第3章 PHP基础

学习 PHP 语言的基本语法是进行 PHP 编程开发的第一步，PHP 语言的语法混合了 C、Java 和 Perl 语言的特点，语法非常灵活，与其他编程语言有很多不同之处，读者如果学习过其他语言，可通过体会 PHP 与其他语言的区别来学习 PHP。

PHP 是运行在服务器端的，而 HTML、CSS、JavaScript 都是运行在浏览器上的。有时也把针对浏览器的网页设计称为 Web 前端开发，而把开发服务器端程序称为 Web 后台编程。

3.1 PHP 语言基础

3.1.1 PHP 代码的基本语法

1. PHP 代码的组成

PHP 是一种可嵌入到 HTML 中的脚本语言。一个 PHP 文件代码可包含如下三部分内容：

(1) HTML 和 CSS。

(2) 客户端脚本(如 JavaScript)，位于< script ></script >之间。

(3) 服务器端脚本，通常位于“<?”与“? >”之间。

其中(1)和(2)是静态网页也具备的，它们都是通过浏览器解释执行，统称为客户端代码。因此，也可以认为 PHP 文件由两部分组成，即客户端代码和服务器端脚本。PHP 可以通俗地认为是把服务器端脚本放在“<?”和“? >”之间。

提示：“<?”和“? >”称为 PHP 脚本的定界符，表示脚本的开始和结束。这是因为在 PHP 文件中，HTML 代码和 PHP 程序代码混杂在一起(即页面和程序没有分离)，必须使用专门的定界符对 PHP 代码进行区分。

2. 简单 PHP 程序示例

(1) 3-1.php：在网页上以 h1 标题的形式输出当前日期和时间。

```
<h1>
<? echo '现在是'.date("Y 年 m 月 d 日 H:i:s");?>
</h1>
```

在该程序中，< h1 >和</h1 >是 HTML 代码，<? … ? >是 PHP 代码。其中，echo 是 PHP 的输出函数，“…”表示这是一个字符串常量，“.”是字符串连接符，date()是时间日期函数，可以按指定的格式获取当前日期和时间。运行程序会在浏览器上以一级标题的形式输出：

```
现在是 2013 年 03 月 18 日 16: 20: 55
```

（2）3-2.php：在网页上输出不同大小的字体，代码如下，运行结果如图 3-1 所示。

```
<html><body>
<?  echo '<p>PHP 代码和 HTML 代码可相互嵌套</p>';
for( $i=3; $i<7; $i++){ ?>
    <font size="<? echo $i;?>">第<? echo $i-2;?>次 Hello World!
</font><br />
<? }?>
</body>
</html>
```

HTML代码嵌入到PHP代码中

PHP代码嵌入到HTML代码中

在 3-2.php 中，使用 for 循环语句循环输出 HTML 代码“< font …>…< br />”。从结构上，这条 HTML 代码被 PHP 代码包含。$i 是程序中定义的一个变量，PHP 规定所有变量名必须以“$”开头。可以看出，PHP 代码可以位于 HTML 代码的任意位置，如标记外：<? for($i=3; $i<7; $i++){ ? >、<? }? >，标记内：<? echo $i-2; ? >，甚至是标记的属性内：<? echo $i; ? >。从结构上看，可以是 HTML 代码中包含 PHP 代码，也可以是 PHP 代码中包含 HTML 代码。实际上，PHP 代码还可与 CSS 或 JavaScript 等浏览器端代码互相嵌入，因为 PHP 解析器只对“<?”和“? >”之间的代码进行处理。

注意：PHP 代码的定界符“<?”和“? >”不能嵌套。如果遇到 HTML 代码（如< font…），就必须立即用“? >”把前面的 PHP 代码结束，即使这段代码并不完整（但其中每行语句必须是完整的）。

（3）3-3.php：用 PHP 程序输出 HTML 代码，实现与 3-2.php 同样的功能。

在 3-2.php 中，由于 PHP 代码和 HTML 代码频繁地交替出现，以致经常需要使用定界符关闭和开始一段 PHP 代码，而如果把 HTML 代码当成字符串通过 PHP 程序来输出，则可避免该问题。代码如下，运行结果如图 3-1 所示。

图 3-1 3-2.php 或 3-3.php 的运行结果

```
<html><body><p>PHP 代码和 HTML 代码可相互嵌套</p>
   <? for( $i=3; $i<7; $i++){
        echo '<font size='. $i .'>第'. ( $i-2) .'次 Hello World!</font><br />';
   }?>
</body></html>
```

提示：使用 PHP 程序输出 HTML 代码是一项常用技巧。总的原则是：如果 PHP 代码之间的 HTML 代码很短，则使用 PHP 程序输出这些 HTML 代码更合适，而如果 PHP 代码之间的 HTML 代码很长，则还是作为外部 HTML 代码合适些。这样会使程序的可读

性改善,并减少编写时出错的概率。

(4) 3-4.php:用 PHP 输出 JavaScript 代码并传递变量值给 JavaScript 或表单。

```
<? $str1 = "Hello";                    //在弹出框中显示
   $str2 = "start PHP";                //在文本框中显示
   echo "<script>";
   echo "alert('". $str1."');"; //在 JavaScript 中使用 $str1 变量
   echo "</script>";?>
<input type = "text" name = "tx" size = 20 value = "<? echo $str1; ?>">
<input type = "button" value = "单击" onclick = "tx.value = '<? echo $str2; ?>'">
```

在该例中,定义了两个变量 $str1 和 $str2,并将字符串赋值给这两个变量(PHP 中没有变量声明语句,变量不需要声明就可使用)。因为 JavaScript 代码也是客户端代码,可以使用 PHP 将 JavaScript 代码作为字符串输出。如果在输出的 JavaScript 代码或表单代码中嵌入了 PHP 变量,就可以把这些服务器端变量值传递到客户端。

运行该程序,会在弹出警告框和文本框中显示 Hello,当单击按钮后,文本框中的内容会变为 start PHP。

3. PHP 代码的 4 种风格

PHP 代码有 4 种风格,即 XML 风格、简短风格、脚本风格和 ASP 风格。使用任意一种都可以将 PHP 代码嵌入到 HTML 中去。

(1) XML 风格。

这种风格的 PHP 定界符是"<? php"和"? >"(<? 和 php 之间不能有空格)。例如:

```
<h1><?php echo '现在是'.date("Y 年 m 月 d 日 H:i:s");?></h1>
```

(2) 简短风格

将定界符<? php 中的 php 省略,就成了简短风格,它的定界符是"<?"和"? >"。要使用简短风格,必须保证 php.ini 文件中的 short_open_tag = On(默认是开启的),本书中的 php 代码都采用这种风格。

(3) 脚本风格。

这种风格将 PHP 代码写在<script>标记对中,例如:

```
<h1><script language = 'php'> echo '现在是'.date("m 月 d 日");</script></h1>
```

(4) ASP 风格。

这种标记风格将 PHP 代码写在"<%"和"%>"中,我们不推荐使用,并且默认是不能使用这种风格的,因为 php.ini 文件中的 asp_tags=Off。

4. PHP 代码的注释

注释即代码的解释和说明,程序执行时,注释会被 PHP 解析器忽略,因此浏览器端看不到 PHP 代码的注释。PHP 支持 3 种风格的注释。

(1) 单行注释(//或#)。

```
<? echo 'PHP 动态网页';          //输出字符串
                                 #单行注释用#号也可以
?>
```

需要注意的是,单行注释的内容中不能含有"?>"否则解释器会认为 PHP 的脚本到此结束了,而去执行"?>"后面的代码。例如:

```
<h1><? echo '这样会出错的';     //不会看到?>会看到
?></h1>
```

(2) 多行注释(/*…*/)。

如果要添加大段的注释,则使用多行注释更方便,但多行注释符不允许嵌套使用。如:

```
<h1><? echo '这样不会出错';     /*多行注释的内容不会被输出?> */
?></h1>
```

5. 编写 PHP 程序的注意事项

(1) PHP 是一种区分大小写的语言,表现在:①PHP 中的变量和常量名是区分大小写的;②PHP 中的类名和方法名,以及一些关键字(如 echo,for)都是不区分大小写的。在书写时,建议除了常量名以外的其他符号都小写。

(2) PHP 代码中的字符均为半角(英文状态下)字符,中文或全角字符只能出现在字符串常量或注释中。

(3) 在"<?"和"?>"内必须是一行或多行完整的语句,如<? for($i=3; $i<7; $i++)?>不能写成<? for($i=3; ?><? $i<7; $i++)?>。

(4) 在 PHP 中,每条语句以";"号结束,PHP 解析器只要看到";"号就认为一条语句结束了。因此,可以将多条 PHP 语句写在一行内,也可以将一条语句写成多行。

3.1.2 PHP 的常量和变量

1. 常量

在程序运行中,其值不能改变的量称为常量,常量通常直接书写,如 10、-3.6、"hello"都是常量,除此之外,还可以用一个标识符代表一个常量,这称为符号常量。在 PHP 中使用 define()函数来定义符号常量,符号常量一旦定义就不能再修改其值。另外,使用 defined()函数可以判断一个常量是否已被定义。例如:

```
<?define("PI","3.1416");                  //定义符号常量 PI,并且区分大小写
define("SITE","网页设计学习网",true);     //定义符号常量 SITE,不区分大小写
echo (defined("PI"));                     //如果已被定义则返回"1"
?>
```

在 PHP 中,还预定义了一些符号常量,如表 3-1 所示(注意__FILE__等常量左右两边是双下画线),这些符号常量可直接使用,如"echo __FILE__;"。

表 3-1 PHP 预定义的符号常量

常　　量	功　　能
__FILE__	存储当前脚本的物理路径及文件名称
__LINE__	存储该常量所在的行号
__FUNCTION__	存储该常量所在的函数名称
PHP_VERSION	存储当前 PHP 的版本号
PHP_OS	存储当前服务器的操作系统名

2. 变量

变量是指程序运行过程中其值可以变化的量,变量包含变量名、变量值和变量数据类型三要素。PHP 的变量是一种弱类型变量,即 PHP 变量无特定数据类型,不需要事先声明,并可以通过赋值将其初始化为任何数据类型,也可以通过赋值随意改变变量的数据类型。下面是一些变量定义(声明)和赋值的例子:

```
<? $str1 = "PHP 变量 1";                    //该变量为字符串变量
 $num = 10 + 2 * 9;                         //该变量为数值型变量
 $_date = "2013 - 9 - 8";                   //该变量为字符串变量,PHP 无日期型数据类型
 $bol = true;                               //该变量为布尔型变量
 $num = '赋值字符串';                        //通过赋值改变变量的数据类型
 $str1 = $num + $_date;
var_dump( $num, $_date, $bol);              // var_dump 函数可输出变量的类型
?>
```

说明:

(1) PHP 变量必须以“$”开头,区分大小写。

(2) 变量使用前不需要声明,PHP 中也没有声明变量的语句。

(3) 变量名不能以数字或其他字符开头,其他字符包括@,# 等;例如:$xm,$_id,$sfzh 都是合法的变量名,而 $-id,$57zhao,$zh fen 都是非法的变量名。变量名长度应小于 255 个字符,不能使用系统关键字作为变量名。

3.1.3 变量的作用域和生存期

1. 变量的作用域

变量的作用域是指该变量在程序中可以被使用的范围。对于 PHP 变量来说,如果变量是定义在函数内部的,则只有这个函数内的代码才可以使用该变量,这样的变量称为“局部变量”。如果变量是定义在所有函数外的变量,则其作用域是整个 PHP 文件,减去用户自定义的函数内部(注意这和 ASP 等其他语言是不同的),称为“全局变量”。例如:

```
<? $a = "全局变量< br >";                   //该变量为全局变量
function fun(){
      echo $a;                              //调用函数也不会输出"全局变量"
      $a = "局部变量、";                    //该变量为局部变量
      echo $a;
}
fun();                                      //输出"局部变量"
echo $a;                                    //输出"全局变量"
?>
```

输出结果为“局部变量、全局变量”。可见函数内不能访问函数外定义的变量。

如果一定要在函数内部引用外部定义的全局变量，或者在函数外部引用函数内部定义的局部变量。可以使用 global 关键字。示例代码(global.php)如下：

```
-------------------------------- global.php ------------------------------
<? $a="全局变量、";
function fun(){
     global $a;                    //为了引用函数外定义的变量$a
     echo $a;
     $a="局部变量、";              //添加 static 试试
     echo $a;                      //输出"局部变量"
     }
fun();                             //调用函数将输出"全局变量、局部变量、"
echo $a;                           //输出"局部变量、"
?>
```

输出结果为“全局变量、局部变量、局部变量、”。

提示：

(1) global 的作用并不是将变量的作用域设置为全局，而是起传递参数的作用；在函数外部声明的变量，如果想在函数内部使用，就在函数内用 global 来声明该变量。

(2) 不能在用 global 声明变量的同时给变量赋值，例如 global $a="全局"是错误的。

(3) global 只能写在自定义函数内部，写在函数外部没有任何用途。

另外，使用 $GLOBALS[]全局数组也能实现在函数内部引用外部变量，例如：

```
<? $a="全局变量、";
function fun(){
     echo $GLOBALS['a'];
     $a="局部变量、";
     echo $a;                      //输出"局部变量"
     }
fun();                             //调用函数将输出"全局变量、局部变量、"
echo $a;                           //输出"全局变量"
?>
```

则输出结果为“全局变量、局部变量、全局变量”。可见 $GLOBALS[]和 global 是有区别的，它只能在函数内部引用外部变量，但不能在函数外部引用函数内部定义的局部变量。

2. 变量的生存期

变量的生存期表示该变量在什么时间范围内存在。全局变量的生存期从它被定义那一刻起到整个脚本代码执行结束为止；局部变量的生存期从它被定义开始到该函数运行结束为止。

可见，一般的局部变量在函数调用结束后，其存储的值会自动被清除，所占的存储空间也会被释放。为了能在函数调用结束后仍保留局部变量的值，可使用静态变量，这样当再次调用函数时，又可以继续使用上次调用结束后的值。静态变量使用 static 关键字定义。例如：

```
<? function Test() {
     static $w = 0;          //声明静态变量$w
     echo $w;
     $w++; }
Test();Test() ;Test() ;Test() ;Test() ;
?>
```

程序的输出结果为01234。而如果去掉程序中的static,则运行结果为00000。

提示:

(1) 静态变量仅在局部函数域中存在,函数外部不能引用函数内部的静态变量。例如将global.php中的"$a="局部变量、";"改为"static $a="局部变量、";",则最后一条语句将输出"全局变量"。

(2) 对静态变量赋值时不能将表达式赋给静态变量。如"static $int=1+2;""static $int= sqrt(9);"都是错误的。

表3-2对三种类型的变量进行了总结。

表3-2 变量根据作用域和生存期分类

类 型	说 明
全局变量	定义在所有函数外的变量,其作用域是整个PHP文件,减去用户自定义的函数内部
局部变量	定义在函数内部的变量,只有这个函数内的代码才可以使用该变量
静态变量	局部变量的一种,能在函数调用结束后仍保留变量的值

