第 2 章　PHP 语言

经过前面预备知识的学习，从本章开始，正式进入 PHP 语言的学习。PHP 作为一种专门用来开发 Web 应用的嵌入式语言，大量借用了 C、C++和 Perl 语言的语法，同时加入了一些其它语法特征，使编写 Web 程序更快更有效。之所以说 PHP 是嵌入式语言，是因为用 PHP 开发的 Web 程序，大多都要在 HTML 文档中插入 PHP 代码，或者使用 PHP 代码生成某些 HTML 文档，以满足 Web 应用的需求和特点。

PHP 一般作为 HTTP 服务器（通常是 Apache）的一个模块运行。这意味着，当用户访问到一个含有 PHP 代码的 Web 页面时，HTTP 服务器就会调用这个模块，通过这个模块来分析并执行该页面的 PHP 代码，最终将执行结果返回给用户。PHP 支持多种数据库，如 MySQL、dBase、MS_SQLServer、Oracle 等。这对于基于数据库的 Web 开发来说是大有裨益的。

PHP 从上世纪 90 年代中期问世以来，已经推出了很多版本，到现在已经是 PHP5。本书所讲述的 PHP 语言，将以 PHP5 为准。本书凡是出现“PHP”的地方，除非特别说明，都将指的是 PHP5。

2.1　基本语法

PHP 的语法和 C、C++等语言的语法很相似，有 C 语言基础的读者，可以非常轻松地掌握 PHP 的基本语法。即便是没有任何语言基础，也是值得庆幸的，那样不会受其它语言的干扰，可以更快速地接受 PHP 的语法。

事实上，PHP 的语法并不复杂，再加上 PHP 提供了大量的预定义函数，使 PHP 开发事半功倍。只要按本书的讲述、一步步地学习下去，再加上自己的一点信心，相信读者会发现 PHP 很容易学习掌握，并且应用起来也很快速方便。 本节，先简单了解一些 PHP 的基本语法。

2.1.1　PHP 分隔符

因为 PHP 是嵌入式脚本语言，需要使用某种分隔符将 PHP 代码和 HTML 的内容区分开来，这里所说的分隔符就是“<?php”和“?>”，它们将 PHP 代码包含在其中，也就是说，所有的 PHP 代码都应该写在“<?php”和“?>”之间。如下代码所示：

```
<p>一个段落</p>                                    //这一行是 HTML，PHP 分析器将会忽略这行代码，不做处理
<?php echo “这段内容由 PHP 代码输出”; ?>      //这一行是 PHP 代码，PHP 分析器将会执行这段代码
<p>另外一个段落</p>
```

注意：凡是在“<?php”和“?>”标记里的内容，PHP 分析器就会认为是 PHP 代码，试着分析并且执行；而在这对标记之外的内容将被忽略。

从上例也可以看出，使用“<?php”和“?>”将 PHP 代码嵌入在了 HTML 中，通过 PHP 代码的执行，将结果和 HTML 代码结合起来，形成一个完整的 Web 页面。

2.1.2 给 PHP 程序添加注释

程序中的注释是指在一个程序文件中，对一个代码块或一条程序语句所作的文字说明，注释是提供给开发人员看的，因此，程序中的注释会被计算机忽略而不会被执行。PHP 中的主要注释风格有：

- 使用符号“//”添加一个单行的注释。
- 使用符号“#”添加一个单行的注释。
- 使用“/*”和“*/”添加一个多行的注释，也可以用来单行注释。

如下代码演示了如何为 PHP 程序添加注释。

```
<?php
echo “Hello World”;//这里是一个单行注释
/*
这里是多行注释
继续注释一行
*/
echo “Hello PHP”;
?>
```

在开发实践中，有时为了调试一个程序，或者保留目前不需要执行的某些代码，经常会将一条语句或整个代码块注释掉。如代码 2-1 所示，在程序中给一些代码添加注释，计算机就不会执行这些代码。

代码 2-1　PHP 程序的注释　2-1.php

```
<?php
echo "test string1";

echo "<br/>";                          //这里添加"<br/>"是为了在 Web 页面上生成新的一行，即换行
echo "<br/>";

//echo "test string2<br/>";
//echo "test string3<br/>";
echo "test string4<br/>";
?>
```

上述程序使用“//”注释了两行 PHP 语句，因此只输出“test string1”和“test string4”，而字符串“test string2”和“test string3”都不会被输出。程序的执行结果如图 2.1 所示。

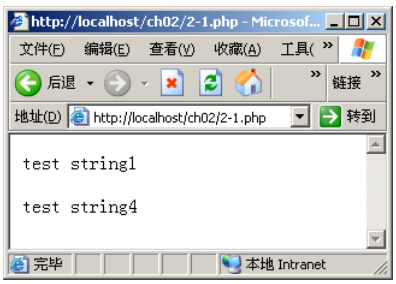

图 2.1　PHP 程序的注释

2.2 变量

变量是任何程序设计语言中一个基本而且重要的概念。本节的内容将讲述 PHP 变量基本概念、变量的类型、PHP 预定义变量以及如何使用 PHP 内置函数对变量进行一些处理。

2.2.1 什么是变量

在程序中可以改变的数据量叫做变量，变量必须有一个名字，用来代表和存放变量的值。PHP 中使用美元符（$）后跟变量名来表示一个变量，如$var 就是一个变量。PHP 中的变量名是区分大小写的，因此$var 和$Var 表示的是不同的两个变量。

PHP 变量的命名需要遵守一定的规则，这个规则是：一个有效的变量名必须由英文字母或下划线开头，后面可以跟任意数量的英文字母、数字、下划线或其组合。如$abc、$_ab_c、$a1b_c2 都是合法的变量名，而$3xyz 就不是合法变量名，因为它以数字开头。

2.2.2 变量的数据类型

PHP 属于弱类型语言。这就是说，变量的数据类型一般不用开发人员指定，PHP 会在程序执行过程中，根据程序上下文环境决定变量的数据类型。如一串数字 “789”，在用 echo 语句输出时，它作为字符串处理，但是做数学运算时，它就作为整数处理。PHP 的变量主要有以下类型：

- 整数类型。
- 浮点类型。
- 字符串类型。
- 布尔类型。
- 数组类型。
- 对象。

下面将介绍除对象之外的数据类型，对象类型将单独放在 14 章做阐述。

1．整数类型：integer

可以在 PHP 中指定的整数包括十进制、八进制和十六进制，整型数值前可以加上符号“+”或“-”。指定方式如下所示：

```
$i = 2468;    //指定一个十进制整数
$i = -1357;   //指定一个负数
$i = 0123;    //指定一个八进制数
$i = 0x456;   //指定一个十六制数
```

对于八进制数，需要在数字前面加上数字 0，对于十六进制数，需要在数字前面加上 0x。

2．浮点数类型：float

PHP 中，浮点数的表示形式有两种：十进制形式和指数形式。浮点数由数字和小数点组成，如 0.1234、1.234 等。下面是一个指定浮点型变量的示例代码。

```
$f = 12.34;      //指定变量$f 的值为 12.34
```

也可以用下面的形式指定指数形式的浮点数。

```
$f = 1.2e3;      //表示将 1.2 乘以 10 的 3 次方指定给变量$f
```

3. 字符串类型：string

一串字符组成一个字符串，如 abcdef 就是一个字符串。在 PHP 中常用双引号（””）或单引号（’’）指定一个字符串。下面例子使用双引号指定一个字符串”PHP string”。

```
$s = “PHP string”;
```

使用双引号指定的字符串，如果字符串中含有变量，那么这个变量将会被其实际内容（即变量的值）替换。如下代码：

```
$v = “string”;
$s = “PHP$v”;
```

其中，由双引号指定的字符串”PHP$v”中含有变量$v，因此$s 的最终结果是：PHPstring。如果希望输出的是$v 本身而不是其变量的值，就需要对特殊符号“$”做转义。在特殊符号“$”前加上反斜杠“\”就可以输出符号“$”本身。如下面的代码所示：

```
$v = “string”;
$s = “PHP\$v”;
```

变量$s 的最终结果将会是：PHP$v。使用反斜杠（\）指定特殊的字符，这叫做字符转义。例如在双引号指定的字符串中，要使用双引号“””，就应该这样写：\”，这样就在字符串中添加了一个双引号。表 2-1 列举了 PHP 主要特殊字符的转义含义。

2-1 转义字符及其含义说明

特殊字符	含义
\r	指定回车符
\n	指定换行符，即生成新的一行
\t	指定水平制表符
\\	指定反斜杠
\$	指定美元符号
\”	指定双引号

在 PHP 中也可以使用单引号指定字符串。与双引号不同的是，单引号指定的字符串，不会对其中的变量用变量的值做替换，也不会对除“\”和“”“之外的字符做转义。代码 2-2 演示了使用单、双引号指定字符串的不同用法，以及如何进行字符转义，从而输出特殊字符。

代码 2-2　PHP 中的字符串和字符转义 2-2.php

```
<?php
$s = "new string";

//下面双引号字符串中的符号"$"未做转义，因此$s 将被替换成其变量的值
$str_1 = "双引号指定的字符串，$s";

//下面双引号字符串中的符号"$"做了转义，因此$s 原封不动，不会被替换为变量$s 的值
$str_2 = "双引号指定的字符串，\$s";

//单引号字符串中的"$"不用做转义即可原样输出
$str_3 = '单引号指定的字符串，$s';

$str_4 = '单引号指定的字符串，\$s';

//在双引号字符串中，输出$、"、\等特殊字符，需要做转义
$str_5 = "在双引号字符串中，输出特殊字符：\$ \" \\ '";
```

```
//在单引号字符串中，输出'，需要做转义
$str_6 = '在单引号字符串中，输出特殊字符：$ " \ \'';

echo $str_1;
echo "<br/>";
echo "<br/>";

echo $str_2;
echo "<br/>";
echo "<br/>";

echo $str_3;
echo "<br/>";
echo "<br/>";

echo $str_4;
echo "<br/>";
echo "<br/>";

echo $str_5;
echo "<br/>";
echo "<br/>";

echo $str_6;
?>
```

