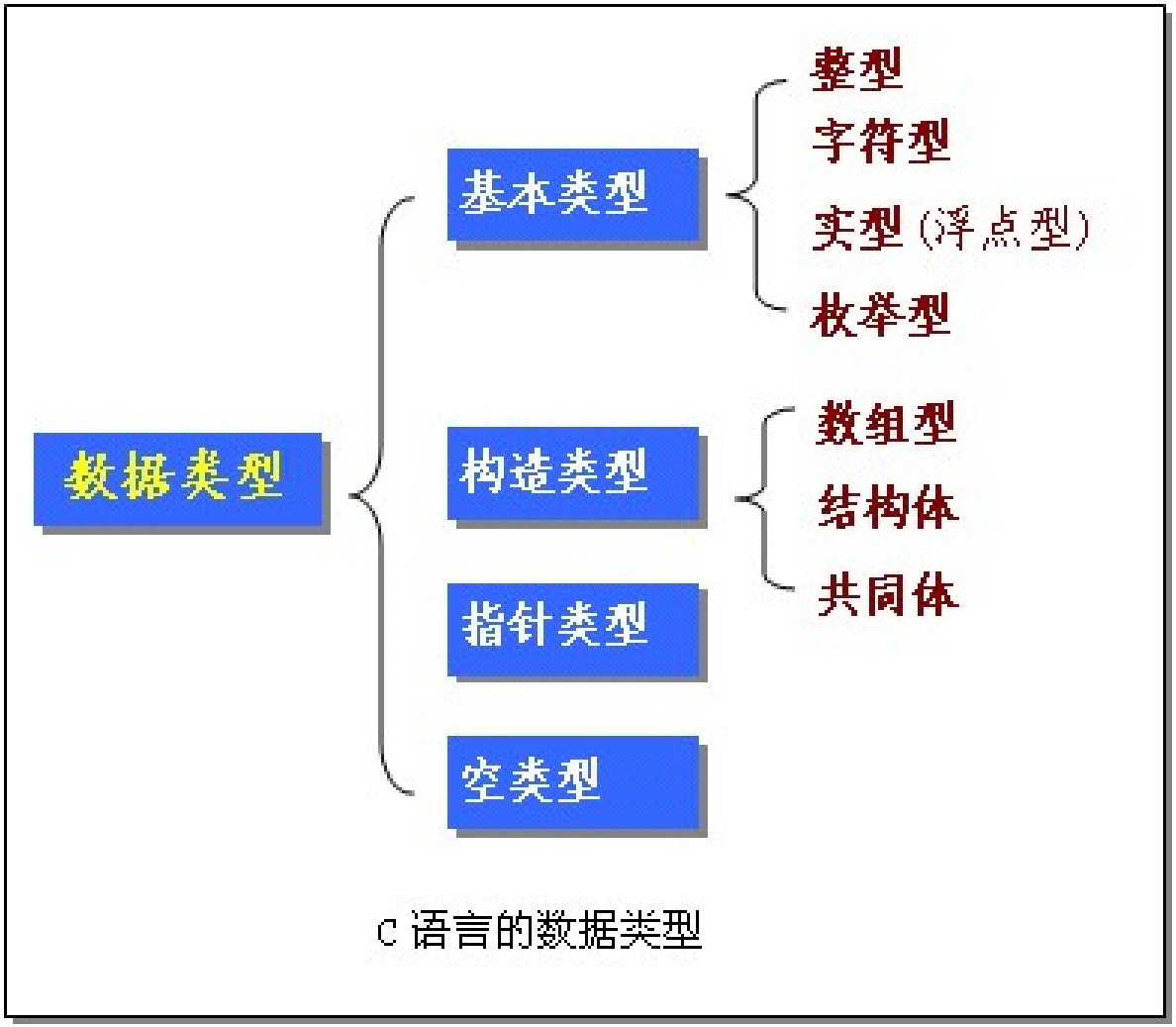
2.1 C 语言的数据类型



C 语言的数据类型

2.1 C 语言的数据类型

类型关键字	所占字节数	取值范围	说明
int	2	-32768~32767	有符号基本整型
unsigned	2	0~65535	无符号基本整型
short	2	-32768~32767	有符号短整型
unsigned short	2	0~65535	无符号短整型
long	4	-2147483648~2147483647	有符号长整型
unsigned long	4	0~4294967295	无符号长整型
float	4	10^{-37}~10^{38}	单精度实型
double	8	10^{-307}~10^{308}	双精度实型
char	1	0 ~255	字符型

2.2　常量与变量

程序运行过程中，其值不能被改变的量称为常量，其值可以改变的量称为变量。

2.2.1　常量

1. 整型常量

（1）十进制整数：由数字 1 ～ 9 开头，其余各位由 0 ～ 9 组成。如 123 、 -789 、 0 等。

（2）八进制整数：由数字 0 开头，其余各位由 0 ～ 7 组成。在书写时要加前缀 “ 0”（零）。如 012 ，表示八进制数 12 。

（3）十六进制整数：由数字 0x 或 0X 开头，其余各位由 0 ～ 9 与字母 a ～ f(0X 开头为 A ～ F) 组成。在书写时要加前缀 “ 0x” 或 “ 0X” 开头。如 0x36 ，代表十六进制数 36 。

2.2　常量与变量

注意： (1) 在 C 语言中 10， 010， 0x10 是 3 个数值完全不同的整数。

(2) 整型数可 long int 、 short int 和 unsigned int 等若干种。

(3) 长整型数在书写时加一个后缀“ L”。

(4) 整型数又可以是正数和负数。

例如： **123**， -123， 0123， -0123， 0x789， -0x789 是合法的；16 、 020 、 0x10 在计算机中的内部表示都相同。

