第七十九节： 各种变量常量的命名规范。

【79.1　命名规范的必要。】

一个大型的项目程序，涉及到的变量常量非常多，各种变量常量眼花缭乱，名字不规范就无法轻松掌控全局。若能一开始就遵守特定的命名规范，则普天之下，率土之滨，都被你牢牢地掌控在手里，天下再也没有难维护的代码。本节教给大家的是我多年实践所沿用的命名规范和习惯，它不是唯一绝对的，只是给大家参考，大家今后也可以在自己的实践中慢慢总结出一套适合自己的命名规范和习惯。

【79.2　普通变量常量的命名规范和习惯。】

在 C51 编译器的平台下，unsigned char ,unsigned int ,unsigned long 三类常用的变量代表了“无符号的 8 位，16 位，32 位”，这类型的变量前缀分别加“u8,u16,u32”来表示。但是这种类型的变量还分全局变量和局部变量，为了有所区分，就在全局变量前加“G”来表示，不带“G”的就默认是局部变量。比如：

```
unsigned char Gu8Number;    //Gu8 就代表全局的 8 位变量
unsigned int Gu16Number;    //Gu16 就代表全局的 16 位变量
unsigned long Gu32Number;   //Gu32 就代表全局的 32 位变量

void HanShu(unsigned char u8Data) //u8 就代表局部的 8 位变量
{
    unsigned char u8Number;    //u8 就代表局部的 8 位变量
    unsigned int u16Number;    //u16 就代表局部的 16 位变量
    unsigned long u32Number;   //u32 就代表局部的 32 位变量
}
```

全局变量和局部变量继续往下细分，还分“静态”和“非静态”，为了有所区分，就在前面增加“ES”或“S”来表示，“ES”代表全局的静态变量，“S”代表局部的静态变量。比如：

```
static unsigned char ESu8Number;    //ESu8 就代表全局的 8 位静态变量
static unsigned int  ESu16Number;    //ESu16 就代表全局的 16 位静态变量
static unsigned long ESu32Number;   //ESu32 就代表全局的 32 位静态变量

void HanShu(unsigned char u8Data) //u8 就代表局部的 8 位变量
{
    static unsigned char Su8Number;    //Su8 就代表局部的 8 位静态变量
    static unsigned int Su16Number;    //Su16 就代表局部的 16 位静态变量
    static unsigned long Su32Number;   //Su32 就代表局部的 32 位静态变量
}
```

刚才讲的只是针对“变量”，如果是“常量”，则前缀加“C”来表示，不管是全局的常量还是局部的常量，都统一用“C”来表示，不再刻意区分“全局常量”和“静态常量”，比如：

```
const unsigned char Cu8Number=1;    //Cu8 就代表 8 位常量，不刻意区分“全局”和“局部”
```

```
const unsigned int Cu16Number=1;    //Cu16 就代表 16 位常量，不刻意区分“全局”和“局部”
const unsigned long Cu32Number=1;   //Cu32 就代表 32 位常量，不刻意区分“全局”和“局部”

void HanShu(unsigned char u8Data) //u8 就代表局部的 8 位变量
{
    const unsigned char Cu8Number=1;  //Cu8 就代表 8 位常量，不刻意区分“全局”和“局部”
    const unsigned int Cu16Number=1;  //Cu16 就代表 16 位常量，不刻意区分“全局”和“局部”
    const unsigned long Cu32Number=1; //Cu32 就代表 32 位常量，不刻意区分“全局”和“局部”
}
```

【79.3　循环体变量的命名规范和习惯。】

循环体变量是一个很特定场合用的变量，为了突出它的特殊，这类变量在命名上用单个字母，可以不遵守命名规范，这里的“不遵守命名规范”就是它的“命名规范”，颇有道家“无为就是有为”的韵味，它是命名界的另类。比如：

```
unsigned char i; //超越了规则约束的循环体变量，用单个字母来表示。
unsigned long k; //超越了规则约束的循环体变量，用单个字母来表示。
void HanShu(unsigned char u8Data) //u8 就代表局部的 8 位变量
{
    unsigned int c; //超越了规则约束的循环体变量，用单个字母来表示。
    for(c=0;c<5;c++)  //用在循环体的变量
    {
        u8Data=u8Data+1;  //u8 就代表局部的 8 位变量
    }

    for(i=0;i<5;i++)  //用在循环体的变量
    {
        u8Data=u8Data+1;  //u8 就代表局部的 8 位变量
    }

    for(k=0;k<5;k++)  //用在循环体的变量
    {
        u8Data=u8Data+1;  //u8 就代表局部的 8 位变量
    }
}
```

