

第一章 变量

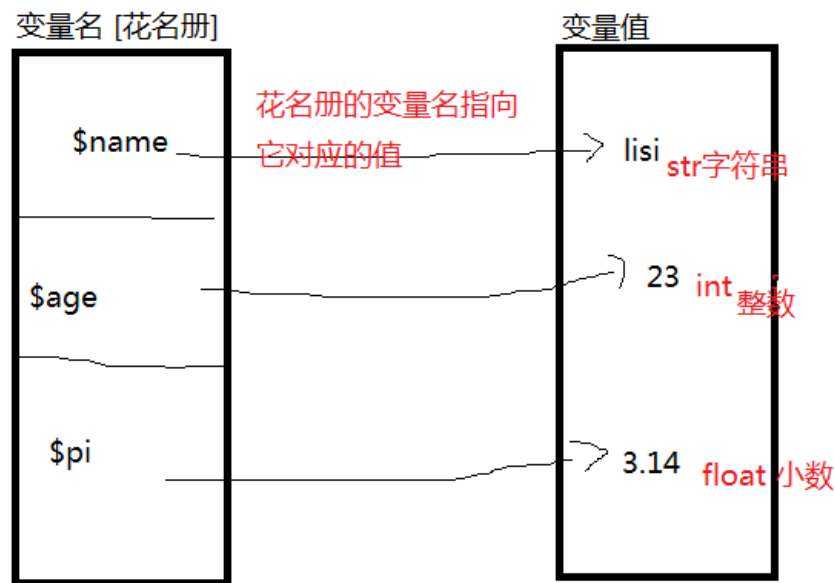
1.1:类型 (lesson 1)

整型,浮点型,字符串,布尔型,数组,对象,NULL,资源
变量是个盒子,盒子里面装的是变量的值
不同的盒子就是变量类型

- 1) 整型 [integer] 数学中的整数
2) 浮点型 [float,double] 数学中的小数
3) 字符串 [string] 一串字符
4) 布尔 [boolean] 真假
5) 数组 [array] 键值对复合数据
6) 对象 [Object] [在后面的面向对象中会学到]
7) NULL 没有值
8) 资源 [resource] “吸管”

变量有8种类型,不必死记硬背,在实际运用中加深理解

1.如何区分变量类型?
在计算机的世界里,最终只有两个数字,0,1
都是一对0,1 ,如何区分是 字符串的0100 0001表示字母A
还是理解为数字 65 呢?



盒子里面的变量值并不是只存储了变量值
还存储了它的变量类型

\$a	01000001	A
	string	
\$b	01000001	65
	integer	
\$null		NULL
	NULL	

NULL是什么？

NULL类型只标注了它的类型为NULL
它的值的字段是空的,NULL是没有值的

```
$a = 3; // 整型
$b = 3.14; // 浮点型
$c = null; // null型
$d = 'hello'; // 字符串
$e = true; // 布尔型

echo $D;
```

变量名称的命名规范:

[a-zA-Z0-9]和下划线(_)

- 1) 变量名是区分大小写的
- 2) 不能以数字开头

```
$a
$_
$_%
$3c
$c3
```

1.2: 变量检测(2)

echo 一个不存在的变量, 会报notice级别的错误,
所以要检测一下这个变量是否存在

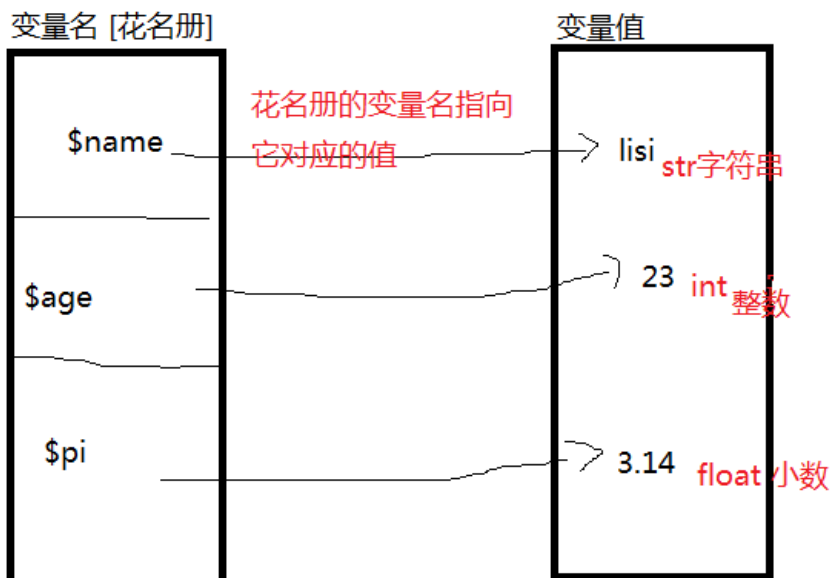
如何检测变量是否存在?

isset — 检测变量是否设置

已声明的变量返回 true, 未声明的变量返回 false.

检测一个变量是否存在:

就是看花名册中是否有这个变量名



```
$b = null;
$c = false;
$d = 0;
$e = '';

// 分别检测上述变量是否存在
if(isset($a)) {
    echo '变量a存在';
} else {
    echo '变量a不存在';
}
```

对于值为 NULL 的变量,也返回 false,因为null没有值
未曾声明的变量,当然也不存在

1.3:类型检测(3)

检测一个变量,php是把它存储成一个什么样的类型的
对于php而言获取它的变量类型是很简单的
因为箱子里面已经存储了它的变量类型
gettype — 获取变量的类型 [现成的系统函数]

```
$a = false;
echo gettype($a);

$b = "1";
echo gettype($b);

$c = 1.11;
echo gettype($c);

$d = 'hello';
echo gettype($d);

$e = null;
echo gettype($e);
```

判断变量是否是某种类型

<code>is_float()</code>	<code>[is_double]</code>	检测变量是否为浮点型
<code>is_int()</code>	<code>[is_integer]</code>	检测变量是否为整型
<code>is_string()</code>		检测变量是否为字符串
<code>is_object()</code>		检测变量是否为对象
<code>is_array()</code>		检测变量是否为数组
<code>is_resource</code>		检测变量是否为资源类型
<code>is_bool</code>		检测变量是否是布尔型
<code>is_null</code>		检测变量是否为 NULL


```
$a = 'hello';
if(is_string($a)) {
    echo 'a是字符串';
} else {
    echo 'a不是字符串';
}
```

1.4:调试打印变量(4)

开发程序的时候,不可避免的要变量的值打印出来看看

如何调试打印变量?

echo 字符串,数字

print_r 打印层次化的数据,比如:数组,对象

var_dump 打印变量的类型及其值[调试代码比较方便]

```
$a = 'hello';
$b = array(1,2,"3");
$c = false;
$d = null;
$e = 18;
$f = true;

//布尔型的true会打印出1,false和null什么都不显示
echo $a,$b,$c,$d,$e,$f,'<hr>';

//print_r 打印层次化的数据,比如数组和对象
print_r($b);
```

```
print_r($c);
print_r($d);
print_r($f);

//不要用echo和print_r打印布尔型的值,因为会干扰我们
//用var_dump打印布尔和null
var_dump($c);
var_dump($d);
```

1.5:类型转换(5)

PHP中,变量的类型是可以随时转换的,非常的灵活
最常见的是字符转与数字之间的相互转换,
或者是数字/字符串 -> 布尔值的转换

字符串到数字的转换

从左到右截取,直到碰到不合法的数字,截取下来的部分转换为数字

```
$a = '12';
$b = $a + 3;
var_dump($b);

$a = '12.5hello'; //$a = '12.5hello99';
$b = $a + 3;
var_dump($b);

$a = 'word12.5hello';
$b = $a + 3;
var_dump($b);
```

数字到字符串的转换

```
$a = 123;
$b = $a . 'hello';
var_dump($b);
```

数字/字符串/数组等到布尔型值的转换

```
$b = 3;
if( $b ) {
    echo 'b is true';
} else {
    echo 'b is false';
}
```

if判断的应该是布尔型的值,那数字3被转成布尔型来理解
那它到底应该理解为真还是假呢?

以下值,都被理解为成布尔型值的假

","0","0.0,0.0,false,NULL,array();

而其它值,都被当成布尔型的真

```
if('' == false) {
    echo '空字符串果然假';
}
```

empty(var) — 检查一个变量是否为空

如果 var 是非空或非零的值,则 empty() 返回 FALSE

换句话说,"","0、0.0、"0"、NULL、FALSE、array();

以及没有任何属性的对象都将被认为是空的,如果 var 为空,则返回 TRUE

```
$arr = array();
if(empty($arr)) {
    echo '变量为空';
}
```

1.6:赋值(6)

赋值有两种方式：

- 1.传值赋值;(两个人看两台电视同一个台)
- 2.引用赋值;(两个人看同一台电视)

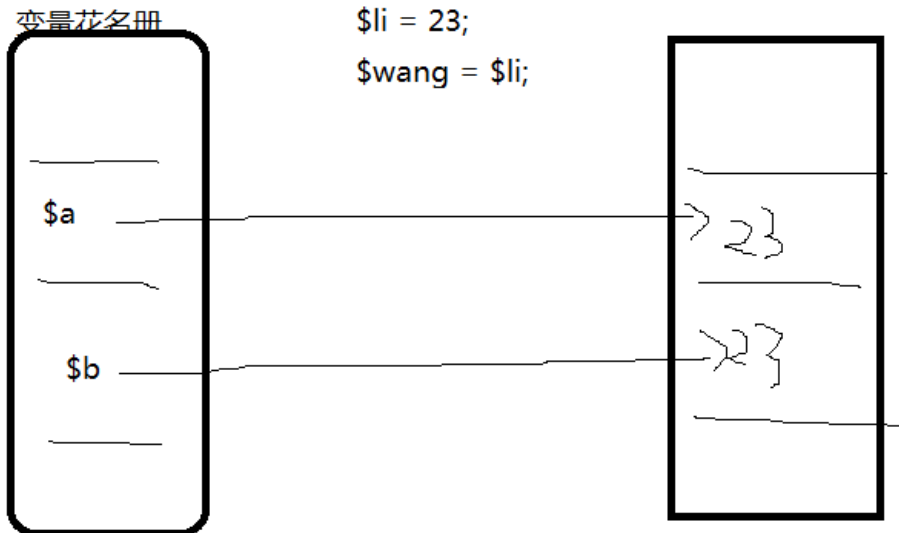
1.传值赋值

变量名其实不是贴在盒子上的,而是有个变量表(像班级的花名册)

变量值和变量类型放在盒子里面;变量表中的变量名指向它所对应的盒子.

```
$li = 23;  
$wang = $li;  
echo $li, '~', $wang;
```

以上代码在内存中发生了什么？



改变\$li的值,\$wang的值会发生变换吗

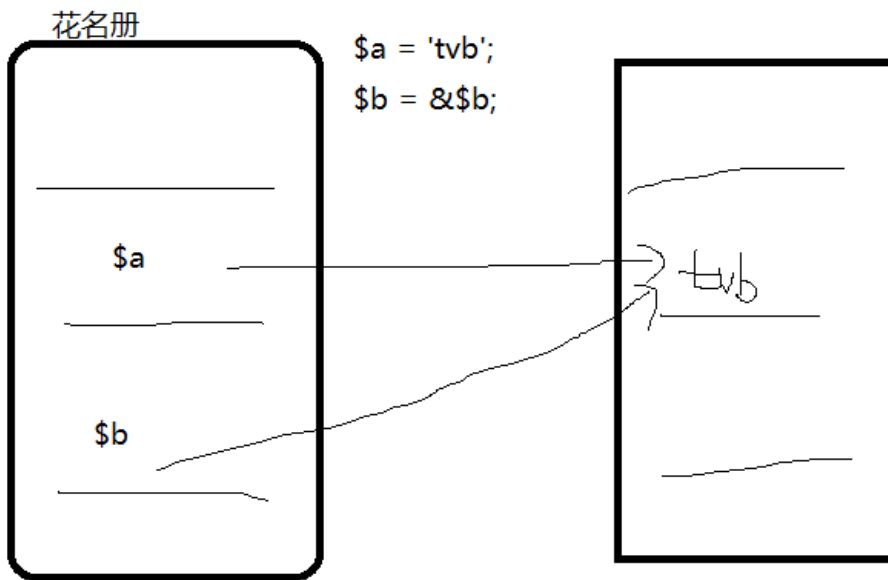
```
$li = 99;  
echo $li, '~', $wang;
```

这个赋值过程,是把\$li的值,赋给\$wang

*2. 引用赋值 *

```
$a = 'tvb';  
$b = &$b; // $a, $b 共同指向同一个值  
echo $a, '~', $b;
```

加&引用赋值



改变\$a的值

```
$a = 'btv';  
echo $a, '~', $b;
```

1.7:销毁(7)

为什么要销毁变量?

