第五十八节： const(或 code)在定义数据时的作用。

【58.1　const 与 code 的关系。】

const 与 code 都是语法的修饰关键词，放在所定义的数据前面时有“不可更改”之意。在 C 语言语法中，const 像普通话全国通用，是标准的语言；而 code 像地方的方言，仅仅适合针对 51 单片机的 C51 编译器环境。而其它大多数单片机的 C 编译器并不支持 code，只支持 const。比如 PIC，stm32 等单片机的 C 编译器都是只认 const 而不认 code 的。通常情况下，const 定义的数据都是放在 ROM 的，但是 51 单片机的 C51 编译器是例外，它并不把 const 定义的数据放在 ROM 区，只有用 code 关键词时它才会把数据放在 ROM 区，这一点相对其它大多数的单片机来说是不一样的。因为本教程是用 51 单片机的 C51 编译器，所以用 code 来替代 const。本节教程所提到的 const，在实际编程时都用 code 来替代。

【58.2　const（或 code）在定义数据时的终极目的。】

在数据定义分配的应用中，const 的终极目的是为了节省 RAM 的开销。从“读”和“写”的角度分析，数据有两种：“能读能写”和“只能读”这两种。 “能读能写”的数据占用 RAM 内存，叫变量，C 语言语法上定义此类数据时“无”const 前缀。 “只能读”的数据占用 ROM 内存，叫常量， C 语言语法上定义此类数据时“有”const 前缀。单片机的 ROM 容量比 RAM 容量往往大几十倍甚至上百倍，相比之下，RAM 的资源显得比较稀缺。因此，把某些只需“读”而不需“写”的数据定义成 const 放在 ROM，就可以节省 RAM 的开销。

【58.3　const（或 code）的应用场合。】

const 可以定义单个常量，也可以定义常量数组。定义单个常量时，通常应用在某个出现在程序多处并且需要经常调整的“阀值”参数，方便“一键更改”的操作。所谓“一键更改”，就是只要改一次 const 所定义初始化的某个常量，整个程序多次出现的此常量就自动更改了。定义常量数组时，通常应用在某个数据转换表，把某些固定的常量预先放到常量数组，通过数组的下标来“查表”。

【58.4　const（或 code）的语法格式。】

定义单个常量和常量数组时的语法是以下这个样子的：

```
const unsigned char x=10;  //定义单个常量。加了 const。
const unsigned char y[12]={31,28,31,30,31,30,31,31,30,31,30,31}; //定义常量数组。加了 const。
```

【58.5　const（或 code）的“能读”和“不可写”概念】

所谓“读”和“写”的能力，其实就是看某数能在赋值符号“=”的“右边”还是“左边”的能力。普通的变量，既可以在赋值符号“=”的“右边”（能读），也可以在赋值符号“=”的“左边”（能写）。比如，下面的写法是合法的：

```
unsigned char k=1;  //这是普通的变量，无 const 前缀。
unsigned char n=2;  //这是普通的变量，无 const 前缀。
n=k; //k 出现在赋值符号“=”的右边，表示能读。合法。
k=n; //k 出现在赋值符号“=”的左边，表示能写，可更改之意。合法。
```

但是如果一旦在普通的变量前面加了 const(或 code)关键词，就会发生“化学变化”，原来的“变量”就变成了“常量”，常量只能“读”，不能“写”。比如：

```
const unsigned char c=1;  //这是常量，有 const 前缀。
unsigned char n=2;  //这是普通的变量，无 const 前缀。
n=c; //c 是常量，能读，这是合法的。这行代码是语法正确的。
c=n; //c 是常量，不能写，这是非法的，C 编译器不通过。这行代码是语法错误的。
```

【58.6　const（或 code）能在函数内部定义吗？】

const(或 code)能在函数内部定义吗？能。语法是允许的。当在函数内部定义数据成 const(或者 code)，在数据的存储结构上，数据也是放在 ROM 区的（实际上在 51 单片机里想把数据放在 ROM 只能用 code 而不能用 const），把数据定义在函数内部，就只能在这个函数里面用，不能被其它函数调用。在作用域的问题上，const(或者 code)的常量数据跟其它变量的数据是一样的。比如：

```
void HanShu(void)
{
    const unsigned char c=1;  //在函数内部定义的 const 常量也是放在 ROM 区存储。
    unsigned char n=2;
    n=c; //c 是常量，在函数内部定义，只能在当前这个 HanShu 函数里调用。
}
```

