

第四十三节：关系符中的关系符:与“&&”，或“||”。

【43.1 关系符中的与“&&”。】

前面在讲关系符的时候，讲了只存在 1 个（判断条件）的情况下，根据这个判断为真还是为假再执行对应的操作，那么，当同时存在 2 个（判断条件）以上的情况下，该如何描述（判断条件）与（判断条件）之间的关系，这就涉及本节所讲的“关系符中的关系符”：与“&&”，或“||”。

先讲“&&”语句，符号“&&”称为“与”，它的含义是：假如有两个以上的（条件判断），当所有的（条件判断）都满足的时候，才认为这个整体判断是真，否则，只要有 1 个（条件判断）不满足，那么整体判断就是假。这个规律，有点像很多开关在电路回路中的串联关系，只有所有串联在回路中的开关都是闭合的状态，这个回路才是畅通的，否则，只要有一个开关是断开的，整个回路就是断开的。

与语句“&&”的常见格式如下：

```
if(第 1 个条件判断&&第 2 个条件判断...&&第 N 个条件判断)
{
    语句 1;
    语句 2;
}
语句 3;
语句 4;
```

在上述格式中，只有 if 语句后面小括号内所有的(条件判断)都满足的时候，整体判断才为真，才会执行到大括号内的“语句 1”和“语句 2”，否则，只要有 1 个不满足，就直接跳到“语句 3”处往下执行。

再举一个具体的例子，比如要取从 70 到 80 之间的所有数据，也就是说，既要大于等于 70，同时又要小于等于 80，程序代码可以这样书写：

```
if(a>=70&&a<=80)  //在 70 到 80 的区间范围（包括边界 70 和 80）
{
    语句 1;
    语句 2;
    .....
    语句 N;
}
```

【43.2 关系符中的或“||”。】

符号“||”称为“或”，它的含义是：假如有两个以上的（条件判断），只要有 1 个条件判断为真，则此整体判断裁定为真，否则，必须所有的（条件判断）都不满足，此整体判断才会裁定为假。这个规律，有点像很多开关在电路回路中的并联关系，并联在回路的多个开关，只要有 1 个开关是闭合的状态，那么这个回路肯定是畅通的，否则，必须全部开关都是断开的，整个回路才会是断开的。

或语句“||”的常见格式如下：

```
if(第 1 个条件判断||第 2 个条件判断...||第 N 个条件判断)
{
    语句 1;
    语句 2;
}
```

语句 3;

语句 4;

在上述格式中，只要 if 语句后面小括号内有 1 个(条件判断)是满足的时候，整体判断马上裁定为真，这时就会执行到大括号内的“语句 1”和“语句 2”，否则，必须全部的(条件判断)都不满足，整体判断才会裁定为假，这时就会直接跳到“语句 3”处往下执行。

再举一个具体的例子，比如要取除了 70 到 80 之间以外的所有数据，也就是说，要么小于 70，或者要么大于 80，可以这样写：

```
if(a<70||a>80) //在 70 到 80 的区间范围以外的数据（不包括边界 70 和 80）
{
    语句 1;
    语句 2;
    .....
    语句 N;
}
```

【43.3 “&”和“&&”，“|”和“||”的区别。】

前面章节讲过运算符的“&”和“|”，它们发音也是“与”和“或”，跟本节讲的关系符“&&”和“||”的发音是同音，因此很容易让初学者混淆。区别如下：

运算符的“&”和“|”，是属于运算符，是强调数与数，变量与变量，个体与个体之间的运算，而不是关系。它们之间的运算，是把一个数或一个变量转化成二进制后，进行二进制的 0 和 1 之间的“与”“或”运算。

关系符的“&&”和“||”，是属于关系符，是强调（条件判断）与（条件判断），关系与关系，整体与整体之间的关系判断，而不是运算。它们之间的关系，关键词是判断。

【43.4 “&&”和“||”的“短路”问题。】

关系符“&&”和“||”居然也有“短路”问题？大家不要惊异，这里所说的“短路”只是强调关系符内部判断的顺序和取舍。“短路”这个词在这里只是业内已经习惯了一种称谓，虽然我个人感觉有一点怪怪的不自然，但是我自己也想不出其它更好的词来描述这种关系，因此就跟业内已习惯的称谓保持一致。

“&&”的“短路”，它内部判断的顺序和取舍是这个样子的：在两个以上的判断中，从左边到右边，依次逐个判断，先判断第 1 个（条件判断），再第 2 个（条件判断）...再第 N 个（条件判断），但是，在此期间，只要发现有 1 个条件是不满足，就马上退出判断，不再继续判断后面的（条件判断），因为，对于“与”的关系符，只要有 1 个条件判断是不满足（假），就可以马上裁定整体判断为假了，没必要继续判断后面的（条件判断）。