3.1.4 可变变量和引用赋值

1. 可变变量

可变变量是一种特殊的变量,这种变量的名称不是预先定义的,而是动态地设置和使用。可变变量一般是使用一个变量的值作为另一个变量的名称,所以可变变量又称为变量的变量。可变变量直观上看就是在变量名前加一个$。例如:

```
<? $a = 'b';                    //定义变量$a
  $b = '一个变量<br>';          //定义变量$b
 echo $ $a;                     // $ $a就是一个可变变量,相当于$b
 $b = '变化后';
 echo $ $a;                     //通过可变变量输出变量$b的值
 $a = 'c';
 echo $ $a;                     //相当于输出变量$c的值
?>
```

输出结果是"一个变量< br >变化后"。由于没有给$c赋值,第三条echo语句不会输出任何内容。

2. 引用赋值

从PHP 4.0开始,提供了"引用赋值"功能。即新变量引用原始变量的地址,修改新变

量的值将影响原始变量，反之亦然。引用赋值使得不同的变量名可以访问同一个变量内容。使用引用赋值的方法是：在将要赋值的原始变量前加一个“&”符号。例如：

```
<? $b = 10;
 $a = "hello ";                    //$a 赋值为 hello
 $b = &$a;                         //变量 $b 引用 $a 的地址
echo $a;                           //输出结果为 hello
 $b = "world ";                    //修改 $b 的值，$a 的值将一起变化
echo $a;                           //输出结果为 world
 $a = "cup";                       //修改 $a 的值，$b 的值将一起变化
echo $b;                           //输出结果为 cup
?>
```

引用赋值的原理如图 3-2 所示。引用赋值后，两个变量指向同一个地址单元，改变任意一个变量的值（即地址中的内容），另一个变量值也会随之改变。

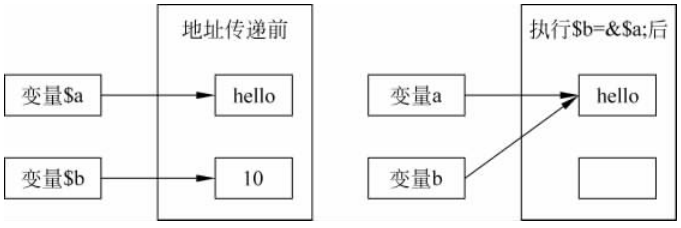

图 3-2 引用赋值——变量地址传递示意图

注意：只有已经命名过的变量才可以引用赋值，例如下面的用法是错误的：

```
$bar = &(25 * 5)
```

3.1.5 运算符和表达式

PHP 中的运算符包括算术运算符、比较运算符、逻辑运算符、赋值运算符、连接运算符等。而表达式就是由常量、变量和运算符组成的，符合语法要求的式子。PHP 主要有 5 种表达式，即：数学表达式（如 3+5 * 7）、字符串表达式（如"abc"."gh"）、赋值表达式（如 $a+= $b）、关系表达式（如 i==5）和逻辑表达式（如 $a||$b&&$c）。

1. 算术运算符

算术运算符有：加（+）、减（-）、乘（*）、除（/）、取余（%）等。算术运算符的运算结果是一个算术值。例如：$a=7/2+4 * 5+1，结果是 24.5。$b=-7%3，结果是-1（对于求余运算符来说，如果被除数是负数，那么取得的结果也是负数）。如果对 10 求余可得到一个数个位上的数字。

如果算术运算符的左右两边有一操作数或两个操作数都不是数值型时，那么会将操作数先转换成数值型，再执行算术运算。

例如：$a=10+'20'，结果为 30。$a='10'+'20'，结果为 30。$a='10'+'2.2ab8'，结果为 12.2。$a='10'+'ab2.2'，结果为 10。$a='10'+true，结果为 11。

字符串转换为数值型的原则是：从字符串开头取出整数或浮点数，如果开头不是数字

的话,就是0。布尔型的True会转换成数值1,False转成数值0。

2. 连接运算符

PHP中字符串连接运算符只有一个,即".",它用于将两个字符串连接起来,组成一个新字符串。如果连接运算符左右两边任一操作数或两个操作数都不是字符串类型,那么会将操作数先转换成字符串,再执行连接操作。例如:

```
$a='PHP'.5;          //$a的值为PHP5,注意数字和.之间要用空格隔开
$b='PHP'.'5';        //$b的值为PHP5
$c="PHP".True;       //$c的值为PHP1
$d=5 . 'PHP';        //$d的值为5PHP
```

提示:如果"."的左右有数字,注意将"."和数字用空格隔开。

可见"."是强制连接运算符,不管左右两边是什么数据类型,都会执行连接运算。

3. 赋值运算符

最基本的赋值运算符是"=",它用于对变量赋值,因此它的左边只能是变量,而不能是表达式。例如:$a=3+5,$b=$c=9都是合法的。此外,PHP还支持像C语言那样的赋值运算符与其他运算符的缩写形式,如"+="".="&="|="等。

如$a+=3等价于$a=$a+3,$a.=3等价于$a=$a.3。

4. 比较运算符

比较运算符会比较其左右两边的操作数,如果比较结果为真,则返回True,否则返回False。PHP中的比较运算符有是否相等(==)、大于(>)、小于(<)、大于或等于(>=)小于或等于(<=)、不等于(!=或<>)、恒等于(===)、非恒等于(!==)。

其中恒等于(===)表示数值相等并且数据类型也相同,非恒等于(!==)表示数值不相等或者数据类型不相等。

例如,若$a=6,$b=3,则$a<$b返回false,$a>$b返回true,$a<>$b返回true。$c="PHP"<"php",返回true。$c="5"==5返回true,$c="5"===5返回false,$c=1==true返回true,$c=1!==true返回true。

5. 逻辑运算符

逻辑运算符用来组合逻辑运算的结果,例如对两个布尔值或两个比较表达式进行逻辑运算,再返回一个布尔值(true或false)。PHP中的逻辑运算符有逻辑非(!)、逻辑与(&&或and)、逻辑或(||或or)、逻辑异或(xor)。

例如:!5<3 && 'b'=="b"返回true,!(5>3 && 'b'==="b")返回false。

逻辑与(&&和and),逻辑或(||和or),虽然含义相同,但它们的优先级却不同。"&&"的优先级比"and"高,"||"的优先级比"or"高。例如:$c=(1 or 2 and 0),会返回true。因为or和and优先级相同,则按自右至左的执行顺序,先执行2 and 0。

而$c=(1 || 2 and 0),会先执行1 || 2,再执行true and 0,最终返回false。

又如 $c=false or 1,返回false,$c=false || 1,返回true。因为"="的优先级比"or"

高，但是比“||”低。

6. 加1/减1运算符

加1/减1运算符与C语言中的加1/减1运算符相同，包括前加(++$a)、后加($a++)、前减(--$a)、后减($a--)4种形式。

前加操作是先加1，再赋值，后加操作是先赋值，再加1。例如：$a=6；$b=++$a，则执行完后，$a=7，$b=7。$a=6；$b=$a++，执行完后，$a=7，$b=6。

前减操作和后减操作的规则与此相同。

7. 条件运算符

条件运算符是一个三元运算符，其语法如下：

```
条件表达式 ? 表达式1 : 表达式2
```

如果条件表达式的结果为true，则返回表达式1的值，否则返回表达式2的值。

例如下面的表达式会得到“Yes”。

```
$c = 10 > 2 ? "Yes" : "No"
```

在分页程序中，常通过条件运算符判断要显示的分页页面，如果获取的分页变量page的值存在，则显示该分页，如果获取不到page变量值，则显示第1页，代码如下：

```
$page = (isset( $_GET['page']))? $_GET['page']:"1";
```

8. 执行运算符

执行运算符，即反引号(`)(键盘上的反引号键在数字1键的左边)。可用来执行Shell命令。在PHP脚本中，将外部程序的命令行放入反引号中，并使用echo()或print()函数将其显示，PHP将会在到达该行代码时启动这个外部程序，并将其输出信息返回，其作用效果与shell_exec()函数相同。例如：

```
<? $output = `dir`;
echo $output;                  //输出当前目录下的内容
echo shell_exec('dir ');       //输出当前目录下的内容，结果同上
?>
```

提示：IIS出于安全性考虑，禁止使用执行运算符，执行运算符只能在Apache中使用。

3.1.6 PHP的字符串

1. 字符串类型

任何由字母、数字、文字、符号组成的零到多个字符的序列都叫做字符串。在Web程序中，经常需要对字符串进行操作，如截取标题、连接字符串常量和变量等。PHP规定字符串的前后必须加上单引号(')或双引号(")，例如："这是一个字符串"、'另一个字符串'、'5'、'ab'、''(空字符串)都是合法的字符串。但单双引号不能混用，如'day"则是非法的。

如果字符串中出现单引号(')或双引号("),则需要使用转义字符(\'或\")来输出,例如:

```
echo 'I\'m a boy';                    //输出结果为 I'm a boy
```

2. 单引号字符串和双引号字符串

单引号表示包含的是纯粹的字符串;而双引号中可以包含字符串和变量名。双引号中如果包含变量名则会被当成变量,会自动被替换成变量值,单引号中的变量名则不会被当成变量,而是把变量名当成普通字符输出。例如:

```
<? $a = 'tang';
 $b = 10;
echo '你好 $a';                       //使用单引号输出 $a
echo '<br>';
echo "你好 $a";                       //使用双引号输出变量
echo "你是第 $b 次光临";              //使用双引号输出变量?>
```

运行结果如下:

```
你好 $a
你好 tang 你是第 10 次光临
```

可以看到,在双引号字符串中,$a 和 $b 被解析成了变量 $a 和 $b 的值。因此建议:如果要书写纯字符串,建议用单引号字符串;如果要对字符串和变量进行连接操作,可以使用双引号字符串,以简化写法,例如:

```
echo "你是第 $b 次光临";              //注意 $b 后面要有个空格
```

等价于:

```
echo '你是第 '. $b .' 次光临';
```

注意:在双引号字符串中,如果变量名后有其他字符的话,要在变量名后加空格,否则 PHP 解析器会认为后面的字符也是变量名的一部分。例如:

```
 $sport = 'basket';
 $hobby = "I like play $sportball.";          //包含变量的错误方法
echo $hobby;
```

则 PHP 解析器认为双引号中的变量是 $sportball,而 $sportball 未定义,视为值为空,因此会输出"I like play ."。为解决这个问题,可以加空格,或用大括号将变量名包含起来。

```
 $hobby = "I like play {$sport}ball.";        //包含变量的正确方法
 $hobby = "I like play $sport ball.";         //包含变量的正确方法
```

双引号比单引号支持更多类的转义字符,双引号支持的转义字符如表 3-3 所示。

表 3-3 双引号支持的转义字符及含义

转义字符	含义	转义字符	含义	转义字符	含义
\n	换行	\t	跳格 Tab	\	反斜杠\
\r	回车	\"	双引号	\$	显示$符号

例如要在双引号字符串中输出$符号、反斜杠和换行符，代码如下：

```
echo "变量\$a='\\t'\n";              //输出结果为:变量$a='\t'(换行)
```

但是换行符会被浏览器当成空格忽略，只有在网页源代码中才能看到换行符的效果。

3. 界定符输出字符串

在 PHP 中，除了可以使用单引号或双引号输出字符串外，还可使用界定符输出字符串或变量，例如：

```
<?$i = '显示该行内容';
echo <<< STD
双引号""可直接输出,\$i同样可以被输出出来.<br>
\$i的内容为:$i
STD;
?>
```

输出结果为：

```
双引号""可直接输出,$i同样可以被输出出来.
$i的内容为:显示该行内容
```

说明：

(1) 程序中的“STD”是自定义的界定符，也可以使用任何其他标识符，只要首尾界定符相同即可。

(2) 开始界定符前面必须有三个左尖括号“<<<”，后面不能有任何空格。结束界定符必须单独另起一行，前后不能有空格或任何其他字符(包括注释符)，否则都会引起语法错误。

(3) 界定符和双引号唯一的区别是界定符中的双引号不需要转义就能显示，因此，如果需要处理大量的内容，同时又不希望频繁使用各种转义字符，则使用界定符更合适。

4. 获取字符串中的字符

在 PHP 中，可以通过给字符串变量加下标的方式获取字符串中的字符。语法为：

```
字符串变量[index]
```

其中，index 指定字符的位置，0 表示第 1 个字符，1 表示第 2 个字符，以此类推。例如：

```
<?$i = 'Tom & Mary';
 echo $i[1] . $i[4];              //输出结果为 o&,因为空格也算一个字符
?>
```

5. 获取字符串的长度

使用 strlen()函数可获取字符串的长度,该函数的参数是一个字符串,例如:

```
<?echo strlen('喜欢 PHP!');?>
```

输出的结果是 8,这是因为每个中文字符占 2 字节,加上后面 4 个英文字符总共占 8 字节。如果要计算中文字符串的长度,可以使用 mb_strlen()函数,例如:

```
<?echo mb_strlen('喜欢 PHP!',"gb2312");    ?>
```

则返回值为 6,将字符编码设置为 GBK 或 gb2312 即可获得正确的中文字符串长度。

3.1.7 PHP 的数据类型和类型转换

数据类型的定义是一个值的集合以及定义在这个值集上的一组操作。定义变量的数据类型就定义了变量的存储方式和操作方法。PHP 中的数据类型如表 3-4 所示。

表 3-4 PHP 中的数据类型

数 据 类 型	具 体 描 述
整型(integer)	即整数,占 4 字节(32 位),取值范围从－2 147 483 648 到 2 147 483 647 之间,可以采用十进制、八进制(0 作前缀)、十六进制(0x 作前缀)表示
浮点型(float)	即实数(包含小数的数),如 1.0、3.14
布尔型(boolean)	只有 true(逻辑真)和 false(逻辑假)两种取值
字符串(string)	是一个字符的序列。组织字符串的字符可以是字母、数字或者符号
数组(array)	由一组相同数据类型的元素组成的数据结构,每个元素都有唯一的编号
对象(object)	是面向对象语言中的一种复合数据类型,对象就是类的一个实例
NULL	空类型,只有一个值 NULL。如果变量未被赋值,或被 unset()函数处理后的变量,其值就是 NULL
资源(resource)	资源是 PHP 特有的一种特殊数据类型,用于表示一个 PHP 的外部资源,例如一个数据库的访问操作,或者打开、保存文件等操作。PHP 提供了一些特定的函数,用于建立和使用资源。
伪类型	只用于函数定义中,表示一个参数可接受多种类型的数据,还可以接受别的函数作为回调函数使用

数据类型的使用往往和变量的定义联系在一起。虽然 PHP 定义变量不需要指定数据类型,但它会根据对变量所赋的值自动确定变量的数据类型。PHP 中数据类型的转换有几种情况。