2.2　常量与变量

2．实型常量

实型常量又称实数、浮点数，有两种表现形式：

（1）十进制实数：**由数字和小数点组成。如　0.149　，　123.0 。**

（2）指数形式：**用带指数记数法来表示，如 123E2 或 123e2**

（注意书写）

3．字符型常量

字符型常量有：字符常量、字符串常量和转义字符三种。

（1）字符常量

字符常量构成：用一对单引号括起来的单个字符。

例如：　'A' 、 'a' 、 'X' 、 '?' 、 '$'　等都是字符常量。

字符常量的值：就是该字符的 ASCII 码值，可以和数值一样参加运算。

例如：字符 'A' 的数值为十进制数 65 。

2.2　常量与变量

3．字符型常量

（2）字符串常量

字符串常量构成：用一对双引号括起来的字符序列。

例如： **"abc"**、"CHINA"、"yes"、"1234"、"How do you do." 等，都是字符串常量。

字符串长度：字符中的字符个数。

例如，"How do you do ."、"Good morning ." 和 "" 的长度分别为 14、13 和 0。

C 语言规定：字符串存储时，系统在末尾自动加一个 '\0' 符号作为字符串的结束标志。

例如：字符串为“ CHINA” 内存中实际存储为：

C	H	I	N	A	\0	……

2.2　常量与变量

3．字符型常量

注意：字符常量与字符串常量的区别。

例如：字符常量'A'与字符串常量"A"的区别是：

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**：字符常量长度为1，字符串常量长度可为任意值（0或某个整数）；

③ **存储要求不同**：字符常量存储的是字符的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'。

2.2　常量与变量

3．字符型常量

（3）转义字符

转义字符： C 语言中用来表示键盘上的控制符和功能符的特殊符号。

例如：回车换行符、换页符等。

形式：反斜杠 "\" 后面跟一个字符或一个数值。

例如： '\n'，为换行， '\101' 与 '\x41' 都表示字符 'A' 。

2.2　常量与变量

3．字符型常量

转义字符	表示含义
\\	将 \ 转义为字符常量中的有效字符（\ 字符）
\'	单引号字符
\"	双引号字符
\n	换行，将当前位置移到下一行开头
\t	横向跳格，横向跳到下一个输出区
\r	回车，将当前位置移到本行开头
\f	走纸换页，将当前位置移到下页开头
\b	退格，将当前位置移到前一列
\v	竖向跳格
\ddd	1 到 3 位 8 进制数所代表的字符
\xhh	1 到 2 位 16 进制数所代表的字符

2.2　常量与变量

【例 2.1 】　转义字符的使用

```
#include <stdio.h>
void main()
{  printf(" ␣ ab ␣ c\t ␣ de\rf\tg\n");   /* ␣ 表示一个空格
   */
   printf("h\ti\b\bj ␣ k");
 }
```

程序运行后在显示屏上的输出结果如下：

```
f ␣␣␣␣␣␣␣ gde
h ␣␣␣␣␣␣ j ␣ k
```

若在打印机上输出，则结果如下：

```
f ab ␣ c ␣␣␣ gde
h ␣␣␣␣␣␣ jik
```

2.2　常量与变量

（4）符号常量

在 C 语言中用符号来代替常量称为符号常量。

习惯上，符号常量名用大写，变量名用小写，以示区别。

【例 2.2】　符号常量的使用。

```
/* 程序功能：计算圆的面积 */
#include <stdio.h>
#define  PI  3.1415926
void  main()
  {  float r,s ;
     r = 5.0;
     s = PI * r*r;
     printf("Area is %f", s);
  }
```

