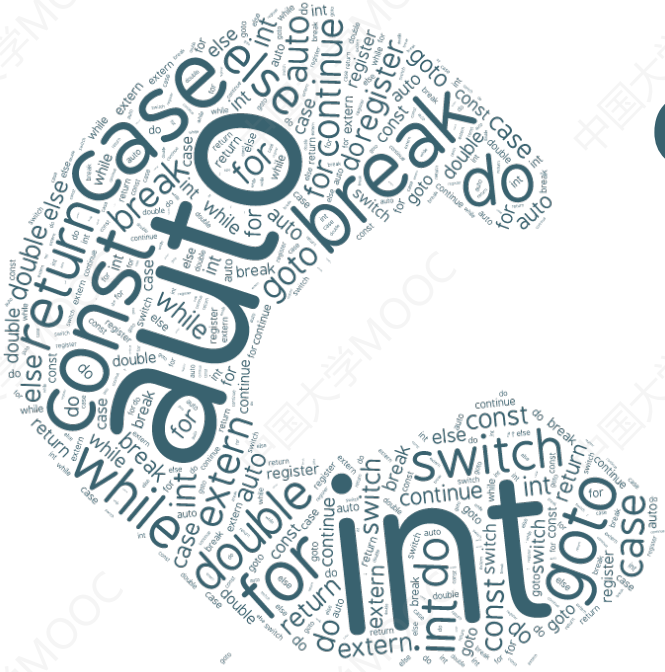


chapter 2

数据类型、运算符和表达式

本章问题

1. 如何准确地书写各种常量?

2. 如何定义(说明)各种类型的变量?

3. 如何准确地利用各种运算符书写表达式?

4. 如何准确地对数据进行输入输出?