上述程序的执行结果如图 2.2 所示。

这里简单提及一下，在 PHP 中通常使用英文句点（.）来连接两个字符串。如代码 2-3 所示。

代码 2-3　PHP 字符串的连接 2-3.php

```
<?php
$s1 = "PHP in ";
$s2 = "Windows";
$s = $s1.$s2;

echo '$s1='.$s1;
echo "<br/>";
echo "<br/>";

echo '$s2='.$s2;
echo "<br/>";
echo "<br/>";

echo '$s='.$s;
echo "<br/>";
echo "<br/>";
?>
```

上述程序中，变量$s 由$s1 和$s2 连接而成，执行结果如图 2.3 所示。

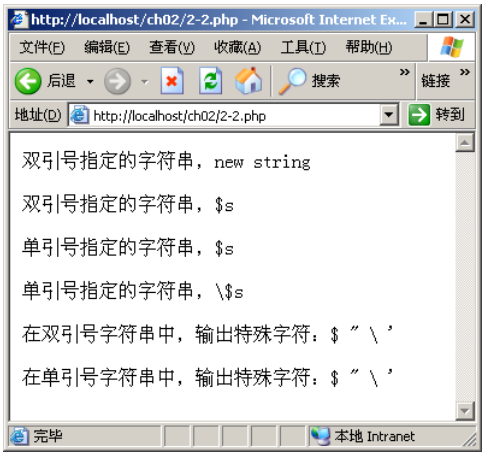

图 2.2　PHP 字符串的用法及字符转义

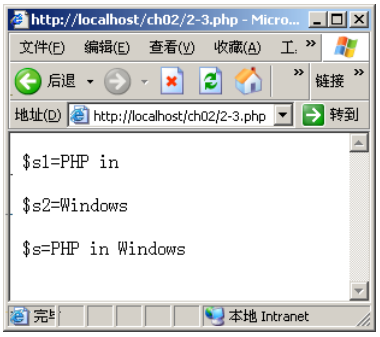

图 2.3　PHP 字符串的连接

4. 布尔类型：boolean

布尔类型是最简单的数据类型，它只有两个取值：TRUE（或 1）或 FALSE（或 0），这两个值都不区分大小写。TRUE（或 1）表示“真”（真值），FALSE（或 0）表示“假”（伪值）。

5. 数组类型：array

前面讲述的都是标量变量，标量变量的含义是，一个被命名的、存储一个数值的空间，而数组是一个被命名的存放一组数值的空间。这里的数值可以是整数、浮点数、字符串，甚至可以是数组、对象等。下面的代码指定了一个数组。

```
$arr = array('spring','summer','fall','winter');
```

在 PHP 中使用 array()语言结构建立一个数组。上面的代码定义了名为$arr 的数组，它包含 4 个值：spring、summer、fall 和 winter。数组中的值叫做“元素”，每个元素和一个“索引”（或称“键”、“下标”）相关联，可以通过“索引”来访问数组元素。例如要输出上例数组$arr 中的元素 spring，可以使用下面的代码：

```
echo $arr[0];
```

输出$arr[0]就会显示字符串 spring。PHP 数组的索引一般从 0 开始计数，要访问$arr 的元素 fall，就要使用$arr[2]。PHP 数组除支持数字索引外，还支持字符串索引，即关联数组，这种数组通过字符串索引和元素关联。示例如代码 2-4 所示。

代码 2-4　关联数组 2-4.php

```
<?php
$sys = array(
"server"=>"Apache",
"os"=>"Windows",
"db"=>"MySQL",
);

echo $sys["server"];
echo "<br/>";
echo "<br/>";
echo $sys["db"];
?>
```

上述代码定义了一个关联数组$sys，通过$sys['server']可以取得数组元素“Apache”。程序执行结

果如图 2.4 所示。

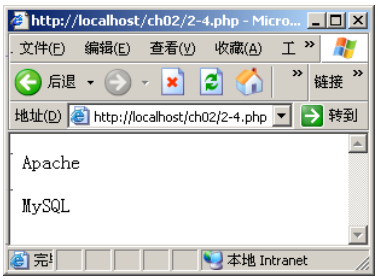

图 2.4　访问关联数组

2.2.3　变量类型的转换

和 C、C++等语言不同，PHP 在定义变量时，不需要明确指定变量的类型。也就是说，把一个整数指定给变量$v，那么$v 就是一个整型变量，如果把一个字符串指定给它，那么它就是一个字符串变量。若要转换类型，在 PHP 程序中也是很自由的，一般不必经过特殊的转换。

当然，PHP 中也可以对变量做强制转换，这点和 C 语言相似，在要转换的变量之前加上目标类型，目标类型用括号括起来。如代码 2-5 所示，将整型变量转换成布尔型。

代码 2-5　变量类型的强制转换　2-5.php

```
<?php
$foo = 10;

echo "转换前：\$foo=".$foo;
echo "<br/>";
echo "<br/>";

$foo = (boolean) $foo;
echo "转换后：\$foo=".$foo;
?>
```

上例程序执行结果如图 2.5 所示。

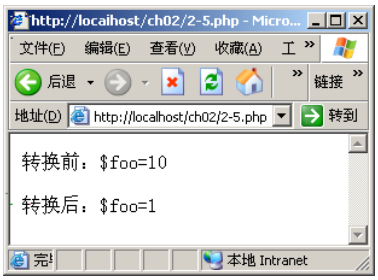

图 2.5　数据类型的强制转换

PHP 中允许的强制类型转换有：

- (int)，(integer)——转换成整型。
- (bool)，(boolean)——转换成 bool 型。
- (float)，(double)，(real)——转换成浮点型。

- (string)——转换成字符串。
- (array)——转换成数组。
- (object)——转换成对象。

2.2.4 可变变量

可变变量是 PHP 中比较特别的一个概念，可变变量是指这样一个变量，它将某个变量的值作为自己的变量名。下面通过一个例子说明，请看代码 2-6。

代码 2-6 可变变量 2-6.php

```
<?php
$i = "abc";          //定义变量$i，其值为 abc
$$i = "xyz";         //将变量$i 的值 abc 作为变量名，$i 被其值 abc 替换，因此，这句等价于$abc = “xyz”

echo "\$i=".$i;
echo "<br/>";
echo "<br/>";

echo "$\$i=".$abc;
?>
```

上述程序中，可变变量$$i 将变量$i 的值 abc 作为变量名，因此，$$i 就是变量$abc。程序执行结果如图 2.6 所示。

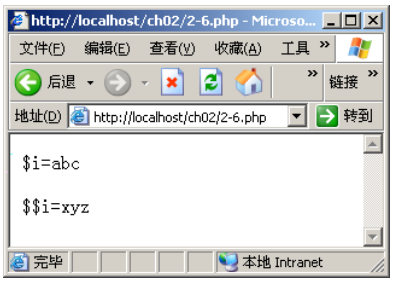

图 2.6 可变变量

2.2.5 PHP 的预定义变量

PHP 提供了大量的预定义变量，可以在程序或文件的任何地方使用它们。这些变量大多数依赖于服务器的版本及其配置。本书主要讲述 PHP5 的预定义变量，这些预定义变量和以前版本的有些不同，有兴趣的读者可自行了解。下面是 PHP 中一些常用的预定义变量。

- $GLOBALS：包含指向当前程序中全局范围内有效的变量，它是一个数组，该数组的索引（或键名）就是全局变量的名称。
- $SERVER：该全局变量是一个包含诸如头信息、路径和脚本位置的数组。常见的$_SERVER 的元素包括 PHP_SELF（当前正在执行的脚本的文件名）、SERVER_ADD（当前执行脚本所在服务器的 IP 地址）、SERVER_NAME（当前执行脚本所在服务器主机的名称）、DOCUMENT_ROOT（当前脚本所在文档的根目录）、SCRIPT_FILENAME（当前执行脚本的绝对路径）、SCRIPT_NAME（当前脚本的路径）、HTTP_REFERER（链接到当前页面的前一页面的 URL）、

REQUEST_URI（访问此页面所需的 URI）等。

- $_GET：通过 HTTP 的 GET 方法提交至脚本的表单变量。
- $_POST：通过 HTTP 的 POST 方法提交至脚本的表单变量。
- $_FILE：通过 HTTP 的 POST 文件上传提交至脚本的变量。
- $_COOKIE：通过 HTTP 的 Cookies 方法提交至脚本的变量。

下面通过代码 2-7，来了解这些变量的使用。其执行结果如图 2.7 所示。

代码 2-7　PHP 的预定义变量　2-7.php

```
<?php
$a = "test string";
echo "通过\$GLOBALS 来取变量值：".$GLOBALS['a'];
echo "<br/>";
echo "<br/>";

echo "当前执行脚本的文件名：".$_SERVER['PHP_SELF'];
echo "<br/>";
echo "<br/>";

echo "当前执行脚本所在的根目录：".$_SERVER['DOCUMENT_ROOT'];
echo "<br/>";
echo "<br/>";

echo "当前执行脚本的的绝对路径：".$_SERVER['SCRIPT_FILENAME'];
echo "<br/>";
echo "<br/>";
?>
```

