第七十五节： 指针的名义（例：一维指针操作二维数组）。

【75.1　指针的名义。】

刚开始接触指针往往有这种感觉，指针的江湖很乱，什么“乱七八糟”的指针都能冒出来，空指针，指针的指针，函数的指针，各种名目繁多的指针，似乎都可以打着指针的名义让你招架不住，而随着我们功力的提升，会逐渐拨开云雾，发现指针的真谛不外乎三个，第一个是所有的指针所占用字节数都一样，第二个是所有指针的操作本质都是“取地址”，第三个是所有各种不同类型的指针之间的转换都可以用“小括号的类型强制转换”。

【75.2　一维指针操作二维数组。】

C 语言讲究门当户对，讲究类型匹配，什么类型的指针就操作什么类型的数据，否则 C 编译器在翻译代码的时候，会给予报错或者警告。如果想甩开因类型不匹配而导致的报错或者警告，就只能使用“小括号的类型强制转换”，这个方法在项目中应用很频繁，也很实用。一维指针想直接操作二维数组也是必须使用“小括号的类型强制转换”。实际项目中为什么会涉及“一维指针想直接操作二维数组”？二维数组更加像一个由行与列组合而成的表格，而且每行单元的内存地址是连续的，并且上下每行与每行之间的首尾单元的内存地址也是连续的，凡是内存地址连续的都是指针的菜。我曾遇到这样一种情况，要从一个二维表格里提取某一行数据用来显示，而这个显示函数是别人封装好的一个库函数，库函数对外的接口是一维指针，这样，如何把二维表格（二维数组）跟一维指针在接口上兼容起来，就是一个要面临的问题，这时有两种思路，一种是把二维数组的某一行数据先用原始的办法提取出来存放在一个中间变量的一维数组，然后再把这个一维数组代入到一维指针接口的库函数里，另一种思路是绕开中间变量，直接把二维数组的某一行的地址强制转换成一维指针的类型，利用“类型强制转换”绕开 C 编译器的报错或警告，实现二维数组跟一维指针“直通”，经过实验，这种方法果然可以，从此对指针的感悟就又上了一层，原来，指针的“取地址”是不仅仅局限于某个数组的首地址，它完全可以利用类型强制转换的小括号“()”与取地址符号“&”结合起来，让指针跟一维数组或者二维数组里面任何一个单元直接关联起来。请看下面两个例子，用一维指针提取二维数组里面某一行的数据，第一个例子是在程序处理中的类型强制转换的应用，第二个例子是在函数接口中的类型强制转换的应用。

【75.3　在程序处理中的类型转换。】

```
unsigned char table[][3]=  //二维数组
{
{0x00,0x01,0x02},  //二维数组的第 0 行数据
{0x10,0x11,0x12},  //二维数组的第 1 行数据
{0x20,0x21,0x22},  //二维数组的第 2 行数据
};

unsigned char *pGu8;    //一维指针
unsigned char  Gu8Buffer[3];    //一维数组，存放从二维数组里提取出来的某一行数据
unsigned char  i; // for 循环的变量
void main()
{
    pGu8=(unsigned char *)&table[2][0];  //利用类型强制转换使得一维指针跟二维数组关联起来。
```

```
    for(i=0;i<3;i++)
    {
        Gu8Buffer[i]=pGu8[i];   //提取二维数组的第 2 行数据，存入到一个一维数组里。
    }

    while(1)
    {

    }
}
```

【75.4　在函数接口中的类型转换。】

在函数接口中，也可以利用类型强制转换来实现函数接口的匹配问题，比如，下面这个写法也是合法的。

```
void GetRowData(unsigned char *pu8); //函数的声明

unsigned char table[][3]=  //二维数组
{
{0x00,0x01,0x02},  //二维数组的第 0 行数据
{0x10,0x11,0x12},  //二维数组的第 1 行数据
{0x20,0x21,0x22},  //二维数组的第 2 行数据
};

unsigned char  Gu8Buffer[3];    //一维数组，存放从二维数组里提取出来的某一行数据

void GetRowData(unsigned char *pu8)  //一维指针的函数接口
{
    unsigned char  i; // for 循环的变量
    for(i=0;i<3;i++)
    {
        Gu8Buffer[i]=pu8[i];   //提取二维数组的某行数据，存入到一个一维数组里。
    }
}

void main()
{
    GetRowData((unsigned char *)&table[2][0]); //利用类型强制转换来兼容一维指针的函数接口

    while(1)
    {

    }
```

```
}
```

【75.5　注意指针或者数组越界的问题。】

上述例子中，二维数组内部只有 9 个数据，如果指针操作的数据超过了这 9 个数据的地址范围，就会导致系统其它无辜的数据受到破坏，这个问题导致的后果是很严重的，这类指针或者数组越界的问题，大家平时做项目时必须留心注意。

【75.6　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 GetRowData(unsigned char *pu8); //函数的声明

    unsigned char table[][3]=  //二维数组
    {
    {0x00,0x01,0x02},  //二维数组的第 0 行数据
    {0x10,0x11,0x12},  //二维数组的第 1 行数据
    {0x20,0x21,0x22},  //二维数组的第 2 行数据
    };

    unsigned char  Gu8Buffer[3];    //一维数组，存放从二维数组里提取出来的某一行数据

    void GetRowData(unsigned char *pu8)  //一维指针的函数接口
    {
        unsigned char  i; // for 循环的变量
        for(i=0;i<3;i++)
        {
            Gu8Buffer[i]=pu8[i];   //提取二维数组的某行数据，存入到一个一维数组里。
        }
    }

    void main() //主函数
    {
        GetRowData((unsigned char *)&table[2][0]); //利用类型强制转换来兼容一维指针的函数接口

        View(Gu8Buffer[0]);  //在电脑端观察存放二维数组某行数据的一维数组的内容
        View(Gu8Buffer[1]);  //在电脑端观察存放二维数组某行数据的一维数组的内容
        View(Gu8Buffer[2]);  //在电脑端观察存放二维数组某行数据的一维数组的内容
        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:32
十六进制:20
二进制:100000

第 2 个数
十进制:33
十六进制:21
二进制:100001

第 3 个数
十进制:34
十六进制:22
二进制:100010
```

分析：

Gu8Buffer[0]是十六进制的 0x20，提取了二维数组第 2 行中的某数据。

Gu8Buffer[1]是十六进制的 0x21，提取了二维数组第 2 行中的某数据。

Gu8Buffer[2]是十六进制的 0x22，提取了二维数组第 2 行中的某数据。

【75.7　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。