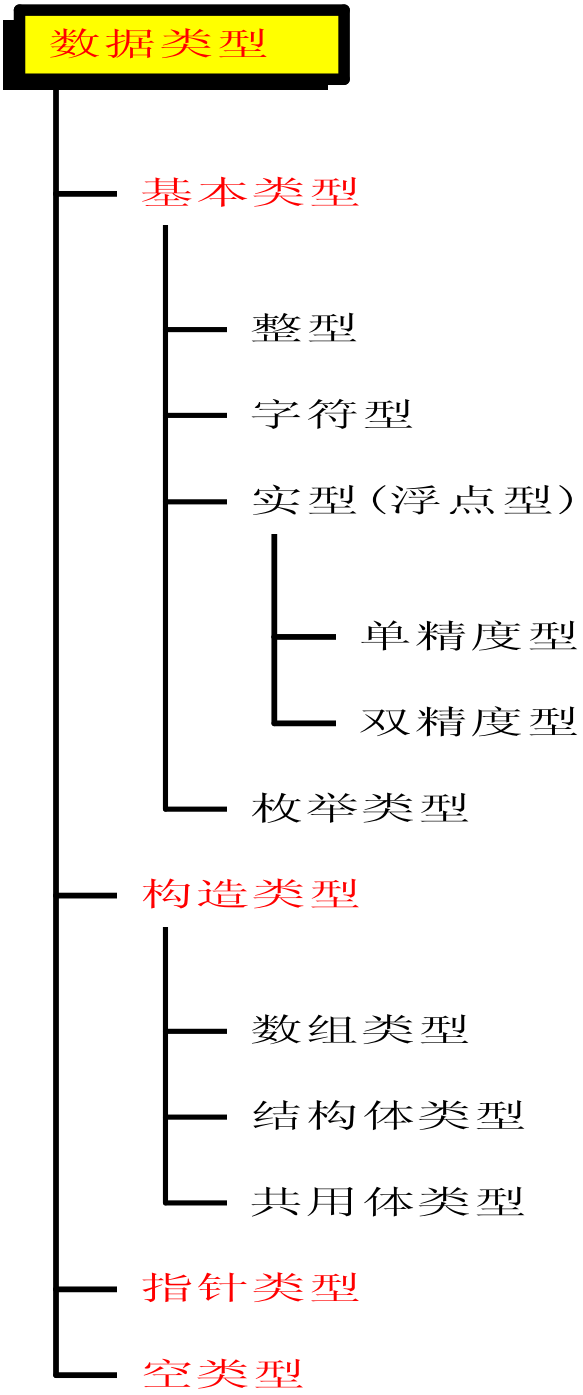
3 数据类型、运算符与表达式

1.1 C 语言的数据类型

在第一章中，我们已经看到程序中使用的各种变量都应预先加以定义，即先定义，后使用。对变量的定义可以包括三个方面：

- 数据类型
- 存储类型
- 作用域

在本章中，我们只介绍数据类型的说明。其它说明在以后各章中陆续介绍。所谓数据类型是按被定义变量的性质，表示形式，占据存储空间的多少，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它类型。也就是说，基本数据类型是自我说明的。
2. 构造数据类型：构造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。在 C 语言中，构造类型有以下几种：

- 数组类型
- 结构体类型
- 共用体（联合）类型

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。其值用来表示某个变量在内存储器中的地址。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。
4. 空类型：在调用函数值时，通常应向调用者返回一个函数值。这个返回的函数值是具有一定的数据类型的，应在函数定义及函数说明中给以说明，例如在例题中给出的 max 函数定义中，函数头为：int max(int a,int b);其中“int ”类型说明符即表示该函数的返回值为整型量。又如在例题中，使用了库函数 sin，由于系统规定其函数返回值为双精度浮点型，因此在赋值语句 s=sin (x);中，s 也必须是双精度浮点型，以便与 sin 函数的返回值一致。所以在说明部分，把 s 说明为双精度浮点型。但是，也有一类函数，调用后并不需要向调用者返回函数值，这种函数可以定义为“空类型”。其类型说明符为 void。在后面函数中还要详细介绍。

在本章中，我们先介绍基本数据类型中的整型、浮点型和字符型。其余类型在以后各章中陆续介绍。

1.1 常量与变量

对于基本数据类型量，按其取值是否可改变又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、浮点常量、浮点变量、字符常量、字符变量、枚举常量、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。整型量包括整型常量、整型变量。