运行结果为：　Area is 78.539815

2.2　常量与变量

注意：使用符号常量的好处是：

（1）含义清楚。定义符号常量名时应考虑“见名知意”。

（2）在需要改变一个常量时能做到“一改全改”。

（3）符号常量的值在其作用域内不能改变，也不能再被赋值。

2.2　常量与变量

2.2.2　变量

变量：在程序运行过程中其值可以改变的量。

变量属性：变量名、变量类型（内存中占用存储单元）和变量的值。

（1）变量名：变量的名字——变量名。

变量命名：遵循标识符命名规则，最好取名时考虑“见名知意”。

例如： max 表示求最大值、 sum 表示求和等。

（2）变量类型：整型变量、实型变量和字符变量三种。不同类型的变量，占用的内存单元（字节）数不同。

（3）变量值：变量在程序运行过程中的取值。在程序中，通过变量名来引用变量的值。

2.2　常量与变量

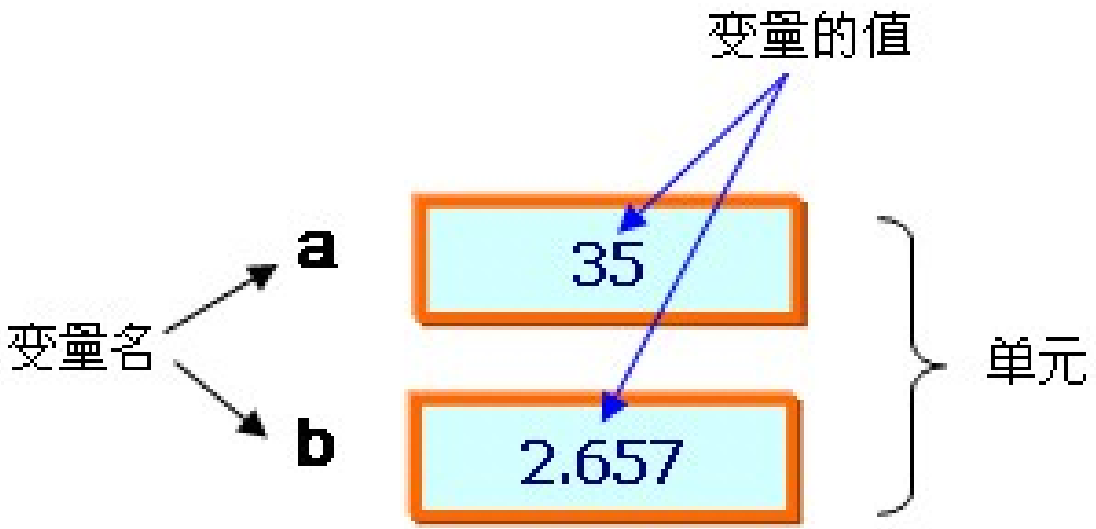


图 2.1　变量的属性

2.2　常量与变量

1．整型变量

（1）整型变量分类

根据占用内存字节数的不同，整型变量分为 4 类：

基本整型（int）、**短整型**（short [int]）、**长整型**（long [int]）和 **无符号整型。**

无符号整型：无符号基本整型（unsigned [int]）、无符号短整型（unsigned short 表示）和无符号长整型（用 unsigned long 表示）三种。

2.2　常量与变量

（2）整型变量占用字节数与值域

各种类型的整型变量占用的内存字节数，随系统而异。

类型关键字	所占字节数	取值范围	说明
int	2	-32768~32767	有符号基本整型
unsigned	2	0~65535	无符号基本整型
short	2	-32768~32767	有符号短整型
unsigned short	2	0~65535	无符号短整型
long	4	-2147483648~2147483647	有符号长整型
unsigned long	4	0~4294967295	无符号长整型

注：对于其它系统中设基本整型占用内存字节数为 n，
其值域为：-2^{n*8-1}～（$2^{n*8-1-1}$）；无符号整型变量的值域为：0～（$2^{n*8}-1$）。

2.2　常量与变量

例如，图 2.2 是在 Turbo C 系统中： int 与 unsigned 最大值。

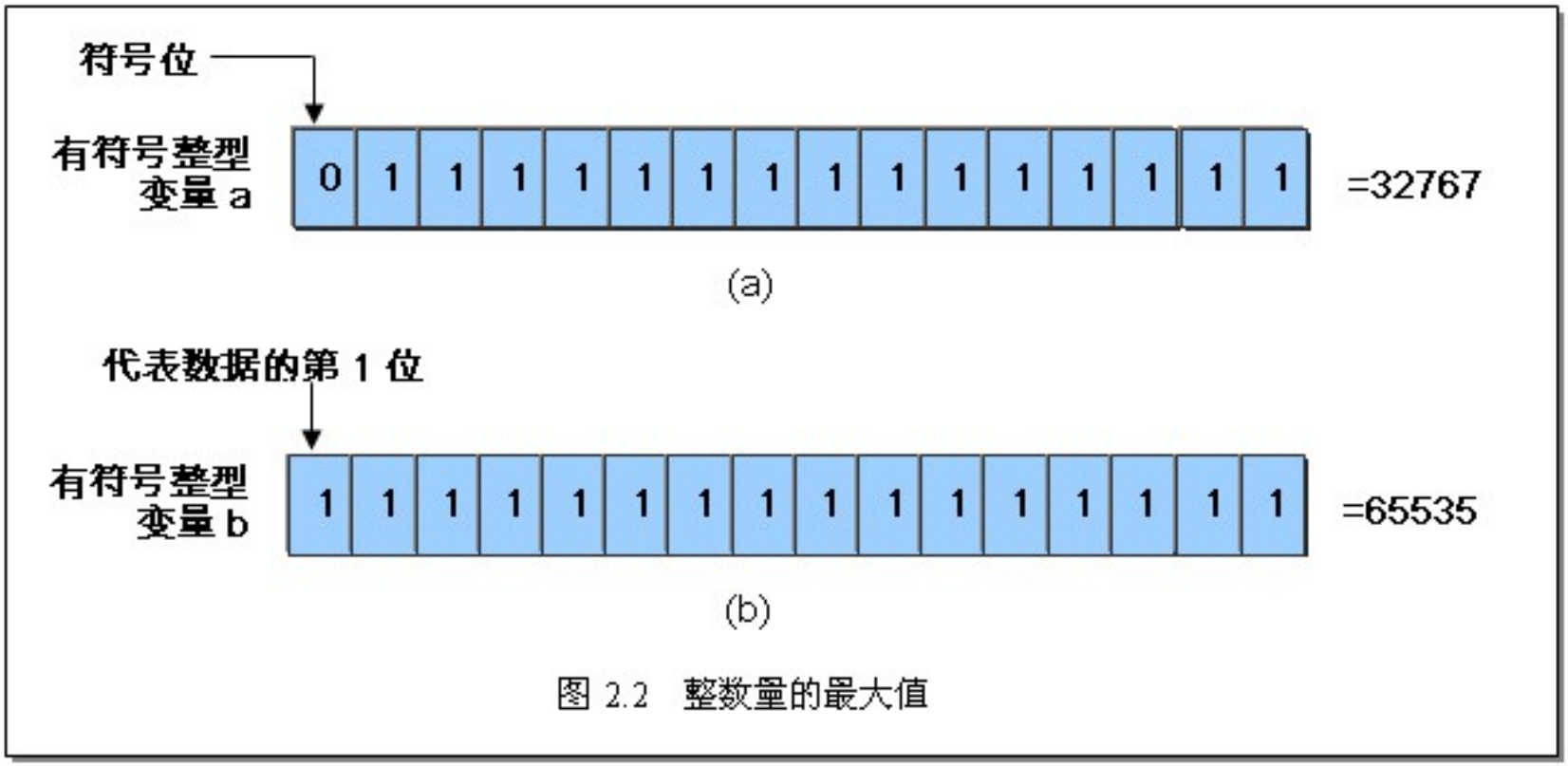


图 2.2　整数量的最大值

2.2　常量与变量

(3) 整型变量的定义

所有变量都必须 “先定义，后使用”。

例如：

```
int a,b;                    /* 定义变量 a、b 为整型 */
unsigned short c,d;         /* 定义变量 c、d 为无符号短整型 */
long e,f;                   /* 定义变量 e、f 为长整型 */
```