因为,有时比较大的数组,或者比较大的对象

特别的GD画图时,比较耗费资源,将它unset掉,可以及时释放出内存

unset(变量名); 销毁指定的变量

首先从变量表(花名册)中删除变量名,再找到它对应的盒子也删除掉.

```
$a = 99;  
//unset($a);  
  
if(isset($a)) {  
    echo 'a存在';  
} else {  
    echo 'a不存在';  
}
```

注意: 引用赋值,如果两个变量指向同一个盒子,当销毁其中一个变量的时候,盒子是不能被销毁掉的.

```
$a = 99;  
$b = &$a;  
unset($a);  
echo $a, $b; //报一个notice的错误
```

重新给\$a赋值一个新值

```
$a = 18;  
echo $a, $b;
```

1.8: 动态变量名(8)

动态变量名,可以体现php非常灵活的一个地方

用变量的值去做另一个变量的名

```
$laoda = 'liubei';  
echo $laoda , '<br >';  
  
$paihang = 'laoda';  
echo $paihang , '~' , $$paihang;
```

```
//排行
$rank = 'paihang';
echo $$$rank;
```

写代码注意简洁和易读性

1.9 常用面试题(章节练习)

<http://www.zixue.it/thread-115-1-1.html>

<http://www.zixue.it/thread-111-1-1.html>

<http://www.zixue.it/thread-123-1-1.html>

<http://www.zixue.it/thread-120-1-1.html>

第二章 运算符

2.1 算数运算符的三个注意点 (9)

- - * / % 加,减,乘,除,取模

1.除以: / (在数学上除数不能为零)

```
$a = 10;
$b = 0;
echo $a / $b;
```

为什么除数不能为零

一个大饼,你把它平均分成0分,请问你如何分?

如果除数为零就意味着是“将被除数分为零份”,那也就是不去分被除数

既然除数为零,那就是什么也不做,也就不会再有除的概念了,所以就不存在除了

2.加: +

```
$a = 2000000000;
$b = 2000000000;
$c = $a + $b;

echo gettype($a) , '<br >';
echo gettype($b) , '<br >';
echo gettype($c) , '<br >';
```

因为int型在php中目前只能存到 +21亿多

当我们运算的得数太大的时候,它就会自动转换成一个更大的类型,浮点型

3.取模: %

```
$a = 1;
$b = 5;
echo $a%$b, '<br />';

$a = 10;
$b = 2;
echo $a%$b, '<br />';

$a = -10;
$b = 3;
echo $a%$b, '<br />';

$a = 10;
$b = -3;
echo $a%$b, '<br />';

$a = -10;
$b = -3;
echo $a%$b, '<br />';
```

取模算法时,结果的正负仅取决于被除数,他和被除数一致
被除数/除数

2.2: 比较运算符 (10)

, >= , < , <= , == , != , === , !==

凡运算,必有运算结果,比较运算符的运算结果是布尔型值

比较运算的运算结果

```
$a = 5;  
$b = 2;  
$c = $a>$b;  
var_dump($c);
```

==和===的区别

```
$c = (3 == '3');  
var_dump($c);  
//== 只验证值是否相等  
  
$c = (3 === '3');  
var_dump($c);  
//要求变量类型相等,且值也要相等;
```

全等于和等于的例子

strpos — 查找字符串首次出现的位置(区分大小写)

如果找到,返回字符串中首次出现的数字位置,如果没找到返回boolean(false)

```
$str = 'abcdef';  
$pos = strpos($str, 'a');  
var_dump($pos);  
  
//0被if理解为false  
if($pos == false) {  
    echo '没找到';  
} else {  
    echo '找到了';  
}  
  
if($pos === false) {  
    echo '没找到';  
} else {  
    echo '找到了';  
}
```

2.3 三元运算符 (11)

一个表达式,有两个结果,如果为真取结果1,如果为假取结果2

false

条件 ? 值1:值2;

true

表达式: 表达式?

结果1: \$a

结果2: \$b

如果表达式为真,返回结果1,如果为假,返回结果2

给定两个数,如何找到较大的数?

```
$a = 10;
```



```
$b = 5;
if($a >= $b) {
    echo $a;
} else {
    echo $b;
}
```

三元运算符

```
$a = 5;
$b = 3;

$c = ($a>=$b? $a : $b);
echo $c;
```

*课后作业: *

给定 \$a,\$b,\$c3 个整型值,返回其中最大的值

提示:嵌套三元运算符,嵌套时,为防止混乱,建议多用()包起来表达式

2.4 逻辑运算符 (12)

&&(并且)

||(或者)

某美女要求,有房且有车,才嫁

```
$house = true;
$car = false;

if( $house && $car ) {
    echo 'marry';
} else {
    echo 'sorry';
}
```

某女要求,有房或有车,就嫁

```
$house = true;
$car = false;

if( $house || $car ) {
    echo 'marry';
} else {
    echo 'sorry';
}
```

或者就是有一个为真就行,而并且需要都为真才行

2.5 递增递减运算符 (13)

递增: ++

递减: --

++,--在后:

++

```
$a = 5;
$b = $a++;
echo $a,$b;//6,5
```

凡运算必有运算结果,当++放在变量后面的时候:

1.\$a先把值赋给了\$b

2.\$a再将自己本身的价值加1

```
$a = 5;
$b = $a--;
echo $a,$b;//4,5
```

当--放在变量后面的时候:

1.\$a先把值赋给了\$b

2.\$a再将自己本身的值减1

++,--在前:

++

```
$a = 5;
$b = ++$a;
echo $a,$b;//6,6
```

凡运算必有运算结果,当++放在变量后面的时候:

1.\$a先把自己本身的值加1

2.\$a再将自己本身的值赋给了\$b

同理

```
$a = 5;
$b = --$a;
echo $a,$b;//4,4
```

当--放在变量后面的时候:

1.\$a先把自己本身的值减1

2.\$a再将自己本身的值赋给了\$b

不推荐使用++,--

因为:++,--的操作缺乏原子性,就是一句话应该就干一件事,但是++,--一句话干了两件事,在复杂的语句下,有可能会产生歧义
python中,没有++,--的操作

```
$a = 5;
$b = $a++;
```

等价于下面3句

```
$a = 5;
$b = $a;
$b = $b+1;
```

每句话都是一个原子操作

虽然多1行代码,但是语义清晰

2.6 字符串运算符 (14)

. 拼接运算

```
$a = 'hello';
$b = 'word';
$c = $a . $b;
echo $c;

$c = $c . '123456';
echo $c;
```

下面哪个运行速度更快?

```
echo $a . $b;
echo $a,$b;
```

字符串并不是可以用逗号拼接的

只是echo可以输出多个变量,用逗号隔开即可
而点是先拼接再输出
所以逗号更快一些,不需要拼接字符串再输出
比如:
我想拿出来两个橡皮泥给你,是两个直接给你更快,

还是将两个橡皮泥揉成一块给你快

2.7 赋值运算符 (15)

凡运算,必有结果,赋值运算也有结果
= 它是将等号右边的值赋给左边的变量,运算结果就是等号右边的值
注意有 2 个作用:1 是赋值,2 是返回值

```
$a = 3;  
$res = ($a = 3);  
var_dump($res);
```

这是一个赋值运算,有 2 个作用 1 是把 3 赋给\$a, 2 是返回运算结果,即 3

2.8 常用面试题

<http://www.zixue.it/thread-151-1-1.html>

第三章 控制结构

3.1 switch case 控制结构 (16)

三大控制结构:
顺序,选择,循环

1.判断名次给与冠军,亚军,季军和谢谢参与

```
$res = 4;  
if($res == 1) {  
    echo '你是冠军';  
} else if($res == 2) {  
    echo '你是亚军';  
} else if($res == 3) {  
    echo '你是季军';  
} else {  
    echo '谢谢参与';  
}
```

2.当有很多种选择的时候

我们可以用switch case(代表有几条分支,往那里分)

1)首先我们给定一个变量[\$res]来判断

```
switch($res)
```

2)再次我们给定它不同的分支

```
$res = 4;  
switch($res) {  
    case 1:  
        echo '冠军';  
  
    case 2:  
        echo '季军';  
  
    case 3:  
        echo '亚军';  
  
    default:
```

```
    echo '谢谢参与';  
}
```

3)加上break

当case判断条件成立时,后面不会继续判断,会去直接执行echo代码

所以我们要在每个后加上break去阻止后面代码的执行

```
$res = 2;  
switch($res) {  
    case 1:  
        echo '冠军';  
        break;  
  
    case 2:  
        echo '季军';  
        break;  
  
    case 3:  
        echo '亚军';  
        break;  
  
    default:  
        echo '谢谢参与';  
}
```

4)利用它的break特性,我们可以简化代码

判断名次给与冠军,季军,亚军3-5,后面都是谢谢参与

比较麻烦的写法:

```
$res = 2;  
switch($res) {  
    case 1:  
        echo '冠军';  
        break;  
  
    case 2:  
        echo '季军';  
        break;  
  
    case 3:  
        echo '亚军';  
        break;  
  
    case 4:  
        echo '亚军';  
        break;  
  
    case 5:  
        echo '亚军';  
        break;  
  
    default:  
        echo '谢谢参与';  
}
```

简化写法

```
$res = 2;  
switch($res) {  
    case 1:  
        echo '冠军';  
        break;  
  
    case 2:  
        echo '季军';  
        break;  
  
    case 3:  
        case 4:  
            case 5:
```

```
case 5:
echo '亚军';
break;

default:
echo '谢谢参与';
}
```

注意:

switch case 适合用在验证多个可能的确切值时使用

不适合用在判断范围

比如 60-80 分及格, 0-59 不及格,81-100 优 这种情况就不能用 switch case

因为case 后面只能写一个确切的值

3.2 while / do-while (17)

循环

1.while

先判断再执行

条件为真就执行,直到条件不为真才截至

如果上来条件就不为真,那一次都不会执行

```
while(条件为真){
    //执行体
}

$i = 1;
while ($i < 10) {
    echo $i , '<br >';
    $i = $i + 1;
}
```

2.do while

先执行再判断,如果条件上来就为假,那至少也能执行一次

```
do {
    //执行体
} while(条件为真)

$i = 10;
do {
    echo $i , '<br >';
    $i = $i + 1;
} while ($i < 10)
```

餐馆,先付钱再吃饭,还是先吃饭再付钱

3.3 for 循环 (18)

for和while都是一种循环

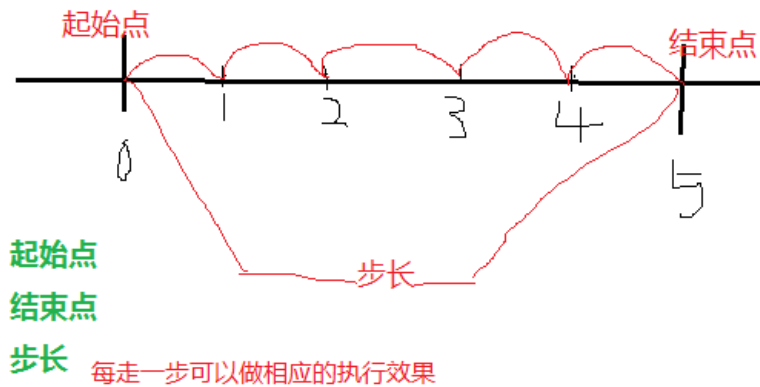
解释:所有循环的共同点

画图详解

在一条数轴上,循环就是有起始点(\$i = 0);