1.1.1 常量和符号常量

在程序执行过程中，其值不发生改变的量称为常量。

- 直接常量(字面常量)：
  - 整型常量：12、0、-3；
  - 实型常量：4.6、-1.23；
  - 字符常量：‘a’、‘b’。
- 标识符：用来标识变量名、符号常量名、函数名、数组名、类型名、文件名的有效字符序列。
- 符号常量：用标示符代表一个常量。在 C 语言中，可以用一个标识符来表示一个常量，称之为符号常量。

  符号常量在使用之前必须先定义，其一般形式为：

  #define 标识符 常量

  其中#define 也是一条预处理命令（预处理命令都以"#"开头），称为宏定义命令（在后面预处理程序中将进一步介绍），其功能是把该标识符定义为其后的常量值。一经定义，以后在程序中所有出现该标识符的地方均代之以该常量值。
- 习惯上符号常量的标识符用大写字母，变量标识符用小写字母，以示区别。

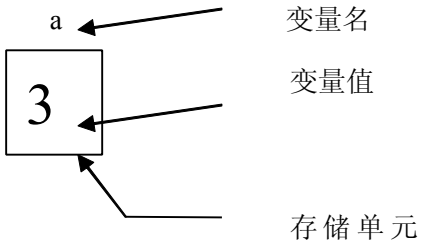
【例 3.1】符号常量的使用。

```
#define PRICE 30
main()
{
   int num,total;
   num=10;
   total=num* PRICE;
   printf("total=%d",total);
}
```

- 用标识符代表一个常量，称为符号常量。
- 符号常量与变量不同，它的值在其作用域内不能改变，也不能再被赋值。
- 使用符号常量的好处是：
  - 含义清楚；
  - 能做到“一改全改”。

1.1.1 变量

其值可以改变的量称为变量。一个变量应该有一个名字，在内存中占据一定的存储单元。变量定义必须放在变量使用之前。一般放在函数体的开头部分。要区分变量名和变量值是两个不同的概念。



1.1 整型数据

1.1.1 整型常量的表示方法

整型常量就是整常数。在C 语言中，使用的整常数有八进制、十六进制和十进制三种。

1) 十进制整常数：十进制整常数没有前缀。其数码为 0～9。

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)。

在程序中是根据前缀来区分各种进制数的。因此在书写常数时不要把前缀弄错造成

结果不正确。

2) 八进制整常数：八进制整常数必须以 0 开头，即以 0 作为八进制数的前缀。数码取值为 0～7。八进制数通常是无符号数。

以下各数是合法的八进制数：

015(十进制为 13)、0101(十进制为 65)、0177777(十进制为 65535)；

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)、-0127(出现了负号)。

3) 十六进制整常数：十六进制整常数的前缀为 0X 或 0x。其数码取值为 0~9，A~F 或 a~f。

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)、0X3H (含有非十六进制数码)。

4) 整型常数的后缀：在 16 位字长的机器上，基本整型的长度也为 16 位，因此表示的数的范围也是有限定的。十进制无符号整常数的范围为 0～65535，有符号数为-32768～+32767。八进制无符号数的表示范围为 0～0177777。十六进制无符号数的表示范围为 0X0～0XFFFF 或 0x0～0xFFFF。如果使用的数超过了上述范围，就必须用长整型数来表示。长整型数是用后缀“L”或“l”来表示的。

例如：

十进制长整常数：

158L (十进制为 158)、358000L (十进制为 358000)；

八进制长整常数：

012L (十进制为 10)、077L (十进制为 63)、0200000L (十进制为 65536)；

十六进制长整常数：

0X15L (十进制为 21)、0XA5L (十进制为 165)、0X10000L (十进制为 65536)。

长整数 158L 和基本整常数 158 在数值上并无区别。但对 158L，因为是长整型量，C 编译系统将为它分配 4 个字节存储空间。而对 158，因为是基本整型，只分配 2 个字节的存储空间。因此在运算和输出格式上要予以注意，避免出错。

无符号数也可用后缀表示，整型常数的无符号数的后缀为“U”或“u”。

例如：

358u,0x38Au,235Lu 均为无符号数。

前缀，后缀可同时使用以表示各种类型的数。如 0XA5Lu 表示十六进制无符号长整数 A5，其十进制为 165。

1.1.1 整型变量

1. 整型数据在内存中的存放形式

如果定义了一个整型变量 i：

```
int i;
i=10;
```

i 10

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

数值是以补码表示的：

- 正数的补码和原码相同；
- 负数的补码：将该数的绝对值的二进制形式按位取反再加 1。

例如：

求-10 的补码：

10 的原码：

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

取反：

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

再加 1，得-10 的补码：

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

由此可知，左面的第一位是表示符号的。

2. 整型变量的分类

1) 基本型：类型说明符为 int，在内存中占 2 个字节。
2) 短整量：类型说明符为 short int 或 short。所占字节和取值范围均与基本型相同。
3) 长整型：类型说明符为 long int 或 long，在内存中占 4 个字节。
4) 无符号型：类型说明符为 unsigned。

