第3章 数据类型、运算符与表达式

3.1 C的数据类型

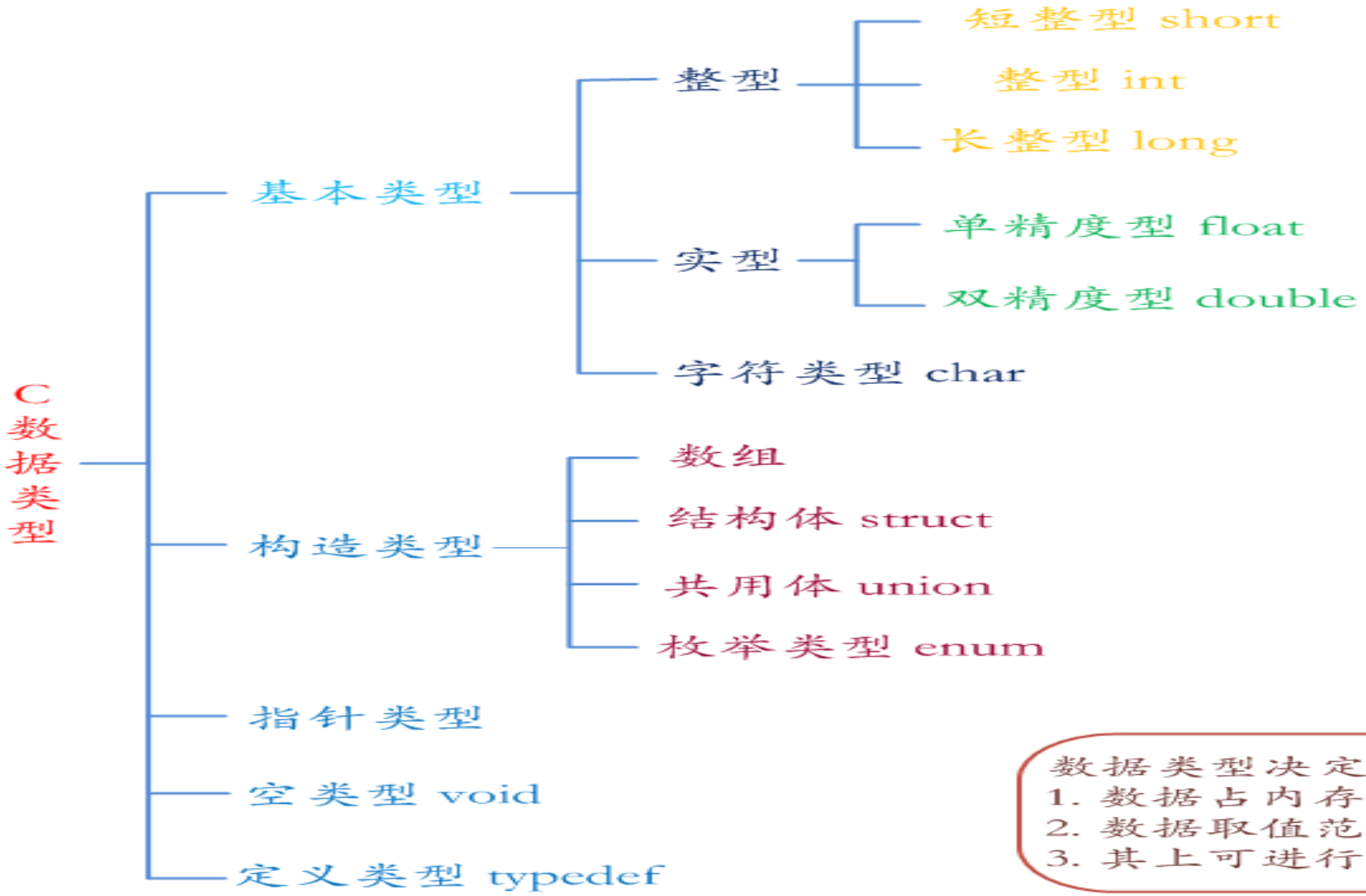
一个程序应包括以下两方面内容：

(1) 对数据的描述。在程序中要指定数据的类型和数据的组织形式，即数据结构(data structure)。

(2) 对操作的描述。即操作步骤，也就是算法(algorithm)。

数据是操作的对象，操作的目的是对数据进行加工处理，以得到期望的结果。打个比方，厨师做菜肴，需要有菜谱。

3.1 C的数据类型



数据类型决定：
1. 数据占内存字节数
2. 数据取值范围
3. 其上可进行的操作

表 3-2 数据类型及取值范围

类型定义标识符	Microsoft Visual C++6.0		Turbo C 2.0	
	长度（bit）	数值表达范围	长度（bit）	数值表达范围
[signed] int	32 位	-2147483648~2147483647	16 位	－32768～32767
[signed] short [int]	16 位	－32768～32767	16 位	－32768～32767
[signed] long [int]	32 位	-2147483648~2147483647	32 位	-2147483648~2147483647
unsigned [int]	32 位	0~4294967295(即 2^{32}-1)	16 位	0~65535
unsigned short [int]	16 位	0~65535	16 位	0~65535
unsigned long [int]	32 位	0~4294967295(即 2^{32}-1)	32 位	0~4 294 967 295
float	32 位	3.4E－38～3.4E＋38	32 位	3.4E－38～3.4E＋38
double	64 位	1.7E－308～1.7E＋308	64 位	1.7E－308～1.7E＋308
[signed] char	8 位	－127～127	8 位	－127～127
unsigned char	8 位	0~255	8 位	0~255
void	0 位	无值	0 位	无值

★标识符

- ❖定义：标识变量名、符号常量名、函数名、数组名、文件名的字符串序列——名字。
- ❖命名规则：
  - ●只能由字母、数字、下划线组成，且第一个字符必须是字母或下划线
  - ●大小写字母含义不同，一般用小写
  - ●不能使用关键字
  - ●TC允许最长32个字符，建议长度不超过8个字符
- ❖使用：先定义、后使用

这些标识符合法吗？
1A、M.D.John、￥123、#33、Tatol、int、max

标识符应该“见名知意”，如 total ， max
标识符应该“不宜混淆”，如 1与1 ， 0与0

C语言关键字列表

- 定义：也称为保留字，在C语言中具有特定含义，专门用作C语言特定成分的标示符，关键字不能作为 用户标示符。
- int if ✕ int int ✕
-

void	char	short	int	long	float	double
signed	unsigned	struct	union	enum	typedef	auto
static	extern	const	register	return	if	else
switch	case	default	do	while	for	break
continue	goto	sizeof	volatile	inline	restrict	_Bool
_Complex	_Imaginary					

课堂练习

- 1. 在C语言中，下列标示符中合法的是（A）
- A _int
- B 3in1-3
- C A_B!D
- D void

课堂练习

2. 下列选项中，均是合法标示符的选项（C）

A _a　　void　　zhangsan

B 5.2　　5.2　　include

C _888　　fun　　_INT

D -12　　const　　2*a

3.2 常量与变量

3.2.1 常量和符号常量:

- 定义：程序运行过程中，其值不能被改变的量（常数）
- 分类：直接常量、符号常量

类型	示例
整型常量	**12 、 0 、 -3**
实型常量	**4.6 、 -1.23**
字符常量	**‘a’、 ‘b’**
符号常量	**PRICE 、 PI**

例1

```
#define  price  30
#include <stdio.h>
int main ( )
{
 int num;
 float total;
 num=100;
 total=num * price/1000;
 printf("total=%d thousands",total);//如果此处用%f呢？
 return 0;
}
```

