第3章

基本数据类型、运算符和表达式

CHAPTER 3

◇ **学习导读**

通过前面内容的学习，已经了解了C语言源程序的基本结构以及源程序包含的基本语句和合法的标识符，那么如何编写一个程序解决实际问题呢？首先，需要确定处理的对象，然后，实现功能。在计算机中，确定处理对象就是描述或定义问题所涉及对象的数据类型，实现功能就是在处理对象之间进行运算，进而通过语法规则编写语句来解决实际问题。

本章详细介绍C语言中常用的基本数据类型以及在数据处理过程中使用的基本运算符和表达式。通过本章的学习，读者将能够用基本数据类型定义问题中使用到的数据，以及用合适的运算符和表达式描述数据的简单处理过程。本章是学习C语言的重要基础。

◇ **内容导学**

(1) 变量和常量的概念。

(2) 各种类型的数据在内存中的存放形式。

(3) 各种类型常量的使用方法。

(4) 各种整型、字符型、浮点型变量的定义和引用方法。

(5) 自动数据类型转换的规则以及强制数据类型转换的方法。

(6) 算术运算符、赋值运算符、逗号运算符以及sizeof运算符的使用方法。

(7) 运算符的优先级和结合性。

◇ **育人目标**

《论语》的“学而不思则罔，思而不学则殆”意为：学习而不思考就会迷惘无所得；思考而不学习就不切于事而疑惑不解——常学习，常思考，定会有收获。

本章的基本数据类型、基本运算符及其构成的表达式都是今后学习的重要基础语法知识，也是所有高级语言学习的基础。作为初学者从基础知识开始，认真学习并了解不同数据类型的区别和用法，基本运算符、表达式的运算规则，养成良好的编程习惯和语法规则意识，可以帮助编程者避免很多基础性的语法错误。


视频讲解

3.1　C 语言的数据类型

本节主要讨论以下问题。

(1) C 语言支持的数据类型有哪些?

(2) C 语言的基本数据类型有哪些?

(3) C 语言的构造数据类型有哪些?

学习 C 语言的最终目的就是编写解决实际问题的程序,而程序的操作对象是数据。计算机在处理数据时,要求数据必须具有确定的数据类型。数据类型决定了数据在内存中占有存储空间的大小、取值范围以及能够参与的运算。C 语言提供了丰富的数据类型,分为基本数据类型、构造数据类型、指针类型和空类型四大类,如图 3-1 所示。

图 3-1　C 语言的数据类型分类

1. 基本数据类型

基本数据类型的主要特点是其值不可以再分解为其他类型。C 语言中,基本数据类型有 3 种,它们分别是整型、实型和字符型。本章将详细介绍这 3 种基本数据类型。

2. 构造数据类型

为了更好地处理并表达更复杂的数据,C 语言提供了构造数据类型。构造数据类型根据已定义的一个或多个数据类型用构造的方法来定义。它的最主要的特点是其值可以分解成一个或更多个成员或元素,而每个成员又是一个独立的基本数据类型或构造数据类型。C 语言中,构造数据类型有 4 种,它们分别是数组类型、结构体类型、共用体类型和枚举类型。

3. 指针类型

指针类型是一种特殊的数据类型,同时又是具有重要作用的数据类型。其值用来表示数据在内存中的地址。指针是 C 语言的精髓,没有学好指针就等于没有学好 C 语言。

4. 空类型

函数在定义时需要给出函数的返回值的类型。若一个函数没有返回值,则其返回值的类型为空,用 void 表示。

3.2　数据的表现形式

本节主要讨论以下问题。

(1) 数据在程序中如何表现? 表现的类型有哪几种?

(2) 常量和变量有什么区别?

(3) 常量有哪些类型? 不同数据类型常量的区别是什么?

(4) 变量有哪些类型? 不同数据类型变量的区别是什么?

程序中处理的数据,除了要求明确规定是哪种数据类型之外,还要求数据以一定的形式

出现。在C语言中,数据的表现形式有常量和变量两种。

3.2.1 常量

程序运行时其值不能改变的数据称为常量,即数据的表现形式为常量。

常量直接在程序中使用,根据其使用的形式分为两种,即直接常量和符号常量。直接常量又称值常量,是指直接将值写出来的常量。值常量有整型常量、实型常量、字符型常量和字符串常量4种。而符号常量是指用标识符表示常量,在使用符号常量前,必须用宏定义对符号常量进行定义。

【例3-1】 输入圆的半径,输出其周长和面积。

算法分析:

已知圆的半径r,求其周长c和面积s的公式分别为$c=2\pi r$,$s=\pi r^2$。对于计算机而言,r是变化的,不确定的值称为变量,同理,依赖于r的周长c和s也是变量。

源程序:

```
1  /*3-1.c*/
2  #include"stdio.h"
3  #define PI 3.1415926                    /*用宏定义说明符号常量PI*/
4  int main(void)
5  {
6        double r,s,c;                     /*定义3个双精度变量r、s、c*/
7        scanf("%lf",&r) ;                 /*输入变量r的值*/
8        c=2*PI*r;                         /*计算c的值*/
9        s=PI*r*r;                         /*计算s的值*/
10    printf("c=%lf,s=%lf\n",c,s);         /*输出c、s的值*/
11    return 0;
12 }
```

程序解析:

该程序中使用的数据有圆的半径r、圆的周长c、圆的面积s、圆周率π以及圆的周长公式中的数2。在这些数据中,前3个数据是会变化的,后两个数据是不变的,因此把前面3个数据定义成变量,分别用标识符r、c和s表示,而后面两个数据则是常量。其中,圆的周长公式中的数2直接写出,称为直接常量或值常量,圆周率π即3.14,该实型常量可以直接写出,也可以在程序中使用PI(π的谐音),这种形式出现的常量称为符号常量。使用符号常量之前,必须在程序开头写一条“#define PI 3.1415926”宏定义预处理命令。这样,符号常量PI在程序编译时会进行宏替换,值为3.1415926。

程序运行结果如下。

```
3↙
c=18.849556,s=28.274333
```

3.2.2 变量

程序运行时其值可以被改变的数据称为变量,即数据的表现形式为变量。

通常用变量来保存程序执行过程中的输入数据、中间结果和最终结果。

在源程序中,为了区别不同的变量,就需要为变量取一个名字,名字也就是一个标识符,

这个标识符就是变量名。在程序运行过程中，通过变量名来访问变量的值。C 语言规定，变量在使用之前，必须先定义，定义的位置一般放在函数体的开头。

1. 变量定义的格式

一般来说，变量定义的基本格式为：

[存储类型]　数据类型 变量名 1[,变量名 2,…,变量名 n];

其中，数据类型决定了系统为变量分配的字节数和数的取值范围，变量 1、变量 2、…、变量 n 是合法的标识符，即变量名。例如以下定义的变量：

```
int    x, y, z;
float  radius, length, area;
char   ch;
```

其中，变量 x、y、z 定义为基本整型，radius、length、area 定义为单精度实型，ch 定义为字符型。定义时，可以定义一个变量，也可以同时定义多个变量。

2. 变量的初始化

在定义时，还可以为变量赋初值，称为变量的初始化。例如以下定义的变量：

```
int r = 3;
```

其中，定义了变量 r 为基本整型，同时 r 的初值为 3。

3. 变量的存储地址

在程序运行期间，编译系统会为程序中定义的每个变量分配一定大小的存储单元，存储单元的初始地址称为该变量的存储地址。程序员在编写程序过程中要引用变量的存储地址时，可以通过取地址运算符(&)来获得，引用格式为：

& 变量名

例如以下语句：

```
scanf("%lf",&r);
```

其中，scanf 表示从标准输入设备(键盘)输入一个双精度类型的数据存入到变量 r 所对应的存储单元中，因此，就需要知道 r 所对应的存储单元地址 &r。

3.3　基本数据类型

本节主要讨论以下问题。

(1) 基本数据类型有哪些？

(2) 不同基本数据类型的常量如何表示？

(3) 不同基本数据类型的变量如何定义？定义之后，其占用的存储空间、表示的范围是什么？

(4) 不同基本数据类型的数据在内存中如何存储？

(5) 字符串常量怎么表示？字符串常量与字符常量有什么区别？

使用一个数据时，不但要知道它的表现形式还要知道它的类型。前面介绍到，数据在程序执行过程中，根据其值是否可以改变分为常量和变量，而基本数据类型分为整型、实型、字符型，因此，基本数据又可以分为整型常量、实型常量、字符型常量以及整型变量、实型变量、

