第 2 章　C 程序的数据描述与计算

2.1　C 程序的词法

任何一种语言都有自己的符号、单字、单词及语句的构成规则。C 语言作为计算机的一种程序设计语言，也有自己的字符集、标识符及命名规则。只有学习、遵从它们，才能编写出符合要求的各种程序来。

2.1.1　字符集

字符是 C 语言最基本的元素，C 语言字符集由字母、数字、空白符、标点符号和特殊字符等组成(在字符串常量和注释中还可以使用汉字等其他图形符号)。由字符集中的字符可以构成 C 语言进一步的语法成分，如标识符、运算符等。

(1)字母：A～Z、a～z。

(2)数字：0～9。

(3)空白符：是空格、制表符(跳格)、换行符(空行)的总称。空白符除了在字符、字符串中有意义外，编译系统忽略其他位置的空白。空白符在程序中只是起到间隔作用。在程序的恰当位置使用空白符将使程序更加清晰，增强程序的可读性。

(4)标点符号、特殊字符。

(5)转义字符：\n(换行)、b(退格)等(后面介绍)。

2.1.2　标识符

C 语言中的标识符有关键字、预定义标识符、用户标识符等几种。

1. 关键字

C 语言系统设置的具有特定含义、专门用途的字符序列称为关键字。关键字不能用于其他用途，只能小写。例如，用来说明变量类型的标识符 int、double 及 if 语句中的 if、else 等都已有专门的用途，它们不能再用作变量名和函数名。

2. 预定义标识符

所谓的预定义标识符，是指在 C 语言中预先定义并具有特定含义的标识符，如 C 语言提供的库函数的名字(如 printf)和预编译处理命令(如 define)等。C 语言允许把这类标识符重新定义另作他用，但这将使这些标识符失去预先定义的原意。鉴于目前各种计算机系统的 C 语言都一致把这类标识符作为固定的库函数或预编译处理中的专门命令使用，因此，为了避免误解，建议用户不要把这类预定义标识符另作他用。

3. 用户标识符

由用户根据需要定义的标识符称为用户标识符，又称为自定义标识符。用户标识符一般用来给常量、变量、函数、数组、类型、文件等命名。

用户标识符命名规则如下：

(1)只能由字母、数字和下划线组成，且第一个字符必须为字母或下划线。

(2)有大小写之分。例如，sum、SUM 和 Sum 是 3 个不同的标识符。在 C 程序中，变量名用小写，常量名用大写，但不绝对。

(3)ANSI C 没有限制标识符长度，但各个编译系统都有自己的规定和限制。有的系统取 8 个字符，Turbo C 则允许取 32 个字符。

(4)标识符不能与关键字同名，最好也不与预定义标识符同名。

如果用户标识符与关键字相同，则在对程序进行编译时系统给出出错信息；如果用户标识符与预定义标识符相同，系统并不报错，只是该预定义标识符将失去原定含义，代之以用户确认的含义，这样可能会引发一些运行时的错误。

(5)标识符应当有一定的意义，做到见名知意，以增加程序的可读性。最好使用英文单词及其组合，便于记忆和阅读，尽量少用汉语拼音来命名。例如：

合法的用户标识符：a1、x2、s_1、s_2、_3、ggde2f_1。

不合法的用户标识符：df 1、1a、d@sina、s * b、+d。

2.2　常量与变量

在 C 语言中，基本数据按其取值是否可改变分为常量和变量，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、浮点常量、浮点变量、字符常量、字符变量、枚举常量和枚举变量。在程序中，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。

2.2.1　常量

在程序的运行过程中，其值不能改变的量称为常量。

1. 常量的类型

在 C 语言中，有整型常量、实型常量、字符型常量、字符串常量等。整型常量还可以进一步分为短整型常量、长整型常量等。

整型常量和实型常量又称为数值型常量，它们有正负之分。基本整型常量只用数字表示，不带小数点，如 12、0、－3 为整型常量。实型常量必须用带小数点的数表示，如 4.6、－1.23 为实型常量，'a' 和 'A' 为字符常量，"abc"和"hello"是字符串常量。

2. 符号常量

在 C 程序中，可以用一个符号名代表一个常量，称为符号常量。符号常量必须在程序中指定，并符合标识符的命名规则。用 define 定义时，前面必须以“＃”开头，命令行最后不加分号。

为了区别于一般的变量名，符号常量通常用大写字母表示。

【例 2-1】 计算圆的面积。

程序代码如下：

```
#include <stdio.h>
#define PI 3.14159
main()
{
    float r,area;
    r=5.0;
    area= PI * r * r;
    printf("area= %f\n",area);
}
```