符号常量

- 一般用大写字母： PRICE 、 PI
- 定义格式： #define 符号常量 常量
- 其值在作用域内不能改变和再赋值。

```
例2 符号常量举例
#define PRICE 30
#include <stdio.h>
int main()
{
    int num,total;
    num=10;
    total=num*PRICE;
    printf("total=%d\n",total);
    return 0;
}
```

运行结果：

```
total=300
```

注 符号常量的优点是：见名知意、一改全改

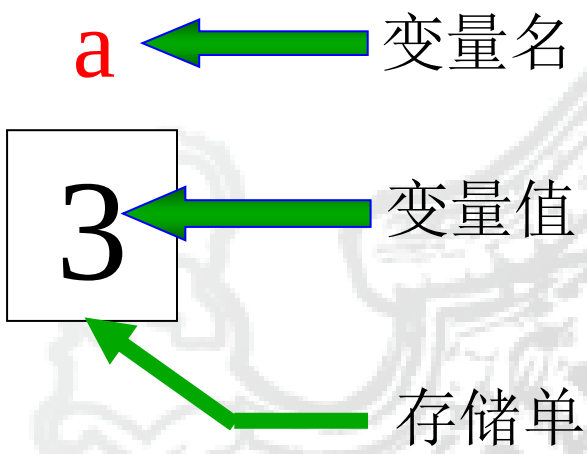
例3 符号常量的使用

```
#define PI 3.14159  /*定义符号常量PI，其值为3.14159*/
#include <stdio.h>
int main( )
{
   /*按照小数格式输出2*PI*1的值*/
  printf ("半径是1的圆的周长是：%f\n", 2*PI*1);
  printf ("半径是1的圆的面积是：%f\n", PI*1*1);
  /*按照小数格式输出PI*1*1的值*/
  return 0;
}
```

3.2.2 变量:

- 定义：其值可以改变的量。
- 定义格式：数据类型　变量名；
- 变量应该有名字，并在内存中占据一定的存储单元。
- 变量名和变量值有不同的含义
  - 变量名实为一个符号地址

```
例4　变量的使用
int main()
{
    int a;
    a=3;
    printf("a=%d",a);
    return 0;
}
```



内存：字节和位

- 内存以字节为单元组成
- 每个字节有一个地址
- 一个字节一般由8个二进制位组成
- 每个二进位的值是0或1

- *世界上只有10种人，一种懂二进制，一种不懂二进制。*

内存：字节和位

十进制数字	0	1	8	10	20	100	-1	-8
二进制数字	0	1	1000	1010	10100	1100100	11111111111111111111111111111111	11111111111111111111111111111000
八进制数字	0	1	10	12	24	144	6114230461	6014030061
十六进制数字	0	1	8	a	14	64	31313131	30303031

十六进制数字的10-15依次表示为：a b c d e f

内存：字节和位

```
#include<stdio.h>
//itoa函数不是标准的C语言库函数，可能在某些系统下未得到实现
//此例子仅供参考学习使用，不要求理解和掌握
int main()
{
        int a = 0;//1 8 10 20 100 -1 -8
        char s2[25],s8[25],s16[25];
        itoa(a,s2,2);
        printf("%s\n",s2);
        itoa(a,s8,8);
        printf("%s\n",s8);
        itoa(a,s16,16);
        printf("%s\n",s16);
}
```

3.3 整型数据

3.3.1 整型常量（也称整数）

可用以下三种形式表示：

(1) 十进制整数：由数字0~9和正负号表示。

如 123, -456, 0

(2) 八进制整数：由数字0开头, 后跟数字0~7表示。

如 0123, -011

例如：0123表示八进制数123，即(123)$_8$，

其值为：$1\times8^2+2\times8^1+3\times8^0$，等于十进制数83。

-011表示八进制数-11，即十进制数-9。

(3) 十六进制整数：由0x开头, 后跟0~9, a~f, A~F表示。

如 0x123, 0xff

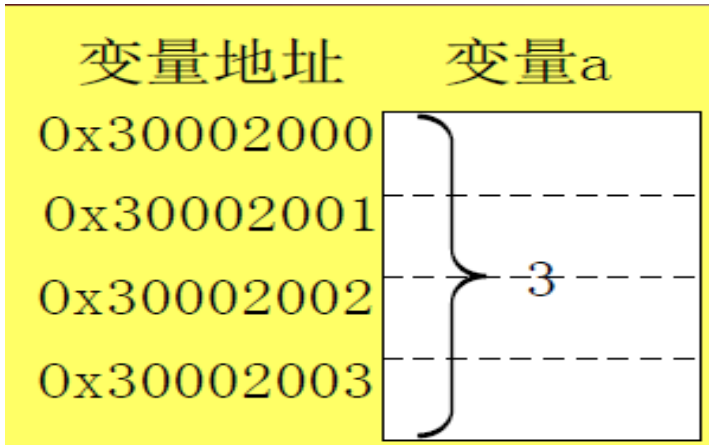
例如： 0x123，代表十六进制数123，

其值为：(123)16=$1\times16^2+2\times16^1+3\times16^0$=256+32+3=291。

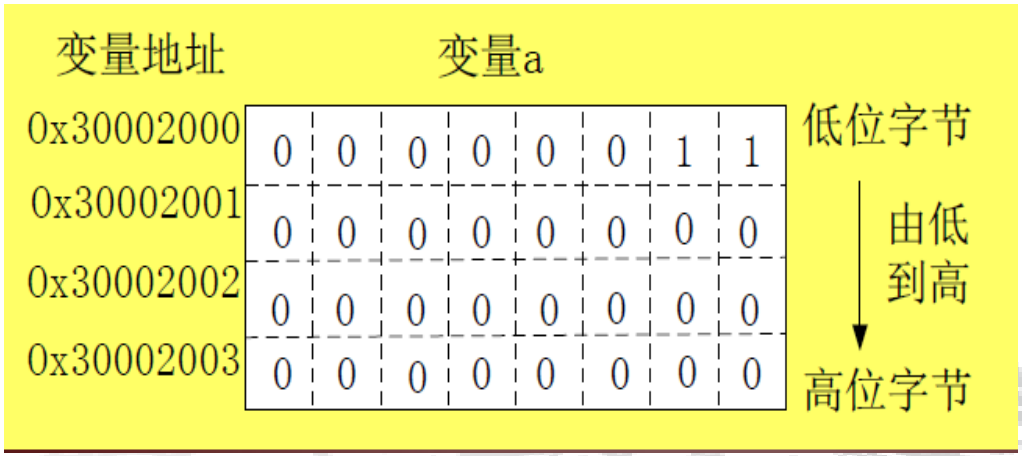
-0x12等于十进制数-18。

3.3.2 整型变量的存放形式

例如：int a=3; /* 定义a为整型变量 */
a=3; /* 给a赋以整数3 */



图a：数据存放的示意图



图b：数据在内存中实际存放的情况

整型变量正负数的转化参考 附录的3.3.2及3.3.3
方法：首位0变1，其他位取反，再加1

3.3.3 整型变量的取值范围(共4个字节即32位)

int 类型的二进制取值范围

符号位： 0代表正数，1代表负数

0 1 1 1 1 1 1 1, 1 1 1 1 1 1 1 1, 1 1 1 1 1 1 1 1,1 1 1 1 1 1 1 1

2^{31} -1 即：2147483647(最大值)

0 1 1 1 1 1 1 1, 1 1 1 1 1 1 1 1, 1 1 1 1 1 1 1 1,1 1 1 1 1 1 1 0

2^{31}-2 即：2147483646

0 1 1 1 1 1 1 1, 1 1 1 1 1 1 1 1, 1 1 1 1 1 1 1 1,1 1 1 1 1 1 0 1

2^{31}-3 即：2147483645

...

