第 3 章 PHP 流程控制语句

程序由一条条语句组成，每条语句都被用来实现一个具体的任务。一般情况下，一段程序代码是顺序执行的，即从头到尾按顺序逐行执行。顺序执行是程序最为基本、最为简单的结构。但有时也需要在某种条件下有选择地执行指定的操作，或者重复地执行某一类程序，这就是所谓的程序流程的控制问题。流程控制语句包括条件控制语句、循环控制语句和跳转语句。合理地使用这些控制结构可以使程序流程清晰、可读性强，从而提高工作效率。

3.1 PHP 的 3 种控制结构

在编程的过程中，所有的操作都是在按照某种结构有条不紊地进行，学习 PHP 语言，不仅要掌握其中的函数、数组、字符串等知识，更重要的是，通过这些知识形成一种属于自己的编程思想和编程方法。要想形成属于自己的编程思想和方法，那么首先就要掌握程序设计的结构，再配合以函数、数组、字符串等知识，逐步形成自己的编程方法。

程序设计的结构大致可以分为顺序结构、选择结构和循环结构等 3 种。在对这 3 种结构的使用中，几乎很少有哪个程序是单独地使用某一种结构来完成某个操作，大多是其中的 2 种或者 3 种结构结合使用。

3.1.1 顺序结构

顺序结构是最基本的结构方式，各流程依次按顺序执行。传统流程图的表示方式与 N-S 结构化流程图的表示方式分别如图 3-1 和图 3-2 所示。执行顺序为：开始→语句 1→语句 2→…→结束。

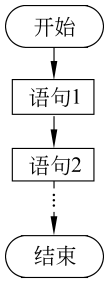
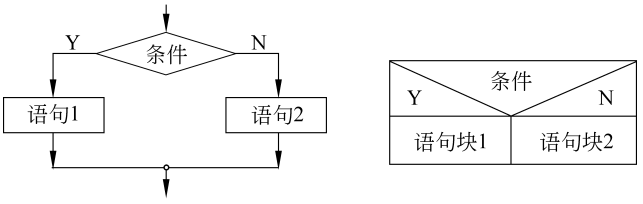


图 3-1 顺序结构传统流程图

图 3-2 N-S 结构化流程图

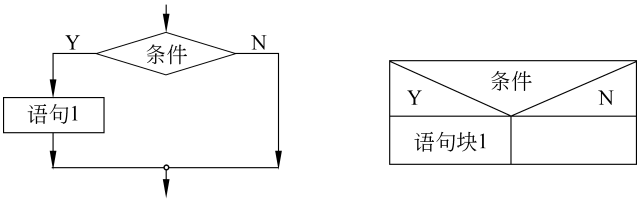
3.1.2 选择结构

选择结构就是对给定条件进行判断，条件为真时执行一个分支，条件为假时执行另一个分支。其传统流程图表示方式与 N-S 结构化流程图表示方式分别如图 3-3 和图 3-4 所示。



(a) 传统流程图　　(b) N-S结构化流程图

图 3-3　条件成立与否都执行语句或语句块



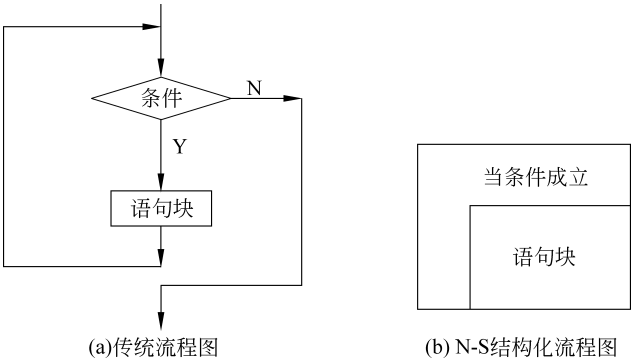
(a) 传统流程图　　(b) N-S结构化流程图

图 3-4　条件为否不执行语句或语句块

3.1.3 循环结构

循环结构可以按照需要多次重复执行一行或者多行代码。循环结构分为两种：前测试型循环和后测试型循环。

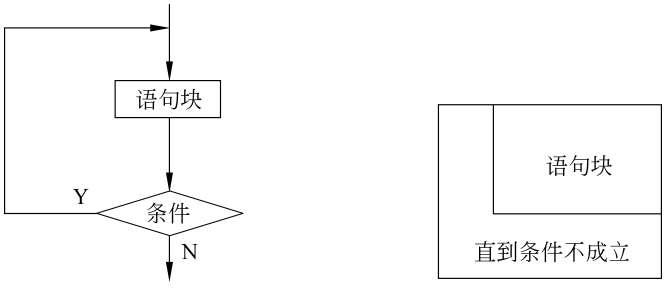
前测试型循环是先判断后执行，是当条件为真时反复执行语句或语句块，条件为假时，跳出循环，继续执行循环后面的语句，流程图如图 3-5 所示。