程序的执行结果如下：

```
area=78.539750
```

程序中使用#define 命令行定义 PI 代表一串字符 3.14159。在对程序进行编译时，凡程序中出现 PI 的地方，编译程序均用 3.14159 来替换。

使用符号常量有如下好处：

(1)含义清楚，见名知意。

(2)修改方便，一改全改。

2.2.2 变量

在程序运行过程中，其值可以改变的量称为变量。程序中用到的所有变量都必须有一个名字作为标识，变量的名字由用户定义。

关于变量的使用，要注意以下几点：

(1)变量名必须符合标识符命名规则，一个变量实质代表内存中的某个存储单元。变量名在程序运行过程中不会改变，变量的值可以改变。

(2)C 语言中的变量必须“先定义，后使用”。对变量的定义通常放在函数体内的前部，但也可以放在函数体的外部或复合语句的开头。

①只有声明过的变量才可以在程序中使用，这使得变量名的拼写错误容易发现。

②声明的变量属于确定的类型，编译系统可方便地检查变量所进行运算的合法性。

③在编译时，根据变量类型可以为变量确定存储空间，“先定义，后使用”使程序效率高。

(3)像常量一样，变量也有整型变量、实型变量、字符型变量等不同类型。在定义变量的同时要说明其类型，以便系统在编译时能根据其类型为其分配相应的存储单元。

2.3 基本数据类型

C 语言提供了丰富的数据类型，其中最常用的是基本数据类型。不同类型的数据，其长度不同。在使用之前，必须先声明数据类型，以便为其分配相应的存储单元。下面依次介绍

整型、实型和字符型数据类型。

2.3.1　整型数据

1. 整型数据的分类

C 语言中的整型数据有短整型(short int)、基本整型(int)、长整型(long int)和无符号型(unsigned)。若不指定变量为无符号型,则变量默认为有符号型(signed)。

不同的编译系统或计算机系统对这几类整型数据所占用的字节数有不同的规定。表 2-2 所列为 ANSI 标准定义的整型数据类型。Turbo C 的规定与此表一致。

表 2-2　整型数据类型

类　型	类 型 名	字　节	取值范围
整型	int	2(16 位)	$-2^{15}\sim2^{15}-1$(−32 768～32 767)
短整型	short	2(16 位)	$-2^{15}\sim2^{15}-1$(−32 768～32 767)
长整型	long	4(32 位)	$-2^{31}\sim2^{31}-1$(−2 147 483 648～2 147 483 647)
无符号整型	unsigned int	2(16 位)	$0\sim2^{16}-1$(0～65 535)
无符号短整型	unsigned short	2(16 位)	$0\sim2^{16}-1$(0～65 535)
无符号长整型	unsigned long	4(32 位)	$0\sim2^{32}-1$(0～4 294 967 295)

2. 整型数据在内存中的存储形式

计算机内存储器的最小存储单位称为“位(bit)”。由于只能存放 0 或 1,因此称为二进制位。大多数计算机把 8 个二进制位组成 1“字节(byte)”,并给每字节分配一个地址。若干字节组成 1“字(word)”,用 1 字来存放一条机器指令或一个数据。

整型数据在内存中以二进制形式存放,事实上以补码形式存放。一个数据占若干字节,最高位为符号位(0 为正,1 为负)。

1)正整数

正整数的补码与原码形式相同。下表为 2 字节正整数最大值在内存中实际存放的情况,2 字节正整数最大值为 $2^{15}-1=32\ 768-1=32\ 767$。

0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

2)负整数

负整数按补码形式存放。负数求补码的方法是:将该数的绝对值二进制按位取反加 1。下表中最后一行的补码为 2 字节最小值在内存中实际存放的情况,2 字节最小值为 $-2^{15}=-32\ 768$。

原码	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
取反	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
加 1																1
补码	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

3)无符号整数

无符号整数按原码形式存放。下表为 2 字节无符号整数最大值在内存中实际存放的情况,2 字节最大值为 $2^{16}-1=65\ 536-1=65\ 535$。

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

C 标准没有具体规定各类整型数据所占用的字节数,只要求 long 型数据长度不短于 int 型,short 型不长于 int 型。具体如何实现,由各计算机系统自行决定。例如,Turbo C 中的 short、int 都是 16 位,long 是 32 位;VC++中的 int、long 都是 32 位,而 short 是 16 位。

3. 整型常量

值为整数的常量称为整型常量,简称为整常量或整数。它包括正整数、负整数和零。C 语言中的整型常量有以下 3 种表示形式:

(1)十进制:就是通常意义下的十进制整常数。例如,123、-456、0。

(2)八进制:以 0 开头,后面跟几位数字(由数字 0～7 组成)。例如,$0123=(123)_8=(83)_{10}$,$-011=(-11)_8=(-9)_{10}$。

(3)十六进制:以 0x 或 0X 开头,后面跟几位数字(数字由 0～9、A～F 组成)。例如,0x123=291,-0x12=-18。

整型常量有短整型、基本整型、长整型、无符号型等不同类型。整型常量后可以用 u 或 U 明确说明数据为无符号整数,用 l 或 L 明确说明为长整型数。例如,011L 表示长整型的八进制数,代表十进制数 9,在计算机中占 4 字节;011 表示整型的八进制数,也代表十进制数 9,但在计算机中占 2 字节。

4. 整型变量

整型变量可以分为短整型、基本整型、长整型、无符号型等不同类型。例如:

```
short s;        /* 定义 s 为短整型变量 */
int i,j,k;      /* 定义 i、j、k 为基本整型变量 */
long n;         /* 定义 n 为长整型变量 */
```

定义以上语句后,编译程序会为 s、i、j、k、n 分别开辟相应字节的空间,而没有在存储单元中存放任何值。此时变量中的值是无意义的,称变量值“无定义”。

C 语言规定,可以在定义变量的同时给变量赋初值,也称为初始化。例如:

```
int i,j,s=0;    /* 定义 i、j、s 为整型变量,s 初始化为 0 */
```