1. 自动类型转换

(1) 如果将不同数据类型的值赋给同一个变量,则变量的数据类型会自动转换,例如:

```
$a = "Hello";     $a = 12;
```

则变量 $a 的数据类型就会由字符串型转换成整型。

（2）如果不同数据类型的变量进行运算操作，则将选用占字节最多的一个运算数的数据类型作为运算结果的数据类型。例如：

```
$a = 1 + 3.14;
$b = 2 + "2.0";
$c = 3 + "php";
var_dump( $a, $b, $c);                    //输出 float(4.14) float(4) int(3)
```

则 $a 的数据类型为浮点型。在第 2 个赋值表达式中，首先将字符串数据"2.0"转换成浮点型数据 2.0，然后进行加法运算，赋值后 $b 的数据类型为浮点型。在第 3 个表示式中，首先将字符串数据转换成整型数据 0，然后进行加法运算，赋值后 $c 的数据类型为整型。

2. 强制数据类型转换

利用强制类型转换可以将数据类型转换为指定的数据类型。其语法如下：

```
(类型名)(变量或表达式)
```

其中类型名包括 int、bool、float、double、real、string、array、object，类型名两边的括号一定不能丢。例如：

```
$a = "2.0";                $b = (int) $a;
$c = (array) $a;           print_r( $c);
```

则 $b 将转换成整型。$c 将转换为数组类型(Array([0] => 2.0)。

虽然强制数据类型转换使用起来很方便，但也存在一些问题，例如字符型数据转换成整型数据该如何转换，整型数据转换成布尔型数据该如何转换，这些都需要一些明确的规定，PHP 为此提供了相关的转换规定，如表 3-5 所示。

表 3-5 PHP 类型转换的规定

源 类 型	目的类型	转换规则
float	integer	保留整数部分，小数部分无条件舍去
Boolean	integer 或 float	false 转换成 0，true 转换成 1
Boolean	string	false 转换成空字符串""，true 转换成字符串"1"
string	integer	从字符串开头取出整数，开头没有的话，就是 0。例如字符串"3M"、"8.6uc"、"x5"会转换成整数 3、8、0
string	float	从字符串开头取出浮点数，开头没有的话，就是 0.0。例如字符串"3M"、"8.6uc"、"x5"会转换成整数 3.0、8.6、0.0
string	Boolean	空字符串""或字符串"0"转换成 false，其他都转换成 true，因此字符串"false"也会转换成 true。
integer float	Boolean	0 转换成 false，非 0 的数都转换成 true
integer float	string	将所有数字转换成字符串，如 12 转换成"12"，3.14 转换成"3.14"
integer float Boolean string	array	创建一个新的数组，第一个元素就是该整数、浮点数、布尔值或字符串

续表

源 类 型	目 的 类 型	转 换 规 则
array	string	字符串"Array"
object	Boolean	没有成员的对象转换成 false,否则会转换成 true

提示：如果用 echo 函数输出布尔值：echo true,会输出字符串"1"；echo false；会输出空字符串。因为任何数据类型输出时都将被转换成字符串。

3.2 PHP 的语句

PHP 的语句可分为条件控制语句、循环控制语句及包含语句,其语法类似于 C 语言。

3.2.1 条件控制语句

在 PHP 中,有 if 语句和 switch 两种条件语句。if 语句又可分为单分支选择 if 语句、双分支选择 if 语句和多分支选择 if 语句三种。

1. 单分支选择 if 语句

一般形式为：

```
if(条件表达式) {
     语句块 }
```

它表示当条件表达式成立时(值为 true),执行“语句块”。例如：

```
if( $ sex == 1) echo "尊敬的先生";
```

如果语句块中包含多条语句,则要使用{}将这些语句包含起来,使它们构成一条复合语句。例如：

```
if( $ a > $ b){
        $ temp = $ a;
        $ a = $ b;
        $ b = $ temp;}
```

2. 双分支选择 if…else 语句

一般形式为：

```
if (条件表达式)
      { 语句块 1 }
else
      { 语句块 2 }
```

表示当条件表达式值为 true 时,执行“语句块 1”,否则执行“语句块 2”。例如：

```
if( $a) $a=0; else $a=1;
```

该语句被称为“开关语句”。即如果$a的值为true或非0,则让$a的值为0,否则让$a的值为1,因此每执行一次都会使$a的值在0和1之间转换。if($a)是if($a==true)的简写形式。

3. 多分支选择 if…elseif…else 语句

一般形式为:

```
if(表达式1)       语句块1
elseif(表达式2)     语句块2
elseif(表达式3)     语句块3
…
else 语句块n
```

它会首先判断表达式1是否成立,如果成立,则执行语句块1,执行完后,直接退出该选择结构,不再判断后面的表达式是否成立。如果表达式1不成立,则再依次判断表达式2到表达式n是否成立,如果成立,则执行对应的语句块i,如果所有表达式都不成立,则执行else后的语句块n。例如要找出3个数中的最大数,程序如下:

```
if( $a<$b) $max= $b;
elseif( $a<$c) $max= $c;
else $max= $a;
```

说明:

(1) if语句还可以嵌套使用,也就是说“语句块”中还可以使用if语句。

(2) if(条件表达式)后一般没有“;”号,如果有“;”,表示if语句的语句块为空语句。

(3) 语句块如果是一条语句则后面一定要有“;”,如果语句块是由{}包含的复合语句,则{}后不要有“;”。

4. switch/case 语句

switch语句是多分支选择if语句的另一种形式,两者可互相转换。在要判断的条件有很多种可能的情况下,使用switch语句将使多分支选择结构更加清晰。一般形式为:

```
switch(变量或算术表达式){
      case(常量1):语句块1
      case(常量2):语句块2
      …
      case(常量n):语句块n
      default: 语句块n+1
}
```

下面的程序根据时间显示不同的问候信息(3-5.php)。

```
<? $a = date(G);              //获取当前时间的小时数
 $a = floor( $a/3);           //将小时数除以 4 并取底
switch ( $a){
      case 2:echo "早上好";break;
      case 3:echo "上午好";break;
      case 4:case 5: echo "下午好";break;
      case 6:echo "晚上好";break;
      default: echo "该睡觉了";break;
}  ?>
```

说明：

(1) case 语句后不能接表示范围的条件表达式，只能接常量。

(2) 各个 case 中的常量必须不相同，如果相同，则满足条件时只会执行前面 case 语句中的内容。

(3) 多个 case 可共用一组语句，此时必须写成“case 4：case 5：”的形式，不能写成“case 4,5：”。

(4) 每个 case 后一般都要有一条 break 语句，这样执行完该 case 语句后就会跳出分支结构，否则，执行完该 case 语句后还会依次执行下面的 case 语句，直到遇到 break 或执行完。

(5) 各个 case 和 default 语句的出现顺序可随意变动。

3.2.2 循环控制语句

循环结构通常用于重复执行一组语句，直到满足循环结束条件时才停止。在 PHP 中，主要有 4 种循环语句，即 for 循环、foreach 循环、while 循环和 do…while 循环。

1. for 循环

for 循环语句是不断地执行循环体中语句，直到相应条件不满足，并且在每次循环后处理计数器。for 语句的一般形式为：

```
for (初始表达式; 循环条件表达式; 计数器表达式)
      { 循环体语句块 }
```

其执行过程为：①执行初始表达式(通常是给循环变量赋初值)；②判断循环条件表达式是否成立，若成立，则执行循环体，否则跳出循环；③执行一遍循环体语句块；④执行计数器表达式(通常是给循环变量计数)；⑤转到第②步判断是否继续循环。

循环可以嵌套，例如要用 for 循环画金字塔，有下面两种写法。

① 写法 1(3-6.php)

```
<div align = "center">
<?
for( $i = 0; $i<5; $i++){
  for( $j = 0; $j<= $i; $j++)
```

```
    echo " * ";
  echo "<br/>";
}
?></div>
```

② 写法 2(3-7.php)

```
<div align = "center">
<?
for( $ i = 0; $ i < 5; $ i++){
   $ a = $ a ." * ";
  echo $ a ."<br/>";
}
>
</div>
```

提示：在对矩阵进行操作时，通常需要双重循环嵌套。

2. foreach 循环

foreach 语句通常用来对数组或对象中的元素进行遍历操作，例如数组中的元素个数未知，则很适合使用 foreach 语句。其一般形式为：

```
foreach (数组名 as $ value)     或者     foreach (数组名 as $ key = > $ value)
      { 循环体语句块 }                          { 循环体语句块 }
```

foreach 语句遍历数组时首先指向数组中第一个元素。每次循环时，将当前数组元素值赋给 $ value，将当前数组索引值赋给 $ key，再让数组指针向后移动直到遍历结束。例如：

```
<?                                                    //3 - 8.php
 $ sports = array( "网球","游泳","短跑","柔道");      //定义并初始化一个数组
echo "我校开展的运动项目有:< br />";
foreach( $ sports as  $ key = > $ value)
     echo  $ key ."::".  $ value . " ";     ?>
```

输出结果为：

```
我校开展的运动项目有:
0:网球 1:游泳 2:短跑 3:柔道
```

3. while 循环

while 语句是前测式循环，即是否终止循环的条件判断是在执行循环体之前，因此循环体可能一次都不会执行。其一般形式为：

```
while (条件表达式) {
     循环体语句块 }
```

例如,要输出一个有三行的 html 表格,可用下面的程序,它的运行结果如图 3-3 所示。

```
<table border="1" width="300" align="center">
<?$i=0;
while($i<3){
      echo "<tr><td>这是第$i行</td></tr>";  //输出表格行
      $i++;
?></table>
```

图 3-3 while 语句的应用

4. do…while 循环

do…while 语句是后测式循环,它将条件判断放在循环之后,这就保证了循环体中的语句块至少会被执行一次,在某些时候这是非常有用的。其一般形式为:

```
do {
      循环体语句块 }
while (条件表达式);                    //注意 while (…)后有;号
```

例如下面的程序会输出段落<p>元素一次:

```
<?$i=0;
do{
      echo "<p>不满足循环条件,仍然会输出一次</p>";
      $i++;}
while($i>1);  ?>
```

想一想:如果将 while($i>1)改成 while($i>0),程序会循环多少次?

5. break 语句

break 语句用来提前终止循环,它可以出现在 while、do…while、for、foreach 和 switch 语句的内部,用来跳出循环语句或 switch 语句。在“穷举法”解题时,通常找到解后就用 break 终止循环。例如要输出一个字符串,各元素之间用“,”号隔开,最后一个元素后没有“,”号,代码如下(break.php):

```
<?$sports=array("网球","游泳","短跑","柔道");
for($i=0;$i<4;$i++){
       echo $sports[$i];
       if($i==3) break;     //最后一个元素不输出",",换成 continue 试试
       echo ",";
}  ?>
```

输出结果为:

```
网球,游泳,短跑,柔道
```

提示:在 PHP 中,break 后还可带参数 n,表示跳出 n 层循环,如“break 2;”会跳出 2 层循环,而其他语言的 break 语句一般不能带参数,只能跳出最近的一层循环。

6. continue 语句

continue 语句用来提前结束本次循环，即不再执行本次循环中 continue 语句后的语句，接着再执行下次循环，因此它不会提前终止循环。例如要用单个循环输出一个三行三列的表格，代码如下(3-9.php)，运行结果如图 3-4 所示。

图 3-4 continue 语句的应用

```
<table border="1" width="200" align="center"><tr>
<?$i=0;
while($i<9){
      echo "<td>第$i格</td>"; //输出表格的单元格
       $i++;
      if($i%3<>0||$i==9) continue;
      echo "</tr><tr>";
}  ?>
</tr></table>
```

提示：如果正好是最后一次循环时用 continue 语句结束本次循环，那就相当于提前终止循环，这种情况下 continue 和 break 可互换。如 break.php 中的 if($i==3) break 可换成 continue，因为 $i=3 时正好是最后一次循环。

3.2.3 文件包含语句

为了提高代码的可重用性，通常会将一些公用的代码放到一个单独的文件中，然后在需要这些代码的文件中，使用包含语句将它们引入。PHP 提供了 4 种形式的包含语句。

1. include 语句

使用 include()语句可以在指定的位置包含一个文件，语法如下：

```
include(path/filename);                    //括号可省略
```

当一个文件被包含时，编译器会将该文件的所有代码嵌入到 include 语句所在的位置。下面是一个例子，文件 file1.php 和 file2.php 位于同一目录下，在 file2.php 中使用 include 语句将 file1.php 包含到其中来。

```
<?     //file1.php
 $name='tangsix';
 $age=33;          ?>

<?     //file2.php
echo "我的名字是 $name<br>";
include('file1.php');                  //也可嵌入 file3.html
echo "我的名字是 $name,我今年 $age 岁.";
?>
```

则编译器在执行 file2.php 前，会把 file1.php 的代码嵌入到 file2.php 中，即 file2.php 等价于：

```
<?    //file2.php 嵌入 file1.php 之后
echo "我的名字是 $name<br>";
$name='tangsix';                    // 这两行是嵌入的 file1.php 的代码
$age=33;
echo "我的名字是 $name,我今年 $age 岁.";?>
```

输出结果为：

```
我的名字是
我的名字是 tangsix,我今年 33 岁.
```

include 语句也能包含 html 文件，这时 include 语句会自动使用“?>”将前面的 php 代码结束，用“<?”开始后面的 php 代码。例如将 file2.php 文件中 include('file1.php')换成 include('file3.html')。file3.html 代码如下：

```
<!--file3.html-->
<center>&copy; 程序员实验室 版权所有</center>
```

则 file2.php 等价于：

```
<?    //file2.php 嵌入 file3.html 之后
echo "我的名字是 $name<br>";?>
<center>&copy; 程序员实验室 版权所有</center>
<?echo "我的名字是 $name,我今年 $age 岁.";
?>
```

提示：如果被包含文件与包含文件不在同一目录中，则需要在被包含文件文件名前加路径。有三个特殊的路径：“../”代表上一级目录，“./”代表当前目录，“/”代表网站根目录(不是虚拟目录)。例如 include('../file4.php')；将包含位于当前目录上一级目录中的 file4.php。

2. include_once 语句

include_once 语句与 include 语句相似，也是用于包含文件。唯一区别是，使用 include_once 包含文件时，如果该文件中的代码已被包含过，则不会再次包含。这样可以避免出现函数重定义、变量重新赋值等问题。

3. require 语句

require 语句也是用于包含文件，但与 include 语句在错误处理上的方式不同。当包含文件失败时(如包含文件不存在)，require 语句会出现致命错误，并终止程序的执行，而 include 语句只会抛出警告信息并继续执行程序。

4. require_once 语句

require_once 语句与 require 语句功能相似。但它在包含文件之前会先检查当前文件代码是否被包含过，如果该文件已经被包含了，则该文件会被忽略，不会被再次包含。

一般来说，由于 include 语句在出错时不会终止程序的执行，并显示警告信息，这容易产生安全问题，因此建议包含 PHP 程序文件时尽量使用 require 或 require_once 语句。

