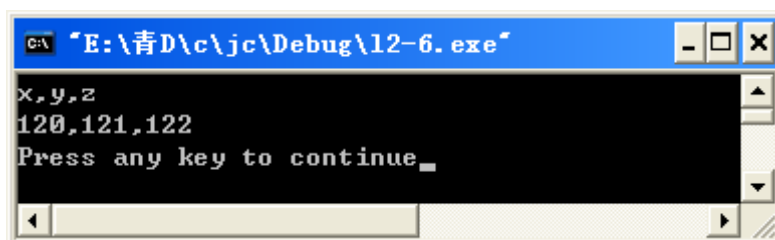
【例 2.6】字符变量的赋值与输出。

程序如下:

```
#include<stdio.h>

void main()
{
    char a,b;
    int c;
    a='x';b='\171';c=122;
    printf("%c,%c,%c\n%d,%d,%d\n",a,b,c,a,b,c);
}
```

运行结果为:



从运行结果可以看出,整型量与字符型可以通用,输出结果取决于输出格式。格式“%c”表示按字符输出,格式“%d”按十进制数输出字符的 ASCII 码值。

【例 2.7】大小写字母的转换。

程序如下:

```
#include<stdio.h>

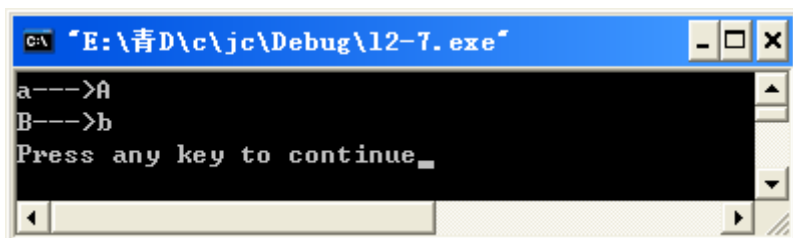
void main()
```

```

{
    int a='a',b='B',c,d;
    c=a-('a'-'A');
    d=b+32;
    printf("%c--->%c\n%c--->%c\n",a,c,b,d);
}

```

运行结果为：



本例中，'a'-'A'实际上就是大写与小写字母 ASCII 码值的差值 32，即 97-65。将某大写字母的 ASCII 码值加 32，即可转换得到其小写字母。

3. 字符串常量

字符串常量是由一对双引号括起来的字符序列。例如，"China"、"Turbo C2.0"、"3"、"\$12.5"、"中国"等都是合法的字符串常量。"3"因为使用了双引号，所以是字符串常量，'3'是字符常量。

字符串常量和字符常量是不同的量。它们之间主要有以下区别。

- (1) 字符常量由单引号括起来，字符串常量由双引号括起来。
- (2) 字符常量只能是单个字符，字符串常量则可以包含 0 个或多个字符。
- (3) 可以把一个字符常量赋予一个字符变量，但不能把一个字符串常量赋予一个字符变量。例如，语句 a="d"是不合法的。在 C 语言中没有相应的字符串变量，但是可以使用一个字符数组来存放一个字符串常量，将在第 6 章具体介绍。

(4) 字符常量占一个字节的内存空间。字符串常量占用的内存字节数等于字符串中字符串长度加 1。增加的一个字节中存放字符"\0" (ASCII 码值为 0 的字符，亦称空字符)。“\0”是字符串的结束标志，系统会自动在字符串末尾加空字符。例如，字符串 "China"在内存中所占的字节可表示为：

C	h	i	n	a	\0
---	---	---	---	---	----

字符常量'a'和字符串常量"a"虽然都只有一个字符，但在内存中的存储情况是不同的。

'A'在内存中占一个字节，表示为：

A

"A"在内存中两个字节，表示为：

A	\0
---	----

系统在对字符串进行操作时，遇到“\0”，即认为该字符串结束，因此，在字符串中要恰当使用字符“\0”。例如，执行语句 printf("this\0 is a student."); 后，输出结果为：this。