图 2.7　PHP 的预定义变量

2.2.6　判断变量的类型

从本小节开始，将介绍一些处理变量的方法。因为在 PHP 中通常通过一些预定义函数来处理变量，所以，需要读者对函数的概念有个大概了解。

简单地说，函数是指完成某种特定功能的代码块，可以向函数传入参数，函数对参数进行处理，并且将处理结果返回给用户。本书将在后面详细介绍函数的概念。

在 PHP 中，可以通过以下函数对变量的类型做判断。

- 函数 is_integer 判断变量是否为整数。
- 函数 is_string 判断变量是否为字符串。
- 函数 is_double 判断变量是否为浮点数。
- 函数 is_array 判断一个变量是否为数组。

代码 2-8 演示了如何使用这些函数。

代码 2-8　判断变量类型的函数 2-8.php

```
<?php
$s = "this is a string";
$i = 9;
$arr = array(2,4,6);

is_string($s);      //返回 TRUE，表示$s 是一个字符串变量
is_string($i);      //返回 FALSE，表示$i 不是一个字符串变量
is_array($arr);   //返回 TRUE，表示$arr 是一个数组
is_array($s);     //返回 FALSE，表示$s 不是一个数组
?>
```

2.2.7　获取变量的类型

在 PHP 中，可以使用预定义函数 gettype 取得一个变量的类型，它接受一个变量作为参数，返回这个变量的类型。如代码 2-9.php 所示，程序执行结果如图 2.8 所示。

代码 2-9　获取变量的类型 2-9.php

```
<?php
$str = "this is a string";
$int = 9;
$bool = FALSE;

echo "\$str 的类型是：".gettype($str);
echo "<br/>";
echo "<br/>";

echo "\$int 的类型是：".gettype($int);
echo "<br/>";
echo "<br/>";

echo "\$bool 的类型是：".gettype($bool);
?>
```

图 2.8　获取变量的类型

2.2.8　设置变量的类型

使用预定义函数 settype 设置一个变量的类型，该函数接受两个参数，第一个参数是变量名，第二个参数是要设置的变量的数据类型。代码 2-10 演示了如何使用这个函数。

代码 2-10　设置变量的类型 2-10.php

```
<?php
$a = 100;

echo "设置前，变量\$a 的类型是：".gettype($a);
echo "<br/>";
echo "<br/>";

settype($a,"string");
echo "设置后，变量\$a 的类型是：".gettype($a);
?>
```

使用函数 settype 重新设置变量$a 之前，$a 是整型变量 100，使用 settype 设置$a 的类型之后，$a 成为字符串变量”100”。程序执行结果如图 2.9 所示。

图 2.9　设置变量的类型

2.2.9　判断一个变量是否已经定义

使用预定义函数 isset 判断一个变量是否已经定义，它接受一个变量作为参数值，返回值如果为 TRUE，说明该变量定义过，否则，说明该变量没有被定义。代码 2-11 演示了如何使用这个函数。

代码 2-11　判断变量是否被定义 2-11.php

```
<?php
$var = "test string";

echo isset($var);   //$var 定义过，返回 TRUE
echo isset($a);     //$a 未被定义，返回 FALSE
?>
```

2.2.10　删除一个变量

使用 unset 语句删除一个变量。从 PHP4 开始 unset 不再有返回值，因此，严格意义上讲，它并不是一个函数，而是一个 PHP 的语言结构。可以用 unset 一次删除多个 PHP 变量。下面的程序演示了如何使用 unset。

代码 2-12　删除一个变量 2-12.php

```
<?php
unset($var);                    //删除单个变量
unset($arr['elem']);            //删除单个数组元素
unset($var1, $var2, $var3);   //一次删除多个变量
?>
```

2.3　常量

和变量相对应的概念是常量。上节介绍了变量、变量的数据类型及对变量的一些操作、处理，这节将介绍常量的概念及使用。

2.3.1　什么是常量

在程序执行过程中，其值不能改变的量叫做常量。这就是说，常量不能再被定义成其它的值。常量也可以分为不同的类型，如 10、0、-12 是整型常量，1.23、-0.45 是浮点型常量，常量的类型从形式上就可以判别。

PHP 中有一些定义好的常量，在程序中可以直接使用。开发人员也可以根据程序的需要，自己定义新的常量。

2.3.2　定义常量

在 PHP 中通过 define()函数定义一个常量。合法的常量名只能以字母和下划线开始，后面可以跟着任意字母、数字或下划线。常量一旦定义就不能再修改或者取消定义。代码 2-13 定义了几个常量，并将它们输出。

代码 2-13　常量的定义 2-13.php

```
<?php
define(TESTSTRING,"Learning PHP");
define(SIZE,100);

echo "常量 TESTSTRING 的值为：".TESTSTRING;
echo "<br/>";
echo "<br/>";

echo "常量 SIZE 的值为：".SIZE;
?>
```

上述程序的执行结果如图 2.10 所示。

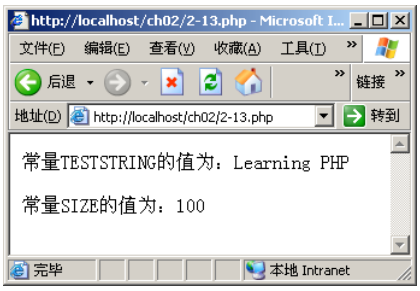

图 2.10　常量的定义和结果输出

注意：按照编码习惯，常量命名一般全部使用大写字母。常量名前面没有$符号，而且常量只能使用 define()函数定义，不能像变量那样使用赋值语句定义。

2.3.3　使用 PHP 预定义常量

PHP 提供了一些常量，可以直接在程序中使用。表 2-2 列举了 PHP 主要的预定义常量及其含义说明。

表 2-2　PHP的预定义常量及其说明

变量名	含义说明
__FILE__	正在执行的PHP程序的文件名。注意：FILE前后各两个下划线，不是各一个。
__LINE__	正在执行的PHP代码所在的行数。注意：LINE前后各两个下划线，不是各一个。
PHP_OS	PHP所运行的操作系统，如Window，UNIX等。
PHP_VERSION	当前PHP的版本。
TRUE	表示真值（1，或非0）的常量。
FALSE	表示伪值（0）的常量。

代码 2-14 演示了这些常用预定义常量的使用。

代码 2-14　使用 PHP 预定义常量 2-14.php

```
<?php
echo "===PHP 常见的预定义常量===";
echo "<br/>";
echo "<br/>";

echo "文件名：".__FILE__;
echo "<br/>";
```

```
echo "<br/>";
echo "当前代码行数：".__LINE__;
echo "<br/>";
echo "<br/>";
echo "PHP 的版本：".PHP_VERSION;
echo "<br/>";
echo "<br/>";
echo "PHP 所运行的操作系统：".PHP_OS;
?>
```

上述程序的执行结果如图 2.11 所示。

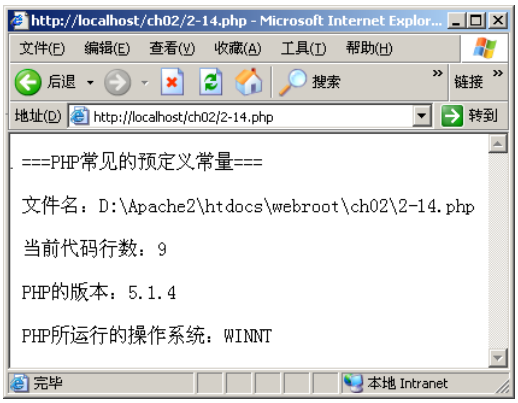

图 2.11　PHP 的预定义常量

2.4　表达式

表达式是指程序中任何有值的部分，PHP 中几乎所有内容都是表达式。如$a=9 就是一个表达式，这个表达式的含义是：将 9 指定给变量$a（即赋值操作，将在下一节介绍）。很明显，“9”的值就是 9，因此“9”本身就是一个表达式。也就是说，“9”是一个值为 9 的表达式，只不过在这里，9 是一个整型常量。同理，变量$a 也是一个值为 9 的表达式。从这个例子可以看出两个值：整数常量“9”和被指定值为 9 的变量$a，但事实上，还有一个值，就是这个$a=9 本身的值，表达式$a=9 的值就是被指定的值——9。

另外一类很常见的表达式就是比较表达式，如$a>$b 等。这些表达式的值要么是 0（表示 FALSE），要么是 1（表示 TRUE），如果表达式成立，则表达式的值为 1，否则，表达式的值为 0。例如代码 2-15 所示。

代码 2-15　比较表达式 2-15.php

```
<?php
$a = 3;
$b = 5;
$c = 5
$a>$b;      //$a>$b 不成立，所以表达式$a>$b 的值为 0
$a<=$b;     //$a<=$b 成立，所以表达式$a<=$b 的值为 1
$b==$c;     //变量$b 的值和变量$c 的值相等，所以表达式$b==$c 的值为 1
?>
```

PHP 支持的比较有：>（大于）、>=（大于等于）、==（等于）、<（小于）、<=（小于等于）。这些表达式通常用在程序执行的流程控制中，如将要在 2.6 节介绍的 if 语句。