有结束点(\$i<=10);有步长(\$i++);

每走一个相应的步长,就执行一次代码(执行语句)



循环的一般要素：初始化(1),判断(2),执行体(3),修改变量(4)

```
for(1[起始点];2[结束点(到结束了没有)];4[步长]) {
    3[执行语句];
}
```

1.for循环打印0-9

```
for($i=0; $i<=9; $i=$i+1) {
    echo $i , '<br >';
}
```

2.步长可以不为1

```
for($i=0; $i<=9; $i=$i+3) {
    echo $i , '<br >';
}
```

3.初始变量可以为多个

```
for($i=1,$j=5 ; $i<=5; $i=$i+1,$j=$j-1){
    echo $i,'~', $j , '<br />';
}
```

4.for循环的改造

可以有其他的写法[类似while循环]

```
for($i=0; $i<=9; $i=$i+1) {
    echo $i , '<br >';
}
```

改造之后

```
$i=0;
for( ; $i<=9 ; ) {
    echo $i;
    $i=$i+1;
}
```

for循环很常用,要数量掌握

3.4 break 与 continue (19)

1.continue 下一个 (用next更形象一点)

整体的循环没有破坏掉,而是跳到下一个循环单位中

```
for($i=1; $i<=10; $i = $i+1){
    if($i == 4) {
        continue;
    }
    echo $i,'<br />';
}
```

```
}
```

2.break 破坏(将后面的执行语句破坏掉)

整体的循环都破坏掉了,循环到此结束掉了

```
for($i=1; $i<=10; $i+=1){  
    if($i == 4) {  
        break;  
    }  
    echo $i, '<br />';  
}
```

在c语言,java中都有类似与continue和break的这个东西

3.生活中的例子,相亲(只讲解)

遇到长得丑的直接continue掉,接着相亲下一个

遇到白马王子,直接break,后面的相亲对象就不见面了

1)continue

```
for($i=1; $i<=10; $i+=1) {  
    if ($i == 4) {  
        echo '照片丑,不约';  
        continue;  
    }  
    echo $i, '<br />';  
}
```

2.break

```
for($i=1; $i<=10; $i+=1) {  
    if ($i == 6) {  
        echo $i, '是白马王子';  
        break;  
    }  
    echo $i, '<br />';  
}
```

3.5 过桥问题 (20)

有了前面学的变量,表达式,控制结构,我们实际做一些编程的小题目.小题目在于锻炼你的编程思维

为什么遇到实际题目无法写出来,因为你学习的是语法,但是你们缺少编程思维,让自己学会像计算机一样思考.

那是不是计算机太聪明了,我们太笨呢?

不是的,恰恰相反,是我们太聪明了,计算机太笨跟不上我们的思想.

变笨需要我们慢慢锻炼!

```
【程序 6 】  
/*  
初始 $money=100000  
判断条件 $money>=5000  
步长2种选择 if($i>50000) else if($i<=50000)  
*/  
  
for($i=1,$money=100000; $money>=5000; $i +=1) {  
    if($money>50000){  
        $money -= $money*0.05;  
    }else if($money<=50000) {  
        $money -=5000;  
    }  
    echo '第', $i, '次过路口,剩下', $money, '元<br />';  
}
```

3.6 九九乘法表 (21)

编程时,遇到不会的问题,要将它拆解,一步一步的来

你不要想一下子就能做出来

九九乘法表									
1x1=1									
2x1=2	2x2=4								
3x1=3	3x2=6	3x3=9							
4x1=4	4x2=8	4x3=12	4x4=16						
5x1=5	5x2=10	5x3=15	5x4=20	5x5=25					
6x1=6	6x2=12	6x3=18	6x4=24	6x5=30	6x6=36				
7x1=7	7x2=14	7x3=21	7x4=28	7x5=35	7x6=42	7x7=49			
8x1=8	8x2=16	8x3=24	8x4=32	8x5=40	8x6=48	8x7=56	8x8=64		
9x1=9	9x2=18	9x3=27	9x4=36	9x5=45	9x6=54	9x7=63	9x8=72	9x9=81	

1.先for循环1-9

```
//for循环打印1-9
for($i=1;$i<=9;$i++){
    echo $i,'<br />';
}
1
2
3
4
5
6
7
8
9
```

2.

```
for($i=1;$i<=9;$i++){
    echo $i,'~';
    for($j=1;$j<=$i;$j++) {
        echo $j;
    }
    echo '<br />';
}

1~1
2~12
3~123
4~1234
5~12345
6~123456
7~1234567
8~12345678
9~123456789
```

3.

```
for($i=1;$i<=9;$i++){
    for($j=1;$j<=$i;$j++) {
        echo $j,'*', $i,'=', $j*$i, '&nbsp;';
    }
    echo '<br >';
}

1*1=1
2*1=2 2*2=4
3*1=3 3*2=6 3*3=9
4*1=4 4*2=8 4*3=12 4*4=16
5*1=5 5*2=10 5*3=15 5*4=20 5*5=25
6*1=6 6*2=12 6*3=18 6*4=24 6*5=30 6*6=36
7*1=7 7*2=14 7*3=21 7*4=28 7*5=35 7*6=42 7*7=49
8*1=8 8*2=16 8*3=24 8*4=32 8*5=40 8*6=48 8*7=56 8*8=64
9*1=9 9*2=18 9*3=27 9*4=36 9*5=45 9*6=54 9*7=63 9*8=72 9*9=81
```

3.7 百钱买百鸡 (22)

1.

```
for($m=1;$m<=100;$m++){
```



```

        for($g=1;$g<=100;$g++){
            for($s=1;$s<=100;$s++){
                if(($m+$g+$s==100) && ($m*3+$g*5+$s/3)==100) {
                    echo '公鸡',$g,'只,母鸡',$m,'只,小鸡',$s,'只<br />';
                }
            }
        }
    }
}
公鸡12只,母鸡4只,小鸡84只
公鸡8只,母鸡11只,小鸡81只
公鸡4只,母鸡18只,小鸡78只

```

2.

```

for($m=1;$m<=100;$m++){
    for($g=1;$g<=100;$g++){
        $s=100-$m-$g;
        if(($m*3+$g*5+$s/3)==100) {
            echo '公鸡',$g,'只,母鸡',$m,'只,小鸡',$s,'只<br />';
        }
    }
}

公鸡12只,母鸡4只,小鸡84只
公鸡8只,母鸡11只,小鸡81只
公鸡4只,母鸡18只,小鸡78只

```

3.

```

for($m=1;$m<=31;$m++){
    for($g=1;$g<=18;$g++){
        $s=100-$m-$g;
        if(($m*3+$g*5+$s/3)==100) {
            echo '公鸡',$g,'只,母鸡',$m,'只,小鸡',$s,'只<br />';
        }
    }
}

```

3.8 实例练习(章节练习)

在学习的变量,运算,与控制结构后,请同学们认真完成以下题目

1)

【程序 1】
 题目：企业发放的奖金根据利润提成。利润(I) 低于或等于 10 万元时，奖金可提 10% ；
 利润高于 10 万元，
 低于 20 万元时，低于 10 万元的部分按 10% 提成，高于 10 万元的部分，可提成 7.5%；
 20 万到 40 万之间时，高于 20 万元的部分，可提成 5%；40 万到 60 万之间时高于 40 万元的部分，可提成 3%；60 万到 100 万之间时，高于 60 万元的部分，可提成 1.5%，高于 100 万元时，超过 100 万元的部分按 1%提成，从
 键盘输入当月利润 I，求应发放奖金总数？
 1. 程序分析：请利用 if else if 做范围的讨论

2)

【程序 2】
 输入三个整数 x,y,z ，找出最大的数

3)

【程序 3】
 题目：打印出所有的“水仙花数” ，所谓“水仙花数”是指一个三位数，其各位数字立方和等于该数本身。
 例如：153 是一个“水仙花数” ，因为 153=1 的三次方+5 的三次方+3 的三次方。
 1. 程序分析：利用 for 循环控制 100-999 个数，每个数分解出个位，十位，百位。

4)

【程序 4】

题目：猴子吃桃问题：猴子第一天摘下若干个桃子，当即吃了一半，还不瘾，又多吃了一个，第二天早上又将剩下的桃子吃掉一半，又多吃了一个。以后每天早上都吃了前一天剩下的一半零一个。到第 10 天早上想再吃时，见只剩下一个桃子了。求第一天共摘了多少。

1. 程序分析：采取逆向思维的方法，从后往前推断

5)

【程序 5】

题目：有一分数序列：2/1，3/2，5/3，8/5，13/8，21/13...求出这个数列的前 20 项之和。

程序分析：请抓住分子与分母的变化规律。

6)

【程序 6】

假设某人有 100,000 现金。每经过一次路口需要进行一次交费。交费规则为当他现金大于 50,000 时每次需要交 5%如果现金小于等于 50,000 时每次交 5,000。请写一程序计算此人可以经过多少次这个路口。

7)

【程序 7】

打印九九乘法表

8)

【程序 8】

《张丘建算经》成书于公元 5 世纪，作者是北魏人。书中最后一道题堪称亮点，通常也被称为“百钱买百鸡”问题，民间则流传着县令考问神童的佳话书中原文如下：
今有鸡翁一，值钱五；鸡母一，值钱三；鸡雏三，值钱一；百钱买鸡百只，问鸡翁、母、雏各几何？
题目的意思是，公鸡 5 文钱 1 只，母鸡 3 文钱 1 只，小鸡 1 文钱买 3 只，现在用 100 文钱共买了 100 只鸡，问：在这 100 只鸡中，公鸡、母鸡和小鸡各是多少只？（设每种至少一只）

9)

【程序 9】

求 1 到 100 内的素(质)数。

注:素数只能被 1 和其自身整除的数。如 2,3,5,7,11

10)

【程序 10,稍难一些,学数组后再做】

一种羊 生命长度为 5 周年

满 2 周年生一只小羊 满四周年生一只小羊，5 周年死

初始有 1 只 0 岁的羊,20 年后有多少羊。

第四章 函数

4.1 函数概念 (23)

函数就是一段封装起来的代码,可以随时调用.

1.如果有三个人来北京

```
echo 'welcome ';  
echo 'to ';  
echo 'Beijing!<br >';
```

```
echo 'welcome ';\necho 'to ';\necho 'Beijing!<br >';\n\necho 'welcome ';\necho 'to ';\necho 'Beijing!<br >';
```

我们需要重复三次,那再来一个人,能否用简单的办法招呼人
函数 -> 几行函数封装起来,再起个名字

```
function wel(){//复合语句\n    echo 'welcome ';\n    echo 'to ';\n    echo 'Beijing!<br >';\n}\nwel();\nwel();\nwel();
```

2.函数可以是一个计算器(通过调用函数,算出两个数的和)

```
function add($a,$b) {\n    echo $a+$b;\n}\nadd(2,3);
```

3.函数可以把运算结果返回给调用者

```
function add($a,$b) {\n    return $a+$b;\n}\necho add(3,5);
```

此计算器的运算过程如下

```
function add($a,$b) {\n    //赋值过程,虽然人眼看不到,但是内部确实如此\n    $a = 3;\n    $b = 5;\n    return $a+$b;\n}\necho add(3,5);
```

函数,如果没有任何参数

它就是纯粹的一段封装代码,外界调用这个函数的时候,就会将此函数中的代码执行一下.