【例 2.8】 使用字符串常量打印图形。

程序如下：

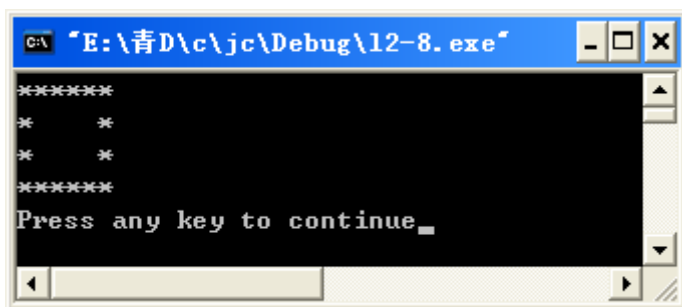
```

#include<stdio.h>
void main()
{
    char a='*';

```

```
printf("%s\n%c    %c\n%c    %c\n%s\n", "*****", a, a, a, a, "*****");
}
```

运行结果为：



输出格式“%s”表示输出字符串。

2.6 运算符及表达式

运算符丰富是 C 语言的一大特点。C 的运算符不仅具有优先级，还有结合性，两者制约了运算的优先顺序。将操作数用运算符连接起来就构成了表达式。

2.6.1 运算符概述

1. 运算符分类

根据运算符对操作数数量的要求，运算符可分为单目、双目、三目等几类。例如，取负号运算符“-”是单目运算符，-a 是对 a 进行一目求负操作；乘号“*”是双目运算符。

根据运算符的功能，可分为如下几类。

- (1) 算术运算符：+（加）、-（减）、*（乘）、/（除）、%（求余或取模）、--（自减 1）、++（自增 1）。
- (2) 关系运算符：>（大于）、<（小于）、==（等于）、>=（大于等于）、<=（小于等于）、!=（不等于）。
- (3) 逻辑运算符：！（逻辑非）、&&（逻辑与）、||（逻辑或）。
- (4) 位运算符：&（按位与）、|（按位或）、~（按位非）、^（按位异或）、<<（按位左移）、>>（按位右移）。
- (5) 赋值运算符：=（简单赋值）、复合算术赋值（+=，-=，*=，/=，%=）、复合位运算赋值（&=，|=，^=，>>=，<<=）。
- (6) 条件运算符：（? : ），是一个三目运算符。
- (7) 逗号运算符：（, ），用于把若干表达式组合成一个表达式。
- (8) 指针运算符：对内存直接操作，包括 *（取内容）和 &（取地址）。
- (9) 强制类型转换运算符：（类型符），用于数值类型的转换。

各类运算符的优先级与结合性，请参阅附录 C。

2. 运算符的优先级与结合性

所谓优先级就是不同运算符在表达式中参加运算的先后次序。C 语言中，在括号“（）”内的表达式先运算，其余的除了赋值运算符外，单目运算符的优先级高于双目运算符，而双目运算符的优先级又高于三目运算符。

一般，算术运算符的优先级>关系运算符的优先级>逻辑运算符的优先级>赋值运算符的优先级>逗号运算符的优先级。

如果表达式包含的运算符优先级别相同，运算方向是从左向右还是从右向左，与运算符的结合方向有关。除了单目运算符、条件运算符和赋值运算符是从右至左外，其他运算符的结合方向都是从左至右。例如：a=b+c+d 的运算顺序是先算 b+c 的值，算得的结果加上 c 的值，然后将结果赋值给变量 a。

2.6.2 算术运算符和算术表达式

算术运算符及其运算优先级如表 2-4 所示。

表 2-4 算术运算符及其优先级

运算符	作用	优先级是否相同	结合方向	优先级
-	取负	是	自右至左	高
--	自减 1			
++	自加 1			
*	乘	是	自左至右	
/	除			
%	取模（取余数）			
+	加	是		低
-	减			

1. 算术运算符的使用

（1）对于除法运算符“/”，如果参与运算的两个量均为整型，结果也为整型，会舍去小数；如果运算量中至少有一个为实型，则结果为双精度实型。

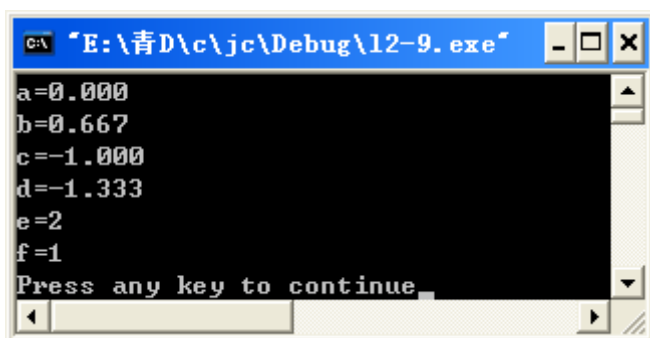
（2）对于取余运算符“%”，取余运算的结果等于两个数相除后的余数。要求参与运算的两个量均为整型。例如，6%5 的值为 1，65%10 的值为 5。