3.3 数组

数组是按一定顺序排列，具有某种数据类型的一组变量的集合。数组中的每个元素都是一个变量，它可以用数组名和唯一的索引（又称“下标”或“键名”）来标识。

3.3.1 数组的创建

1. 使用 array()函数创建数组

PHP 的数组不需要定义，可以直接创建。创建数组一般使用 array()函数，假设要创建一个包含 4 个元素的一维数组，简单形式是：

```
$citys = array( "长沙","衡阳","常德","湘潭");
```

则该数组的长度为 4。各个数组元素的索引值分别为：0、1、2、3，如 $citys[1]表示第 2 个数组元素。

如果要自行给每个元素的索引赋值，可以使用完整形式定义：

```
$citys = array('cs' =>'长沙','hy' =>'衡阳','cd' =>'常德','xt' =>'湘潭');
```

可见，完整形式增加了对数组元素索引的赋值，这时各个数组元素的索引值分别为：cs、hy、cd、xt，如 $citys[hy]表示第 2 个数组元素（注意此时不能再使用 $citys[1]访问该数组元素）。

2. 直接给数组元素赋值创建数组

也可以创建一个空数组，然后再给每个数组元素赋值。例如：

```
$citys = array( );                  //该句可省略
$citys[1] = "长沙";   $citys[3] = "常德";   $citys[ ] = "湘潭";
print_r ( $citys);                  //打印数组
```

上述代码输出结果为：

```
Array ( [1] => 长沙 [3] => 常德 [4] => 湘潭 )
```

可见，如果给数组元素赋值时不写该元素的索引值，则该数组元素默认的索引值为数组中最大的索引值加 1。如果数组中没有正整数形式的索引值，则默认的索引值为 0。

实际上创建空数组的语句也可省略，那样就是通过直接给数组元素赋值创建数组了，但不推荐这样做。

3. 创建数组注意事项

（1）如果数组元素的索引是一个浮点数，则下标将被强制转换为整数，如 $citys[3.5]

将转换成$citys[3]。如果下标是布尔型数据,则将被强制转换成1或0,如果下标是一个整数字符串,则将被强制转换成整数。$citys["3"]将转换成$citys[3]。

(2) 如果数组元素的索引是字符串,则最好要给下标加引号,如$citys=array('cs'=>'长沙',…)不要写成$citys=array(cs=>'长沙',…)。$citys['cs']="长沙"不要写成$citys[cs]="长沙"。否则程序的运行效率将大打折扣。

(3) 与其他语言相比,PHP数组具有如下特点:

① 数组索引既可以是整数,也可以是字符串。如果索引值是整数,则称为索引数组,如果索引值是字符串,则称为关联数组。如果既有整数又有字符串,则称为混合数组。

② 数组长度可以自由变化。

③ 同一数组中各元素的数据类型可以不同,甚至数组元素可以又是数组。例如:

```
$hybrid=array("长沙",0731,true,array("天心区","雨花区","芙蓉区"));
```

输出结果为:

```
Array ( [0] => 长沙 [1] => 473 [2] => 1 [3] => Array([0] => 天心区 [1] =>雨花区 [2] => 芙蓉区) )
```

3.3.2 访问数组元素或数组

1. 访问数组元素

数组元素也是变量,访问单个数组元素最简单的方法就是通过"数组名[索引]"的形式访问。例如:$i=$citys[3]、echo $citys[1]。也可以使用大括号访问数组元素,例如:echo $arr{3}。

如果要访问所有数组元素,可以使用foreach语句遍历数组。

2. 添加、删除、修改数组元素

如果给已存在的数组元素赋值,将修改这个数组元素的值,给不存在的数组元素赋值,将添加新的数组元素,而要删除数组元素,一般使用unset()方法。例如:

```
<?    $arr = array(11,22,33,44);
  $arr[0]=66;                          //修改数组元素
  $arr[1]='长沙';                      //修改数组元素
 unset($arr[2]);                       //删除数组元素
  $arr[]=55;                           //添加数组元素
  $arr[5]=88;                          //添加数组元素
 print_r ($arr);  ?>
```

运行结果为:

```
Array ( [0] => 66 [1] => 长沙 [3] => 44 [4] => 55 [5] => 88 )
```

注意删除的元素下标不会被新添加的数组元素占用。

3. 访问数组

数组名代表整个数组，将数组名赋值给变量能够复制该数组，数组名前加“&”表示该数组的地址，数组同样支持传值赋值和传地址赋值。例如：

```
<? $ citys = array('长沙','衡阳','常德','湘潭');
 $ urban = $ citys;                    //复制数组(传值赋值)
 $ urban[1] = '娄底';                  //修改新数组元素的值
print_r ( $ citys);                    //打印原数组
print_r ( $ urban);                    //打印新数组
//下面为传地址赋值
 $ loc = & $ citys;                    //引用复制数组(传地址赋值)
 $ loc[1] = '郴州';                    //修改新数组元素的值
print_r ( $ citys);                    //打印原数组
print_r ( $ loc);                      //打印新数组
?>
```

输出结果为：

```
Array ( [0] => 长沙 [1] => 衡阳 [2] => 常德 [3] => 湘潭 )
Array ( [0] => 长沙 [1] => 娄底 [2] => 常德 [3] => 湘潭 )
Array ( [0] => 长沙 [1] => 郴州 [2] => 常德 [3] => 湘潭 )
Array ( [0] => 长沙 [1] => 郴州 [2] => 常德 [3] => 湘潭 )
```

可见引用赋值会使新数组和原数组指向同一个数组的存储区，修改新数组的值，原数组的值也随之改变。

3.3.3 多维数组

创建多维数组同样有两种方法，一是使用 array()函数，二是直接给数组元素赋值。例如要创建一个如下的二维数组：

"玫瑰"	"百合"	"兰花"	
"苹果"	"香蕉"	"葡萄"	"龙眼"

使用 array()函数创建的代码如下：

```
$ arr = array(array("玫瑰","百合","兰花"),array("苹果","香蕉","葡萄","龙眼") );
```

由于这个语句没有给索引赋值，默认的索引如下：

[0][0]	[0][1]	[0][2]	
[1][0]	[1][1]	[1][2]	[1][3]

要访问二维数组的元素，可以使用“数组名[索引 1][索引 2]”的形式访问。例如：

```
echo $arr[1][2];                    //访问数组元素,输出葡萄
$arr[0][3]="茉莉";                  //添加数组元素
$arr[1][3]="桂圆";                  //修改数组元素
unset($arr[1][0]);                  //删除数组元素
```

输出结果为：

```
葡萄 Array ( [0] => Array ( [0] => 玫瑰 [1] => 百合 [2] => 兰花 [3] => 茉莉 ) [1] =>
Array ( [1] => 香蕉 [2] => 葡萄 [3] => 桂圆 ) )
```

3.3.4 操作数组的内置函数

PHP提供了大量对数组操作的内置函数，可以对数组进行统计、快速创建、排序等操作。

1. count()函数

count()函数可返回数组中元素的个数，语法格式为：int count(array arr[, int mode])。例如：

```
<?$citys=array('长沙','衡阳','常德','湘潭');
echo count($citys);                 //输出 4
$arr = array(array("玫瑰","百合","兰花"),array("苹果","香蕉","葡萄","龙眼"));
echo count($arr);                   //输出 2
echo count($arr,1);                 //输出 9,第 1 维 2 个,第 2 维 7 个,共 9 个
  ?>
```

说明：如果数组是多维数组，则count()函数默认也是统计第一维的元素个数，如果要统计多维数组中所有元素的个数，可以将mode参数的值设置为1。

2. max()、min()、array_sum()函数

max()和min()函数可分别返回数组中最大值元素和最小值元素。array_sum()可统计所有元素值的和。例如：

```
<?$score=array(70,80,92,60);
echo max($score);                   //输出 92
$grade=array('A','C','D','B');
echo min($grade);                   //输出 A
echo array_sum($score);             //输出 302
?>
```

3. array_count_values()函数

该函数用于统计数组中所有值出现的次数，并将结果返回到另一数组中。例如：

```
<? $ level = array(2,1,3,1,2,3,2,4,3,1,4);
 $ tmp = array_count_values( $ level);
print_r( $ tmp); //输出 Array ( [2] => 3 [1] => 3 [3] => 3 [4] => 2 )
?>
```

4. explode()函数

explode()函数通过切分一个具有特定格式的字符串而形成一个一维数组,数组中的每个元素就是一个子串。语法为:explode(separator, string[,limit])。例如:

```
<? $ str = '湖南 湖北 广东 河南';
  $ arr = explode(" ", $ str);              //通过切分生成数组 $ arr
 print_r( $ arr);
?>
```

输出结果为:

```
Array ( [0] => 湖南 [1] => 湖北 [2] => 广东 [3] => 河南 )
```

说明:

(1) explode 的分隔符可以是空格等一切字符,但不能为空字符串"";

(2) limit 参数表示所返回数组元素的最大个数。

5. implode()函数

implode()使用连接符将数组中的元素连接起来形成一个字符串。它实现了与 explode()相反的功能。例如:

```
<? $ grade = array('A','C','D','B');
 $ link = implode(" -- ", $ grade);
echo  $ link;                        //输出 A -- C -- D -- B
?>
```

6. range()函数

range()函数可以快速创建一个从参数 start 到 end 的数字数组或字符数组。语法为:array range(mixed start, mixed end)。例如:

```
<?   $ score = range(2,5);              //等价于 $ score = array(2,3,4,5);
print_r( $ score);                      //输出 Array ( [0] => 2 [1] => 3 [2] => 4 [3] => 5 )
 $ score = range('D','A');              //等价于 $ score = array('D', 'C', 'B','A');
?>
```

7. 排序函数

数组排序函数如表 3-6 所示。其中,sort 函数按元素“值”的升序对数组进行排序,rsort()函数按元素“值”的降序对数组进行排序;asort()按元素“值”的升序进行排序,并保持元素

“键值对”不变，arsort()按元素“值”的降序进行排序，并保持元素“键值对”不变；ksort()按照索引值升序进行排列，并保持元素“键值对”不变，krsort()按照索引值降序进行排列，并保持元素“键值对”不变。

表 3-6　数组排序函数

函　　数	排 序 依 据	排 序 规 则	“键值对”是否改变
sort()	元素值	升序	是
rsort()	元素值	降序	是
asort()	元素值	升序	否
arort()	元素值	降序	否
ksort()	索引值	升序	否
krsort()	索引值	降序	否
natsort()	元素值	升序	否
natcasesort()	元素值	升序	否
shuffle()	元素值	随机乱序	是

可见，排序函数中有 a 表示 association，表示排序过程中保持“键值对”的对应关系不变。r 表示 reverse，表示按降序进行排序。k 表示 key，表示按照数组元素的“键”进行排序。示例程序如下：

```
<?   $pic = array('img12.gif','img10.gif','img2.gif','img1.gif','img01.gif');
sort( $pic);                              //将 sort 依次换成 asort,rsort,arsort
print_r ( $pic);
?>
```

输出结果依次为：

```
Array ( [0] => img01.gif [1] => img1.gif [2] => img10.gif [3] => img12.gif [4] => img2.gif )
Array ( [4] => img01.gif [3] => img1.gif [1] => img10.gif [0] => img12.gif [2] => img2.gif )
Array ( [0] => img2.gif [1] => img12.gif [2] => img10.gif [3] => img1.gif [4] => img01.gif )
Array ( [2] => img2.gif [0] => img12.gif [1] => img10.gif [3] => img1.gif [4] => img01.gif )
```

natsort()函数是用“自然排序”的算法对数组 arr 元素的值进行升序排序，所谓自然排序是指小数总排在大数前，如 img2.gif 将排在 img10.gif 之前。

```
<?   $pic = array(1 =>'img12.gif','c' =>'img10.gif','b' =>'img2.gif',0 =>'img01.gif');
ksort( $pic);                          //将函数依次换成 krsort,natsort
print_r ( $pic);
?>
```

输出结果依次为：

```
Array ( [0] => img01.gif [b] => img2.gif [c] => img10.gif [1] => img12.gif )
Array ( [1] => img12.gif [0] => img01.gif [c] => img10.gif [b] => img2.gif )
Array ( [0] => img01.gif [b] => img2.gif [c] => img10.gif [1] => img12.gif )
```

8. array_reverse()函数

array_reverse()函数用来对数组元素进行逆序排列,返回逆序后的新数组。例如:

```
<?   $color = array('a' =>'blue','red','green','red');
 $result = array_reverse( $color);
print_r( $result); //返回 Array ( [0] => red [1] => green [2] => red [a] => blue )
?>
```

9. array_unique()函数

array_unique()函数可删除数组中重复的元素,返回没有重复值的新数组。例如:

```
<? $color = array('a' =>'blue','red','b' =>'blue','green', 't' =>'red');
 $result = array_unique( $color);
print_r( $result);                    //输出 Array([a] => blue [0] => red [1] => green)
?>
```

10. 搜索函数

搜索函数用来检查数组中是否存在某个值或某个键名,假设示例数组为 $color=array('a'=>'blue','red','green','red'),则各搜索函数的功能如表 3-7 所示。

表 3-7 数组搜索函数及功能

函 数	功 能	示 例
in_array(mixed target, array arr)	检查数组中是否存在某个值,返回 true 或 false	in_array('red', $color),返回 true
array_search(mixed target, array arr)	检查数组中是否存在某个值,如果存在则返回其对应的索引值,否则返回 false	array_search('blue', $color),返回 a
array_key_exists(mixed key, array arr)	检查数组中是否存在指定的键,返回 true 或 false	array_key_exists(3, $color),返回 false
array_keys(array arr, mixed search)	返回数组中所有的键名,将其保存到一个新数组中; 若指定了 search,则只返回该值对应的键名	array_keys($color),返回 Array([0]=> a [1]=> 0 [2]=> 1 [3]=> 2) array_keys($color,'red'),返回 Array([0] => 0 [1] => 2)
array_values(array arr)	返回数组中所有的值,将其保存到一个新数组中	array_values($color),返回 Array([0]=> blue [1]=> red [2]=> green [3]=> red)

提示:如果想检查数组中是否含有某个键名,可以用 array_search()函数,如果想获取所有匹配的键名,则应该使用 array_keys()函数,并设置 search 参数。

11. 数组和变量间的转换函数

(1) list()函数。

list()函数可以用数组中的元素为一组变量赋值,从而通过数组得到一组变量。语法格式为:

```
void list(var1, var2,…, var n) = array arr
```

list 要求数组 arr 中所有键为数字,并要求数字键从 0 开始连续递增。例如:

```
<? $ str = '湖南 湖北 广东 河南';
 $ arr = explode(" ", $ str);      // $ arr = array ([0] =>湖南 [1] =>湖北 [2] =>广东 [3] =>河南)
list( $ s1, $ s2, $ s3) = $ arr;   // $ s1 = '湖南', $ s2 = '湖北', $ s3 = '广东'
echo $ s1."< br >". $ s2."< br >". $ s3."< br >";  ?>
```