注：变量定义一般是放在一个函数的开头的声明部分。

2.2　常量与变量

【例 2.3 】 整型变量的定义与使用。

```
#include <stdio.h>
void main()
  { int a,b,c,d ;                  /* 定义 a, b, c, d 为整型变量 */
    unsigned u ;                   /* 定义 u 为无符号整型变量 */
    a = 12 ;   b = -24 ;   u = 10 ;
    c = a + u ;   d = b + u ;
    printf("a + u = %d ,  b + u = %d\n",c,d) ;
  }
```

运行结果为:

```
a + u = 22 ,  b + u = -14
```

2.2　常量与变量

2．实型变量

C 语言的实型变量，分为两种：

（1）单精度型（**float**）：**占 4 个字节（32 位）7 位有效数字。**

（2）双精度型（**double**），**占 8 个字节 15 位有效数字。**

例如：

```
float x,y;          /* 定义 x 、 y 为单精度实数 */
double b;           /* 定义 b 为双精度实数 */
long double c;      /* 定义 c 为长双精度实数 */
```

2.2　常量与变量

3．字符变量

字符变量的类型关键字为 **char** ，占用一个字节内存单元。

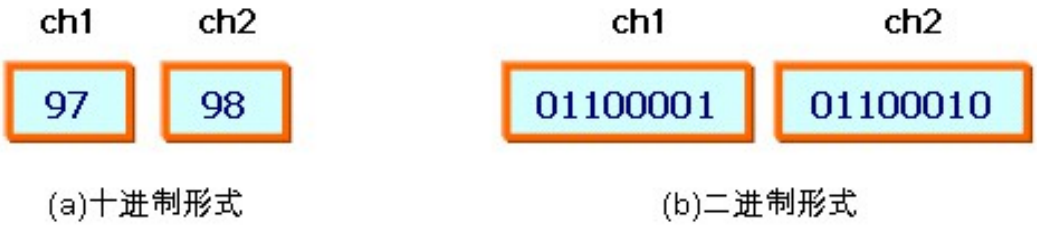
（1）变量值的存储

字符变量在内存中占一个字节。以该字符的 **ASCII** 码值（无符号整数）存储到内存单元中。

例如：

```
char ch1 , ch2 ;          /* 定义两个字符变量： ch1 , ch2*/
ch1 = 'a' ; ch2 = 'b';    /* 给字符变量赋值 */
```

存储如图所示：



2.2　常量与变量

（ 2 ）特性

C 语言中字符型数据与整型数据之间可以通用。

例如：

① 字符型数据既可以以“字符形式”或者“整数形式”输出。

【例 2.4 】字符变量的字符形式输出和整数形式输出。

```
#include <stdio.h>
void  main()
  { char ch1,ch2 ;
     ch = 'a' ;   ch2 = 'b' ;
      printf("ch1=%c,ch2=%c\n",ch1,ch2) ;        /*  字符形式输出
*/
     printf(“ch1= %d,ch2 = %d\n”,ch1,ch2) ;          /*  整数形式输出
*/
  }
```

程序运行结果：

```
ch1= a,ch2 = b
ch1= 97,ch2 = 98
```

注意： 字符数据占一个字节，它只能存放 0-255 范围内的整数。

2.2　常量与变量

② 允许对字符数据进行算术运算。

【例 2.5 】　字符数据的算术运算。

```
#include <stdio.h>
 void  main()
{  char ch1,ch2 ;
    ch1 = 'a' ;   ch2 = 'B' ;
    printf("ch1=%c,ch2=%c\n" ,  ch1-32,ch2+32) ;
                                        /* 字母的大小写转换  */
}
```

程序的运行结果：

ch1= A,ch2 = b

2.3　变量赋初值

变量赋初值又称为变量初始化，有其两种方法：

（1）先定义一个变量，然后再给它赋一个值，例如：

```
int  a;
a=  8  ;
```

（2）在定义变量的同时就对变量进行初始化，例如：

```
char  ch = 'a' ;
float b =2.345 ;
int   x, y = 3 ;          /* 部分变量赋初值，对 y 赋初值 3 */
```

2.3　变量赋初值

【例 2.6 】变量赋初值

```
#include "stdio.h"
void  main()
  { int x,y = 3 ;
    char  ch ;
    ch = 'a' ;
    printf("%d  %d  %c",x,y,ch) ;
  }
```