无符号型又可与上述三种类型匹配而构成：

- 无符号基本型：类型说明符为 unsigned int 或 unsigned。
- 无符号短整型：类型说明符为 unsigned short。
- 无符号长整型：类型说明符为 unsigned long。

各种无符号类型量所占的内存空间字节数与相应的有符号类型量相同。但由于省去了符号位，故不能表示负数。

有符号整型变量：最大表示 32767

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

无符号整型变量：最大表示 65535

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

下表列出了 Turbo C 中各类整型量所分配的内存字节数及数的表示范围。

类型说明符	数的范围	字节数
int	-32768~32767　即-2^{15}~（$2^{15}-1$）	2
unsigned int	0~65535　即 0~（$2^{16}-1$）	2
short int	-32768~32767　即-2^{15}~（$2^{15}-1$）	2
unsigned short int	0~65535　即 0~（$2^{16}-1$）	2
long int	-2147483648~2147483647 即-2^{31}~（$2^{31}-1$）	4
unsigned long	0~4294967295　即 0~（$2^{32}-1$）	4

以 13 为例：

int 型：

00	00	00	00	00	00	11	01

short int 型：

00	00	00	00	00	00	11	01

long int 型：

00	00	00	00	00	00	00	00	00	00	00	00	00	00	11	01

unsigned int 型：

00	00	00	00	00	00	11	01

unsigned short int 型：

00	00	00	00	00	00	11	01

unsigned long int 型：

00	00	00	00	00	00	00	00	00	00	00	00	00	00	11	01

3. 整型变量的定义

变量定义的一般形式为：

类型说明符　变量名标识符，变量名标识符，...；

例如：

int a,b,c; (a,b,c 为整型变量)

long x,y; (x,y 为长整型变量)

unsigned p,q; (p,q 为无符号整型变量)

在书写变量定义时，应注意以下几点：

- 允许在一个类型说明符后，定义多个相同类型的变量。各变量名之间用逗号间隔。类型说明符与变量名之间至少用一个空格间隔。
- 最后一个变量名之后必须以“；”号结尾。
- 变量定义必须放在变量使用之前。一般放在函数体的开头部分。

【例 3.2】整型变量的定义与使用。

```
main()
{
   int a,b,c,d;
   unsigned u;
   a=12;b=-24;u=10;
   c=a+u;d=b+u;
   printf(“a+u=%d,b+u=%d\n”,c,d);
}
```

4. 整型数据的溢出

【例 3.3】整型数据的溢出。

```
main()
{
   int a,b;
   a=32767;
   b=a+1;
   printf("%d,%d\n",a,b);
 }
```

32767：

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

-32768

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

【例 3.4】

```
main(){
   long x,y;
   int a,b,c,d;
   x=5;
   y=6;
   a=7;
   b=8;
   c=x+a;
   d=y+b;
   printf("c=x+a=%d,d=y+b=%d\n",c,d);
}
```

从程序中可以看到：x, y 是长整型变量，a, b 是基本整型变量。它们之间允许进行运算，运算结果为长整型。但 c，d 被定义为基本整型，因此最后结果为基本整型。本例说明，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以后介绍。

1.1 实型数据

1.1.1 实型常量的表示方法

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。它有二种形式：十进制小数形式，指数形式。

1) 十进制数形式：由数码 0~ 9 和小数点组成。

   例如：

   0.0、25.0、5.789、0.13、5.0、300.、-267.8230

   等均为合法的实数。注意，必须有小数点。

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（a 为十进制数，n 为十进制整数）

   其值为 $a*10^n$。

   如：

   2.1E5 (等于 $2.1*10^5$)

   3.7E-2 (等于 $3.7*10^{-2}$)

0.5E7 (等于 $0.5*10^7$)

-2.8E-2 (等于$-2.8*10^{-2}$)

以下不是合法的实数：

345 (无小数点)

E7 (阶码标志 E 之前无数字)

-5 (无阶码标志)

