

## PART8

- 源码、反码、补码。

原码：

就是一个二进制数字，从“数学观念”上来表达出的形式。其中，我们规定：  
一个数字的最左边一位是“符号位”，0表示正数，1表示负数；

反码：

正数的反码就是其本身（即不变）；

负数的反码是：符号位不变，其他位取反；

补码：

正数的补码就是其本身（即不变）；

负数的补码是：符号位不变，其他位取反后+1——即反码+1

tip：计算机内部的运算，实际全都是使用补码进行的，而且运算的时候，符号位不再区分，直接也当做“数据”参与运算：

- 位运算符的应用：管理一组事物的开关状态

什么是开关状态？

现实中，有很多数据都是只有2种结果（值）的，对应的其实就是我们的布尔类型的值。

这里，所谓管理一组事物的开关状态，应该理解为其实就是管理若干个只有2个状态的“数据符号”。

比如：有5个灯泡，对应5个状态数据。

这5个灯泡，就有 25 种状态呢？

这里的管理目标是：使用一个变量，就可以表达若干个数据的“当前状态”。具体有3个任务：

- 1，通过该变量，可以获取任何一个数据的当前状态。
- 2，通过该变量，可以将一个一个数据的状态“关闭”；
- 3，通过该变量，可以将一个一个数据的状态“开启”；

代码示例如下：

```
<?php
header('Content-type:text/html; charset=utf-8');

//1.可以指定任意一个“当前状态”；
//2.可以打开的任意开关
//3.可以关闭的任意开关

define("D1", 1); //代表对应二进制为： 00000001
define("D2", 2); //代表对应二进制为： 00000010
define("D3", 4); //代表对应二进制为： 00000100
define("D4", 8); //代表对应二进制为： 00001000
define("D5", 16); //代表对应二进制为： 00010000

$state = 10; //代表对应二进制为： 00001010表示第二个和第四个有效

if(($state & D1) > 0){
    echo "<br/>light 1 on";
}
else {
    echo "<br/> light 1 off";
}

if(($state & D2) > 0){
    echo "<br/>light 2 on";
}
else {
    echo "<br/>light 2 off";
}
```

```
//做一个所有灯的整体显示：
function ShowAll(){
    echo "<p>";
    for($i = 1; $i <= 5; ++$i){
        $s = "D" . $i;
        if( ($GLOBALS['state'] & constant($s)) > 0 ){
            echo "灯{$i}亮， ";
        }
        else{
            echo "灯{$i}灭， ";
        }
    }
    echo "</p>";
}
echo "<br />初始所有灯的状态：";
showAll();
```

```
//需求2a：请打开灯3：
$state = $state | D3;
echo "<br />灯3打开后：";
showAll();
```

```
//需求2b：请打开灯5：
$state = $state | D5;
echo "<br />灯5打开后：";
showAll();
```

```
//需求3，可以关闭指定的任意一个灯泡
//也只要按照如下算法就可以打开：
//$state = $state & (~对应灯的常量值);
//需求3a：请关闭灯2：
$state = $state & (~D2);
echo "<br />灯2关闭后：";
showAll();
//需求3b：请关闭灯5：
$state = $state & (~D5);
echo "<br />灯5关闭后：";
showAll();
```

```
//需求3c：请关闭灯1（其实该灯本来就是关的）：
$state = $state & (~D1);
echo "<br />灯1关闭后：";
showAll();
?>
?>
```

- 数组运算符

有这些：

+: 数组联合，也可以理解为“数组串联”。

将右边的数组项合并到左边数组的后面，得到一个新数组。如有重复键，则结果以左边的为准

```
$arr1 = array(5=>10, 8=>20, 10=>30);
```

```
$arr2 = array(3=>33, 2=>22);
```

```
$r1 = $arr1 + $arr2; //结果为：array(5=>10, 8=>20, 10=>30, 3=>33, 2=>22)
```