【79.4　数组的命名规范和习惯。】

数组有四种应用场合，一种是普通数组，一种是字符串，一种是表格，一种是信息。在命名上分别加入后缀“Buffer,String,Table,Message”来区分，但是它们都是数组。比如：

```
unsigned int  Gu16NumberBuffer[5];  //后缀是 Buffer。16 位的全局变量数组。用在普通数组。
unsigned char Gu8NumberString[5];   //后缀是 String。8 位的全局变量数组。用在字符串。

//根据原理图得出的共阴数码管字模表
code unsigned char Cu8DigTable[]=//后缀是 Table。这里的 code 是代表 C51 的常量(类似 const)。
{
0x3f,  //0       序号 0
0x06,  //1       序号 1
0x5b,  //2       序号 2
0x4f,  //3       序号 3
0x66,  //4       序号 4
0x6d,  //5       序号 5
0x7d,  //6       序号 6
0x07,  //7       序号 7
0x7f,  //8       序号 8
0x6f,  //9       序号 9
0x00,  //不显示  序号 10
};

void HanShu(unsigned char u8Data) //u8 就代表局部的 8 位变量
{
    unsigned char u8NumberMessage[5];  //后缀是 Message。8 位的局部变量数组。用在信息。
}
```

【79.5　指针的命名规范和习惯。】

指针的前缀加“p”来区分。再往下细分，指针有全局和局部，有“静态”和“非静态”，有“8 位宽度”和“16 位宽度”和“32 位宽度”，有变量指针和常量指针。比如：

```
unsigned char *pGu8NumberString;  //pGu8 代表全局的 8 位变量指针
void HanShu(const unsigned char *pCu8Data) //pCu8 代表局部的 8 位常量指针
{
    unsigned char *pu8NumberBuffer;            //pu8 代表局部的 8 位变量指针
    static unsigned int *pSu16NumberBuffer;    //pSu16 代表局部的 16 位静态变量指针
    static unsigned long *pSu32NumberBuffer;   //pSu32 代表局部的 32 位静态变量指针
}
```

【79.6　结构体的命名规范和习惯。】

结构体的前缀加“t”来区分。再往下细分，指针有全局和局部，有“静态”和“非静态”，有结构体变量和结构体指针。比如：

```
struct StructSignData  //带符号的数
{
```

```
    unsigned char  u8Sign;  //符号  0为正数 1为负数
    unsigned long u32Data;  //数值
};

struct StructSignData GtNumber; //Gt代表全局的结构体变量。
void HanShu(struct StructSignData *ptData) //pt代表局部的结构体指针
{
    struct StructSignData tNumber; //t代表局部的结构体变量。
    static struct StructSignData StNumber; //St代表局部的静态结构体变量。
}
```

【79.7　宏常量的命名规范和习惯。】

所谓“宏常量”往往是指用#define语句定义的常量。宏常量的所有字符都用大写字母。比如：

```
#define DELAY_TIME   30  //宏常量所有字符都用大写字母。DELAY_TIME代表延时的时间。
void HanShu(void)
{
    delay(DELAY_TIME); //相当于delay(30),这里的delay代表某个延时函数(这里没有具体写出来)
}
```

【79.8　首字符用大写字母以及下划线“_”的灵活运用。】

两个以上的英文单词连在一起命名时，每个单词的首字符用大写，其余用小写，这样可以把每个单词“断句”开来，方便阅读。如果遇到两个英文单词连在一起不好“断句”的情况（比如某个英文单词全部是大写字母的专用名词），只要在两个英文单词之间插入下划线“_”就可以清晰的“断句”了。比如：

```
unsigned long Gu32GetFileLength; //GetFileLength寓意“获取某个文件的长度”。
unsigned char Gu8ESD_Flag; //ESD是专业用名词，代表“静电释放”的意思。用下划线“_”断句。
```