运行结果如下：

```
251  3  a
```

注：　x 没有进行初始化，输出为随机不定值 251 ，下次可能是其他的结果。

2.4　各类数值型数据间的混合运算

在同一个表达式中出现多种数据类型的混合运算时，先将各种类型数据转换成同一类型数据，然后才能运算求值。转换规则如图 2.4 所示。

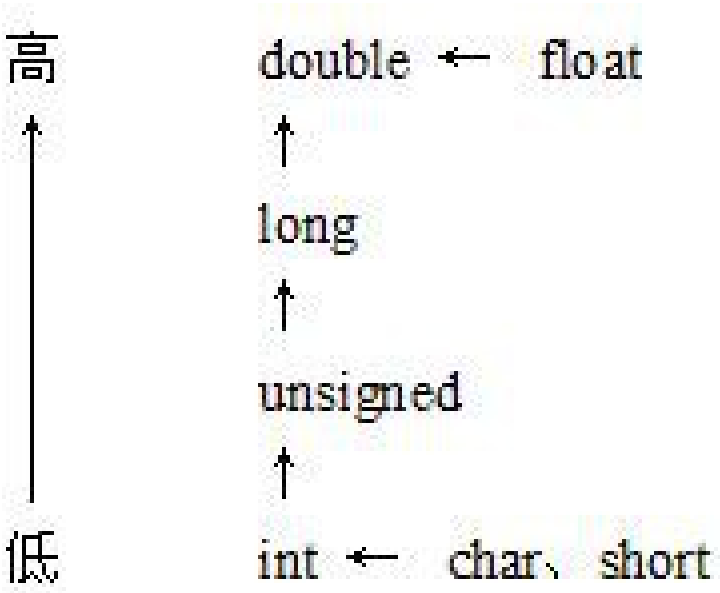


图2.4 转换规则

2.4　各类数值型数据间的混合运算

（1）规范化

图 **2.4** 中水平方向的向左箭头的转换为规范化转换。

（2）保值转换

图 **2.4** 中向上方向的向上箭头的转换为保值转换（即将类型等级较低转换成等级较高的）。

例如：有如下定义：

```
int   m ;
float   n ;
double   d ;
longint   e ;
```

对表达式：　**('c' + 'd') * 20 + m * n - d ／ e**

问：上述表达式最后的结果是什么类型？

2.4　各类数值型数据间的混合运算

求 (‘c’ + ‘d’) * 20 + m * n - d / e 的转换过程是：

① 计算 (‘c’ + ‘d’) 时，先将 ‘ c’ 和 ‘ d’ 转换成整型数 99 、 100 ，运算结果为 199 ；

② 计算 m * n 时，先将 m 和 n 都转换成为双精度型；

③ 将 e 转换成双精度型， d / e 结果为双精度型；

④ 假设所用计算机是先计算运算符左边操作数，那么 ('c' + 'd') * 20 计算后结果为 3980 ，再将 3980 转换成双精度型，然后与 m * n 的结果相加，再减去 d / e 的结果，表达式计算完毕，结果为双精度。

2.5　C 语言的运算符和表达式

C 语言表达式是由运算符、常量及变量组成；
运算符（即操作符）是对运算对象（又称操作数）进行某种操作的符号。有二元（双目）、一元（单目）运算符。
对于运算符应从以下不几个方面掌握：

1．运算符号
2．运算规则，即所进行的操作
3．运算的优先级别
4．运算顺序
5．运算对象
6．运算结果

2.5　C 语言的运算符和表达式

1．五种基本算术运算符

+ （加法）、 - （减法 / 取负）、 * （乘法）、 / （除法）、 % （求余数）

运算规则与代数运算基本相同，但有以下不同之处需要说明：

（1）除法运算 /

两个整数相除商为整数，小数部分被舍弃。

例如： **5 / 2 = 2**　　而： **5.0/2 = 2.5** 。

（2）求余数运算 %

参加运算的两操作数均为整型数据，否则出错。结果是整除后的余数。在 Turbo C 中运算运算结果的符号与被除数相同。

例如： **7%3=1** ， **7%-3=1** ，（商分别为 **2** 、 **-2** ）。

-7%3=-1 ， **-7%-3=-1** ；（商分别为 **-2** 、 **2** ）。

2.5　C 语言的运算符和表达式

2．表达式和算术表达式

(1) 表达式的概念。

表达式：**用运算符和括号将常量、变量和函数等连接起来符合 C 语言语法规则的式子。**

单个常量、变量或函数构成的表达式称为简单表达式。

(2) 算术表达式的概念。

算术表达式：**用算术运算符和括号将常量、变量和函数等连接起来的符合 C 语言语法规则的式子**。

例如：**3 + 6 * 9**、**(x + y) / 2 – 1**、**5-a** 等，都是算术表达式。

2.5　C 语言的运算符和表达式

(3) 表达式求值

① 运算顺序

例如：先乘除后加减。

② 结合性

同级运算的结合方向称为结合性。

例如：**算术运算符的结合方向是“从左至右”，即先左后右，称左结合。**

2.5　C 语言的运算符和表达式

3．强制类型转换

一般格式为：

(< 要转换成的数据类型名 >)　(< 被转换的表达式 >)

功能：**将一个表达式强制转换成所需类型**。

例如：

(double)　a　　**将**变量 *a* **的**值转换**成** *double* **型**

(int)(x + y)　　**将** *x+y* **的**结**果**转换**成** *int* **型**

(float) 5 / 2　　**等价于** *((float)5)/2* **, 将** *5* 转换**成**实**型，再除以** *2 (=2.5)*

(float)(5 / 2)　　**将** *5* **整除** *2* **的**结**果** *(2)* 转换**成**实**型** *(2.0)*

注意：强制转换类型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，x 原定为 float 型，则 (double) x 只是将变量 x 的值转换成一个 double 型的中间量，其 x 的数据类型并未转换成 double 型，仍为 float 型。