如果有参数,那就相当于一个加工机器

将外部传来的参数,就是加工机的原料,将参数带到函数内部进行加工.再返回一个结果出来;

4.2 函数定义格式 (24)

定义格式是死记硬背的,没有什么技巧

注意:

无论函数有没有参数[可以没有参数],都需要在函数名后面加上小括号

```
function 函数名([参数1],[参数2],...[参数n]) {\n    //函数体,就是php语句\n    return 某值/表达式//\n}
```

return的作用:

函数就好比一个豆浆机器,参数就像黄豆和水

我们将黄豆和水放进去, 如果没有产出豆浆,那就没有任何作用;

return返回一个结果

谁能捕捉到这个调用结果呢?调用结束后,调用行可以捕捉return的结果

函数的命名规范:

对于函数,命名规则和变量是一样的,但是函数不区分大小写.

注意:

虽然不区分大小写,但是声明时和调用时大小写要保持一致

```
function jia() {  
    return $a = 3;  
}  
echo JIA();
```

4.3 函数执行与返回流程 (25)

1.函数是机器,调用才能执行

```
function say() {  
    echo 'hello<br />';  
}  
  
function he($a,$b) {  
    $sum = $a + $b;  
    $a +=1;  
    $b -=1;  
    return $sum;  
}
```

我们的php在运行,是/wamp/php/php.exe[执行引擎]在解释和执行的

执行权:

执行权进入函数,执行函数体,函数结束后,交回执行权

```
echo 'good good study<br >';  
say(); //执行权进入到say函数,函数结束后,交回执行权  
echo 'day day up';
```

什么时候交回执行权?函数结束后

在return后加入语句

```
function he($a,$b) {  
    $sum = $a + $b;  
    $a +=1;  
    $b -=1;  
    return $sum;  
    echo '~~~~~!';  
}  
  
echo he(3,2);
```

碰到return语句,也要交回执行权

所以在return后再写语句是不执行的,因为执行权已经交回

4.4 函数传参方式 (26)

传参的过程发生了什么?

函数是一台加工机器,我们应该投入原料,返回结果,中间过程尽量不要输出

```
function test($a) {  
    $a +=1;  
    return $a;  
}  
  
$b = 4;  
echo test($b);//5
```

分析以上代码执行的过程:

```
/*function test($a) { //函数中的参数传的是什么值
```

```

$a = $b; //这里是将$b的值读出来赋给了$a(变量可以给它赋值也可以读它的值)
$a +=1;
return $a;
}*/
echo test(5); //这里是直接给定函数的参数一个直接的值
echo $b; //4, 读出来的值赋给$a, $a改变并不会影响到$b

```

变量传值的两种方式

(传值赋值和引用赋值)

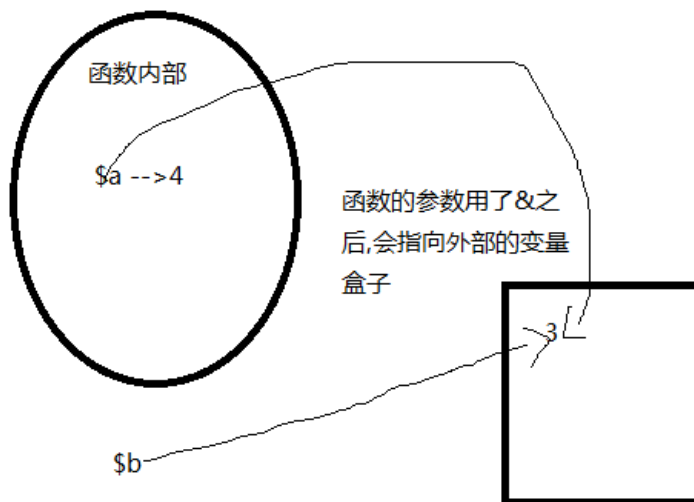
此处可以使用引用传参:

```

function test(&$a) {
    $a +=1;
    return $a;
}

$b = 4;
echo test($b); //5
echo $b; //5

```



这样传值不推荐,因为一个函数在内部运行的时候,影响到了它外部的一个环境.

它破坏了函数的封装性,不推荐使用

注意: 函数应该是一个加工机器,它是一个独立的黑盒子

它应该只:接受参数/处理/返回值;

处理的过程中,它不应该对外界造成影响

4.5 可选参数 (27)

1.相乘加倍函数

```

function mult($num,$rate) {
    return $num*$rate;
}
echo mult(4);
echo mult(4,5);

```

2.如果两个参数,我只给定一个,另一个参数不指定,但是默认有一定的参数 -> 可选参数

```

function mult($num,$rat=5) {
    return $num*$rat;
}
echo mult(4);

```

1)echo mult(4);

过程:

```

function mult($num,$rat=5) {
    $num = 4;

```

```

    $rat = 5;
    return $num*$rat;
}
echo mult(4);

```

2)echo mult(4,6);

过程:

```

function mult($num,$rat=5) {
    $num = 4;
    $rat = 5;
    $rat = 6;
    return $num*$rat;
}
echo mult(4,6);

```

那可以第一参数有默认值,第二个没有吗

```

function mult($num=5,$rate) {
    return $num*$rate;
}
echo mult(4); //参数报错
echo mult(,5); //语法错误
echo mult(4,5); //默认值没有意义

```

这样调用是没什么意义的,调用时只给一个值默认是第一个,会覆盖参数的第一个默认值,导致函数的第二个参数没有值;
如果调用时,给定两个参数,函数参数的默认值就没什么意义了

注意:

如果有默认参数,应写在最后

4.6 函数作用域 (28)

变量的作用域,就是变量在函数里,和不在函数里

对于php而言,函数的作用域非常简单,它就区分函数内和函数外

```

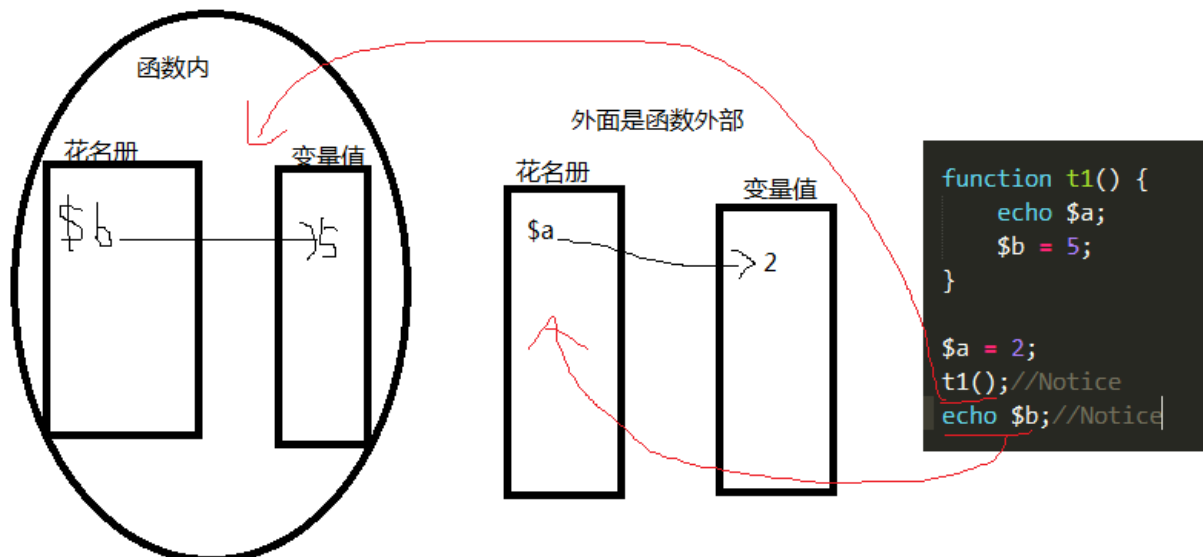
function t1() {
    echo $a;
    $b = 5;
}

$a = 2;
t1();//Notice
echo $b;//Notice

```

函数内部和函数外部分别有两个花名册

函数内部的变量有它自己的花名册



函数内部的变量和函数外部的变量互不干扰

函数内变量成为"局部变量";

在php页面中声明的,且在函数外部变量成为"全局变量";

不推荐:

如果想在函数内部操作全局变量也是可以的

1.可以用&符号传参(之前课有讲)

2.global

```
function t1() {  
    global $a; //这句话是声明,$a这个变量就去全局的花名册中找  
    $a +=1;  
    echo $a ;  
}  
  
$a = 2;  
echo $a, '<br >';  
t1();  
echo ' <br >';  
echo $a, '<br >';
```

3.\$GLOBALS(数组)[不推荐使用]

可以将\$GLOBALS看成外部变量的花名册

```
$b = 5;  
$c = 'hello';  
print_r($GLOBALS);  
  
function t2() {  
    $GLOBALS['c'] = 'word';  
    $GLOBALS['d'] = 'welcome';  
}  
  
t2();  
echo $c;  
echo $d;
```

\$GLOBALS是系统给定的一个超级全局变量

在页面的任何部分,包括函数 ,方法等 ,都可以直接访问

(一共超全局变量有九个,后面我们会学到)

注意: 外部和函数内如果想发生关系,应该使用传参的方式

4.7 动态调用函数 (29)

php非常的灵活,我们在学习变量的时候,知道,可以拿一个变量的值当另一个变量的名

变量值也可以当函数名来调用

```
function good() {  
    echo 'hi';  
}  
  
function bad() {  
    echo 'gun';  
}  
  
$heart = 'good';  
good(); //hi  
$heart(); //hi 这是一个函数的调用,然而并没有使用函数名来调用  
  
$heart = 'bad';  
bad(); //gun  
$heart(); //gun
```

这是php比较灵活的一点,在其他语言中,如果想实现相同的效果,相对来说会比较麻烦.

4.8 函数相关作业

1:函数内部如何使用全局变量?(提示:上课时让思考的 global 和\$GLOBALS)

2:练习使用超全局变量,能打印\$_GET,\$_POST 的值,并会取来访者的 IP

3:如何检测一个函数是否已定义(课上未讲,请同学们找手册或百度)

4:考一道面试题(考点为\$GLOBALS 数组)

写出如下程序的输出结果

```
<?php
$GLOBALS['var1'] = 5;
$var2 = 1;
function get_value(){
    global $var2;
    $var1 = 0;
    return $var2++;
}
get_value();
echo $var1."\n";
echo $var2;
?>
```

4.9 时间戳函数 (30)

php在web开发领域占据很大的比重,能达到80%左右,为什么?

它发展的这些年来,是专门为web开发所做的

它不像java可以做桌面,移动和网站等.

它全部的心思都放在web上,那么它积累了大量的函数,有4,5千个之多.

对于做网站而言都非常之常用,很方便和实用.

对于我们开发网站而言,我们能想到的字符串操作,数组操作等等,它基本上都有系统函数.

如果我们对这些函数不熟悉,想对字符串拆分,反转等费了很大力气去实现,

但这都是有现成的函数可以实现的,所以对于php的常用系统函数,我们一起要掌握熟练.可以让我们的开发事半功倍.

时间戳函数:

time() 返回自从 Unix 纪元（格林威治时间 1970 年 1 月 1 日 00:00:00）到当前时间的秒数。

microtime() 返回当前 Unix 时间戳和微秒数

例子:

用户注册的时候我们需要记录用户的注册时间,如果不用时间戳记录,像这样

2015年7月28日 12:45:29;

2015年3月1日 16:20:33;

帮我查出24小时之内注册的用户,那么往前推一天,3.1日往前推是2.28还是2.29呢?

