第七十节：“万能数组”的结构体。

【70.1　结构体与数组。】

结构体是数组，但不是普通的数组，而是一种“万能数组”。普通数组，是依靠严格的数组下标（类似编号）来识别某个具体单元的（或者称“寻址”），期间，如果要往数组插入或者删除某些单元，后面所有单元的下标编号都会发生改变，牵一发而动全身，后面其它单元的下标序号自动重新排列，原来某个特定的单元的下标发生了改变，也就意味着“名字”发生了改变，这种情况在编写程序的时候，就意味着很多代码需要随着更改调整，给程序员带来很多不便。怎么办？结构体此时横空出世，扭转了这种“不便”的局面。之所以称结构体为“万能数组”，是因为结构体内部没有“下标编号”，只有名字。结构体与普通数组的本质区别是，结构体是靠“名字”来寻址的，不管你往结构体里插入或者删除某些单元，其它单元的“名字”不会发生改变，隔离效果好，左邻右舍不会受影响。除此之外，结构体内部的成员变量是允许出现不同的数据类型的，比如 unsigned char,unsigned int,unsigned long 这三种数据类型的变量都可以往同一个结构体里面“填充”，不受类型的局限，真正做到“万能”级。而普通数组就没有这个优越性，普通数组要么清一色都是 unsigned char，要么清一色都是 unsigned int，要么清一色都是 unsigned long，不能像结构体这么“混合型”的。结构体的这种优越性，在大型程序的升级和维护时体现得非常明显。

【70.2　“造模”和“生成”和“调用”。】

结构体的使用，有三道标准工序“造模”和“生成”和“调用”。塑胶外壳，必须先开模具（造模），然后再用模具印出外壳（生成），再把外壳应用于日常生活中（调用）。结构体也一样，先“造”结构体的“模”（造模），再根据这个“模”来“生成”一个结构体变量（生成），然后在某函数里使用此变量（调用）。例子如下：

```
struct StructMould     //“造模”
{
   unsigned char  u8Data_A;
   unsigned int   u16Data_B;
   unsigned long  u32Data_C;
};

struct StructMould  GtMould;  //“生成”一个变量 GtMould。

void main()
{
    GtMould.u8Data_A=1;      //依靠成员的“名字”来“调用”
    GtMould.u16Data_B=2;     //依靠成员的“名字”来“调用”
    GtMould.u32Data_C=3;     //依靠成员的“名字”来“调用”

    while(1)
    {

    }
}
```

把上述程序转换成“普通数组”和“指针”的形式，给大家一个直观的对比，代码如下：

```
unsigned char Gu8MouldBuffer[7];  //相当于结构体变量 GtMould

unsigned char *pu8Data_A;
unsigned int  *pu16Data_B;
unsigned long *pu32Data_C;

void main()
{
    pu8Data_A=(unsigned char *)&Gu8MouldBuffer[0];  //依靠数组的下标[0]来“调用”
    *pu8Data_A=1;

    pu16Data_B=(unsigned int *)&Gu8MouldBuffer[1];  //依靠数组的下标[1]来“调用”
    *pu16Data_B=2;

    pu32Data_C=(unsigned long *)&Gu8MouldBuffer[3]; //依靠数组的下标[3]来“调用”
    *pu32Data_C=3;

    while(1)
    {

    }
}
```

分析：上述两种代码，目标都是把 1,2,3 这三个数字存放在一个数组里。第一种用结构体的方式，第二种用普通数组的方式。

【70.3　例程练习和分析。】

现在编写一个练习的程序：

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

   struct StructMould     //“造模”
   {
      unsigned char  u8Data_A;
      unsigned int   u16Data_B;
      unsigned long  u32Data_C;
   };

   struct StructMould  GtMould;  //“生成”一个变量 GtMould。
```

```
void main() //主函数
{
     GtMould.u8Data_A=1;          //依靠成员的“名字”来“调用”
     GtMould.u16Data_B=2;         //依靠成员的“名字”来“调用”
     GtMould.u32Data_C=3;         //依靠成员的“名字”来“调用”

     View(GtMould.u8Data_A);      //把结构体成员 GtMould.u8Data_A 发送到电脑端观察
     View(GtMould.u16Data_B);     //把结构体成员 GtMould.u16Data_B 发送到电脑端观察
     View(GtMould.u32Data_C);     //把结构体成员 GtMould.u32Data_C 发送到电脑端观察

    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:1
十六进制:1
二进制:1

第 2 个数
十进制:2
十六进制:2
二进制:10

第 3 个数
十进制:3
十六进制:3
二进制:11
```

分析：

GtMould.u8Data_A 为 1。

GtMould.u16Data_B 为 2。

GtMould.u32Data_C 为 3。

【70.4　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。