2.5　C 语言的运算符和表达式

4．自增（++）、自减（--）运算符

自增（++）运算是使单个变量的值增 1；

自减（--）运算使单个变量的值减 1。

自增、自减运算符都有两种用法：

(1) 前置运算：

运算符放在变量之前：++ 变量、-- 变量

先使变量的值增 (或减)1 然后再以变化后的值参与其他运算，即先增（减）、后运算。

(2) 后置运算：

运算符放在变量之后：变量 ++ 、变量 --

变量先参与其他运算，然后再使变量的值增（或减）1，即先运算、后增（减）。

2.5　C 语言的运算符和表达式

例如：如果 i 的原值等于 3，则执行下面的赋值语句：

① j = ++ i;　　（*i* 的值先增 *1* 变成 *4*，再赋给 *j*，*j* 的值为 *4*）

② j = i ++ ;　　（先将 *i* 的值 *3* 赋给 *j*，*j* 的值为 *3*，然后 *i* 增 *1* 变成 *4*）

【例 2.7】自增、自减运算符的用法与运算规则示例

```
#include <stdio.h>
void  main()
 {  int x = 6, y ;
    printf("x = %d\n
    y = ++x ;                                    ) , 然后再赋值给 y(=
7)*/
    printf("y = ++x :
     y = x --  ;                    /* 后置运算：先将 x 的值 (=7) 赋值给 y(=7) ,
然后 x 再减 1(=6)*/
    printf("y = x-- :   x = %d,y = %d\n", x , y) ;
 }
```

程序运行结果：

```
x = 6
y = ++x :   x = 7, y = 7
y = x-- :   x = 6 , y = 7
```

2.5　C 语言的运算符和表达式

说明：

（1）自增、自减运算常用于循环语句中使循环控制变量加（或减）1。

（2）自增、自减运算符不能用于常量和表达式。

例如，5++ 、--(a+b) 等都是非法的。

（3）连续使用同一变量进行自增或自减时很易出错。

例如：(x++) + (x++) + (x++)= ？（假设 x 的初值 =3）

解：表达式的值等于 9，变量 x 的值变为 6。为什么？请思考。

（4）书写时最好采用能理解的写法，避免误解。

如：不要写成 i+++j 形式，可产生 (i++)+j 或 i+(++j) 二义性，最好写成 (i++)+j 或 i+(++j) 的形式。

但 C 语言规定：从左到右取尽可能多的符号组成运算符。i+++j 应理解为 (i++)+j。

（5）在 printf() 函数中从右向左计算。

例如：设 i 的初值为 5：

```
Printf("%d,%d",i,i++);
```

输出结果为：

6，5

2.5　C 语言的运算符和表达式

2.5.2　关系运算符和关系表达式

关系运算就是将两个数据进行“比较运算”，判定两个数据是否符合给定的关系，如果条件成立结果为“真”；否则条件不成立结果为“假”。

1．关系运算符及其优先次序

（1）关系运算符

< (小于)，　<= (小于或等于)，　> (大于)，
>= (大于或等于)，　== (等于)，　!= (不等于)

注意：“ 等于”关系运算符是双等号“ ==”，**而不是单等号“ =”**(赋值运算符)。

2.5　C 语言的运算符和表达式

（2）关系运算符优先级

前 4 个优先级相同，后 2 个相同，且前 4 个高于后 2 个。

即：（ < 、 <= 、 > 、 >= ）→（ == 、 != ）

（3）与其他运算的优先级

关系运算符的优先级低于算术运算符，但高于赋值运算符。

即：算术运算符 →（ < ， <= ， > ， >= ）→ (== ， != ）→ 赋值运算符

2．关系表达式

关系表达式：用一个关系运算符将两个表达式（可以是算术表达式、关系表达式、逻辑表达式、赋值表达式或字符表达式等）连接起来，进行关系运算的式子。

2.5　C 语言的运算符和表达式

例如：　a>b，　a + b > c - d，　(a = 3) <= (b = 5)，‘ a’ >= ‘b’，

(a > b) == (b > c)　**都是合法的关系表达式。**

关系表达式的值是一个逻辑值（非“真”即“假”）。C 语言中用整数“ 1” 表示“逻辑真”，用整数“ 0” 表示“逻辑假”。

例如，假设 **int x = 3，　y = 4，　z = 5**，则：

（ 1 ） **x > y** 的值为 **0** 。

（ 2 ） **(x > y) != z** 的值为 **1** 。

（ 3 ） **x < y < z** 的值为 **1** 。

（ 4 ） **(x < y) + z** 的值为 **6** 。

再次强调：**关系表达式的值，还可以参与其他种类的运算。**

2.5　C 语言的运算符和表达式

2.5.3　逻辑运算符和逻辑表达式

关系表达式只能描述单一条件，例如“ x >= 0”。如果需要描述“ x >= 0”且“ x < l0”，就不能能写为“ 0<=x<10”，必须借助于逻辑表达式了。

1．　逻辑运算符及其优先次序

（1）逻辑运算符及其运算规则

逻辑运算符：

&&	逻辑与（相当于“并且”）
||	逻辑或（相当于“或者”）
!	逻辑非（相当于“否定”）

其中，“ **&&**”和“ **||**”是双目运算符；而“ **!**”是单目运算符。

2.5　C 语言的运算符和表达式

例如，**下面的表达式都是逻辑表达式：**

```
(x >= 0 ) && (x < l0)   /* x>=0，并且 x < 10 */
( x < 1 ) || ( x > 5 )   /* x<l，或者 x>5 */
!(x==0)                  /* 否定 x = 0，即：x 不等于 0 时，条件成立 */
(year % 4== 0 ) && (year % 100 ! = 0 ) || (year % 400==0 )
                         /*year 能被 4 整除，同时不能被 100 整除；或者，year 能被 400 整除*/
```