(2) extract()函数。

extract()函数能利用一个数组生成一组变量,其中变量名为数组元素的键名,变量值为数组元素的值。如果生成的变量名和已有的变量名冲突,则可使用其第 2 个参数按一组规则来处理。

(3) compact()函数。

compact()函数利用一组变量返回一个数组,它实现了和 extract()函数相反的功能。数组元素的键名为变量名,数组元素的值为变量值。extract()函数和 compact()函数的示例如下:

```
<?   $ citys = array( "cs" =>"长沙","hy" =>"衡阳",cd =>"常德",xt =>"湘潭");
extract( $ citys);                    // $ cs = '长沙', $ hy = '衡阳', $ cd = '常德', $ xt = '湘潭'
echo $ xt;                            //输出湘潭
 $ newcitys = compact('cs','cd','xt'); //用变量组成数组
print_r( $ newcitys);                 //输出 array([cs] =>长沙 [cd] =>常德 [xt] =>湘潭)
?>
```

12. 数组指针函数

每一个 PHP 数组在创建之后都会建立一个“当前指针”(current),该指针默认指向数组的第一个元素;通过指针函数可获取指针指向的元素值或键名,也可移动当前指针,对数组进行遍历。数组指针函数如表 3-8 所示。

表 3-8 数组指针函数

函 数	功 能
current()	返回当前指针所指元素的“值”
key()	返回当前指针所指元素的“键名”
next()	移动指针使指针指向下一个元素
prev()	移动指针使指针指向上一个元素
end()	使指针指向最后一个元素,并返回当前指针所指元素的值
reset()	使指针指向第一个元素,并返回当前指针所指元素的值
each()	以数组形式返回当前指针所指的元素,该数组有 4 个元素,其中键名为 1 和 value 的元素值为当前元素的值,键名为 0 和 key 的元素值为当前元素的键名

例 3.1 数组指针的操作

```
<?   $citys = array( "cs" =>"长沙","hy" =>"衡阳",cd =>"常德",xt =>"湘潭");
echo key( $citys).' '.current( $citys).' '.next( $citys).' '.next( $citys).'<br>';
echo prev( $citys).' '.end( $citys).' '.reset( $citys).'<br>';
print_r (each( $citys)).'<br>';
?>
```

输出结果为：

```
cs 长沙 衡阳 常德
衡阳 湘潭 长沙
Array ( [1] => 长沙 [value] => 长沙 [0] => cs [key] => cs )
```

例 3.2 数组的遍历

利用 next()函数和循环语句可以遍历数组，以实现和 foreach 语句类似的功能。例如：

```
<? $citys = array( "cs" =>"长沙","hy" =>"衡阳","cd" =>"常德","xt" =>0);
  reset( $citys);
do{
  echo key( $citys).' => '.current( $citys);}
while(next( $citys)!== false);            //不要写成 while(next( $citys));
?>
```

输出结果为：

```
cs => 长沙 hy => 衡阳 cd => 常德 xt => 0
```

提示：上例中 do…while 语句的循环条件不要简写成 next($citys)，因为若某个数组元素的值为空或 0，则值也会被当成 false 处理，导致遇到 0 就会终止循环。

3.4 PHP 的内置函数

PHP 提供了大量的内置函数，用于方便开发者对字符串、数值、日期、数组等各种类型的数据进行处理。内置函数无须定义就可使用，如 date()函数就是 PHP 的一个内置函数。

3.4.1 字符串相关函数

在 PHP 程序开发中对字符串的操作非常频繁。例如用户在注册时输入的用户名、密码以及用户留言等都被当作字符串来处理。很多时候要对这些字符串进行截取、过滤、大小写转换等操作，这时就需要用到字符串处理函数。常用的字符串处理函数如表 3-9 所示。

表 3-9 常用的字符串函数及功能

函　数	功　能	示　例
strlen (string)	返回字符串的长度(中文算两个字符)	strlen ("abc8"),返回 4
trim(string)	去掉字符串两端的空格	trim(" abcd * "),返回" abcd * "

续表

函　数	功　能	示　例
ltrim (string)、rtrim (string)	去掉字符串左边或右边的空格	ltrim("　abcd＊ "),返回"abcd＊　"
substr (string, start, [length])	从字符串的第 start 个字符开始,取长为 length 的子串。如果省略 Length,表示取到字符串的结尾,如果 start 为负数表示从末尾开始截取,如果 length 为负数,则表示取到倒数第 length 字符	substr("2010-9-6",5),返回"9-6" substr("2010-9-6",2,4),返回"10-9" substr("2010-9-6",2,-2),返回"10-9" substr("2010-9-6",-3,3),返回"9-6"
str_replace(find, replace, string, [&count])	替换字符串中的部分字符,将 find 替换为 replace,如果有参数 count,还可获取替换了多少处	str_replace("AB","＊","ABCabc"),返回"＊Cabc"
strtr (string, find, replace)	等量替换字符串中的部分字符,将 find 替换为 replace,如果 find 和 replace 长度不同,则只替换两者中的较小者	strtr("Hilla Warld","ial","eo"),返回" Hello World"(i 替换成 e,a 换成 o)
substr _ replace (string, replace, start, [length])	从字符串的第 start 个字符开始,用 replace 替换长度为 length 的字符,若省略 length,将替换到结尾	substr_replace("ABCabc","＊",3),返回"ABC＊" substr_replace("ABCabc","＊",3,2),返回"ABC＊c"
strtok(string,split)	根据 split 指定的分隔符把字符串分割为更小的字符串	
strpos (string, find, [start])	返回子串 find 在字符串 string 中第一次出现的位置,如果未找到该子串,则返回 false,如果有 start 参数,表示开始搜索的位置	strpos("ABCabc","bc"),返回 4 strpos("ABCabc","bc",5),返回 false
strstr(string,search)	返回从 search 开始,字符串的其余部分。如果未找到所搜索的字符串,则返回 false	strstr("ABCabc","ab"),返回"abc"
strcmp(str1,str2)	返回两个字符串比较的结果。str1 小于 str2,比较结果为－1; str1 等于 str2,比较结果为 0; str1 大于 str2,比较结果为 1	strcmp ("ABC","abc"),返回－1 strcmp ("abc","abc"),返回 0 strcmp ("abc", "aa"),返回 1
strrev(string)	反转字符串	strrev("Hello"),返回"olleH"
str_repeat(string,repeat)	把字符串重复指定的次数	str_repeat(".",6),返回"……"
nl2br(string)	将 string 中的\n 转换为换行标记 < br/>	nl2br("a\nb"),返回"a < br /> b"
strip_tags(string,[allow])	去除字符串中的 HTML、XML、PHP 标记	strip_tags("Hello < b > world! "),返回"Hello world!"
chr(number)	返回与指定 ASCII 码对应的字符	chr(13),返回回车符 chr(0x52),返回"R"
ord(string)	返回字符串中第一个字符的值	ord("h"),返回 104

上述这些字符串函数都严格区分大小写。如果希望不区分则可使用：strpos()大小写不敏感函数是 stripos()，strstr()大小写不敏感函数是 stristr()，str_replace()大小写不敏感函数是 str_ireplace()，strcmp()大小写不敏感函数是 strcasecmp()。另外 strchr()是 strstr()的别名。

除此之外，还有字符串大小写转换函数。①strtolower($str)：字符串转换为小写；②strtoupper($str)：字符串转换为大写，③ucfirst($str)：将函数参数的第一个字符转换为大写；④ucwords($str)：将每个单词的首字母转换为大写。

在表 3-9 的函数中，strpos()函数有查找字符串中是否含有某个特定子串的功能，只要检测其返回值不恒等于 false 即可(注意：不能用返回值是否等于 0 来判断，因为如果特定子串的位置是第 0 个字符，其返回值也为 0)。str_replace()函数除了可替换字符串中的字符外，如果替换后的字符串为空，则能过滤掉被替换字符串中的某些字符。这两个函数的应用示例如下。

例 3.3 对查询关键词描红加粗(str_ireplace()函数的应用)，运行结果如图 3-5 所示。

```
<?  $content = "«Web 标准网页设计与 ASP»";        //假设这是待查询信息
$find = "网页设计";                                //假设这是查询关键词
$out = str_ireplace( $find,"<b style='color:red'>$find</b>", $content);
echo $out."<br>";
?>
```

例 3.4 对用户输入的字符串进行检查并过滤掉非法字符(strpos()函数的应用)。

```
<?
$Patternstr = "黄|黑|走私|发票|枪支|东突";          //定义要过滤的非法字符串集
$Pattern = explode("|", $Patternstr);              //将字符集分割成数组
//print_r( $Pattern);
$inputstr = "黑色黄色东突枪支弹药走私物品增值发票";//假设这是用户输入的字符串
for( $i = 0; $i < count( $Pattern); $i++){         //分别对数组中每个字符串进行查找
  if (strpos( $inputstr, $Pattern[ $i])!== false) {  //如果找到字符集中的某个字符串
$outstr = str_replace( $Pattern[ $i],"", $inputstr); //将该字符串过滤掉
$inputstr = $outstr;}                              //让输入的字符串等于这次过滤后的字
                                                   //符串,以便进行下次过滤
}
echo $outstr."<br>";
?>
```

程序的输出结果为：色色弹药物品增值。

例 3.5 用字符串函数来判断 Email 或 IP 地址的格式是否正确，运行结果如图 3-6 所示。

```
<? $email = "tangsix@163.com";
if (strpos ( $email, "@") && strpos ( $email, ".") && strpos ( $email, "@") < strpos
( $email, "."))
echo "Email 格式正确<br/>";
//判断 IP 地址是否正确,用到了 explode 函数
$IP = "59.51.24.54";
$arr = explode(".", $IP);
if (count( $arr) == 4)
echo "IP 格式正确,IP 前两位为 $arr[0].$arr[1].*.*";
?>
```

图 3-5　例 3.3 的运行结果

图 3-6　例 3.5 的运行结果

3.4.2　日期和时间函数

在 Web 应用程序中，经常需要获取当前的日期时间信息，例如在论坛中要记录发言的日期和时间等，使用 PHP 提供的日期函数能方便地获取日期时间。

1. date()函数

date(string,[stamp])是最常用的日期时间函数，用来返回或设置当前日期或时间。例如：

```
echo date("Y-m-d");                    //输出 2013-04-23
echo date("y年m月d");                  //输出 13年04月23
echo date("h:i:s");                    //输出 10:44:46
```

其中，Y、m、d 等是 date()函数 string 参数中的格式字符，常见的格式字符如表 3-10 所示。除了格式字符外的字符都是普通字符，它们将按原样显示，如“年”“-”等。

表 3-10　date()函数的格式字符及其说明

字　符	说　明	字　符	说　明
Y	以 4 位数显示年	H	以 24 小时制显示小时(会补 0)
y	以 2 位数显示年	G	以 24 小时制显示小时(不补 0)
m	以 2 位数显示月(会补 0)	h	以 12 小时制显示小时(会补 0)
n	以数字显示月(不补 0)	g	以 12 小时制显示小时(不补 0)
M	以英文缩写显示月	i	以 2 位数显示分钟(会补 0)
d	以 2 位数显示日(会补 0)	s	以 2 位数显示秒(会补 0)
j	以数字显示日(不补 0)	t	显示该日期所在的月有几天，如 31
w	以数字显示星期(0～6)	z	显示该日期为一年中的第几天
D	以英文缩写显示星期	T	显示本地计算机的时区
l	以英文全称显示星期	L	判断是否为闰年，1 表示是

提示：PHP 解析器默认采用格林尼治时间，使得调用时间函数与实际时间相差 8 小时。为此，需要设置 PHP 的时区，打开 php.ini 文件，将“; date.timezone”修改为“date.timezone =PRC”即可。

2. getdate()函数

getdate()函数也能返回当前的日期时间，但它会返回各种时间字段到数组中。例如：

```
<?   $today = getdate();
     print_r( $today);                    // $today 是 getdate()函数返回的数组
echo " $today[mon]月 $today[mday]日";   // mon 和 mday 是数组元素的索引
?>
```

其中,print_r()是用于递归打印数组或对象的语句,可以将数组整体输出。运行结果为:

```
Array ( [seconds] => 58 [minutes] => 8 [hours] => 12 [mday] => 26 [wday] => 5 [mon] =>
4 [year] => 2013 [yday] => 115 [weekday] => Friday [month] => April [0] => 1366974538 ) 4
月 26 日
```

3. time()函数

time()函数会返回当前时间的时间戳。所谓时间戳是指从 1970/1/1 日 0:0:0 到指定日期所经过的秒数。例如当前时间为 2013-04-28 11:58:17,则 time()返回的时间戳是 1367146697。因此利用 time()可对时间进行加减。

```
<?   $nextWeek = time() + (7 * 24 * 60 * 60); //1 周 = 7 天 * 24 小时 * 60 分 * 60 秒
echo '现在是:'. date('Y-m-d') ."<br>";
echo '下一周是: '. date('Y-m-d', $nextWeek) ;
?>
```

则输出结果是:

```
现在是:2013-04-28
下一周是:2013-05-05
```

提示:如果 date()函数带有 2 个参数,则可以设置时间。第 2 个参数必须是一个时间戳,它将使 date()返回时间戳设置的时间。例如 date('Y-m-d', 0)将返回 1970-01-01。

4. mktime()函数

mktime()函数会返回自行设置的时间的时间戳。与 date()函数结合使用可以对日期进行加减运算及验证。其语法为:int mktime(时,分,秒,月,日,年)。例如:

```
echo date("Y-m-d",mktime(0,0,0,12,36,2012));
```

表示设置时间为 2012 年 12 月 36 日,则 mktime()会自动校正时间越界,输出结果为:2013-01-05。

如果要在今天日期的基础上加 12 天,可以使用:

```
echo date("Y-m-d",mktime(0,0,0,date(m),date(d)+12));
```

输出结果为:2013-05-08(注:系统当前日期为 2013-04-26)。上述代码中省略了年的参数,因为 mktime()函数的参数可以按照从右至左的顺序省略,任何省略的参数都会被设置为当前时间值。

5. strtotime()函数

strtotime()函数可将日期时间(英文格式)解析为时间戳。其功能相当于date()函数设置时间的逆过程。date()函数(带有两个参数时)可以将时间戳设置为时间,而strtotime()是将时间解析为时间戳。

```
<?  echo strtotime("now");                                    //输出时间戳:1367148939
echo strtotime("+5 hours");                                   //输出加5小时后的时间戳
echo date('Y-m-d',strtotime("+1 week"));                      //利用返回的时间戳设置时间
echo strtotime("+1 week 3 days 7 hours 5 seconds");
?>
```

可见,使用strtotime()函数也可用来对时间进行加减。

6. checkdate()函数

checkdate(月,日,年)函数可判断参数指定的日期是否为有效日期。如果是,就返回true,否则返回false。例如checkdate(10,3,2014)返回true,因为2014/10/3日是存在的。而checkdate(13,3,2012)返回false。

