第五十五节：static 静态局部变量在函数中的重要作用。

【55.1　变量前加入 static 关键词后发生“化学反应”】

前面介绍了两类变量，一类全局变量，一类局部变量。定义时，在任何一类变量前面加入 static 关键词，变量原有的特性都会发生某些变化，此时 static 像化学里的催化剂，具有神奇的功能。加 static 关键词的书写格式如下：

```
static unsigned char a;   //全局变量前加的 static 关键词
void hanshu(void)
{
    static unsigned char i;    //局部变量前加的 static 关键词
}
```

【55.2　在全局变量前加 static】

static 读作“静态”，全局变量前加 static，称为静态全局变量。静态全局变量和普通全局变量的功能大体相同，仅在有效范围(作用域)方面有差异。假设整个工程有多个文件组成，普通全局变量的有效范围能覆盖全部文件，在任何一个文件里都畅通无阻。而静态全局变量只能在当前定义的那个文件里起作用，活动范围完全被限定在一个文件，彷佛被加了紧箍咒，由不得你任性。这部分的内容有个大致印象就可以，暂时不用深入研究，等以后学到多文件编程时再关注，因为我当前的程序例子只有一个源文件，还没涉及多文件编程。

【55.3　在局部变量前加 static】

这是本节重点。我常把局部变量比喻宾馆的客房，客人入住时被分配在哪间客房是随机临时安排的，第二天退房时宾馆会把客房收回继续分配给下一位其他的客人，是临时公共区。而加入 static 后的局部变量，发生了哪些变化？加入 static 后的局部变量，称为静态局部变量。静态局部变量就像宾馆的 VIP 客户，VIP 客户财大气粗，把宾馆分配的客房永远包了下来，永远不许再给其它客人入住。总结了静态局部变量的两个重要特性：

第一个，静态局部变量不会在函数调用时被初始化，它只在单片机刚上电时被编译器初始化了一次，因为它的内存模型不是被分配在“栈”，而是在全局变量同一类数据区，拥有自己唯一的地址。但是跟全局变量又有差别，全局变量的有效范围（作用域）是整个工程，而静态局部变量毕竟是“局部”，仅局限于当前函数。而普通局部变量，众所周知，每次被函数调用时，都会被重新初始化。

第二个，每次函数调用时，静态局部变量比普通局部变量少开销一条潜在的“初始化语句”，原因是普通局部变量每次被函数调用时都要重新初始化，而静态局部变量不用进行这个操作。也就是说，静态局部变量比普通局部变量的效率高一点，虽然这个“点”的时间开销微不足道，但是不留意这“点”，写程序时容易出现瑕疵。

【55.4　静态局部变量的应用场合】

静态局部变量适用在那些频繁调用的函数，比如 main 函数主循环 while(1)里直接调用的所有函数，还有以后讲到的定时器中断函数，等等。因为静态局部变量每次被调用都不会被重新初始化，用在这类函数时就省去了每次初始化语句的时间。还有就是那些规定不能被函数初始化的场合，比如在很多用 switch 搭建的程序函数里，这类 switch 程序俗称为状态机思路。

【55.5　能用全局变量替代静态局部变量吗】

能用全局变量替代静态局部变量吗？能。哪怕在整个程序里全部用全局变量都可以。全局变量是一把牛刀，什么场合都用牛刀虽然也能解决问题，但是显得鲁莽没有条理。尽量把全局变量，普通局部变量，静态局部变量各自优势充分发挥出来才是编程之道。能用局部变量的尽量用局部变量，这样可以减少全局变量的使用。当局部变量帮分担一部分工作时，最后全局变量只起到一个作用，那就是在各函数之间传递信息。局部变量与全局变量的分工定位明确了，程序代码阅读起来就没有那么凌乱，思路也清晰很多。

【55.6　程序分析】

编写一个程序来学习刚才讲到的内容，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----------------------------------------*/
unsigned char hanshu(void);         //函数声明
unsigned char hanshu_static(void);  //函数声明

unsigned char hanshu(void)  //函数定义
{
   unsigned char i=0;   //普通局部变量，每次函数调用都被初始化为 0.

   i++;  //i 自加 1

   return i;
}

unsigned char hanshu_static(void)  //函数定义
{
   static unsigned char i=0;   //静态局部变量，只在上电是此初始化语句才起作用。

   i++;  //i 自加 1

   return i;
}

void main() //主程序
{
  unsigned char a; //用来接收函数返回的结果。
  unsigned char b;
  unsigned char c;

  unsigned char d; //用来接收函数返回的结果。
  unsigned char e;
  unsigned char f;
```

```
    //下面函数内的 i 是普通局部变量。
    a=hanshu();  //函数内的 i 每次重新初始化为 0，再自加 1，所以 a 等于 1。
    b=hanshu();  //函数内的 i 每次重新初始化为 0，再自加 1，所以 b 等于 1。
    c=hanshu();  //函数内的 i 每次重新初始化为 0，再自加 1，所以 c 等于 1。

    //下面函数内的 i 是静态局部变量，第一次上电后默认为 0，就不会再被初始化，
    d=hanshu_static(); //d 由 0 自加 1 后等于 1。
    e=hanshu_static(); //e 由 1 自加 1 后等于 2。
    f=hanshu_static(); //f 由 2 自加 1 后等于 3。

    GuiWdData0=a;    //把 a 这个变量放到窗口变量 0 里面显示，下同。
    GuiWdData1=b;
    GuiWdData2=c;
    GuiWdData3=d;
    GuiWdData4=e;
    GuiWdData5=f;

  /*---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 1。
变量 b 为 1。
变量 c 为 1。

变量 d 为 1。
变量 e 为 2。
变量 f 为 3。

【55.7　为什么中止此连载帖的更新了】

我以前一直想写两本书，一本讲单片机入门基础，一本讲单片机程序框架。现在发现，单片机基础和程序框架并没有明显的分水岭，基础中有框架，框架中有基础，应该合二为一，读起来才会连贯舒畅。所以我决定中止当前已写到 55 节的《从业十年，教你单片机入门基础》连载帖，另外新开一个连载帖叫《从单片

机基础到程序框架（连载）》，希望大家关注我的新连载帖。

再提一下我 2014 年写的《从业将近十年，手把手教你单片机程序框架》，一方面受到很多网友的好评，另一方面也有一些热心网友提出了宝贵的意见，我今天看来，确实还有一些可待改进的地方。本来计划在 2017 年重写《……单片机程序框架》那个老帖，现在看来不用那么折腾了，只要把《……单片机程序框架》的内容也整合到新开的帖子里就可以了，这样对我也比较省事。我的时间计划是，先花 4 年时间写一个初稿，然后再花 2 年时间重写一次，最后再花 1 年时间整理成书，整个过程大概 7 年时间左右，今年是 2016 年，估计到 2023 年左右就可以新书出版了。

感谢各位朋友的支持。