0 0 0 0 0 0 0 0, 0 0 0 0 0 0 0 0, 0 0 0 0 0 0 0 0,0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 , 1 1 1 1 1 1 1 1, 1 1 1 1 1 1 1 1,1 1 1 1 1 1 1 1 -1

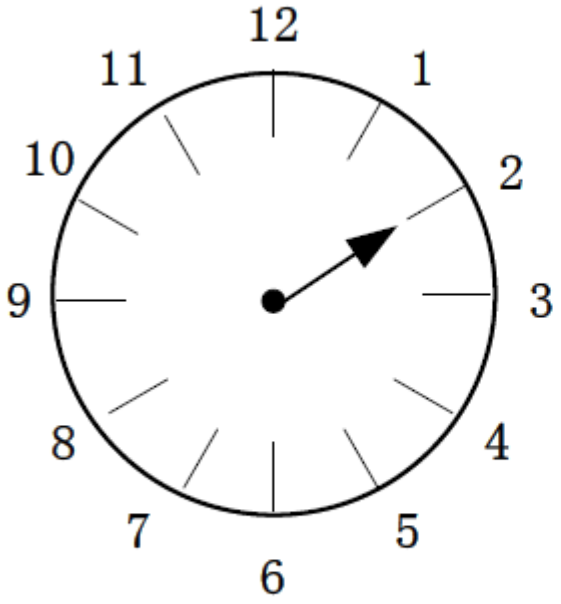
1 1 1 1 1 1 1 1 , 1 1 1 1 1 1 1 1, 1 1 1 1 1 1 1 1,1 1 1 1 1 1 1 0 -2

...

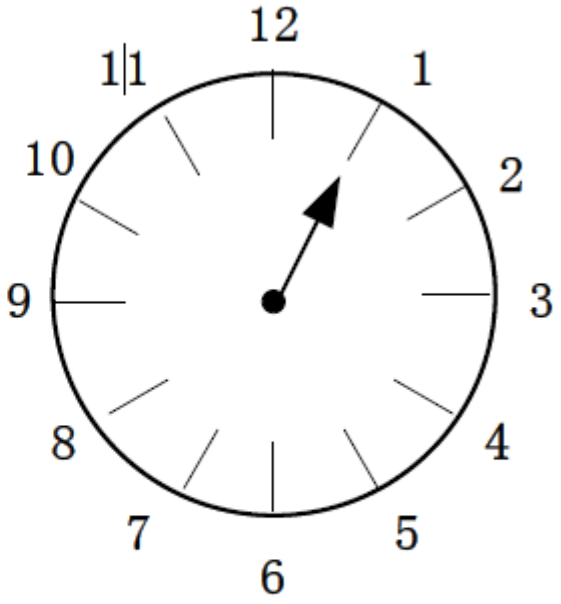
1 0 0 0 0 0 0 0, 0 0 0 0 0 0 0 0, 0 0 0 0 0 0 0 0,0 0 0 0 0 0 0 0

-2^{31} 即：-2147483648(最小值)

整型变量的取值范围

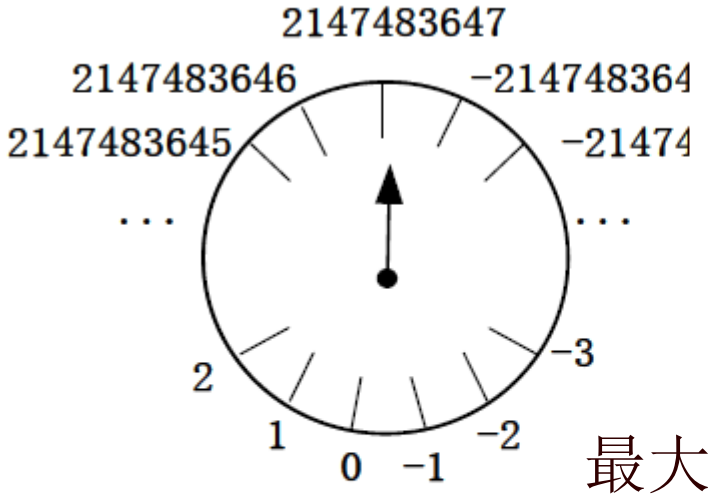


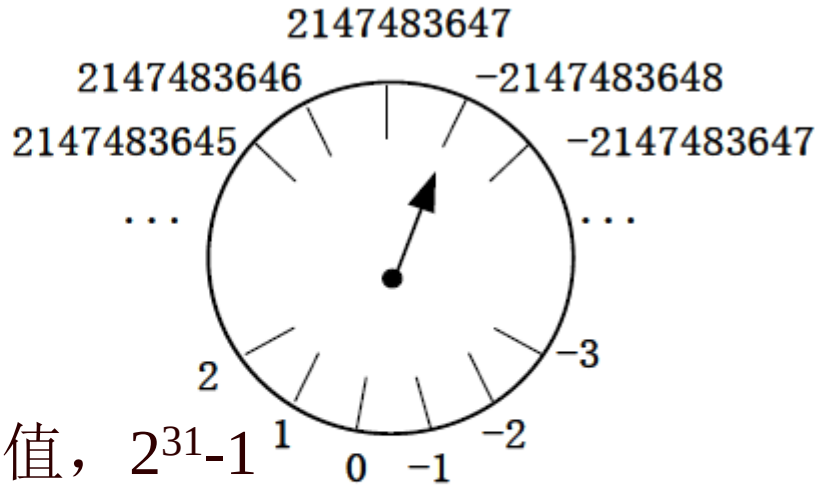
1点＋1点＝2点



12点＋1点＝1点

- 整型变量：4个字节
- 取值范围-2147483648 - 2147483647





最大值，$2^{31}-1$

2147483646+1=

2147483647

2147483647+1=

-2147483648

最小值，-2^{31}

整型变量的定义

- 原则1：要先定义，后使用 ——强制类型定义

一般定义格式：

数据类型　　变量名1 [,变量名2,...,变量名n];

决定分配字节数

存储变量名必须为合法标识符

（1）int num;

num=100;

（2）int num =100;

（3）int a，b，c=5;

表示指定a、b、c为整型变量，只对c初始化，c的值为5。

变量赋值

原则2：先定义后赋值。

格式：

变量 = 常量(也可以是变量、表达式等) ;

功能：将赋值运算符右端的值复制一份传递给左端的变量。要求赋值符号左端必须是变量。

例5 定义一个变量

```
#include <stdio.h>
int main( )
{
int a; /* 定义变量a */
printf ("%d\n",a); /* 按照整型的格式输出a的值 */
return 0;
}
```

思考：

1. 分析输出结果
2. 可否去掉 int a; ?