【例 2-2】 整型变量的定义与使用。

程序代码如下:

```
#include <stdio.h>
main()
{
    int a,b,c,d;
    unsigned u;
    a=12; b=-24; u=10;
    c=a+u; d=b+u;
```

```
    printf("%d,%d\n",c,d);
}
```

程序的运行结果如下：

22,－14

【例 2-3】 整型数据的运算。

程序代码如下：

```
#include <stdio.h>
main()
{
    int a,b;
    a=32767;
    b=a+1;
    printf("\na=%d,a+1=%d\n",a,b);
    a=-32768;
    b=a-1;
    printf("\na=%d,a-1=%d\n",a,b);
}
```

程序的运行结果如下：

a=32767,a+1=－32768

a=－32768,a－1=32767

注意：在 Turbo C 2.0 等环境中，一个整型变量只能容纳－32 768～32 767 范围内的数，无法表示大于 32 767 或小于－32 768 的数。遇到此情况就发生“溢出”现象，但运行时并不报错。因此，在进行计算时要尽量避免“临界数据的运算”。将变量 b 改成 long 型或在 VC++环境中运行就可以得到预期的结果。

2.3.2　实型数据

真题测试

1. 实型常量

实型常量又称为实数或浮点数。在 C 语言中，实型常量只能用十进制表示，有以下两种表示形式：

(1)十进制小数形式。由数字和小数点组成(必须有小数点)，如 123.0、.123(只有小数位)、123.(只有整数位)、0.0。

(2)指数形式(又称科学表示法)。这种表示形式包含数值部分和指数部分，数值部分的表示方法同十进制小数，指数部分是一个可正可负的整型数，这两部分用字母 e 或 E 连接起来。这种形式类似于数学中的指数形式。在数学中，一个数可以用幂的形式表示，如 123×10^3，在 C 语言中，123e3、1.23E5 都是它的合法表示。

使用指数形式表示很大或很小的数比较方便。

使用实型常量需要注意以下几点：

(1)字母 e 或 E 之前必须有数字，e 后面的指数必须为整数。例如，e3、2.1e3.5、.e3、e 都

不是合法的指数形式。

(2)规范化的指数形式。这种形式要求在字母 e 或 E 之前的小数部分,小数点左边应当有且只能有一位非零数字。例如,2.3478e2、3.0999E5、6.46832e12 都属于规范化的指数形式。

用指数形式输出实数时,都是按规范化的指数形式输出的。

(3)许多 C 编译系统将实型常量作为双精度实数来处理,这样可以保证较高的精度,缺点是运算速度降低。在实数的后面加字符 f 或 F,如 1.65f、654.87F,使编译系统按单精度处理实数。

2.实型变量

C 语言中的实型变量分为单精度型(float)、双精度型(double)和长双精度型(long double)。对于每一个实型变量也都应该先定义后使用,可以在定义变量的同时进行初始化。例如:

```
float  a,b;          /*定义 a、b 为单精度型变量*/
double  x,y,z;       /*定义 x、y、z 为双精度型变量*/
double t=1.0,s=0;    /*定义变量 t、s 为双精度型变量,同时进行了初始化*/
```

与整数存储方式不同,实型数据是按照指数形式存储的。系统将实型数据分为小数部分和指数部分分别存放。例如,很多编译系统以 24 位表示小数部分,8 位表示指数部分。小数部分占的位数多,实型数据的有效数字多,精度高;指数部分占的位数多,则表示的数值范围大。

标准 C 没有规定用多少位表示小数部分,多少位表示指数部分,由 C 编译系统自定。一般的单精度型数据占 4 字节,双精度型数据占 8 字节,长双精度型数据占 10 字节。实型变量是使用有限的存储单元存储的,因此能提供的有效位数是有限的,即实型数据的存储是有误差的。

【例 2-4】 实型数据的舍入误差(实型变量只能保证 7 位有效数字,后面的数字无意义)。

程序代码如下:

```
#include <stdio.h>
main()
{
    float a,b;
    a=123456.789e5;
    b=a+20;
    printf("a=%f,b=%f\n",a,b);
    printf("a=%e,b=%e\n",a,b);
}
```

程序的运行结果如下:

```
a=12345678848.000000,b=12345678848.000000
a=1.234568e+010,b=1.234568e+010
```

由于实数存在舍入误差,使用时要注意以下几点:

(1)不要试图用一个实数精确表示一个大整数,因为浮点数是不精确的。

(2)实数一般不判断“相等”,而是判断接近或近似。

(3)避免直接将一个很大的实数与一个很小的实数相加、相减,否则会“丢失”小的数。

(4)分析数据,根据需要选择数据类型是单精度还是双精度。

2.3.3　字符型数据

在处理数据时,经常会遇到像姓名、性别和住址等具有文本特征的数据,这些数据属于字符型数据。

1.字符常量

用单引号引起来的单个字符为字符常量。例如:

合法的字符常量:'a''A''5''0'',。

不合法的字符常量:"a"(双引号不合法)、'abc'(超过 1 个字符不能用单引号)。

字符常量在内存中占 1 字节,存放的是字符的 ASCII 码值。字符常量 'A' 的值为 65,字符常量 'a' 的值为 97。

2.转义字符常量

转义字符是以“\”开头的具有特殊含义的字符,这些字符常量也必须引在一对单引号内,如 '\n' 代表回车符。表 2-3 列出了 C 语言中的转义字符。

表 2-3　转义字符

字符形式	功　能	字符形式	功　能
\n	回车换行	\\	反斜杠字符
\t	横向跳过若干格(Tab 键)	\'	单引号字符
\v	竖向跳格	\"	双引号字符
\r	回车符	\ddd	3 位八进制数代表一个字符
\f	换页符	\xhh	两位十六进制数代表一个字符
\b	退格符(Backspace 键)	\0	空值