当前的时间戳-24小时的秒数(246060) = 一天前的时间戳

获取当前时间戳的函数

看手册分析time函数,手册详解分析

time 函数名

(PHP 4, PHP 5) 适用版本

time — 返回当前的 Unix 时间戳

函数功能

说明

函数返回值的类型

int time (void)

函数的参数void表示没有参数

返回自从 Unix 纪元（格林威治时间 1970 年 1 月 1 日 00:00:00）到当前时间的秒数。

Tip

自 PHP 5.1 起在 `$_SERVER['REQUEST_TIME']` 中保存了发起该请求时刻的时间戳。

Example #1 time() 例子

```
<?php
$nextWeek = time() + (7 * 24 * 60 * 60);
                // 7 days; 24 hours; 60 mins; 60secs
echo 'Now:      '. date('Y-m-d') . "\n";
echo 'Next Week: '. date('Y-m-d', $nextWeek) . "\n";
?>
```

使用案例,及一些巧妙的
用法

```
echo time(), '<br >';
```

在后面我们记录文章发布的时间戳

```
$pubtime = time();
```

mixed microtime ([bool \$get_as_float])

手册看它的参数

mixed是混合的意思,代表返回的类型不止一种

精确到微妙的函数

```
echo microtime(), '<br >';
```

利用microtime可以计算脚本的运行时间,

在脚本执行开始处和结尾处都给定一个时间戳

百钱买百鸡计算运行时长

```
$start = microtime(true);
for($m=1;$m<=100;$m++){
    for($g=1;$g<=100;$g++){
        for($s=1;$s<=100;$s++){
            if(($m+$g+$s==100) && ($m*3+$g*5+$s/3)==100) {
                echo '公鸡', $g, '只, 母鸡', $m, '只, 小鸡', $s, '只<br />';
            }
        }
    }
}
$end = microtime(true);
echo $end-$start;
```

4.10 时间戳格式化 (31)

date()

注意:

如果我们格式化后时间跟现在有时差

找到php.ini

将时区更改为以下即可

```
date.timezone = PRC
```

date — 格式化一个本地时间 / 日期

时间戳好记录,但是对人眼直观上没有什么意义,不够直观

我们在前台显示出来需要将记录下来的时间戳格式化之后再显示.

1)前一天的时间

```
$lastday = time()-24*3600;  
echo date('Y年m月d日 H:i:s' , $lastday),'<br >';
```

2)不写第二个参数,默认是time();当前时间戳

```
echo date('Y年m月d日 H:i:s'),'<br >';  
  
echo date('Y-m/d H:i:s' , $lastday),'<br >';
```

3)国外写法,将年份放在后面也是可以的

```
echo date('m/d Y H:i:s' , $lastday),'<br >';
```

4.11 日期解析函数 (32)

给定一个具体日期时间,将这个日期转换为当前时间戳

mktime() — 取得一个日期的 Unix 时间戳

strtotime() — 将任何英文文本的日期时间描述解析为 Unix 时间戳

checkdate() — 验证日期是否合法

1.mktime()

```
mktime();给定一个日期,返回这个日期的时间戳  
echo mktime(18,30,16,8,22,1992);
```

2.strtotime()

```
strtotime();可以口语化的转化时间戳(有很多,不必全记住)  
echo strtotime("now");  
echo strtotime("-1 day");  
echo strtotime("+1 week");
```

3.checkdate()

```
checkdate() 负责检测日期是否合法  
var_dump(checkdate(2,30,2001));  
var_dump(checkdate(2,29,1996));  
var_dump(checkdate(7,30,2015));
```

4.12 日期时间相关函数

日期时间函数相关面试题:

<http://www.zixue.it/thread-131-1-1.html>

<http://www.zixue.it/thread-102-1-1.html>

第五章 字符串

5.1 字符串定义方式 (33)

字符串的操作在php中十分丰富

1.如何定义字符串:

单引号,双引号

```
$str1 = 'hello';
```

```
$str2 = "word";  
echo $str1, ' ', $str2;
```

2.如何定义大段的字符串

有两种方式,heredoc和nowdoc

1)heredoc 对应大段的""(双引号)字符串

要求非常的严格

开头前后不能有空格

中间不能插注释,前后保持一致,大小写均可

```
$str = <<<here  
第一段  
第二段  
第三段  
here;  
  
echo $str;
```

2)nowdoc 对应大段的"(单引号)字符串

```
$str2 = <<<'NOW'  
第一段  
第二段  
第三段  
NOW;  
  
echo $str2;
```

5.2 转义,变量解析,与速度的对比 (34)

探讨单双引号字符串的区别,他们的区别主要体现在:转义,变量解析和速度

1.转义

```
$str1 = "\" \\ \n \t \v \"$";  
$str2 = '\" \\ \n \t \v \' '$';  
  
echo $str1, '<br >', $str2;
```

注意:

双引号转义的较多,单引号只能转义两个,\和\\,其他的都不转义

2.对变量的解析

```
$name = "lili";  
$age = '23';  
$intro = "my name is $name,and i am $age";  
$intro1 = 'my name is $name,and i am $age';  
echo $intro, '<br >', $intro1;
```

注意:

双引号可以解析变量的值,单引号不解析

一般支付功能都会有个密钥,而密钥一般是自己随机敲的字符串

```
// $key = "adlkfjiu$adkjf232";  
$key = 'adlkfjiu$adkjf232';  
echo $key;
```

注意:

当我们用双引号包住字符串的时候,如果不是变量,不能出现\$后面不能跟合法的变量名;

所以一般情况下,我们不需要解析字符串中的变量的时候,有先用单引号包住字符串就不会这样的问题了;

3.速度

因为单引号不必考虑过多的转义,以及变量解析,所以速度上比双引号酷快;

字符串,数组的键\$arr['name'],优先使用单引号;

5.3 字符串常用函数 (35)

字符串对于做web开发而言,非常之常用;(每天都离不开它,计算它的长度,切割,反转等)

字符串函数非常丰富且强大,一定要认真练习!

【字符串长度函数】

int strlen(\$str) 计算字符串长度

int mb_strlen (string \$str [, string \$encoding])

【查找字符串位置函数】

strpos(\$str,search,[int]):查找 search 在\$str 中的第一次位置从 int 开始;

stripos(\$str,search,[int]):函数返回字符串在另一个字符串中第一次出现的位置。该函数对大小写不敏感

strrpos(\$str,search,[int]):查找 search 在\$str 中的最后一次出现的位置从 int

【字符串替换函数】

str_replace(search,replace,\$str):从\$str 中查找 search 用 replace 来替换

str_ireplace(search,replace,\$str):

strtr(\$str,search,replace):这个函数中 replace 不能为"";

substr_replace(\$str,\$rep,\$start[,length])\$str 原始字符串,\$rep 替换后的新

字符串,\$start 起始位置,\$length 替换的长度,该项可选

【截取子字符串函数】

substr(\$str,int start[,int length]):从\$str中start位置开始提取[length 长度的字符串]。

strstr(\$str1,\$str2): 从\$str1(第一个的位置)搜索\$str2 并从它开始截取到结束字符串;若没有则返回 FALSE。

stristr() 功能同 strstr, 只是不区分大小写。

strrchr() 从最后一次搜索到的字符处返回;

【分割,连接,反转函数】

str_split(\$str,len):把\$str 按 len 长度进行分割返回数组

explode(search,\$str[,int])

implode —— 将数组用特定的分割符转变为字符串

【空白处理函数】

string trim (string \$str [, string \$charlist])

string ltrim (string \$str [, string \$charlist])

string rtrim (string \$str [, string \$charlist])

chunk_split(\$str,2);向\$str 字符里面按 2 个字符就加入一个空格;

str_pad —— 对字符串进行两侧的补白

【字符转义函数】

addslashes (string \$str)

stripslashes (string \$str)

get_magic_quotes_gpc()

htmlspecialchars —— 将字符串中一些字符转换为 HTML 实体

htmlspecialchars_decode —— htmlspecialchars()函数的反函数, 将 HTML 实体转换为字符

html_entity_decode —— htmlentities ()函数的反函数, 将 HTML 实体转换为字符

htmlentities —— 将字符串中所有可转换字符转换为 HTML 实体

【字符串比较函数】

int strcmp(\$str1,\$str2): (字符串比较)

strcasecmp() 同上 (不分大小写)

计算字符串长度

strlen

mb_strlen

strlen — 获取字符串长度 (计算的是字节数)

int strlen (string \$string)

```
$str1 = 'abc d';
$str2 = '中国人';
echo strlen($str1);
echo strlen($str2);
```

用记事本分别保存:中国,一个gbk,一个utf8,分别观察占几个字节

有些文档了为了区分编码,到底按照哪种编码解析它

计算机都是存的0,1

有些文档就会在utf8编码的文档中加上3个人眼无法看到的字节

这个字节称为BOM头,正好占3个字节

如果我们用记事本建utf8的文档,就会带来这个问题;

为什么去掉BOM头?

学习session和cookie的时候,有BOM头会报错

因为在session或者cookie启用前有了输出

如何去掉BOM头?

用editplus[右下角编码带utf8+ 表示有BOM]和sublime(默认无BOM),都可以去掉BOM头]

gbk一个中文占2个字节,utf8一个中文占3个字节

mb_strlen — 获取字符串的长度 (计算的是字符数)

mb 宽字节--国际编码支持(英文,法文中文等都支持)

注意,第二参数要声明字符编码

```
$str2 = '中国人';
echo mb_strlen($str2,'utf-8');
```

字符串位置

strpos— 查找字符串首次出现的位置

查找一个子串在一个大串中首次出现的位置,计算机中,位置从0记数

```
$str = 'abcde';
echo strpos($str,'c');
```

注意:

此函数可能返回布尔值 FALSE,但也可能返回等同于 FALSE 的非布尔值,

因为字符串位置是从0开始,而不是从1开始的。

```
$str = 'abcde';
if(strpos($str,'a') == false) {
    echo '不存在';
} else {
    echo '存在';
}

if(strpos($str,$str2) === false) {
    echo '不存在';
} else {
    echo '存在';
}
```

替换字符串

str_replace

strtr

str_replace — 子字符串替换

```
$reply = 'fuck you';
echo str_replace('fuck', 'f**k', $reply);
```

网站公开发表或者评论,应该转换为'文明用语'是很必要的.涉及非法发表言论是不允许的.

strtr — 转换指定字符

替换一批字符串,批量替换

第二个参数为数组,键是待替换的,值是替换后的

1)第一种写法

```
$str = '男人,女人,男孩,女孩';  
echo strtr($str,array('男'=>'女','女'=>'男'));  
//这个是将字符串看成一个整体,不会出现乱码
```

2)第二种写法

```
$str = '男人,女人,男孩,女孩';  
echo strtr($str,'男','屁'), '<br >';  
echo strtr($str,'人','屁');  
//此行是按字节
```

替换,中文易乱码,推荐上一种用法

截取子字符串

substr

substr — 返回字符串的子串

substr('大字符串','从哪个位置开始截取',[截取几个])

从0开始截取

(第三个参数不写默认截取到大串最后)

1)

```
$str = 'helloworld';  
echo substr($str,5),'<br >';  
echo substr($str,0,5),'<br >';  
echo substr($str,3,5);
```

2)始终保持截取字符串后三位,倒着来,用负数

```
echo substr($str,-3),'<br >';  
echo substr($str,-3,1);
```

3)第3个 为整数: 代表截取的长度

如果为负 代表结束位置,从后往前数.

```
echo substr($str,-5,-2),'<br >';
```

拆分字符串

explode

explode — 使用一个字符串分割另一个字符串,返回一个数组

拆分字符串,按某个指定的标志,把字符串拆的数组

1)发表博文的时候会发布标签

```
$str='php,mysql,study';
```

在写入数据库的时候我们需要将这几个标签拆成独立的标签

```
$str='php,mysql,study';  
print_r(explode(',',$str));
```

拼接字符串

implode

implode — 将一个一维数组的值转化为字符串

1)将一个数组的值,拼接为字符串

```
$arr = array('name'=>'zhangsan','age'=>23,'gender'=>'man');  
echo implode('/', $arr);
```

分析文件后缀名
strpos 字符串位置
substr 截取字符串

```
$file = 'a.jpg';  
echo substr($file, strpos($file, '.'), '<br >';  
echo substr($file, strpos($file, '.')+1);
```

5.5 字符串相关面试题

<http://www.zixue.it/forum.php?mod=viewthread&tid=249>
<http://www.zixue.it/forum.php?mod=viewthread&tid=134>
<http://www.zixue.it/forum.php?mod=viewthread&tid=121>
<http://www.zixue.it/forum.php?mod=viewthread&tid=189>
<http://www.zixue.it/forum.php?mod=viewthread&tid=137>
<http://www.zixue.it/forum.php?mod=viewthread&tid=149>
<http://www.zixue.it/forum.php?mod=viewthread&tid=151>
<http://www.zixue.it/forum.php?mod=viewthread&tid=208>

第六章 综合案例之文件管理系统 (36)

1. 实现的文件管理样式图

文件管理系统

名称	操作
a. txt	查看

```
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <title>Document</title>  
</head>  
<body>  
    <h1>文件管理系统</h1>  
    <table border="1">  
        <tr>  
            <td>名称</td>  
            <td>操作</td>  
        </tr>  
        <tr>  
            <td>a. txt</td>  
            <td>查看</td>  
        </tr>  
    </table>  
</body>  
</html>
```

2. 展示当前目录[当前目录是指这个php所在的目录]

打开目录 -> 去读条目 -> 关闭目录

手册 -> fopen -> directory -> opendir

打开[资源]->读取->关闭
opendir - 打开目录句柄
readdir - 从目录句柄中读取条目
closedir - 关闭目录句柄

打开当前目录,并将当前目录的文件读取出来

```
<?php
$path = '.';
$fh = opendir($path);
echo readdir($fh), '<br >';
echo readdir($fh), '<br >';
echo readdir($fh), '<br >';
echo readdir($fh), '<br >';
?>
```

为什么会有.和..