53.-E3 (负号位置不对)

2.7E　(无阶码)

标准 C 允许浮点数使用后缀。后缀为“f”或“F”即表示该数为浮点数。如 356f 和 356. 是等价的。

【例 3.5】说明了这种情况。

```
main(){
   printf("%f\n ",356.);
   printf("%f\n ",356);
   printf("%f\n ",356f);
}
```

1.1.1 实型变量

1. 实型数据在内存中的存放形式

实型数据一般占 4 个字节（32 位）内存空间。按指数形式存储。实数 3.14159 在内存中的存放形式如下：

+	.314159	1
数符	小数部分	指数

- 小数部分占的位（bit）数愈多，数的有效数字愈多，精度愈高。
- 指数部分占的位数愈多，则能表示的数值范围愈大。

2. 实型变量的分类

实型变量分为：单精度（float 型）、双精度（double 型）和长双精度（long double 型）三类。

在 Turbo C 中单精度型占 4 个字节（32 位）内存空间，其数值范围为 3.4E-38～3.4E+38，只能提供七位有效数字。双精度型占 8 个字节（64 位）内存空间，其数值范围为 1.7E-308～1.7E+308，可提供 16 位有效数字。

类型说明符	比特数（字节数）	有效数字	数的范围
float	32（4）	6~7	10^{-37}~10^{38}
double	64(8)	15~16	10^{-307}~10^{308}
long double	128(16)	18~19	10^{-4931}~10^{4932}

实型变量定义的格式和书写规则与整型相同。

例如：

float x,y; (x,y 为单精度实型量)

double a,b,c; (a,b,c 为双精度实型量)

3. 实型数据的舍入误差

由于实型变量是由有限的存储单元组成的，因此能提供的有效数字总是有限的。如下例 。

【例 3.6】实型数据的舍入误差。

```
main()
{float a,b;
 a=123456.789e5;
 b=a+20
   printf("%f\n",a);
   printf("%f\n",b);
}
```

注意：1.0/3*3 的结果并不等于 1。

【例 3.7】

```
main()
   {
    float a;
    double b;
    a=33333.33333;
    b=33333.33333333333333;
    printf("%f\n%f\n",a,b);
 }
```

- 从本例可以看出，由于 a 是单精度浮点型，有效位数只有七位。而整数已占五位，故小数二位后之后均为无效数字。
- b 是双精度型，有效位为十六位。但 Turbo C 规定小数后最多保留六位，其余部分四舍五入。

1.1.1 实型常数的类型

实型常数不分单、双精度，都按双精度 double 型处理。

1.1 字符型数据

字符型数据包括字符常量和字符变量。

1.1.1 字符常量

字符常量是用单引号括起来的一个字符。

例如：

'a'、'b'、'='、'+'、'?'

都是合法字符常量。

在 C 语言中，字符常量有以下特点：

1) 字符常量只能用单引号括起来，不能用双引号或其它括号。
2) 字符常量只能是单个字符，不能是字符串。
3) 字符可以是字符集中任意字符。但数字被定义为字符型之后就不能参与数值运算。如'5'和 5 是不同的。'5'是字符常量，不能参与运算。

1.1.1 转义字符

转义字符是一种特殊的字符常量。转义字符以反斜线"\"开头，后跟一个或几个字符。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，在前面各例题 printf 函数的格式串中用到的“\n”就是一个转义字符，其意义是“回车换行”。转义字符主要用来表示那些用一般字符不便于表示的控制代码。

常用的转义字符及其含义

转义字符	转义字符的意义	ASCII 代码
\n	回车换行	10
\t	横向跳到下一制表位置	9
\b	退格	8
\r	回车	13
\f	走纸换页	12
\\	反斜线符"\"	92
\'	单引号符	39
\”	双引号符	34
\a	鸣铃	7
\ddd	1～3 位八进制数所代表的字符	
\xhh	1～2 位十六进制数所代表的字符	

广义地讲，C 语言字符集中的任何一个字符均可用转义字符来表示。表中的\ddd 和\xhh 正是为此而提出的。ddd 和 hh 分别为八进制和十六进制的 ASCII 代码。如\101 表示字母"A"，\102 表示字母"B"，\134 表示反斜线，\XOA 表示换行等。

【例 3.8】转义字符的使用。

```
main()
{
  int a,b,c;
  a=5; b=6; c=7;
  printf(“   ab   c\tde\rf\n”);
  printf(“hijk\tL\bM\n”);
  }
```