例6 变量赋值

```
#include <stdio.h>
int main( )
{
int a; /* 定义变量a */
a = 3; /* 将3赋值给a */
printf ("%d",a); /* 按照整型的格式输出a的值 */
return 0;
}
```

思考：在printf语句前面加上a=5; 结果如何？

例7 定义两个变量

```
#include <stdio.h>
int main( )
{
int a, b;                          /* 定义变量a */
scanf ("%d,%d",&a, &b);            /* 从键盘输入值给变量a和b中*/
printf ("%d, %d\n",a,b);           /* 按照整型的格式分别输出a和b的值 */
return 0;
}
```

变量初始化

定义：在定义变量的同时给它赋值，称为变量初始化。

格式：

数据类型 变量 = 常量(也可以是变量、表达式等)；

等价于：

数据类型 变量;

变量 = 常量(也可以是变量、表达式等)；

例8 变量初始化

```
#include <stdio.h>
int main( )
{
int a = 3; /* 定义变量a，同时将a初始化为3，与第6个例
   子略有不同*/
printf ("%d",a); /* 按照整型的格式输出a的值 */
return 0;
}
```

思考：在printf语句前面加上int a=5; 结果如何？

例9 变量初始化和赋值

```
#include <stdio.h>
int main( )
{
int a, b = 2;    /* 定义变量a和b，将b初始化为2*/
int c;           /* 定义变量c */
a = 8; /* 将1赋值给变量a和c，如果此行注释掉会如何？ */
c = a+b;
printf ("%d, %d, %d ",a,b,c); /* 按照整型的格式分别输出a、
   b和c的值 */
return 0;
}
```

思考：若将变量a和b都初始化为2，应如何写？

例10 变量初始化和赋值

```
#include <stdio.h>
int main( )
{
int a, b = 2;    /* 定义变量a和b，将b初始化为2*/
int c;           /* 定义变量c */
  scanf ("%d,%d",&a, &b); /* 从键盘输入值给变量a和b 中*/
  c = a+b;
printf ("a=%d,b=%d,c=%d ",a, b, c); /* 按照整型的格式分别输出a、b和c
   的值 */
return 0;
}
```

思考：若将变量a和b都初始化为3，应如何输入？

总结C语言程序的基本结构

```
/* 注释 */
#include <stdio.h> /* 输入输出函数在此声明 */
#include <xxxxx.h> /* 其他函数声明所在 */
int main( )
{
变量定义语句;   /* 例如 int a, b, c;*/
  输入语句;        /* 例如 a = 2; b = 1;  或 scanf ("%d",&a);
  */
运算处理语句;  /* 例如 c = a+b; */
输出语句;        /* 例如 printf ("c = %d/n", c); */
return 0;
}
```

注意：变量定义语句要放在所在函数中其他语句的前面

❖两个字节的整型数据的溢出

整型变量最大值32767	01	11	11	11	11	11	11	11
加1后是 –32768的补码形式	10	00	00	00	00	00	00	00

- 此情况称为“溢出”，运行时不报错，编程时要注意

```
例11  整型数据的溢出
#include <stdio.h>
int main( )
{
  int a , b;
  a= 32767;
  b= a+1;
  printf("%d , %d \n ",a,b);
  return 0;
 }
```

改为：long b;
结果是什么？

运行结果：

```
32767 , -32768
```

练习1：判断程序是否有错，若有，找出错误的原因并修改，若无，给出输出结果。

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

练习2：判断程序是否有错，若有，找出错误的原因并修改，若无，给出输出结果。

```
#include <stdio.h>
int main( )
{
int a;
int b;
a = b = 10;
printf("a,b\n",a,b);
return 0;
}
```

练习3：判断程序是否有错，若有，找出错误的原因并修改，若无，给出输出结果。

```
#include <stdio.h>
int main( )
{
int a;
int b;
a = b = 10;
 printf(" a=%d,b=%d\n ",a,b);
return 0;
}
```

练习4：判断程序是否有错，若有，找出错误的原因并修改，若无，给出输出结果。

```
#include <stdio.h>
int main( )
{
  int a=10,b;
  b = a;
    printf(" a=%d,b=%d\n ",a,b);
  return 0;
}
```

练习5：判断程序是否有错，若有，找出错误的原因并修改，若无，给出输出结果。

```
#include <stdio.h>
int main( )
{
int a,b;
a = b = 10;
printf("a=%d,b=%d\n",a,b);
return 0;
}
```

练习6：判断程序是否有错，若有，找出错误的原因并修改，若无，给出输出结果。

```
#include <stdio.h>
int main( )
{
  int a;
  a = 10;
  int b = a;
    printf(" a=%d,b=%d\n ",a,b);
  return 0;
}
```

课堂练习：修改以下程序

```
#include <stdio.h>
int main( )
{
  a = 3,b = 5;
  c= a*b
  printf("The product:",c);
  return 0;
}
```

3.4 实型数据

3.4.1 实型常量的表示方法

- 实数(real number)又称浮点数(floating-point number)。
- 实数有两种表示形式：
  - (1)十进制小数形式:由数字和小数点组成(注意必须有小数点)。

    例如：.123、123.、123.0、0.0都是十进制小数形式。

  (2) 指数形式:字母e(或e)之前必须有数字，且e后面的指数必须为整数。

  例如：123e3代表123×10^3

3.4 实型数据

```
#include <stdio.h>
int main( )
{
  float a;
  a = 1e-1;// 10e-1 8e-2
  float b = a*10;
  printf(" a=%f,b=%f\n ",a,b);
  return 0;
}
```

不合法的指数形式的实数：

345　　(无小数点)

e7　　(阶码标志E之前无数字)

- 53-e3　　(负号位置不对)
- 2.7e　　(无阶码)

2.1e3.5　　(阶码不是整数)

注意：

（1）规范化的指数形式：即在字母e(或e)之前的小数部分中，小数点左边应有一位(且只能有一位)零的数字。

例如：123.456：123.456e0、12.3456e1、1.23456e2、0.123456e3、0.0123456e4、0.00123456e5

（2）一个实数在用指数形式输出时，是按规范化的指数形式输出的。

3）实型常量默认是双精度（double），有效位数是15~16位。

4）如要指定它为单精度（float），可以加后缀“f”或“F”，如 3.6f，有效位数为6~7位。

3.4.2 实型变量

1. 实型数据在内存中的存放形 式

- 实型数据在内存中占4个字节（32位）
- 在内存中分成3部分（数符、小数部分和指数部分），指数为2的幂次。指数部分采用规范化的指数形式。

实数 3.14159 在内存中的存放形式可以用图3.7示意。

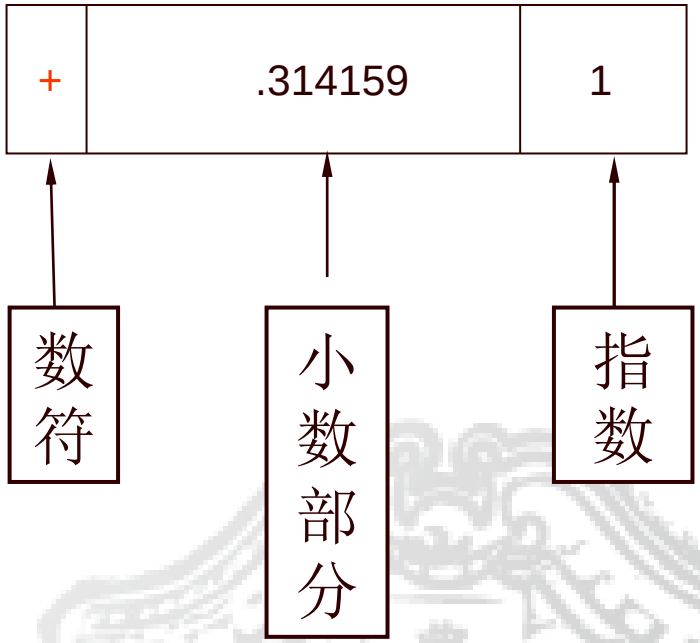


图3.7

2. 实型变量的分类

类　型	位数	有效数字	取值范围
单精度 float	32位	6~7	-3.4×10^{-38}~3.4×10^{38}
双精度 double 型	64位	15~16	-1.7×10^{-308}~1.7×10^{308}