字符型变量。下面详细介绍基本数据类型的常量和变量的表示及表示范围。


视频讲解

3.3.1 整型数据

1. 整型常量的表示形式

整型常量即整型常数。在C语言中,整型常量的表示形式有以下3种。

(1) 十进制整型常量。由数字0～9和正负号组成,如10、－200、897等是合法的十进制整型常量,而056、23F是不合法的十进制整型常量,其中056含有前导0,23F中含有不合法的字母F。

(2) 八进制整型常量。由数字0～7组成,且以数字0开头,如056、071等是合法的八进制整型常量,而086、54是不合法的八进制整型常量,其中086含有不合法数字8,54不是以0开头。

(3) 十六进制整型常量。由数字0～9、字母A～F或a～f组成,且以0X或0x开头,如0x12、0x1ab等是合法的十六进制整型常量,而7B、0X34H是不合法的十六进制整型常量,其中7B不是以0X或0x开头,0X34H中含有不合法的字母H。

2. 整数在内存中的表示

通过第1章的学习,已经了解到数据以二进制形式存储在计算机中,其机内表示形式通常有3种,即原码、反码和补码。其中补码可以连同符号位一起参与运算。因此,计算机系统规定:整数的数值在内存中用补码的形式存放。在TC 2.0或BC 3.1下,一个整数默认情况下需要2字节(16位)的内存单元存放;而在Dev-C++ 6.3、VC++ 6.0下,则需要4字节(32位)。下面通过具体的例子来观察不同进制整数在内存中的表示形式。

1) 十进制整数

(1) 设有十进制整数17,根据求补码的规则得出17在内存中的表示形式。

① 若用16位的内存单元存放该整数,结果为$(17)_{补}$=0000 0000 0001 0001。左边第1个数位为符号位。根据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的原则,可以得出十进制整数17存放在16位的存储单元中的存放形式,如图3-2所示。

② 若用32位的内存单元存放该整数,结果为$(17)_{补}$=0000 0000 0000 0000 0000 0000 0001 0001。左边第1个数位为符号位。其在内存中的存放形式如图3-3所示。

低字节	0001 0001	低地址
高字节	0000 0000	高地址

图3-2 十进制整数17存放在16位的存储单元中的存放形式

低字节	0001 0001	低地址
	0000 0000	
	0000 0000	
高字节	0000 0000	高地址

图3-3 十进制整数17存放在32位的存储单元中的存放形式

(2) 设有十进制整数－17,根据求补码的规则得出在内存中的表示形式。

① 若用16位的内存单元存放该整数,结果为$(-17)_{补}$=1111 1111 1110 1111。左边第

1 个数位为符号位。其在内存中的存放形式如图 3-4 所示。

② 若用 32 位的内存单元存放该整数，结果为$(-17)_{补}$＝1111 1111 1111 1111 1111 1111 1110 1111。左边第 1 个数位为符号位。其在内存中的存放形式如图 3-5 所示。

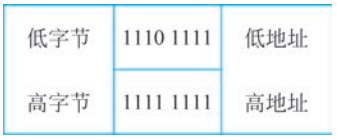

图 3-4　十进制整数－17 存放在 16 位的存储单元中的存放形式

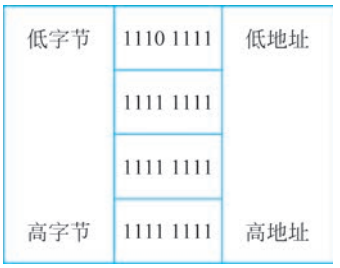

图 3-5　十进制整数－17 存放在 32 位的存储单元中的存放形式

2）八进制整数

（1）设有八进制整数 052，根据不同进制数之间的转换方法及求补码的规则得出下面的结果。

$(052)_8=(101010)_2$，则若用 16 位的内存单元存放该数，结果为$(052)_{补}$＝0000 0000 0010 1010。若用 32 位的内存单元存放该数，结果为$(052)_{补}$＝0000 0000 0000 0000 0000 0000 0010 1010。

（2）设有八进制整数－052，根据不同进制数之间的转换方法及求补码的规则得出下面的结果。

若用 16 位的内存单元存放该数，结果为$(-052)_{补}$＝1111 1111 1101 0110。若用 32 位的内存单元存放该数，结果为$(-052)_{补}$＝1111 1111 1111 1111 1111 1111 1101 0110。

3）十六进制整数

设有十六进制整数 0X85AB，$(0X85AB)_{16}=(1000\ 0101\ 1010\ 1011)_2$，根据不同进制数之间的转换方法及求补码的规则得出下面的结果。

（1）若用 16 位的内存单元存放该数，结果为$(0X85AB)_{补}$＝1000 0101 1010 1011。因此，数据 0X85AB 在内存中的存放的形式为 1000 0101 1010 1011。从该形式中可以看出，最高位为 1，说明该数为负数且为其真值的补码在内存中的存放形式，其对应的真值二进制数为－111 1010 0101 0101，即十进制数－31317。

（2）若为 32 位的内存单元存放该数，结果为$(0X85AB)_{补}$＝0000 0000 0000 0000 1000 0101 1010 1011。因此，数据 0X85AB 在内存中的存放的形式为 0000 0000 0000 0000 1000 0101 1010 1011。从该形式中可以看出，最高位为 0，说明该数为正数且为其真值的补码在内存中的存放形式，其对应的真值二进制数为 1000 0101 1010 1011，即十进制数 34219。

3. 整型变量

C 语言中，当处理的数据以变量的形式出现时，必须先定义后使用。

（1）整型变量的定义。定义整型变量的基本格式为：

```
整型类型名　变量名 1[,变量名 2,…,变量 n];
```

在使用此格式定义整型变量时，必须遵守以下规则。

① 整型类型名属于关键字，必须小写。

② 整型类型名与变量名之间要有空格分开。

③ 可以同时定义多个变量,但变量名间要用“,”分开。

④ 在定义变量时也可以对变量赋初值,具体格式为:

```
整型类型名  变量名1=值1[,变量名2=值2,…,变量n=值n];
```

⑤ 最后以“;”结尾。

例如:

```
int  a;              /*定义整型变量a*/
int  a,b,c;          /*定义整型变量a、b、c*/
int  a=3,b;          /*定义整型变量a、b,同时给变量a赋初值3*/
int  a=3,b=4;        /*定义整型变量a、b,同时给变量a、b分别赋初值3、4*/
```

(2) 整型变量的类型。整型变量的基本类型标识符是int。C语言还允许编程者在定义整型变量时,在int前面加上修饰符,这些修饰符可以是unsigned(无符号)、signed(有符号,可以省略)以及short(短)和long(长)。这样,整型变量的类型就有6种。不同的类型决定了该数的范围以及在内存中占有的空间大小。

例如:

```
long int a;
```

该定义表示定义了一个长整型变量a,在程序执行时为变量a分配4字节。

不同整型变量的类型所表示的数的范围及分配的字节数如表3-1所示([]表示此部分在定义时可省略,不同类型所占的字节数是指在Dev-C++ 6.3环境下的取值)。

表3-1 整型变量的类型

类型	种 类	保 留 字	字节数	取 值 范 围
整型	有符号基本整型	[signed] int	4	−2147483648～+2147483647
	有符号短整型	[signed] short [int]	2	−32768～+32767
	有符号长整型	[signed] long [int]	4	−2147483648～+2147483647
	无符号基本整型	unsigned [int]	4	0～4294967295
	无符号短整型	unsigned short [int]	2	0～65535
	无符号长整型	unsigned long [int]	4	0～4294967295