另一个有重复键的例子：

```
$arr1 = array(5=>10, 8=>20, 10=>30);
```

```
$arr2 = array(8=>33, 2=>22);
```

```
$r1 = $arr1 + $arr2; //结果为：array(5=>10, 8=>20, 10=>30, 2=>22)
```

= = : 如果两个数组具有相同的键名和键值（可以顺序不同，或类型不同），则返回true

```
$arr1 = array(3=>33, 2=>22);
```

```
$arr2 = array(2=>" 22" , 3=>" 33" );
```

此时，\$arr1和\$arr2是相等的（ = = ）

!=  
== : 如果两个数组具有相同的键名和键值且顺序和类型都一样, 则返回true  
!= =

- 错误控制运算符@ :

通常就用在一个地方:

\$link = @mysql\_connect( "数据库服务器地址", "用户名", "密码" );

作用是: 如果该连接数据的语句失败 (比如连接不上), 则屏蔽该失败的错误提示!

- 运算符的优先级

运算符, 都有优先级问题!

记住以下几条就可以了:

要意识到运算符有优先级问题

括号最优先, 赋值最落后 (通常)

先乘除后加减

大致: 单目运算符) 算术运算符) 比较运算符) 逻辑运算符 (除了 "非" 运算)

能查到手册: 《语言参考》运算符) 运算符的优先级。

- 流程控制

流程图基本符号:

只是人们习惯上使用的一些图形符号, 以代表一定的含义, 帮组别人理解流程过程

流程走向: 

开始结束:

开始:

结束:

语句 (块):

语句块: 可以描述其中要做的事。

判断:

判断: 描述判断的条件

Y

N

输入输出:

输入: xx 数据

输出: xx 数据

- if分支结构

基本语法形式如下:

```
if (条件判断1) {  
    分支1;  
}  
else if (条件判断2) {  
    分支2;  
}  
else if (条件判断3) {  
    分支3;  
}  
.....  
else {  
    //else分支  
}
```

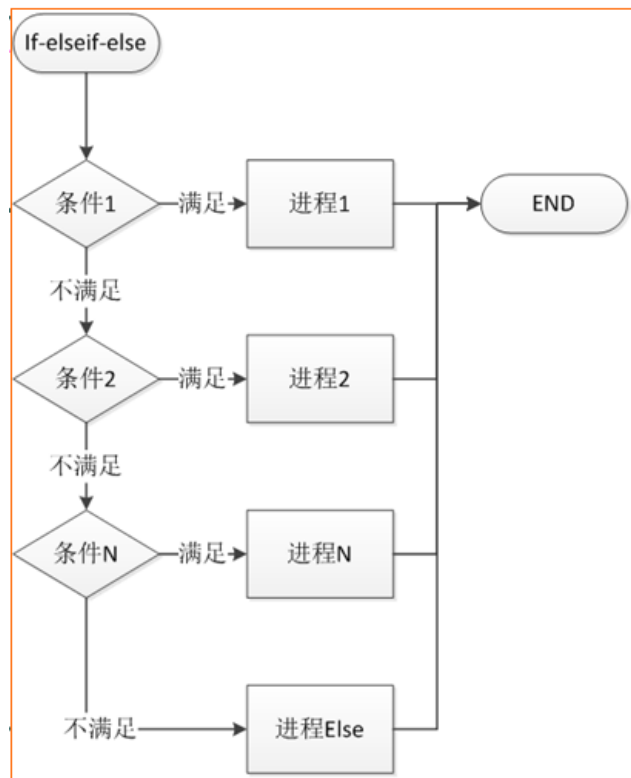
说明：

1，其中，else if部分可以重复若干次，也可以完全省略！

2，其中，else部分可以完全省略。

3，该if语句会从前往后（从上往下）依次判断条件，如果某个条件满足了，就会执行其中对应的分支，然后就结束if分支结构语句！

4，如果前面所有条件都不满足，就会执行最后的else分支（前提是有else分支）。



- 
- switch分支结构

形式：

switch ( 表达式 ) {

case 条件值1 :