在Web程序开发中,可使用checkdate()对用户输入的日期格式合法性进行检查。

3.4.3 检验函数

检验函数用来检查变量是否定义、是否为空,获得变量的数据类型,取消变量定义等。

1. isset()函数

isset($var)函数用来检查变量$var是否定义。该函数参数为变量名(带$号),如果变量已经定义,并且其值不为NULL,则返回true,否则返回false。

```
<?  echo isset($test);              //返回false,输出空字符串
    $test=null;
    echo isset($test);              //仍然返回false   ?>
```

通俗地说,如果有这个变量,则isset($var)返回true,否则返回false。

2. empty()函数

empty()函数用来检查变量是否为空。所谓变量为空包括两种情况:①变量未定义;②变量的值为""、0、"0"、NULL、FALSE、以及空数组、没有任何属性的对象等。例如:

```
<?  $var=0;
echo empty($var);                   //变量值为0,返回1
echo isset($var);                   //变量存在,且值不为null,返回1
echo empty($str);                   //变量不存在,返回1
?>
```

因此,如果要检测变量是否定义,尽量用isset()方法。

3. unset()函数

unset($var)函数用来取消变量 var 的定义。该函数的参数为变量名,函数没有返回值。需注意的是:如果在某个自定义函数中用 unset()取消一个全局变量,则只是局部变量被取消,而在调用环境中的变量仍将保持调用 unset()之前一样的值。例如:

```
<?   $foo = 'alive';
function destroy_foo() {
    global $foo;
    unset($foo);                    //在函数中删除变量$foo,实际上只是局部变量被删除
}
destroy_foo();
echo $foo;                          //仍将输出 alive
unset ($bar[1]);                    // unset 也能删除数组元素
unset ($foo1, $foo2, $foo3);        // 同时删除多个变量
?>
```

4. gettype()函数

gettype()函数用来返回变量或常量的数据类型,返回值包括 integer、double、string、array、object、unknown type 等。其语法格式为:string gettype (mixed var)。例如:

```
<? $foo = 'bar';
echo gettype($foo).'<br>';              //输出 string
$bar = array("aa",12,true,2.2,"test",50);
echo gettype($bar[1]);                  //输出 integer
?>
```

虽然 gettype()函数可用来获取数据类型,但由于 gettype()函数在内部进行了字符串的比较,所以它的运行速度较慢。建议使用下面介绍的 var_dump()函数和 is_*()函数来代替。

5. var_dump()函数

var_dump()函数用来返回变量或常量的数据类型和值,并将这些信息输出。例如:

```
<?$a = 3.1; $b = '天涯';
var_dump($a,$b);                    //输出$a、$b的数据类型和值
$c = array(1, '2', array("a", "b", "c"));
var_dump ($c);                      //输出$c的数据类型和值
?>
```

输出结果为:

```
float(3.1) string(4) "天涯" array(3) { [0]=> int(1) [1]=> string(1) "2" [2]=> array(3)
{ [0]=> string(1) "a" [1]=> string(1) "b" [2]=> string(1) "c" } }
```

在调试程序时,经常使用该函数查看变量或常量的值、数据类型等信息。

6. is_*()系列函数

is_*()系列函数包括 is_string()、is_int()、is_float()、is_bool、is_null()、is_array()、

is_object()、is_numeric()、is_resource()、is_integer()、is_long()、is_real()等。它们用来判断变量是否为某种数据类型。如果是,则返回 true,否则返回 false。例如:is_string()可以判断变量是否为字符串数据类型,is_int()判断变量是否为整型,而 is_numeric()判断变量是否为数字或由数字组成的字符串。例如:

```
<? $a = 3.1;
echo is_float( $a);                    //返回 true
 $b = '13307473544';                   // $b 是由数字组成的字符串
echo is_numeric( $b);                  //返回 true
?>
```

7. settype()函数

settype()函数可以进行强制数据类型转换。转换规则遵循表 3-5 的规定。其语法格式为:int settype(string var, string type),参数 type 为下列的类型之一:integer、double、string、array 与 object。例如:

```
<? $a = 3.1;
settype( $a, integer);                 //将变量 $a 转换成整型
echo $a;                               //输出 3
 $b = "false";
settype( $b, bool);                    //将变量 $b 转换成布尔型
echo $b;                               //返回 true
?>
```

8. eval()函数

eval()函数可以动态执行函数内的 PHP 代码,该函数的参数是一个字符串,eval()会试着执行字符串中的代码。示例代码如下:

```
<?eval('$a = 5 + 3;');                 //执行赋值语句
  echo $a;                             //输出 8
  eval('var_dump( $a);');              //输出 int(8)
?>
```

虽然 eval()函数非常好用,但是,eval()函数执行代码时效率是十分低的。并且,eval()容易产生安全性问题,在获取表单中用户输入的数据时,应过滤这些数据中的关键词 eval,因为它允许用户去执行任意代码,这是很危险的。

3.4.4 数学函数

数学函数的参数和返回值一般都是数值型,常用的数学函数如表 3-11 所示。

表 3-11 常用的数学函数及其功能

函 数	功 能	示 例
round(val [,int precision])	返回按指定位数四舍五入的数值,如果省略 precision,则返回整数	round(3.41),返回 3 round(3.45,1),返回 3.5

续表

函　数	功　能	示　例
eil(val)	返回大于并最接近 val 的整数	ceil(3.45),返回 4
floor(val)	返回小于并最接近 val 的整数	floor(3.45),返回 3
intval(val)	返回 val 的整数部分	intval('3.6a'),返回 3 intval(3.6),返回 3
abs(num)	返回 num 的绝对值	abs(－3.43),返回 3.43
sqrt(num)	返回数 num 的平方根	sqrt(16),返回 4
pow(base, exp)	计算次方值,base 为底,exp 为幂	pow(2,3),返回 8
log(num[, base])	计算以 e 为底的对数	log(10),返回 2.3025…
exp(num)	返回自然对数 e 的幂次方	exp(10),返回 22026.…
rand(int min, int max)	返回 min 到 max 之间的伪随机数	rand(2,9),返回 2 到 9 之间的整数
srand(int seed)	播下随机数发生器种子	已被淘汰,不建议使用
int getrandmax (void)	返回调用 rand()可能返回的最大值	
sin(arg)等三角函数	包括 sin()、cos()、tan()等	sin(pi()/6),返回 0.5
max(num1, num2, …, numn)	返回若干参数中的最大值	max(2,3,3.5),返回 3.5
min(num1, num2, …, numn)	返回若干参数中的最小值	min(2,3,3.5),返回 2
decbin(num)	十进制数转换为二进制	decbin(6),返回 110
bindec(num)	十进制数转换为二进制	bindec(11),返回 3
dechex	十进制数转换为十六进制	dechex(13),返回"d"
decoct	十进制数转换为八进制	decoct(13),返回"15"
base _ convert (num, from,to)	在任意进制之间转换数字	base_convert('1a',16,10),返回"26"
number_format(num, preci, [point] ,[sep])	格式化数字字符串	number_format(3.142,2),返回"3.14" number_format(1314.5205, 3,"."," "),返回"1 314.521"

3.5 自定义函数及使用

在 3.4 节中学习了很多内置函数,使用这些函数可方便地完成某些功能。但有时候要实现某种功能,却没有现成的内置函数可用,这时就需要自己编写函数来完成这些功能。

3.5.1 函数的定义和调用

函数就像一台机器,这台机器的输入是一些“原料”(对应函数的参数),进行加工后再把“结果”输出(通过 return 语句),函数可以有一个或多个参数,但只能有一个输出。我们在设计函数之前首先要想清楚它的输入和输出。

1. 函数的定义

定义函数的语法如下：

```
function 函数名([形参 1, 形参 2, …, 形参 n]){
     函数体
     [return 返回值]
}
```

其中，function 是 PHP 定义函数的关键字，函数名是自定义函数的名称，必须符合变量的命名规则。参数是函数的输入接口，函数通过参数接收“外部”数据。函数体是函数的功能实现。return 用来返回函数的执行结果，如果不需要返回结果，可以没有 return 语句。

2. 函数的调用

函数调用有三种方式。即：①函数调用语句；② 赋值语句；③函数嵌套调用。

3. 函数调用语句

如果函数没有返回值(无论是否有参数)，通常使用函数调用语句调用函数，形式为：

```
函数名([实参 1, 实参 2, …, 实参 n]);
```

下面是无参函数(左)和有参函数(右)的调用举例，都是用来打印一行字符：

```
<?  function hello(){
  echo "**********";
}
hello();   //调用无参函数
?>
```

```
<?  function hello( $ n, $ star){
   for( $ i = 0; $ i<$ n; $ i++)
        echo  $ star;
}
hello(8,'&'); / * 调用有参函数 * /   ?>
```

例 3.6　设计函数判断手机号码格式是否正确(函数调用语句举例)。

```
<?function isTel( $ tel) {
      if (strlen( $ tel) == 11 && is_numeric( $ tel))
            echo "手机号码格式正确";
      else
            echo "格式不正确,请重新输入";}
  isTel("13388888888");          //调用有参函数
?>
```

4. 赋值语句调用函数

如果函数有返回值，通常使用赋值语句将函数的返回值赋给一个变量。形式为：

```
变量名 = 函数名([实参 1, 实参 2, …, 实参 n]);
```

例 3.7　限制输出字符串的长度(赋值语句调用函数举例)。

函数 Trimtit()的功能是：如果输入的字符串 $ tit 长度大于指定的长度 $ n，则返回截

取的指定长度字符串并加“…”，如果长度小于或等于指定长度，则返回原字符串。

```
<?  function Trimtit( $ tit, $ n){                          //注意函数的输入为两个类型不同的参数
if (mb_strlen( $ tit,'GB2312')> $ n)
          return mb_substr( $ tit,0, $ n,'GB2312')."…";//返回函数值
     else
          return  $ tit;                                    //返回函数值
     }
 $ str = "航空母舰辽宁舰 2012 年完成舰载机着舰";          //测试字符串
 $ out = Trimtit( $ str,14) ;                               //调用函数
echo  $ out;                                                //输出:航空母舰辽宁舰 2012 年完成…
?>
```

说明：

(1) 函数的参数类型可以各不相同，如上例中的 $ tit 是字符串，而 $ n 是数值型。

(2) 函数中只有一条 return 语句会被执行，return 语句以后的函数代码将不会被执行。

(3) mb_strlen()和 mb_substr()分别是 strlen()和 substr()处理中文字符的版本，这两个函数都必须带有指定编码类型的参数，如'GB2312'。如果处理的字符串中有中文，一定要用这两个函数，因为 substr()不仅会把中文当成 2 个字符，在处理某些中文字符时还会产生乱码。

例 3.8 替换特殊字符为字符实体(赋值语句调用函数举例)。

有时用户在表单中提交了一段字符串，这段字符串中可能有回车、空格等特殊字符，由于 HTML 源代码会忽略回车、空格等字符，会导致这些格式丢失，因此有必要将它们用字符实体替代，使这些格式在浏览器中能保留下来，下面是替换特殊字符的函数。

```
<?
function myReplace( $ str){
    $ str  = str_replace("<","&lt;", $ str) ;          //替换<为字符实体 &lt;
    $ str  = str_replace(">","&gt;", $ str);           //替换>为字符实体 &gt;
    $ str  = str_replace(chr(13),"< br >", $ str);     //替换回车符为换行标记< br >
    $ str  = str_replace(chr(32)," ", $ str);     //替换空格符为字符实体  
      return  $ str ;                                  //返回函数值
}
 $ str = "< font color = 'red'> abc </font >";         //测试字符串
echo  $ str.'< br >';
echo myReplace( $ str);
?>
```

输出结果为：

```
abc(红色的)
< font color = 'red'> abc </font >
```

实际上，PHP 提供了内置函数 htmlentities()可以完成自定义的 myReplace 函数的功能，但是 htmlentities()不会将空格替换成字符实体“ ”，而且字符串中如果有中文，使用 htmlentities()会产生乱码。

由于函数的返回值只能有一个，如果要返回多个数值，可以让函数返回一个数组。

例 3.9 设计一个函数,输入是一个整数,输出是这个整数各位上的数字。

```
<?  function aval( $ num){                    // aval()用来求 $ num 各位上的数字
          for( $ i = 0; $ num > = 1; $ i++){
                $ arr[ $ i] = $ num % 10;     //对 10 取余得到个位数
                $ num = $ num/10;}            //除以 10 后十位数变成个位数
          return $ arr;                       // $ arr 保存了各位上的数
   }
   print_r(aval(54262));                      //调用函数,将返回各位上的数字
?>
```

输出结果为:Array([0] => 2 [1] => 6 [2] => 2 [3] => 4 [4] => 5)。

5. 函数的嵌套调用

函数可以嵌套调用,即把函数调用作为另一函数的参数。例如:

```
<? function sum( $ a, $ b){
      return $ a + $ b;}
      echo sum(7,sum(3,5));            //函数作为另一函数的参数调用
?>
```

例 3.10 过滤字符串中的 HTML 标记。

有时需要把文本中的 HTML 标记都过滤掉,过滤的思路是:首先找到第 1 个 HTML 标记的开始和结束位置("<"和">"),将"<"左边的字符与">"右边的字符连接在一起,这样就去掉了第 1 个 HTML 标记,再把过滤后的字符串赋值给原字符串,进行下次过滤,直到文本中找不到 HTML 标记为止。

```
<?                                        // right 函数:截取字符串 $ s 右边的 $ n 个字符
function right( $ s,  $ n) { return  $ n? substr( $ s,  - $ n): ''; }
function noHtml( $ str){                  // noHTML 函数:去除字符串 $ str 中的 HTML 标记代码
while (strpos( $ str,'<')!== false || strpos( $ str,'>')!== false) {//如果字符串中有"<"或">"
   $ begin = strpos( $ str,'<');          //找到"<"符的位置
   $ end = strpos( $ str,'>');            //找到">"符的位置
   $ length = strlen( $ str) - $ end - 1;  //">"符右边的字符串长度
                                          //将"<"符左边的字符串和">"符右边的字符串连接在一起
   $ filterstr = substr( $ str,0, $ begin) . right( $ str, $ length);  //在函数体内调用另一函数
      $ str =  $ filterstr;               //把一次过滤后的字符串赋给原字符串,以便进行下次过滤
}
return $ str;                             //返回函数值
}
 $ str = "< font size = 9 > abc </font >";   //测试字符串
echo noHtml( $ str);                      //输出结果为"abc"
?>
```

实际上,PHP 提供了内置函数 strip_tags()可实现 noHtml()函数的功能。

3.5.2 变量函数和匿名函数

变量函数类似于可变变量,它的函数名为变量。使用变量函数可实现通过改变变量值

的方法调用不同的函数。例如在例 3.10 的“?>”前插入如下代码：

```
$func = 'noHtml';                    //将一个函数名赋值给变量
echo $func($str);                    //相当于 echo noHtml($str),输出结果为"abc"
$func = 'right';
echo $func($str,7);                  //相当于 echo right($str,7),输出结果为"</font>"
```