使用转义字符需要注意以下几点:

(1)转义字符常量,如 '\n''\101''\141' 只代表一个字符。

(2)反斜线后的八进制数可以不用 0 开头,如 '\101' 代表的就是字符常量 'A'。也就是说,在一对单引号内,可以用反斜线后跟一个八进制数来表示一个 ASCII 字符。

(3)反斜线后的十六进制数只能由小写字母 x 开头,不允许用大写的 X,也不能用 0x 开头,如 '\x41' 代表字符常量 'A'。也就是说,在一对单引号内,可以用反斜线后跟一个十六进制数来表示一个 ASCII 码字符。

3.字符串常量

字符串常量是由双引号括起来的一串字符,如"How are you ?""CHINA""a"。

C 语言规定:在每个字符串的结尾加一个字符串结束标志,以便系统据此判断字符串是

否结束。C 规定以 '\0'(ASCII 码为 0 的字符)作为字符串结束标志。

请注意字符常量与字符串的区别。例如,字符型常量 'a' 占 1 字节,而字符串常量"a"占 2 字节;'\n' 是转义字符,占 1 字节,"ab\n"是字符串常量,占 4 字节;""代表一个空串,占 1 字节存放 '\0'。

注意:字符串只能是常量,C 语言中没有字符串变量。

4. 可对字符量进行的运算

字符数据以 ASCII 码存储的形式与整数的存储形式类似,这使得字符型数据和整型数据之间可以通用(当作整型量)。例如:

'B'－'A'＝66－65＝1　　'a'＋1＝97＋1＝'b'

(1)利用算术运算把大写字母转换成小写字母或把小写字母转换成大写字母。例如:

'A'＋32＝65＋32＝97＝'a'　　'b'－32＝98－32＝66＝'B'

(2)通过算术运算把数字字符转换成整数值或把一位整数转换成数字字符。例如:

'9'－'0'＝57－48＝9　　4＋'0'＝4＋48＝52＝'4'

(3)字符量还可以进行关系运算。例如:

'a'＞'b'

因为在 ASCII 码表中,'a' 的值为 97,小于 'b' 的值 98,所以,关系运算的结果为“假”,此关系表达式的值为 0。

5. 字符型变量

字符型变量用来存放字符数据,同时只能存放一个字符。在 C 语言中,字符型变量用关键字 char 进行定义,在定义的同时也可以初始化。例如:

```
char c1,c2,c3;
char ch='A';
```

所有编译系统都规定以 1 字节来存放一个字符,或者说一个字符型变量在内存中占 1 字节。当把字符放入字符型变量时,字符型变量中的值就是该字符的 ASCII 码值,这使得字符型数据和整型数据之间可以通用(当作整型量)。具体表现如下:

(1)可以将整型数据赋值给字符变量,也可以将字符数据赋值给整型变量。

(2)可以对字符数据进行算术运算,相当于对它们的 ASCII 码进行算术运算。

(3)一个字符数据既可以以字符形式输出(ASCII 码对应的字符),也可以以整数形式输出(直接输出 ASCII 码)。

【例 2-5】 大小写字母的转换。

程序代码如下:

```
#include <stdio.h>
main()
{
    char c1,c2,c3;
    c1='a';
    c2='b';
    c1=c1-32;
```

```
    c2=c2-32;
    printf("\n%c %c\n",c1,c2);
    printf("%d %d\n",c1,c2);
}
```

程序的运行结果如下：

```
A B
65 66
```

程序的作用是将两个小写字母转换为大写字母。在 ASCII 码表中，小写字母比对应的大写字母的 ASCII 码大 32。C 语言允许字符型数据与整数直接进行算术运算，字符数据既可以字符形式输出，也可以整数形式输出。

2.4　C 语言的运算符和表达式

C 语言的运算符非常丰富，除了控制语句和输入/输出以外，几乎所有的基本操作都作为运算符处理，所以，C 语言运算符可以看作操作符。C 语言丰富的运算符构成了 C 语言丰富的表达式。

在 C 语言中除了提供一般高级语言的算术运算符、关系运算符和逻辑运算符外，还提供赋值运算符、位操作运算符、自增自减运算符等，甚至数组下标、函数调用都可作为运算符。本节主要介绍算术运算符(包括自增自减运算符)、赋值运算符、逗号运算符，其他运算符在以后相关章节中将陆续进行介绍。

2.4.1　算术运算符与算术表达式

1. 算术运算符

常见的算术运算符包括＋、－、*、/、%和正负号，主要对数值型数据进行一般的算术运算，其运算规则、运算对象、结合性如表 2-4 所列。

表 2-4　　算术运算符及其运算规则

对象个数	名　　称	运 算 符	运算规则	运算对象类型	结果类型	结 合 性
单目	正	＋	取正值	整型或实型	整型或实型	自右向左
	负	－	取负值			
双目	加	＋	加法	整型或实型	整型或实型	自左向右
	减	－	减法			
	乘	*	乘法			
	除	/	除法			
	模(求余)	%	整数取余	整型	整型	