例如:

```
int a;                    /*定义一个有符号基本整型变量a*/
unsigned int a =8;        /*定义一个无符号基本整型变量a,并赋初值8*/
unsigned a = -20;         /*定义一个无符号基本整型变量a,并赋初值-20 */
short int a =10;          /*定义一个有符号短整型变量a,并赋初值10 */
unsigned long a = 2;      /*定义一个无符号长整型变量a,并赋初值2 */
long a = 245968;          /*定义一个有符号长整型变量a,并赋初值245968*/
```

实际上,不管是哪种类型的数据,它们在计算机内存中都以其补码形式表示,当把数的最高位当成符号位时,则为有符号数;当把最高位当成数据位看待时,则变为无符号数。

例如:

```
unsigned int a = -2;
printf("%d",a);           /*有符号输出,则为-2 */
printf("%u",a);           /*无符号输出,则为4294967294*/
```

4. 整型常量的类型

整型变量的类型有 6 种，那么整型常量的类型是不是也有 6 种呢？C 语言根据整型常量的值来决定整型常量的类型，具体判定规则如下。

(1) 根据常量值所在范围确定其数据类型。在 TC 2.0 或 BC 3.1 下，若整型常量的值为－32768～32767，C 语言则认为它是 int 型常量；若整型常量的值为－2147483648～2147483647，C 语言则认为它是 long 型常量。

(2) 长整型。整型常量后加字母 l 或 L，C 语言认为它是 long int 型常量。如 483L、571。

(3) 无符号整型。整型常量后加字母 u 或 U，C 语言认为它是 unsigned int 型常量；加字母 ul 或 UL，则表示是无符号长整型(unsigned long int)常量，如 5846UL、5874789012ul。

3.3.2　实型数据


视频讲解

1. 实型常量的表现形式

实型也称为浮点型。实型常量也称为实数或者浮点数。在 C 语言中，实数只采用十进制表示。它用十进制小数和十进制指数两种形式表示。

(1) 十进制小数形式。十进制小数形式表示的实型数据由数字 0～9 和小数点组成，如 0.83、36.12、85.0 等。当小数点前面只有 0 或者后面只有 0 时，0 则可以省略，但小数点不可以省略，如.83、85.等。

(2) 指数形式。十进制指数形式表示的实型数据是由十进制数、阶码标志(e 或 E)以及阶码组成。其一般形式为 aE(e)±n，其中 a 为十进制数，又称为尾数部分，n 为十进制整数，又称为指数部分，两者必不可少；±表示指数的正负，当指数为正时，＋号可以省略。数学中表示为 $a\times10^{\pm n}$。如：3.2×10^{5} 可以写成 3.2E＋5、0.32E＋6 等合法的指数形式。

可见，实型数据 3.2×10^{5} 可用多种不同的指数形式表示。为了统一起见，C 语言中定义了一种标准化的指数形式。标准化的指数形式是 E 或 e 之前的尾数部分的小数点左边为零且右边第 1 位为非零的数字，如 30.75 的标准化指数形式为 0.3075e2。这种标准化的指数形式主要用于存储，提高数据的精确度。但在以指数形式输出时，一般使用规范化的指数形式。规范化的指数形式是指 E 或 e 之前的尾数部分的小数点左边第 1 位为非零的数字，如 30.75 的规范化指数形式为 3.075e＋1。

2. 实数在内存中的表示

计算机中的实数按指数形式存放。通常一个实数需要 4 字节的内存，计算机将这 32 位分成 S、E 和 M 三部分：最高位 S 是尾数的符号位，其余的 31 位分成 E 和 M 两部分，E 部分用来存放实数的阶码部分，M 部分用于存放实数的尾数部分，如图 3-6 所示。

图 3-6　实数的存储格式

在 C 语言中，具体如何分配由 C 语言的编译器来决定。按照 IEEE 754 标准，常用浮点数的存储分配格式如表 3-2 所示。

表 3-2 常用浮点数的存储分配格式

	符号位 S	阶码 E	尾数 M	总位数
单精度浮点数	1	8	23	32
双精度浮点数	1	11	52	64
长双精度浮点数	1	15	64	80

对于浮点数来说,尾数部分占的位数越多,它所能表示的精度越高;阶码部分占的位数越多,它所能表示的值越大。

【例 3-2】 将十进制数 78.125 表示成机器内的 32 字节的二进制数形式。

计算方法如下:

(1) 将 78.125 表示成二进制数。

$$(78.125)_{(10)}=(0.781\ 25\times10^{2})_{(10)}$$

则

$$\text{尾数}(0.781\ 25)_{(10)}=(0.110\ 010\ 000\ 000\ 000\ 000\ 000\ 00)_{(2)}$$

$$\text{阶码}(2)_{(10)}=(00000010)_{(2)}$$

(2) 根据此过程可以得出符号位、尾数和阶码。

① 符号位:该数为正数,故第 31 位为 0,占一个二进制位。

② 阶码:从第 30 位到第 23 位,共占 8 个二进制位。

③ 尾数为小数点后的部分 0.78125,即 11001。因为尾数共 23 个二进制位,在后面补 18 个 0,即 11001000000000000000000。

所以,78.125 以 32 字节存储在内存中的存放方式如表 3-3 所示。

表 3-3 78.125 以 32 字节存储在内存中的存放方式

符号位(共 1 位,第 31 位)	阶码(共 8 位,第 30 位至第 23 位)	尾数(共 23 位,第 22 位至第 0 位)
0	0000 0010	1100 1000 0000 0000 0000 000

3. 实型变量

实型变量有单精度类型、双精度类型和长双精度类型 3 种。这 3 种类型对应的标识符分别是 float、double、long double。实型变量的定义格式与整型变量的定义格式一样,只是数据类型标识符不同。

例如:

```
float r;            /* 定义 1 个单精度类型的实型变量 r */
double r,c,s;       /* 定义 3 个双精度类型的实型变量 r、c、s */
float r = 3,c,s;    /* 定义 3 个单精度类型的实型变量 r、c、s,同时变量 r 赋初值 3 */
```

不同实型变量的类型所表示的数的精度及分配的字节数,如表 3-4 所示。

表 3-4 实型变量的类型

类型	种 类	保 留 字	字 节 数	精确表示的数字个数
实型	单精度类型	float	4	6～7
	双精度类型	double	8	15～16
	长双精度类型	long double	16	18～19

4. 实型常量的类型

实型常量的类型有单精度类型和双精度类型两种。所有的实型常量都按照 double 类型处理，若要表示单精度类型，则在数值后面加个 F 或 f，如 0.12f。

3.3.3　字符型数据

有了整型数据和实型数据，就足够处理数学计算了。实际上，计算机除了处理一般的数值信息外，还要处理其他大量的非数值信息，如文本信息。那么，计算机是怎么处理文本信息的呢？字符型数据就是用来表示那些非数值信息的文本数据，如英文字母、符号、汉字等。

字符型数据本质上是整型数据，在内存中存储该字符数据的 ASCII 且分配 1 字节的存储单元。下面介绍字符型常量和变量。

1. 字符型常量

字符常量有下面两种表示方法。

(1) 用一对单引号将一个直接输入的字符引起来。直接输入的字符是指通过键盘直接输入的字符。如'a'、'b'和'?'等都是合法的字符常量。

(2) 使用转义字符。对于那些无法直接输入的字符以及某些特殊的字符，需要用由一对单引号括起来的转义字符来表示。转义字符是一种特殊的字符常量，具有特定的含义，不同于原有字符的意义，故称为转义字符。所有的转义字符都以“\”开头，后面跟一个字符或多个数字。C 语言中常用的转义字符及其含义如表 3-5 所示。

表 3-5　常用的转义字符及其含义

形　式	转义字符	含　义
\字母	\a	响铃
	\n	换行，光标移到下一行的开头
	\r	回车，光标移到本行的开头
	\t	水平制表，跳到下一个制表位(Tab)位置
	\f	换页，光标移到下页的开头
	\b	退格，将光标移到前一列
	\0	空字符，作为字符串的结束标记
\符号	\\	代表一个反斜杠字符
	\'	代表一个单撇号字符
	\"	代表一个双撇号字符
\数字	\ddd	代表一个字符，其中 ddd 是这个字符 ASCII 码的八进制形式
\字母数字	\xhh	代表一个字符，其中 hh 是这个字符 ASCII 码的十六进制形式