逻辑运算符的运算规则：

① **&&**：当且仅当两个运算量为“真”结果为“真”，否则为“假”。

② **||**：当且仅当两个运算量为“假”结果为“假”，否则为“真”。

③ **！**：当运算量为“真”时结果为“假”；当运算量为“假”时结果为“真”。

2.5　C 语言的运算符和表达式

例如，假定 x = 5，则

(x >= 0) && (x < l0) 的值为“真”，

(x <- 1) || (x > 5) 的值为“假”。

（ 2 ）逻辑运算符的运算优先级

① 逻辑运算符优先级

逻辑非的优先级最高，逻辑与次之，逻辑或最低。

即：**！（非）→ &&（与）→ ||（或）**

② 与其他种类运算符的优先关系

！（非）→算术运算→关系运算→ **&&（与）→ ||（或）**

→ 赋值运算

2.5　C 语言的运算符和表达式

2．逻辑表达式

逻辑表达式：**用逻辑运算符将 1 个或多个表达式连接起来，进行逻辑运算的式子。**

例如：“ (year % 4 == 0) && (year % 100 != 0) || (year % 400 == 0)”

就是一个判断年份是否是闰年的逻辑表达式。

逻辑表达式的值：**是一个逻辑值 (非“真”即“假”) 。整数“ 1” 表示“逻辑真”、用“ 0” 表示“逻辑假”。**

例如：假设 num=12，则：

(1)　!num 的值＝ 0 。

(2)　num>=1 && num<=31 的值＝ 1 。

(3)　num || nun > 31 的值 ＝1 。

2.5　C 语言的运算符和表达式

3．说明

（1）逻辑运算符两侧的操作数可以是任何类型的数据，如实型、字符型等。

（2）在计算逻辑表达式时，不一定所有的表达式都要求解。

① 对于与运算，只要一个操作数为“假”，结果为“假”。不需求解第二个操作数。

② 对于或运算，只要一个操作数为“真”，结果为“真”。不需求解第二个操作数。

例如：假设 **int m、n、a、b、c、d；**

m = n = a = b = c = d = 1；

表达式：**(m = a > b) && (n = c > d)** 结果为 **0**：

计算时，因 (m = a > b) 的值为 0，不需计算表达式 (n = c > d)。因 (n = c > d) 是“真”还是“假”，其值都为“假”。

练习

- 判断某个整数 i 是否为偶数

  (i%2==0)

- 要判断某一年 year 是否为闰年，要满足以下两个条件之一： 1 ）能被 4 整除，但不能被 100 整除； 2 ）能被 4 整除，又能被 400 整除。

  (y%4==0 && y%100!=0) || y%400==0

- 判断三边 a,b,c 是否能构成三角形

  (a+b>c && b+c>a && c+a>b)

2.5　C 语言的运算符和表达式

2.5.4　赋值运算符和赋值表达式

1. 赋值运算

符号“=”就是赋值运算符，它的作用是将一个表达式的值赋给一个变量。

赋值运算符的一般形式为：

＜变量＞＝＜赋值表达式＞

注意：被赋值的变量必须是单个变量，且必须在赋值运算符的左边。

例如：　x = 5　　/* 将 5 赋值给变量 x */

　　y = (float)5/2　　/* 将表达式的值（=2.5）赋值给变量 y */

当表达式值的类型与变量类型不一致，系统自动将表达式的值转换成被赋值变量的数据类型后再赋值给变量。

2.5　C 语言的运算符和表达式

2．复合赋值运算

在赋值符“ =” 之前加上其他双目运算符可构成复合赋值符。

复合赋值运算的一般格式为：

＜变量＞＜双目运算符＞＝＜表达式＞

它等价于：

＜变量＞ ＝＜变量＞ 双目运算符 (表达式)

例如：　x + = 3　　/* 等价于 $x = x + 3$ */

y *= x + 6　　/* 等价于 $y = y * (x + 6)$，而不是 $y = y * x + 6$ */

C 语言中有 10 种复合赋值运算符：

+= ，　-= ，　*= ，　/=, %= ；　　/* 复合算术运算符 (5 个) */

&= ，　^= ，　|= ，　<<= ，　>>= ；　　/* 复合位运算符 (5 个) */

2.5　C 语言的运算符和表达式

3．赋值表达式

赋值表达式：**由赋值运算符或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。**

一般格式：

< 变量 >[< 复合赋值运算符 >]= < 表达式 >

赋值表达式也有一个值，被赋值变量的值，就是赋值表达式的值。

例如：“ a = 5” 为赋值表达式，　a 的值“ 5” 就是**表达式的值**。

2.5　C 语言的运算符和表达式

2.5.5　条件运算符和条件表达式

1．一般格式：

<表达式 1> ？ <表达式 2> : <表达式 3>

C 语言中唯一的一个三目运算符，三个表达式类型可以各不相同。但通常“表达式 1”为逻辑表达式或关系表达式。

2．运算规则

计算“表达式 1”的值为非 0（“真”），则结果等于“表达式 2”的值；否则，结果等于“表达式 3”的值。

2.5　C 语言的运算符和表达式

3．运算符的优先级别与结合性

条件运算符的优先级，高于赋值运算符，但低于关系运算符和算术运算符。其结合性为“从右到左”（即右结合性）。

例如：设 a = 2，　c = 'a',　　f = 3.0 则下列表达式

a > 0 ? a : -a	结果为　2
f == 3.0 ? a <= c : a >= c	结果为　1
(f > 0) ? ((a > 0) ? 2 : 1) : ((a>0) ? 1 : 0)	结果为　2
(a >= 0) ? (a = 1) : (a = 0)	结果为　1