(a)传统流程图　　(b) N-S结构化流程图

图 3-5　前测试型循环流程图

后测试型循环是先执行后判断，即先执行语句或语句块，再进行条件判断，直到条件为假时，跳出循环，继续执行循环后面的语句，否则一直执行语句或语句块，流程图如图 3-6 所示。



(a) 传统流程图　　(b) N-S结构化流程图

图 3-6　后测试型循环流程图

在 PHP 中，大多数情况下程序都是以这 3 种结构的组合形式出现。其中，顺序结构很容易理解，就是直接输出程序运行结果，而选择和循环结构则需要一些特殊的控制语句来实现，包括以下 3 种控制语句。

（1）条件控制语句：if、else、elseif 和 switch。

（2）循环控制语句：while、do…while、for 和 foreach。

（3）跳转控制语句：break、continue。

3.2　条件控制语句

所谓条件控制语句，就是对语句中不同条件的值进行判断，进而根据不同的条件执行不同的语句。在条件控制语句中主要有两个语句：if 条件控制语句和 switch 多分支语句。

3.2.1　if 条件控制语句

if 条件控制语句是所有流程控制语句中最简单、最常用的一个，根据获取的不同条件判断执行不同的语句，应用范围十分广泛，无论程序大小几乎都会应用到该语句。其语法如下：

```
if (expr)
  statement ;                    //这是基本的表达式
if () {}                         //这是执行多条语句的表达式
if () {}else {}                  //这是通过 else 延伸了的表达式
if () {}elseif() {} else {}      //这是加入了 elseif,可同时判断多个条件的表达式
```

参数 expr 按照布尔求值。如果 expr 的值为 true，将执行 statement；如果值为 false，则忽略 statement。if 语句可以无限层地嵌套到其他 if 语句中去，实现更多条件

的执行。

【例 3-1】 if 语句的应用。

```
<?php
    //使用 rand()函数生成一个随机数
    $num =rand(1,10);
    if($num %2 ==0){
        echo "\$num=$num <BR>";
        echo "$num 是一个偶数。";
    }
?>
```

运行结果为：

```
$num=2
2 是一个偶数。
```

提示：rand()函数用来生成一个随机数，格式为 int rand(int mix,int max)，该函数返回一个 mix 到 max 之间的随机数。如果没有参数，则返回 0 到 RAND MAX 之间的随机整数。

其中，else 的功能是当 if 语句在参数 expr 的值为 false 时执行其他语句，即在执行的语句不满足该条件时，执行 else 后大括号中的语句。

【例 3-2】 if…else 的应用。

```
<?php
    //使用 rand()函数生成一个随机数
    $num =rand(1,10);
    if($num %2 ==0){
        echo "\$num=$num <BR>";
        echo "$num 是一个偶数。";
    } else {
        echo "\$num=$num <BR>";
        echo "$num 是一个奇数。";
    }
?>
```