【例 2.9】除法与取余运算示例。

程序如下：

```
#include<stdio.h>
void main()
{
    float a,b,c,d;
    int e,f;
    a=2/3;
    b=2/3.0;
    c=-4/3;
    d=-4.0/3;
    e=2%3;
    f=7%3;
    printf("a=%.3f\nb=%.3f\nc=%.3f\nd=%.3f\n",a,b,c,d);
    printf("e=%d\nf=%d\n",e,f);
}
```

运行结果为：



运算前，C 系统要将参加运算的操作数转换成同一类型的数据，转换方向是向长度较长的数据类型转换，运算结果的数据类型就是转换后的数据类型。

2 和 3 均为整型数，2/3 的结果也为整型数，舍掉小数，所以 a 的值为 0。

2/3.0 中至少有一个为浮点数，运算结果也为浮点型数。所以 b 的值为 0.667。

不论是正数还是负数，大多数系统处理整除时都是舍去小数取整数部分作为运算结果的。所以 c 的值为-1。

(3) 除法运算符与取余运算符配合，可以拆分一个整型数得到其各个数位。

一般，一个整数 a 某数位的值为：

$$a/\text{位权}\%10$$

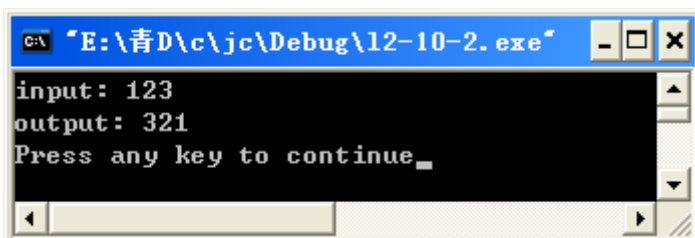
例如，3869 百位上的数值为 $3869 / 10^2 \% 10 = 8$ 。

【例 2.10】 将一个三位数倒序显示。

程序如下：

```
#include<stdio.h>
void main()
{
    int a,w0,w1,w2;
    printf("input:");
    scanf("%d",&a);
    w0=a%10;
    a=a/10;
    w1=a%10;
    a=a/10;
    w2=a;
    printf("output:%d%d%d\n",w0,w1,w2);
}
```

运行结果为：



程序执行后，从键盘输入 123 后回车，则输出 321。

各个数位的分解，在这段代码中，采用了相同的除 10 再取余 10 的运算，便于将来应用到循环程序中。分解后原数 a 被改变。也可以采用下面程序段，分解后 a 不变。

w0=a%10;

```
w1= a/10%10;
```

```
w2= a/100%10;
```

(4) 自加 1、自减 1 运算符只能用于变量，可以写在变量的前面也可以写在变量的后面，但两者的意义有所不同。以应用于变量 *i* 为例，有如下几种形式。

✧ ++*i*：变量 *i* 自加 1 后，再参与其他运算。

✧ --*i*：变量 *i* 自减 1 后，再参与其他运算。

✧ *i*++：变量 *i* 参与运算后，*i* 再自加 1。

✧ *i*--：变量 *i* 参与运算后，*i* 再自减 1。

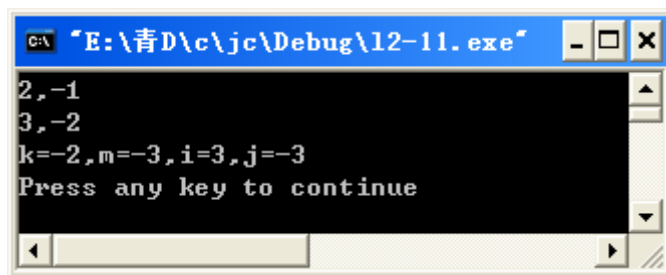
【例 2.11】自加 1、自减 1 运算符的功能示例。

程序如下：

```
#include<stdio.h>

void main()
{
    int i=2, j=-2, k, m;
    printf("%d, %d\n", i++, ++j);
    printf("%d, %d\n", i--, --j);
    k=-i++;
    m=--j;
    printf("k=%d, m=%d, i=%d, j=%d\n", k, m, i, j);
}
```