1.1.1 字符变量

字符变量用来存储字符常量，即单个字符。

字符变量的类型说明符是 char。字符变量类型定义的格式和书写规则都与整型变量相

同。例如：

```
char a,b;
```

1.1.1 字符数据在内存中的存储形式及使用方法

每个字符变量被分配一个字节的内存空间，因此只能存放一个字符。字符值是以 ASCII 码的形式存放在变量的内存单元之中的。

如 x 的十进制 ASCII 码是 120，y 的十进制 ASCII 码是 121。对字符变量 a,b 赋予'x'和'y'值：

```
a='x';
b='y';
```

实际上是在 a,b 两个单元内存放 120 和 121 的二进制代码：

a：

0	1	1	1	1	0	0	0

b：

0	1	1	1	1	0	0	1

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。

整型量为二字节量，字符量为单字节量，当整型量按字符型量处理时，只有低八位字节参与处理。

【例 3.9】向字符变量赋以整数。

```
main()
{
  char a,b;
  a=120;
  b=121;
  printf("%c,%c\n",a,b);
  printf("%d,%d\n",a,b);
}
```

本程序中定义 a，b 为字符型，但在赋值语句中赋以整型值。从结果看，a，b 值的输出形式取决于 printf 函数格式串中的格式符，当格式符为"c"时，对应输出的变量值为字符，当格式符为"d"时，对应输出的变量值为整数。

【例 3.10】

```
  main()
  {
  char a,b;
  a='a';
  b='b';
  a=a-32;
  b=b-32;
  printf("%c,%c\n%d,%d\n",a,b,a,b);
```

}

本例中，a，b 被说明为字符变量并赋予字符值，C 语言允许字符变量参与数值运算，即用字符的 ASCII 码参与运算。由于大小写字母的 ASCII 码相差 32，因此运算后把小写字母换成大写字母。然后分别以整型和字符型输出。

1.1.1 字符串常量

字符串常量是由一对双引号括起的字符序列。例如： "CHINA" ， “C program”，"$12.5" 等都是合法的字符串常量。

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，字符串常量则可以含一个或多个字符。
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。在 C 语言中没有相应的字符串变量。这是与 BASIC 语言不同的。但是可以用一个字符数组来存放一个字符串常量。在数组一章内予以介绍。
4) 字符常量占一个字节的内存空间。字符串常量占的内存字节数等于字符串中字节数加 1。增加的一个字节中存放字符"\0" (ASCII 码为 0)。这是字符串结束的标志。

   例如：

   字符串 "C program" 在内存中所占的字节为：

C		p	r	o	g	r	a	m	\0

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

   'a'在内存中占一个字节，可表示为：

a

   "a"在内存中占二个字节，可表示为：

a	\0

1.1 变量赋初值

在程序中常常需要对变量赋初值，以便使用变量。语言程序中可有多种方法为变量提供初值。本小节先介绍在作变量定义的同时给变量赋以初值的方法。这种方法称为初始化。在变量定义中赋初值的一般形式为：

类型说明符 变量 1= 值 1，变量 2= 值 2，……；

例如：

```
int a=3;
int b,c=5;
float x=3.2,y=3f,z=0.75;
char ch1='K',ch2='P';
```

应注意，在定义中不允许连续赋值，如 a=b=c=5 是不合法的。

【例 3.11】

```
main()
```

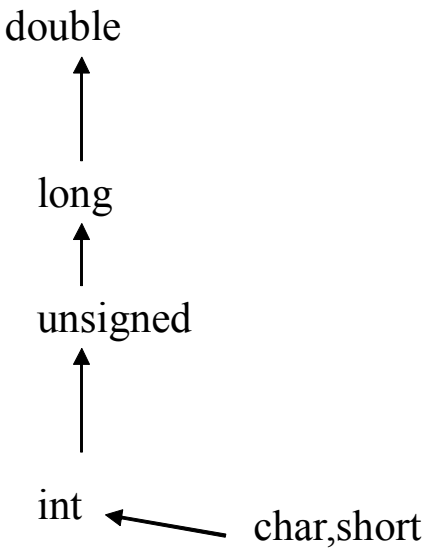
```
{
 int a=3,b,c=5;
 b=a+c;
 printf("a=%d,b=%d,c=%d\n",a,b,c);
}
```

1.1 各类数值型数据之间的混合运算

变量的数据类型是可以转换的。转换的方法有两种，一种是自动转换，一种是强制转换。自动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。
2) 转换按数据长度增加的方向进行，以保证精度不降低。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 单精度量运算的表达式，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。
5) 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左边长时，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。

