第七十六节： 二维数组的指针。

【76.1 二维数组指针的用途。】

前面章节讲了一维指针操作二维数组，本质是通过“类型强制转换”实现的，这种应用局限于某些特定的场合，毕竟一维有 1 个下标，二维有 2 个下标，一维和二维在队形感上是有明显差别的，强行用一维指针操作二维数组会破坏了代码原有的队形感，大多数的情况，还是用二维指针操作二维数组。

二维指针主要应用在两个方面，一方面是 N 个二维数组的“中转站”应用，另一方面是函数接口的应用。比如，当某项目有 N 个二维数组表格时，要通过某个变量来切换处理某个特定的表格，以便实现“N 选一”的功能，此时，二维指针在这 N 个二维数组之间就起到中转站的作用。又，当某个函数接口想输入或者输出一个二维数组时，就必然要用到二维指针作为函数的接口参数。

【76.2 二维指针的“中转站”应用。】

举一个例子，有 3 个现有的二维数组，通过某个变量来选择切换，把某个二维数组的数据复制到指定的一个缓存数组中。

```
code unsigned char table_1[3][3]=  //第 1 个现有的二维数组
{
{0x00,0x01,0x02},
{0x10,0x11,0x12},
{0x20,0x21,0x22},
};

code unsigned char table_2[3][3]=  //第 2 个现有的二维数组
{
{0xA0,0xA1,0xA2},
{0xB0,0xB1,0xB2},
{0xC0,0xC1,0xC2},
};

code unsigned char table_3[3][3]=  //第 3 个现有的二维数组
{
{0xD0,0xD1,0xD2},
{0xE0,0xE1,0xE2},
{0xF0,0xF1,0xF2},
};

unsigned char SaveBuffer[3][3];  //指定的一个缓存数组

unsigned char TableSec;  //选择变量
const unsigned char (*pTable)[3];  //“中转站”的二维指针
unsigned char R,L;  //复制数据时用到的 for 循环变量
void main()
```

```
{
    TableSec=2; //选择第 2 个现有的二维数组
    switch(TableSec)  //根据选择变量来切换选择某个现有的二维数组
    {
       case 1:  //选择第 1 个现有二维数组
          pTable=table_1;  //二维指针 pTable 在这里关联指定的数组，起到中转站的作用。
          break;
       case 2:   //选择第 2 个现有二维数组
          pTable=table_2;  //二维指针 pTable 在这里关联指定的数组，起到中转站的作用。
          break;
       case 3:    //选择第 3 个现有二维数组
          pTable=table_2;  //二维指针 pTable 在这里关联指定的数组，起到中转站的作用。
          break;
    }

    //通过二维指针 pTable 来复制数据到指定的缓存数组 SaveBuffer
    for(R=0;R<3;R++)  //行循环
    {
        for(L=0;L<3;L++)  //列循环
        {
            SaveBuffer[R][L]=pTable[R][L]; //这里能看到，二维指针维护了二维数组的队形感
        }
    }

    while(1)
    {

    }
}
```

【76.3　二维指针在“函数接口”中的应用。】

把上述例子“复制过程”的代码封装成一个函数，实现的功能还是一样，有 3 个现有的二维数组，通过某个变量来选择切换，把某个二维数组的数据复制到指定的一个缓存数组中。

```
//函数声明
void CopyBuffer(const unsigned char (*pTable)[3],unsigned char (*pSaveBuffer)[3]);

code unsigned char table_1[3][3]=  //第 1 个现有的二维数组
{
{0x00,0x01,0x02},
{0x10,0x11,0x12},
{0x20,0x21,0x22},
```

```
};

code unsigned char table_2[3][3]=  //第 2 个现有的二维数组
{
{0xA0,0xA1,0xA2},
{0xB0,0xB1,0xB2},
{0xC0,0xC1,0xC2},
};

code unsigned char table_3[3][3]=  //第 3 个现有的二维数组
{
{0xD0,0xD1,0xD2},
{0xE0,0xE1,0xE2},
{0xF0,0xF1,0xF2},
};

unsigned char SaveBuffer[3][3];  //指定的一个缓存数组

unsigned char TableSec;  //选择变量

//*pTable 是输入接口带 const 修饰，*pSaveBuffer 是输出结果的接口无 const。
void CopyBuffer(const unsigned char (*pTable)[3],unsigned char (*pSaveBuffer)[3])
{
    unsigned char R,L;  //复制数据时用到的 for 循环变量

    for(R=0;R<3;R++)  //行循环
    {
        for(L=0;L<3;L++)  //列循环
        {
            pSaveBuffer[R][L]=pTable[R][L]; //这里能看到，二维指针维护了二维数组的队形感
        }
    }

}
void main()
{
    TableSec=2; //选择第 2 个现有的二维数组
    switch(TableSec)  //根据选择变量来切换选择某个现有的二维数组
    {
       case 1:  //选择第 1 个现有二维数组
          CopyBuffer(table_1,SaveBuffer); //二维指针在这里分别体现了输入和输出接口作用
          break;
       case 2:   //选择第 2 个现有二维数组
          CopyBuffer(table_2,SaveBuffer); //二维指针在这里分别体现了输入和输出接口作用
```

```
            break;
        case 3:    //选择第 3 个现有二维数组
            CopyBuffer(table_3,SaveBuffer); //二维指针在这里分别体现了输入和输出接口作用
            break;
    }
    while(1)
    {

    }
}
```