运行结果为：



第一行 printf 中，*i*++先输出 *i* 的值，*i* 再加 1；++*j* 是 *j* 先自加 1 后，再输出。所以输出结果为 2 和 -1。输出后，*i* 为 3，*j* 为 -1。

第二行 printf 中，*i*--先输出 *i* 的值，*i* 再减 1；--*j* 是 *j* 先自减 1 后，再输出。所以输出结果为 3 和 -2。输出后，*i* 为 2，*j* 为 -2。

语句 *k*=-*i*++；中，-*i*++要理解为-(*i*++)，因为“++”和“--”运算符只能用于变量，-*i* 是表达式。所以，*i* 的值先取负赋给 *k* 后再加 1，所以 *k* 的值为 -2，*i* 自加 1 后的值为 3。

(5) 由于 C 语言中的运算符、表达式书写和使用的规定不很严格，灵活性较大，所以，为避免产生歧义和误解，应注意书写形式，并适当使用括号。

【例 2.12】自加 1、自减 1 运算符的表达式求值。

程序如下：

```
#include<stdio.h>

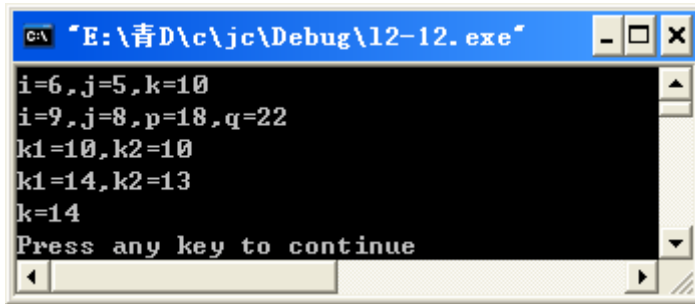
void main()
{
    int i=5, j=5, k, p, q;
    k=i+++j;
    printf("i=%d, j=%d, k=%d\n", i, j, k);
    p=(i++)+(i++)+(i++);
    q=(++j)+(++j)+(++j);
}
```

```

printf("i=%d, j=%d, p=%d, q=%d\n", i, j, p, q);
printf("k1=%d, k2=%d\n", k++, k++);
printf("k1=%d, k2=%d\n", ++k, ++k);
printf("k=%d\n", k);
}

```

运行结果为：



```

C:\E:\青D\c\jc\Debug\12-12.exe
i=6, j=5, k=10
i=9, j=8, p=18, q=22
k1=10, k2=10
k1=14, k2=13
k=14
Press any key to continue

```

$i++ + j$ 系统理解为 $(i++) + j$ ，而不是 $i + (++)j$ ，所以， $i + j$ 的值 10 赋给 k 后， i 再自增 1 变为 6。 $i++ + j$ 最好写成 $(i++) + j$ ，以避免误解。

对于 $p = (i++) + (i++) + (i++)$ 应理解为三个 i 相加，故 p 的值为 18，然后 i 再自加 1 三次， i 最后的值为 9。对于 $q = (++)j + (++)j + (++)j$ 则不然，不同的 C 编译器理解不同。在 Turbo C2.0 中，理解为 j 先自加 1 三次，变为 8，然后三个 8 相加， q 为 24；而在 VC++6.0 中 q 的值为 22。所以，在实际应用中，应力求不写 “ $q = (++)j + (++)j + (++)j$ ” 这样的代码，避免不同编译器下求值结果不一致情况的发生。可以改写为：

```

++j;
++j;
++j;
q=3*j;