有关算术运算符需要说明以下几点：

(1)除法运算符/，如果两个整数相除，结果则为整数，如 5/3 的结果为 1，舍去小数部分。

如果两个运算对象中至少有一个是实型，那么结果就是实型。

(2)如果参加＋、－、＊、/运算的两个数有一个为实数，则结果为 double 型，因为所有实数都按 double 型进行计算。

(3)求余运算符%，要求两个操作数均为整型，结果为两数相除所得的余数。求余也称为求模。一般余数的符号与被除数符号相同。例如，8%5＝3，－8%5＝－3，8%－5＝3。

(4)双目运算符优先级，＊、/、%同级，＋、－同级，且前三个都高于后两个。

2. 算术表达式

算术表达式是指用算术运算符和括号将运算对象(也称操作数)连接起来的、符合 C 语法规则的表达式。运算对象可以是常量、变量、函数等。例如，a＊b/c－1.5＋'a'。

需要注意的是，C 语言算术表达式与数学表达式的书写形式有一定的区别，具体如下：

(1)C 语言算术表达式的乘号(＊)不能省略。例如，数学表达式 b^2-4ac 对应的 C 表达式应该写成"b＊b－4＊a＊c"。

(2)C 语言表达式中只能出现字符集允许的字符。例如，数学表达式 πr^2 对应的 C 表达式应该写成"PI＊r＊r"(其中 PI 是已经定义的符号常量)。

(3)C 语言算术表达式不允许有分子分母的形式。

(4)C 语言算术表达式只使用圆括号改变运算的优先顺序(不要使用{}、[])。可以使用多层圆括号，此时，左、右括号必须配对，运算时从内层括号开始，由内向外依次计算表达式的值。

3. 运算符的优先级与结合性

C 语言规定了进行表达式求值过程中各运算符的优先级和结合性。

(1)运算符的优先级。在表达式求值时，按运算符的优先级高低次序执行。例如，表达式 a－b＊c 等价于 a－(b＊c)，“＊”运算符的优先级高于“－”运算符。

(2)运算符的结合性。若在一个运算对象两侧的运算符的优先级相同，则按规定的结合方向处理。有以下两种结合方向：

左结合性(自左向右结合方向)：运算对象先与左侧的运算符结合。

右结合性(自右向左结合方向)：运算对象先与右侧的运算符结合。

(3)在书写多个运算符的表达式时，应当注意各个运算符的优先级，确保表达式中的运算符能以正确的顺序参与运算。对于复杂表达式，为了清晰起见，可以加圆括号“()”强制规定计算顺序。在算术表达式中，可使用多层圆括号，但左右括号必须配对。运算时，从内层圆括号开始，由内向外依次计算表达式的值。

2.4.2 赋值运算符和表达式

在 C 语言中，可以通过赋值运算直接为变量提供数据。赋值运算是 C 程序中使用最为广泛的一种运算。

1. 赋值运算符

赋值运算符用赋值符号(＝)表示，它的作用就是将一个数据赋给一个变量。

2. 赋值表达式

赋值表达式用于计算右边表达式的值，把右边表达式的值赋给左边的变量。其格式如下：

变量＝表达式

例如，表达式 x＝y 的作用是将变量 y 存储单元中的数据赋给变量 x，x 中原有的数据被替换掉。赋值后，变量 y 中的内容不变。

需要说明的是：

(1)赋值运算符左边必须是变量，右边可以是常量、变量、函数调用或是由常量、变量、函数调用组成的表达式。例如，x＝10、y＝x＋10、y＝func()都是合法的赋值表达式，而 a＋b＝c 是非法的。

(2)赋值符号“＝”不同于数学的等号，它没有相等的含义，而是进行“赋予”操作。

例如，在 C 语言中，x＝x＋1 是合法的(数学上不合法)，它的含义是取出变量 x 的值加 1，再存放到变量 x 中。

(3)赋值运算时，当赋值运算符两边的数据类型不同时，将由系统自动进行类型转换。转换原则为先将赋值号右边表达式类型转换为左边变量的类型，然后赋值。具体如下：

①将实型数据(单、双精度)赋给整型变量，舍弃实数的小数部分。

②将整型数据赋给单、双精度实型变量，数值不变，但以浮点数形式存储到变量中。

③将 double 型数据赋给 float 型变量时，截取其前面 7 位有效数字，存放到 float 型变量的存储单元中(32 位)。但应注意数值范围不能溢出。将 float 型数据赋给 double 型变量时，数值不变，有效位数扩展到 64 位。

④字符型数据赋给整型变量时，由于字符只占 1 字节，而整型变量为 2 字节，因此，将字符数据(8 位)放到整型变量低 8 位中。

⑤将带符号的整型数据(int 型)赋给 long 型变量时，要进行符号扩展，即将整型数的 16 位送到 long 型的低 16 位中，若 int 型数值为正，则 long 型变量的高 16 位补 0，若 int 型数值为负，则 long 型变量的高 16 位补 1，以保证数值不变。反之，若将一个 long 型数据赋给一个 int 型变量，则只将 long 型数据中低 16 位原封不动地送到整型变量(截断)中。

⑥将 unsigned int 型数据赋给 long int 型变量时，不存在符号扩展问题，只要将高位补 0 即可。将一个 unsigned 型数据赋给一个占字节相同的整型变量时，将 unsigned 型变量的内容原样送到非 unsigned 型变量中，但若数据范围超过相应整数的范围，则会出现数据错误。