3. 实型变量的定义

- float x,y;　　（指定x、y为单精度浮点型变量）
- double z;　　（指定z为双精度浮点型变量）

❖浮点型数据的舍入误差

- 数据超过有效位数，则产生误差
- 要避免一个很大的数与一个很小的数加减

例12 浮点型数据的舍入误差

```
#include <stdio.h>
int main( )
{
  float a , b;
  a= 123456.789e5;
  b= a+20;
  printf("%f \n%f\n",a,b);
  return 0;
 }
```

注意：程序运行时，输出b的值与a相等。
原因是：一个实型变量只能保证的有效数字是7位有效数字，后面的数字无意义

运行结果：

```
12345678848.000000
12345678848.000000
```

```
#include<stdio.h>

int main()
{
        //float x = 1999998.0f/1999999.0f;
        //float y = 1.0f;
        float x = 19999998.0f/19999999.0f;
        float y = 1.0f;
        printf("%f %f\n",x,y);
        if(x-y==0)
        {               printf("yes\n");}
        else
        {               printf("no\n"); }
}
```

观察两种情况下的显示结果

注意，浮点数的“等号”判断

- 例13 整型/实型数据的输入输出。

```
int main()
{
    float a, b;
    int c, d;
    double e,f;
    scanf("%f,%f", &a, &b);      /* 按照实型的格式输入两个值，分别存入变量a，b中 */
    printf("%.2f,%.3f" , a, b);      /* 按照实型的格式输出变量a，b的值，分别保留2和3位小数 */
    scanf("%d %d", &c, &d);      /* 按照整型的格式输入两个值，分别存入变量c，d中 */
    printf("%d %d" , c, d );     /* 按照整型的格式输出变量c，d的值 */
    scanf("%lf,%lf", &e, &f);      /* 按照实型的格式输入两个值，分别存入变量e，f中 */
    printf("%.2lf,%.3lf" , e, f);    /* 按照实型的格式输出变量e，f的值，分别保留2和3位小数 */
    return 0;
}
```

3.5 字符型数据

3.5.1 字符常量

- 定义:用单引号括起来的单个字符或转义字符

如 'a' 'A' '\n' '\t'

- 字符常量的值：该字符的ASCII码值

如 'a'——97 'A'——65 '\n'——10 '\t'——9

- 定义格式：char 变量名 = 值

char ch=65 与 char ch= 'A' 与char='\101'是等效的

- 转义字符:反斜线（“\”）后面跟一个字符或一个代码值表示。这是一种“控制字符”，不能在屏幕上 显示出来。

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

转义字符	转义字符的作用	ASCII 码
\n	回车换行，将光标位置移到下一行开头	10
\r	回车不换行，将光标位置移到本行开头	13
\t	水平制表(Tab)，将光标位置移到下一个制表位置	9
\b	退格，将光标位置移到前一列	8
\f	换页，将光标位置移到下一页开头	12
\a	响铃	7
\0	空操作符，字符串结束标志	0
\’	单引号字符	34
\”	双引号字符	39
\\	一个反斜杠字符	92
\ddd	1到3位八进制数所代表的字符	
\xhh	1到2位十六进制数所代表的字符	

- 例14 转义字符的使用
- #include <stdio.h>
- int main( )
- {
- /* 输出包含转义字符在内的一系列字符*/
- printf ("\"hello\rworld \\\tHELLO\bWORLD\"\n");
- return 0;
- }


图 2-7(a)


图 2-7(b)


图 2-7(c)

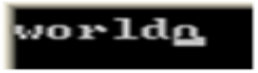


图 2-7(d)

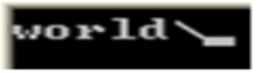


图 2-7(e)


图 2-7(f)


图 2-7(g)


图 2-7(h)


图 2-7(i)


图 2-7(j)


图 2-7(k)

例15　转义字符的使用

```
#include <stdio.h>
int main( )
{  printf("  ab  c\t de\rf\tg\n");
   printf("h\ti\b\bj k");
   return 0;
}
```

显示结果：

```
f       gde
h       j k
```

3.5.2 字符变量

- 存放字符常量，占用一个字节，存放一个字符
- 定义形式：　　赋值：

char c1,c2;　　c1='a';c2='b';

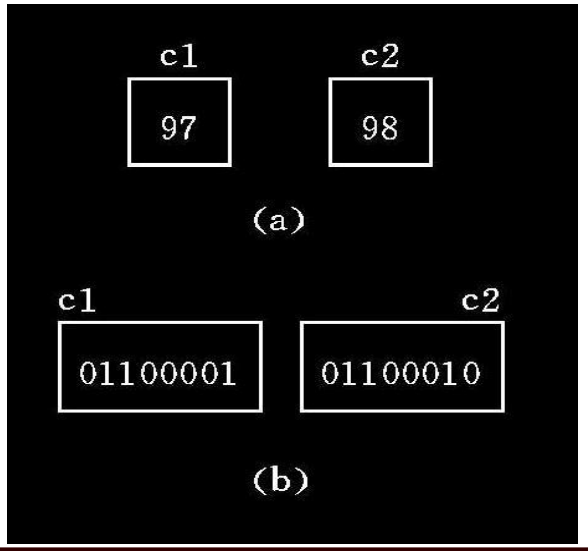
★ 字符数据在内存中的存储形式及其使用方法

- 以二进制存放字符的ASCII码值（0~255整数）
- 与整数的存储形式类似

❖ 以字符或整数形式输出

例如：字符 ‘a’的ASCII代码为97，
‘b’为98，

- 在内存中变量c1、c2的值如图所示。实际上是以二进制形式存放的。



输出形式取决于printf函数中的格式符

格式符为 “%c”时输出的变量值为字符
格式符为 “%d"时输出的变量值为整数

运行结果：

```
 a    b
97   98
```

例16 向字符变量赋整数

```
#include <stdio.h>
int main( )
{ char c1,c2 ;
  c1=97 ;
  c2=98 ;
 printf("%c %c \n",c1,c2);
 printf("%d %d \n",c1,c2);
}
```

- 思考：如何求大写字母 'F' 、'H' （或其他）分别是第几个字母？

- 思路：大写字母的ASCII码是从’A’到’Z’依次排序的，可以根据其ASCII码相减的结果来判断

- 'A'是第1个字母， 'B'是第2个字母，...

- 小写字母同理

★对字符数据进行算术运算

- 实质是对其ASCII值进行算术运算

例17 大小写字母的转换

```
#include <stdio.h>
int main( )
{char c1,c2 ;
  c1='a';
  c2= 'b';
  c1=c1-32;
  c2=c2-32;
  printf("%c %c ",c1,c2);
 }
```

小写字母比大写字母的ASCII码大$(32)_{10}$

运行结果：A B

★字符型与整型间互相赋值

★ 为避免歧义，尽量减少此类写法

例18 互相赋值

```
#include <stdio.h>
int main( )
{int c1;
  char c2 ;
  c1='a';
  c2=98 ;
  c1=c1-32;
  c2=c2-32;
  printf("%c %c ",c1,c2);
 }
```

例19 数字字符的使用

```
#include <stdio.h>
int main( )
{
/* 定义字符变量ch1并初始化为ASCII码值为9的字符*/
char ch1 = 9;
/* 定义字符变量ch2并初始化为数字字符9*/
char ch2 = '9';
printf ("***************\n");
/* 按照字符格式分别输出ch1和ch2中存储的字符，并换行*/
printf ("ch1:%c,ch2:%c\n",ch1,ch2);
printf ("***************\n");
return 0;
}
```