```

第三、四条 `printf` 中的两个参数，同样有运算次序问题。函数参数的运算次序是从右至左。实际应用中也要注意避免误解，适当改写。

2. 算术表达式和算术运算符的优先级、结合性

表达式是将常量、变量、函数用运算符连接起来构成的式子。一个表达式的计算结果及其类型就是这个表达式的值和类型，表达式的求值按运算符的优先级和结合性规定顺序进行。用算术运算符和括号将运算对象连接起来构成的式子是算术表达式。例如， $a + b$ 、 $(a + 5) / c$ 、 $(-b + \text{sqr}(x)) / (2 * a)$ 等。

运算符的优先级指不同运算符同时出现时，运算的优先顺序。C 语言中，运算符的优先级共分 15 级，优先级别高的优于优先级别低的先计算。当一个运算量两侧的运算符优先级相同时，按运算符的结合性所规定的结合方向处理。结合性有左结合性和右结合性两种，大部分运算符是左结合的，典型的右结合运算符是赋值运算符。例如， $x = y = 10$ ，由于右结合性，应先执行 $y = 10$ ，再执行 $x = (y = 10)$ 运算。

2.6.3 赋值运算符和赋值表达式

1. 赋值运算符和赋值表达式

由赋值运算符 “ $=$ ” 连接的式子称为赋值表达式。一般形式为：

变量 = 表达式

“表达式”部分可以是任意类型的表达式，具体可以是常量、变量或者表达式。赋值运算符左侧必须是变量名。例如， $a+2=b+5$ 是错误的赋值表达式，下列表达式均为合法赋值表达式。

```
x=' 10'  
x=w  
y=sin(a)+b  
a=(b=(c=3))
```

赋值表达式的功能是计算表达式的值，然后将值赋给“=”号左面的变量。赋值运算符具有右结合性。

在进行赋值运算时，允许赋值运算符两边的数据类型不一致，但必须都是数值或字符型数据。系统会自动进行类型转换，以保证“=”号左面的变量类型不变，并尽量保证转换前后的值不变。例如：浮点数据赋值给整型变量时，将舍去小数部分；整型数据赋值给浮点型变量时，值不变，保存形式变为浮点型，即增加小数部分（小数部分值为0）。

【例 2.13】赋值表达式示例。

程序如下：

```
#include<stdio.h>  
void main()  
{  
    int a;  
    float b;  
    a=1234.56;  
    b=123;  
    printf("a=%d,b=%.2f\n",a,b);  
}
```

运行结果为：



浮点数 1234.56 赋给整型量 a 后，舍去小数部分。浮点型量的输出格式符是“%f”。

赋值表达式也可以出现在其他语句中。例如，语句 `printf("%d",a=b);` 中的赋值表达式 `a=b` 完成先将 b 的值赋给变量 a，然后输出 a 的值。

2. 复合赋值运算符

在赋值运算符“=”前加上某些二目运算符，构成复合赋值运算符。有+=、-=、*=、/=、%=、<<=、>>=、&=、^=、|=共 10 种。

例如：

$a+=b$	等价于 $a=a+b$
$a\%=b$	等价于 $a=a\%b$
$a/=b-c$	等价于 $a=a/(b-c)$

注意， $a/=b-c$ 理解为 $a=a/(b-c)$ ，运算符右侧的表达式应整体参与计算。

复合赋值运算符有利于编译处理，能提高编译效率，产生质量较高的目标代码。

2.6.4 逗号运算符和逗号表达式

逗号运算符“,”也称顺序求值运算符。用逗号运算符把多个表达式连接起来的式子,称为逗号表达式。逗号表达式的一般形式为:

表达式 1, 表达式 2, 表达式 3……

逗号表达式的求值顺序是从左至右顺序求表达式的值,最右边表达式的值作为整个表达式的值。

逗号运算符的优先级别在所有运算符中是最低的。例如:执行语句 $a=(1*2, 3*4);$ 后, a 的值为 12, 取第 2 个表达式 $3*4$ 的结果赋给 a 。若将语句改为 $a=1*2, 3*4;$ 形式, 因为逗号运算符的优先级低于赋值运算符, 所以 a 的值为 2, 。

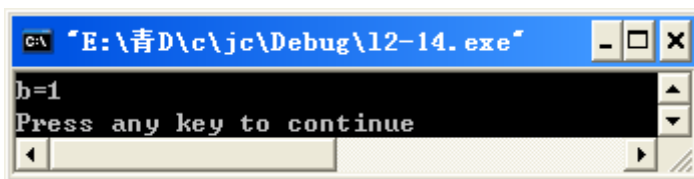
前面出现的 `printf` 函数中的“,”号是函数参数分隔符, 不是逗号运算符。