2. 字符型变量

字符型变量的定义格式与其他类型变量的定义格式一样，只是数据类型标识符不同。字符型变量的类型标识符为 char。

例如：

```
char a;            /* 定义一个字符型类型的变量 a */
char a,b,c;        /* 定义 3 个字符型类型的变量 a、b、c */
char a = 'A',b,c;  /* 定义 3 个字符型变量 a、b、c,同时变量 a 赋初值'A' */
```

3. 字符数据在内存中的表示

在内存中,一个字符型数据占用1字节(8位),以其ASCII码的二进制形式存放。这个ASCII码值就是一个无符号整数,其形式与整数的存储形式一样,所以C语言允许字符型数据与整型数据之间通用。

因此,对一个字符型的变量,可以赋予一个字符常量,也可以赋予一个整数。在输出一个字符型数据时,可以以字符格式输出,也可以整数格式输出。若以字符格式输出时,系统自动将存储单元中的ASCII码转换为字符,然后输出;若以整数格式输出时,系统直接将其ASCII码作为整数输出。

【例3-3】 转义字符的使用。

源程序:

```
1  /*3-3.c*/
2  #include<stdio.h>
3  int main( )
4  {
5     printf ("\101\t\x42\tC\n");
6     printf ("\"Hello    \"\n");
7     printf ("\\C Program\\\n");
8     return 0;
9  }
```

程序解析:

该程序函数体共有4行,主要由函数printf完成信息的输出。

第5行“printf ("\101\t\x42\tC\n");”中,包含转义字符\101、\t、\x42和\n,根据常用转义字符表中描述的意义,分别表示字符'A'、制表符、'B'和换行。

第6行“printf ("\"Hello.\"\n");”中,包含转义字符'\"'、'\n',分别表示字符双引号"和换行。

第7行“printf ("\\C Program\\\n ");”中,包含转义字符\\、\n,分别表示字符'\'和换行。

因此,根据以上分析,程序运行结果如下。

```
A       B       C
"Hello."
\C Program\
```

【例3-4】 字符型数据和整型数据通用。

源程序:

```
1  /*3-4.c*/
2  #include"stdio.h"
3  int main()
4  {
5     int x,y;
6     char c1,c2;
7     x='a';                          /*字符常量赋给整型变量*/
8     y='b';
9     c1=97;                          /*整型常量赋给字符变量*/
10    c2=98;
11    c1=c1-32;                       /*字符变量进行算术运算*/
12    c2=c2-32;
```

```
13    printf("%d %d\n",x,y);          /*整型变量以整型格式输出*/
14    printf("%c %c\n",x,y);          /*整型变量以字符型格式输出*/
15    printf("%c %c\n",c1,c2);        /*字符型变量以字符型格式输出*/
16    printf("%d %d\n",c1,c2);        /*字符型变量以整型格式输出*/
17    return 0;
18  }
```

程序解析见程序后面的注释，以第 13 行为例，表示输出两个整型变量 x 和 y，都以%d 的形式输出，即以十进制整型输出，虽然 x 和 y 赋值为字符型，但 x 和 y 的结果是这两个字符'a'和'b'对应的 ASCII 码。该程序运行结果如下。

```
97 98
a b
A B
65 66
```

3.3.4　字符串常量

1. 字符串常量的定义

C 语言规定，由一对双引号括起来的字符序列称为字符串常量。如"123"、"abc456"、" "(空字符串)等都是合法的字符串常量。

字符串中字符的个数称为字符串长度。长度为 0 的字符串(即一个字符都没有的字符串)称为空串，表示为" "(一对紧连的双引号)。例如，"How do you do."、"Good morning."等，都是字符串常量，其长度分别为 14 和 13。

若字符串常量中出现反斜杠和双引号作为字符串中的有效字符，则必须以转义字符给出。例如：字符串 C:\program\chap3 则应写成"C:\\program\\chap3"的形式。

2. 字符串在内存中的表示

C 语言规定：在存储字符串常量时，由系统在字符串的末尾自动加一个'\0'作为字符串的结束标志。例如字符串"China"在内存中的存储形式如图 3-7 所示。

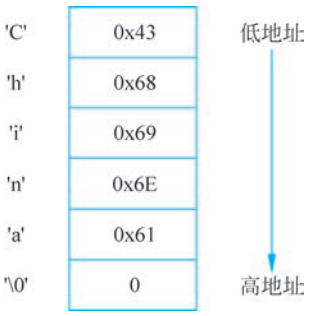

图 3-7　字符串"China"在内存中的存储形式

综上所述，字符常量'A'与字符串常量"A"有三点区别。

(1) 定界符不同。字符常量使用单引号；而字符串常量使用双引号。

(2) 长度不同。字符常量的长度固定为 1；而字符串常量的长度是≥0 的整数。

(3) 存储要求不同。字符常量存储字符的 ASCII 码值；而字符串常量，除了要存储有效的字符外，还要存储一个结束标志'\0'。

3.4　常用运算符与表达式

本节主要讨论以下问题。

(1) C 语言中常用的运算符有哪些？这些运算符按照操作对象的个数不同可以分为哪

些类型？按照功能来分，又分为哪些类型？

(2) 算术运算符的功能是什么？算术运算符有哪些？其优先级和结合性怎样？什么是算术表达式？如何计算算术表达式？

(3) 自增自减运算符的功能是什么？自增自减运算符有哪些？其优先级和结合性怎样？

(4) 赋值运算符的功能是什么？赋值运算符有哪些？其优先级和结合性怎样？什么是赋值表达式？如何计算赋值表达式？

(5) 类型转换的原理是什么？自动类型转换的规则是什么？强制类型转换运算符的功能是什么？如何使用强制类型转换运算符？其优先级怎样？

(6) 逗号运算符的功能是什么？其优先级和结合性怎样？什么是逗号表达式？如何计算逗号表达式？

(7) sizeof 运算符的功能是什么？其优先级和结合性怎样？如何计算值？

(8) 位运算符的功能是什么？位运算符有哪些？其优先级和结合性怎样？什么是位运算表达式？如何计算位运算表达式？

变量用来存放数据，运算符则用来处理数据。用运算符将变量和常量连接起来的符合C语言语法规则的式子称为表达式。这些常量和变量称为运算符的操作数，每个表达式都有一个值。在以后的章节中，将陆续介绍C语言支持的运算符和表达式。

C语言提供了丰富的运算符。根据运算符所需要操作数的个数，可以把运算符分成以下3种类型。

(1) 单目运算符：运算时，需要一个操作数的运算符。

(2) 双目运算符：运算时，需要两个操作数的运算符。

(3) 三目运算符：运算时，需要三个操作数的运算符。

图 3-8 C语言的运算符

在C语言中，运算符和表达式相当丰富，数量繁多。在表达式的计算中，不仅要考虑运算符的优先级，还要考虑同一优先级的运算符具有的结合性。因此对于运算符的学习，可以从运算符的功能、与运算量的关系、运算符的优先级、运算符的结合性以及运算结果的类型5方面进行学习和掌握。

C语言提供了13类运算符，如图3-8所示。

C语言规定了运算符的优先级和结合性。结合性是C语言的独有概念。所谓结合性是指，当一个操作数两侧的运算符具有相同的优先级时，该操作数是应该与左边的运算符结合，还是应该与右边的运算符结合。若是自左至右的结合方向，称为左结合性；反之，称为右结合性。除单目运算符、赋值运算符和条件运算符是右结合性外，其他运算符都是左结合性。

下面介绍最常用的算术运算符、赋值运算符、强制类型转换运算符、求字节长度运算符、逗号运算符、位运算符及其表达式，其他运算符将在以后的章节中陆续介绍。


视频讲解

3.4.1　算术运算符及其表达式

1. 算术运算符

C 语言提供的算术运算符的功能是进行加、减、乘、除四则运算，算术运算符主要包括 5 种：加(＋)、减(－)、乘(＊)、除(/)和取余(％)。

算术运算符都是双目运算符，＊、/、％优先级相同且高于优先级相同的＋、－。此外，＋、－、＊、/可用于任何数据类型间的算术运算，而％只能用于整型数据的算术运算。例如，5％2 的值为 1，5/2 的值为 2，5.0/2 的值为 2.5，5.0％2 则是不合法的表达式。因此，C 语言规定，两个整数相除(/)，其商为整数，否则与数学运算保持一致。