可见，当某个变量名后有小括号时，PHP 就会试着去找这个变量的值，然后去运行和该值同名的函数。但变量函数不能用于语言结构，如变量值不能为 echo、print、isset、empty、include、require 等。

在 PHP 5.3 以上版本中，开始支持匿名函数。匿名函数就是没有函数名的函数，例如：

```
<? $greet = function($name){ //定义匿名函数,并将其赋给变量 $greet
   echo 'hello '.$name;};
   $greet('World'); //调用匿名函数,输出 hello World
   $greet('PHP');
?>
```

可见，为了调用匿名函数，常将匿名函数赋给一个变量，那么该变量就相当于函数名。但使用匿名函数更重要的原因，是为了实现函数的闭包。

3.5.3 传值赋值和传地址赋值

函数的参数赋值有两种方法，即传值赋值和传地址赋值。

1. 传值赋值

默认情况下，函数的参数赋值采用传值赋值方式，即将实参值复制给形参值。例如：

```
<?  function add1($val){
        $val++;
        return $val;     }
$age = 18;
echo add1($age).' ';
echo add1($age).' ';
echo $age; /* 运行结果为 19 19 18 */
?>
```

上述程序的执行过程是：

(1) 函数只有在被调用时才会执行。因此，程序执行的第一条语句是“$age=18;”，PHP 预处理器为 $age 分配第一个存储空间。

(2) 执行语句 echo add1($age).' ';，此时自定义函数 add1()被调用，PHP 预处理器为函数参数 $val 分配存储空间，将实参值 18 复制给 $val。

(3) $val 进行加 1 运算，使 $val 的值为 19，但 $age 的值仍为 18。

(4) 函数调用结束时，PHP 预处理器回收函数调用期间分配的所有内存，此时 $val 消失。

(5) 第二次调用函数时，又将 $age 的值复制给 $val，因此 $val 的值仍为 18。

2. 传地址赋值

函数的参数也可以使用传地址赋值,即将一个变量的“引用”传递给函数的参数。和变量传地址赋值一样,在函数的参数名前加“&”就能实现传地址赋值。示例代码如下:

```
<?   function add1(&$val){
         $val++;
         return $val;}
$age=18;
echo add1($age).' ';                    //注意传地址赋值时,函数参数不能是常量
echo add1($age).' ';
echo $age;                              /* 运行结果为 19 20 20 */
?>
```

上述程序的执行过程是:

(1) 程序执行的第一条语句是“$age=18;”,PHP 预处理器为 $age 分配第一个存储空间。

(2) 程序执行到“echo add1($age).' '; ”,此时自定义函数 add1()被调用,PHP 预处理器为函数参数 $val 分配存储空间,由于这里是传地址赋值,形参 $val 和变量 $age 都指向同一个变量值 18 的地址,因此 $val 的值变为 18。

(3) 程序执行到“$val++;”时,形参 $val 修改地址中的值为 19,由于变量 $age 也指向该地址,因此变量 $age 的值也变为了 19。

(4) 函数调用结束时,PHP 预处理器回收函数调用期间分配的所有内存,此时 $val 消失。但函数外变量 $age 的值不会改变,仍然为 19。

(5) 第二次调用时,$val 又会修改 $age 指向地址中的值,使 $age 的值变为 20。

可见,使用传值赋值的方式为函数参数赋值,函数无法修改函数体外的变量值;若使用传地址的方法为函数参数赋值,则函数可以修改函数体外的变量值。

但不管使用哪种赋值方式,函数参数(或函数体内变量)的生存周期是函数运行期间,若要延长函数体内变量的生存期,需使用 static 关键字;函数参数(或函数体内变量)的作用域是函数体内有效,若要扩大函数体内变量的作用域,需使用 global 关键字。

3.6 面向对象编程

面向对象编程(Object Oriented Programming,OOP)是一种编程思想,目前在大型应用软件开发中应用非常广泛。

面向对象的程序由对象构成(object),把所有的对象都划分成类(class),每个类定义了一组静态的属性和动态的方法。对象之间通过传递消息互相联系,驱动整个系统来运转。类是具有相同或相似的结构、操作与约束关系的对象组成的集合。对象是对某个类的具体化实例,每个类都是具有某些共同特征对象的抽象。

3.6.1 类和对象

在现实中，任何一个具体事物都可以看作一个对象，例如一个人、一辆汽车、一场电影等都是对象。对象包含属性和方法。将张三这个人看做对象，则该对象具有下列属性和方法。

```
对象:张三{
    属性:姓名、性别、身高、体重、年龄等;
    方法:吃饭、走路、说话等;
}
```

对于对象的属性，我们可以用变量来描述，例如 $age=33 表示张三的年龄是 33 岁。对于对象的方法，则可以用函数来定义，例如：function walk(){…}用来描述走路。

由于现实中很多对象都属于同一类，因此还可以把同一类的对象看成一个类，例如所有的人都属于“人”类。则对象可看成是类的一个实例，例如张三是“人”类的一个实例。因此定义对象前都要先定义类。

1. 类的定义

在 PHP 中，使用 class 关键字可以定义一个类。语法格式为：

```
class 类名{
     定义成员变量
     定义成员函数
}
```

可见，类实际上就是一组静态属性和动态方法的集合，将它们封装在一起就形成了一个类。例如，要定义一个类 Mystr，代码如下：

```
<?  class Mystr{                          //定义 Mystr 类,注意类名后面没有小括号
    var $str;
    function output(){
         echo 'Hello PHP';
    }
}
?>
```

说明：在类定义中，使用关键字 var 来定义成员变量。在定义成员变量时，也可直接对它赋值。

类成员变量又可分为两种，一种是公有变量，用关键字 public 或 var 定义；一种是私有变量，用关键字 private 定义。公有变量可以在类的外部被访问，它是类与其他类或用户交流的接口。用户可通过公有变量向类中传递数据，也可以通过公有变量获取类中的数据。私有变量在类的外部无法访问，以保证类的设计思想和内部结构并不完全对外公开，这就是面向对象中的封装性。

下面是定义公有变量和私有变量的例子(3-10.php)。

```
class userInfo{
      public $userName;
      private $pwd;
      function output(){
        echo $this->userName;
      }
}
```

在类 userInfo 中,使用公有变量来保存用户名,使用私有变量来保存用户密码。

说明:

(1) 类一旦定义后,系统会自动为其创建一个 $this 的伪变量,代表类自身。

(2) 如果类的成员函数中要访问类中的变量或其他函数,必须使用"$this->变量名"或"$this->函数名"访问,例如 $this-> userName。不能简单使用 $userName 来访问,也不能写成 $this-> $userName,更不能写成 userInfo-> userName。

(3) 如果要在类外面访问类中定义的变量和方法,必须先创建该类的对象,然后用"对象名->变量名"或"对象名->方法名"来访问。

(4) "->"是 PHP 中的成员选择运算符。该运算符表示右边的变量或函数属于左边的类或对象。注意区分"->"和"=>","=>"是初始化数组元素时分隔"键"和"值"的符号。

2. 构造函数和析构函数

在定义类时可以在类中定义一个特殊的函数——构造函数,用来执行一些初始化的任务,例如对属性赋初值等。PHP 规定构造函数的名称必须为"__construct"。

例如,在 userInfo 类中定义一个构造函数(3-11.php)。

```
class userInfo{
      public $userName;
      private $pwd;
      function __construct(){              //定义构造函数
            $this->userName = 'Admin';     //为类中的变量赋初值
            $this->pwd = '123';
      }
      function output(){
           echo $this->userName;
      }
}
```

说明:

(1) 构造函数名"__construct"是以两个下画线开头。

(2) 构造函数不能被主动调用,例如"对象名->__construct()"是错误的。只有在使用关键字 new 创建对象时系统才会自动调用构造函数。

与构造函数相对应的是析构函数,析构函数会在某个对象的所有引用被删除或者对象被销毁时执行。也就是说,如果定义了析构函数,则对象在销毁前会调用析构函数。

PHP 规定析构函数的名称为"__destruct()",析构函数不能带有任何参数。

3. 定义对象

对象是类的实例，可以使用 new 关键字来创建对象。定义一个类 userInfo 的对象 $user 的代码如下：

```
$user = new userInfo();
```

则 $user 就是一个对象(类型为 object 的变量)，定义了对象后，就可使用对象来访问类中的成员变量或成员方法。例如：

```
$user = new userInfo();
echo $user->userName;                    //访问类中的变量
$user->output();                         //访问类中的函数
```

注意：

(1) 如果类中的构造函数包含参数，则在创建对象时，也需要提供相应的参数。

(2) 对象只能访问类中的公有变量和函数，如果试图访问类中的私有变量或函数，如 echo $user->pwd，则程序会出错，提示不能访问私有属性。

下面是一个定义类和对象的综合实例(3-12.php)。

```
<? class Person{
     var $name;                          //人的名字
     var $sex;                           //人的性别
     var $age;                           //人的年龄
     function say($word)                 //人有说话的方法
          {echo $this->name.'对你说:'.$word;}
    function run($step)                  //人有走路的方法
          {echo "<br>然后走了".$step."步";}
    }
$p1 = new Person();                      //创建类 person 的对象 $p1
$p1->name = "张三";                      //设置对象的属性,形式为"对象名->属性名"
$p1->say('您好');                        //访问对象的方法,形式为"对象名->方法名"
$p1->run(5);
?>
```

输出结果为：

```
张三对你说:您好
然后走了 5 步
```

4. 操作符“::”

相比伪变量 $this 只能在类的内部使用，操作符“::”更加强大。它可以在没有声明对象的情况下直接访问类中的变量或方法。例如：下面的代码可在类外访问 Person 类的方法。

```
Person::run(8);          //将其放在 Person 类代码的外部,将输出"然后走了 8 步"
```

操作符“::”可用于访问静态变量、静态方法和常量,还可用于覆盖类中的成员变量和方法。其语法格式为:

```
关键字 :: 变量名/方法名/常量名
```

其中关键字可以分为以下三种情况。

(1) 类名:用来调用本类中的变量、常量和方法。

(2) self:用来调用当前类中的静态成员和常量。

(3) parent:用来调用父类中的变量、常量和方法。

5. instanceof 关键字

instanceof 关键字用来检测某个对象是否属于某个类,它返回一个布尔值。例如:

```
echo $p1 instanceof Person          //返回 true
```

3.6.2 类的继承和多态

1. 继承

继承是指子类可以继承一个或多个父类的属性和方法,并可以重写或添加新的属性或方法。通过对已有类的继承,可以逐步扩充类的功能。继承的这些特性简化了对象和类的创建,增加了代码的可重用性。

例如要设计三个类:“动物”类、“人”类和“学生”类。则可以先定义动物类,将动物类作为父类,人类作为子类,通过继承动物类的一些属性和方法就可以简化人类的设计,并可以添加人类的新属性和方法(例如国籍、说话等)。同样地,学生类又可看成是人类的子类。

在 PHP 中,用 extends 关键字可实现类的继承。语法格式为:

```
class 子类名 extends 父类名
{    定义子类的成员变量
     定义子类的成员函数
}
```

提示:PHP 不支持多重继承,即一个子类不能有多个父类。

下面创建了一个类 Students,并使它继承于类 Person,代码如下(3-13.php):

```
<?
class Person{                                    //定义父类
 function __construct( $name, $sex){             //定义构造函数
       $this->name = $name;
       $this->sex = $sex;
       }
 function say(){                                 //定义说话的方法
       echo '我叫:'. $this->name;
       echo ' 性别:'. $this->sex.'<br>';
       }
}
```

```
class Students extends Person{                    //定义子类并继承父类
 public $school;
 function study($scholl){                         //定义上学的方法
       echo '我在'. $scholl.'上学';
       }
}
$student = new Students('小新','男');             //创建一个子类的对象
$student->say();                                  //调用父类的方法
$student->study('石鼓书院');                      //调用子类的方法
?>
```

运行程序,输出结果为:

```
我叫:小新 性别:男
我在石鼓书院上学
```

说明:程序中子类 Students 通过继承父类 Person,调用了父类中的方法和属性,如显式调用了父类中的 say()方法,通过创建对象隐式调用了父类中的构造方法。同时子类也可调用自己定义的方法 study()。

2. 多态

多态好比有一个成员方法让大家去吃饭,这个时候有的人用筷子吃,有的人用勺子吃,还有的人用叉子和勺子一起吃。虽然是同一种方法,但调用时却产生了不同的形态,就是多态。

在面向对象中,多态指多个函数使用同一个名字,但参数个数、参数数据类型不同。调用时,虽然方法名相同,但会根据参数个数或者类型自动调用对应的函数。

多态可通过继承或接口来实现。下面是一个通过继承实现多态的例子(3-14.php)。

```
<?
class Person{                                     //定义父类
 function __construct($name,$sex){                //定义构造函数
        $this->name = $name;
        $this->sex = $sex;
 }
 }
 class Students extends Person{                   //定义 Person 的子类 Students
       public $school;
       function study(){                          //定义上学的方法
            echo '我在上学<br>';}
 }
 class dxs extends Students{                      //定义 Students 的子类 dxs
 function study(){                                //定义上学的方法
       echo $this->name.'在读大学<br>';}
 }
 class xxs extends Students{                      //定义 Students 的子类 xxs
 function study(){                                //定义上学的方法
       echo $this->name.'在念小学<br>';}
 }
```

```
function rightstudy( $ obj) {                    //定义函数,该函数不属于任何类
      if ( $ obj instanceof Students)            //如果该对象是 Students 的实例
            $ obj -> study();                    //调用该对象的 study()方法
      else echo '出现错误!< br >';
 }
 $ s1 = new dxs('小新','男');                    //创建 dxs 类的对象 $ s1
 rightstudy( $ s1);
 $ s2 = new xxs('小花','女');                    //创建 xxs 类的对象 $ s2
 rightstudy( $ s2);
 $ s3 = new Students('小文','女');               //创建 Students 类的对象 $ s3
 rightstudy( $ s3);
 ?>
```

运行程序,输出结果为:

```
小新在读大学
小花在念小学
我在上学
```

说明:程序通过继承 Students 类创建了两个子类:dxs 和 xxs。在两个子类及父类中都定义了 study()方法。通过 instanceof 检测对象类型,这样无论增加多少种 Students 类的子类,都能调用到正确的方法,并且不需要对 rightstudy()函数进行修改。

多态使我们将编程的重点放在接口和父类上,而不必考虑对象具体属于哪个类的问题。

虽然不使用多态,而使用条件判断语句判断参数的个数或类型也能使调用函数时自动调用相应的函数,但那样就不得不在函数中多写很多条件语句来判断,并且使不同的功能都集中到一个函数中了。