运行结果为：

```
$num=3
3 是一个奇数。
```

在同时判断多个条件的时候，PHP 提供了 elseif 语句来扩展需求。elseif 语句放置在 if 和 else 语句之间，满足多条件同时判断的需求。

if 语句的流程如图 3-7、图 3-8 和图 3-9 所示。

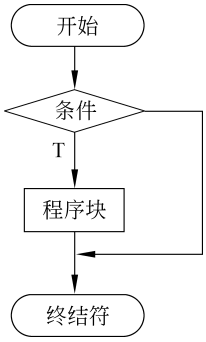


图 3-7　if 语句流程图

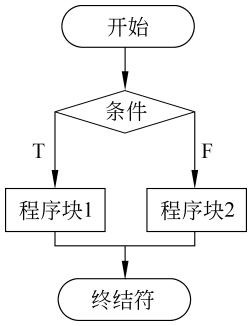


图 3-8　if…else 语句流程控制图

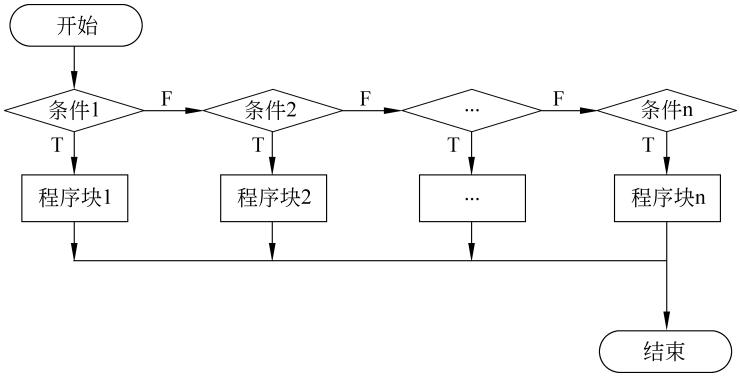


图 3-9　elseif 语句的流程控制图

【例 3-3】 从文本框输入一个百分制分数，单击“提交”按钮后，输出成绩等级：90 分以上记为“A”，80～89 分记为“B”，70～79 分记为“C”，60～69 分记为“D”，60 分以下记为“D”。

```
<html >
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>百分制分数</title>
    </head>
    <body>
        <form id="form1" name="form1" method="post" action="">
            <input type="text" name="score" id="score" />
            <input type="submit" name="button" id="button" value="提交" />
        </form>
        <?php
            if (isset($_POST["button"])){
                $score=$_POST["score"];
                if ($score>=90) $grade='A';
                elseif ($score>=80) $grade='B';
```

```
                elseif ($score>=70) $grade='C';
                elseif ($score>=60) $grade='D';
                else $grade='E';
                echo "成绩等级为".$grade;
            }
        ?>
    </body>
</html>
```

运行结果为：

```
输入：86
显示：成绩等级为 B
```

3.2.2 switch 多分支语句

switch 语句和 if 条件控制语句类似，实现将同一个表达式与多个不同值比较，获取相同值，并且执行相同值所对应的语句。其语法如下：

```
<?php
    switch ( expr ){                    //expr 条件为变量名称
        case expr1:                     //case 后的 expr1 为变量的值
            statement1;                 //冒号后的是符合该条件时要执行的部分
            break ;                     //使用 break 来跳出循环体
        case expr2 :
            statement2 ;
            break ;
        default:
            statementN;
            break;
    }
?>
```

参数说明如表 3-1 所示。

表 3-1 switch 语句的参数介绍

参 数	说 明
expr	表达式的值，即 switch 语句的条件变量的名称
expr1	放置于 case 语句之后，是要与条件变量 expr 进行匹配的值中的一个
statement1	在参数 expr1 的值与条件变量 expr 的值相匹配时执行的代码
break	终止语句的执行，即当语句在执行过程中，遇到 break 就停止执行，跳出循环体
default	case 的一个特例，在任何其他 case 都不匹配的情况下与之匹配，并且是最后一条 case 语句

switch 语句的流程控制如图 3-10 所示。

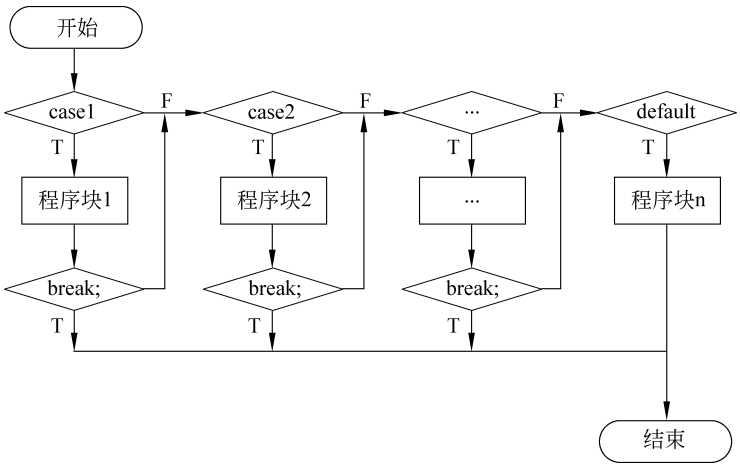


图 3-10　switch 语句的流程控制图

注意：

(1) 表达式的类型可以是数值型或者字符串型。

(2) 多个不同的 case 可以执行同一个语句块。

【例 3-4】 应用 switch 语句判断成绩的等级情况。

```
<?php
    $cont=49;                    //以下代码实现了根据$cont 的值,判断成绩等级的功能
    switch($cont) {
        case $cont=?=100;        //如果$cont 的值等于 100,则输出“满分”
            echo"满分";
            break;
        case $cont>=90;          //如果$cont 的值大于等于 90,则输出“优秀”
            echo"优秀";
            break;
        case $cont>=60;          //如果$cont 的值大于等于 60,则输出“及格”
            echo"及格";
            break;
        default:                 //如果$cont 的值小于 60,则输出“不及格”
            echo "不及格";
?>
```

