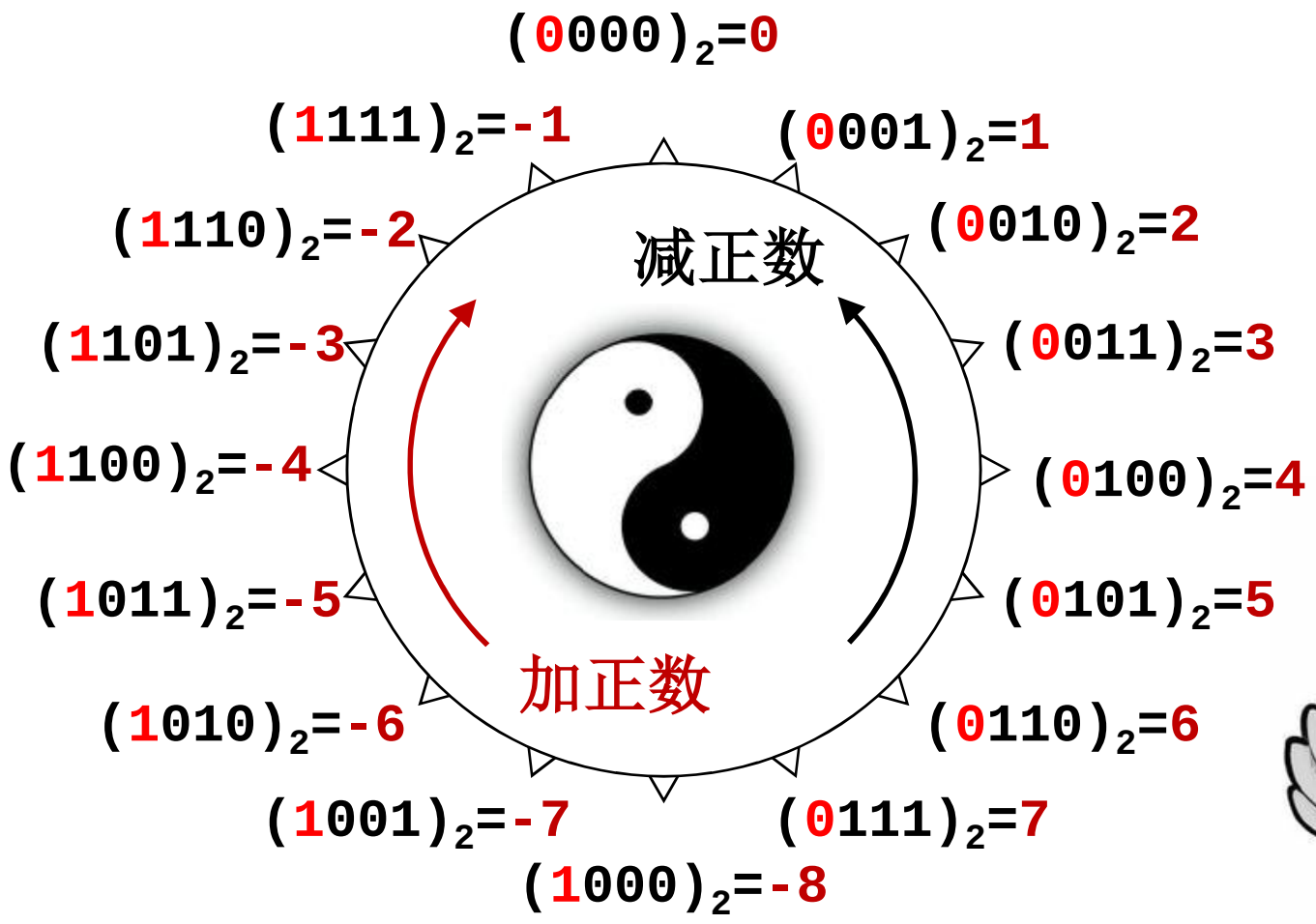
第3章 顺序程序设计（2）

复习回顾

- Ø 上次课的内容
  - u 词法与词汇
  - u 常量
  - u 变量
  - u 数据类型
    - l 整型
    - l 浮点型
    - l 。。。
  - u 欣赏一下