3.5.3 字符串常量

- 定义：用双引号(“ ”)括起来的字符序列
  - “How do you do”，“CHINA”，“a”，“$123.45”
- 存储：每个字符串尾自动加一个 ‘\0’ 作为字符串结束标志

例　字符串“hello”在内存中

h	e	l	l	o	\0

例　空串　“ ”

\0

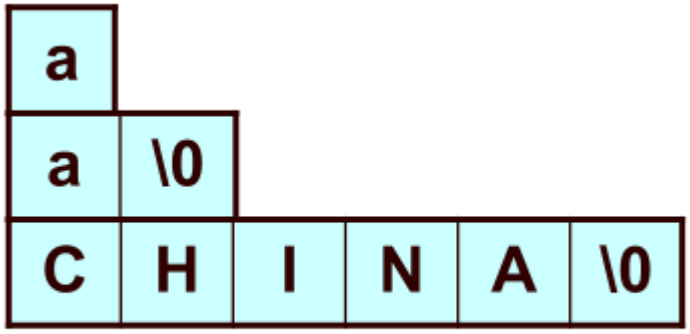
- 字符常量与字符串常量不同

例　‘a’ | a |　　“a” | a | \0 |

没有字符串变量，
只能用字符数组存放

例：
```
char  ch;
ch=“A”;
```
×

例：
```
char  ch;
ch=‘A’;
```
✓



存储方式

- char c; c='a';　　是正确的。
- char c; c="a" ;　　是错误的。
- char c; c="CHINA" ; 也是错误的。
- 不能把一个字符串赋给一个字符变量。

C规定：系统在每一个字符串的结尾自动加一个字符串结束标志’\0’，以便系统判断字符串是否结束。’\0’字符的ASCII码值为0，是空操作字符，无显示

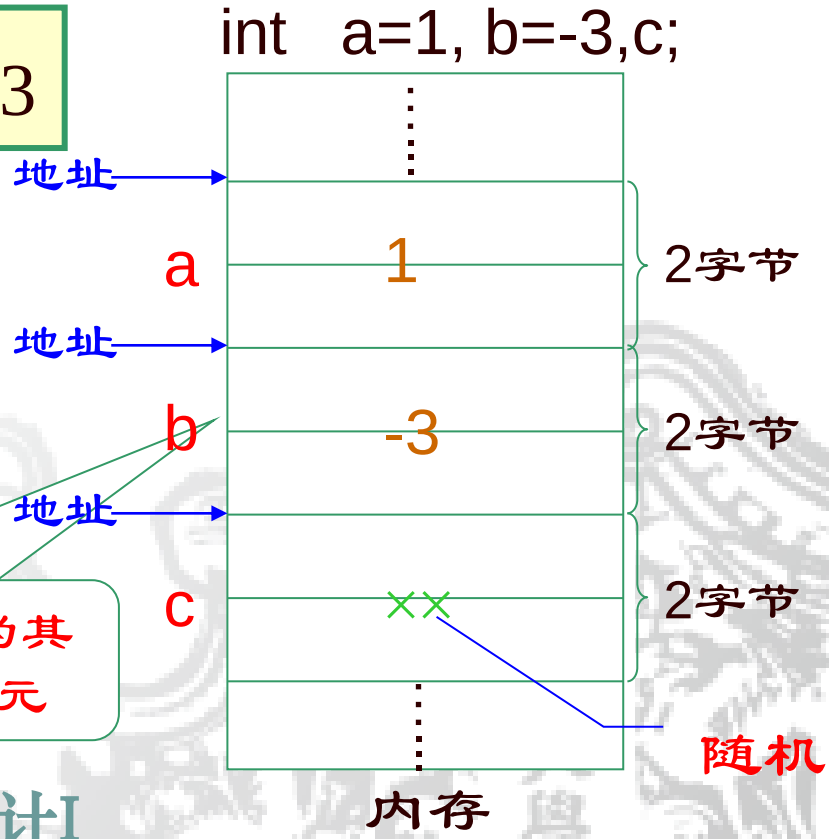
变量总结

- 变量的使用：先定义，后使用
- 变量定义位置：一般放在函数开头
- 变量初始化:可以在定义时赋初值

错！ int a=b=c=3

例:

```
int  a=1,b= -3,c;
float  data=3.67;
char  ch='A';
int  x=1,y=1,z=1;
int  x=y=1;（×）
```



编译程序根据变量定义为其分配指定字节的内存单元

3.6 C的运算符简介

- C语言的运算符范围很宽，把除了控制语句和输入输出以外的几乎所有的基本操作都作为运算符处理。

C运算符

算术运算符：（+ - * / % ++ --）
关系运算符：（< <= == > >= !=）
逻辑运算符：（（! && ||）
位运算符 ：（<< >> ~ | ^ &）
赋值运算符：（= 及其扩展）
条件运算符：（?:）
逗号运算符：（,）
指针运算符：（* &）
求字节数 ：（sizeof）
强制类型转换：（类型）
分量运算符：（. ->）
下标运算符：（[]）
其它 ：（（） -）

★学习运算符应注意：

- 运算符功能
- 与运算对象关系
  - 要求运算对象个数
  - 要求运算对象类型
- 运算符优先级别
- 结合方向
- 结果的类型

- 运算符：表示各种运算的符号。
- 运算对象：常量、变量、函数
- 运算符优先级别
  - 在表达式求值时，先按运算符的优先级别高低次序执行。

$$a-b*c \Longleftrightarrow a-(b*c)$$

- 运算符的结合方向(结合性)
  - 如果在一个运算对象两侧的运算符的优先级别相同，如a-b+c，则按规定的“结合方向”处理。
  - 算术运算符的结合方向:“自左至右”，即先左后右。“自左至右的结合方向”又称“左结合性”，即运算对象先与左面的运算符结合。
  - 运算符的结合方向:“自右至左”，即右结合性(例如，赋值运算符)。

3.7 算术运算符和算术表达式

❖基本算术运算符： + － * / %

- 运算对象：常量、变量、函数等
- 结合方向：从左向右
- 优先级： － ----->* / % -----> + －
  (2) (3) (4)

例 5%2 = 1
-5%2 = -1
1%10 = 1
5%1 = 0
5.5%2 (×)
a*b/c-1.5+'a'

说明：

- “－”可为单目运算符时，右结合性
- 两整数相除，结果为整数

例 5/2 = 2
-5/2.0 = -2.5

- %要求两侧均为整型数据，求值结果与第一个数的符号一致。
- + － * / 运算的两个数中有一个数为实数，结果是double型

- 算术表达式：由算术运算符和圆括号将运算对象连接起来的，符合C语言语法规范的式子。
  - 整个式子的运算结果作为表达式的值
  - 运算结果的类型作为表达式的类型

- 例如：9%3 就是一个算术表达式，此表达式的值为0，
- 表达式的类型是int类型。

- 例20 表达式的使用

- 假如星期日代表0，星期一代表1，星期二代表 2 …，变量today存放今天的数字，问tomorrow是多少，today过后的第100天呢？

- tomorrow = today + 1 正确吗 ??
- 参考答案： tomorrow = (today + 1) % 7
- 第100天: hund = (today + 100) % 7

- 课堂练习

- 有任意一个三位数，如 number = 468; 如何
- 获取此三位数各位上面的数字？

- 参考答案：
- 个位: u = number % 10 ;
- 十位: t = (number / 10) % 10;
- 百位: h = number /100 ;

自增、自减运算符

- 作用：使变量值加1或减1
- 种类：
  - 前置　++i，--i　（先执行i+1或i-1，再使用i值）
  - 后置　i++, i--　（先使用i值, 再执行i+1或i-1）