下图表示了类型自动转换的规则。



【例 3.12】

```
main(){
float PI=3.14159;
int s,r=5;
s=r*r*PI;
```

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

本例程序中，PI 为实型；s，r 为整型。在执行 s=r*r*PI 语句时，r 和 PI 都转换成 double 型计算，结果也为 double 型。但由于 s 为整型，故赋值结果仍为整型，舍去了小数部分。

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。

其一般形式为：

(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 转换为实型

(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(单个变量可以不加括号)，如把(int)(x+y)写成(int)x+y 则成了把 x 转换成 int 型之后再与 y 相加了。
2) 无论是强制转换或是自动转换，都只是为了本次运算的需要而对变量的数据长度进行的临时性转换，而不改变数据说明时对该变量定义的类型。

【例 3.13】

```
main(){
  float f=5.75;
  printf("(int)f=%d,f=%f\n",(int)f,f);
}
```

本例表明，f 虽强制转为 int 型，但只在运算中起作用，是临时的，而 f 本身的类型并不改变。因此，(int)f 的值为 5(删去了小数)而 f 的值仍为 5.75。

1.1 算术运算符和算术表达式

C 语言中运算符和表达式数量之多，在高级语言中是少见的。正是丰富的运算符和表达式使 C 语言功能十分完善。这也是 C 语言的主要特点之一。

C 语言的运算符不仅具有不同的优先级，而且还有一个特点，就是它的结合性。在表达式中，各运算量参与运算的先后顺序不仅要遵守运算符优先级别的规定，还要受运算符结合性的制约，以便确定是自左向右进行运算还是自右向左进行运算。这种结合性是其它高级语言的运算符所没有的，因此也增加了 C 语言的复杂性。

1.1.1 C 运算符简介

C 语言的运算符可分为以下几类：

1. 算术运算符:用于各类数值运算。包括加(+)、减(-)、乘(*)、除(/)、求余(或称模运算，%)、自增(++)、自减(--)共七种。

2. 关系运算符:用于比较运算。包括大于(>)、小于(<)、等于(==)、 大于等于(>=)、小于等于(<=)和不等于(!=)六种。
3. 逻辑运算符:用于逻辑运算。包括与(&&)、或(||)、非(!)三种。
4. 位操作运算符:参与运算的量，按二进制位进行运算。包括位与(&)、位或(|)、位非(~)、位异或(^)、左移(<<)、右移(>>)六种。
5. 赋值运算符:用于赋值运算，分为简单赋值(=)、复合算术赋值(+=,-=,*=,/=,%=)和复合位运算赋值(&=,|=,^=,>>=,<<=)三类共十一种。
6. 条件运算符:这是一个三目运算符，用于条件求值(?:)。
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:用于取内容(*)和取地址(&)二种运算。
9. 求字节数运算符:用于计算数据类型所占的字节数(sizeof)。
10. 特殊运算符:有括号()，下标[]，成员(→，.)等几种。

1.1.1 算术运算符和算术表达式

1. 基本的算术运算符

- 加法运算符“+”：加法运算符为双目运算符，即应有两个量参与加法运算。如 a+b,4+8 等。具有右结合性。
- 减法运算符“-”：减法运算符为双目运算符。但“-”也可作负值运算符，此时为单目运算，如-x,-5 等具有左结合性。
- 乘法运算符“*”：双目运算，具有左结合性。
- 除法运算符“/”：双目运算具有左结合性。参与运算量均为整型时，结果也为整型，舍去小数。如果运算量中有一个是实型，则结果为双精度实型。

〖例 3.14〗

```
main(){
printf("\n\n%d,%d\n",20/7,-20/7);
printf("%f,%f\n",20.0/7,-20.0/7);
}
```

本例中，20/7，-20/7 的结果均为整型，小数全部舍去。而 20.0/7 和-20.0/7 由于有实数参与运算，因此结果也为实型。

- 求余运算符(模运算符)“%”:双目运算，具有左结合性。要求参与运算的量均为整型。求余运算的结果等于两数相除后的余数。

【例 3.15】

```
main(){
  printf("%d\n",100%3);
}
```

本例输出 100 除以 3 所得的余数 1。

2. 算术表达式和运算符的优先级和结合性

表达式是由常量、变量、函数和运算符组合起来的式子。一个表达式有一个值及其类型，它们等于计算表达式所得结果的值和类型。表达式求值按运算符的优先级和结合性规定的顺

序进行。单个的常量、变量、函数可以看作是表达式的特例。

算术表达式是由算术运算符和括号连接起来的式子。

- **算术表达式：**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 语法规则的式子。

以下是算术表达式的例子：

a+b

(a*2) / c

(x+r)*8-(a+b) / 7

++I

sin(x)+sin(y)