.和..为虚拟目录,代表当前目录和上级目录

用cmd窗口(直接dir查看即可)

3.while一直读取出来

1)错误1循环

```
//错误写法
/*
while( readdir($fh) !== false ) {
    echo readdir($fh), '<br >';
}
*/
```

2)错误2循环

当有个文件为0,后面的文件显示不出来

```
while( $row = readdir($fh) ) {
    echo $row, '<br >';
}
```

3)正确的写法

```
$path = '.';
$fh = opendir($path);

while( ($row = readdir($fh) ) !== false ) {
    echo $row, '<br >';
}

closedir($fh);
```

在php页面中是可以任意嵌套html的

只要不包含在<?php?>中,都理解为html标签

```
<meta charset="utf-8">
<?php
$path = '.';
$fh = opendir($path);
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
</head>
<body>
    <h1>文件管理系统</h1>
    <table border="1">
        <tr>
            <td>名称</td>
            <td>操作</td>
        </tr>
        <?php
        while( ($row = readdir($fh) ) !== false ) {
            echo '<tr>';
            echo '<td>'.$row.'</td>';
            echo '<td>查看</td>';
```



```

        echo '</tr>';
    }?>
</table>
</body>
</html>

```

文件管理系统

名称	操作
.	查看
..	查看
01.txt	查看
02.txt	查看
03.txt	查看
1.php	查看
10.html	查看
10.php	查看
12.php	查看
13-1.php	查看
14.php	查看
2.php	查看
20.php	查看
21.php	查看

3.如果当前目录,是目录能点进去去看目录下的内容

```

<meta charset="utf-8">
<?php
$path = isset($_GET['dir'])? $_GET['dir'] : '.';
$fh = opendir($path);

?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
</head>
<body>
    <h1>文件管理系统</h1>
    <table border="1">
        <tr>
            <td>名称</td>
            <td>操作</td>
        </tr>
        <?php
        while( ($row = readdir($fh) ) !== false) {
            echo '<tr>';
            echo '<td>'.$row.'</td>';
            echo '<td><a href="9.php?dir=' , $path , '/' , $row , '>查看' , '</a></td>';
            echo '</tr>';
        }
        closedir($fh);
    ?>
    </table>
</body>
</html>

```

改进:

改进1: 对于普通文件,直接打开看内容 is_dir is_file

改进2: 防范地址栏恶意打 ../../.. 查看上级甚至磁盘内容 realpath 返回一个绝对路径

以及字符串相关处理函数,判断路径是否很短,短到www了

改进3: 加个input表单,通过[submit],提交,即可创建一个目录(例:mp3目录),is_dir mkdir,先判断目录是否存在

第七章 数组

7.1 数组定义方式 (37)

数组是php中,非常常用非常重要的一个数据结构,

php本身的数组非常的强大

如何来定义一个数组?

数组是一种复合数据,可以装下多个值,每个值用不同的键来区分.(键值对应)

键 -> 箱子上的编号

值 -> 箱子里面的内容

```
$arr = array('name'=>'zhangsan','age'=>19);  
print_r($arr);
```

7.2 数组类型 (38)

数组的分类:

索引数组; 关联数组; 二维数组; 多维数组

不需要刻意记录,它的循环都是用的 foreach()

性质不会因为类型变化而变化

1.索引数组

键不必有特殊意义,纯数字,从0递增的数组,这种叫"索引数组"

像c语言,js中键的规律就是这样的

```
$arr = array(0=>'a',1=>'b',2=>'c',3=>'d');  
print_r($arr);
```

2.关联数组

字符串做键,一般能体现该单元的内容

如 age->年龄, '关联数组'

//在其他语言中,类似这样的数据结构不叫数组

c语言中:结构体;java/python中:dict[字典],js中:对象

```
$arr = array('name'=>'lisi','age'=>'23');  
print_r($arr);
```

数组的单元,可以储存哪几种类型的值?

php中的8种类型都可以

那意味着,我在数组中的值中再存放一个数组也是可以的

3.二维数组 多维数组

打开一个箱子,发现里面又是一组箱子

\$arr2为二维数组,如果层次更深,那就是多维数组

```
$arr2 = array('name'=>'lili','hobby'=>array(0=>'basketball',1=>'football',2=>'pingpang'));  
print_r($arr);
```

7.3 数组键规则 (39)

数组的键不写可以吗?

如果不写它的键是以什么样的规则自动生成的

1.键可以不分配键,系统会0,1,2...递增分配

```
$arr = array('a','b','c');  
print_r($arr);
```

2.如果有的键分配,有的不分配

会从之前的键中,取最大的键开始递增

```
$arr1 = array('name'=>'lisi','age'=>23,'running');
print_r($arr1);

$arr2 = array(0=>'a',1=>'b','c',5=>'d','e','f');
print_r($arr2);
```

3.如果键分配重复了怎么办

对于数组它的键是不能重复的

如果键重复,后面的同名键会覆盖前面的

```
$arr3 = array('a','b','c',2=>'d',2=>'e');
print_r($arr3);
```

注意:

数组的键只有两种类型,整型和字符串

键为浮点型,字符串型的整数,和null

键为浮点数->向下取整

如果字符串恰好理解为一个整数,也转为整数

null按空字符串来理解

```
$arr = array(2=>'布',2.5=>'尔','2.5'=>'教','2'=>'育',null=>'好');
print_r($arr);
```

在实际开发中:

不会遇到这些比较麻烦的数组

一般是关联数组和自动生成的索引数组

7.4 操作数组单元 (40)

数组是一个复合数据,里面放了很多数据

print_r()将里面所有的数据都打印了出来

需要对数组的具体单元进行操作

数组单元的 增,删,改,查

1.单独取出某一个单元的值,用键来取 \$数组名[键]

```
$arr = array('a','b','c');
echo $arr[1], '<br >';
```

2.取出二维数组的值

数组的层级顶多到3层,否则人无法直观的理解

```
$arr2 = array('name'=>'lisi','hobby'=>array('basketball','football','pingpang'));
echo $arr2['hobby'][0];
```

3.更改一个数组单元的值

```
$arr2['hobby'][0] = 'swiming';
print_r($arr2);
```

4.增加一个数组单元

```
$arr2['area'] = 'beijing';
print_r($arr2);

$arr2['area'] = 'shanghai';
print_r($arr2);
```

5.删除数组单元

```
unset($arr2['hobby']);
print_r($arr2);
```

7.5 遍历数组 (41)

1. 取出数组的每个单元

```
$arr = array('a','b','c','d','e');  
echo $arr[0], '<br >';  
echo $arr[1], '<br >';  
echo $arr[2], '<br >';  
echo $arr[3], '<br >';  
echo $arr[4], '<br >';
```

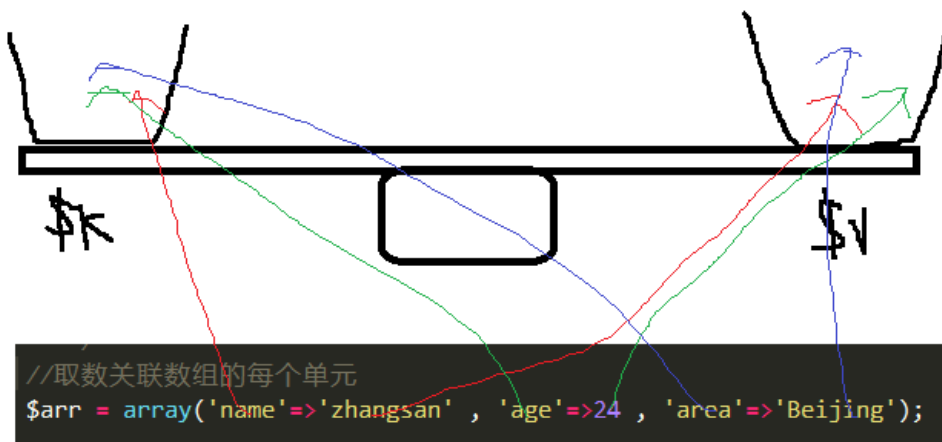
索引数组可以这样

```
for($i=0;$i<count($arr);$i++) {  
    echo $arr[$i], '<br >';  
}
```

2. 关联数组如何取出每个数组单元?

它的键是字符串,没有任何规律

托盘称图讲解foreach循环遍历数组



foreach是专门用来循环数组的,速度非常快

foreach 里面的键值的变量名(\$k,\$v) 为任意合法的变量名

```
$arr = array('name'=>'zhangsan' , 'age'=>24 , 'area'=>'Beijing');  
  
foreach($arr as $k=>$v) {  
    echo $k, ': ', $v, '<br >';  
}
```

3. 只循环值

```
foreach($arr as $v) {  
    echo $v, '<br >';  
}
```

只循环键

```
foreach($arr as $k) { //错误的  
    echo $v, '<br >';  
}  
  
foreach($arr as $k=>){ //语法错误  
}
```

4. array_keys — 返回数组中所有的键名

foreach 没有办法单循环取出键

5. 把下面这个的数组的每个单元值变成原来的 2 倍

```
//
```

```
$stu = array('lisi'=>3, 'wang'=>5, 'zhao'=>6);

$stu = array('lisi'=>3, 'wang'=>5, 'zhao'=>6);
print_r($stu);

foreach(){}

print_r($stu); //array('lisi'=>6, 'wang'=>10, 'zhao'=>12);
```

10分钟时间完成这道题

1)常见错误1:

```
foreach($stu as $k=>$v){
    echo $v*2;
}
print_r($stu);
```

2)常见错误2:

```
foreach($stu as $k=>$v){
    $v = $v*2;
}
print_r($stu);
```

3)正确:

```
foreach($stu as $k=>$v) {
    $stu[$k] = $v*2;
}
print_r($stu);
```

7.6 数组游标操作 (42)

数组的游标操作并不是经常使用

但是有利于我们理解数组是怎样循环的

数组内部有一个这样的游标,会指向当前数组的单元

```
current - 返回数组中的当前单元
next - 将数组中的内部指针向前移动一位
prev - 将数组的内部指针倒回一位
end - 将数组的内部指针指向最后一个单元
reset - 将数组的内部指针指向第一个单元(重置内部游标)
each - 返回数组中当前的键 / 值对并将数组指针向前移动一步
list - 把数组中的值赋给一些变量
```

1.刚创建数组,它的游标放在数组的0号单元

获取当前数字的游标所指向的单元的值

```
$arr = array('a', 'b', 'c', 'd');
echo current($arr), '<br >'; //a
```

2.游标向前移动一位

```
next($arr);
echo current($arr), '<br >'; //b

next($arr);
echo current($arr), '<br >'; //c
```

