第四十六节： 一维数组。

【46.1 数组是什么？】

数组就是一堆变量或常量的集合。把一个数组里面某一个变量或者常量称为数组的元素，反过来也可以这么说，元素的集合就是数组。数组的最大特点就是内部所有的元素的地址都是挨家挨户相连的，同花顺似的，以第一个元素（下标是 0 的元素）为首地址，后来元素的地址挨个依次增大。首地址在 RAM 中的绝对地址往往是编译器自动分配的，我们不用管，可以看成是随机的。多说一句，在某些单片机，也可以通过特定的 C 语言关键词，强制要求编译器按我们的意愿，来分配到 RAM 中指定的某个绝对地址，这部分的内容这里暂时不讲。继续刚才的话题，首地址就像是一个坐标原点，一旦被编译器确定下来它在 RAM 中的地址，那么后面其它元素的地址都是在此基础上依次增大的，有规律的。正因为这个特点，数组在项目中往往起到缓存的作用。比如，在通信的项目中，用来作为一串数据的接收缓存。在界面显示的项目中，某个 16x16 点阵汉字的字模，需要一个内含 32 个元素的数组来作为缓存。在读写文件的项目中，也需要一个大数组来作为文件内容的缓存。在某些涉及复杂算法的项目，以数组作为缓存，并且通过配合循环语句或者指针，就可以快速批量的处理数据（循环语句和指针的相关知识后面章节会讲到）。总之，在项目应用中，数组无处不在。

数组分为一维数组，二维数组，三维数组。一维数组应用最广，二维数组其次，三维数组最少用。所以本教程只讲一维数组和二维数组，本节先讲一维数组。

【46.2 一维数组的书写格式和特点。】

一维数组不带初始化时候的定义格式如下：

```
数据类型 数组名[数组元素总数 N];
```

数据类型是指 unsigned char,unsigned int,unsigned long 这类关键词；数组名就是由字母和数字组合而成的字符串，遵循常用变量的命名规则；N 是数字，代表此数组内部有多少个元素。比如：

```
unsigned char x[3];  //这里的 3 是数组内部元素的总数，但不是下标。
```

上述这一行代码，就相当于一条语句定义了 3 个变量，这 3 个变量分别是 x[0],x[1],x[2]，但是不存在 x[3]这个变量。这里，具体元素中括号内的“0，1，2”称为数组的下标，代表某个具体的元素。由此可见，数组有“批量定义”的特点。同时也要发现，此数组定义的 N 是 3，代表内含 3 个元素变量，但是具体到某个元素的时候，下标不是从 1 开始，而是从 0 开始，最后一个也不是 3 而是 2。可以这样描述，某个数组有 N 个元素，它具体元素的下标是从 0 开始，到 N-1 结束。那么问题来，如果一个数组明明最大只有 N 个元素，但是我在操作某个具体的元素时，非要用下标 N 或者 N+1，也就是说，如果超过数组的范围的操作，会出现什么问题？后果严重吗？答案是：会导致数组越界出现异常或者编译不通过，可能会破坏其它数据，后果是严重的。因此大家使用数组的时候，要注意数组不能越界的问题。

刚刚讲了一维数组不带初始化的定义格式，现在接着讲带初始化的定义格式，如下：

```
数据类型 数组名[数组元素总数 N]={元素 0, 元素 1，…元素 N-1};
```

比如：

```
unsigned char y[3]={10,11,12};
```

此数组一行代码定义了 3 个变量，分别是 y[0], y[1], y[2]。而 y[0]初始化为 10，y[1]初始化为 11，y[2]初始化为 12。

在程序中，操作数组某个变量元素时，下标可以是常量，比如 y[0],此时的 0 就是常量；下标也可以是变量，比如 y[i]，此时的 i 就是变量。再强调一次，作为下标的常量或者变量 i 的数值必须小于数组定义时的元素个数，否则就会导致数组越界出现异常或者编译不通过。