运行结果为：

```
不及格。
```

注意：if 和 switch 语句可以从使用的效率进行区别，也可以从实用性角度区分。如果从使用的效率进行区分，在对同一个变量的不同值进行条件判断时，使用 switch 语句

的效率相对更高一些，尤其是判断的分支越多越明显。

如果从语句实用性的角度区分，那 switch 语句肯定不如 if 条件语句。if 条件语句是实用性最强和应用范围最广的语句。

在程序开发的过程中，if 和 switch 语句的使用应该根据实际情况而定，不要因为 switch 语句的效率高就一味地使用，也不要因为 if 语句常用就不应用 switch 语句。要根据实际情况，具体问题具体分析，使用最适合的条件语句。在一般情况下可以使用 if 条件语句，但是在实现一些多条件的判断中，特别是在实现框架的功能时，就应该使用 switch 语句。

3.3 循环控制语句

循环语句是在满足条件的情况下反复地执行某一个操作。在 PHP 中，提供了 4 种循环控制语句，分别是 while 循环语句、do…while 循环语句、for 循环语句和 foreach 循环语句。

3.3.1 while 循环语句

while 循环语句是循环控制语句中最简单的一个，也是最常用的一个。while 循环语句先对表达式的值进行判断，当表达式的值为非 0 值时，执行 while 语句中的内嵌语句；当表达式的值为 0 值时，则不执行 while 语句中的内嵌语句。该语句的特点是：先判断表达式，后执行语句。while 循环控制语句的操作流程如图 3-11 所示。

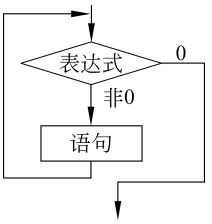


图 3-11 while 循环控制语句的操作流程

其语法如下：

```
while (expr){
    statement;              /* 先判断条件，当条件满足时执行语句块，否则不向下执行 */
}
```

只要 while 表达式 expr 的值为 true，就重复执行嵌套中的 statement 语句；如果 while 表达式的值一开始就是 false，则循环语句一次也不执行。

【例 3-5】 将 10 以内的偶数输出，若不是则不输出。

```
<?php
    $num =1;
```

```
    $str ="10 以内的偶数为: ";
    while($num <=10){
        if($num %2 =?=0){
            $str .=$num." ";
        }
        $num++;
    }
    echo $str;
?>
```

运行结果为：

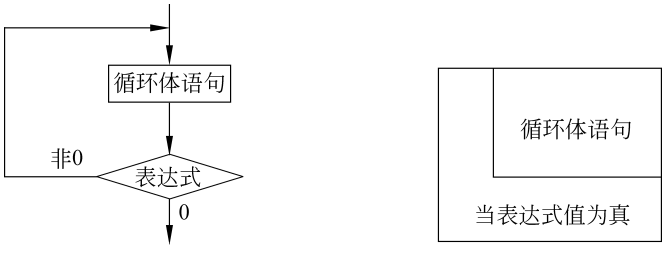
```
10 以内的偶数为: 2 4 6 8 10
```

3.3.2　do…while 循环语句

do…while 语句也是循环控制语句中的一种，使用方式与 while 相似，也是通过判断表达式的值来输出循环语句。其语法如下：

```
do{    /* 程序在未经判断之前先进行了一次循环，循环到 while 部分才判断条件，即使条件
          不满足，程序也会运行一次 */
    statement;
}while(expr);
```

该语句的操作流程是：先执行一次指定的循环体语句，然后判断表达式的值，当表达式的值为非 0 时，返回重新执行循环体语句，如此反复，直到表达式的值等于 0 为止，此时循环结束。其特点是先执行循环体，然后判断循环条件是否成立。do…while 循环语句的操作流程如图 3-12 所示。



(a) 操作流程图　　(b) N-S流程图

图 3-12　do…while 循环语句的操作流程

【例 3-6】 通过 do…while 语句计算一个员工的工龄工资增加情况。

```
<?php
    $a=1;                    //定义变量$a 的值为 1
```

```
    $year=5;
    do{
        $price=50*12*$a;
        echo "您第".$a."年的工龄工资为<b>".$price."</b>元<br>";
        $a++;
    }while($a<=$year);
?>
```

运行结果为：

```
您第 1 年的工龄工资为 600 元
您第 2 年的工龄工资为 1200 元
您第 3 年的工龄工资为 1800 元
您第 4 年的工龄工资为 2400 元
您第 5 年的工龄工资为 3000 元
```