目录
content

2.1 数据类型

2.2 常量

2.3 变量

2.4 运算符和表达式

2.5 位运算

2.6 各类数值型数据间的混合运算

2.7 输入输出的进一步讨论

2.1 数据类型

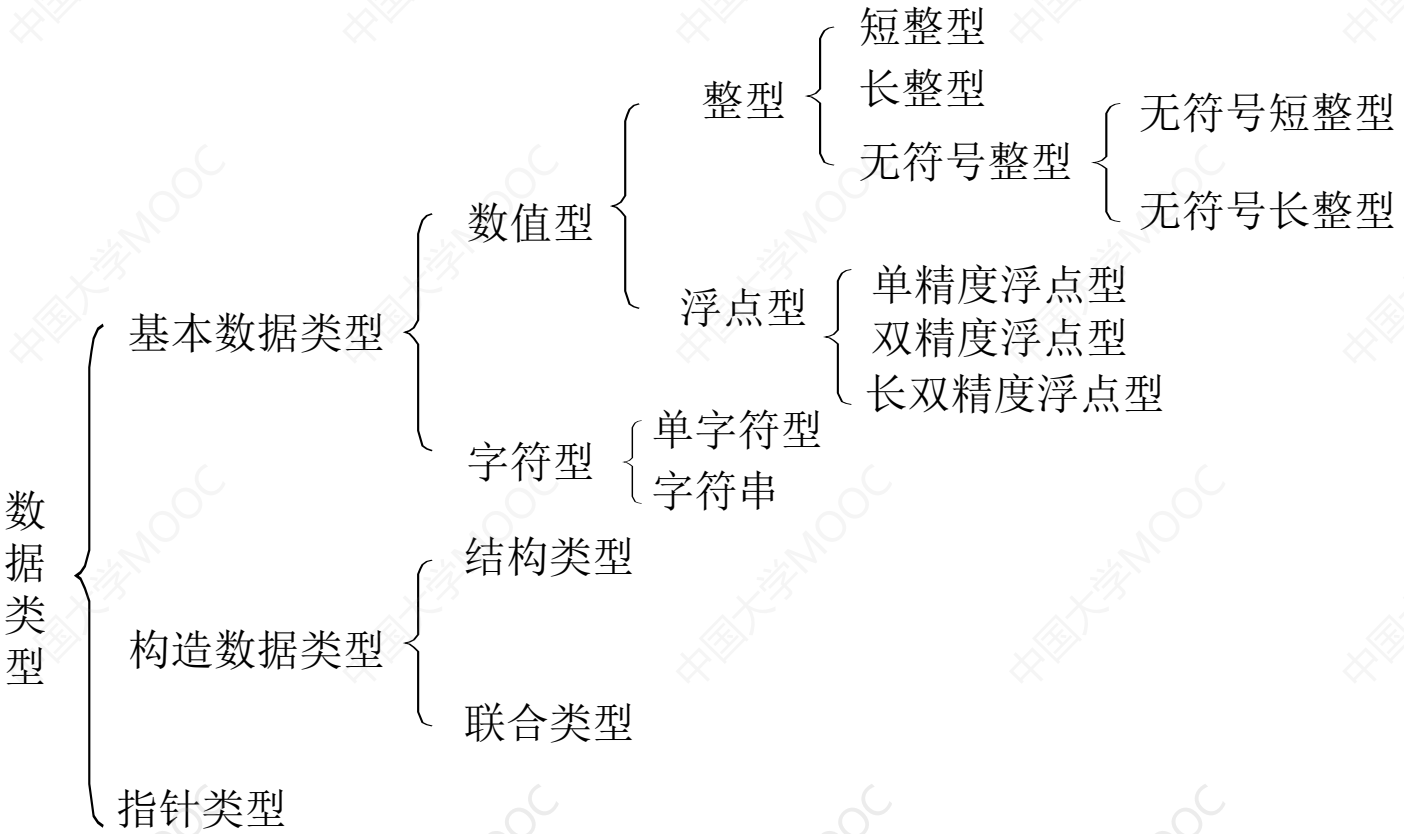
数据处理是计算机的基本功能之一，数据处理的对象是数据。

在高级语言程序设计中，数据在计算机中的存储长度决定数据值的范围。

为了数据存储和处理的需要，C编译程序将数据划分为不同的数据类型，并为每一种数据类型规定了在内存中的存储单元字节数和对该数据类型数据所能进行的运算。

数据类型

在C语言中，按**被定义数据的性质、表达形式、占存储空间以及构造特点其**数据类型分为：



2.2.1 整型常量

- **十进制**整型常量：例如 56、-100、2004；
- **八进制**整常量：**以0开头**,数码取值为0～7。如：017(十进制为15)、0101(十进制为65)、0177777(十进制为65535)
- **十六进制**整常量：前缀为0X或0x，其数码取值为0～9，A～F或a～f。如：0X2A(十进制为42)、0xA0 (十进制为160)、0XFFFF (十进制为65535)；
- **整型常量的后缀**：长整型数是用后缀**“L”或“l”**来表示的。如：158L (十进制为158),012L (十进制为10)

2.2.2 浮点型常量

两种表示形式：

十进制小数形式

指数形式

(1) 十进制小数形式：由数字**0～9**和**小数点**组成(注意必须有小数点)如：0.0、3.14 、.56、300.

(2)指数形式：

[±] [整数部分] [.] [小数部分] [(e,E)±n] [后缀]

如：-1.23456e+4

两条规则：

1) 一个浮点数可以无整数部分或小数部分，但不能二者全无；

2) 一个浮点数可以无小数点或指数部分，但不能二者全无。

2.2.3 字符型常量

字符型常量:用**单引号括起**来的一个字符 ，单引号中的内容不能是单引号，双引号和反斜线。字符常量的值就是该字符的ASCII码值。一个字符常量在内存中只占一个字节

如：‘a’ 、‘C’ 、‘ ’ =’ 、‘+’ 、‘?’

转义字符是一种特殊的字符型常量。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。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，

（常用的转义字符说明见下页）

2.2.3 字符常量--转义字符表示

字符类型	字符表示	字符含义	ASCII码值	“\ddd”表示	“\xhh”表示
字母	‘a’	字母（a）	97	\141	\x61
数字	‘0’	数字（0）	48	\060	\x30
特殊字符	‘\a’	鸣铃	7	\007	\x07
	‘\’’	单引号符（’）	96	\140	\x60
	‘\”’	双引号符（”）	34	\042	\x22
	‘\\’	反斜线符（\）	92	\134	\x5C
空格符	‘\n’	回车换行	10	\012	\x0A
	‘\f’	走纸换页	12	\014	\x0C
不能显示的字符	‘\0’	空字符	0	\000	\x00
	‘\b’	退格	8	\010	\x08
	‘\r’	回车	13	\015	\x0D

八进制数

2.2.3 转义字符使用例

```
#include <stdio.h>
void main( )
{
    char ch;
    ch = '\44';  //将ASCII码为44(八进制)即36(十进制)的字符赋给ch

    printf("ch is %c\n",ch);
                //输出字符，ASCII码为36对应的字符为$
}
```

\ddd的转义字符形式

运行结果：
ch is $

字符串常量

字符串常量：由一对**双引号括起**的字符序列
如：“ CHINA”,” C program”等。

字符串“CHINA”在内存中所占的字节为：

‘C’	‘H’	‘I’	‘N’	‘A’	‘\0’

2.2.5 字符型常量与字符串常量区别

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。

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符数加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。

2.2.6 符号常量

其定义的一般形式为：

#define 标识符 常量

其功能是用标识符代替后面常量。

如： #define G 9.8

使用符号常量的好处是：含义清楚；能做到“一改全改”。

```
#include <stdio.h>
#define PI 3.14159      //定义符号常量PI, 值为3.14159
void main( )
{
   double radius = 10.0;
   double perimeter;
   double area;

   perimeter = 2 * PI * radius;        //使用符号常量
   area = PI * radius *radius;         //使用符号常量
      printf(“radius=%lf,perimeter=%lf,area=%lf\n”,  \
                                        radius,perimeter,area);
}
```

运行结果：
radius=10.000000,perimeter=62.831800,area=314.159000

变量

变量：在程序执行过程中，值可以改变的量。

- 变量定义的格式:

<存储类型> 数据类型 变量名 <= 初值>;

如： int i ; //定义了一个int 型(整型)变量i;

```
short int a;          unsigned int cd;          //整型变量
long int b;           unsigned long int ud;     //整型变量
char cc;              signed char c2;           //字符型变量
float f1;                                       //浮点型变量
double df;            long double ldf;          //浮点型变量
int j = 2;
```

变量

- 一条语句定义多个相同数据类型变量,格式为:

  <存储类型> 数据类型 变量名表;