3.游标向头部(向左)移动一位

```
prev($arr);
echo current($arr); //b
```

4.游标指向尾部

```
end($arr);
echo current($arr); //d
```

5.游标指向头部(重置游标)

```
reset($arr);
echo current($arr); //a
```

6.用游标的知识,取出关联数组的值

```
$arr = array('name'=>'lili','age'=>23,'area'=>'Bj');
//不够强大和完善
while($row = current($arr)) {
    echo $row,'<br >';
    next($arr);
}
```

以下关联数组无法获取全部值(value为false)

```
$arr = array('name'=>'lili','age'=>false,'area'=>'Bj');
while($row = current($arr)) {
    echo $row,'<br >';
    next($arr);
}
```

7.each();遍历数组

可以读取当前指针指向的单元的 键/值,且以数组形式返回
且移动指针

```
$arr = array('name'=>'lili','age'=>false,'area'=>'Bj');
print_r(each($arr));
print_r(each($arr));

$arr = array('name'=>'lili','age'=>false,'area'=>'Bj');
while( $row=each($arr) ) {
    print_r($row);
}
```

8.list();同时给多个变量赋值

将数组的0号单元赋值给左侧的\$a,1号单元赋值给左侧的\$b...依次类推

```
$arr = array('a','b','c');
list($a,$b,$c) = $arr;
echo $a,$b,$c;
```

9.each();加list()获取数组的键和值

```
$arr = array('name'=>'lili','age'=>false,'area'=>'Bj');
while( $row=each($arr)) {
    list($k,$v) = $row;
    echo $k.':',$v,'<br >';
}
```

7.7 数组常用函数 (43)

```
count()
array_key_exists()
in_array()
array_change_key_case();
array_count_values();
array_fill();
array_filter();
array_values()
array_keys()
array_push()
array_pop()
```

```
array_shift()
array_unshift()
Sort(),rsort(),Usort(),asort(),
ksort(),natsort(),natcasesort()
array_merge()
array_merge_recursive()
array_diff()
array_diff_assoc()
array_intersect()
array_intersect_assoc()
array_flip();
array_unique()
array_reverse()
array_sum();
shuffle()
range()
array_rand()
```

1.count — 计算数组中的单元数目或对象中的属性个数

```
echo count(array('a','b','c'));//3
```

1)如果不是数组则返回1

```
echo count('helloworld');//1
```

2)如果是null则返回0

```
echo count(null);//0
```

2.array_key_exists — 检查给定的键名或索引是否存在于数组中

1)判断数组中的键名是否存在(isset)

```
$arr = array('a'=>'青龙','b'=>'白虎','c'=>null);
if(isset($arr['b'])) {
    echo 'zai';
} else {
    echo 'bzai';
}

//isset碰到null返回false
if(isset($arr['c'])) {
    echo 'zai';
} else {
    echo 'bzai';
}
}
```

2)用array_key_exists

```
$arr = array('a'=>'青龙','b'=>'白虎','c'=>null);
if(array_key_exists('c', $arr)) {
    echo 'zai';
} else {
    echo 'bzai';
}
}
```

3.in_array — 检查数组中是否存在某个值

```
$arr = array('a'=>'青龙','b'=>'白虎','c'=>null);
if(in_array('青龙', $arr)){
    echo 'cunzai';
} else {
    echo 'bcunzai';
}
}
```

7.8 案例之小羊繁殖问题 (44)

【程序 10, 稍难一些, 学数组后再做】

一种羊 生命长度为 5 周年

满 2 周年生一只小羊 满四周年生一只小羊, 5 周年死

初始有 1 只 0 岁的羊, 20 年后有多少羊。

想做这道题, 需要再学几个数组函数

array_push — 将一个或多个单元压入数组的末尾

```
$arr = array('a','b','c');  
array_push($arr,'d');  
print_r($arr);
```

array_pop — 将数组最后一个单元弹出

```
$arr = array('a','b','c');  
array_pop($arr);  
print_r($arr);
```

array_shift — 将数组开头的单元移出数组

```
$arr = array('a','b','c');  
array_shift($arr);  
print_r($arr);
```

array_unshift — 在数组开头插入一个或多个单元

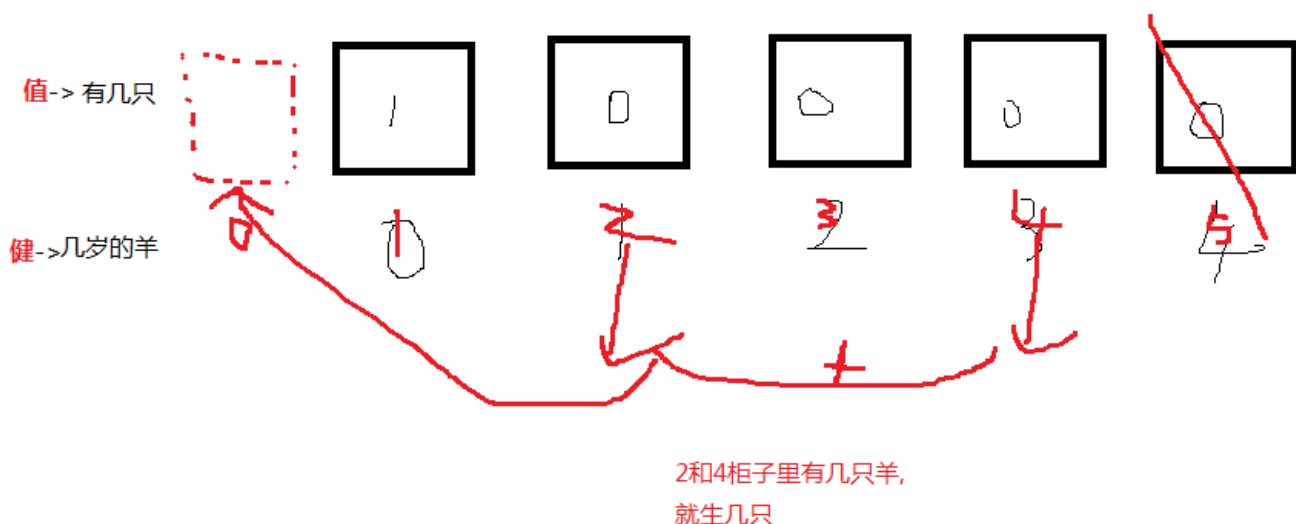
```
$arr = array('a','b','c');  
array_unshift($arr, 'wow');  
print_r($arr);
```

小羊繁殖

一种羊 生命长度为 5 周年

满 2 周年生一只小羊 满四周年生一只小羊, 5 周年死

初始有 1 只 0 岁的羊, 20 年后有多少羊。



假设0岁-4岁的羊分别关在5个笼子里面

```
$arr = array('1','0','0','0','0');
```

```
for($i=1;$i<=20;$i++) {
```

```
//第一年,那只0岁的羊一岁了,则0岁羊笼子中没羊了,5岁羊笼子应该死掉了
```

```
array_unshift($arr, '0');
```

```
array_pop($arr);
```

```
//print_r($arr);
```

```
//第2年和第4年,在2岁羊和4岁羊的笼子里面的羊都生下了羊放在了0岁羊的笼子里面
```

```
$arr[0] = $arr[2]+$arr[4];
```



```
}  
print_r($arr);  
echo array_sum($arr);//178
```

array_sum — 计算数组中所有值的和

猴子选大王：
n 只猴子围成一个圈，按顺时针方向从 1 到 n 编号；
然后从 1 号猴子开始，延顺时针方向报数，报到 m 的猴子出局
再从出局的猴子的下一只猴子开始报数，如此重复，直到剩下一只猴子，它就是大王。
设计程序要求：
1) 要求用户输入开始的猴子数 n，报数的最后一个数字 m
2) 给出猴王的初始编号
参考答案:<http://www.zixue.it/thread-10510-1-1.html>

思路: 将所有的猴子排成一排, 假设为一个数组, 如果这个单元不是m, 则放在这个数组的最后一个单元中,
如果是m则删除;

7.9 数组面试题

```
$arr1 = array(1,2,3,4,5);  
$arr2 = array('a','b','c','d','e');  
$arr3 = array(1,'a',2,'b',3,'c',4,'d',5,'e');  
//观察 arr1,arr2,arr3 的规律, 写一个函数,  
//参数为$arr1,$arr2,  
//输出$arr3
```

第八章 超全局变量 (45)

特殊的数组,是系统提供给我们的.

特点:在页面的任意部分,无论是函数里,还是正常的页面里,都能随时获取到这几个变量,它不受作用域的干扰.这意味着它们在一个脚本的全部作用域中都可使用,这就是超全局变量

超级全局变量
\$_GET // 地址栏上获得的值
\$_POST // POST 表单发送的数据
\$_REQUEST // 既有 GET,也有 POST 的内容(GET和POST的并集)

//后面会学到以下三个
\$_SESSION
\$_COOKIE
\$_FILES

\$_SERVER //web服务器的环境
\$_ENV // 服务器操作系统的环境变量,如操作系统型,linux,win,mac,环境变量等等

\$GLOBALS //引用全局作用域中可用的全部变量

面试题:<http://www.zixue.it/thread-266-1-1.html>

1.函数调用,写一个html,有post和get

\$_GET \$_POST \$_REQUEST

```
$arr = array(1,2);  
  
function t1() {  
    t2();  
}  
  
function t2() {  
    print_r($arr);  
    print_r($_GET);  
    print_r($_POST);  
    echo '<hr>';  
    print_r($_REQUEST);  
}
```

```
t1();
```

2.GET和POST都有id,那么\$_REQUEST怎样显示

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
</head>
<body>
  <form action="1.php?id=3" method="post">
    <p>姓名:<input type="text" name="name" id=""></p>
    <p>email:<input type="text" name="emai" id=""></p>
    <p><input type="text" name="id" id="" value="999"></p>
    <input type="submit" value="submit">
  </form>
</body>
</html>
```

POST的优先级似乎要比GET的要高,这决定于php的配置文件,php.ini

request_order = "GP"

request有合并的意思,先合并GET后POST,后者覆盖前者

3.\$_SERVER — 服务器和执行环境信息

是一个包含了诸如头信息(header)、路径(path)、以及脚本位置(script locations)等等信息的数组

对于我们做网站而言,服务器指的是web服务器

web服务器就是给我们提供网页服务的这种环境的软件

在win下是apache,在linux下nginx

这里的环境指的是apache的一个运行环境

```
echo '<pre>';
print_r($_SERVER);

SERVER_SOFTWARE    服务用到的软件
SERVER_NAME        域名
SERVER_ADDR        服务器地址
SERVER_PORT        服务器端口
REMOTE_ADDR        远程地址(来访者的IP)
DOCUMENT_ROOT      文件根目录
SCRIPT_FILENAME    当前运行的文件

访问: http://192.168.1.100/0903php/1.php
```

4.\$_ENV (enviroment 环境) 操作系统的环境

打印为空

操作系统的环境和安全密切相关,别人上传个.php就可以轻易查看到我们系统的环境,或者黑客利用漏洞,运行个脚本,看到了操作系统环境,很容易给他的破解带来方便.

所以在php.ini里面禁用了这个超全局变量

variables_order = "GPCSE"

即使禁止使用ENV,我们也可以获取环境变量

getenv — 获取一个环境变量的值

echo getenv('USERNAME');

在实际使用中,是禁用掉ENV的,如果想使用ENV的值,则可以用getenv获取

5.\$GLOBALS 它就像是全局变量的一个花名册

//分析过程

```
$a = 1;
$b = 2;
function t1() {
  print_r($GLOBALS);
  $GLOBALS['a'] = 99;
}
```

```
t1();  
echo $a;//99
```

第九章 常量 (46)

意义
声明
特点
检测
动态常量名
案例

变量特点: 可以读取值,也可以修改它的值

常量: 值一旦定义,不可更改

什么时候用到常量,某个重要的值我们不希望被更改,就可以使用常量.

就像现在的p2p网贷,有个固定的利率,不会修改

```
$money = 10000;  
$rate = 1.1;  
echo $money = $money * $rate,'<br >';
```

当我们更改了利率,现金也就变了.但是利率不能被修改

```
$rate = 1.2;  
echo $money = $money * $rate;
```

如何定义常量:

define — 定义一个常量

```
define('PI', 3.14);  
echo PI; //常量名前不用加$
```

常量的命名:

除了不用加\$了,跟变量的命名是一样的,不过习惯上全大写

常量特点:

声明后值不能修改值,也不能重新声明,也不能销毁.