还有一点要说明的是，“$a=5”和“$a=5;”是不同的，前者是一个表达式，后者是一条语句。PHP 程序中的每条语句都要以“;”结束。

2.5 运算符

运算符是指，通过一个或多个表达式，来产生另外一个值的某些符号，如“+”、“%”“.”等都是运算符。在表达式 2+1 中，运算符“+”有两个操作数 1 和 2。具有两个操作数的运算符叫做双目运算符。具有一个操作数的运算符叫做单目运算符，如表达式-6，运算符“-”只有一个操作数 6，因此，这里的“-”是单目运算符。

运算符有优先级，即表达式中多种运算符同时出现时，哪种运算符应该首先被应用。这节将介绍基本的运算符，最后介绍基本运算符的优先级。

2.5.1 赋值运算符

在 PHP 中，符号“=”不表示相等，而表示赋值。它的含义是将一个值指定给一个变量，更专业一点地说，它表示把“=”右边的表达式的值赋给左边的变量，如“$a=5”表示将 5 赋给$a。赋值表达式的值也就是所赋的值，这就是说，“$a=5”的值是 5。除了上述最基本的赋值方式外，还有一种所谓的“组合赋值”，通过例子来说明它的含义和用法。请看代码 2-16。

代码 2-16 组合赋值运算 2-16.php

```
<?php
$a = 12;
$a += 5;      //等价于$a = $a +5，即$a = 17
echo "\$a = ".$a;
?>
```

上面代码中，“$a+=5”等价于“$a=$a+5”。首先使用表达式“$a+5”的值，再将这个表达式的结果赋给$a，因为表达式$a 的值是 5，故表达式“$a+5”的值是 17，再将 17 赋给$a，所以最终$a 的值是 17。上面程序执行结果如图 2.12 所示

图 2.12 赋值运算符的使用

2.5.2 算术运算符

PHP 的算术运算符有加（+）、减（-）、乘（*）、除（/）和取模（%）、取反（-，即取负值）。

这些运算符的用法和学校里学到的数学知识一样，代码 2-17 演示了这些运算符的用法，图 2.13 显示了程序的执行结果。

代码 2-17　算术运算符 2-17.php

```
<?php
$a = 12;
$b = 5;

$add = $a+$b;
$sub = $a-$b;
$mult = $a*$b;
$div = $a/$b;
$mod = $a%$b;
$neg = -$a;

echo "\$a + \$b = ".$add;
echo "<br/>";
echo "<br/>";
echo "\$a - \$b = ".$sub;
echo "<br/>";
echo "<br/>";
echo "\$a * \$b = ".$mult;
echo "<br/>";
echo "<br/>";
echo "\$a / \$b = ".$div;
echo "<br/>";
echo "<br/>";
echo "\$a % \$b = ".$mod;
echo "<br/>";
echo "<br/>";
echo "-\$a = ".$neg;
?>
```

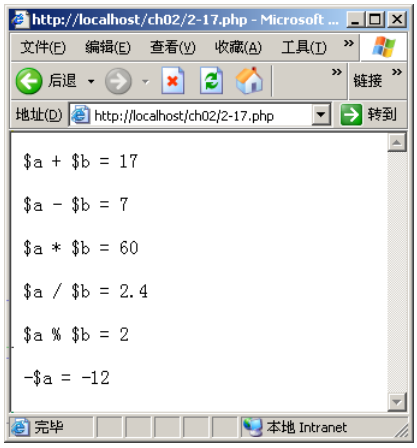

图 2.13　算术运算符的使用

提示：a%b（取模运算）表示取$a 除以$b 的余数。

2.5.3 递增/递减运算符

PHP 有和 C 语言风格相同的递增/递减运算符。递增是指对当前表达式的值增加 1，递减正相反，对表达式的值减 1。本书仅讲述整数表达式的递增/递减运算，下面分别介绍四种风格的递增/递减运算。

- $a++：先返回$a 的值，然后将$a 的值加 1。
- ++$a：先将$a 的值加 1，然后将$a 返回。
- $a--：先返回$a 的值，然后将$a 的值减 1。
- --$a：先将$a 的值减 1，然后返回$a 的值。

通过下面的示例代码 2-18，可以看出这些运算符及运算方式的不同。

代码 2-18　PHP 的递增/递减运算符 2-18.php

```
<?php
echo '<h3>后加递增 $a++</h3>';
$a = 5;
echo '$a = ' . $a++ . '<br />';      //$a++先返回$a 的值 5，所以这里输出 5，然后变量$a 自加 1 赋给$a
echo '$a = ' . $a . '<br />';        //上一行$a 输出之后，$a 已经加 1，所以这里$a 的值为 6

echo '<h3>前加递增 ++$b</h3>';
$b = 5;
echo '$b = ' . ++$b . '<br />';
echo '$b = ' . $b . '<br />';

echo '<h3>后减递减 $c--</h3>';
$c = 5;
echo '$c =' . $c-- . '<br />';
echo '$c = ' . $c . '<br />';

echo '<h3>前减递减 --$d</h3>';
$d = 5;
echo '$d = ' . --$d . '<br />';
echo '$d = ' . $d . '<br />';
?>
```

该程序的执行结果如图 2.14 所示。

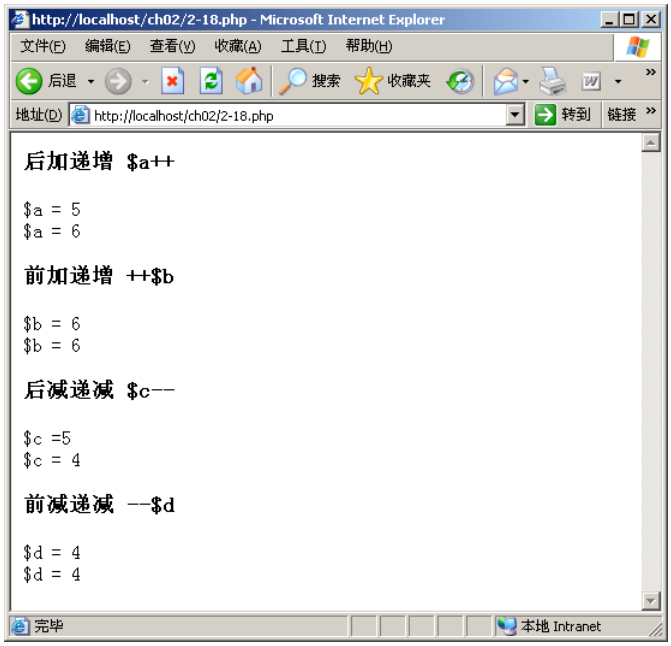

图 2.14　自增自减运算

2.5.4　字符串运算符

字符串运算符只有一个，即字符串的连接运算符“.”。这个运算符将两个字符串连接成一个新的字符串。在 2.2.2 小节介绍字符串变量时，简单提到过这个运算符。

其实在此之前，已经在很多示例程序中使用过这个运算符。比如程序要显示一个执行结果，会用“.”将一些内容连接起来，然后输出。如示例代码 2-19 所示。

代码 2-19　字符串运算符 2-19.php

```
<?php
$s1 = "Hello ";
$s2 = "everyone";

echo $s1.$s2;
echo "<br/>";
echo "<br/>";

$s1 .= "friend!";       //等价于$s1 = $1."friend"
echo $s1;
?>
```

程序执行结果如图 2.15 所示。

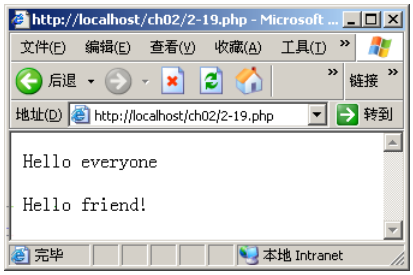

图 2.15　字符串连接

2.5.5　逻辑运算符

表 2-3 列举了逻辑运算符的运算结果。

表 2-3　逻辑运算符

运算符	名称	举例	结果
!	逻辑非	!$a	如果$a为FALSE，则!$a为TRUE。反之，!$a为FALSE。
&&（或and）	逻辑与	$a && $b	如果$a和$b都为TRUE，则$a && $b为TRUE。 如果$或$b任意一个为FALSE，则$a && $b为FALSE。
||（或or）	逻辑或	$a || $b	如果$a或$b任意一个为TRUE，则$a || $b为TRUE。 如果$a和$b都为FALSE，则$a || $b为FALSE
xor	逻辑异或	$a xor $b	如果$a或$b任意一个为TRUE，但不同时为TRUE，则$a xor $b为TRUE。

PHP 中存在两种不同形式的“与”和“或”运算符，是因为它们的优先级不一样。代码 2-20 演示了逻辑运算符的使用。

代码 2-20　PHP 的逻辑运算符 2-20.php

```
<?php
$b = FALSE;
echo !$b;      //$b 的值为 FALSE，所以!$b 的值为 TRUE，这里输出 1
//(14>=5)的值为 TRUE，('A'>'B')的值为 FALSE，TRUE||FALSE 的值为 TRUE，所以整个表达式的值为 TRUE
(14>=5)||('A'>'B');
//('B'>'A')的值为 TRUE，(8<7)的值为 FALSE，TRUE&&FALSE 的值为 FALSE，所以整个表达式的值为 FALSE
('B'>'A')&&(8<7);
?>
```

