

第七十二节： 结构体的指针。

【72.1 结构体指针的重要用途。】

结构体指针有两个重要用途，一个是结构体数据的拆分和打包，另一个是作为结构体数据在涉及函数时的参数入口。

什么是“结构体数据的拆分和打包”？结构体本质是一个数组，数组内可能包含了许多不同数据长度类型的成员，当我们直接操作某个具体的成员时，只改变某个成员的数值，不影响其它成员，这个就是“拆分”的角度。那么，什么是“打包”？当涉及整个结构体数据的存储或者传输（通信）给另外一个单片机时，这时候有两种选择，一种是一个成员一个成员的挨个处理，这种“拆分”的处理方式比较繁琐麻烦，另外一种就是把整个结构体当作一个以字节为单位的整体数组来处理，这种处理方式就是高速便捷的“打包”处理，但是关键的问题来了，我们把整个结构体数据以字节的方式“打包”传递给另外一个单片机，但是这个单片机接收到我们一组数据后，如何把这“一包”以字节为单位的数组转换成相同的结构体变量，以便在它的程序处理中也能以“拆分”的角度直接处理某个具体的成员变量，这时就涉及到结构体指针的作用。

什么是“作为结构体数据在涉及函数时的参数入口”？结构体数据一般内部包含了很多成员，当要把这一包数据传递给某个函数内部时，这个函数要给结构体数据预留参数入口，这时，如果函数以结构体成员的角度来预留入口，那么有多少个成员就要预留多少个函数的参数入口，可阅读性非常差，操作起来也麻烦。但是，如果以指针的角度来预留入口，那么不管这个结构体内部包含多少个成员，只需要预留一个指针参数入口就够用了，这就是绝大多 32 单片机在写库函数时都采用结构体指针作为函数的参数入口的原因。

结构体指针这两个重要用途后续章节会深入讲解，本节的重点是先让大家学会结构体指针的基础知识，为后续章节做准备。

【72.2 结构体指针的基础。】

操作结构体内部某个具体变量时，有两种方式，一种是成员调用的方式，另一种是指针调用的方式。C 语言语法为了区分这两种方式，专门设计了两种不同的操作符号。成员调用方式采用小数点“.”的符号，指针调用方式采用箭头“->”的符号。例子如下：

```
struct StructMould_1
{
    unsigned char  u8Data_A;
    unsigned long  u32Data_B;
};

struct StructMould_1  GtMould_1;  // “生成” 一个变量。    // 占用 5 个字节。
struct StructMould_1  *ptMould_1;  // 定义一个结构体指针。  // 占用 3 个字节。

void main() // 主函数
{
    GtMould_1.u8Data_A=5;    // “成员调用” 的方式，用小数点符号 “.”

    ptMould_1=&GtMould_1;    // ptMould_1 指针与变量 GtMould_1 建立关联。
    ptMould_1->u8Data_A=ptMould_1->u8Data_A+5; // “指针调用” 的方式，用箭头符号 “->”
```

```

        while(1)
        {
        }
    }

```

分析：上述例子中，信息量很大，知识点有两个。

第一个知识点：为什么结构体变量 GtMould_1 占用 5 个字节，而结构体指针 *ptMould_1 只占用 3 个字节？结构体变量 GtMould_1 所占的内存是由结构体成员内部的数量决定的，而结构体指针 *ptMould_1 是由 C 编译器根据芯片硬件寻址范围而决定的，在一个相同的 C 编译器系统中，所有类型的指针所占用的字节数都是一样的，比如在本教程中所用 8 位单片机的 C51 编译器系统中，unsigned char *, unsigned int *, unsigned long *, 以及本节的 struct StructMould_1 *, 都是占用 3 个字节（题外话，我前面第 60 节中所说的“凡是 32 位以下的单片机的指针都是占用 4 个字节”是有误的，抱歉）。32 位单片机的指针往往都是 4 个字节，而某些 64 位的 PC 机，指针可能是 8 个字节，这些内容大家只要有个大概的了解即可。

第二个知识点：结构体成员 GtMould_1.u8Data_A 经过第一步的“成员调用”直接赋值 5，紧接着经过“指针调用”的累加 5 操作，最后 GtMould_1.u8Data_A 的数值是 10（5+5）。

【72.3 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_1
{
    unsigned char  u8Data_A;
    unsigned long  u32Data_B;
};

struct StructMould_1  GtMould_1;  // “生成” 一个变量。    // 占用 5 个字节。
struct StructMould_1  *ptMould_1;  // 定义一个结构体指针。  // 占用 3 个字节。

void main() //主函数
{
    GtMould_1.u8Data_A=5;    // “成员调用” 的方式，用小数点符号 “.”

    ptMould_1=&GtMould_1;    // ptMould_1 指针与变量 GtMould_1 建立关联。
    ptMould_1->u8Data_A=ptMould_1->u8Data_A+5; // “指针调用” 的方式，用箭头符号 “->”

    View(sizeof(GtMould_1));    // 在电脑端观察变量 GtMould_1 占用多少个字节。
    View(sizeof(ptMould_1));    // 在电脑端观察指针 ptMould_1 占用多少个字节。
    View(GtMould_1.u8Data_A);    // 在电脑端观察结构体成员 GtMould_1.u8Data_A 的最后数值。
    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:5  
十六进制:5  
二进制:101  
  
第 2 个数  
十进制:3  
十六进制:3  
二进制:11  
  
第 3 个数  
十进制:10  
十六进制:A  
二进制:1010
```

分析：

变量 GtMould_1 占用 5 个字节。

指针 ptMould_1 占用 3 个字节。

结构体成员 GtMould_1.u8Data_A 的最后数值是 10。

【72.4 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。