如果使用 do…while 语句计算员工的工龄工资，当变量 a 的值等于 6 时，会得到一个意想不到的结果。下面就来具体地操作一下，看看具体会得到一个什么样的结果。定义变量 a 的值为 6，重新执行示例，其代码如下：

```
<?php
    $a=6;                //当直接定义变量$a 的值为 6 时，仍可以输出第 6 年的工资
    $year=5;             //定义初始变量$year=5
    do{
        $price=50*12*$a;
        echo "您第".$a."年的工龄工资为<b>".$price."</b>元<br>";
        $a++;
    }while($a<=$year); //当$year 等于 5 时程序没有停止，继续计算第 6 年工资，当$year 等
                       //于 6 时，判断条件不符合，停止循环，但是第 6 年的工资已经输出了
?>
```

运行结果为：

```
您第 6 年的工龄工资为 3600 元
```

注意：这就是 while 和 do…while 语句之间的区别。do…while 语句是先执行后判断，无论表达式的值是否为 true，都将执行一次循环；而 while 语句则是先判断表达式的值是否为 true，如果为 true 则执行循环语句，否则将不执行循环语句。

do…while 循环语句的条件表达式后边必须加上分号作为该语句的结束。

编写这个示例意在说明 while 语句与 do…while 语句在执行判断上的一个小小区别，在实际的程序开发中不会出现上述的这种情况。

3.3.3 for 循环语句

for 循环语句是 PHP 中最复杂的循环控制语句，拥有 3 个条件表达式。其语法如下：

```
for (expr1; expr2; expr3){
    statement
}
```

for 循环语句的参数说明如表 3-2 所示。

表 3-2　for 循环语句的参数说明

参 数	说　明
expr1	必要参数。第 1 个条件表达式，在第一次循环开始时被执行，即初始值
expr2	必要参数。第 2 个条件表达式，在每次循环开始时被执行，决定循环是否继续，即循环条件
expr3	必要参数。第 3 个条件表达式，在每次循环结束时被执行，即循环变量
statement	必要参数。满足条件后，循环执行的语句，即循环体

其执行的过程是：首先执行 expr1；然后执行 expr2，并对 expr2 的值进行判断，如果值为 true，则执行 for 循环语句中指定的内嵌语句，如果值为 false，则结束循环，跳出 for 循环语句；最后执行 expr3（切记是在 expr2 的值为 true 时），返回 expr2 继续循环执行。for 循环语句的操作流程如图 3-13 所示。

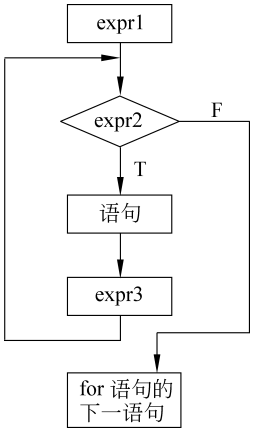


图 3-13　for 循环语句的操作流程

【例 3-7】 使用 for 循环来计算 2～100 中所有偶数之和。

```
<?php
    $b="";
    for($a=0;$a<=100;$a+=2){                    //执行 for 循环
        $b=$a+$b;                               //计算所有偶数之和
    }
    echo "结果为：<b>".$b."</b>";
?>
```

运行结果为：

```
结果为：2550
```

注意：在编程时，有时会遇到使用 for 循环的特殊语法格式来实现无限循环。语法格式为：

```
for(;;){
    …
}
```

对于这种无限循环可以通过 break 语句来跳出循环。例如：

```
for(;;){
    if(x <20)
        break;
    x++;
}
```

3.3.4 foreach 循环语句

foreach 循环控制语句自 PHP 4 开始被引入，主要用于处理数组，是遍历数组的一种简单方法。如果将该语句用于处理其他的数据类型或者初始化的变量，将会产生错误。该语句的语法有两种格式。

格式一：

```
foreach (array_expression as $value){
    statement
}
```

格式二：

```
foreach (array_expression as $key =>$value){
    statement
}
```

参数 array_expression 指定要遍历的数组，其中 $ value 是数组的值，$ key 是数组的键名；statement 是满足条件时要循环执行的语句。

在格式一中，当遍历指定 array_expression 数组，每次循环时，将当前数组单元的值赋给变量 $ value，并且将数组中的指针移动到下一个单元。

在格式二中的应用是相同的，只是在将当前单元的值赋给变量 $ value 的同时，将当前单元的键名也赋给了变量 $ key。

说明：

当把 foreach 语句用于其他数据类型或者未初始化的变量时会产生错误。为了避免

这个问题，最好使用 is_array()函数先来判断变量是否为数组类型。如果是，再进行后面操作。

【例 3-8】 foreach 输出数组元素值的应用。

```
<?php
    $a =array(1,2,3,4,5,6);
    foreach($a as $b)
        echo $b;
?>
```