如:

```
int i , j , k ;                    // i,j,k是变量名, int 是数据类型。
double d1,d2,d3,d4;        // 定义d1,d2,d3,d4四个双精度型变量。
float f1=1.0, f2=2.0,f3=5.0;   //定义三个浮点型变量并分别赋值。
```

2.3.1 一种特殊的变量--指针变量

指针变量：用于存储地址的变量

定义格式为

<存储类型> 数据类型 *变量名 <= 地址值>;

如：

1） float *pf; // 定义一个浮点型指针变量

2）在预先定义整型变量i基础上，可定义指针变量pa：

```
int *pa = & i;
```

2.3.2 变量命名遵循的原则

- 必须是以英文字母或下划线开头的，由字母、数字和下划线组成的字符序列。
- 不能与C语言的关键字(保留字)重名，
- C语言对变量名的大小写敏感。
- 另外，在C语言的长期使用过程中还形成了一些约定俗成的规则：
  - 尽量使变量名能够表达出该变量的含义。
  - 用户最好不要用下划线来作为变量名的开头。
  - 习惯上符号常量的标识符用大写字母，变量标识符可大小写结合(不全用大写)。

2.3.3 BC31中数据类型的长度与值域

类型名	说明	值域	字节数
int	整型	-32768 ~ 32767	2
signed int	有符号整型	-32768 ~ 32767	2
unsigned int	无符号整型	0 ~ 65535	2
short int	短整型	-32768 ~ 32767	2
signed short int	有符号短整型	-32768 ~ 32767	2
unsigned short int	无符号短整型	0 ~ 65535	2
long int	长整型	-2147483648 ~ 2147483647	4
signed long int	有符号长整型	-2147483648 ~ 2147483647	4
unsigned long int	无符号长整型	0 ~ 4294967295	4
float	单精度浮点型	约$\pm\|3.4\times10^{-38}\|$ ~ $\pm\|3.4\times10^{38}\|$ 有效数位7位	4
double	双精度浮点型	约$\pm\|1.7\times10^{-308}\|$ ~ $\pm\|1.7\times10^{308}\|$ 有效数位15位	8
long double	长双精度浮点型	约$\pm\|3.4\times10^{-4932}\|$ ~ $\pm\|3.4\times10^{4932}\|$ 有效数位18位	10
char	字符型	-128 ~ 127	1
signed char	有符号字符型	-128 ~ 127	1
unsigned char	无符号字符型	0 ~ 255	1

2.3.4 字符变量与整型量之间的联系

```
#include <stdio.h>
void main( )
{
    char ch = 'a';
    int i = ch;
    printf ("%c ASCII is %d\n",ch,ch);   //将字符量按整型量处理
    printf ("%c ASCII is %d\n",i,i);     //将整型量按字符量处理
}
```

运行结果：
a ASCII is 97
a ASCII is 97

2.4 运算符和表达式

运算符分类：

- 按运算对象的数目：

  单目运算符

  双目运算符

  三目运算符

- 按照其功能：

算术运算符、赋值运算符、

关系运算符、 逻辑运算符、

位运算符、自增自减运算符、

条件运算符、逗号运算符等等。

由**运算符**和**运算量**所组成的符合C的语法的算式。

- C语言中的表达式分类：

  算术表达式、关系表达式、

  逻辑表达式、赋值表达式、

  条件表达式、逗号表达式

  混合表达式等。

注：无论什么表达式，都会返回一个结果(或值)。

2.4.1 算术运算符与算术表达式

算术运算符	使用形式	含义
+	单目或双目运算符	单目运算表示正号，双目运算表示加法
-	单目或双目运算符	单目运算表示减号，双目运算表示减法
*	双目运算符	乘法运算
/	双目运算符	除法运算
%	双目运算符	取模运算（求余数）

运算规则：

1. +、-、*、/运算符的运算量可为任何整型或浮点型的常量、变量、有返回值的函数以及表达式。
2. 在x/y中，表达式y的取值也不能为0，否则将出现错误。
3. %运算符要求运算量必须是整型，且%后面的运算量不能为0。
4. 当双目运算符的两个操作数的数据类型相同时，它们的运算结果的类型与操作数类型相同。
5. **当双目运算符的两个操作数的类型不同时，运算前遵循类型的一般转换规则将运算量自动转换成相同的类型，运算结果的类型与转换后的运算量的类型相同**

例2.4 五种算术运算示例

```
#include <stdio.h>
void    main( )
{
    int      x, y;
    float    x1, y1;

    x = 15;
    y = 6;
    x1 = 15.0;
    y1 = 6.0;

    printf("x = %d , y = %d\n", x, y);
    printf("x + y = %d\n", x + y);
    printf("x – y = %d\n", x - y);
    printf("x * y = %d\n", x * y);
    printf("x / y = %d....%d\n", x / y,   x % y);
    printf("x1 / y1 = %f \n", x1 / y1);
}
```

特别关注求余运算

运行结果：

```
x = 15 , y = 6
x + y = 21
x – y = 9
x * y = 90
x / y = 2....3
x1 / y1 = 2.500000
```

2.4.2 关系运算符与关系表达式

- C语言中的关系运算符：

< (小于)　　<=(小于或等于)　　= =(等于)

> (大于)　　>=(大于或等于)　　!= (不等于)