【例 2.14】逗号表达式的运算示例。

程序如下:

```
#include<stdio.h>
void main()
{
    int a, b;
    a=10;
    b=(a=a-5, a/5);
    printf("b=%d\n", b);
}
```

运行结果为:



本程序中, 先运算括号内的逗号表达式的值, 然后将其赋给变量 b 。执行 $a=a-5$ 后, a 的值为 5, 再执行 $a/5$ 结果为 1, 则逗号表达式的值为 1, 所以 b 的值为 1。

2.6.5 不同数据类型间的转换

在表达式中, 参与运算量的类型是可以转换的。转换的方法有两种: 一种是自动转换, 一种是强制转换。

1. 自动类型转换

自动类型转换发生在不同数据类型的量混合运算时, 由编译系统自动完成。自动转换遵循以下规则。

(1) 若参与运算量的类型不同, 则先转换成同一类型, 然后进行运算。运算结果的数据类型也就是转换后的数据类型。

(2) 转换按数据长度增加的方向进行, 以保证不降低精度。如 `int` 型和 `long` 型运算时, 先把 `int` 量转成 `long` 型后再进行运算。

(3) 所有的浮点运算都是以双精度进行的, 即使仅含 `float` 单精度量的表达式, 也要先转换成 `double` 型, 再作运算。

(4) `char` 型和 `short` 型参与运算时, 必须先转换成 `int` 型。

在赋值运算中，赋值号两边量的数据类型不同时，赋值号右边量的类型将转换为左边量的类型。如果右边量的数据类型长度比左边长，将丢失一部分数据，这样会降低精度，丢失的部分按四舍五入向前舍入。不同数据类型间的自动转换规则如图 2-3 所示。

2. 强制类型转换

强制类型转换是通过类型转换运算来实现的。一般形式为：

(类型说明符)(表达式)

其功能是把表达式的运算结果强制转换成类型说明符所表示的类型。

例如：

(float) a 把 a 的值转换为 float 型

(int) (x+y) 把 x+y 的结果转换为整型

在使用强制转换时应注意以下问题。

(1) 类型说明符和表达式都必须加括号(表达式是单个变量可以不加括号)。例如，如果将 (int) (x+y) 写成 (int)x+y，则变成将 x 转换成 int 型之后再与 y 相加。

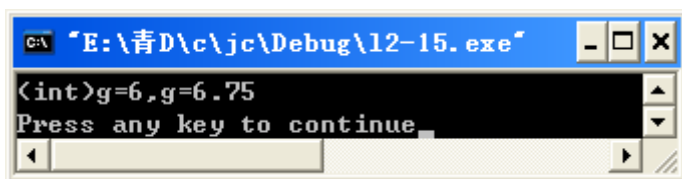
(2) 无论是强制转换还是自动转换，都只是为了本次运算的需要而对变量的数据类型进行的临时性转换，不改变数据定义时对该变量所定义的类型。

【例 2.15】强制类型转换示例。

程序如下：

```
#include<stdio.h>
void main()
{
    float g=6.75;
    printf("(int)g=%d, g=%.2f\n", (int)g, g);
}
```

运行结果为：



本例表明，g 虽经强制转换为 int 型，但只在运算中起作用，是临时的，而 g 本身的类型并不改变。因此，(int)g 的值为 6(截去了小数)，而 g 的值仍为 6.75。

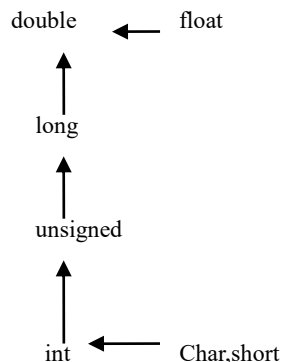


图 2-3 数据类型的自动转换

2.6.6 位运算

位运算是指对二进制位的运算，是对字节或字中的实际位进行检测、设置或移位。位运算只适用于字符型和整型变量以及它们的变体，对其它数据类型不适用。除了按位取反是单目运算符外，其他位运算符都是双目运算符。

