第五十四节：return 语句在函数中的作用以及容易被忽略的四个功能。

【54.1　return 深入讲解】

“return”在英语单词中有“返回”的意思，在上一节提到，凡是“有输出函数”，函数内部必须有一个“return+变量或者常量”与之配套，表示返回的结果给外部调用者接收，这个初学者都很容易理解，容易被忽略的是 return 另外四个功能：

第一个是 return 语句隐含了立即退出的功能。退出哪？退出当前函数。只要执行到 return 语句，就马上退出当前函数。即使 return 语句身陷多层 while 或者 for 的循环中，它也毫不犹豫立即退出当前函数。

第二个是 return 语句可以出现在函数内的任何位置。可以出现在第一行代码，也可以出现在中间的某行代码，也可以出现在最后一行的代码，它的位置不受限制。很多初学者有个错觉，以为 return 只能出现在最后一行，这是错的。

第三个是 return 语句不仅仅可以用在“有输出函数”，也可以用在“无输出函数”，也就是可以用在前缀是 void 的函数里。回顾上一节，在“有输出函数”函数里，return 后面紧跟一个变量或者常量，表示返回的数，但是在“无输出函数”里，因为是“无输出”，此时 return 后面不用跟任何变量或者常量，表示返回的是空的。此时 return 主要起到立即退出当前函数的作用。

第四个是 return 语句可以在一个函数里出现 N 次，次数不受限制，不一定必须只能一次。不管一个函数内有多少个 return 语句，只要任何一个 return 语句被单片机执行到，立即退出当前函数。

【54.2　中途立即退出的功能。】

下面的书写格式是合法的：

```
void hanshu(void)   //无输出/函数的定义。
{
    语句 1;
    return; //立即退出当前函数。
    语句 2;
    return; //立即退出当前函数。
    语句 3;
    return; //立即退出当前函数。
}
```

分析：

当 hanshu 此函数被调用时，单片机从“语句 1”往下执行，当遇到第一个 return 语句后，马上退出当前函数。在此函数里，后面的“语句 2”等代码永远不会被执行到。

【54.3　身陷多层 while 或者 for 的循环时的惊人表现。】

下面的书写格式是合法的：

```
void hanshu(void)   //无输出/函数的定义。
{
```

```
    语句 1;
    while(1)   //第一个循环
    {
        while(1)   //第二个循环中的循环
        {
            return; //立即退出当前函数。
        }
        语句 2;
        return; //立即退出当前函数。
    }
    语句 3;
    return; //立即退出当前函数。
}
```

分析：

当 hanshu 此函数被调用时，单片机从“语句 1”往下执行，先进入第一个循环，接着进入第二个循环中的循环，然后遇到第一个 return 语句，马上退出当前函数。在此函数里，后面的“语句 2”等代码也是永远不会被执行到。虽然表面看起来有那么多可怕的循环约束着，但是一旦碰上 return 语句都是浮云，立刻退出当前函数。

【54.4　“有输出函数”时是什么样子的？】

把上面例子中“无输出函数”改成“有输出函数”后：

```
unsigned char hanshu(void)   //有输出/函数的定义。
{
    unsigned char a=9;

    语句 1;
    while(1)   //第一个循环
    {
        while(1)   //第二个循环中的循环
        {
            return a; //返回 a 变量的值，并且立即退出当前函数。
        }
        语句 2;
        return a; //返回 a 变量的值，并且立即退出当前函数。

    }
    语句 3;
    return a; //返回 a 变量的值，并且立即退出当前函数。
}
```

分析：

因为此函数是“有输出/函数”，所以 return 语句后面必须配套一个变量或者常量，而执行顺序跟前面 54.3 的例子是一样的。当 hanshu 函数被调用时，单片机从“语句 1”往下执行，先进入第一个循环，接着进入第二个循环中的循环，然后遇到第一个“return a”语句，马上退出当前函数。在此函数里，后面的“语句 2”等代码也是永远不会被执行到的。再一次说明了，return 语句不仅有返回某数的功能，还有立即退出的重要功能。

【54.5　项目应用时，中间的 return 语句往往是跟 if 语句搭配使用的。】

前面的例子只是为了解释 return 语句的执行顺序和功能，实际项目中，如果中间有多个 return 语句，中间的 return 语句不可能像前面的例子那样单独使用，它往往是跟 if 语句一起搭配使用的，否则单独用 return 就没有什么意义。比如：

```
void hanshu(void)   //无输出/函数的定义。
{
    语句 1;
    if(某条件满足)
    {
        return; //立即退出当前函数。
    }
    语句 2;
    if(某条件满足)
    {
        return; //立即退出当前函数。
    }
    语句 3;
}
```

分析：

单片机从“语句 1”开始往下执行，至于在哪个“return”语句处退出当前函数，就要看哪个 if 的条件满不满足了，如果所有的 if 的条件都不满足，此函数会一直执行完最后的“语句 3”才退出当前函数。

【54.6　函数和变量的命名规则。】

函数的名字和变量的名字一样，第一个字符不能是数字，必须是字母或者下划线“_”，后面紧跟的第 2 个字符开始可以是数字。在 C 语言中名字是区分大小写的。可以用下划线“_”，但是不可以用横杠“-”。名字不能跟 C 编译系统已经征用的关键字重名，比如不能用“unsigned ”,“char”,“static”等系统关键词，跟古代时不能跟皇帝重名一样，要避尊者讳。

【54.7　练习程序。】

写一个简单的除法函数，在除法运算中，除数不能为 0，如果发现除数为 0，就立即退

出当前函数，并且返回运算结果默认为 0。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----------------------------------------------*/

//函数的声明。
unsigned int chu_fa(unsigned int bei_chu_shu,unsigned int chu_shu);

unsigned int a;//此变量用来接收除法的运算结果。
unsigned int b;//此变量用来接收除法的运算结果。


//函数的定义。
unsigned int chu_fa(unsigned int bei_chu_shu,unsigned int chu_shu)
{
    unsigned int shang;  //返回的除法运算结果：商。
    if(0==chu_shu)   //如果除数等于 0，就立即退出当前函数，并返回 0
    {
        return 0; // 退出当前函数并且返回 0.此时后面的代码不会被执行。
    }

    shang=bei_chu_shu/chu_shu;  //除法运算的算法

    return shang;  //返回最后的运算结果：商。并且退出当前函数。
}

void main() //主程序
{



    a=chu_fa(128,0);  //函数调用。128 除以 0，把商返回给 a 变量。
    b=chu_fa(128,2);  //函数调用。128 除以 2，把商返回给 b 变量。

    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。

/*---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 0。

变量 b 为 64。

下节预告：static 静态局部变量在函数中的重要作用。

（未完待续）