第五十节： 循环体内的 continue 和 break 语句。

【50.1　continue 语句。】

通常情况下，单片机在循环体里从第一行的“入口条件”开始往下执行，直至碰到循环体的边界“底部花括号”，才又折回到第一行的“入口条件”准备进行新一轮的循环。但是，若中途碰到 continue 语句，就会提前结束当前这一轮的循环，只要碰到 continue 语句，就立即折回到第一行的“入口条件”准备进行新一轮的循环。注意，continue 语句“结束”的对象仅仅是“当前这一轮的循环”，并没有真正结束这个循环的生命周期。它好像拦路虎，遇到它，它说“你回去，第二天再来。”这台词里的“第二天再来”就强调这个循环体的生命周期还没有真正结束。举一个具体的例子，如下：

```
while(…)或者 for(…)   //循环体的条件判断入口处
{    //循环体开始
    语句 1;
    语句 2;
    continue;
    语句 3;
    ……
    语句 N;
}    //循环体结束
```

分析：上述语句中，单片机从“循环体的条件判断入口处”开始往下执行，碰到 continue 就马上折回到“循环体的条件判断入口处”，继续开始新一轮的循环，因此，这段代码，continue 后面的“语句 3”至“语句 N”是永远也不会被执行到的。因为 continue 的拦截，上述语句等效于：

```
while(…)或者 for(…)   //循环体的条件判断入口处
{    //循环体开始
    语句 1;
    语句 2;
}    //循环体结束
```

问题来了，既然可以如此简化，还要 continue 干什么，不是多此一举？在实际应用中，continue 肯定不会像上面这样单独使用，continue 只有跟 if 语句结合，才有它存在的意义。例如：

```
while(…)或者 for(…)   //循环体的条件判断入口处
{    //循环体开始
    语句 1;
    语句 2;
    if(某条件)
    {
        continue;
    }
    语句 3;
    ……
    语句 N;
```

```
}    //循环体结束
```

【50.2　break 语句。】

continue 语句提前结束当前这一轮的循环，准备进入下一轮的新循环，强调“某次结束”，但不是真结束。break 语句是直接跳出当前循环体，是真正的结束当前循环体，强调循环体的“生命结束”。举例如下：

```
while(…)或者 for(…)    //循环体的条件判断入口处
{    //循环体开始
    语句 1;
    语句 2;
    break;
    语句 3;

    ……
    语句 N;
}    //循环体结束
语句(N+1);  //循环体之外语句
```

分析：上述语句中，单片机从“循环体的条件判断入口处”开始往下执行，突然碰到 break 语句，此时，立即无条件跳出当前循环体（无需判断 while 或者 for 的条件），直接执行到循环体之外的“语句(N+1)”，break 后面的“语句 3”至“语句 N”也没有被执行到。实际项目中，break 也往往会配合 if 一起使用，例如：

```
while(…)或者 for(…)    //循环体的条件判断入口处
{    //循环体开始
    语句 1;
    语句 2;
    if(某条件)
    {
        break;
    }
    语句 3;

    ……
    语句 N;
}    //循环体结束
语句(N+1);  //循环体之外语句
```

【50.3　break 语句能跳多远？】

break 语句能跳多远？预知答案请先看以下例子：

```
while(…)
{
```

```
        语句 1;
        语句 2;
        while(…)
        {
            语句 3;
            break;
            语句 4;
        }
        语句 5;
    }
    语句 6;
```

分析：上述例子中，在 while 循环里面有藏着第二个 while 循环，像这种循环之中还有循环的情况，通常称为循环嵌套。单片机从上往下执行，当遇到 break 后，它会跳到“语句 5”那里呢，还是会跳到“语句 6”那里？正确答案是“语句 5”那里，这说明了 break 语句的“有效射程”仅仅刚好能跳出当前的循环体。也就是说，在上述循环嵌套的例子中，最内层的 break 只能跳出最内层的循环体，不能跳到最外层的“语句 6”那里，如果需要继续跳出最外层的“语句 6”那里，可以继续在外层的循环体内再增加一个 break 语句。

【50.4　还有哪些语句可以无条件跳出循环体？】

除了 break 以外，还有 return 和 goto 语句可以跳出循环体。这部分的内容大家只需大概了解一下即可。return 语句比 break 语句还厉害，它不仅仅跳出当前循环体，还是跳出了当前函数，也就是提前结束了当前函数，这部分的内容后面章节会讲到，暂时不用管。而 goto 语句在 C 语言中大家都公认不建议用，因为它很容易扰乱大家常用的 C 语言编程结构，我本人也从来没有用过 goto 语句，因此不再深入讲解它。

【50.5　例程练习和分析。】

编写一个程序来熟悉一下 continue 和 break 语句的使用。

程序代码如下：

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

    unsigned char a=0;  //观察这个数最后的变化
    unsigned char b=0;  //观察这个数最后的变化
    unsigned char c=0;  //观察这个数最后的变化
    unsigned char d=0;  //观察这个数最后的变化

    unsigned char i;  //控制循环体的条件判断变量
void main() //主函数
{
    //i<6 的条件判断是在进入循环体之前判断，而 i 的自加 1 是在执行完一次循环体之后才自加的。
    for(i=0;i<6;i++)
    {
       a=a+1;    //被执行了 6 次，分别是第 0,1,2,3,4,5 次
```

```
        if(i>=3)  //当 i 等于 3 的时候，开始“拦截”continue 后面的代码。
        {
           continue;  //提前结束本次循环，准备进入下一次循环
        }
        b=b+1;  //被执行了 3 次，分别是第 0,1,2 次
    }

    //i<6 的条件判断是在进入循环体之前判断，而 i 的自加 1 是在执行完一次循环体之后才自加的。
    for(i=0;i<6;i++)
    {
        c=c+1;   //被执行了 4 次，分别是第 0,1,2,3 次
        if(i>=3)  //当 i 等于 3 的时候，直接跳出当前循环体，结束此循环体的“生命周期”。
        {
           break; //马上跳出当前循环体
        }
        d=d+1;   //被执行了 3 次，分别是第 0,1,2 次
    }

    View(a);  //把第 1 个数 a 发送到电脑端的串口助手软件上观察。
    View(b);  //把第 2 个数 b 发送到电脑端的串口助手软件上观察。
    View(c);  //把第 3 个数 c 发送到电脑端的串口助手软件上观察。
    View(d);  //把第 4 个数 d 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:6
十六进制:6
二进制:110

第 2 个数
十进制:3
十六进制:3
二进制:11

第 3 个数
```

```
十进制:4
十六进制:4
二进制:100

第 4 个数
十进制:3
十六进制:3
二进制:11
```

分析：

变量 a 为 6。

变量 b 为 3。

变量 c 为 4。

变量 d 为 3。

【50.6　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。