按位运算符有如下 6 种。

&	按位与		按位或	^	按位异或
~	按位取反	>>	按位右移	<<	按位左移

按位运算的结果取决于两个操作数的每一个相对应的二进制位。运算规则如下。

- ✧ 与运算：两个二进制位均为 1 时，结果为 1，否则为 0。
- ✧ 或运算：两个二进制位均为 0 时，结果为 0，否则为 1。
- ✧ 异或运算：两个二进制位相同结果为 0，不相同结果为 1。
- ✧ 求反：将二进制位按位取反。即 1 变 0，0 变 1。
- ✧ 移位：将每一个二进制位向右或向左移动一位，移位后，一端的位被“挤掉”，另一端空出的位以 0 填补。只要值不溢出，左移一位相当于原值乘以 2；右移 n 位相当于原值除以 2^n 。

例如，求 $-14 \& 3$ 的值。假设机器内部用两个字节来表示数据，则结果为 2。计算过程如下。

1111111111110010	-----	-14 的补码
&) 0000000000000011	-----	3 的补码
0000000000000010	-----	-14&3 的补码

【例 2.16】 与和或运算的应用。

程序如下：

```
#include<stdio.h>
void main()
{
    int a=0x16,b=0xf,c;
    int d=0x30,e=0x07,f;
    c=a&b;
    f=d|e;
    printf("c=%#x,f=%#x\n",c,f);
}
```

运行结果为：



利用按位与，可以取出某个数的某几位二进制值或保留某个数的某几位二进制值。16 进制数 0x16 (00010110) 与 0xf (00001111) 相与，高 4 位被屏蔽掉，取得低 4 位，所以结果为 6 (00000110)。

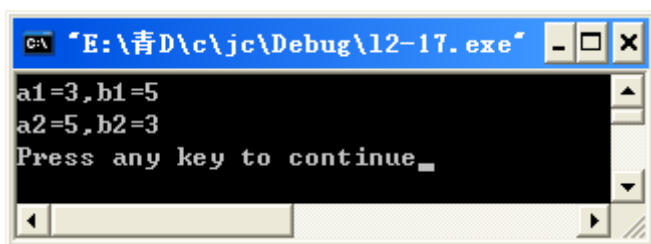
输出格式符 “%x” 规定按 16 进制输出数据，加 “#” 号，则数据会加上 16 进制数前缀 0x。

【例 2.17】 交换两个变量的值。

程序如下：

```
#include<stdio.h>
void main()
{
    int a=3,b=5;
    printf("a1=%d,b1=%d\n",a,b);
    a=a^b;
    b=b^a;
    a=a^b;
    printf("a2=%d,b2=%d\n",a,b);
}
```

运行结果为:



本程序中, 开始 a 的值为 3、b 的值为 5, 处理后, a 的值为 5、b 的值为 3。处理过程如下。

	0000000000000011	-----	3 的补码 (a)
^)	0000000000000101	-----	5 的补码 (b)
	0000000000000110	-----	6 的补码赋值给 a
^)	0000000000000101	-----	5 的补码 (b)
	0000000000000011	-----	3 的补码赋值给 b
^)	0000000000000110	-----	6 的补码 (a)
	0000000000000101	-----	5 的补码赋值给 a

利用异或, 可以将某个数的某些二进制位翻转, 即原来位值为 1 的变为 0, 原来位值为 0 的变为 1。如果将某个数异或 0xf, 则是将该数的后四位翻转。

习 题

一、选择题

- 以下选项中可作为 C 语言合法整数的是 ()。

A、1011B B、0383 C、0Xffb D、x3a
- 若函数中有定义语句 “int k;”, 则 ()。

A、系统将自动给 k 赋初值 0 B、这时 k 中的值无意义, 或称无定义

C、系统将自动给 k 赋初值-1 D、这时 k 中无任何值
- 设有定义 “int x=2;”, 下列表达式中, 值不为 6 的是 ()。

A、x*=x+1 B、x++, 2*x C、x*=(1+x) D、2*x, x+=2
- 若变量均已正确定义并赋值, 以下合法的 C 语言赋值表达式是 ()。

A、x=y==5 B、x=n%2.5 C、x+n=I D、x=5=4+i
- 设变量 x 为 float 型且已赋值, 则以下语句中能将 x 中的数值保留到小数点后一位, 并将第二位四舍五入的是 ()。