“||”的“短路”，它内部判断的顺序和取舍是这个样子的：在两个以上的判断中，从左边到右边，依次逐个判断，先判断第 1 个（条件判断），再第 2 个（条件判断）...再第 N 个（条件判断），但是，在此期间，只要发现有 1 个条件是满足，就马上退出判断，不再继续判断后面的（条件判断），因为，对于“或”的关系符，只要有 1 个条件判断是满足（真），就可以马上裁定整体判断为真了，没必要继续判断后面的（条件判断）。

上述文字中的“从左到右”就是“顺序”，“马上退出”就是“取舍”。这种关系之所以称谓为“短路”，我猜测可能是把“&&”和“||”比喻成在电路的回路中，只要有个1个地方短路了，就可以马上裁定这个回路是短路的，就不用再判断其它地方了。

【43.5 例程练习和分析。】

现在编写一个实验程序，一共有8个给定的数，要统计其中数值从70到80之间的数有几个，统计其中取除了70到80之间以外的数有几个。

程序代码如下：

```
/*---C 语言学习区域的开始。-----*/
unsigned char x1=90; //给定的第1个数
unsigned char x2=65; //给定的第2个数
unsigned char x3=85; //给定的第3个数
unsigned char x4=79; //给定的第4个数
unsigned char x5=95; //给定的第5个数
unsigned char x6=65; //给定的第6个数
unsigned char x7=75; //给定的第7个数
unsigned char x8=85; //给定的第8个数

unsigned char a=0; //统计从70到80的变量总数
unsigned char b=0; //统计除了70到80以外的变量总数

void main() //主函数
{
    //第一部分:统计“从70到80之间的数有多少个。

    if(x1>=70&& x1<=80) //如果条件为真，则执行下面大括号里面的语句。
    {
        a++; //相当于 a=a+1，用来统计从70到80的总数
    }

    if(x2>=70&& x2<=80) //如果条件为真，则执行下面大括号里面的语句。
    {
        a++; //相当于 a=a+1，用来统计从70到80的总数
    }

    if(x3>=70&& x3<=80) //如果条件为真，则执行下面大括号里面的语句。
    {
        a++; //相当于 a=a+1，用来统计从70到80的总数
    }

    if(x4>=70&& x4<=80) //如果条件为真，则执行下面大括号里面的语句。
    {
```

```

    a++;    //相当于 a=a+1, 用来统计从 70 到 80 的总数
}

if (x5>=70&& x5<=80)    //如果条件为真, 则执行下面大括号里面的语句。
{
    a++;    //相当于 a=a+1, 用来统计从 70 到 80 的总数
}

if (x6>=70&& x6<=80)    //如果条件为真, 则执行下面大括号里面的语句。
{
    a++;    //相当于 a=a+1, 用来统计从 70 到 80 的总数
}

if (x7>=70&& x7<=80)    //如果条件为真, 则执行下面大括号里面的语句。
{
    a++;    //相当于 a=a+1, 用来统计从 70 到 80 的总数
}

if (x8>=70&& x8<=80)    //如果条件为真, 则执行下面大括号里面的语句。
{
    a++;    //相当于 a=a+1, 用来统计从 70 到 80 的总数
}

//第二部分:统计除了 70 到 80 之间以外的数有多少个。

if (x1<70|| x1>80)    //如果条件为真, 则执行下面大括号里面的语句。
{
    b++;    //相当于 b=b+1, 用来统计除了 70 到 80 以外的总数
}

if (x2<70|| x2>80)    //如果条件为真, 则执行下面大括号里面的语句。
{
    b++;    //相当于 b=b+1, 用来统计除了 70 到 80 以外的总数
}

if (x3<70|| x3>80)    //如果条件为真, 则执行下面大括号里面的语句。
{
    b++;    //相当于 b=b+1, 用来统计除了 70 到 80 以外的总数
}

if (x4<70|| x4>80)    //如果条件为真, 则执行下面大括号里面的语句。
{
    b++;    //相当于 b=b+1, 用来统计除了 70 到 80 以外的总数
}

```

```

    if(x5<70||x5>80) //如果条件为真，则执行下面大括号里面的语句。
    {
        b++; //相当于 b=b+1，用来统计除了 70 到 80 以外的总数
    }

    if(x6<70||x6>80) //如果条件为真，则执行下面大括号里面的语句。
    {
        b++; //相当于 b=b+1，用来统计除了 70 到 80 以外的总数
    }

    if(x7<70||x7>80) //如果条件为真，则执行下面大括号里面的语句。
    {
        b++; //相当于 b=b+1，用来统计除了 70 到 80 以外的总数
    }

    if(x8<70||x8>80) //如果条件为真，则执行下面大括号里面的语句。
    {
        b++; //相当于 b=b+1，用来统计除了 70 到 80 以外的总数
    }

    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。

    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:2

十六进制:2

二进制:10

第 2 个数

十进制:6

十六进制:6

二进制:110

分析：

变量 a 为 2。（数值从 70 到 80 之间的有 x4, x7 这 2 个）

变量 b 为 6。（除了 70 到 80 之间以外的有 x1, x2, x3, x5, x6, x8 这 6 个）

通过实验结果，发现在单片机上的实验结果和我们的分析是一致的。

【43.6 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。