第五十三节：函数的作用和四种常见书写类型。

【53.1　函数的作用和分类。】

函数的作用。通常把一些可能反复用到的算法或者过程封装成一个函数，函数就是一个模块，给它输入特定的参数，就可以输出想要的结果，比如一个加法函数，只要输入加数和被加数，然后就会输出相加结果之和，里面具体的算法过程只要写一次就可以重复调用，极大的节省单片机程序容量，也节省程序开发人员的工作量。

函数的分类。从输入输出的角度来看，有四种常见的书写类型。分别是无输出无输入，无输出有输入，有输出无输入，有输出有输入。输出是看函数名的前缀，前缀如果是 void 表示无输出，否则就是有输出。输入是看函数名括号里的内容，如果是 void 或者是空着就表示无输入，否则就是有输入。输出和输入是比较通俗的说法，专业一点的说法是，有输出表示函数有返回，无输出表示函数无返回。有输入表示有形参，无输入表示无形参。下面举一个加法函数的例子，分别用四种不同的函数类型来实现。大家对比一下它们在书写方面有哪些不同，又有哪些规律。

【53.2　无输出无输入的函数。】

```
unsigned char a;   //此变量用来接收最后相加结果的和。
unsigned char g=2;
unsigned char h=3;
void hanshu(void)   //无输出无输入函数的定义。
{
    a=g+h;
}

main()
{
     hanshu();   //函数调用时的样子。
}
```

分析：

void hanshu(void)，此函数名的前缀是 void，括号内也是 void，属于无输出无输入函数。这类函数表面看是无输出无输入，其实在实际应用中也可以通过全局变量来输入输出，比如上面的例子就是靠 a,g,h 这三个全局变量来传递信息，只不过表达方式上像隐藏起来一样没有那么直观。

【53.3　无输出有输入的函数。】

```
unsigned char b;   //此变量用来接收最后相加结果的和。
void hanshu(unsigned char i,unsigned char k)     //无输出有输入函数的定义
{
    b=i+k;
}
```

```
main()
{
    hanshu(2,3);   //函数调用时的样子。
}
```

分析：

void hanshu(unsigned char i,unsigned char k)，此函数名的前缀是 void，括号内是(unsigned char i,unsigned char k)，属于无输出有输入的函数。括号的两个变量 i 和 k 是函数内的局部变量，也是跟对外的桥梁接口，它们有一个专业的名称叫形参。外部要调用此函数时，只要给括号填入对应的变量或者数值，这些变量和数值就会被复制一份传递给作为函数形参的局部变量，从而外部调用者跟函数内部就发生了数据信息的传递。这种书写方式的特点是把输入接口封装了出来。

【53.4　有输出无输入的函数。】

```
unsigned char c;    //此变量用来接收最后相加结果的和。
unsigned char m=2;
unsigned char n=3;
unsigned char hanshu(void)        //有输出无输入函数的定义。
{
    unsigned char p;
    p=m+n;
    return p;
}

main()
{
    c= hanshu();   //函数调用时的样子。
}
```

分析：

unsigned char hanshu(void)，此函数名的前缀是 unsigned char 类型，括号内是 void，属于有输出无输入的函数。函数前缀的 unsigned char 表示此函数最后退出时会返回一个 unsigned char 类型的数据给外部调用者。而且这类函数内部必须有一个 return 语句配套，表示立即退出当前函数并且返回某个变量或者常量的数值给外部调用者。这种书写方式的特点是把输出接口封装了出来。

【53.5　有输出有输入的函数。】

```
unsigned char d;      //此变量用来接收最后相加结果的和。
unsigned char hanshu(unsigned char r,unsigned char s)        //有输出无输入函数的定义
```

```
{
    unsigned char t;
    t=r+s;
    return t;
}

main()
{
    d= hanshu(2,3);   //函数调用时的样子。
}
```

分析：

unsigned char hanshu(unsigned char r,unsigned char s)，此函数名的前缀是 unsigned char 类型，括号内是(unsigned char r,unsigned char s)，属于有输出有输入的函数。输入输出的特点跟前面介绍的函数一样，不多讲。这种书写方式的特点是把输出和输入接口都封装了出来。

【53.6　调用函数时特别要注意。】

注意：在函数调用时，凡是“void”关键词都要省略，否则编译不通过。

【53.7　练习程序。】

现在把上述的 4 种加法函数写成一个程序，最后把程序编译后下载到坚鸿 51 学习板观察结果。请直接复制第十节模板程序，添加和修改 main 程序代码如下：

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

//第 1 种：无输出无输入函数的声明。
void hanshu_1(void);

//第 2 种：无输出有输入函数的声明。
void hanshu_2(unsigned char i,unsigned char k);

//第 3 种：有输出无输入函数的声明。
unsigned char hanshu_3(void);

//第 4 种：有输出无输入函数的声明。
unsigned char hanshu_4(unsigned char r,unsigned char s);

//第 1 种：无输出无输入
unsigned char a;  //此变量用来接收最后相加结果的和。
unsigned char g=2;
```

```
unsigned char h=3;


//第 2 种：无输出有输入
unsigned char b;  //此变量用来接收最后相加结果的和。


//第 3 种：有输出无输入
unsigned char c;   //此变量用来接收最后相加结果的和。
unsigned char m=2;
unsigned char n=3;

//第 4 种：有输出有输入
unsigned char d;    //此变量用来接收最后相加结果的和。


//第 1 种：无输出无输入函数的定义。
void hanshu_1(void)
{
   a=g+h;
}

//第 2 种：无输出有输入函数的定义
void hanshu_2(unsigned char i,unsigned char k)
{
   b=i+k;
}


 //第 3 种：有输出无输入函数的定义。
unsigned char hanshu_3(void)
{
unsigned char p;
p=m+n;
return p;
}

//第 4 种：有输出无输入函数的定义
unsigned char hanshu_4(unsigned char r,unsigned char s)
{
unsigned char t;
t=r+s;
return t;
}
```

```
void main() //主程序
{

    hanshu_1();         //第 1 种：无输出无输入函数。
    hanshu_2(2,3);       //第 2 种：无输出有输入函数。
    c=hanshu_3();       //第 3 种：有输出无输入函数。
    d=hanshu_4(2,3);      //第 4 种：有输出无输入函数。



    GuiWdData0=a;    //把 a 这个数值放到窗口变量 0 里面显示。
    GuiWdData1=b;    //把 b 这个数值放到窗口变量 1 里面显示。
    GuiWdData2=c;    //把 c 这个数值放到窗口变量 2 里面显示。
    GuiWdData3=d;    //把 d 这个数值放到窗口变量 3 里面显示。


/*---C 语言学习区域的结束
--------------------------------------------------------------------------------*/
    while(1)
    {
       initial();
       key_service();
       display_service();
    }

}
```

查看运算结果的方法。如何在坚鸿 51 学习板上观察变量？按下 S1 或者 S5 按键即可切换显示不同的窗口，从而显示不同的变量。按下 S9 按键不松手就可以切换到十六进制的显示界面，松开手后会自动切换到十进制的界面。16 个 LED 灯显示的就是当前变量的二进制数，亮代表 1，灭代表 0。上坚鸿 51 学习板观察程序执行的结果如下：

变量 a 为 5。
变量 b 为 5。
变量 c 为 5。
变量 d 为 5。

下节预告：return 语句在函数中的作用。
（未完待续）