l C语言程序员专属的浪漫情书

再见补码：四位二进制数补码图



基本数据类型之三：字符型

字符型（char）：一个字符按其对应的ASCII码的整数形式存储。因此字符型数据类似于小范围的整型数据。

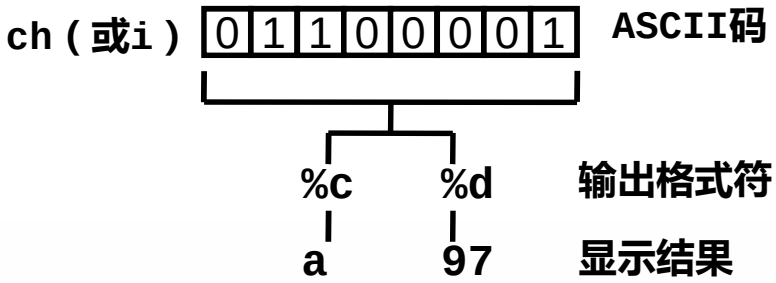
类型	类型标识符	在内存所占的字节数	数值范围
基本字符型	char	1	依编译系统而定
有符号字符型	signed char	1	-128 ~ 127 即-2^7~(2^7-1)
无符号字符型	unsigned char	1	0 ~ 255 即 0~(2^8-1)

如何测定你的系统的char是signed还是unsigned？

```
1.  #include <stdio.h>
2.  int main()
3.  {
4.      char c = 255; //把一个128～255之间的整数赋值给字符变量c
5.      printf("c=%d\n", c);//若输出负数，则为signed，否则为unsigned
6.      return 0;
7.  }
```

字符型变量与整数的关系举例

- 字符型变量的值可以赋给一个整型变量；
  - 如右例第8行
- 字符型变量可以和整数混合运算；
  - 如右例第7行
- 整数值可以以字符的形式输出，反之亦然。
  - 如右例第9、10行




```
1.  #include <stdio.h>
2.  int main()
3.  {
4.      char ch;
5.      int i;
6.      ch = 'A';
7.      ch = ch+32;
8.      i = ch;
9.      printf("%d is %c\n",ch,i);
10.     printf("%c is %d\n",ch,ch);
11.     return 0;
12. }
```

基本数据类型之四：空型

- 空值型：又称无值型
  - 用关键字void表示, void型的值集为空集,可出现在函数定义的头部,表示该函数没有返回值
  - 例如

```
1.void main()
2.{
3.      prinf("An example of void type.\n");
4.}
```

怎样确定常量的类型

Ø在CodeBlocks中

- u 整型常量
  - l 凡在-2^{31}~$2^{31}-1$之间的不带小数点的数都作为int型，分配4个字节
- u 浮点型常量
  - l 凡以小数形式或指数形式出现的实数，**都按双精度(double型）处理，分配8个字节**
  - l 所以`float a=3.14159;` 会有“警告”

小结：数据信息=位+类型

- 同样的二进制状态根据不同的数据类型将呈现不同的值
  - 例如：已知某个变量占有2个字节，它的二进制状态是$(1111111111111111)_2$，请问它的值是多少？
    - 正确答案：不知道！
    - 如果是`unsigned short`类型，那么值是 65535
    - 如果是`short`类型，那么值是 -1
- 同样的表达方式可以有不同的解读！

运算符列表

(1) 算术运算符（7个）： +、-、*、/、%、++、--

(2) 关系运算符（6个）： >、<、>=、<=、 ==、 !=

(3) 赋值运算符（11个）：=、+=、-=、*=、/=、%=、&=、|=、^=、<<= 、>>=

(4) 逻辑运算符（3个）： ！ 、&& 、||

(5) 位运算符（6个）： << 、>> 、~ 、| 、^ 、& ←双目运算符

(6) 条件运算符（1个）： ？ ：

(7) 逗号运算符（1个）： ，

(8) 指针运算符（2个）： *、& ←单目运算符

(9) 求字节数运算符（1个）： sizeof

(10) 分量运算符（2个）： .、->

(11) 下标运算符（1个）： [ ]