(++i)-(j++)+(k--)

- **运算符的优先级：**C 语言中，运算符的运算优先级共分为 15 级。1 级最高，15 级最低。在表达式中，优先级较高的先于优先级较低的进行运算。而在一个运算量两侧的运算符优先级相同时，则按运算符的结合性所规定的结合方向处理。
- **运算符的结合性：**C 语言中各运算符的结合性分为两种，即左结合性(自左至右)和右结合性(自右至左)。例如算术运算符的结合性是自左至右，即先左后右。如有表达式 x-y+z 则 y 应先与“-”号结合，执行 x-y 运算，然后再执行+z 的运算。这种自左至右的结合方向就称为“左结合性”。而自右至左的结合方向称为“右结合性”。 最典型的右结合性运算符是赋值运算符。如 x=y=z,由于“=”的右结合性，应先执行 y=z 再执行 x=(y=z)运算。C 语言运算符中有不少为右结合性，应注意区别，以避免理解错误。

3. 强制类型转换运算符

其一般形式为：

(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 转换为实型

(int)(x+y)　　把 x+y 的结果转换为整型

4. 自增、自减运算符

自增 1，自减 1 运算符:自增 1 运算符记为“++”，其功能是使变量的值自增 1。

自减 1 运算符记为“--”，其功能是使变量值自减 1。

自增 1，自减 1 运算符均为单目运算，都具有右结合性。可有以下几种形式：

++i　　i 自增 1 后再参与其它运算。

--i　　i 自减 1 后再参与其它运算。

i++　　i 参与运算后，i 的值再自增 1。

i--　　i 参与运算后，i 的值再自减 1。

在理解和使用上容易出错的是 i++和 i--。 特别是当它们出在较复杂的表达式或语句中时，常常难于弄清，因此应仔细分析。

【例 3.16】

```
main(){
int i=8;
printf("%d\n",++i);
printf("%d\n",--i);
printf("%d\n",i++);
printf("%d\n",i--);
```

```
  printf("%d\n",-i++);
  printf("%d\n",-i--);
}
```

i 的初值为 8，第 2 行 i 加 1 后输出故为 9；第 3 行减 1 后输出故为 8；第 4 行输出 i 为8 之后再加 1(为 9)；第 5 行输出 i 为 9 之后再减 1(为 8) ；第 6 行输出-8 之后再加 1(为 9)，第 7 行输出-9 之后再减 1(为 8)。

【例 3.17】

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

这个程序中，对 P=(i++)+(i++)+(i++)应理解为三个 i 相加，故 P 值为 15。然后 i 再自增 1 三次相当于加 3 故 i 的最后值为 8。而对于 q 的值则不然，q=(++j)+(++j)+(++j)应理解为 q 先自增 1，再参与运算，由于 q 自增 1 三次后值为 8，三个 8 相加的和为 24，j 的最后值仍为 8。

1.1 赋值运算符和赋值表达式

1. 赋值运算符

简单赋值运算符和表达式:简单赋值运算符记为“=”。由“= ”连接的式子称为赋值表达式。其一般形式为：

变量=表达式

例如：

x=a+b

w=sin(a)+sin(b)

y=i+++--j

赋值表达式的功能是计算表达式的值再赋予左边的变量。赋值运算符具有右结合性。因此

a=b=c=5

可理解为

a=(b=(c=5))

在其它高级语言中，赋值构成了一个语句，称为赋值语句。 而在 C 中，把“=”定义为运算符，从而组成赋值表达式。 凡是表达式可以出现的地方均可出现赋值表达式。

例如，式子:

x=(a=5)+(b=8)

是合法的。它的意义是把 5 赋予 a，8 赋予 b，再把 a,b 相加，和赋予 x，故 x 应等于 13。

在 C 语言中也可以组成赋值语句，按照 C 语言规定，任何表达式在其未尾加上分号就构成为语句。因此如

x=8;a=b=c=5；

都是赋值语句，在前面各例中我们已大量使用过了。

2. 类型转换

如果赋值运算符两边的数据类型不相同，系统将自动进行类型转换，即把赋值号右边的类型换成左边的类型。具体规定如下：

1) 实型赋予整型，舍去小数部分。前面的例子已经说明了这种情况。
2) 整型赋予实型，数值不变，但将以浮点形式存放，即增加小数部分(小数部分的值为 0)。
3) 字符型赋予整型，由于字符型为一个字节，而整型为二个字节，故将字符的 ASCII 码值放到整型量的低八位中，高八位为 0。整型赋予字符型，只把低八位赋予字符量。