【58.7　例程练习和分析。】

本教程使用的是 51 单片机的 C51 编译器，编写程序时为了让常量数据真正存储在 ROM 区，因此，本教程的程序例子都是用 code 替代 const。

本例程讲两个例子，一个是单个常量，一个是常量数组。

（1）单个常量。举的例子是“阀值”的“一键更改”应用。根据考试的分数，分两个等级。凡是大于或者等于 90 分的就是“优”，串口助手输出显示“1”。凡是小于 90 分的就是“良”，串口助手输出显示“0”。这里的“90 分”就是我所说的“阀值”概念，只要用一个 const 定义一个常量数据来替代“90”，当需要调整“阀值”时，只要更改一次此定义的常量数值就可以达到“一键更改”之目的。

（2）常量数组。举的例子是，查询 2017 年 12 个月的某个月的总天数，用两种思路实现，一种是 switch 分支语句来实现，另一种是 const 常量数组的“查表”思路来实现。通过对比这两种思路，你会发现 const 常量数组在做“转换表”这类“查表”项目时的强大优越性。

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

//函数的声明。
unsigned char HanShu_switch(unsigned char u8Month);
unsigned char HanShu_const(unsigned char u8Month);

//数据的定义。
code unsigned char Cu8Level=90;  //需要调整“阀值”时，只需更改一次这里的“90”这个数值。
code unsigned char Cu8MonthBuffer[12]= //每个月对应的天数。从数组下标 0 开始，0 代表 1 月...
{31,28,31,30,31,30,31,31,30,31,30,31};
```

```
unsigned char a; //用来接收函数返回的结果。
unsigned char b;
unsigned char c;
unsigned char d;

//函数的定义。
unsigned char HanShu_switch(unsigned char u8Month)  //用 switch 分支来实现。
{
   switch(u8Month)
   {
      case 1:   //1 月份的天数
           return 31;
      case 2:   //2 月份的天数
           return 28;
      case 3:   //3 月份的天数
           return 31;
      case 4:   //4 月份的天数
           return 30;
      case 5:   //5 月份的天数
           return 31;
      case 6:   //6 月份的天数
           return 30;
      case 7:   //7 月份的天数
           return 31;
      case 8:   //8 月份的天数
           return 31;
      case 9:   //9 月份的天数
           return 30;
      case 10:  //10 月份的天数
           return 31;
      case 11:  //11 月份的天数
           return 30;
      case 12:  //12 月份的天数
           return 31;
      default:  //万一输错了其它范围的月份，就默认返回 30 天。
          return 30;
   }
}

unsigned char HanShu_const(unsigned char u8Month) //用 const 常量数组的“查表”来实现。
{
   unsigned char u8GetDays;
   u8Month=u8Month-1;  //因为数组下标是从 0 开始，0 代表 1 月份，1 代表 2 月份。所以减去 1。
```

```
    u8GetDays=Cu8MonthBuffer[u8Month]; //这就是查表，马上获取常量数组表格里固定对应的天数。
    return u8GetDays;
}

void main() //主函数
{
    //第（1）个例子
    if(89>=Cu8Level)  //大于或者等于阀值，就输出 1。
    {
       a=1;
    }
    else  //否则输出 0。
    {
       a=0;
    }

    if(95>=Cu8Level)  //大于或者等于阀值，就输出 1。
    {
       b=1;
    }
    else  //否则输出 0。
    {
       b=0;
    }

    //第（2）个例子
    c=HanShu_switch(2);  //用 switch 分支的函数获取 2 月份的总天数。
    d=HanShu_const(2);   //用 const 常量数组“查表”的函数获取 2 月份的总天数。

    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。

    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:0
十六进制:0
二进制:0

第 2 个数
十进制:1
十六进制:1
二进制:1

第 3 个数
十进制:28
十六进制:1C
二进制:11100

第 4 个数
十进制:28
十六进制:1C
二进制:11100
```

分析：

a 为 0。

b 为 1。

c 为 28。

d 为 28。

【58.8　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。