- 说明：

1)关系运算符都是双目运算符，它用来比较两个运算量之间的关系。

2)用关系运算符将前、后两个运算量连接起来的式子称为“关系表达式”，这两个运算量可以是任意表达式。

- **3) 当关系表达式成立时，表达式的结果为整数1，否则为整数0 。**

例2.5 关系表达式示例

```
#include <stdio.h>

void main( )
{
    char ch1,ch2;

    ch1 = 'a';
    ch2 = 'b';

    printf("%c == %c----%d\n",ch1,ch2,ch1 == ch2);
    printf("%c  < %c----%d\n",ch1,ch2,ch1 < ch2);
    printf("%c  > %c----%d\n",ch1,ch2,ch1 > ch2);
    printf("%c  <= %c----%d\n",ch1,ch2,ch1 <= ch2);
    printf("%c  >= %c----%d\n",ch1,ch2,ch1 >= ch2);
    printf("%c  != %c----%d\n",ch1,ch2,ch1 != ch2);
}
```

运行结果：

```
a == b----0
a  < b----1
a  > b----0
a <= b----1
a  >= b----0
a  != b----1
```

2.4.3 逻辑运算符与逻辑表达式

- C语言中的逻辑运算符

  &&(逻辑与)

  ||　(逻辑或)

  !　(逻辑非)

- 简单示例：

  X = 0； Y = 2；

则：　X && Y ---- 0

　　X || Y　　---- 1

　　!Y　　--- -0

- 逻辑运算真值表

表达式X	表达式Y	!X	!Y	X&&Y	X||Y
非0	非0	0	0	1	1
非0	0	0	1	0	1
0	非0	1	0	0	1
0	0	1	1	0	0

例2.6 逻辑运算符&&使用示例

```
#include <stdio.h>

void main( )
{
    int a , b , c , max;

    a = 10;
    b = 20;
    max = b;
    c = ( a > b ) && ( max = a ) ;

    printf("a = %d , b = %d , c = %d , max  =%d\n",a,b,c,max);
}
```

(a>b)的结果为0，不再运算右表达式 (max=a)

运行结果：
a = 10 , b = 20 , c = 0 , max = 20

规定：

1) 对于逻辑与（&&）运算,若左表达式为“假”，则无需判断右表达式的值即可以断定逻辑表达式的值为假；只有当左表达式为“真”时,才需要继续判断右表达式。

2) 对于逻辑或（||）运算，当左表达式为“真”时，则无需判断右表达式的值即可以断定逻辑表达式的值为真；只有当左表达式为“假”时,才需要继续判断右表达式。

2.4.4 自增和自减运算

- 自增、自减运算符分别为:(**单目运算符**)

  ++(自增)

  --（自减)

- ++和--分别都有两种不同的形式:

前置式: ++i、--i

后置式: i++、i--

运算说明:

1. **前置运算**是变量先自增1或自减1后,再参与其他的运算，即**先变后用**。

如:x = 0 ; y = -- x + x;

结果为x=-1,y=-2

2. **后置运算**是该变量先以原来的值参加其它运算，然后再自增1或自减1,即**先用后变**;

如: x = 10 ; y = x++ + x;

结果为x = 11 , y =20;

3. **自增自减运算符只能作用于变量，不能用于常量和表达式。**

例2.7 自增自减运算符的运算示例

```
#include <stdio.h>

void main( )
{
      int x,y;

      x = 0;
      y = 10;

      printf("x = %d , y = %d\n", x ++ , -- y);
      printf("x = %d , y = %d\n", x , y);
      printf("x = %d , y = %d\n", ++x , y--);
}
```

变量x先输出值，再自增，变量y先自减,再输出值

变量x先自增，再输出值，变量y先输出值，再自减

运行结果：

```
x = 0 , y = 9
x = 1 , y = 9
x = 2 , y = 9
```

2.4.5 赋值运算符与赋值表达式

- **基本赋值运算符** “=” 是**双目运算符**,其表达式形式为:

  左值表达式 = 右值表达式

如:

```
int      i , j ;
char     m , n ;
float    x , y ;
double   z ;
```

j = i　　i和j的类型相同,无需转换,直接将i的值赋给j

i = m　　m由char型向int型转换，将转换后的值赋给i

z = x * i　　x* i的结果为double型，然后赋值给z

i = m < n　　m < n的结果为整型，无需转换，直接将值赋给i

i = j = 10　　这是一个多重赋值表达式，赋值运算符按从右至左结合，即相当于i=(j=10),先将10赋给j,而括号中的赋值表达式（j=10）的值就是赋值后的j的值,再将其赋给i。

2.4.5 赋值运算符与赋值表达式

- **复合赋值运算符是**在赋值运算符“=”**前加上其他运算符构成。**C语言中的复合赋值运算符有10种：+=、-=、*=、/=、%=、&=、|=、^=、<<=、>>=。其表达式形式为：