分支1 ;

【break ;】 //是可以省略部分，不是语法所必须；

case 条件值2 :

分支2 ;

【break ;】 //是可以省略部分，不是语法所必须；

.....

default :

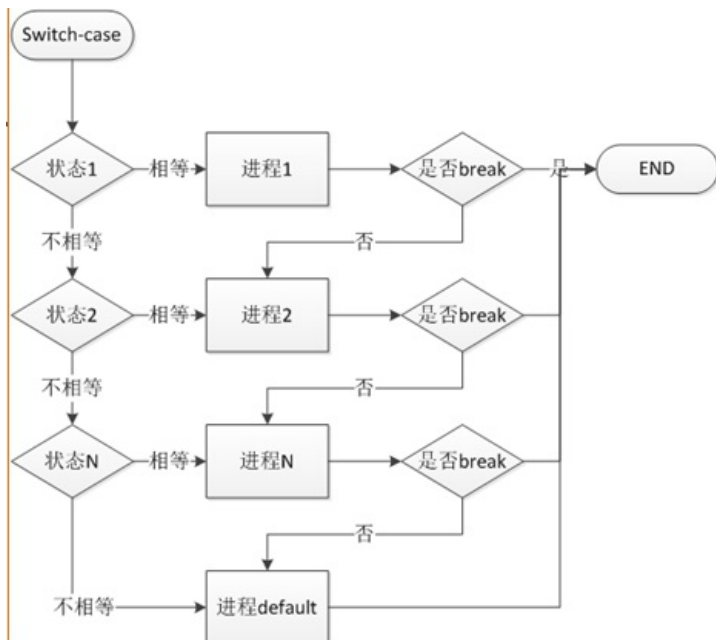
default 分支 ;

}

说明：

1，将表达式的结果数据，跟“条件值1”进行“相等判断”，如果相等，就执行分支1，否则继续对后续值进行判断。。。。

2，如果某个分支判断为相等，则执行该分支语句后，并且如果其中没有break语句，则会直接进入下一个分支继续执行，而不会再去判断下一个分支的条件值了，并直到碰到break语句才会跳出。



示例：

```

<?php
header('Content-type:text/html; charset=utf-8');

$year = 2017;
$month = 7;
//1;
switch ($month) {
    case '1':
        echo "31天";
        break;
    case '3':
        echo "31天";
        break;
    case '5':
        echo "31天";
        break;
    case '7':
        echo "31天";
        break;
    case '8':
        echo "31天";
        break;
    case '10':
        echo "31天";
        break;
    case '12':
        echo "31天";
        break;
    case '2':
        if( ($year % 4 == 0 && $year % 100 != 0) || ($year % 400 == 0)){
            echo "29天";
        }else{
            echo "29天";
        }
        break;
    default:
        echo "30天";
        break;
}

```

```
//2

switch ($month) {
    case '1':
    case '3':
    case '5':
    case '7':
    case '8':
    case '10':
    case '12':
        echo "31天";
        break;
    case '2':
        if( ($year % 4 == 0 && $year % 100 != 0) || ($year % 400 == 0)){
            echo "29天";
        }else{
            echo "28天";
        }
        break;
    default:
        echo "30天";
        break;}
}
```

- for循环结构

```
echo "<br/>";
echo "<hr/>";

for($i = 1; $i <= 9 ; ++$i){
    echo "$i";
    echo "<br/>";
}

for($j = 1; $j <= 9 ; ++$j){
    for($k = 1; $k <= $j; ++$k ){
        echo "*";
    }
    echo "<br/>";
}

echo "<br/>";
echo "<hr/>";

echo "<pre>";
for($j = 1; $j <= 9 ; ++$j){
    for($k = 1; $k <= $j; ++$k ){
        echo "$j * $k = " . ($j*$k) . "\t";
    }
    echo "<br/>";
}
echo "</pre>";
>>
```

[illegible]