中括号内的 N 什么时候是“数组的元素总数”，什么时候是“数组的元素下标”，这个问题对初学者很容易混淆。其实很简单，定义的时候是“数组的元素总数”，操作调用具体某个元素的时候是“数组的元素下标”。

【46.3　什么情况下可以省略定义的元素总数？】

一维数组在定义时，如果预先给它填写若干个初始化的数据，在语法上，也可以省略中括号[N]里面的元素总数 N，这样编译器在编译时会根据初始化的总数来自动识别和定义此一维数组实际元素总数，分配对应数量的内存 RAM。比如：

```
unsigned char  y[3]={10,11,12};  //没有省略元素总数的写法
```

跟

```
unsigned char  y[]={10,11,12};   //在初始化的情况下，省略了元素总数的写法。
```

的意义是一样的，都是合法的，都是 C 语言所允许的。注意，省略元素个数时必须要有初始化的数据，否则，编译器不知道此数组的长度，可能导致编译出错。

这个功能在实际应用中有什么作用呢？在实际应用中，此项功能一般会用在常量数组里，而不是变量的数组里。当在数组定义的前面加上“const”或者“code”(针对 51 单片机)的关键词时，原来“变量”的数组就会变成“常量”的数组，这时，如果把常量的数组用来作为某个转换表格，此功能就很实用。因为作为转换表格的常量数组，我们在编程程序的过程中，有可能随时往里面添加数组，这个时候，不用我们刻意去计算和调整数组的元素总数 N，给我们写程序带来了便利。对于这个功能的应用，大家先有一个感性的认识即可，暂时不用深入去了解，因为后续的章节还会讲解这方面的内容。

【46.4　例程练习和分析。】

现在编写一个程序来熟悉一下一维数组的使用。

程序代码如下：

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

     unsigned char  x[3];  //此处的 3 不是下标，而是元素总数，里面的 3 个变量没有初始化
     unsigned char  y[3]={10,11,12}; //里面三个元素变量 y[0],y[1],y[2]分别初始化为 10,11,12
     unsigned char  i=0; //定义和初始化一个变量。用来做 x 数组的下标。

void main() //主函数
{

     x[i]=25;  //此时下标 i 为 0.相当于把 25 赋值给 x[0]
     i=i+1;    //i 由 0 变成 1.
     x[i]=26;  //此时下标 i 为 1.相当于把 26 赋值给 x[1]
     i=i+1;    //i 由 1 变成 2.
     x[i]=27;  //此时下标 i 为 2.相当于把 27 赋值给 x[2]
     x[i]=x[i]+1; //此时 x[2]自加 1 变成了 28
```

```
    View(x[0]);  //把第 1 个数 x[0]发送到电脑端的串口助手软件上观察。
    View(x[1]);  //把第 2 个数 x[1]发送到电脑端的串口助手软件上观察。
    View(x[2]);  //把第 3 个数 x[2]发送到电脑端的串口助手软件上观察。
    View(y[0]);  //把第 4 个数 y[0]发送到电脑端的串口助手软件上观察。
    View(y[1]);  //把第 5 个数 y[1]发送到电脑端的串口助手软件上观察。
    View(y[2]);  //把第 6 个数 y[2]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:25
十六进制:19
二进制:11001

第 2 个数
十进制:26
十六进制:1A
二进制:11010

第 3 个数
十进制:28
十六进制:1C
二进制:11100

第 4 个数
十进制:10
十六进制:A
二进制:1010

第 5 个数
十进制:11
十六进制:B
二进制:1011

第 6 个数
十进制:12
```

```
十六进制:C
二进制:1100
```

分析：

变量元素 x[0]为 25。
变量元素 x[1]为 26。
变量元素 x[2]为 28。
变量元素 y[0]为 10。
变量元素 y[1]为 11。
变量元素 y[2]为 12。

【46.5　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。