(4)C 语言的赋值符号(＝)除了表示一个赋值操作外，还是一个运算符，也就是说，赋值运算符完成赋值操作后，整个赋值表达式还会产生一个所赋的值，这个值还可以利用。

赋值表达式的求解过程如下：

①计算赋值运算符右侧的表达式的值。

②将赋值运算符右侧表达式的值赋值给左侧的变量。

③整个赋值表达式的值就是被赋值变量的值。

例如，分析表达式 x＝y＝z＝3＋5。

根据优先级：原式→x＝y＝z＝(3＋5)。

根据结合性(自右向左)：x＝y＝z＝(3＋5)→x＝(y＝(z＝(3＋5)))→x＝(y＝(z＝3＋5))。

z＝3＋5：先计算 3＋5，将结果 8 赋值给变量 z，z 的值为 8，z＝3＋5 整个赋值表达式的值为 8。

y＝(z＝3＋5)：将上面 z＝3＋5 整个赋值表达式的值 8 赋给变量 x，x 的值为 8，(y＝(z＝3＋5))整个赋值表达式的值为 8。

x=(y=(z=3+5));将上面 y=(z=3+5)整个赋值表达式的值 8 赋给变量 x,x 的值为 8,整个表达式 x=(y=(z=3+5))的值为 8。

最后,x、y、z 都等于 8。

(5)赋值运算符的优先级只高于逗号表达式,比其他运算符的优先级都低,且具有自右向左的结合性。

3. 复合赋值表达式

复合赋值运算符由一个双目运算符和一个赋值运算符构成。C 语言规定可以使用 10 种复合赋值运算符,其中与算术有关的复合赋值运算符有+=、-=、*=、/=、%=(注意:两个符号之间不能有空格)。

例如,n+=1 与 n=n+1 等价;n*=m+3 与 n=n*(m+3)等价。a+=a-=a+a 等价于 a=a+(a=a-(a+a))。若 a 为 9,则表达式的值为-18。

采用复合赋值运算符一是为了简化程序,使程序简练,二是为了提高编译效率。

注意:赋值运算符、复合赋值运算符的优先级比算术运算符低。

2.4.3 自增、自减运算符和表达式

自增、自减运算符是 C 语言中最具特色的两个单目运算符,其操作对象只有一个,这两个运算符既可以放在操作对象之前,也可以放在操作对象之后。它们的功能是自动将运算对象增 1 或减 1,然后把运算结果回存到运算对象中。例如:

(1)++k、--k。

(2)k++、k--。

上述两种形式在独立使用时效果一样,都等价于 k=k+1 或 k=k-1。否则,第(1)种形式是先使 k 增 1 或减 1,再使用 k;第(2)种形式是先使用 k 的值,再将 k 增 1 或减 1。例如:

```
i=3;printf("%d",++i);
/*输出 4*/
i=3;printf("%d", i++);
/*输出 3*/
```

需要注意以下几点:

(1)++i、--i(前置运算),先自增、自减,再参与运算;i++、i--(后置运算),先参与运算,再自增、自减。

(2)自增、自减运算符只用于变量,而不能用于常量或表达式。例如,6++、(a+b)++、(-i)++都不合法。

(3)++、--的结合方向是自右向左(与一般算术运算符不同)。例如,有表达式-i++,其中 i 的值为 3。由于负号运算符与自加运算符的优先级相同,但结合方向是自右向左,即相当于-(i++)。此时++属于后缀运算符,表达式的值为-3,i 的值为 4。

(4)自增、自减运算符常用于循环语句中,使循环变量自动加 1,也用于指针变量,使指针指向下一个地址。

(5)C 语言中函数调用时,如果自增或自减出现在实参中,实参的求值顺序,C 标准也无统一规定。例如:

```
i=3;printf("%d,%d",i,i++);
```

在 Turbo C 环境下，执行结果为 4,3；在 VC 环境下，执行结果为 3,3。

当出现 m+++n 这样的表达式时，C 编译系统在处理时尽可能多地（自左向右）将若干字符组成一个运算符，上式被理解成（m++）+n，而不是 m+（++n）。

当出现（k++）+（k++）+（k++）这样的表达式时（设 k 初值为 3），Turbo C 先用 k 的原值三连加，算出表达式的值为 9 后，再让 k 连续自增 3 次，最终 k 的值为 6；当出现（++k）+（++k）+（++k）这样的表达式时（设 k 原值仍为 3），Turbo C 先让 k 连续自增 3 次，最终 k 的值为 6，再用 6 值连加，算出表达式的值为 18；当出现（++k）+（k ++）+（++k）这样的表达式时（设 k 原值仍为 3），Turbo C 先按第一、第三个括号内的式子让 k 自增 2 次，值为 5，用 5 连加，算出表达式的值为 15，再依据第二个括号内的式子让 k 自增 1 次，最终 k 的值为 6。

总之，尽量不要写别人看不懂（难看懂）也不知道系统会怎样执行的程序。

2.4.4　逗号运算符和逗号表达式

逗号运算符是 C 语言提供的又一种特殊的运算符。用逗号运算符将两个或多个表达式连接起来，构成一个完整的表达式，称为逗号表达式。

逗号表达式的一般形式如下：

表达式 1,表达式 2,……,表达式 *n*

逗号表达式的求解过程是：自左向右，求解表达式 1，求解表达式 2，……，求解表达式 *n*。整个逗号表达式的值是表达式 *n* 的值。

注意：逗号表达式的优先级最低，结合性为自左向右。

例如，表达式“a=3 * 5,a++,a * 4”的值为 64，变量 a 的值为 16。

逗号表达式主要用于将若干表达式“串联”起来，表示一个顺序的操作（计算），在许多情况下，使用逗号表达式的目的只是想分别得到各个表达式的值，而并非一定需要得到和使用整个逗号表达式的值。