2. 表达式和算术表达式

用运算符将运算对象连接起来、符合 C 语言语法规则的式子，称为表达式。运算对象可以是常量、变量、函数或其他表达式。单个常量、变量或函数，可以看作表达式的特例。将单个常量、变量或函数构成的表达式称为简单表达式，其他表达式称为复杂表达式。

若表达式中的运算符都是算术运算符，称为算术表达式。例如，3＋14、56＊17％10 都是合法的算术表达式。

3. 算术运算符的优先级和结合性

在计算算术表达式值时，必须考虑其运算符的优先级和结合性，先乘、除和取余，再加、减，同级运算符的计算顺序是从左到右，即左结合性。例如，在计算 a－b＋c 时，先执行 a－b；然后执行加 c 的运算。

3.4.2　自增自减运算符、负号运算符

C 语言中，减号(－)是一个算术运算符，其实也是一个负号运算符。负号运算符是一个单目运算符。例如，a＝－5，－a 的值则为 5。

C 语言还提供了两个用于算术运算的运算符：自增(＋＋)和自减(－－)运算符。自增、自减运算符属于单目运算符，它们的结合性为右结合性。这两个运算符的优先级比其他 5 个算术运算符(加、减、乘、除、取余)的优先级高。

1. 作用与用法

自增(＋＋)运算符的作用是使单个变量的值增 1，自减(－－)运算符的作用是使单个变量的值减 1。在使用时，自增(＋＋) 和自减(－－) 运算符只可以放在变量之前或之后，不能放在常量和表达式的前或后。例如，10＋＋、－－(x＋y)等都是非法的。

自增自减运算符常用于循环语句中，使循环变量增 1 或减 1，还用于指针变量中，使指针下移(或上移)一个存储单元。例如，若对变量 a 进行增 1，可以写成＋＋a 或 a＋＋；若对变量 a 进行减 1，可以写成－－a 或 a－－。

2. 运算规则

(1) 前置运算。自增(＋＋)和自减(－－)运算符放在变量之前称为前置运算。例如，＋＋a 和－－a。此时，运算规则为表示变量自身先加或减 1，然后使用变量运算后的值参与其他运算。

例如：

```
int a = 5,b;
```

请分析下面表达式执行完后 a、b 的值。

```
b = ++a
```

此表达式中有两个运算符++和=,++放在 a 之前,因此,a 先加 1 的值为 6,再参与=运算,将 a 的值 6 赋给 b,则 b 的值为 6。此执行过程等价于以下两个表达式:a=a+1,b=a。

(2) 后置运算。自增(++)和自减(--)运算符放在变量之后称为后置运算。例如:a++和 a--。此时,运算规则为先使用变量的原值参与其他运算,结束之后变量自身加 1 或减 1。

【例 3-5】 自增、自减运算符的用法。

源程序:

```
1  /*3-5.c*/
2  #include"stdio.h"
3  int main()
4  {
5     int x = 6, y;
6     printf("x = %d\n",x);
7     y = ++x;
8     printf("x = %d,y = %d\n",x,y);
9     y = x--;
10    printf("x = %d,y = %d\n",x,y);
11    return 0;
12 }
```

程序解析:

(1) 程序中第 7 行表达式语句“y=++x;”中有两个运算符:++和=。++放在 x 之前,因此先执行++运算,即 x 自身加 1 等于 7,然后将 x 的值赋给 y,y 的值为 7。此执行过程等价于以下两个表达式:x=x+1,y=x。

(2) 程序中第 9 行表达式语句“y=x--;”中有两个运算符:--和=。--放在 x 之后,因此先执行=运算,即将 x 的值赋给 y,y 的值为 7,然后执行--运算,即 x 自身减 1 等于 6。此执行过程等价于以下两个表达式:y=x,x=x-1。

根据以上分析过程,该程序的运行结果如下。

```
x = 6
x = 7,y = 7
x = 6,y = 7
```

3.4.3 赋值运算符及其表达式

1. 赋值运算符

赋值符号“=”就是 C 语言中的赋值运算符,它的作用是将一个表达式值赋给一个变量。赋值运算符是双目运算符,它的优先级比算术运算符、自增自减运算符的优先级低,是右结合性。

2. 赋值表达式

由赋值运算符将一个变量和一个表达式连接起来的表达式,称为赋值表达式。它的一

般格式是：

变量 = 表达式

注意，赋值运算符“=”的左侧只能是变量，右侧可以是变量、常量、函数调用或其他复杂表达式。功能是将表达式值赋给变量。

若表达式值的类型与被赋值变量的类型不一致，系统会自动地将表达式值转换成被赋值变量的数据类型，然后赋值给变量。

例如：

```
int a,b,c;
a = 30;
b = a + 10;
c = a + b/10;
```

3. 赋值语句

C 语言规定，赋值表达式末尾加上分号就构成一条赋值表达式语句，简称为赋值语句。例如，“a=30;”“b=a+10;”“c=a+b/10;”就是 3 条赋值语句。

4. 复合的赋值运算符

复合赋值运算符是由一个双目运算符和“=”结合在一起构成，复合赋值运算符包括+=、-=、*=、/=、%=、≪=、≫=、^=、&=、|=。由复合赋值运算符将一个变量和一个表达式连接起来的表达式，称为赋值表达式。它的一般格式是：

变量 op = 表达式

此赋值表达式等价于：

变量 = 变量 op(表达式)

其中，op 表示算术或位运算符，当表达式为简单表达式时，表达式外的一对圆括号才可缺省，否则可能出错。

例如：

```
a += 10          /* 等价于 a = a + 10 */
a * = b + 2      /* 等价于 a = a * (b + 2),而不是 a = a * b + 2 */
```

5. 赋值表达式的应用

1）多个变量连续赋值

例如：

假设有 int a，b=5，c=4，求 a-=a=b+c 表达式的值。

分析：

此表达式为赋值表达式，且含有多个赋值运算符。按照赋值运算符的右结合性，分为以下两步完成。

（1）a=b+c，将 b+c 的值 9 赋给 a，因此，a 的值为 9。

（2）a-=a，等价于 a=a-a，将 a-a 的值为 0 赋给 a，因此，a 的值为 0。

2）赋值表达式的嵌套

例如：

设有表达式 a=(b=2)+(c=3)，则 a=？

分析：

此表达式为赋值表达式,且含有括号运算符、加法运算符和赋值运算符。由于括号的优先级最高,因此,先计算括号中的表达式,将 2 赋给 b,同理赋值表达式 c=3 的值为 3,然后进行加法运算 b+c,值为 5,最后将值 5 赋给 a,即 a=5。


视频讲解

3.4.4 强制类型转换运算符

C 语言提供了丰富的数据类型,当不同类型的数据在一起进行运算时,这些数据的类型可以相互转换。转换的方法有两种:一种是自动类型转换;另一种是强制类型转换。

1. 自动类型转换

若一个运算符两侧的操作数的数据类型不同,则系统按“先转换、后运算”的原则,首先将数据自动转换成同一类型,然后在同一类型数据间进行运算。

自动类型转换是由机器直接完成的,其转换规则如图 3-9 所示。

使用图 3-9 的转换规则时,需遵守以下规则。

(1) 图 3-9 中的横向左箭头表示必定进行的转换:char 和 short 型数据在参与运算时会自动转换为 int 型,float 型数据在参与运算时会自动转换为 double 型。

(2) 图 3-9 中的纵向短箭头表示 int 型、unsigned 型、long 型和 double 型之间的转换方向。若 int 型和 double 型数据进行混合运算,int 型数据会自动转换为 double 型,运算的结果也是 double 型。

(3) 图 3-9 中的纵向长箭头表示转换类型的级别高低,低级别的数据类型会转换成高级别的数据类型,并不表示从下至上依次转换。例如:

```
char ch;
int i;
float f;
double d;
```

那么表达式“result=(ch/i)+(f*d)-(f+i)”的结果类型是哪一种?