A、x=x*10+0.5/10.0; B、x=(x*10+0.5)/10.0;

C、x=(int)(x*10+0.5)/10.0; D、x=(x/10+0.5)*10.0;
- 已定义 ch 为字符型变量, 以下赋值语句中错误的是 ()。

A、ch='\'; B、ch=50+3; C、ch=NULL; D、ch='\xaa';
- 已定义 c 为字符型变量, 则下列语句中正确的是 ()。

A、c='107'; B、c="107"; C、c=107; D、c="k";
- 以下程序运行后的输出结果是 ()。


```
#include<stdio.h>
void main()
{ char m;
m='a'-32; printf("%c\n",m);
}
```

A、a B、A C、c=97 D、129

9. 若以下变量均是整型，且有语句 `num=sum=7;`，则执行表达式 `sum=num++`, `sum++`, `++num` 后，`sum` 的值是 ()。

A、7 B、8 C、9 D、10

10. 若有 `int k=7, x=12;`，则能使值为 3 的表达式是 ()。

A、`x%=(k%=5)` B、`x%=(k-k%5)`
C、`x%=k-k%5` D、`(x%=k)-(k%=5)`

11. 在 VC++6.0 中，为了计算 `s=10!`，定义变量 `s` 时应该使用的数据类型是 ()。

A、`int` B、`unsigned` C、`long` D、以上三种类型均可

12. 执行下面程序段的输出结果是 ()。

```
int x=023, y=5, z;
z=2+(y+=y++, x+8, ++x);
printf("%d,%d\n", x, z);
```

A、18, 13 B、19, 14 C、22, 21 D、20, 22

二、填空题

1. 程序中使用的各种变量都应先_____，后_____。

2. 基本数据类型有_____、_____、_____等。

3. 下面与 043 完全等值的项有_____。

(1) 15 (2) 0x23 (3) 3.5E+1 (4) 17+18
(5) 3.5E1 (6) 0X23 (7) 45 (8) 0x17+0xC

4. 指出下列哪些是 C 语言合法的标识符_____。

<code>hat_1</code>	<code>\$a^*</code>	<code>cat1</code>	<code>a11</code>	<code>SUM#</code>	<code>Dollar</code>
<code>2inf</code>	<code>s_exp</code>	<code>piece_f</code>	<code>SIN</code>	<code>union</code>	<code>float</code>
<code>_</code>	<code>2a</code>	<code>true</code>	<code>false</code>	<code>main</code>	<code>A\$?</code>

5. 指出下列哪些是 C 语言合法的常量，并指出其类型_____。

12, 345	'A'	08	005	3e0	o6
"a"	'\\'	'xo'	'\05'	e3	1.2e+5
0xf12	π	<code>sin(5)</code>			

三、简答题

1. 写出下列代数式对应的 C 语言表达式。

(1) $6 + \frac{(4+x)^2}{2+y}$

(2) $\frac{\sin^2(a+b)}{4x^2y}$

2. 若 `x` 为 `float` 型，其原值为 5，`a=2`, `b=4.7`；，写出下列表达式运算后 `x` 的值。

(1) `x=(int)(b-a)%3*a/4-a`
(2) `x=(x=b+1)+(int)(b)%10/2.0`
(3) `x+=x`
(4) `x-=x`
(5) `x*=x+x`
(6) `x/=x+x`
(7) `x+=x-=x*=x`
(8) `x%=x`
(9) `x=3*4, 5*6`

3. 写出下面程序的运行结果。

```

#include<stdio.h>
void main()
{
    int a=2;
    printf("abcdefghijk\n");
    printf("lmnop/n");
    printf("I am a /n beginner of C! \n ");
    printf("I am a \n beginner of C\n");
    printf("%d + %d = %d\n",a, a, a+a);
}

```

4. 写出下面程序的输出结果。

```

#include<stdio.h>
void main()
{
    int i, j;
    float s, b, a;
    char c;
    long m, n;
    i=5; j=-3;
    a=25.5; b=3.0;
    m=a/b; n=m+i/j;
    printf("%d\n", n);
}

```