2.5.6　比较运算符

比较运算符用来对两个值进行比较。表 2-4 列举了主要的比较运算符及其可能的运算结果。

2-4　比较运算符

运算符及名称	举例	结果
==（等于）	$a == $b	如果 $a 等于 $b，则$a == $b的值为TRUE。
!=（不等于）	$a != $b	如果 $a 不等于 $b，则$a != $b的值为TRUE。
===（全等于）	$a === $b	如果 $a 等于 $b，并且它们的类型也相同，则$a === $b的值为TRUE。
>（大于）	$a > $b	如果 $a 大于 $b，则$a > $b的值为TRUE。
>=（大于等于）	$a >= $b	如果 $a 大于或者等于 $b，则$a >= $b的值为TRUE。
<（小于）	$a < $b	如果 $a 小于 $b，则$a < $b的值为TRUE。

<=（小于等于）	$a <= $b	如果 $a 小于或者等于 $b，则$a <= $b的值为TRUE。

关于比较运算的内容，在前面讲述表达式时已经有过提及，并且已经举过示例程序，这里不再重复举例。

2.5.7 运算符的优先级

事实上，在小学的数学知识中，就已经学习过运算符的优先级。比如 1+2×3 的结果是 7，不是 9。因为×号的优先级高于+号的优先级。只不过在 PHP 中，运算符不仅限于加减乘除。下面列举一些常见的 PHP 运算符的优先级，最上面的优先级最高。

- new（new 运算符，将在后面讲述）
- ++、--（递增、递减运算符）
- *、/、%
- +、-、.
- &&
- ||
- ? :（条件运算符，将在后面讲述）
- =（赋值运算符，包含+=、*=、.=等）
- and
- xor
- or

2.6 PHP 程序的流程控制

所有的 PHP 程序都由语句构成，程序就是一系列语句的序列。计算机通过执行这些语句可以完成特定的功能。一般情况下，程序都是从第一条语句开始执行，按顺序执行到最后一句。但有时因为某种情况，需要改变程序的执行顺序，这就需要对程序的流程进行控制。本节将讲述 PHP 程序的各种流程控制结构。

2.6.1 程序流程控制概述

计算机程序的执行方式有 3 种：顺序执行、选择执行、循环执行，如图 2.16 所示。通过使用这 3 种控制结构，可以改变程序的执行顺序，以满足开发人员解决问题的需求。

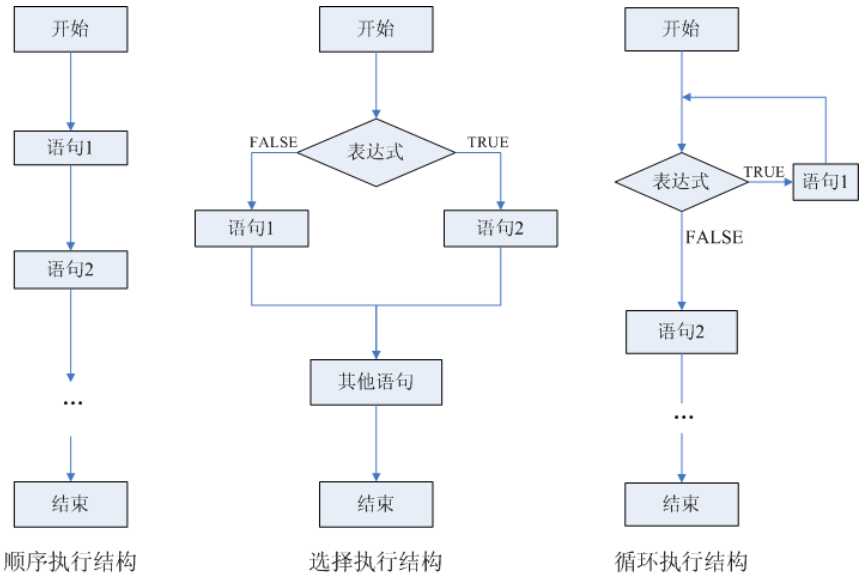

图 2.16　PHP 程序的 3 种执行结构

顺序结构使程序从第一条语句开始，按顺序执行到最后一句。在选择结构中，程序可以根据某个条件是否成立，选择执行不同的语句。在循环结构中，可以使程序根据某种条件和指定的次数，使某些语句执行多次。

PHP 程序都是由一系列语句组成，语句通常以分号结尾。此外，可以使用一对花括号“{”和“}”将一组语句组成一个语句组。例如：

```
{
$i = 123;
$s = “This is a string”;
}
```

2.6.2　条件控制语句：if 和 if…else 语句

PHP 通过一系列条件控制语句完成程序的选择执行流程。PHP 中使用 if、if…else（elseif）语句构建选择程序结构。

1．if 语句

if 条件语句的结构如下所示。

```
if(expr)
    statement
```

上述结构的含义是，如果表达式 expr 的值为真（TRUE），才会执行语句 statement。也就是说，当表达式 expr 成立，语句 statement 才会被执行，否则，表达式 expr 不成立（即 expr 的值为 FALSE），那么语句 statement 被忽略，不会执行。代码 2-21 是一个 if 条件语句的示例程序。

代码 2-21　if 语句 2-21.php

```
<?php
$a = 2;
$b = 3;

echo '$a = '.$a;
```

```
echo '<br/>';
echo '$b = '.$b;
echo '<br/>';
echo '<br/>';

if($a<$b)
    echo "$a 小于 $b";
?>
```

上述程序中，表达式$a<$b 的值为 1（TRUE），所以程序执行 echo 语句输出“$a 小于 $b”。if 后面可以跟一个空语句，即只加一个分号“;”的语句，表示当条件成立时，什么都不做。如下面的代码。此时，if 后跟了一个使用分号结束的空语句，表示当表达式$a<$b 的值为 1 时，程序什么都不做。

```
if($a<$b) ;
```

2．if…else 语句

if…else 语句的结构如下所示。

```
if(expr)
    statement1
else
    statement2
```

if…else 结构的含义是：如果表达式 expr 的值为真，程序执行语句 statement1，否则程序执行 statement2。两个语句只能由一个被执行，另外一个将会被忽略。下面的代码 2-22 演示了 if…else 语句的使用方法。

代码 2-22　if…else 语句 2-22.php

```
<?php
$a = 2;
$b = 3;

if($a>$b)
    echo "$a 大于 $b";
else
    echo "$a 小于或等于 $b";
?>
```

上述程序中，表达式$a>$b 的值为 0（FALSE），所以 if 后的语句 echo ‘$a 大于 $b’被忽略，不会执行，转去执行 else 后面的 echo 语句。在 PHP 中，else 并不是单独的语句，它和 if 语句必须成对使用，也就是说不能将其从 if 语句中分离出来单独使用。

如果判断条件成立时要执行的语句只有一条，那么 if 语句后可以不加花括号“{”和“}”，就像上面几个示例程序写的那样。如果判断条件成立时有多条语句要执行，那么这些语句应该组成语句组，放在一对花括号里。如下面的示例代码 2-23，当条件成立时，需要执行 3 条语句，因此它们都在花括号里。

代码 2-23　if…else 语句 2-23.php

```
<?php
$a = 2;
$b = 3;

if($a<$b)
{
```

```
    echo '$a = '.$a;
    echo "<br/>";
    echo "<br/>";

    echo '$b = '.$b;
    echo "<br/>";
    echo "<br/>";

    echo '$a 小于 $b';
}
else
{
    echo "$a 和 $b 比较";
    echo '$a 大于等于 $b';
}
```

该程序的执行结果如图 2.17 所示。

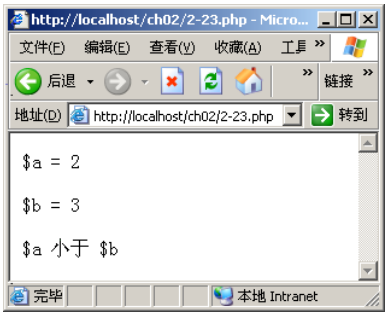

图 2.17　if…else 语句的使用

3．if…elseif 语句

if…elseif 语句的结构如下所示。

```
if(expr1)
    statement1
elseif(expr2)
    statement2
…
else
     statement
```

if…elseif 结构的含义是：如果表达式 expr1 的值为 1，程序执行语句 statement1；否则，判断表达式 expr2，如果 expr2 的值为 1，程序执行语句 statement2；否则，如果有其它表达式需要判断，则依次判断下去，如果所有表达式的值都不为 1，则执行 else 后的 statemnet 语句。如果其中有一个表达式的值为 1，那么它的语句将被执行，因此，剩下的表达式将不会做判断，程序直接从控制结构中跳出，接着执行后续代码。代码 2-24 演示了 if…elseif 语句的使用。

代码 2-24　if…elseif 语句　2-24.php

```
<?php
$a = 2;
$b = 2;
```

```
echo '$a = '.$a;
echo '<br/>';
echo '$b = '.$b;
echo '<br/>';
echo '<br/>';

if($a<$b)
    echo '$a 小于$b';
elseif($a>$b)
    echo '$a 大于$b';
elseif($a==$b)
    echo '$a 等于$b';
else
    'error!';
?>
```

上述程序中，因为只有表达式$a= =$b 成立，所以只有语句“echo '$a 等于$b'”被执行。当这条语句被执行后，程序将从 if…elseif 控制结构中跳出，程序将不会执行到代码 elseif($a>$b)，即程序不会再去做$a>$b 是否成立的判断。程序执行结果如图 2.18 所示。