(12) 强制类型转换运算符（1个）：（数据类型名）

基本的算术运算符

- 键盘无×号，运算符×以*代替
- 键盘无÷号，运算符÷以/代替
- **两个整数相除的结果为整数！**舍去小数部分，多数C编译系统采取“**向零取整**”的方法，即5/3=1，-5/3=-1，3/(-5)=0
- %运算符要求参加**求模运算的对象必须为整数**，结果也为整数，如5%3=2

求模其实没那么简单

Ø猜猜CodeBlocks的实验结果

- u (-5) % 3 = -2　　u (-5) % 6 = -5
- u 5 % (-3) = 2　　u 5 % (-6) = 5
- u (-5) % (-3) = -2　　u (-5) % (-6) = -5

Ø带负数的求模运算结果的规律

- u 模的绝对值可按正数求模获得；
- u 模（余数）的符号依赖于被除数；
- u 可能因编译器的具体实现不同而发生变化！

求模：数学 vs 计算机

Ø数学：$m\%n = m - n * \lfloor m / n \rfloor$

- 其中，m 、n 的结果是精确值，$\lfloor x \rfloor$ 表示向下取整，
- 例如：

$(-3)\%2 = -3 - 2 * \lfloor (-3)/2 \rfloor = -3 - 2 * \lfloor -1.5 \rfloor = -3 - 2 * (-2) = 1$

Ø计算机（以CB为例）$m\%n = m - n * (m/n)$

- 其中，(m/n) 的结果是整数值，经常是近似值
- 例如：$(-3)\%2 = -3 - 2 * ((-3)/2) = -3 - 2 * (-1) = -1$
- 对于计算机，数学的带负数的求模代价太大，伤不起！

C语言里的表达式

- 表达式是显示如何运算的公式，目的是获得运算的结果
  - **最简单的表达式**：变量和常量
    - 如a，1
  - **复杂的表达式**把运算符应用于运算对象
    - 如x-(3*y)，其中(3*y)是子表达式
  - **每个表达式都有结果值，具有特定的类型**
    - 如1+2的类型为int
    - 如果没有产生值，类型为void

运算符的优先级和结合性

- Ø 当表达式包含多个运算符时，解释表达式就产生困难，C语言通过**运算符优先级规则**加以解决
  - u 如 x+y*z，先求y*z的值，然后与x相加
  - u 详见教材附录D（记住**乘除优先于加减**就差不多啦）
  - u 分清“左结合性”（如x-y+z) 与“右结合性”(如x=y=z)
- Ø 对优先级和结合方向感觉不确定怎么办？
  - u **加括号！！！！！！！！！！！！！！**

运算符与表达式举例

- **单目算术运算符**：-1（取负），+2（取正）、!a（取反）
- **双目算术运算符**：0+1、2-3、4*5、6/7、8%9
- **三目算数运算符**：c = (a>b)?a:b;
- **算术表达式举例：** 数学式 vs C算术表达式

$\frac{-b+\sqrt{b^2-4ac}}{2a}$	(-b+sqrt(b*b-4*a*c))/(2*a)
$(x+\sin x)e^{4x}$	(x+sin(x))*exp(4*x)
$\frac{pr^2}{a+b}$	3.14159*r*r/(a+b)

注意：C程序中，

（1）乘号*不能省略，如 2x+3y **是错的！**

（2）表达式总是从左到右同一行书写；

（3）无中括号和大括号，小括号可以多层。

赋值运算符

Ø简单赋值运算符：=

- u 如a=4，把常量4赋值给变量a；
- u 再如a=a+1，取出变量a中原来的值加1后放在a的存储单元里
- u 编程时，通常运算符不修改其运算对象的值。赋值运算符的独特作用：**改变左侧运算对象的值**！