【例 2-6】 逗号表达式示例。

程序代码如下：

```
#include <stdio.h>
main()
{
    int x,a;
    x=(a=3,6*3);      /* a=3,x=18 */
    printf("%d,%d\n",a,x);
    x=a=3,6*a;        /* a=3,x=3 */
    printf("%d,%d\n",a,x);
}
```

程序的运行结果如下：

```
3,18
3,3
```

2.4.5 不同类型数据间的转换

1. 表达式计算中的数据类型转换

C 语言允许不同类型的数据混合运算，当不同类型的数据在运算符的作用下构成表达式时要进行类型转换，即把不同的类型先转换成统一的类型，再进行计算。

通常数据之间的转换遵循的原则是“类型提升”，即如果一个运算符有两个不同类型的数据操作，那么在运算之前，先将较低类型的数据提升为较高类型，再进行运算，其结果数据是较高类型。一般来说，运算过程中系统自动转换的称为隐式转换。

说明：

(1)转换按数据长度长的类型转换。例如，int 型数据和 long 型数据运算时，先把 int 型数据转换为 long 型后再进行计算。

(2)所有浮点数运算都是以 double 型数据进行的。若一个表达式仅含 float 型数据，也要先把 float 型数据转换为 double 型数据，再进行运算，结果自然是 double 型数据。

(3)char 型数据和 short 型数据参与运算时，必须先将它们转换为 int 型，结果为 int 型。

2. 强制类型转换表达式

强制转换是通过类型转换运算来实现的，用于把表达式的结果强制转换为类型说明符所表示的类型。其一般形式如下：

(类型说明符)表达式

例如：

```
(int)a          /* 将 a 的结果强制转换为整型 */
(int)(x+y)      /* 将 x+y 的结果强制转换为整型 */
(float)a+b      /* 将 a 的内容强制转换为浮点型，再与 b 相加 */
```

说明：

(1)类型说明和表达式都需要加括号(单个变量可以不加括号)。

(2)无论是隐式转换还是强制转换都是临时转换，不改变数据本身的类型和值。

资料
等级考试重难点

【例 2-7】 强制类型转换示例。

程序代码如下：

```
#include <stdio.h>
main()
{
    float f=5.75;
    printf("(int)f=%d\n",(int)f);  /* 将 f 的结果强制转换为整型，输出 */
    printf("f=%f\n",f);            /* 输出 f 的值 */
}
```

程序的运行结果如下：

```
(int)f=5
f=5.750000
```

综 合 实 例

实例 1　编写一个 C 程序，判断并输出用户输入的整数是奇数还是偶数。

参考程序如下：

```
#include <stdio.h>
int main()
{
    int n;
    printf("输入一个整数 n:");
    scanf("%d",&n);
    printf("%d 是一个%s\n",n,(n%2==0?"偶数":"奇数"));
    return(0);
}
```

实例 2　编写一个程序，获得当前使用的编译系统所分配的数据类型、常量、变量和表达式所占内存字节数。

参考程序如下：

```
#include <stdio.h>
int main()
{
    int a=2;
    printf("%d,",sizeof(int));
    printf("%d,",sizeof(long));
    printf("")
}
```