分析过程:

根据图 3-9 的转换规则,在表达式 result 的计算过程中,各操作数类型的转换过程如图 3-10 所示。

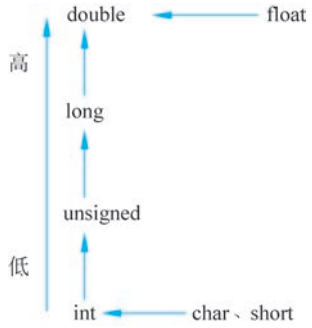

图 3-9 自动类型转换规则

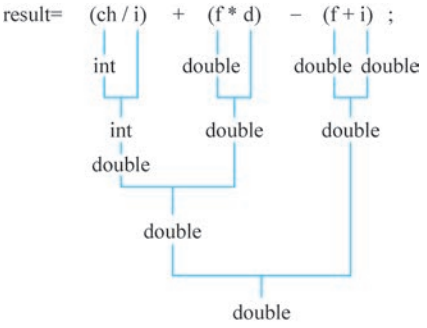

图 3-10 在表达式 result 的计算过程中,各操作数类型的转换过程

2. 强制类型转换

在进行数据处理时，有时需要把数据类型转换成规定的数据类型或者在赋值时保持左右类型一致，这时可以采用强制类型转换符“()”达到要求。强制类型转换又称为显式类型转换。强制类型转换的一般格式为：

(类型名)(表达式)

强制类型转换符“(类型名)”中的类型名是指要转换成的目标类型名，表达式是指需要转换的数据对象，数据对象可以是常量、变量或其他更复杂的表达式。当被转换的表达式是一个简单表达式时，圆括号可以缺省。

强制类型转换得到的是一个所需类型的中间量，原表达式类型并不发生变化。例如，(double)a 是得到一个将变量 a 转换成 double 型的中间值去参与运算，而变量 a 的数据类型并未转换成 double 型。

例如：

```
int a;
float b;
(float)(a)   /* 等价于(float)a,得到变量 a 强制转换成 float 型的一个值 */
(int)(b)     /* 等价于(int)b,得到变量 b 强制转换成 int 型的一个值 */
(int)(b+a)  /* 得到表达式(b+a)强制转换成 int 型的一个值 */
(int)b+a    /* 得到变量 b 强制转换成 int 型的一个值,将这个值与变量 a 相加,运算结果为 int 型 */
```

3.4.5　逗号运算符及其表达式

逗号运算符(,)又称顺序求值运算符，是 C 语言中优先级最低的运算符，具有左结合性。使用该运算符能够将多个表达式连接起来，用逗号连接起来的表达式称为逗号表达式。逗号表达式的一般格式是：

表达式 1,表达式 2,…,表达式 n

例如：8,a+4,b=a 是一个逗号表达式(假设已定义变量 a、b)。

逗号表达式的求值顺序是从左到右依次求取每个用逗号分隔的表达式，最后一个表达式的值就是整个逗号表达式的值。

例如，逗号表达式“(a=3*5,a/5),a+20”的值为 35，求解过程如下。

(1) 求 a=3*5，得 a=15；

(2) 求 a/5=3，但 a 的值未改变；

(3) 求 a+20=35。

因此，逗号表达式的值为 35。

3.4.6　sizeof 运算符

C 语言中提供了一个能获取数据或数据类型所占内存大小的运算符——sizeof，称为求字节长度运算符，是单目运算符。其使用的一般格式是：

sizeof(数据或数据类型名)

例如：

```
double a;
```

```
char b;
```

则表达式 sizeof(a)的值为 8,sizeof(char)的值为 1,sizeof(a+b)的值为 8。

3.4.7 位运算符及其表达式

位运算是对字节或字节内部的二进制进行测试、设置、移位或逻辑的运算,是 C 语言的重要特色之一,利用位运算可以实现与硬件密切相关的操作。位运算有两类:位逻辑运算和移位运算。实现这两种运算的运算符分别为位逻辑运算符和移位运算符,统称为位运算符。运用位运算符将操作数连接起来的式子称为位运算表达式。

1. 位逻辑运算符

位逻辑运算符有~、&、^和|四种,它们的类型、功能、运算规则如表 3-6 所示,优先级由高到低依次为~、&、^、|,其中,按位取反运算符~是单目运算符,所以是右结合性,其他三个运算符的优先级相同,且满足左结合性。

表 3-6 位逻辑运算符

类型	位逻辑运算符	功 能	操作数 1 位	操作数 2 位	结 果	运算规则描述
单目	~	按位取反	0		1	0 按位取反为 1,1 按位取反为 0
			1		0	
双目	&	按位与	0	0	0	只有两个操作数对应的位均为 1 时,结果才为 1,其他情况均为 0
			0	1	0	
			1	0	0	
			1	1	1	
	|	按位或	0	0	0	只有两个操作数对应位均为 0 时,结果才为 0,其他情况均为 1
			0	1	1	
			1	0	1	
			1	1	1	
	^	按位异或	0	0	0	两个操作数对应位的值相同时结果为 0,否则为 1
			0	1	1	
			1	0	1	
			1	1	0	

【例 3-6】 令 a=10,b=8,c=a&b,求 c 的值。

```
    0 0 0 0 1 0 1 0  (10)
&   0 0 0 0 1 0 0 0  (8)
-------------------
    0 0 0 0 1 0 0 0  (8)
```

所以 c=8。

【例 3-7】 令 a=12,b=9,c=a|b,求 c 的值。

```
    0 0 0 0 1 1 0 0  (12)
|   0 0 0 0 1 0 0 1  (9)
-------------------
    0 0 0 0 1 1 0 1  (13)
```

所以 c=13。

【例 3-8】　令 a=10,b=8,c=a^b,求 c 的值。

```
      0  0  0  0  1  0  1  0   (10)
^     0  0  0  0  1  0  0  0   (8)
-------------------------------
      0  0  0  0  0  0  1  0   (2)
```

所以 c=2。

【例 3-9】　a=8,c=～a,求 c 的值。

```
^     0  0  0  0  1  0  0  0   (8)
-------------------------------
      1  1  1  1  0  1  1  1   (245)
```

所以 c=245。

2. 移位运算符

移位运算是指对二进制操作数向左移或向右移的操作。移位运算符有左移(≪)和右移(≫),它们都是双目运算符,优先级相同,满足左结合性。移位运算的具体实现方式有循环移位、逻辑移位和算术移位。

1) 左移位运算符(≪)

左移位的运算规则是指在移位过程中,操作数的二进制位向左移动,右端空出的位补 0,左端移出的位被舍弃。从左移位的规则可以看出,对于无符号数而言,左移 n 相当于乘以 2^n。

其语法格式为 a≪n,其中 a 是操作数,可以是整型或字符型的变量或表达式,n 是移位的位数,只能是整数。功能是将 a 中的二进制位数向左移动 n 位。

例如,a=12,二进制形式为 00001100,则 a≪2 表示将各个二进位顺序左移 2 位,得到 00110000,即十进制数 48。

2) 右移位运算符(≫)

右移位的运算规则是指在移位过程中,操作数的二进制位向右移动,左端空出来的位补 0 或 1,右端移出的位被舍弃。这里补 0 还是补 1 取决于操作数是有符号数还是无符号数,具体如下。

(1) 对于无符号数向右移时,左端空出端补 0。

(2) 对于有符号数向右移时,如果采用逻辑位移,则不管是正数还是负数,左端一律补 0;如果采用算术位移,则正数右移,左端的空位全部补 0,负数右移,左端的空位全部补 1。Turbo C 和 Dev-C++编译采用算术右移。

其语法格式为 a≫n,其中 a 是操作数,可以是整型或字符型的变量或表达式;n 是移位的位数,只能是整数。功能是将 a 中的二进制位数向右移动 n 位。从右移位的规则可以看出,对于无符号数而言,右移 n 位相当于除以 2^n。