【例 3.18】

```
main(){
int a,b=322;
float x,y=8.88;
char c1='k',c2;
a=y;
x=b;
a=c1;
c2=b;
printf("%d,%f,%d,%c",a,x,a,c2);
}
```

本例表明了上述赋值运算中类型转换的规则。a 为整型，赋予实型量 y 值 8.88 后只取整数 8。x 为实型，赋予整型量 b 值 322， 后增加了小数部分。字符型量 c1 赋予 a 变为整型，整型量 b 赋予 c2 后取其低八位成为字符型(b 的低八位为 01000010，即十进制 66，按 ASCII 码对应于字符 B)。

3. 复合的赋值运算符

在赋值符“=”之前加上其它二目运算符可构成复合赋值符。如+=,-=,*=, /=,%=,<<=,>>=,&=,^=,|=。

构成复合赋值表达式的一般形式为：

变量　双目运算符=表达式

它等效于

变量=变量 运算符 表达式

例如：

a+=5　　等价于 a=a+5

x*=y+7　　等价于 x=x*(y+7)

r%=p　　等价于 r=r%p

复合赋值符这种写法，对初学者可能不习惯，但十分有利于编译处理，能提高编译效率并产生质量较高的目标代码。

1.1 逗号运算符和逗号表达式

在C语言中逗号“,”也是一种运算符，称为逗号运算符。 其功能是把两个表达式连接起来组成一个表达式， 称为逗号表达式。

其一般形式为：

表达式 1，表达式 2

其求值过程是分别求两个表达式的值，并以表达式 2 的值作为整个逗号表达式的值。

【例 3.19】

```
main(){
int a=2,b=4,c=6,x,y;
y=(x=a+b),(b+c);
printf("y=%d,x=%d",y,x);
}
```

本例中，y 等于整个逗号表达式的值，也就是表达式 2 的值，x 是第一个表达式的值。对于逗号表达式还要说明两点：

1) 逗号表达式一般形式中的表达式 1 和表达式 2 也可以又是逗号表达式。

例如：

表达式 1，(表达式 2，表达式 3)

形成了嵌套情形。因此可以把逗号表达式扩展为以下形式：

表达式 1，表达式 2，…表达式 n

整个逗号表达式的值等于表达式 n 的值。

2) 程序中使用逗号表达式，通常是要分别求逗号表达式内各表达式的值，并不一定要求整个逗号表达式的值。

并不是在所有出现逗号的地方都组成逗号表达式，如在变量说明中，函数参数表中逗号只是用作各变量之间的间隔符。

1.1 小结

1.1.1 C 的数据类型

基本类型，构造类型，指针类型，空类型

1.1.1 基本类型的分类及特点

	类型说明符	字节	数值范围
字符型	char	1	C 字符集
基本整型	int	2	-32768～32767
短整型	short int	2	-32768～32767
长整型	long int	4	-214783648～214783647

无符号型	unsigned	2	0～65535
无符号长整型	unsigned long	4	0～4294967295
单精度实型	float	4	3/4E-38～3/4E+38
双精度实型	double	8	1/7E-308～1/7E+308

1.1.1 常量后缀

L 或 l　　长整型
U 或 u　　无符号数
F 或 f　　浮点数

1.1.1 常量类型

整数，长整数，无符号数，浮点数，字符，字符串，符号常数，转义字符。

1.1.1 数据类型转换

・自动转换:在不同类型数据的混合运算中，由系统自动实现转换，由少字节类型向多字节类型转换。不同类型的量相互赋值时也由系统自动进行转换，把赋值号右边的类型转换为左边的类型。
・强制转换:由强制转换运算符完成转换。

1.1.1 运算符优先级和结合性

一般而言，单目运算符优先级较高，赋值运算符优先级低。算术运算符优先级较高，关系和逻辑运算符优先级较低。多数运算符具有左结合性，单目运算符、三目运算符、赋值运算符具有右结合性。

1.1.1 表达式

表达式是由运算符连接常量、变量、函数所组成的式子。 每个表达式都有一个值和类型。表达式求值按运算符的优先级和结合性所规定的顺序进行。