习 题

一、选择题

1. 下列(　　)PHP 变量的名称是错误的。

A. $5-zhao　　B. $s_Name　　C. $_if　　D. $This

2. 语句"echo 'happy'. 1+2 .'345';"输出结果为:(　　)。

A. 2345　　B. happy3345　　C. happy12345　　D. 运行出错

3. ? : 运算符相当于以下 PHP 语句(　　)。

A. if…else　　B. switch　　C. for　　D. break

4. 语句"for($k=0; $k=1; $k++);"和语句"for($k=0; $k==1; $k++);"的执行次数分别是:(　　)。

A. 无限次和 0　　B. 0 和无限次　　C. 都是无限次　　D. 都是 0

5. 如果要提前离开 for 循环,可以使用语句(　　)。

A. pause　　B. return　　C. exit　　D. break

6. 如果要使程序的运行在循环内跳过后面的语句,直接返回循环的开头,应在循环内

使用下面语句(　　)。

A. goto　　B. jump　　C. continue　　D. break

7. 对于 for($i=100; $i<=200; $i+=3),循环运行结束后,变量$i的值是(　　)。

A. 201　　B. 202　　C. 199　　D. 198

8. 下列(　　)代表无穷循环。

A. for(;;)　　B. for()　　C. foreach(,)　　D. do(1)

9. 如果函数有多个参数,则参数之间必须以下列(　　)符号分开。

A. ,　　B. :　　C. &　　D. ;

10. 如果要从函数返回值,必须使用下列(　　)关键词。

A. continue　　B. break　　C. exit　　D. return

11. 下列关于函数的说法,(　　)是错误的。

A. 函数具有重复使用性

B. 函数名的命名规则和变量命名规则相同,必须以$作为函数名的开头

C. 函数可以没有输入和输出

D. 如果把函数定义写在条件语句中,那么必须当条件表达式成立时,才能调用该函数

12. 如果要在函数内定义函数外也可访问的变量,必须使用(　　)关键词。

A. public　　B. var　　C. static　　D. global

13. 如果想保留函数内局部变量的值,必须使用(　　)关键词。

A. private　　B. var　　C. static　　D. global

14. (　　)函数可用来取得四舍五入的值。

A. ceil　　B. floor　　C. round　　D. abs

15. (　　)函数可以用来取得次方值。

A. sqrt　　B. pow　　C. exp　　D. rand

16. (　　)函数可以用来取得当前的时间信息。

A. getdate　　B. gettime　　C. mktime　　D. time

17. (　　)函数可以将字符串逆序排列。

A. chr　　B. ord　　C. strstr　　D. strrev

18. (　　)函数可以将数组中各个元素连接成字符串。

A. implode　　B. explode　　C. str_repeat　　D. str_pad

19. (　　)函数可以将换行符转换成 HTML 换行标记。

A. nl2br　　B. substr　　C. strcmp　　D. strlen

20. 数组是通过(　　)来区分它所存放的元素的。

A. 长度　　B. 值　　C. 索引　　D. 维度

21. 在默认情况下,PHP 数组中第一个元素的索引是(　　)。

A. 0　　B. 1　　C. 空字符串　　D. 不一定

22. PHP 规定数组的索引可以为以下哪两种形式(多选)?(　　)

A. 布尔　　B. 浮点型　　C. 整数　　D. 字符串

23. (　　)可以用来访问数组的元素。

A. ->　　B. =>　　C. ()　　D. []

24.（　　）运算符可以用来比较两个数组是否不相等。

A. ＋　　B. ！＝　　C. <>　　D. ！＝＝

25. 如果数组 $a=array(0=>5,1=>10), $b=array(1=>15,2=>20), $c= $a+ $b,则 $c 等于下列。

A. array ([0] => 5 [1] => 10 [2] => 20)

B. array ([0] => 5 [1] => 15 [2] => 20)

C. array ([0] => 5 [1] => [2] => 20)

D. array ([0]=> 5 [1]=> 10 [2]=> 15 [3]=> 20)

26. 假设 $a=array(0=>'a',1=>'b'), $b=array(1=>'b',0=>'a'),则 $a== $b 和 $a=== $b 的值分别是（　　）。

A. true true　　B. true false　　C. false false　　D. false true

27. 假设 $a=array('a','b','c','d'),则依次调用 next($a); next($a); next($a); prev($a); 后,current($a)会返回（　　）。

A. 'a'　　B. 'b'　　C. 'c'　　D. 'd'

28. 假设 list($x, $y)= array(10,20,30,25),则 $y 的值是（　　）。

A. 10　　B. 20　　C. 30　　D. 25

29.（　　）函数可以将数组中的索引和值互相交换。

A. array_reverse()　　B. array_walk()　　C. array_flip()　　D. array_pad()

30. 假设 $a=array(10,25,30,25,40),则 array_sum($a)会返回（　　）。

A. array ([0] => 105)　　B. array ([0] => 130)

C. 105　　D. 130

31. 假设 $a=range(1,20,5),则 print_r($a)为（　　）。

A. array (1, 6, 11, 16)　　B. array (1, 20, 5)

C. array (5, 10, 15, 20)　　D. array (5, 10, 15)

32. 假设 $a=array('x','y');,则 $a=array_pad($a,4,'z');,会返回（　　）。

A. array ('x','y','z','z')　　B. array ('z','z','z','z')

C. array ('x','x','x','z')　　D. array ('x','y','z',0)

33.（　　）运算符可以用来访问对象的成员。

A. ：：　　B. =>　　C. ->　　D. .

34.（　　）运算符可以直接访问类内的方法或常量,而无须创建对象。

A. ：：　　B. =>　　C. ->　　D. .

35.（　　）语句可以在子类调用父类的构造函数。

A. base：：__construct()　　B. this：：construct()

C. parent：：__destruct()　　D. parent：：__construct()

36. 关于构造函数的说法,（　　）是错误的。

A. 使用 new 创建对象时会自动运行构造函数

B. 名称只能为__construct

C. 子类会继承父类的构造函数

D. 不可以有参数

37. 如果一个对象的实例要调用该对象自身的方法函数“mymeth”，则应使用(　　)。

A. $self-> mymeth()　　B. $this-> mymeth()

C. $current-> mymeth()　　D. $this：：mymeth()

38. 如果类中的成员声明时没有使用限定字符，则成员属性的默认值是(　　)。

A. private　　B. protected　　C. public　　D. final

39. 在类中定义的析构方法是在(　　)被调用的。

A. 类创建时　　B. 创建对象时

C. 删除对象时　　D. 不会自动调用

40. PHP 中调用类文件中的 this 表示(　　)。

A. 用本类生成的对象变量　　B. 本页面

C. 本方法　　D. 本变量

41. 下关于类的说法，(　　)是错误的。

A. 父类的构造函数与析构函数不会被自动调用

B. 成员变量需要用 public protected private 修饰，在定义变量时不再需要 var 关键字

C. 父类中定义的静态成员，不可以在子类中直接调用

D. 包含抽象方法的类必须为抽象类，抽象类不能被实例化

二、填空题

1. PHP 是_____的缩写，PHP 文件中可包含_____、_____、_____三部分的代码。

2. 当把布尔值转换为整型时，true 会转换成_____，false 转换成_____。当把布尔值转换成字符串时，true 会转换成_____，false 转换成_____。

3. 检测一个变量是否设置需要使用_____函数，检测一个变量是否为空需要使用_____函数。

4. 对变量进行引用赋值时，引用的变量名前必须加_____。

5. 对于用 $arr = array(1, 2, array('h')) 定义的数组，数组元素 'h' 的索引值是_____，count($arr,1)将返回_____。

6. 若要显示“xxxx 年 xx 月 xx 日 星期 x　xx：xx：xx”，应设置 date()函数的参数为_____。

7. substr('abcdef', 1, 3) 的返回值是_____，substr('abcdef',-2) 的返回值是_____。

8. 如果字符串 $a="test"，$b="es"，对 $a 进行处理得到 $b 的方法是_____。

9. 函数 strpos("xxPPppXXpx","pp")的返回值是_____。

10. 实现中文字符串无乱码的截取方法是_____。

11. echo count("abc"); 的输出结果是_____。

12. 对数组进行升序排序并保留索引关系，应使用的函数是_____。

13. 假设网站目录为 E:\news，网站的 admin 目录中的 sh.php 中有包含语句 require 'inc/conn.php';，则应保证文件 conn.php 位于_____目录下，如果将该文件包含命令改成 require '/inc/conn.php';，则应保证文件 conn.php 位于_____目录下。

14. 假设要输出正确的 HTML 代码,下列 PHP 代码中写法正确的有_____。

(1) < ta <? = "b" ? > le border="1">

(2) < ta <? = b ? > le border="1">

(3) < ta <? echo 'b' ? > le border="1">

(4) < p align="<? = "right" ? >">段落</p >

(5) < p align='<? = "right" ? >'>段落</p >

(6) < p <? ="align='right'" ? >>段落</p >

(7) < p <? ='align="right"' ? >>段落</p >

(8) <? 'for($i=1; $i < 5; $i++)'? >

(9) <? for($i=1; $i < 5; $i++) ? >

(10) <? for($i=1;? ><? $i < 5; $i++) ? >

(11) <% for i= 1 to 5 %>

(12) <? = "< table border='1'>" ? >

(13) < font size="<? = 6? >">天

(14) < style > p{ height:<? = 58? > px;}</style >

三、问答题

1. 如果要将一个变量的数据类型由字符串型强制转换成整型,有哪几种方法?

2. 在页面 A 中定义的普通变量 $b 可以在页面 B 中使用吗(页面 A、B 不存在包含关系)?

3. 变量 $this 指的是对象本身,对不对?

4. PHP 允许父类有多个子类,也允许子类有多个父类,对不对?

5. 用 PHP 输出前一天的时间,要求格式为 2006-5-10 22:10:11。

6. 包含文件操作常用的 4 种函数是什么? 各适合应用于哪种场合?

四、编程题

1. 编写 PHP 程序,计算 1～100 之间所有偶数的总和,然后把结果输出出来。

2. 编写程序,在网页上输出一个三角形形式的九九乘法表。

3. 编写程序,使用 while 循环计算 4096 是 2 的几次方,然后输出结果。

4. 编写程序,先声明一个数组{5,8,2,3,7,6,9,1,8,4,3,0},然后输出数组中最大元素和最小元素的索引值。

5. 编写一个实现字符串翻转的函数。

6. 编写一个函数,使用字符串处理函数获得文件的扩展名,如输入 ab.jpg,输出 jpg。

7. 编写一个函数,输入是一个小于 8 位的任意位数的整数,输出是这个整数各个位上的数。要求分别用两种方式实现:①直接在函数内部用 echo 语句输出,函数没有返回值;②用字符串处理函数截取该整数各位上的数。函数的返回值是一个数组,数组中各元素保存了各个位上的数。

8. 编写一个可计算某整数四次方的函数,该函数的输入是一个整数,输出是该数的四次方。然后调用该函数计算 16 的四次方,并输出结果。

9. 编写一个用来判断某整数是否是质数的函数,该函数的输入是一个整数,如果该整数是质数,就返回 true,否则返回 false,然后调用这个函数输出 2～100 内所有的质数。

10. 任意输入一个整数,使用函数的方法判断该数是否为偶数。

11. 编写一个函数,实现以下功能,将字符串"cute_boy"转换成"CuteBoy","how_are_

you"转换成"HowAreYou"。

12. 编写一个函数,输入是5个分数,输出是去掉一个最高分和去掉一个最低分后的平均分。

13. 将3.4.1节中的例3.4改写成函数,即输入是待过滤的字符串和非法字符集,输出是过滤后的字符串,并调用该函数实现例3.4的功能。

14. 编写函数,计算两个文件的相对路径(例如$a='/a/b/c/d/e.php'; $b='/a/b/12/34/c.php';,则计算出$b相对于$a的相对路径应该是../../c/d)。

15. 先根据原理写出下列程序的运行结果,然后上机验证结果是否正确。

(1) 运行结果为:

```
$a = "hello";
$b = &$a;
unset($b);
$b = "world";
echo $a;
```

(3) 运行结果为:

```
$n = 10; $nn = 100;
$a = '$nnn';
$b = "$nnn";
$c = $a.$b;
echo $c;
```

(5) 运行结果为:

```
$c = 5;
$d = 0;
if($c = $d++)
      echo $d;
else
      echo $c;
```

(2) 运行结果为:

```
$str = "true or false;";
if(eval($str))
     echo 1;
else
     echo 0;
```

(4) 运行结果为:

```
    $d = 3; $y = 1;
    while($d > 0) {
     $x = ($y+1) * 2;
     $y = $x;
     -- $d; }
    echo $x;
```

(6) 运行结果为:

```
$str = 'Heng_yang';
$arr = explode('_', $str);
$res = implode('', $arr);
echo $res;
```

16. 写出下列程序的运行结果:

(1) 运行结果为:

```
$num = 2;
$a = 2; $b = 1;
for($i=1; $i<=$num; $i++){
     $s = $s + $a / $b;
     $t = $a;
     $a = $a + $b;
     $b = $t;}
echo $s;
```

(2) 运行结果为:

```
$num =10;
function multiply(){
    $num = $num *10;
}
multiply();
echo $num;
```

(3) 运行结果为:

```
function fun($a){
if($a>1)
   $r = $a * fun($a - 1);
else $r = $a;
return $r;
}
echo fun(3);
```

(4) 运行结果为:

```
function rev($var){
  $res="";
for($i=0, $j=strlen($var); $i<$j; $i++) {
    $res = $var[$i].$res;
     }
  return $res;
}
$tmp="hengyang";
$res=rev($tmp);
echo $res;
```

(5) 运行结果为：

```
$b = 20;
$c = 40;
$a = $b > $c ? ($c - $b) ? 1 :
($b - $c) > 0 : ($b + $c) ? 0 : $b
* $c;
echo $a;
```

(6) 运行结果为：

```
$arr = array(1, 1);
for($i = 2; $i < 20; $i++){
    $arr[$i] = $arr[$i - 1] +
$arr[$i - 2];
}
for($i = 0; $i < count($arr); $i++){
      if ($arr[$i] % 5 == 0){
            echo $arr[$i];
            break;
      }
}
```

(7) 运行结果为：

```
function t($n){
  static $num = 1;
  for($j = 1; $j <= $n; $j++) {
    if($j >= 4 && $j < 15)
       $num++;
    t($n - $j);
       if($j == 20)
         $num--;
    }
    return $num;
}
echo t(5);
```

(8) 运行结果为：

```
for($i = 100; $i < 1000; $i++){
      $a = intval($i /100);
      $b = intval($i / 10) % 10;
      $c = $i % 10;
      if(pow($a, 3) + pow($b, 3) + pow($c, 3)
== $i && $i % 10 == 0)
            echo $i;
}
```

(9) 运行结果为：

```
function fun($n){
  if($n == 3) return 1;
     $t = 2 * (fun($n + 1) + 1);
  return $t;}
echo fun(1);
```