例如,a=12,二进制形式为 00001100,则 a≫2 表示将各个二进制位顺序右移 2 位,得到 00000011,即十进制数 3。

3. 复合赋值位运算符

除了位非运算符(～)外,其他位运算符均能和赋值运算符“=”结合起来构成复合赋值运算符,它们是:&=、|=、^=、≫=、≪=。

例如,对于 a=5,b=4,则有:

(1) a&=b 等价于 a=a&b=5&4=101&100=100=4。

(2) a|=b 等价于 a=a|b=5|4=101|100=101=5。

(3) a^=b 等价于 a=a^b=5^4=101^100=001=1。

(4) a≫=2 等价于 a=a≫2=5≫2=101≫2=1。

(5) a≪=2 等价于 a=a≪2=5≪2=101≪2=10100=20。

4. 位运算的特殊用途

通过位运算符可以实现一些特殊操作,如位与运算可以将操作数置零、取操作数中的某些位和将操作数的某些位保留,位或运算可以将操作数的某些位变为 1,位异或操作可以将操作数的特定位置反、保留原值。

【例 3-10】 已知 a=80,用位运算符实现以下操作:

首先,将 a 变成 0。

然后,将 a 的 3～5 位置 1,其他位不变。

最后,将 a 的低 4 位置反,其他位不变。

a 所对应的二进制为 01010000,分析过程如下。

首先,将 a 变成 0,根据位与运算规则,只要找到一个操作数,这个操作数具有以下特点:原数中为 1 的位,操作数中对应的位为 0,操作数的其他位可以是 0 和 1 中的任何情况,然后将两个数进行与运算。下面就是其中一种操作过程:

```
    0  1  0  1  0  0  0  0  (80)
&   0  0  0  0  1  0  0  1  (9)
-------------------------------
    0  0  0  0  0  0  0  0  (0)
```

然后,将 a 的 3～5 位置 1,其他位不变。根据位或运算规则,只要找到一个操作数,这个操作数满足以下特点:与原数这些位对应的操作数相应位为 1,其他位为 0,然后将两个数进行或运算。操作过程如下:

```
    0  1  0  1  0  0  0  0  (80)
|   0  0  1  1  1  0  0  0  (56)
-------------------------------
    0  1  1  1  1  0  0  0  (120)
```

最后,将 a 的低 4 位置反,其他位不变。根据位异或运算规则,只要找到一个操作数,这个操作数满足以下特点:与原数低 4 位对应的操作数相应位为 1,其他位为 0,然后将两个数进行异或运算。操作过程如下:

```
    0  1  0  1  0  0  0  0  (80)
^   0  0  0  0  1  1  1  1  (15)
-------------------------------
    0  1  0  1  1  1  1  1  (95)
```

3.5　常见数学运算表达式在 C 语言中的表示

本节主要讨论以下问题。

常见数学运算表达式在 C 语言中正确表达应该注意哪些问题?

在进行数据处理时，有很大一部分表达式是数学表达式。若选择 C 语言程序设计，就需要借助 C 语言丰富的运算符，将此数学表达式转换为 C 语言支持的表达式进行计算，才能得到正确的结果。转换规则如下所述。

1. 基本符号的变化

(1) 在 C 语言中，表达式的乘号不能省略。

(2) 数学中的分数符号“—”要改成 C 语言中的除号“/”。在转换成“/”时，要注意分子和分母的类型，必要时要进行类型转换，才能保证最终结果的等价性。如分式$\frac{2}{5}$直接转换成 2/5 就会出问题，因为前者的结果是 0.4，后者的结果为 0，所以此时要进行类型转换，可以写成 2.0/5 或者(float)2/5。

(3) 数学中的“≤”号要改成 C 语言中的“<=”号。

(4) 数学中的 π 和 e 是个常值，但在 C 语言中不能直接写 π 和 e，必须将它们的值直接写出或者将其定义为符号常量。

(5) 在 C 语言中，平方根不能直接使用，需要借助 C 中的库函数 sqrt。

(6) 数学中的绝对值“| |”符号，也不能直接使用，需要借助 C 中的库函数 abs 或者 fabs。

例如：

① 圆面积公式 $s=\pi r^2$，对应的 C 语言表达式为 s=3.14 * r * r。

② 球体积公式 $v=\frac{4}{3}\pi r^3$，对应的 C 语言表达式为 v=4.0/3 * 3.14 * r * r * r。

2. 借助库函数

除了基本符号的变化外，还有一些特殊符号，如根号、数的多次方、绝对值或者正弦、余弦等，若表达式出现了这些符号或运算时，就需要使用 C 语言提供的库函数。当程序中用到了 C 语言提供的库函数时，就需要告知程序这些库函数来自哪个头文件。像刚才提到的根号、绝对值和三角函数等，它们都需要使用头文件“math.h”中的相关数学库函数，因此，需要将文件包含预处理命令 #include "math.h"放在程序开头。具体 C 语言提供的库函数参见附录 E。例如：

(1) $\sqrt{a}$ 对应的 C 语言表达式为 sqrt(a)。

(2) sin60°对应的 C 语言表达式为 sin(60 * 3.1415926/180)。

(3) int a,b;|a+b|对应的 C 语言表达式为 abs(a+b)。

(4) a^3 对应的 C 语言表达式为 a * a * a。

(5) x^y 对应的 C 语言表达式为 pow(x,y)。

3.6　本章小结

3.6.1　知识梳理

本章所介绍的主要内容是 C 语言中的基本数据类型，即整型数据、实型数据和字符型数据的常量和变量，以及作用于这些数据类型的基本运算符及其表达式，即算术运算符及其表达式、自增自减运算符和负号运算符、赋值运算符及其表达式、强制类型转换运算符、逗号

运算符及其表达式、sizeof 运算符和位运算符及其表达式。

本章的内容比较繁杂,学起来比较枯燥,但本章的内容是学好 C 语言的基础,每个 C 语言程序员必须熟练掌握。在学习的过程中,不必过分强求记住每一个细节,可以在后续写程序的过程中遇到了再到本章来查阅相关需要注意的问题。这样,学习的目的性就更强,也更容易明白之前学习的意义,也就能更快更好地记住和掌握。本章知识导图如图 3-11 所示。

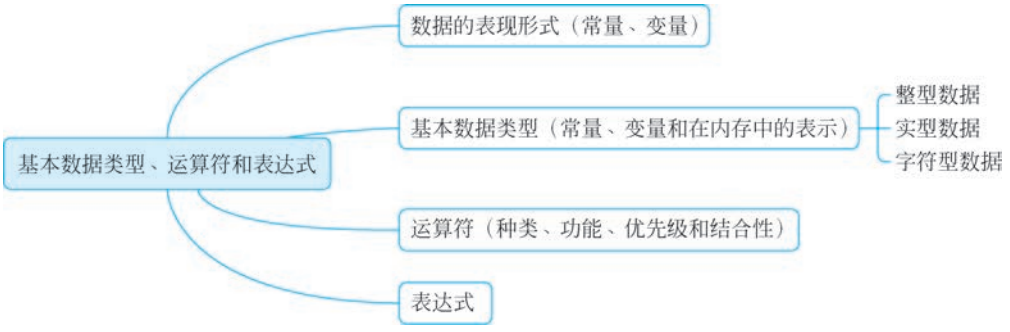

图 3-11 本章知识导图

3.6.2 常见上机问题及解决方法

1. 使用未定义的变量

C 语言要求变量必须先定义,后使用。

例如:

```
#include "stdio.h"
int main()
{
 a=1;b=a+2;      /*a,b 未定义*/
 printf("%d,%d",a,b);
 return 0;
}
```

2. 一行 C 语句后面漏掉“;”

C 语言要求每一条语句都要以“;”结尾,但在输入程序过程中常常会漏掉末尾的分号,特别是在使用++、--时。

例如:

```
#include "stdio.h"
int main()
{
  int a;              /*以下 3 条语句都漏掉了“;”*/
  scanf("%d",a)
  a++
  printf("%d",a)
  return 0;
}
```

3. 语序颠倒

表达式在进行计算时,要确保相应操作数都有确定的值,也就是有些变量被定义后,需要赋初值才能开始使用。因此,下面的程序输出的结果不是 5。

```
#include "stdio.h"
int main()
{
 int a,b;
 a = b + 4;          /* b 没有赋初值 */
 b = 1;
 printf("%d",a);
 return 0;
}
```