【76.4　二维指针“类型强制转换”的书写格式。】

unsigned char *pu8,unsigned int *pu16,unsigned int *pu32 这些指针的书写定义都是很有规则感的，相比之下，二维指针的定义显得缺乏规则感，比如定义的二维指针变量 unsigned char (*pTable)[3]，不规则在哪？就在于二维指针的变量 pTable 嵌入到了括号中去，跟符号“*”捆绑在一起，这时就会冒出一个问题，如果我要强制某个指针变量为二维指针怎么办？下面的例子已经给出了答案。

```
unsigned char table[3][3]=  //二维数组
{
{0xD0,0xD1,0xD2},
{0xE0,0xE1,0xE2},
{0xF0,0xF1,0xF2},
};

unsigned char (*pTable)[3];

void main()
{
    pTable=(unsigned char (*)[3])table;  //这里，强制类型转换用 unsigned char (*)[3]
}
```

总结：二维数组的强制类型转换用这种书写格式（unsigned char (*)[N]）,这里的 N 是代表实际项目中某数组的“列”数。

【76.5　例程练习和分析。】

现在编写一个练习程序。

```
/*---C 语言学习区域的开始。-----------------------------------------------*/
void CopyBuffer(const unsigned char (*pTable)[3],unsigned char (*pSaveBuffer)[3]);

code unsigned char table_1[3][3]=  //第 1 个现有的二维数组
{
{0x00,0x01,0x02},
```

```
{0x10,0x11,0x12},
{0x20,0x21,0x22},
};

code unsigned char table_2[3][3]=  //第 2 个现有的二维数组
{
{0xA0,0xA1,0xA2},
{0xB0,0xB1,0xB2},
{0xC0,0xC1,0xC2},
};

code unsigned char table_3[3][3]=  //第 3 个现有的二维数组
{
{0xD0,0xD1,0xD2},
{0xE0,0xE1,0xE2},
{0xF0,0xF1,0xF2},
};

unsigned char SaveBuffer[3][3];  //指定的一个缓存数组

unsigned char TableSec;  //选择变量

//*pTable 是输入接口带 const 修饰，*pSaveBuffer 是输出结果的接口无 const。
void CopyBuffer(const unsigned char (*pTable)[3],unsigned char (*pSaveBuffer)[3])
{
    unsigned char R,L;  //复制数据时用到的 for 循环变量

    for(R=0;R<3;R++)  //行循环
    {
        for(L=0;L<3;L++)  //列循环
        {
            pSaveBuffer[R][L]=pTable[R][L]; //这里能看到，二维指针维护了二维数组的队形感
        }
    }

}

void main() //主函数
{
   TableSec=2; //选择第 2 个现有的二维数组
    switch(TableSec)  //根据选择变量来切换选择某个现有的二维数组
    {
       case 1:  //选择第 1 个现有二维数组
          CopyBuffer(table_1,SaveBuffer); //二维指针在这里分别体现了输入和输出接口作用
```

```
            break;
        case 2:   //选择第 2 个现有二维数组
            CopyBuffer(table_2,SaveBuffer); //二维指针在这里分别体现了输入和输出接口作用
            break;
        case 3:    //选择第 3 个现有二维数组
            CopyBuffer(table_3,SaveBuffer); //二维指针在这里分别体现了输入和输出接口作用
            break;
    }

    View(SaveBuffer[0][0]);  //在电脑端观察某个二维数组第 0 行数据第 0 个元素的内容
    View(SaveBuffer[0][1]);  //在电脑端观察某个二维数组第 0 行数据第 1 个元素的内容
    View(SaveBuffer[0][2]);  //在电脑端观察某个二维数组第 0 行数据第 2 个元素的内容
    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:160
十六进制:A0
二进制:10100000

第 2 个数
十进制:161
十六进制:A1
二进制:10100001

第 3 个数
十进制:162
十六进制:A2
二进制:10100010
```

分析：

SaveBuffer[0][0]是十六进制的 0xA0，提取了第 2 个二维数组的第 0 行第 0 个数据。

SaveBuffer[0][1]是十六进制的 0xA1，提取了第 2 个二维数组的第 0 行第 1 个数据。

SaveBuffer[0][2]是十六进制的 0xA2，提取了第 2 个二维数组的第 0 行第 2 个数据。

【76.6　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。