思考与练习

1. 选择题

(1)下列可以作为 C 语言用户标识符的是(　　)。

A. void，define，WORD　　B. A3_B3，_123，abc

C. FOR，－abc，Case　　D. 2a，Do，Sizeof

(2)下列运算符优先级最高的是(　　)。

A. 关系运算符　　B. 赋值运算符

C. 算术运算符　　D. 逻辑运算符

(3)sizeof(float)是(　　)。

A. 一种函数调用　　B. 一个不合法的表示形式

C. 一个整型表达式　　D. 一个浮点表达式

(4)下列字符串中,常量不正确的是(　　)。

A. 'abc'　　B. "12'12"　　C. "0"　　D. " "

(5)下列 4 个选项均是合法整型常量的是(　　)。

A. 160,－0xffff,011　　B. －0xcdf,01a,0xe

C. －01,986,012,0668　　D. －0x48a,2e5,0x

(6)以下选项中,不属于 C 语言类型的是(　　)。

A. signed short int　　B. unsigned long int

C. unsigned int　　D. long short

(7)数值 029 是一个(　　)。

A. 八进制数　　B. 十六进制数

C. 十进制数　　D. 非法数

(8)在 C 语言中,要求运算数必须是整型的运算符是(　　)。

A. /　　B. ＋＋　　C. !＝　　D. %

(9)当 c 的值不为 0 时,以下能将 c 的值赋给变量 a、b 的是(　　)。

A. c＝b＝a　　B. (a＝c)||(b＝c)

C. (a＝c)&&(b＝c)　　D. a＝c＝b

(10)若有说明语句“char c＝'\72';”,则变量 c(　　)。

A. 包含 1 个字符　　B. 包含 2 个字符

C. 包含 3 个字符　　D. 说明不合法,c 的值不确定

(11)设有说明语句“char w;int x;float y;double z;”,则表达式“w＊x＋z－y”值的数据类型为(　　)。

A. float　　B. char　　C. int　　D. double

(12)长整型数据在内存中的存储形式是(　　)。

A. ASCII 码　　B. 原码　　C. 反码　　D. 补码

(13)字符型常量在内存中的存储形式是(　　)。

A. ASCII 码　　B. BCD 码　　C. 内部码　　D. 十进制码

(14)若 x、i、j 和 k 都是 int 型变量,则计算表达式“x＝(i＝4,j＝16,k＝32)”后 x 的值为(　　)。

A. 4　　B. 16　　C. 32　　D. 52

(15)若有代数式$\frac{3ae}{bc}$,则下列 C 语言表达式中,不正确的是(　　)。

A. a/b/c＊e＊3　　B. 3＊a＊e/b/c

C. 3＊a＊e/b＊c　　D. a＊e/c/b＊3

(16)表达式“109!＝99”的值是(　　)。

A. 1　　B. 非空值　　C. 0　　D. true

(17)若变量 t 为 double 类型,表达式“t＝1,t＊5”,则 t 的值为(　　)。

A. 1　　B. 6.0　　C. 2.0　　D. 1.0

(18)表示关系“x＜＝y＜＝z”的 C 语言表达式为(　　)。

A. (x<=y)&&(y<=z)　　B. (x<=y) AND (y<+z)

C. (x<=y<=y)　　D. (x<=y)&(y<=z)

(19)在 C 语言中,int、char 和 short 三种类型数据所占用的内存(　　)。

A. 均为 2 字节　　B. 由用户自己定义

C. 由所用机器的字长决定　　D. 是任意的

(20)下列程序的执行结果是(　　)。

```
#include <stdio.h>
void main()
{
    int i=2;
    printf("%d,%d\n", ++i, --i);
}
```

A. 1, 1　　B. 2, 1　　C. 1, 2　　D. 2, 2

(21)字符串常量"\\\22a,0\n"的长度是(　　)。

A. 8　　B. 7　　C. 6　　D. 5

(22)下列叙述中,错误的是(　　)。

A. 在 C 语言程序中,逗号运算符的优先级最低

B. 在 C 语言程序中,APH 和 aph 是两个不同的变量

C. 若变量 a 和 b 的类型相同,在计算了赋值表达式"a=b"后,b 中的值将放入 a 中,而 b 中的值不变

D. 当从键盘输入数据时,对于整型变量只能输入整型数值,对于实型变量只能输入实型数值

(23)下列叙述中,正确的是(　　)。

A. C 语言中既有逻辑类型也有集合类型

B. C 语言中没有逻辑类型但有集合类型

C. C 语言中有逻辑类型但没有集合类型

D. C 语言中既没有逻辑类型也没有集合类型

2. 填空题

(1)已知"int x=6;",则执行语句"x+=x--x*x;"后 x 的值是_____。

(2)若 w=1, x=2, y=3, z=4, 则条件表达式"w>x?w:y<z? y:z"的结果是_____。

(3)若"int m=5,y=2;",则计算表达式"y+=y--m*=y"后 y 的值是_____。

(4)在 C 语言中,一个 int 型数据在内存中如果占 2 字节,则 int 型数据的取值范围为_____。

(5)已知字母 a 的 ASCII 码为十进制数 97,且设 ch 为字符型变量,则表达式"ch='a'+'8'-'3'"的值为_____。

(6)若 x 和 n 均为 int 型变量,且 x 和 n 的初值均为 5,则计算表达式"x+=n++"后 x 的值为_____,n 的值为_____。

(7)若有定义"int a=2, b=3;float x=3.5,y=2.5;",则表达式"(float)(a+b)/2+

(int)x%(int)y"的值为_____。

(8)以下程序的输出结果是_____。

```
#include <stdio.h>
int main()
{
    int k=2,i=2,m;
    m=(k+=i*=k);
    printf("%d,%d\n",m,i);
    return 0;
}
```

(9)以下程序的执行结果是_____。

```
#include <stdio.h>
int main()
{
    int a,b,c;
    a=b=1;
    c=a++-1;
    printf("%d,%d,", a, c);
    c+=-a+++(++b||++c);
    printf("%d,%d\n",a,c);
    return 0;
}
```

(10)以下程序的执行结果是_____。

```
#include <stdio.h>
int main()
{
    int n=023;
    printf("%d\n",--n);
    return 0;
}
```

(11)以下程序的执行结果是_____。

```
#include <stdio.h>
void main()
{
    short int i=10;
    printf("%d,%d,%d",--i,--i,i--);
}
```

(12)以下程序的执行结果是_____。

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

```
void main()
{
    short int a=-32768, b;
    b=a-1;
    printf("a= %d,b= %d", a, b);
}
```

(13)以下程序的执行结果是_____。

```
#include <stdio.h>
void main()
{
    int x=042, y=067, z;
    z=(x>>2) & (y<<3);
    printf(" %d\n",z);
}
```

(14)以下程序的执行结果是_____。

```
#include <stdio.h>
void main()
{
    int x=10,y=9;
    int a,b,c;
    a=(--x==y++)?--x:++y;
    b=x++;
    c=y;
    printf(" %d, %d, %d\n",a,b,c);
}
```

(15)以下程序输入 123456789,其输出结果是_____。

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