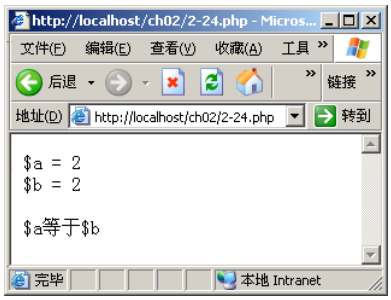

图 2.19　if…elseif 语句的使用

2.6.3　switch 结构

上一小节讲述了使用 if 或者 if…else 语句的选择控制结构，if 语句通常需要计算逻辑表达式的值。这小节将介绍另一种选择控制结构——switch 结构，它不需要计算逻辑表达式的值。switch 的语法结构如下所示。

```
switch(expr)
{
case value1:
    statement1
    break;
case value2:
    statement2
    break;
…
case valuen:
    statementn
    break;
default:
    statement
```

```
}
```

注意：在 PHP 中，switch、case、break 和 default 都是保留关键字。

switch 结构首先计算表达式 expr 的值，如果 expr 的值与某个 case 的值匹配，则从 case 后面的语句开始执行，直到遇到 break 语句（该语句将在后面详细介绍）或整个 switch 结构结束。比如，如果 expr 的值是 value2，那么语句 statement2 将会被执行。

如果 expr 的值不与任何 case 值匹配，则执行 default 后面的语句。如果没有 default 语句，而且表达式 expr 的值不与任何 case 值匹配，那么程序从 switch 结构中跳出。代码 2-25 是一个 switch 结构的示例程序。

代码 2-25　switch 结构 2-25.php

```
<?php
$a = 3;
switch($a)
{
    case 1:
        echo "It's January";
        break;
    case 2:
        echo "It's February";
        break;
    case 3:
        echo "It's March";
        break;
    case 4:
        echo "It's April";
        break;
    default:
        echo "Other months";
}
?>
```

程序执行结果如图 2.20 所示。

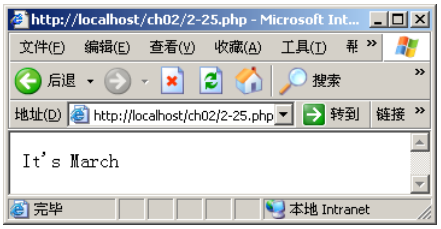

图 2.20　switch 结构

2.6.4　循环控制语句：for 循环语句

前两小节讲述的是程序的选择控制流程，这节开始讲述程序的循环控制流程。循环控制流程可以控制程序，在满足某些条件的时候，某些语句被循环执行多次。

PHP 的循环控制主要有：for 语句、while 语句和 do…while 语句。本节先介绍 for 循环语句。

for 循环语句的结构如下：

```
for(初始化语句; 循环条件表达式; 更新语句)
    statement
```

该语句括号中的初始化语句、循环条件和更新语句，叫做 for 循环控制语句，它们控制着语句的循环体，即语句 statement，该语句即可以是单条语句，也可以是由花括号“{”和“}”括起来的语句组。

for 语句的执行过程如下：

（1）执行初始化语句。

（2）判断循环条件表达式，如果其值为 TRUE（非 0），则执行 for 语句的循环体 statement 语句。然后执行更新语句。

（3）接着执行第 2 步，直到循环条件的值为 FALSE 为止。

图 2.21 更清楚地说明了 for 语句的执行过程。代码 2-26 是一个使用 for 循环语句的示例，它通过 for 语句，循环输出 10 以内的偶数。

代码 2-26　for 循环语句的使用 2-26.php

```
<?php
echo "输出 10 以内的偶数：";
echo "<br/>";
echo "<br/>";

for($i=0;$i<=10;$i++)
{
    if($i%2==0)
    {
        echo $i;
        echo "<br/>";
    }
}
?>
```

上述程序中的 for 循环中，首先执行初始化语句$=1;，将变量$i 的值指定为 1。接着，计算循环条件$<=10。因为此时$i<=10 的值为 TRUE，所以执行 if 语句块。然后执行更新语句，$i++将变量$i 的值更新为 2 后，再次计算循环条件，以此类推。当$i 的值变成 11 时，循环条件的值为 FALSE，此时 for 循环终止，程序跳出 for 循环结构，接着执行后续代码。

程序执行结果如图 2.22 所示。

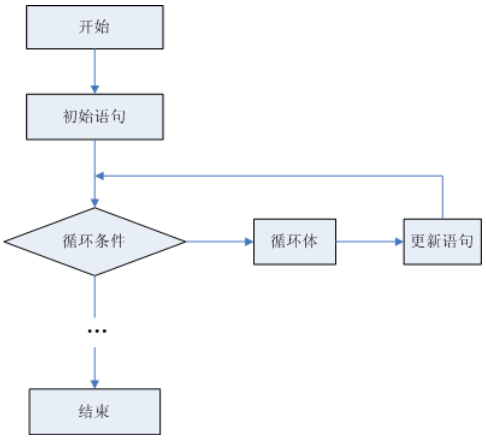

图 2.21　for 循环执行示意图

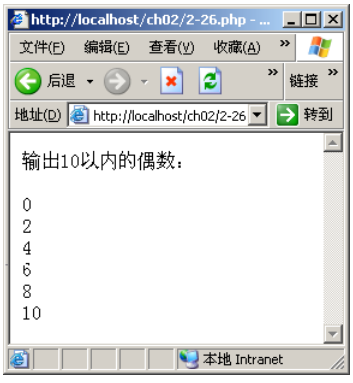

图 2.22　for 循环语句的使用

2.6.5 while 循环语句

除了 for 循环语句之外，还可以使用 while 语句控制程序循环执行。while 循环语句的结构如下：

```
while(expr)
    statement
```

这里的语句 statement 可以是单条语句，也可以是语句组。该结构的执行流程是：当表达式 expr 的值为真时，就执行循环体——语句 statement，然后再次计算表达式 expr 的值，直到 expr 的值为假，程序中断循环，跳出 while 循环结构。代码 2-27 演示了如何使用 while 语句，它通过 while 循环，输出 0 和 20 以内的 5 的倍数。

代码 2-27　while 语句的使用　2-27.php

```
<?php
$a = 0;

while($a<=20)
{
    echo "$a";
    echo "<br/>";
    $a = $a+5;
}
?>
```

上述程序变量$a 的值初始化为 0。whlie 语句首先计算表达式$a<=20 的值，该表达式的值为 TRUE，所以执行 while 语句中的循环体：echo “$a
”;和$a=$a+5;，该循环体第一条语句将输出变量$i 的值——0，第二条语句将变量的值更改为 5。接着，while 语句中的条件表达式被重新计算，因为此时$a 的值为 5，表达式$a<=20 的值仍为 TRUE，所以继续执行循环体中的语句。这个过程反复进行，直到条件表达式$a<=20 的值为 FALSE 为止。程序执行结果如图 2.23 所示。

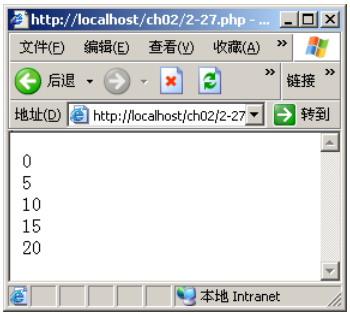

图 2.23　while 循环语句的使用

2.6.6 do…while 循环语句

这小节讲述第三种循环控制结构：do…while 语句，它也可以控制程序循环执行。do…while 循环语句的结构如下：

```
do
    statement
while(expr);
```

这里的语句 statement 可以是单条语句，也可以是语句组。该结构的执行流程是：程序首先执行语

句 statement，然后再计算表达式 expr 的值。如果表达式 expr 的值为 TRUE，就再次执行语句 statement。

注意：do…while 循环控制流程与 for 和 while 不同的是，该语句的循环体至少会执行一次，因为 do…while 语句是在循环体执行后，才做条件表达式的判断。

代码 2-28 通过 do…while 循环计算 1 到 50 的数字的和，如下所示：

代码 2-28　do…while 循环语句的使用　2-28.php

```
<?php
$i = 1;
$s = 0;

do
{
    $s = $s + $i;
    $i++;
}
while($i<=50);

echo "1 + 2 + 3 +...+ 49 + 50 = ".$s;
?>
```

上述程序执行结果如图 2.24 所示。

图 2.24　do…while 语句的使用

2.6.7　break 和 continue 语句

在 2.6.3 小节中介绍 switch 语句时，提及过 break 语句，它可以使程序流程跳出 switch 结构。除此之外，break 语句还可以在 for、while 和 do…while 语句中使用，这样可以使程序立即跳出该循环结构。请看示例代码 2-29：

代码 2-29　break 语句的使用　2-29.php

```
<?php
define(PI,3.14);

for($r=1;$r<=10;$r++)
{
    $area = PI * $r * $r;
    if($area>100)
        break;

    echo "r=$r, area=$area";
```

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

上述程序计算半径 1 到 10 的圆的面积，直到面积大于 100 时为止。当面积 area>100 时，执行 break 语句，中断循环，不再执行剩余的几次循环。程序的执行结果如图 2.25 所示。从程序的执行结果可以看出，for 循环只执行了 5 次就因 break 语句而退出，剩下的 5 次循环没有执行。continue 语句的作用是结束当前的循环，即跳过该循环体中剩余的语句，转而执行下次循环，如果循环条件满足的话。

continue 语句和 break 语句的区别是，continue 语句只是结束本次循环，而 break 语句是终止整个循环的执行，不再做条件的判断。代码 2-30 演示了 continue 语句的使用。

代码 2-30　continue 语句的使用　2-30.php

```
<?php
for($a=100;$a<=200;$a++)
{
    if($a%3==0)
        continue;
    echo $a;
    echo "<br/>";
}
?>
```

