第二十八节：整除求余。

【28.1　整除求余“%”。】

上一节讲到，求商求余都是属于整除运算，区别是：求商返回商，求余返回余，求商是“/”，求余是“%”。求余的运算符号恰好就是我们平时常用的百分号“%”，之所以选择百分号作为求余的运算符号，我猜测是因为，在小于100%的数据中，如果我们仔细回味一下百分号的分子与分母的关系，其实就隐含了一层淡淡的求余的味道。

整除求余的通用格式：

```
“保存变量”=“被除数”% “除数1” % “除数2”...% “除数N”;
```

跟之前讲的加减运算一样，赋值符号“=”左边的“保存变量”必须是变量，右边的可以是变量和常量的任意组合。如果右边只有两个参与运算的数据，就是整除求余的常见格式。

整除求余的常见格式：

```
“保存变量”=“被除数” % “除数” ;
```

现在深入分析一下整除求余的运算规律。

（1）当除数等于0时。

我们都知道，数学运算除数是不允许等于0的，如果在单片机中非要让除数为0，余数会出现什么结果？我在keil的C51编译环境试过，发现有一个规律：如果除数是变量的0，那么余数等于被除数。如果除数是常量的0，那么余数等于1。还有一种特殊的情况是编译不通过的，这种情况是“当被除数是变量，而除数是常量的0”。比如：

```
unsigned char a;
unsigned char b;
unsigned char k=10;
unsigned char y=0; //除数初始化为0

a=23%y;  //除数变量y里面是0，a的结果等于被除数23。
b=23%0;  //除数是常量0，b的结果是1。
b=k%0;   //这种特殊情况编译不通过：被除数是变量，而除数是常量的0。
```

平时做项目要尽量避免“除数是0”的情况，离它越远越好，但是既然除数不能为0，为什么我非要做“除数为0”时的实验呢？意义何在？这个实验的意义是，虽然我知道除数为0时会出错，但是我不知道这个错到底严不严重，会不会导致整个程序崩溃，当我做了这个实验后，我心中的石头才放下了，万一除数为0时，最多只是运算出错，但是不至于整个程序会崩溃，这样我心里就有了一个底，当哪天我某个程序崩溃跑飞时，我至少可以排除了“除数为0”这种情况，引导我从其它方面去找bug。

（2）当被除数小于除数时。余数等于被除数本身。比如：

```
unsigned char c;
c=7%10;  //c的结果是7。
```

（3）当被除数等于除数时。余数等于0。比如：

```
unsigned char d;
d=10%10;  //d的结果是0。
```

（4）当被除数大于除数时。余数必然小于除数。比如：

```
unsigned char e;
unsigned char f;
e=10%4;  //e 的结果是 2。
f=10%3;  //f 的结果是 1。
```

（5）当除数等于 1 时。余数必然等于 0。

```
unsigned char g;
g=7%1;  //g 的结果是 0。
```

【28.2　整除求余的自除简写。】

当被除数是“保存变量”时，存在自除求余的简写。

```
“保存变量”=“保存变量” %  “除数” ;
```

上述自除求余的简写如下：

```
“保存变量” % =“除数” ;
```

比如：

```
unsigned char h=9;
h%=5;  //相当于 h=h%5; 最后余数的计算结果是 4。
```

【28.3　整除求余有没有“自除 1”的特殊写法？】

加减法有自加 1“++g”和自减 1“g--”的特殊写法，但是求余的除法不存在这种自除 1 的特殊写法，因为任何一个数除以 1 的余数必然等于 0，所以求余的自除 1 没有任何意义，因此 C 语言语法中没有这种特殊写法。

【28.4　整除求余的溢出。】

不管是求商还是求余，除法的溢出规律跟加法的溢出规律是一样的，所以不再多举例子。在实际项目中，为了避免一不小心就溢出的问题，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。

【28.5　例程练习和分析。】

现在编写一个程序来验证刚才讲到的整除求余：

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;
```

```
    unsigned char e;
    unsigned char f;
    unsigned char g;
    unsigned char h=9;  //初始化为9。

    unsigned char k=10;  //初始化为10。
    unsigned char y=0; //除数变量初始化为0。

    //（1）当除数等于0时。
    a=23%y;
    b=23%0;
  //  b=k%0;  //这种特殊情况编译不通过：“被除数”是变量，而“除数”是常量的0。

    //（2）当被除数小于除数时。
    c=7%10;

    //（3）当被除数等于除数时。
    d=10%10;

    //（4）当被除数大于除数时。
    e=10%4;
    f=10%3;

    //（5）当除数等于1时。
    g=7%1;

    //（6）自除求余的简写。
    h%=5;  //相当于h=h%5;

    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。
    View(g);              //把第7个数g发送到电脑端的串口助手软件上观察。
    View(h);              //把第8个数h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:23
十六进制:17
二进制:10111

第 2 个数
十进制:1
十六进制:1
二进制:1

第 3 个数
十进制:7
十六进制:7
二进制:111

第 4 个数
十进制:0
十六进制:0
二进制:0

第 5 个数
十进制:2
十六进制:2
二进制:10

第 6 个数
十进制:1
十六进制:1
二进制:1

第 7 个数
十进制:0
十六进制:0
二进制:0

第 8 个数
十进制:4
十六进制:4
二进制:100
```

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。

【28.6　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。