赋值表达式

Ø 一般形式：

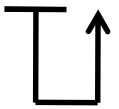
变量 赋值运算符 表达式

- 注意：赋值运算符左边必须是变量，右边可以是任意表达式（包括赋值表达式）。
- 如，3=a 或者 (a=b)=3*4 或者 #define N 10
  N=100 都是不合法的
- 以下是合法的，但强烈不推荐这么写！
  - a=b=c=5
  - a=5+(c=6)
  - int a=2; printf("%d",a=3);

复合赋值运算符

- 一般形式： 变量 双目运算符= 表达式
  - 等价于：变量 = 变量 双目运算符 表达式
  - 例如：
    - x%=10 //等价于x=x%10
    - x*=x-y //等价于x=x*(x-y)
    - x/=x+y //等价于x=x/(x+y)
    - x+=y //等价于x=x+y

① x*=x+y
② x*=x+y



得 =x*(x+y)
③ x=x*(x+y)

- 老手用来简化程序，提高编译效率，新手慎用！

复合赋值运算符举例

Ø设a=12，计算a+=a-=a*a的值

1. 根据赋值计算的右结合性，先计算a-=a*a的值，相当于a=a-a*a，值为-132。所以变量a的值为-132
2. 接着再进行a+=-132的运算，相当于a=a+(-132)，因而最后变量a的值为-264

古怪的运算符和表达式

- Ø自增（++）与自减（--）运算符
  - u属于单目运算符
    - l单独的i++和++i相当于i=i+1
    - l单独的i--和--i相当于i=i-1
  - u只能用于变量，不能用于常量或表达式
    - l如5++和(a+b)++都是不合法的
  - u当作为表达式的一部分，运算符位于变量的左侧和右侧，具体含义不同！

++与--造成的小困惑

Ø设i=3，判断下列表达式的结果：

u(1)j=++i;(2)j=i++;

l(1)表示i的值先变成4，再赋给j，j的值为4

l(2)表示先将i的值3赋给j，j的值为3，然后i变成4

u(1)printf("%d",++i);(2)printf("%d",i++);

l(1)等价于先i=i+1;然后printf("%d",i);结果输出4

l(2)等价于先printf("%d",i);然后i=i+1;结果输出3

++与--造成的困惑进阶

Ø 下面程序执行后a和b的值各是多少？

```
1. int a,b;
2. a = 1;
3. b = (++a*a++)+(++a);
```

CodeBlocks运行结果：
a=4, b=7

u 【猜测】若表达式多次出现对同一个变量的++(或--)运算，就必须先从左开始处理完所有的左侧++(或--)之后，再利用变量的当前值参与所在的子表达式运算，最后处理整个式子所有的右侧++(或--)

Ø 终极困惑：i+++j

u 是理解为(i++)+j呢？

u 还是i+(++j)?

建议：永远不要采用这样的写法，加括号以避免歧义。

逗号运算符和逗号表达式

1、逗号运算符

在C语言中，逗号","既是一个分隔符，又是一个运算符，即逗号运算符。它是所有C运算符中优先级别最低的，其结合性是从左往右。

2、逗号表达式

【语法格式】表达式1,表达式2,表达式3,…,表达式n

【求解过程】从左往右计算各个表达式的值，并且规定把最后一个表达式n的值作为整个逗号表达式的值。

【例如】 逗号表达式 16+5,6+8 的值为14

逗号表达式 a=2*3,a*4 的值为24 思考：a=？

赋值过程中的类型转换

- 如果赋值运算符两侧的类型一致，则直接进行赋值，如`int a; a = 234;`
- 若不一致，但都是算术类型时，系统自动将右侧类型转换成左侧类型后进行赋值
  - 具体5种情况，见教材（P62）
  - 不同类型之间赋值，常出现数据的失真，但又不属于语法错，编译系统不会报错，全靠经验解决问题