上述程序将 100 到 2000 之间不能被 3 整除对数输出。当$a 能被 3 整除时，执行 continue 语句，结束本次循环，继续执行 for 循环语句。只有$a 不能被 3 整除时，才使用 echo 语句将该数字输出。该程序执行结果如图 2.26 所示。

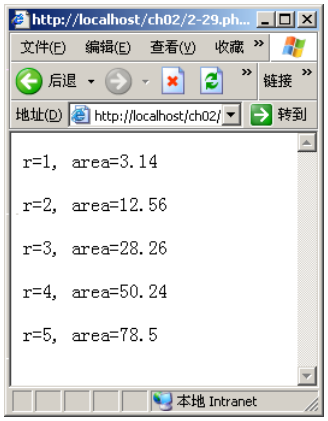

图 2.25　break 语句的使用

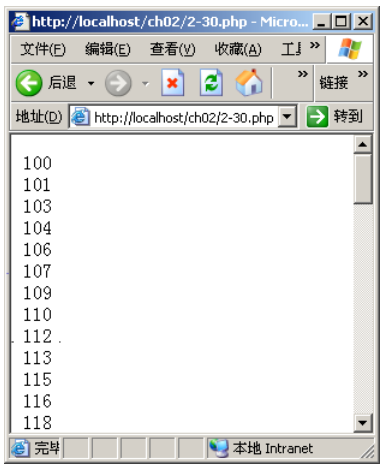

图 2.26　continue 语句的使用

2.6.8　条件运算符：？：

条件运算符（?:）的用法是：

```
expr1 ? expr2 : expr2
```

可以看出，条件运算符有 3 个操作数，所以它是三目运算符。它的计算规则是：如果表达式 expr1 的值为 TRUE，那么整个表达式的值就取 expr2 的值，否则，就取 expr3 的值。下面的代码使用条件运

算符来获取两个数中最大的那个数。

```
$max = ($a>=$b) ? $a : $b
```

当表达式$a>=$b 为 TRUE，那么整个表达式的值就是变量$a 的值，这意味着，会将$a 的值赋给变量$max，从而取得$a 和$b 之间较大的那个数。

当然，上诉代码也可以使用 if…else 语句实现，但使用条件运算符可以使程序更精炼，而且条件运算符在判断条件较复杂的情况下，比 if…else 语句执行更快速。

2.7 函数

在很多编程语言中都有函数这个概念。函数将为解决某一问题而编写的代码组织在一起，使得在解决同一个问题时，可以重复这些代码。本节将介绍 PHP 中函数的概念、构建和调用函数等内容。

2.7.1 PHP 中函数的概念

在数学知识里，函数是由参数的定义域和在这个参数定义域上的某种规则组成的。当选定某一参数时，函数的值也是惟一确定的。例如，有这样一个数学函数：f(x)=2x+3，那么就有 f(1)=5，f(3)=9。这里的 1、3 都是函数 f 的参数，而 5、9 都是这些参数对应的函数 f 的值。

PHP 语言中的函数和数学中函数的概念很相似，只不过 PHP 中的函数不仅仅是做一些数学运算，而是要完成更多、更复杂的功能。

在程序设计中，经常将一些常用的功能模块编写成函数，放在公用函数库中，供程序或其它文件使用。函数就像一些小程序，用它们可以组成更大的程序。函数之间也可以相互调用，完成更复杂的功能，但它们之间是相互独立的，互不隶属。

从使用角度来看，PHP 的函数可以分为两种：PHP 的预定义函数和用户自定义的函数。用户可以在自己的程序或 PHP 文件中直接使用预定义函数，PHP 提供过了大量的、功能丰富的预定义函数，供 PHP 开发人员使用，极大地提高了开发效率。用户自定义的函数，是开发人员专门用来解决特定需求的功能模块。

2.7.2 定义函数和调用函数

PHP 使用下面的语法定义一个函数：

```
function func_name(param_list)
{statement}
```

其中 function 是 PHP 的保留关键字，表示开始定义一个函数。func_name 是函数名，由开发人员自行指定，函数名以字母或下划线开始，后跟任意字母、数字或下划线。函数名后的一对括号，用来存放函数的参数 param_list，如果所定义的函数不需要传入参数，括号内留空，但不能没有括号。最后花括号括住的语句 statement 叫做函数体，它可以是单条语句，也可以是多条语句，这些语句完成函数所要实现的功能。下面的代码演示了如何定义了一个函数。

提示：函数的参数根据需要，可以有多个。

```
function say_hello()
{
    $name = “Jack”;
```

```
    echo “Hello,”.$name;
}
```

这段代码很简单，它定义了一个名叫 say_hello 的函数，该函数没有参数，所以函数名后面的括号内留空，函数体有两条语句，第一句是赋值语句，将某个人名赋给变量$name，第二句输出一个含这个人名的字符串。实际上，函数的功能就是用 echo 语句输出一条问候语。

一个定义好的函数，需要在程序其它地方使用，才能发挥它的功能。调用一个函数的最简单的用法就像下面代码这样：

```
function_name();
```

其中 function_name 是所要调用的函数的名称。代码 2-31 演示了如何在程序中调用上面定义的函数 say_hello()。

代码 2-31　函数的调用 2-31.php

```
<?php
function say_hello()
{
    $name = "Jack";
    echo "Hello,".$name;
}

say_hello();      //在这里调用上面定义的函数 say_hello
?>
```

该程序首先定义了一个函数 say_hello()，但此时不会执行函数中的 echo 语句，因为此时只是函数的定义，计算机不需要执行任何实际代码。在这段代码最后，使用 say_hello()来调用函数，此时，程序转入函数内部执行，这时执行 echo 语句，输出一条问候语。该程序的执行结果如图 2.27 所示。

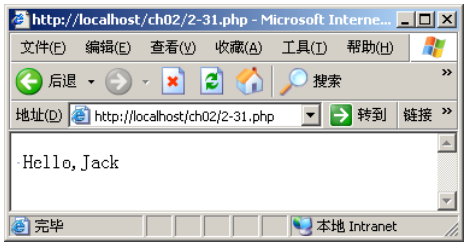

图 2.27　函数的定义和调用

另外，一个 PHP 文件中，函数定义的位置和调用函数的位置，一般不需要确定的前后书写关系。也就是说，函数的定义可以在 PHP 文件的任何位置，调用也可以在 PHP 文件的任何位置，函数的定义不一定必须在函数调用的前面书写。例如代码 2-31 中函数的调用，也可以写在最前面，而在最后定义函数 say_hello。

2.7.3　函数的参数和函数的返回值

上小节的函数 say_hello()只能向“Jack”显示问候语，如果想向更多的人显示问候语，该怎么做呢？这就是函数参数的问题。在大多数情况下，调用函数都会有数据的传递，这就是前面讲到的函数参数。将参数传递个函数，函数根据不同的参数会完成不同的功能，或有不同的输出。

在函数 say_hello()的例子中，如果想要向更多的人显示问候语，可以传递一个参数给函数 say_hello()，这个参数就是不同的人名，在函数体内输出这个变量即可。代码 2-32 演示了带有参数的函数的用法。

代码 2-32　调用带有参数的函数 2-32.php

```
<?php
function say_hello($some_name)
{
    echo "Hello,".$some_name;
    echo "<br/>";
    echo "<br/>";
}

say_hello("Jenny");      //这里使用参数“Jenny”调用函数 say_hello
say_hello("Harry");
say_hello("Ema");
?>
```

上面的程序，定义了带有一个参数的函数 say_hello($some_name)。接着三次调用该函数，但每次传给函数参数的值不一样。第一次调用时，传给函数的值是“Jenny”，因此函数参数$some_name 的值即为“Jenny”，这时就会输出“Hello,Jenny”。同样的道理，其它的调用就会显示对其它人名的问候语。程序执行结果如图 2.28 所示。

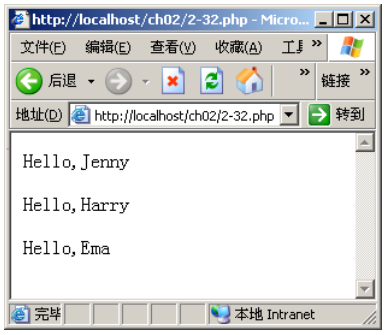

图 2.28　调用带有参数的函数

PHP 的函数的参数可以使用默认值。通过一个实例程序来了解函数默认参数的使用。请看代码 2-33。

代码 2-33　函数的默认参数 2-33.php

```
<?php
function say_hello($some_name = "Jack")
{
    echo "Hello,".$some_name;
    echo "<br/>";
    echo "<br/>";
}

say_hello();             //不使用任何参数调用函数 say_hello 时，函数将使用函数定义的默认参数“Jack”
say_hello("Jenny");      //使用参数“Jenny”调用函数 say_hello
say_hello("Harry");
say_hello("Ema");
?>
```

该程序对函数 say_hello 做了一点小的改动，在函数定义时，为函数的参数$some_name 赋了默认值“Jack”，这样当程序中调用该函数，但又没有传值给函数参数时，函数就会在函数体内使用这个默认

值“Jack”，如果调用时传值给函数参数，那么个这个默认值就不再起作用，而是在函数体内使用用户传入的值。程序首先使用 say_hello()调用该函数，没有传入任何值给函数，那么这时函数体执行后，就会输出“Hello,Jack”。程序的整个执行结果如图 2.29 所示。

图 2-29　使用函数的默认参数

有时，希望在调用一个函数之后，能得到一个确定的值，这就是函数的返回值。PHP 函数中使用 return 语句取得函数的返回值，return 语句会将函数中一个确定的值带回到调用函数的地方。如下面的示例代码 2-34 演示了如何获得函数的返回值。