例

```
j=3; k=++j;
j=3; k=j++;
j=3; printf("%d",++j);
j=3; printf("%d",j++);
a=3;b=5;c=(++a)*b;
a=3;b=5;c=(a++)*b;
```

- 几点说明：
  - ++ -- 不能用于常量和表达式，如 5++，(a+b)++
  - ++ -- 结合方向： 自右向左
  - 优先级：- ++ -- ------>* / % ----->+ -
    (2) (3) (4)
  - 该运算符常用于循环语句中，使循环变量加减1

例 -i++ ⟺ -(i++)
i=3; printf("%d",-i++); 输出-3

❖有关表达式使用中的问题说明

- 不同系统对运算符和表达式的处理次序不同，尽可能写通用性强的语句
- 不要写有歧义和不知系统如何执行的程序

例 int i = 3 ; 表达式 (i++) + (i++) + (i++) 的值是？
可能性1：结果为3+4+5，即12 正确
可能性2：结果为3+3+3，即 9

- 课堂练习
- 执行下列语句的结果为__11,11____。
- i=012;
- printf("%d,",++i);
- printf("%d", i++);

- 执行下列语句的结果为___6, 6___。
- int sum=1,pad=5;
- sum=++pad;
- printf("%d, %d\n",pad, sum);

- 课堂练习
- 以下程序输出的结果是___9,10___。
- int i = 010, j = 10;
- printf("%d,%d\n", ++i, j--);

- 以下程序输出的结果是___i=3, j=-2 ___。
- int i, j;
- i = 2;
- j = -i++;
- printf(“i=%d, j=%d\n”,i, j);

3.8 赋值运算符和赋值表达式

❖简单赋值运算符

- 符号：　=
- 格式：　变量标识符=表达式
- 作用：将一个数据（常量、变量或表达式）赋给一个变量
- 左侧必须是变量，不能是常量或表达式
- 结合方向：　自右向左
- 优先级：　14级

例　a=3;　　d=func();　　c=d+2;

例　3=x-2*y;　a+b=3;　（×）

例　a=(b=5) ⟺ a=b=5

❖复合赋值运算符

- 种类：+= -= *= /= %= 《= 》= &= ^= |=
- 含义： exp1 op= exp2 ⇔ exp1 = exp1 op exp2
- 右下角

a+=3 ⟺ a=a+3

x*=y+8 ⟺ x=x*(y+8)

x%=3 ⟺ x=x%3

❖赋值表达式

- 形式：<变量> <赋值运算符> <表达式>
- 赋值表达式的值与变量值相等,且可嵌套

例：

```
a=b=c=5          //表达式值为5， a, b, c值为5
a=(b=5)          //  b=5;a=5
a=5+(c=6)        //表达式值11， c=6, a=11
a=(b=4)+(c=6)    //表达式值10， a=10, b=4, c=6
a=(b=10)/(c=2)   //表达式值5， a=5, b=10, c=2
```

- 例21 求复合赋值表达式的值

```
#include <stdio.h>
int main( )
{
    int a = 3;
    int b = 5;
    printf("%d\n", a+=a-=a*a);
    printf("%d\n", b+=b-=b*=b);
    return 0;
}
```

```
-12
0
```

- 课堂练习
- 写出下面赋值表达式运算后a的值,设原来
- int a=12, n=5;
- 1.a+=a　　24
- 2.a-=2　　10
- 3.a*=2+3　　60
- 4.a%=(n%=2)　　0
- 5.a/=a+a　　0
- 6.a+=a-=a*=a　　0

3.9 逗号运算符和逗号表达式

- 形式：表达式1,表达式2,……表达式n
- 结合性:从左向右
- 优先级: 15，级别最低
- 逗号表达式的值：等于表达式n的值
- 用途：常用于循环for语句中

例　a=1;b=2;c=3;
　　printf("%d,%d,%d",a,b,c);
例　for(i = 0;i<10;i++){}

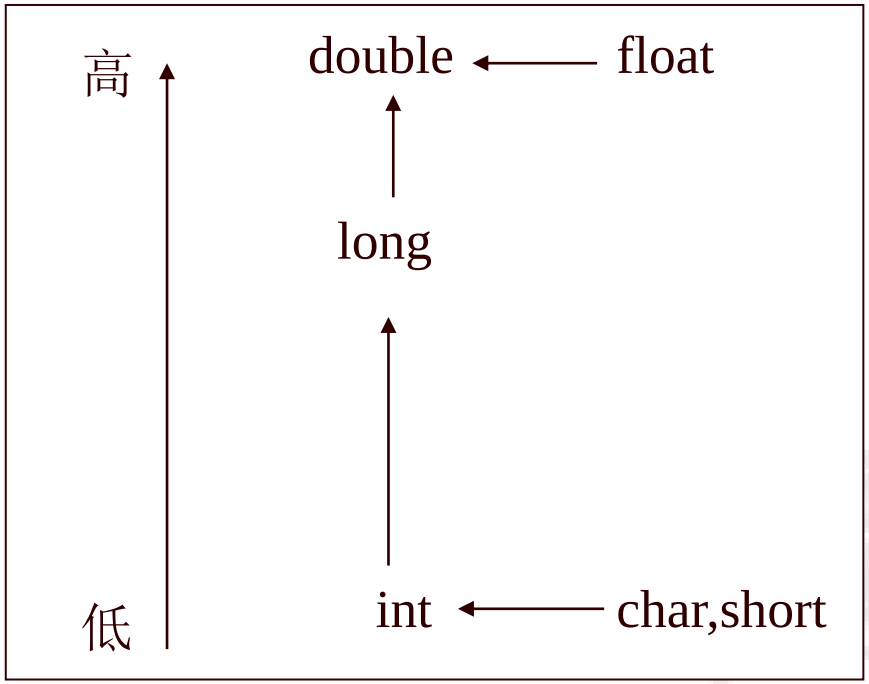
3.10 类型转换

★自动转换

❖什么情况下发生

- 运算转换------不同类型数据混合运算时
- 赋值转换------把一个值赋给与其类型不同的变量时
- 输出转换------输出时转换成指定的输出格式
- 函数调用转换------实参与形参类型不一致时转换

- 运算转换规则
  - 不同类型(整型、实型、字符型)数据运算时先自动转换成同一类型

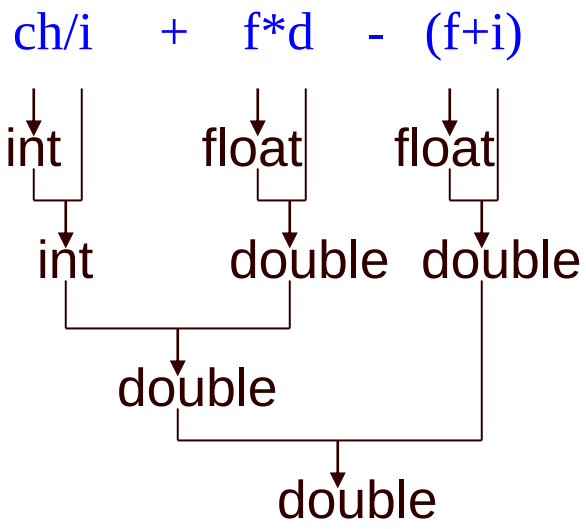


说明:

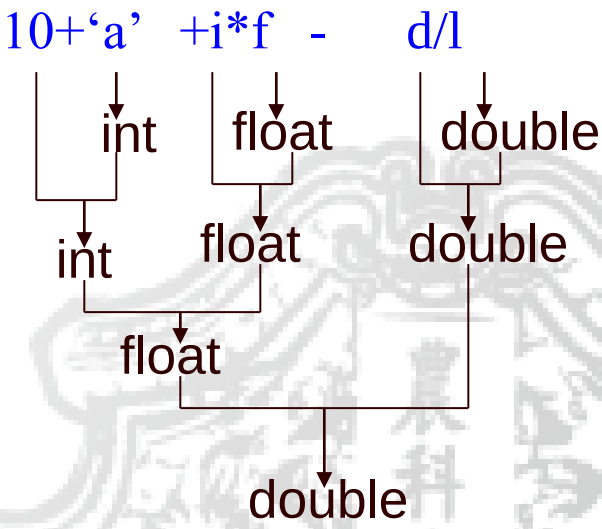
←——— 必定的转换

↑ 运算对象类型不同时转换

例 char ch;
int i;
float f;
double d;



例 int i;
float f;
double d;
long l;



❖赋值转换规则

- ❖使赋值号右边表达式值自动转换成其左边变量的类型
- （1）若"="左边的类型低于右边的类型，一般为截取方式。
- （2）若"="左边的类型高于右边的类型，一般为补齐方式。

例　float f ;　int i=10;　f=i;
则　f=10.0

例　int i;
i=2.56;　//结果i=2;

★强制转换（见P56强制类型转换运算符部分）

❖一般形式：（类型名）（表达式）

例：(int)(x+y)

(int)x+y

(double)(3/2)

(int)3.6

强制类型转换运算符

❖说明：强制转换得到所需类型的中间变量，原变量类型不变

```
例22
#include <stdio.h>
int main()
{   float  x;
    int  i;
    x=3.6;
    i=(int)x;
    printf("x=%f,i=%d",x,i);
}
结果：  x=3.600000,i=3
```

表达式仅一个变量时，括号可以省略

精度损失问题

较高类型向较低类型转换时可能发生

- 例23 强制类型转换

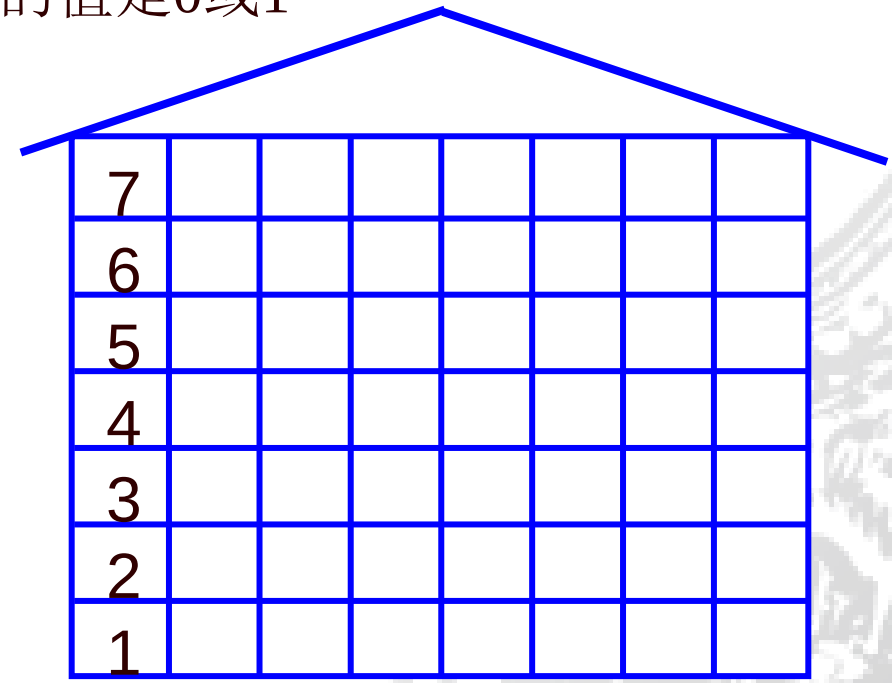
```
#include <stdio.h>
int main( )
{
    int a = 3, b = 5;
    float x,y,z;
    x = a/b;
    y = (float)a/b;
    z = (float)(a/b);
    printf("x = %f\n",x);
    printf("y = %f\n", y);
    printf("z = %f\n", z);
    return 0;
}
```

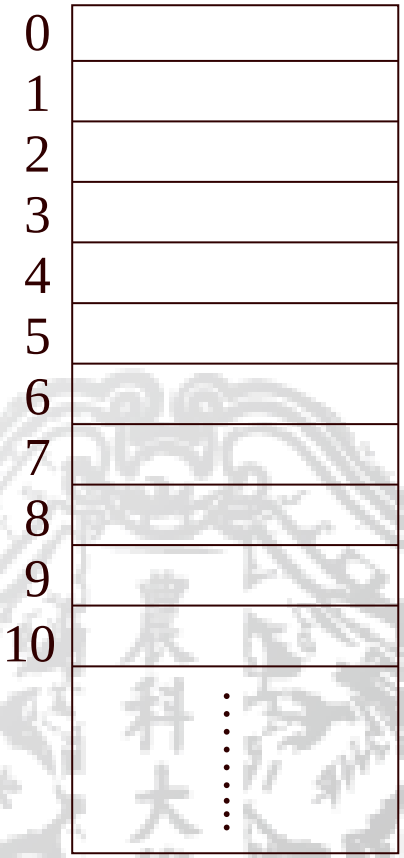
```
x = 0.000000
y = 0.600000
z = 0.000000
```

3.3.4 整型常量的表示方法(补充知识)

★补充知识：字节和位

- ❖内存以字节为单元组成
- ❖每个字节有一个地址
- ❖一个字节一般由8个二进制位组成
- ❖每个二进位的值是0或1





3.3.4 整型常量的表示方法

★补充知识：数值的表示方法（原码、反码和补码）

- ❖原码：最高位为符号位，其余各位为数值本身的绝对值
- ❖反码：
  - 正数：反码与原码相同
  - 负数：符号位为1，其余位对原码取反
- ❖补码：
  - 正数：原码、反码、补码相同
  - 负数：最高位为1，其余位为原码取反，再对整个数加1

（用一字节表示数）

	原码	反码	补码
+7	00000111	00000111	00000111
-7	10000111	11111000	11111001
+0	00000000	00000000	00000000
-0	10000000	11111111	00000000
数的范围	01111111~ 11111111 (-127~+127)	01111111~ 10000000 (-127~+127)	01111111~ 10000000 (-128~+127)

- C语言以补码(complement) 表示的
- 负数补码转换成十进制数：最高位不动，其余位取反加1

3.3.5 整型变量的存放形式

★ 数据在内存中以二进制补码形式存放

★ 每一个整型变量在内存中占2个字节

10的原码	00	00	00	00	00	00	10	10
反码	00	00	00	00	00	00	10	10
补码	00	00	00	00	00	00	10	10
-10的原码	10	00	00	00	00	00	10	10
取绝对值	00	00	00	00	00	00	10	10
反码	11	11	11	11	11	11	01	01
补码	11	11	11	11	11	11	01	10

整数的最左二进制位是符号位，0正、1负

字符型数据取值范围

说明	数据类型名	字节	取值范围
[有符号]字符型	**char**	**1**	**-128～127**
无符号字符型	**unsigned char**	**1**	**0～255**

- 说明：很多系统将将字符处理成带符号的整数，即
- signed char型。它的取值范围是-128～127。

- 如果使用ASCII码为0～127间的字符，由于字节中最高位为0，用%d输出时，会输出一个正整数。
- 如果使用ASCII码为128～255间的字符，由于在字节中最高位为1，用%d格式符输出时，就会得到一个负整数。