运行结果为：

```
1 2 3 4 5 6
```

3.4　跳 转 语 句

跳转语句有两个：break 语句和 continue 语句。跳转语句使用起来非常简单而且容易掌握，主要原因是它们都被应用在指定的环境中，如 for 循环语句中。在程序运行中，也可以使用 return 语句和 exit 语句来终止程序的运行。本节将进行详细的介绍。

3.4.1　break 跳转语句

break 关键字可以终止当前的循环，包括 while、do…while、for、foreach 和 switch 在内的所有循环控制语句。

break 语句不仅可以跳出当前的循环，还可以指定跳出哪几重循环。其格式为：

```
break  n;
```

参数 n 指定要跳出的循环数量。break 关键字的流程图如图 3-14 所示。

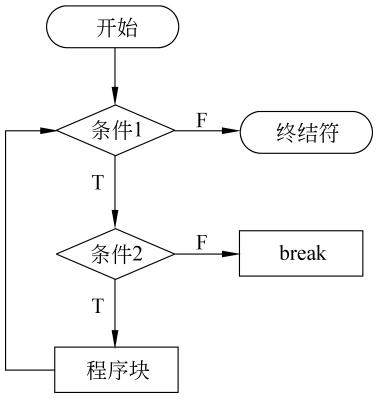


图 3-14　break 关键字的流程图

【例 3-9】 计算半径 1～10 的圆面积，直到面积大于 100 时为止。

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

运行结果为：

```
r=1, area=3.14
r=2, area=12.56
r=3, area=28.26
r=4, area=50.24
r=5, area=78.5
```

3.4.2 continue 跳转语句

程序执行 break 后，将跳出循环，并开始继续执行循环体的后续语句。continue 跳转语句的作用没有 break 那么强大，只能终止本次循环，而进入到下一次循环中。在执行 continue 语句后，程序将结束本次循环的执行，并开始下一轮循环的执行操作。continue 也可以指定跳出几重循环。continue 跳转语句的流程图如图 3-15 所示。

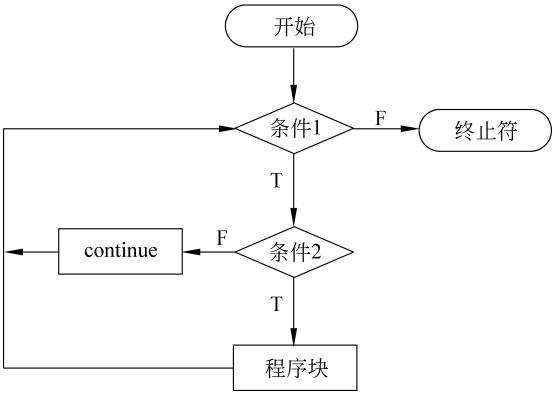


图 3-15 continue 跳转语句的流程图

【例 3-10】 使用 for 循环来计算 1～100 中所有奇数的和。在 for 循环中，当循环到偶数时，使用 continue 实现跳转，然后继续执行下一轮循环的运算。

```
<?php
    $sum=0;
    for($i=1;$i<=100;$i++){
        if($i%2=?=0){
            continue;
        }
        $sum=$sum +$i;
    }
    echo $sum;
?>
```

运行结果为：

```
2500
```

说明：

break 和 continue 语句都有实现跳转的功能，但两者是有区别的：continue 语句只是结束本次循环，并不是终止整个循环的执行，而 break 语句则是结束整个循环。

3.4.3　return 语句

在大部分编程语言中，return 语句可以将函数的执行结果返回，PHP 中的 return 的用法也类似。return 的作用是将函数的值返回给函数的调用者，如果在全局作用域内使用 return 关键字，那么将终止脚本的执行。return 语句在函数中使用时，有以下两个作用。

（1）如果在函数中执行了 return 语句，它后面的语句将不会被执行，也就是将退出该函数。

（2）return 语句可以向函数调用者返回函数体中任意确定的值，也就是常说的函数返回值。

【例 3-11】 return 语句应用示例。

```
<?php
    function sum($a,$b){
        return $a +$b;
        return $a * $b;
    }
    $temp =sum(3,5);
    echo $temp;
?>
```

在上述例子中，输出的结果为 8，而后面的 return $a * Sb 语句将不会被执行，也就是将函数的返回值给 $temp，然后进行输出。

3.4.4　exit 语句

程序执行过程中，总会发生一些错误，例如被零除、打开一个不存在的文件或者数据

库连接失败等情况。当程序发生错误时，应用控制程序应立即终止执行剩余代码，PHP 的 exit 语句（或者 die 语句）可以实现这个功能。exit 语句终止整个 PHP 程序的执行，即后续代码不会执行。

exit 语句的语法格式为：

```
void exit ( [string message] )
```

exit 语句的功能是，输出字符串信息 message，然后终止 PHP 程序的运行。

注意：字符串信息 message 必须位于小括号内。

【例 3-12】 exit 语句的应用。

```
<?php
    @($a =2/0) or exit("发生被零除错误!");
    echo "exit 后面的语句将不会运行!";
?>
```