4. 混淆字符常量和字符串常量

C 语言中字符常量用一对单引号引起来，而字符串常量用一对双引号引起。例如：'A' 表示字符常量，而"A"则表示字符串常量。字符常量只能赋值给字符型变量，而字符串常量只能赋值给字符数组字符型指针或利用 strcpy 复制到字符数组中。

例如：

```
char ch,str[10];
char *p;
ch = 'M';
p = "12345";
strcpy(str,"abcd");
/* 以下语句则是错误的 */
ch = "A";
p = '1';
```

5. 定义变量时漏掉“;”

C 语言中变量定义的格式是“类型标识符 变量名 1[,变量名 2,…,变量名 n];”。但初学者常常在定义变量时，漏掉变量名后面的分号，尤其是同时定义多个变量时更是如此。

例如：

```
#include "stdio.h"
int main()
{
 int a,b,c          /* 连续定义 3 个变量，却忘记了“;” */
 a = 2;
 b = a * a;
 c = b * a;
 printf("%d",c);
 return 0;
}
```

6. 定义变量时数据类型关键字与变量名之间无空格

在输入程序时很容易漏掉类型标识符与变量名之间的空格。下面定义变量的方式是错误的。

```
inta,b;             /* int 和 a 之间没有空格 */
```

7. 对表达式进行强制类型转换时漏掉了“()”

在进行强制类型转换时，通常要将表达式和类型标识符都用“()”括起来。例如，(int)(f * 10/21)表示将 f * 10/21 的结果转换为 int 型。若漏掉了表达式的“()”，则变成了(int) f * 10/21，这样就表示先将 f 转换为 int 型再乘以 10 除以 21，其结果与(int)(f * 10/21)可能会不一样。

此外,如果强制类型转换运算符漏掉了类型标识符的“()”,便会出现语法错误,在编译时不能通过,例如“k=int(f*10/21);”是错误的。

8. 用'\'表示\

C语言中字符“\”是转义控制字符引导符,在C语言编译器对字符“\”的解释要看“\”后面跟的是什么字符,如果是“\n”,则表示换行;如果是“\'”,则表示字符单引号。但是,表示字符“\”就要用“'\\'”。

当字符串包含转义字符时,就要注意字符串的长度。例如,字符串"ab12\0245\d\\"的长度为8,这8个字符分别是:'a'、'b'、'1'、'2'、'\024'、'5'、'\d'和'\\'。

9. 使用关键字作为变量名

C语言中关键字都有各自的特殊用途。例如,int表示整型类型,用来定义整型变量;break用于跳出循环语句或switch语句。但是,初学者有时为了给变量取一个有意义的名字而又想不起其他更好的英文单词时,就会不经意地使用这些关键字来为变量取名。例如:

```
int break , int;    /* 关键字 break 和 int 都不能作为变量名 */
```

习题3

1. 如下变量在内存中的地址如何引用?

a　b　c　sum　average

2. 下列哪些是整型常量?

789　087　0x345　84a　0234　234

3. 下列哪些是实型常量?

8.12　1.　.12　123.123　11E2　2.e+3　3.34e3　5.4e9.9　e4　64.235e

4. 下列哪些是字符常量?

'a'　'ab'　'b'　'?'　'2'　'\234'　'\x23'　'\25'　'\x3a'

5. 下列数据在内存中分别占多少字节?

int a;　long b;　char c;double d;　"234"　"\234mnox"

6. 假设有以下定义:

```
int a;  float b;  char c;
```

下列表达式结果分别是什么类型?

① a+b　② b+c　③ (int)c+b　④ (int)(b+c)

7. 求下列表达式分别执行完后,变量a、b的值分别是多少(假设a的初值是8)?

① b=--a;

② b=a--;

8. 下列各表达式的值是多少?

```
int a = 5,b = 9,c = 7;
char d = 'A';
float e = 2.0,f = 1e1;
```

① b/a　② b%c　③ c/f　④ a+b-e

9. 计算下列赋值表达式的值(a、b 的值分别为 6 和 5)。

① a＝b　② a＋＝b　③ a＋＝b＊＝a　④ a－＝a/b

10. 选择题

(1) 以下叙述中错误的是(　　)。

A. 用户所定义的标识符允许使用关键字

B. 用户所定义的标识符应尽量做到“见名知意”

C. 用户所定义的标识符必须以字母或下画线开头

D. 用户定义的标识符中,大、小写字母代表不同标识

(2) 以下叙述中错误的是(　　)。

A. C 语句必须以分号结束

B. 复合语句在语法上被看作一条语句

C. 空语句出现在任何位置都不会影响程序运行

D. 赋值表达式末尾加分号就构成赋值语句

(3) 以下能正确定义且赋初值的语句是(　　)。

A. int n1＝n2＝10;　　B. char c＝32;

C. float f＝f＋1.1;　　D. double x＝12.3E2.5;

(4) 以下选项中可作为 C 语言合法常量的是(　　)。

A. －80.　　B. －080　　C. －8e1.0　　D. －80.0e

(5) 有以下程序

```
#include "stdio.h"
int main()
{   int m = 12,n = 34;
    printf("%d%d",m++,++n);
    printf("%d%d\n",n++,++m);
    return 0;
}
```

程序运行后的输出结果是(　　)。

A. 12353514　　B. 12353513　　C. 12343514　　D. 12343513

(6) 以下符合 C 语言语法的实型常量是(　　)。

A. 1.2E0.5　　B. 3.14.159E　　C. 5E－3　　D. E15

(7) 若以下选项中的变量已正确定义,则正确的赋值语句是(　　)。

A. x1＝26.8%3　　B. 1＋2＝x2

C. x3＝0x12　　D. x4＝1＋2＝3

(8) 以下叙述中正确的是(　　)。

A. C 语言程序中注释部分可以出现在程序中任意合适的地方

B. 大括号“{”和“}”只能作为函数体的定界符

C. 构成 C 语言程序的基本单位是函数,所有函数名都可以由用户命名

D. 分号是 C 语句之间的分隔符,不是语句的一部分

(9) 以下选项中可作为 C 语言合法整数的是(　　)。

A. 10110B　　B. 0386　　C. 0Xffa　　D. x2a2

(10) 下列关于单目运算符++、--的叙述正确的是(　　)。

A. 它们的运算对象可以是任何变量和常量

B. 它们的运算对象可以是 char 型变量和 int 型变量,但不能是 float 型变量

C. 它们的运算对象可以是 int 型变量,但不能是 double 型变量和 float 型变量

D. 它们的运算对象可以是 char 型变量、int 型变量和 float 型变量

(11) 若有以下程序:

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

执行后的输出结果是(　　)。

A. 8,6　　B. 8,3　　C. 6,4　　D. 7,4

(12) 以下选项中,与 k=n++完全等价的表达式是(　　)。

A. k=n,n=n+1　　B. n=n+1,k=n　　C. k=++n　　D. k+=n+1

(13) 以下选项中不属于 C 语言的类型的是(　　)。

A. signed short int　　B. unsigned long int

C. unsigned int　　D. long short

(14) 假定 x 和 y 为 double 型,则表达式 x=2,y=x+3/2 的值是(　　)。

A. 3.500000　　B. 3　　C. 2.000000　　D. 3.000000

(15) 下列选项中,合法的 C 语言关键字是(　　)。

A. VAR　　B. cher　　C. integer　　D. default

(16) 若 a 为 int 类型,且其值为 3,则执行完表达式 a+=a--=a*a 后,a 的值是(　　)。

A. -3　　B. 9　　C. -12　　D. 6

11. 程序设计题

(1) **本利计算**。小明每年过完春节后定期存入本金 capital 元,利率为 rate,存期为 n 年,编程帮助小明计算到期时能从银行得到的本利之和 deposit。假设利息公式如下:

$$利息=capital\times rate\times n$$

(2) **整数的算术运算**。输入两个整数 a 和 b,分别计算并输出两个整数的和、差、积、商和余数。