第二十六节：连乘、自乘、自乘简写，溢出。

【26.1　连乘。】

上一节的乘法例子中，右边的乘数只有两个。实际上，C 语言规则没有限制乘数的个数，它的通用格式如下：

```
“保存变量”=“乘数 1”*“乘数 2”...*“乘数 N”;
```

当右边的乘数个数超过两个的时候（这里暂时把平时所说的被乘数也归类为乘数），这种情况就是“连乘”。每个乘数的属性没有限定，可以是常量，也可以是变量。比如：

```
unsigned char x=3;   //定义一个变量 x，初始化默认为 3.
unsigned char y=6;   //定义一个变量 y，初始化默认为 6.
unsigned char k=2;   //定义一个变量 k，初始化默认为 2.
a=2*5*3;  //乘数全部是常量。a 的结果为 30。
b=k*x*y;  //乘全部是变量。b 的结果为 36。
c=x*5*y;  //乘数，有的是常量，有的是变量。c 的结果为 90。
```

连乘的运行顺序是，赋值符号“=”右边的乘数挨个相乘，把每一次的运算结果放在一个临时的隐蔽中间变量里，这个隐蔽的变量我们看不到，是单片机系统内部参与运算时的专用寄存器，等右边所有乘数连乘的计算结果出来后，再把隐蔽变量所保存的计算结果赋值给左边的“保存变量”。

【26.2　自乘与自乘简写。】

什么是自乘？当赋值符号“=”右边的乘数只要其中有一个是“保存变量”本身时，这种情况就是“自乘”，常见格式如下：

```
“保存变量”=“保存变量”*“乘数 1”;
“保存变量”=“保存变量”*(“乘数 1”*“乘数 2”...*“乘数 N”);
```

上述自乘计算式可以简写成如下格式：

```
“保存变量”*=“乘数 1”;
“保存变量”*=“乘数 1”*“乘数 2”...*“乘数 N”;
```

这种格式就是“自乘简写”。现在举几个例子如下：

```
unsigned char d=5;       //定义一个变量 d，初始化默认为 5.
unsigned char e=5;       //定义一个变量 e，初始化默认为 5.
unsigned char f=5;       //定义一个变量 f，初始化默认为 5.

unsigned char x=3;   //定义一个变量 x，初始化默认为 3.
unsigned char y=6;   //定义一个变量 y，初始化默认为 6.
unsigned char k=2;   //定义一个变量 k，初始化默认为 2.
d*=6;     //相当于 d=d*6;最后 d 的结果为 30。
e*=x;     //相当于 e=e*x;最后 e 的结果为 15。
f*=2*y*k; //相当于 f=f*(2*y*k);最后 f 的结果为 120。
```

【26.3　有没有“自乘 1”的特殊写法？】

之前在讲加法的自加和减法的自减运算时，还给大家介绍了它们另外一种特殊的简写方式。比如减法运算，当右边只有 2 减数，当一个减数是“保存变量”，另一个是常数 1 时，格式如下：

“保存变量”=“保存变量”-1；

这时候，可以把上述格式简写成如下两种格式：

“保存变量”--；

--“保存变量”；

这两种格式也是俗称的“自减 1”操作。比如：

```
g--;  //相当于 g=g-1 或者 g-=1;
--h;  //相当于 h=h-1 或者 h-=1;
```

那么，本节所讲的自乘运算，有没有“g**”或者“**h”这种特殊的“自乘 1”写法？答案很明显，C 语言里没有“自乘 1”这种特殊写法。因为任何一个数“自乘 1”还是等于它本身，所以在乘法运算中这种特殊写法就没有存在的意义。多说一句，如果某天有朋友在某个地方看到“**h”这类语句，它的本意跟“自乘”没关系，而是跟 C 语言的另一块知识点“指针”有关。