代码 2-34　函数的返回值 2-34.php

```
<?php
define(PI,3.14);

for($r = 3; $r<=8; $r++)
{
      $s = get_circle_area($r);
      echo "r=$r, area=$s";
      echo "<br/>";
      echo "<br/>";
}

function get_circle_area($radius)
{
      $area = PI * $radius * $radius;
      return $area;
}
?>
```

上述程序定义了一个计算圆面积的函数 get_circle_area，并在函数体最后，将计算结果用 return 语句返回。在 for 循环里多次调用函数 get_circle_area，每次调用，函数 get_circle_area 都会将计算结果返回赋给变量$s，接着输出圆的半径及其对应的圆的面积。程序的执行结果如图 2-30 所示。

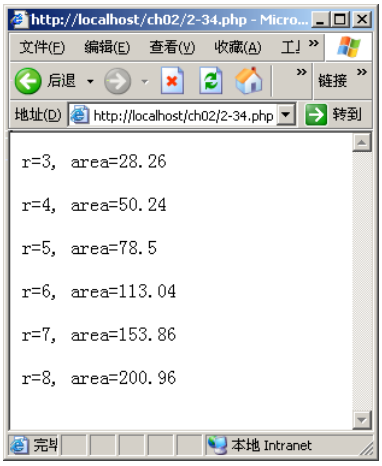

图 2.30　函数的返回值

2.7.4　PHP 函数的传值与传址

上小节讲述的向函数传入参数，是按传值方式传入的。传值的含义是指，在函数体内，会生成一个传入值的拷贝，在函数内部对参数的修改，不会影响到传入的值。有时，因为开发的需要，希望在函数内部能够修改传入的值，这是就需要对函数参数使用传址方式。传址的含义是指，在函数体内，真实引用传入的值，这意味着，在函数体内使用的函数参数，和传入的值完全是同一个，而不单单是传入值的一个拷贝。这时，在函数内部修改了参数的值，同时也就修改了传入的值。

在 PHP 中，要想在函数参数中传址，需要在定义函数时，在参数前加上符号：&。代码 2-35 演示了 PHP 函数如何使用传址方式传递参数。

代码 2-35　传址方式传递函数参数 2-35.php

```
<?php
$i = 100;

function func(&$n)
{
    $n = $n+100;        //因为传址传入变量$i，所以这里的变量$n 引用的就是$i 本身，此句等价于$i = $i+100
}

echo "调用函数 func 前：\$i=$i";
echo "<br/>";
echo "<br/>";

func($i);                    //将$i 传入函数 func，因为是传址方式，所以此时函数内的变量$n 就是变量$i
echo "调用函数 func 后：\$i=$i";
?>
```

上述程序定义函数 func，该函数接受一个传址方式的参数。执行函数体内的语句“$n = $n +100;”同时会使变量$i 的值发生改变。程序的执行结果如图 2.31 所示，从中可以看出，变量$i 的值原来为 100，因为经过参数传址方式的调用，在函数 myfunc 中使用的就是变量$i 本身，所以变量$i 的值被修改为 200。

为了和传址方式有个对比，将上述函数定义改为“function func($n)”，即按传值方式传递参数给函数 func，看看结果有什么不同。请看代码 2-36。

代码 2-36　传值方式传递函数参数 2-36.php

```
<?php
$i = 100;

function func($n)
{
    $n = $n+100;
}

echo "调用函数 func 前：\$i=$i";
echo "<br/>";
echo "<br/>";

func($i);
echo "调用函数 func 后：\$i=$i.<br/>";
?>
```

程序的执行结果如图 2.32 所示。

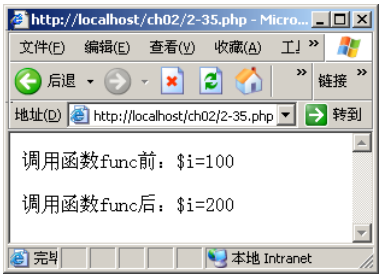

图 2.31　传址方式传递函数参数

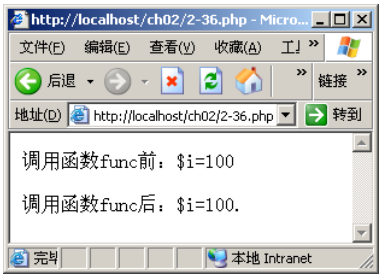

图 2.32　传值方式传递函数参数

从这个执行结果中可以看出，因为是传值方式传递给函数参数，在函数体内使用的是变量$i 的一个拷贝，并不是$i 本身，所以尽管在函数体内执行了加法运算，但并没有影响到变量$i 的原始值。

2.7.5　函数和变量作用域

变量的作用域就是变量的有效范围。对于大多数 PHP 变量，作用域只能有一个。但是，在用户自定义函数中，存在一个单独的局部函数范围。在一个函数内部定义的变量是局部变量，它只在本函数内有效，它的作用域就是当前的函数之内。这就是说，一个在函数外部定义的变量，不会在函数内部起作用，反之亦然。请看如下所示的代码 2-37。

代码 2-37　变量作用域演示程序（1） 2-37.php

```
<?php
$var = "some text";

function test()
{
    echo $var;
}
```

```
test();
?>
```

这段代码愿意想通过调用函数 test()，来输出变量$var 的值，但通过浏览器没有看到任何输出内容。这是因为，两个$var 变量的作用域是不同的，第 1 个$var 是一个全局变量，虽然它被赋值，但它的作用域，或者说它的有效范围并不在函数 test()内，这一点和 C、Java 等语言完全不同，在 C 等语言中，全局变量在函数中自动生效，除非被局部变量覆盖。第 2 个$var 是局部变量，echo 语句引用的是第 2 个$var（其作用域是局部的），它没有被定义（或者说没有被赋值），因此该代码 2-37 不会在浏览器中显示出任何字符串内容。对上述代码稍作改动，读者可以更清楚地看到两个变量的有效范围。请看如下所示的代码 2-38。

代码 2-38　变量作用域演示程序（2） 2-38.php

```
<?php
$var = "some text";

function test()
{
    $var = "some text in function";
    echo '这是局部变量$var：'.$var;
}

echo '这是全局变量$var：'.$var;
echo '<br/>';
echo '<br/>';
test();
?>
```

这段代码的执行结果如图 2.33 所示。

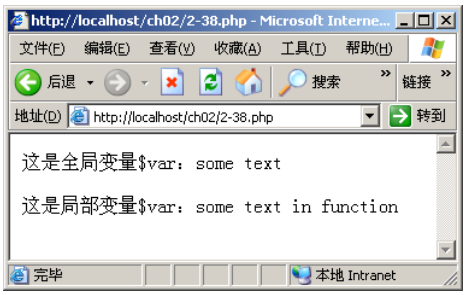

图 2.33　变量的作用域

从这个执行结果可以看出，在函数 test()外部定义的变量$var，不会在函数内部生效。在函数 test()中输出的变量$var 是在其内部定义的，而在函数 test()外部定义的全局变量$var 并不会在函数内部生效。如果希望在函数内部引用全局变量，需要在函数内部使用 global 关键字。PHP 中 global 关键字的作用是，在函数内部声明某个变量为全局变量，从而在函数内部使用该变量。代码 2-39 演示了 global 关键字的作用，如下所示。

代码 2-39　在函数内部使用 global 关键字 2-39.php

```
<?php
$a = 1997;
$b = 1998;
```

```
function sum()
{
      global $a,$b;
      $b = $a + $b;
}

echo '$a='.$a;
echo '<br/>';
echo '$b='.$b;
echo '<br/>';
echo '<br/>';

sum();
echo '$a + $b = '.$b;
?>
```

这段代码首先定义了两个全局变量$a 和$b，然后定义求和函数 sum()，在该函数内，使用 global 关键字引用全局变量求和，并将求和结果赋值给变量$b，如代码第 7 行和第 8 行所示。这段代码的执行结果如图 2.34 所示。如果将代码 2-39 中，第 7 行注释掉，再次执行该程序，可以看到全局变量$a 和$b 没有在函数 sum()内生效，其执行结果如图 2.35 所示。

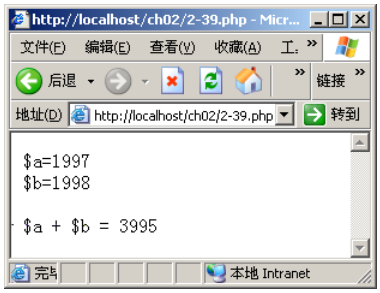

图 2.34　在程序中使用 global 关键字

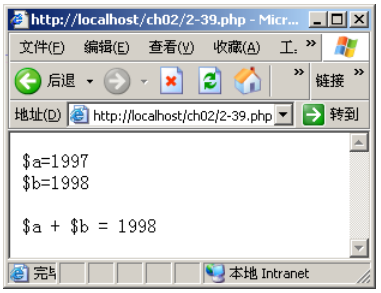

图 2.35　在程序中未使用 global 关键字

2.8　小结

本章主要介绍了 PHP 的基本语法，包括变量类型、常量、表达式、运算符、程序控制流程和函数的基本概念。其中变量、表达式、程序控制流程和函数是本章的重点知识，读者应该熟练掌握这些内容。

流程控制是程序设计中非常关键的地方，虽然这些控制语句看起来很简单，但使用过程中，如果因为逻辑错误，将可能导致死循环发生。