  左值表达式 op= 右值表达式

  等价于 **左值表达式 = 左值表达式 op 右值表达式。**

如：

i += j;　　等价于 i = i + j;

x *= y – 5;　　等价于 x = x * (y-5)

m <<= 2;　　等价于 m = m << 2

2.4.6 条件运算符与条件表达式

- 一般形式：

表达式1 ？ 表达式2 ： 表达式3

- 操作过程：

判断表达式1的值，如果为非0值，则求解表达式2的值，并将其作为该条件表达式的值；

如果表达式1的值为0，则求解表达式3的值，并将其作为该条件表达式的值。

- 示例：如 a = 10， b = -5，求以下条件表达式的结果值。

```
c = (b > 0 ? (a + b) :( a - b));
```

结果为c = 15

计算表达式3的结果作为整个条件表达式的结果。

表达式1为关系表达式，其结果值为0

详细代码见p31例 2.8

2.4.7 逗号运算符与逗号表达式

- 逗号运算符是**顺序运算符**，用它构成的逗号表达式形式为：

表达式1，表达式2，表达式3，……，表达式n

- 运算过程：先求表达式1的值，然后再求表达式2的值，依次计算下去，最后表达式n的值也就是该逗号表达式的值。
- 示例1：如果定义整型变量a和 b,且a=10,求以下表达式的结果;

```
b = (a++ , a % 3) ;
```

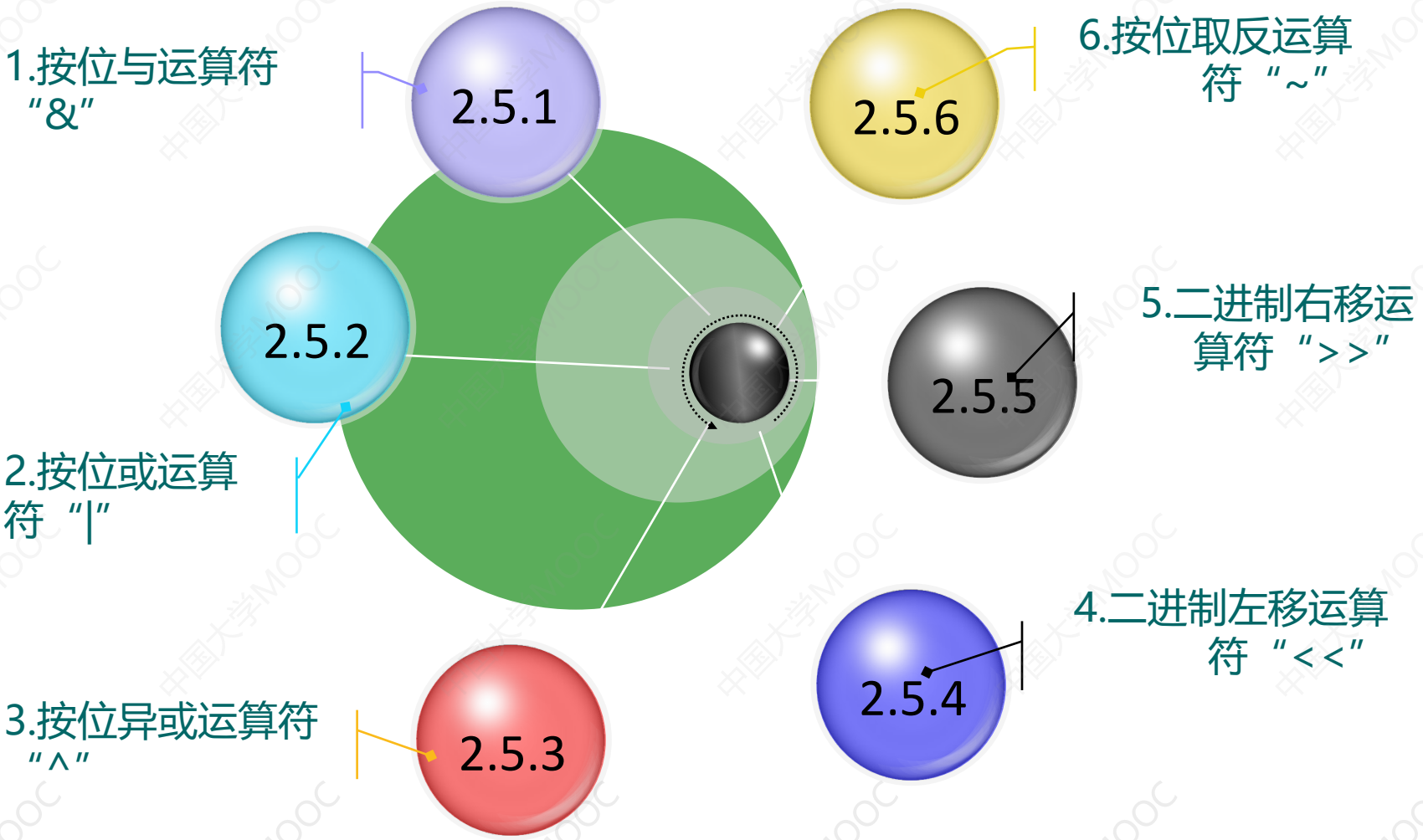
表达式2的结果为2，同时也是整个逗号表达式的结果，然后把2赋值给b。

表达式1的结果值10

结果为： b = 2

思考： 如果 b = (a++,a % 3),换成 b = (a++ , a++,a % 3) 结果为多少 ？

2.5 位运算


1.按位与运算符“&”
2.5.1
6.按位取反运算符“~”
2.5.6
2.5.2
5.二进制右移运算符“>>”
2.5.5
2.按位或运算符“|”
4.二进制左移运算符“<<”
2.5.4
3.按位异或运算符“^”
2.5.3

2.5.1 按位与运算符“&”