运行结果为：

```
发生被零除错误!
```

之所以 exit 不是函数，而是一个语言结构，是因为上述例子可以修改为：

```
<?php
    @($a =2/0) or exit;
    echo "exit 后面的语句将不会运行!";
?>
```

PHP 还提供了 die 语言结构来终止程序的运行，die 可以看作是 exit 的别名。例如，上述例子可以修改为：

```
<?php
    @($a =2/0) or die("发生被零除错误!");
    echo "die 后面的语句将不会运行!";
?>
```

3.5 PHP 文件间包含

引用外部文件可以提高代码的重用性，这是 PHP 编程的重要技巧。PHP 提供了 4 个非常简单却很有用的包含函数（include()、require()、include_once()、require_once()），它们允许重复使用任何类型的代码。使用任意一个语句均可将一个文件载入 PHP 脚本中，从而提高代码的重用性，提高代码维护和更新效率。

3.5.1 include()函数

include()函数的语法格式如下：

```
mixed include(string resource)
```

include()函数的功能是,将一个资源文件 resource 载入到当前 PHP 程序中。字符串参数 resource 是一个资源文件的文件名,该资源可以是本地 Web 服务器上的资源,如图片、HTML 页面、PHP 页面等,也可以是互联网上的资源。若找不到资源文件 resource,返回 false;若找到资源文件 resource,且资源文件 resource 没有返回值时,返回整数 1,否则返回资源文件 resource 的返回值。

注意:

(1) 使用 include()函数载入文件时,如果被载入的文件中包含 PHP 语句,这些语句必须使用 PHP 开始和结束标记进行标识。

(2) resource 资源是互联网上的某个资源时,需要把配置文件 php.ini 中的选项 allow_url_ include 设置为 on(allow_url_include=on),否则不能引用互联网资源。

【例 3-13】 程序文件位于同一个目录下的 include 语句的应用(即 include.php 和 main.php 位于同一个目录下)。

程序文件一:include.php。

```
<?php
    $color ='red';
    $fruit ='apple';
    echo "这是被引用的文件输出!<br/>";
?>
```

程序文件二:main.php。

```
<?php
    echo "A $color $fruit<br/>";
    include "include.php";
    echo "A $color $fruit<br/>";
?>
```

运行程序文件二的结果:

```
Notice: Undefined variable: color in C:\xampp\htdocs\chap3\index.php on
line 16
Notice: Undefined variable: fruit in C:\xampp\htdocs\chap3\index.php on
line 16
A
这是被引用的文件输出!
A red apple
```

3.5.2　include()函数和 require()函数的区别

应用 require()函数来调用文件,其应用方法与 include()函数类似,但还存在一定的

区别，具体如下。

（1）在使用 require()函数调用文件时，如果被引用文件发生错误或不能找到被引用文件，引用文件将给出警告信息及致命错误信息，然后终止程序运行。而 include()函数在没有找到文件时则会输出警告信息，但不会终止脚本的处理。

（2）使用 require()函数调用文件时，只要程序一执行，会立刻调用外部文件；而通过 include()函数调用外部文件，则只有程序执行到该函数时，才会调用外部文件。

【例 3-14】 include()和 require()函数的区别示例。

```
<?php
    echo "<h2>include 包含不在的 conn.php 文件的结果：</h2>";
    include( "conn.php") ;
    echo "include 的内容";
?>
<?php
    echo "<h2>require 包含不在的 conn.php 文件的结果：</h2>";
    require "conn.php" ;
    echo "require 的内容";
?>
```