2.5　C 语言的运算符和表达式

【例 2.8 】从键盘上输入一个字符，如果它是大写字母，则把它转换成小写字母输出；否则，直接输出。

```
void  main()
{   char ch;
    printf("Input a character:");
    scanf("%c",&ch);
    ch = (ch >= 'A' && ch <= 'Z') ? (ch+32) : ch;
    printf("ch = %c\n",ch);
}
```

2.5　C 语言的运算符和表达式

2.5.6　逗号运算符和逗号表达式

用逗号运算符“，”连接起来的式子称为逗号表达式。逗号运算符又称顺序求值运算符。

1．一般形式：

< 表达式 1>,< 表达式 2>,……,< 表达式 n> 。

2．求解过程：

从左至右，依次计算各表达式的值，最后“表达式 n”的值即为整个逗号表达式的值。

例如：　a = 3 * 5，　a * 4 的值 =60

先求解 a=3*5 得 a=15 ；再求　a*4=60 ，所以逗号表达式的值 = 60 。

又例如：　(a = 3 * 5，　a*4)，　a+5 的值 =20 ：

2.5　C 语言的运算符和表达式

2.5.7　求字节数运算

求字节数运算运算符：　**sizeof(类型 / 变量)**

功能：　**sizeof 运算符求变量或类型的字节长度**。

例如：　**sizeof (double)** 为 **8**

sizeof (int)　为 **2**

```
float f;
int i;
```

i = sizeof(f) 的值为 **4** 。

2.5　C 语言的运算符和表达式

2.5.8　位逻辑运算

操作符	名称	运算规则	主要用途
&	按位与	对应位均为 1 时才为 1，否则为 0	屏蔽数的某些位，其余位保留不变
|	按位或	对应位均为 0 时才为 0，否则为 1	将一个数的某（些）位置 1，其余位保留不变
^	按位异或	对应位相同时为 0，不同时为 1	使一个数的某（些）位翻转，其余各位不变
~	按位取反	各位求反，即 1 变 0，0 变 1	

2.5　C 语言的运算符和表达式

说明：

（1）所有操作数都必须首先转换成二进制数再按位运算。

（2）位运算符优先级别：

~ → & → | → ^

（3）位双目运算符优先级别低于关系运算符，高于逻辑运算符，运算自左向右运算。

关系运算符→（& → |）→逻辑运算符

（4）~运算符自右向左运算；&、|是自左向右运算。

2.5　C 语言的运算符和表达式

例如：设有定义： int a=3 ， b=9 ， 则有：

a & b = 1	a | b = 11	a ^ b = 10	~b=246
00000011	00000011	00000011	
& 00001001	| 00001001	^ 00001001	~ 00001001
00000001	00001011	00001010	11110110

2.5　C 语言的运算符和表达式

2.5.9　位移运算和位运算赋值运算符

1．位移运算

操作符	名称	运算规则	说明
<<	左移	a << b，a 左移 b 位	使操作数各位左移，低位补 0，高位移出舍去。
>>	右移	a >> b，a 右移 b 位	使操作数各位右移，移出的低位舍去，高位： 1.对无符号数和有符号中的正数，补 0 （2）有符号数中的负数，取决于所使用的系统：补 0 的称为“逻辑右移”，补 1 的称为“算术右移”

2.5　C 语言的运算符和表达式

优先级别：（<<，>>）仅次于算术双目运算符，运算方向从左向右运算。

（1）左移运算符

左移运算符把一个数的二进位全部左移若干位。

例如：　b << 3　将 *b* 的各位数字左移 *3* 位，右补 *0* 。若 *b* = *13*，即二进制数 *00001101*，左移 *3* 位得 *01101000*，　即十进制数 *104* 。

二进制数左移 1 位相当于该数乘以 2^1 ，左移 2 位相当于该数乘 4 。……

（2）右移位运算符

例如：b >> 2　将 *b* 的各二进位右移 *2* 位。移到右端的低位被舍弃，对无符号数，高位补 *0* 。 如 *b* = *14* 时：

b 为 *00001110*，　*b*>>*2* 为 *00000011*：*10*此二位舍弃

2.5　C 语言的运算符和表达式

右移一位相当于除以 2 ，右移 n 位相当于除以 2^n 。

右移时，应注意符号位问题。

补入 0 的称为“逻辑右移”。补入 1 的称 “算术右移”。

例如： m 为八进制数 113755 。

m ： 1001011111101101

m>>1 ： 0100101111110110 （逻辑右移时）

m>>1 ： 1100101111110110 （算术右移时）

2.5　C 语言的运算符和表达式

2．位运算赋值运算符

操作符	名称	运算规则	说明
&= |= ^= <<= >>=	位与赋值 位或赋值 按位异或赋值 位左移赋值 位右移赋值	a&=b，等价于 a=a&b a|=b，等价于 a=a|b a^=b，等价于 a=a^b a<<=b，等价于 a=a<<b a>>=b，等价于 a=a>>b	操作数均为整型

位运算赋值运算符优先级：6 个运算符同级，**与赋值符级别相同，运算方向自右向左运算。**

例如：设 **a =6**，**b = 3**，**a**，**b** 均为无符号整型，则：

b &= a，结果 **a** 不变，**b** 为 **2**

a <<= b，结果 **b** 不变，**a** 为 **48**

2.5　C 语言的运算符和表达式

2.5.10．运算符的优先级与结合性

结合性**就是指操作数两侧的运算符相同优先级时，该操作数是先与左边，还是先与右边的运算符结合。若先与左面的运算符结合，称为左结合性（即从左至右运算）。反之，称为右结合性（从右至左运算）。**

总结得出：

结合性：除单目运算符、条件运算符和赋值运算符是右结合性外，其他运算符都是左结合。

优先级顺序：表中第 1 级最高，……，第 15 级“,”号运算符最低。