- 按位与运算是对两个操作数**逐位“求与”**。
- 运算真值表：

位1	位2	位1&位2
0	0	0
0	1	0
1	0	0
1	1	1

- 示例:a=0x96,b=0x80,求a&b.

```
  a=0x96  二进制表示 1001 0110
& b=0x80  二进制表示 1000 0000
-----------------------------
=   0x80  二进制表示 1000 0000
```

2.5.1 按位与的作用

(1)将某些位清零

例如：a=0x55,**要将a的低四位清零**，那就要将a与一**0xf0**进行按位与运算。

运算过程如下：

```
  a=0x55 二进制表示 0101  0101
& b=0xf0 二进制表示 1111  0000
-------------------------------
=   0x50 二进制表示 0101  0000
```

(2)取数中的特定位

例如a=0x55,**要将保持a的低四位**而其它位清零，那就要将a与**0x0f**进行按位与运算。

运算过程如下：

```
  a=0x55 二进制表示 0101  0101
& b=0x0f 二进制表示 0000  1111
-------------------------------
=   0x05 二进制表示 0000  0101
```

2.5.2 按位或运算符“|”

- 按位或运算：**对两个操作数逐位“相或”。**

- 运算真值表

位1	位2	位1|位2
0	0	0
0	1	1
1	0	1
1	1	1

- 示例:

a=0x36,b=0x55,求a | b。

```
   a=0x36  二进制表示 0011  0110
 | b=0x55  二进制表示 0101  0101
-------------------------------------
 =    0x77  二进制表示 0111  0111
```

2.5.2 按位或的作用

(1)将数的某些位置1

如a=0x55,要将a的**低四位置1**，那就要将a与**0x0f**进行按位或运算

运算过程如下：

```
  a=0x55 二进制表示 0101  0101
| b=0x0f  二进制表示 0000  1111
-------------------------------
=   0x5f  二进制表示 0101  1111
```

(2)把一串二进制数连接到另一串二进制数后

- 在实际应用中有时也需要将一串二进制数**连接到**另一串二进制数后。
- 如a=0x55，要连接的数据为8位二进制串**0xaa**.

运算过程如下：

```
  a=0x55＋8 个 0.  0101  0101  0000  0000
| 0xaa                0000  0000  1010  1010
-----------------------------------------------
=   0x55aa            0101  0101  1010  1010
```

2.5.3 按位异或运算符“^”

- 按位异或运算：**将两个操作数逐位“相异或”。**
- 运算真值表：

位1	位2	位1^位2
0	0	0
0	1	1
1	0	1
1	1	0

- 示例：

a = 0x36，b = 0x0f，求a^b.

```
    a=0x36   二进制为 0011 0110
^   b=0x0f   二进制为 0000 1111
-------------------------------
=     0x39   二进制为 0011 1001
```

从所得的结果看，某位要保持不变就异或0，某位要取反就异或1。

2.5.4 二进制左移运算符“<<”

- **运算规则**：把数据**向左移动若干位,移出左边界的所有位都将丢失,右侧新增加的位为0。**

- 示例：int a = 4 , a << 2的结果为16

  二进制 0000 0100 << 2 为0001 0000

2.5.5 二进制右移运算符“>>”

- **运算规则**：二进制右移运算符把**数据向右移动若干位,移出右边界的所有位都将丢失,**左侧的新位的补充遵循下面的规则：

(1) 对于无符号数,右移时左侧的新位一律补0,称为“逻辑右移”

(2) 对于有符号数,若符号位是0,则左侧新位一律补0；若符号位是1,则左侧新位一律补1,称为“算术右移”

- 示例1： 变量a**无符号数**，a = 8，二进制表示为**0000 1000**， a >> 2(a右移两位)，得：0000 0010，结果为2
- 示例2：变量b**有符号数**，b = -10，二进制补码为**1111 0110**， b >> 1(b右移一位,左侧补1)， 得：1111 1011， 结果为 -5

2.5.6 按位取反运算符“~”

- 按位取反运算符是将**操作数进行逐位“取反”**。

- 例如:

  变量 a = 0x6a，二进制表示为0110 1010，按位取反后为10010101，所以~a的结果为0x95。

2.6 各类数值型数据间的混合运算

2.6.1 自动类型转换

- 在C语言中字符型、整型和浮点型数据可以在同一个表达式中使用。C语言编译系统会按照一定的准则自动进行类型转换。

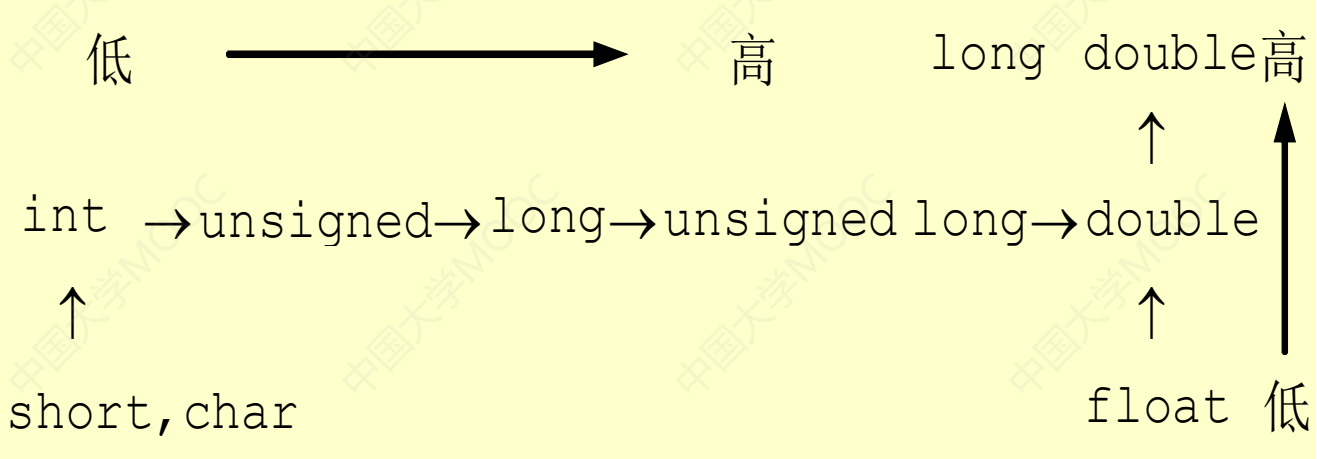
- 当出现下列三种情况时发生自动类型转换：
  - **当双目运算符的两个运算量结果的类型不相同且进行算术运算时；**
  - **当一个值赋予一个不同类型的变量时；**
  - **调用函数实现数据类型转换。**(后面章节介绍)

(本节中重点介绍前两种转换)

2.6.1.1 算术运算时的自动类型转换

- **转换规则：**值域较窄的类型向值域较宽的类型转换
  （“值域”就是类型所能表示的值的最大范围）

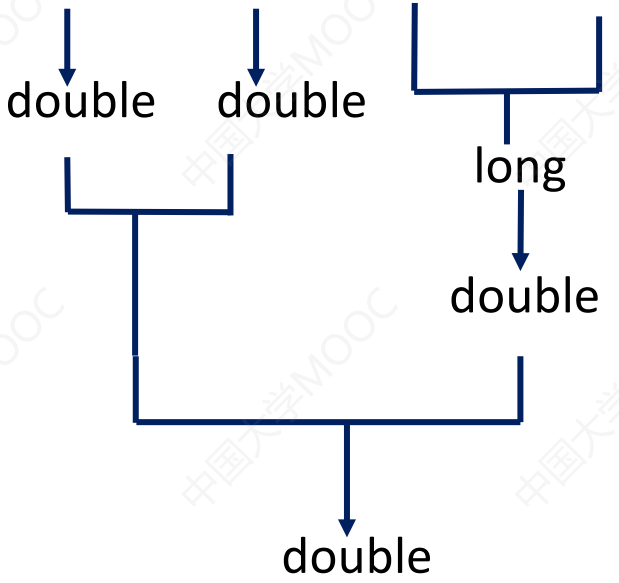
- 算术转换遵循的转换方向如图所示:



2.6.1.1 算术运算时的自动类型转换示例

例如：

```
float f = 3.6;
int n = 6;
long k = 21;
double ss = f   *   n   +   k   /   2;
```



2.6.1.2 赋值运算时的自动类型转换(隐式转换)

- 赋值转换将右值表达式结果的类型转成左值表达式的数据类型。赋值转换具有强制性，它不受算术转换规则的约束，转换结果的类型完全由左值表达式的类型决定。
- 示例：

```
int i, j;
float m;
```

则表达式 i = m * j 的转换过程为：

赋值运算符**右值表达式的值为double类型**，经过赋值转换强制变成**int类型**。

2.6.1.3 强制类型转换(显示转换)

- 一般形式为：

(类型名) 表达式

- 作用：将表达式转换成“类型名”所指定的类型。

- 示例：
```
float x = 6.5;
int  y  = (int)x;
```

后一语句将**单精度x强制转换为int类型，并赋值给y**，则y的值为6，但是变量x的类型仍是单精度浮点型，变量x仍为6.5 。

类型变换小结：

无论是自动类型转换还是强制类型转换，都只是将变量或常量的值的类型进行暂时的转换，用于参与运算和操作，而变量和常量本身的类型和数值并没有改变。

2.6.2 运算符的优先级

- 当在一个表达式中出现多个不同运算符的时候，运算符的**优先级**决定运算顺序，结合性决定了运算是**从左往右**还是**从右往左**。

- 示例： int x，y，z；
  z = y<= -x + 2 && !x；

表达式中各运算符的优先顺序为：

“!”=“-” （负号运算符）

>“+” （加法运算符）

>“<=”

>“&&”

>“=”。

因此上式等价于 z = (y <= (-x + 2)) &&(!x);

- 表2-8列出了C语言中运算符的**优先级**和**结合性**。（见课本P37）

2.7 输入输出的进一步讨论

2.7.1 格式化输出函数printf

- 函数调用形式：

printf(“输出格式”，**输出列表**);

- 常用的输出格式控制符

格式字符	说明
d	以带符号的十进制形式输出整数(整数不输出符号)
o	以八进制无符号形式输出整数
x,X	以十六进制无符号形式输出整数,用x则输出十六进制数的a-f时以小写形式输出.用X时,则以大写字母输出
u	以无符号十进制形式输出整数
c	以字符形式输出,只输出一个字符
s	输出字符串
f	以小数形式输出单双精度数,隐含输出6位小数
e,E	以指数形式输出实数,如用"E",则输出时,指数以大写"E"表示(如1.2E+02)