输出结果如图 3-16 所示。

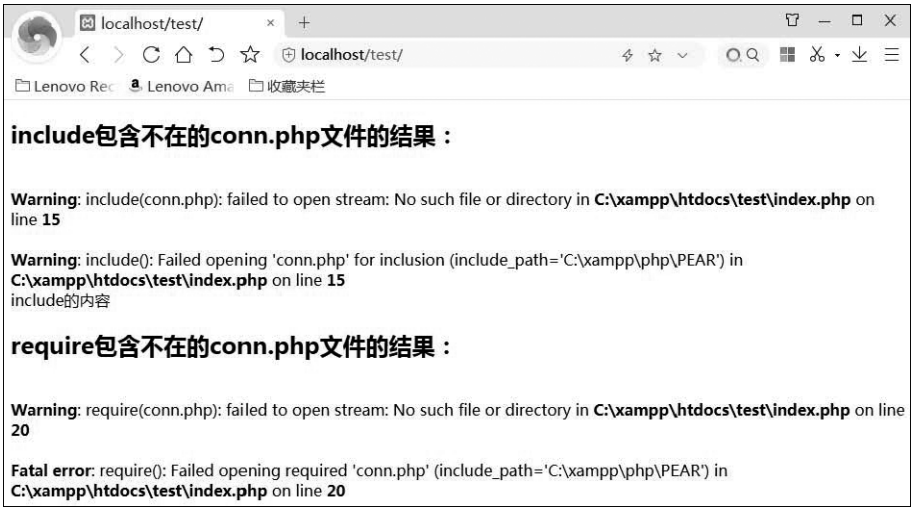


图 3-16　include()和 require()函数的区别示例运行结果

从例 3-14 的结果可以看到，include 包含不存在的文件时，是不会终止脚本运行的，而 require 会终止脚本运行。

3.5.3　include_once()函数

随着程序资源规模的扩大，同一程序多次使用 include()或者 require()函数的情况时有发生，而多次引用同一个资源也变得不可避免，这可能导致文件的引用混乱问题。为了

解决这类问题，PHP 提供了 include_once()函数和 require_once()函数，以确保同一个资源文件只能引用一次。

include_once()函数是 include()函数的延伸，其作用与 include()函数几乎是相同的，唯一的区别在于，include_once()函数会在导入文件前先检测该文件是否在该页面的其他部分被导入过，如果是则不会重复导入该文件。例如，要导入的文件中存在一些自定义函数，那么如果在同一个程序中重复导入这个文件，在第二次导入时便会发生错误，因为 PHP 不允许相同名称的函数被重复声明。

include_once()函数的语法如下：

```
void include_once (string filename);
```

filename 参数是指定的完整路径文件名。

include_once()函数的功能：将一个资源文件 resource 载入到当前 PHP 程序中。若找不到资源文件 resource，include_once()函数返回 false；若找到资源文件 resource，且该资源文件是第一次载入，include_once()函数返回整数 1；若找到资源文件 resource，且该资源文件已经载入，include_once 语句返回 true。

【例 3-15】 应用 include_once()函数引用并运行指定的外部文件 top.php。

```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=
        gb2312">
        <title>应用 include_once()函数包含文件</title>
    </head>
    <body>
        <table width="779" border="0" cellpadding="0" cellspacing="0">
            <tr>
                <td><?php include_once("top.php");?></td>
            </tr>
        </table>
    </body>
</html>
```

top.php 文件代码如下：

```
<html>
    <head>
        <title>被包含文件</title>
    </head>
    <body>
        <table width="779" height="80" border="0" cellpadding="0"
        cellspacing="0">
```

```
            <tr>
                <td bgcolor="#33CCFF">使用 include_once 语句引用的文件</td>
            </tr>
        </table>
    </body>
</html>
```

注意 include_once()函数和 require_once()函数的区别：include_once()函数在脚本执行期间调用外部文件发生错误时，产生一个警告，而 require_once()函数则导致一个致命的错误。

思考与练习

1. continue、break 语句在循环中分别起到什么作用？
2. 描述一下 include()函数和 require()函数的区别。
3. 举例说明 while 循环语句和 do…while 循环语句在应用上的不同点。
4. 关于 exit 语句与 die()函数的说法正确的是(　　)。
   A. 执行 exit 语句后会停止执行后面的脚本，而 die()函数无法做到
   B. 执行 die()函数后会停止执行后面的脚本，而 exit 语句无法做到
   C. die()函数等价于 exit 语句
   D. die()函数与 exit 语句有直接关系
5. 以下代码在页面上输出数据的行数是(　　)。

```
$attr =array(1,2,3,4);
while(list($key,$value) =each($attr))
{
    echo $key."=>".$value."<br>";
}
while(list($key,$value) =each($attr))
{
    echo $key."=>".$value."<br>";
}
```

A. 4　　B. 6　　C. 8　　D. 12

6. 语句 for($k=0;$k=1;$k++)和语句 for($k=0;$k==1;$k++)执行的次数分别是(　　)。
   A. 无限和 0　　B. 0 和无限　　C. 都是无限　　D. 都是 0
7. 可以结束循环的语句是(　　)。
   A. break　　B. if　　C. continue　　D. switch
8. 不论循环条件判断的结果是否为 true，(　　)循环至少执行一次。
   A. for　　B. while　　C. do…while　　D. 以上都可以