【26.4　乘法的溢出。】

乘法的溢出规律跟加减法的溢出规律是一样的。举一个例子如下：

```
unsigned char m=30;
unsigned char n=10;
unsigned char a;
a=m*n;
```

分析：m 与 n 相乘，相当于 30 乘以 10，运算结果是 300（十六进制是 0x012c）保存在一个隐藏中间变量，根据前面加减法运算的规律，我猜测这个隐藏中间变量可能是 unsigned int 类型，然后再把这个中间变量赋值给单字节变量 a，a 只能接收十六进制的低 8 位字节 0x2c，所以运算后 a 的数值由于溢出变成了十六进制的 0x2c（十进制是 44）。由于乘法的溢出规律跟加减法的溢出规律是一样的，所以不再多举例子。在实际项目中，为了减少溢出的现象，我建议，不管加减乘除，凡是参与运算的变量全部都应该转化成 unsigned long 变量，转化的方法已经在前面章节讲过，不再重复讲解这方面的内容。

【26.5　例程练习和分析。】

现在编写一个程序来验证刚才讲到的连乘和自乘简写：

程序代码如下：

```
/*---C 语言学习区域的开始。-----------------------------------------------*/

void main() //主函数
{
    unsigned char a;
    unsigned char b;
    unsigned char c;
    unsigned char d=5;      //定义一个变量 d，初始化默认为 5.
    unsigned char e=5;      //定义一个变量 e，初始化默认为 5.
    unsigned char f=5;      //定义一个变量 f，初始化默认为 5.

    unsigned char x=3;      //定义一个变量 x，初始化默认为 3.
    unsigned char y=6;      //定义一个变量 y，初始化默认为 6.
```

```
    unsigned char k=2;     //定义一个变量 k，初始化默认为 2.

    //第 1 个知识点：连乘。
    a=2*5*3;               //乘数全部是常量。a 的结果为 30。
    b=k*x*y;               //乘数全部是变量。b 的结果为 36。
    c=x*5*y;               //乘数，有的是常量，有的是变量。c 的结果为 90。

    //第 2 个知识点：自乘的简写。
    d*=6;                  //相当于 d=d*6;最后 d 的结果为 30。
    e*=x;                  //相当于 e=e*x;最后 e 的结果为 15。
    f*=2*y*k;              //相当于 f=f*(2*y*k);最后 f 的结果为 120。

    View(a);               //把第 1 个数 a 发送到电脑端的串口助手软件上观察。
    View(b);               //把第 2 个数 b 发送到电脑端的串口助手软件上观察。
    View(c);               //把第 3 个数 c 发送到电脑端的串口助手软件上观察。
    View(d);               //把第 4 个数 d 发送到电脑端的串口助手软件上观察。
    View(e);               //把第 5 个数 e 发送到电脑端的串口助手软件上观察。
    View(f);               //把第 6 个数 f 发送到电脑端的串口助手软件上观察。

    while(1)
    {
    }
}

/*---C 语言学习区域的结束。-----------------------------------------------*/
```

在电脑串口助手软件上观察到的程序执行现象如下：

```
开始...

第 1 个数
十进制:30
十六进制:1E
二进制:11110

第 2 个数
十进制:36
十六进制:24
二进制:100100

第 3 个数
十进制:90
十六进制:5A
二进制:1011010
```

```
第 4 个数
十进制:30
十六进制:1E
二进制:11110

第 5 个数
十进制:15
十六进制:F
二进制:1111

第 6 个数
十进制:120
十六进制:78
二进制:1111000
```

分析：

通过实验结果，发现在单片机上的计算结果和我们的分析是一致的。

【26.6　如何在单片机上练习本章节 C 语言程序？】

直接复制前面章节中第十一节的模板程序，练习代码时只需要更改“C 语言学习区域”的代码就可以了，其它部分的代码不要动。编译后，把程序下载进带串口的 51 学习板，通过电脑端的串口助手软件就可以观察到不同的变量数值，详细方法请看第十一节内容。