赋值过程中的类型转换举例

```
1.  #include <stdio.h>
2.  int main()
3.  {
4.      int i; short s; char c;
5.      float f; double d;

6.      printf("【执行赋值】d=f=23; i='i'\n");
7.      d=f=23; i='i';
8.      printf("结果正常：d=%lf,f=%f,i=%d\n",d,f,i);

9.      printf("【执行赋值】i=3.56; f=123.456789\n");
10.     i = 3.56; f = 123.456789;
11.     printf("损失精度：i=%d,f=%f\n",i,f);

12.     printf("【执行赋值】c=289; i=32767;s=i+1;\n");
13.     c=289;  i=32767;s=i+1;
14.     printf("发生\"截断\"：c=%d,s=%d\n",c,s);
15.     return 0;
16. }
```

```
【执行赋值】d=f=23; i='i'
结果正常：d=23.000000,f=23.000000,i=105
【执行赋值】i=3.56; f=123.456789
损失精度：i=3,f=123.456787
【执行赋值】c=289; i=32767;s=i+1;
发生"截断"：c=33,s=-32768
Press any key to continue
```

赋值中的转换：浮点型与整型

- 将浮点型转换为整型时，将舍弃浮点数的小数部分，只保留整数部分，**内存形式将发生改变**；
- 将整型赋值给浮点型变量，数值不变，可想像成改为浮点形式（即小数点后带若干个0），**内存形式也将发生改变**。

```
1. int i = 10.123;        //i的值将为10
2. float f = 10;          //f的值将为10.000000
3. i = 13.123 - i;        /* 等号右边值为3.123,
                             i的值为3 */
```

赋值中的转换：单双精度浮点型

- float类型转为double类型时只需在尾部加0，即可延长为double类型；
- double类型转为float型时，通过截取尾数来实现，截取前一般要进行四舍五入操作（截取的位数和是否四舍五入与具体实现有关）。

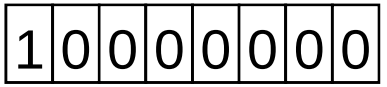
```
1. float f;
2. double d = 12345.678;
3. f = d;   // 据CB结果，f的值为12345.677734
```

赋值中的转换：char与int

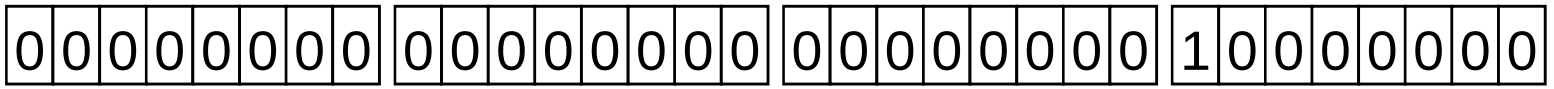
- int 型赋值给 char 型变量时，只保留其最低8位，高位部分舍弃；
- char 型赋值给 int 型变量时，将使用字符的ASCII代码进行赋值，某些编译器不管其值大小都作为正数处理，另一些在转换时，若其值大于127，就作为负数处理。

```
1. int j = 'A';    //j的值为65
2. char c = 128;   //截取低8位
3. int i = c;      //可能是128或-128
```