2.7.1 格式化输出函数printf

- 附加格式说明符表：

字符	说　明
字母l	用于长整型整数,可加在格式符d,o,x,u前面
m(代表一个正整数)	数据最小宽度
n(代表一个正整数)	对实数,表示输出n位小数;对字符串,表示截取的字符个数
-	输出的数字或字符在域内向左靠；无"-"时,在域内向右靠

2.7.1 格式化输出函数printf

（1）整数输出

1) %d，按十进制整数型数据的实际长度输出。

2) %md，m为指定的输出字段的最小宽度。如果数据的位数小于m，则左端补以空格，若大于m，则按实际位数输出。

示例: printf("%3d,%3d",x,y);

若x=21,y=12345，则输出结果为：【空格】21,12345

3) %ld，输出长整型数据。

示例：若定义long int x=12345678;则采用如下形式输出x正确：

printf("%ld",x);

则输出结果为：12345678

上例中如用“%d”输出，就会发生错误，因为整型数据的范围为-32768~32767。对于long型数据，应当用“%ld”格式输出。对于长整型数据，也可以指定字段宽度，例如将上例中的“%ld”改为“%10ld”

(2) 字符串输出

1) **%s**，输出指定的字符串

例如：printf("%s","CHINA");

输出“CHINA”字符串（不包括双引号）。

2) **%ms**，输出的字符串占m列，如字符串本身长度大于m，则突破m的限制，将字符串全部输出。若串长小于m，则左补空格。

3) **%-ms**，如果串长小于m，则在m列范围内，字符串向左靠，右补空格。

4) **%m.ns**，输出占m列，但只取字符串中左端n个字符。这n个字符输出在m列的右侧，左补空格。

5) **%-m.ns**，其中m、n含义同上，n个字符输出在m列范围的左侧，右补空格。如果n>m，则m自动取n值，即保证n个字符正常输出。

例2.9 字符串格式化输出例

```
#include<stdio.h>

void main( )
{
      printf("%3s,%7.2s,%.4s,%- 5.3s\n","HUST","HUST","HUST","HUST");
}
```

运行结果：

HUST, ␣ ␣ ␣ ␣ ␣ HU,HUST,HUS ␣ ␣

上例中其中第3个输出项，格式说明为“%.4s”，即只指定了n，没指定m，自动使m=n=4，故占4列。

(3) 浮点数输出

1) %f，不指定字段宽度，由系统自动指定，使整数部分全部如数输出，并输出6位小数。应当注意，并非全部数字都是有效数字。单精度实数的有效位数一般为7位。

2) %m.nf，指定输出的数据共占m列，其中有n位小数，输出数据右靠齐。如果数值长度小于m，则左端补空格。

3) %-m.nf，与%m.nf基本相同，只是使输出的数值向左端靠齐，右端补空格。

数据结果为：357.987000_357.987000_ _ _ _357.99_357.99_357.99_ _ _ _

```
#include <stdio.h>
void main( )
{
    float x=357.987;
    printf("%f  %10f%10.2f  %.2f  %-10.2f\n", x , x , x , x , x );
}
```

例2.10 浮点数格式化输出例

```
#include <stdio.h>
void main( )
{
    float x=357.987;

    printf("%f  %10f  %10.2f  %.2f  %-10.2f\n", x , x , x , x , x );
}
```

运行结果：

357.987000_357.987000_ _ _ _ _357.99_357.99_357.99_ _ _ _

上例中其中第4个输出项，格式说明为“%.2f”，即只指定了n，整数部分原样输出，按照四舍五入原则保留小数点后两位数字输出，输出共占6列。

2.7.2 格式输入函数scanf

- 函数调用形式：

scanf(“格式控制”，地址列表);

- 常用输入格式控制符

格式字符	说明
d	用来输入有符号的十进制整数
u	用来输入无符号的十进制整数
o	用来输入无符号的八进制整数
x, X	用来输入无符号的十六进制整数（大小写作用相同）
c	用来输入单个字符
s	用来输入字符串，在输入时以空白字符开始，以一个空白字符结束。
f	用来输入实数，可以用小数形式或指数形式输入

- 附加格式说明符

字符	说明
l	用于输入长整型数据（可用%ld,%lo,%lx）以及double型数据(可用%lf)
h	用于输入短整型数据（可用%hd,%ho,%hx)
域宽	指定输入数据所占宽度（列数），域宽应为正整数
*	表示本输入项在读入后不赋给相应变量

2.7.2 格式输入函数scanf

说明：

(1) 对**unsigned型变量**所需的数据，可以用**%u，%d或%o，%x**格式输入。

(2) 可以**指定输入数据所占有列数**，系统自动按它截取所需数据。

(3) 如果在%后有一个“*”附加说明符，表示**跳过指定的列数**。

(4) **输入数据时不能规定精度**

本章小结

- 准确地书写各种常量，正确定义(说明)各种类型的变量，理解不同数据类型的变量有不同的取值范围；
- 正确使用运算符来表示现实世界中的问题，正确书写各种表达式，对于较为复杂的表达式，建议使用圆括号来区分计算的优先顺序，这有助于提高程序的可读性；
- 掌握位运算的计算问题；
- 理解数据类型之间的强制数据类型转换问题；
- 理解运算符的优先级；
- 熟练掌握格式化输入输出库函数的使用。