c（char型）

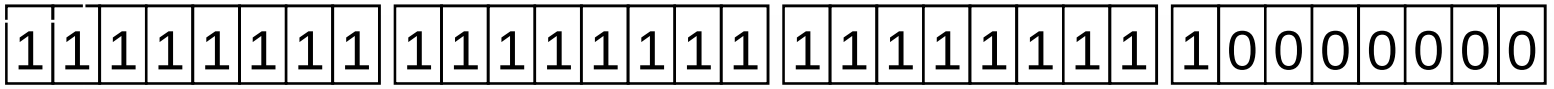


i（int型,128）



或者

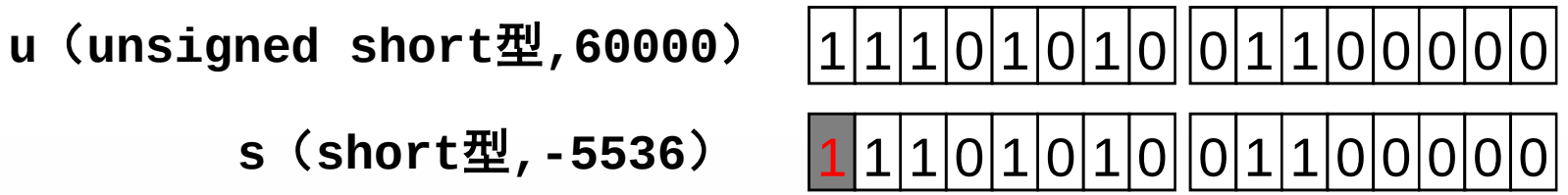
i（int型,-128）



赋值中的转换：有/无符号数

- 将一个无符号整型数据赋值给一个占据同样长度内存的有符号整型变量时，**内存形式不变，但外部值可能改变**；
- 将一个有符号整型数据赋值给长度相同的无符号变量时，**内存形式不变，外部表示总是无符号的**

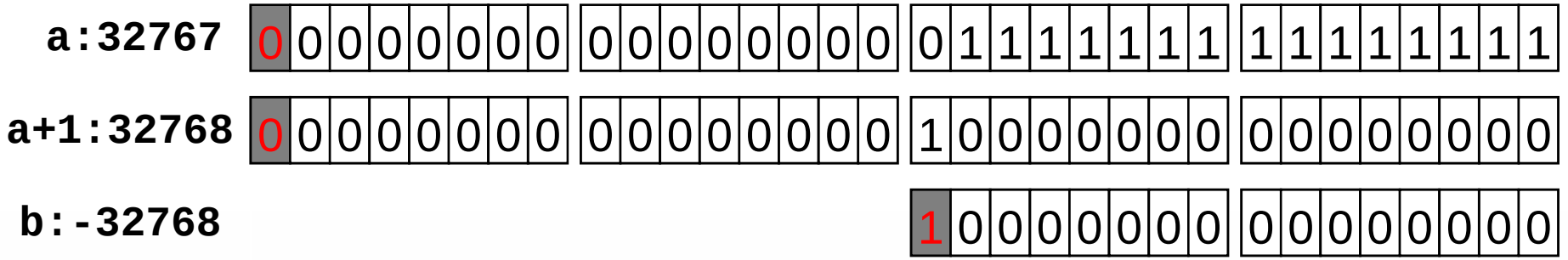
```
1. unsigned short u = 60000; //short int占16位
2. short s = u;              //s的值为-5536
3. u = s;                    //u的值为60000
```



赋值中的转换：“截断”现象

Ø 将一个占字节多的整型数据赋值给一个占字节少的整型变量或字符变量时，只将其低字节原封不动地送到被赋值的变量（即发生“截断”）。

```
1. int a = 32767;   //int占32位
2. short b;         //short占16位
3. b = a+1;         //b的值为-32768
```



不同数据类型间的混合运算

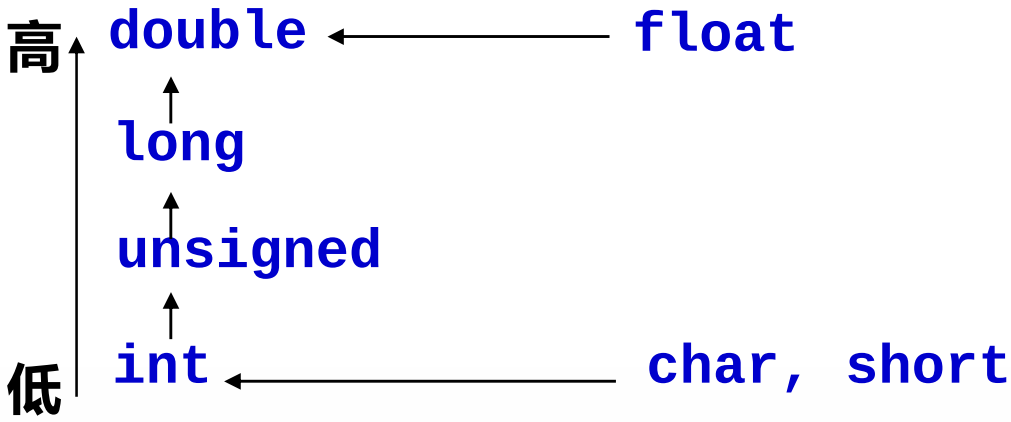
- 一个双目运算符两边的操作数的**类型必须一致**，才能进行运算操作。
- C语言允许在一个表达式中的双目运算符两边出现不同数据类型的操作数。
  - 运算时，C编译器会先对其中的一些操作数按约定的规则自动进行类型转换（即**隐式类型转换**），使得双目运算符两边的操作数的类型一致，然后进行运算。
  - 例如 `3+'A'`，结果是 `int` 类型的 `68`

隐式类型转换：C语言的潜规则

Ø 无条件的隐式类型转换：

- u 所有的char型和short型数据在运算前都必须转换为int型；
- u 所有的float型数据都必须转换成double型。
- u 即使运算符的两个操作数同是以上某种类型，也要进行类型转换

Ø 统一类型的隐式类型转换：如果双目运算符两边的操作类型不一致，则需要将其中类型较低的转换为较高的类型，然后基于同一类型进行运算。



隐式类型转换举例

```
1. float f = 30.0;
2. double result = 'A'+ 'B' - f ;
```

说明：f为float型变量，则算术表达式 'A'+'B'-f 求值时，不同数据类型的转换及运算顺序为：

（1）进行 'A'+ 'B' 的运算：先将'A'和 'B'都转换成 int 型（无条件转换），得到65和66，然后对这两个 int 型数据进行加法运算，结果为 int 型数据131。

（2）进行 131-f 的运算：先将 int 型的 131 和 float 型的 f 都转换成 double 型，然后相减（其中，int 到 double 型的转换是统一类型转换，float 到 double 型的转换是无条件转换），得到的结果是double型。

“逆天”的强制类型转换运算符

- 可以用强制类型转换运算符将一个表达式转换成所需类型
  - 一般形式：(类型名)(表达式)
    - 例如：(double)a; (float)(5%3);
      y=(int)x%3; 5/(float)3;
  - 注意：强制类型转换得到的是一个临时值，存放在临时存储单元，运算对象本身的类型和值并没有变化
    - 例如：a=(int)x; x的整数部分（即临时值）赋值给a，x本身不变。

强制类型转换运算符的优先级

- Ø因为C语言将强制类型转换运算符视为**一元运算符**，其优先级要高于二元运算符
  - u 例如：
    - l 编译器会将 `(int)f+d` 解释为 `((int)f)+d`
    - l 若 f 为 float 型，d 为 double 型，`(int)f+d` 的结果类型将是 double 型
    - l 若期望最终得到 int 型的计算结果，表达式应改为 `(int)(f+d)`

强制类型转换的作用举例

Ø求精确平均值

- u比如希望求a,b,c三个整型变量精确平均值，表达式应写为(double)(a+b+c)/3，或者(a+b+c)/(double)3

Ø防止溢出

- u比如double d=111111*111111，等式右边计算将溢出，d得到诡异的结果；若改为(double)111111*111111则不会溢出

作业 2016/09/27

Ø 完成下列实验，提交纸版实验报告，汇报观察到的现象并试解释原因（下次课9月30日课间交）

- u 实验A1：设有int a,b=<你的学号末尾4位>;表达式a=b*b的值是多少？改成末尾6位呢？如果末尾9位呢？试解释其中原因
- u 实验A2：在取学号末尾4位、6位、9位情况下，把实验A1中所有数换成单精度浮点数类型，结果如何？试解释其中原因
- u 实验A3：在取学号末尾4位、6位、9位情况下，把实验A1中所有数换成双精度浮点数，结果如何？ 试解释其中原因
- u 实验A4：表达式int c=<学号末尾4位>*<学号末尾4位>的值是多少？改成末尾6位和9位呢？和实验A1的结果有没有不同？
- u 实验A5：在取学号末尾4位、6位、9位情况下，把实验A4中所有数换成单/双精度浮点数类型，结果又如何？试解释其中原因

Ø 上机练习（不用交）：编译运行本讲义例程