```
define('PI', 3.14);  
PI = 3.15; //语法错误  
  
define('PI' , 3.18);  
echo PI;  
  
unset('PI');
```

检测常量是否存在:

isset — 检测变量是否设置

defined — 检查某个名称的常量是否存在

```
//感叹号(!)逻辑求反  
if(! defined('RATE')) {  
    define('RATE',1.1);  
}  
  
if(! defined('RATE')) {  
    define('RATE',1.1);  
}  
  
echo RATE;
```

先检测再定义的好处,不用担心重复定义报错

动态常量名

```
if(! defined('RATE')) {  
    define('RATE',1.1);  
}  
  
$name = 'RATE';  
echo $name,<br >;
```

动态常量名跟动态变量名不一样,动态变量名前面有\$符号(\$\$name),
会强行让变量的值按变量名来解析

那如果让变量值作为常量的名呢?

constant — 返回一个常量的值

```
echo constant($name); //拿$name的值当常量的名看,并解析输出
```

开发中常用的效果

在e框架,tp框架中经常碰到这样的代码

逻辑运算的短路特性(后面会讲到)

```
defined('DEBUG') || define('DEBUG',false); //框架系统代码容易看到这句话
```

可以理解的写法

```
define('DEBUG',true); //用户可以先自定义  
  
if(! defined('DEBUG')) {  
    define('DEBUG',false); //程序定义  
}  
var_dump(DEBUG);
```

常量的作用域

常量的作用域,不分全局或是局部,在页面的任意处都可以访问

```
function t1() {  
    t2();  
}  
  
function t2() {  
    t3();  
}  
  
function t3() {  
    define('PI',3.14);  
}  
t1();  
echo PI;
```

注意点:

定义常量的值,不能是数组,对象,资源

一般定义为 字符串,整型,浮点,布尔

第十章 文件包含 (47)

区别:include 和 require

include include_once 包含

require require_once 必须

1.文件包含的作用:

文件包含的作用在于代码的重用.

我们可以把常用的代码段写一个文件里,
当需要这些代码时,引入这个文件就可以了.

2.对比 include和require

1)分别用include 和 require引入一个存在的php文件

```
$a = 3;
```

2.php的代码如下

```
$a +=1;*/
```

```
$a = 3;  
include('./2.php');  
//require('./2.php');
```

2)再分别用引入一个不存在的 2-1.php文件

```
$a = 3;  
include('./2-1.php');  
//include被包含文件不存在时,代码会尽量往下执行,报警告错误,warning  
  
require('./2-1.php');  
//require被包含文件不存在时,代码不会再往下执行,报致命错误,fatal error  
  
echo $a;
```

什么时候用include和require?

底层库等,很重要的文件,没有它不能继续执行,就用require

如果是第三方的广告代码等,可以用include

3.加_once和不加once的区别

_once 作用: 只引入 1 次,如果之前已引用过,不再重复引用.

```
$a = 3;  
  
include('./2.php');  
include('./2.php');  
include('./2.php');  
  
include_once('./2.php');  
include_once('./2.php');  
include_once('./2.php');  
  
echo $a;
```

如果引入一些库文件,里面有很多函数,我们知道函数不能重复定义,

引入多次肯定会出问题

注意: 不加_once 速度快些.

4.被包含文件里可以像函数一样用 return

```
/*2.php 内容如下  
return array('a','b','c');  
*/  
  
$arr = include('./2.php');  
print_r($arr);
```

面试题:

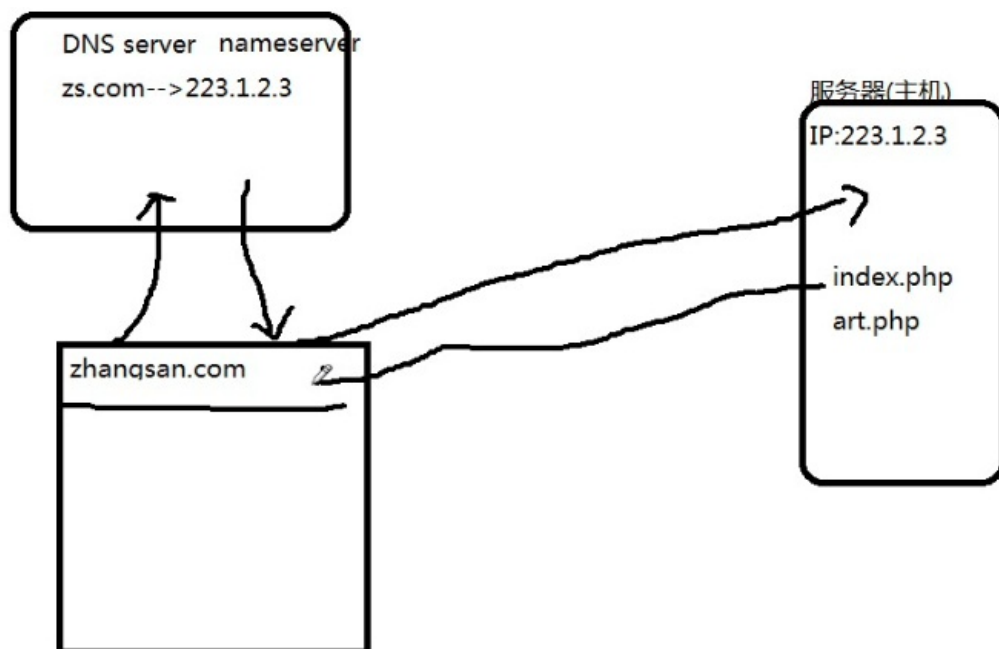
<http://www.zixue.it/thread-175-1-1.html>

第十一章 综合案例

11.1 IP 域名及 DNS 概念 (48)

IP: 是计算机互联网中的"门牌号", 192.168.1.123 (局域网 IP)

在互联网上发布的网站所用的 IP 是公网 IP.



域名: 域名--映射-->IP

http://www.baidu.com

域名 hosts 文件, DNS 服务器

在最早的 appnet 实验室, 有几十台机器, 相互连接起来.

相互访问, 有 IP 的概念, 有机器名的概念.

为了解决机器名与 IP 的转换问题,

实验室的人想了一个简单办法, -----hosts 文件.

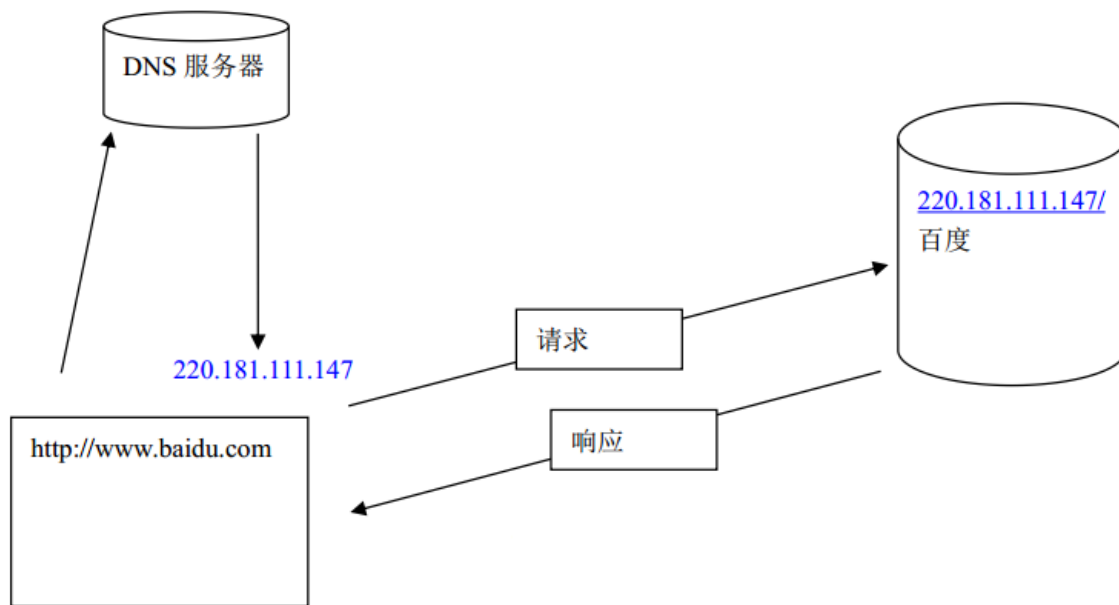
IP1	TOM
IP2	allen
IP3	white

随着机器的增多, hosts 要不断的新增记录, hosts 文件维护起来越来越麻烦.

于是就建立起专门服务器, 专门负责解析域名与 IP 的关系.

这种专门的服务器叫 "DNS 服务器"

全球有 10 多台根域名 DNS 服务器, 分布式在全球有好多子域名服务器; 每天都在承受几百亿次的方位



出于历史的先后顺序,hosts文件和DNS服务器都解析了一个域名
谁的优先级更高? -> hosts文件

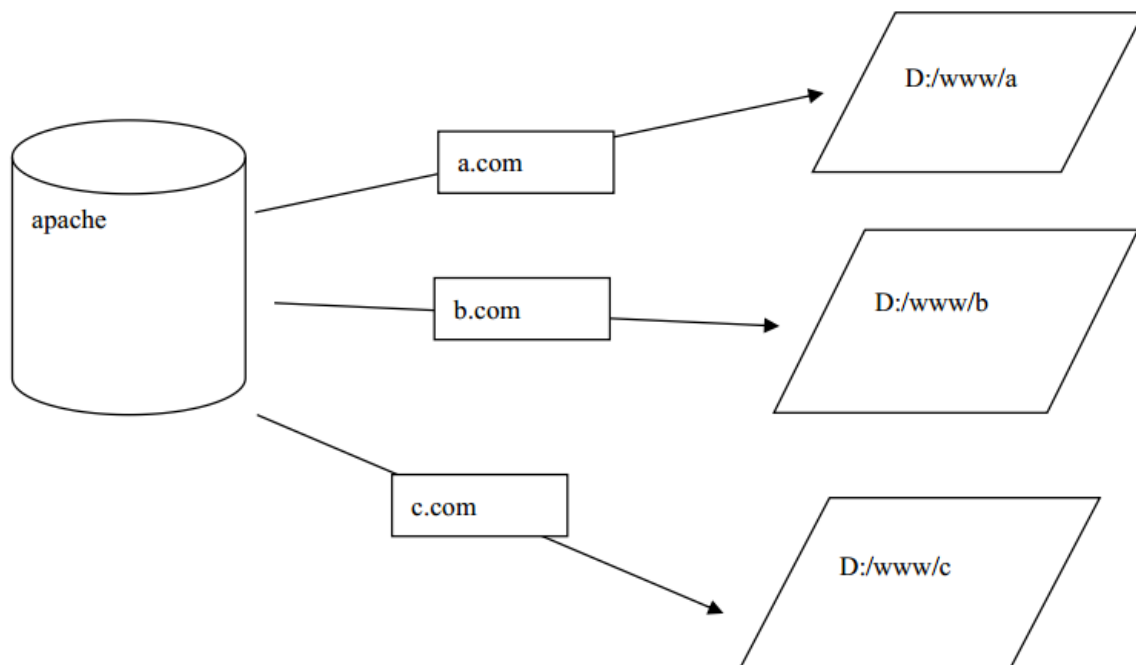
linux /etc/hosts

windows C:/Windows/System32/drivers/etc/hosts

有些木马文件就会串改我们的hosts文件,
将他们自己的网站,解析到baidu的地址上,他们网站的访问
量剧增,可以解析到上千万的ip

167.8.91 baidu.com

11.2 Apache 虚拟主机配置 (49)



空间很便宜,才几十就能一年

一台服务器上,往往放置着上百个甚至上千个的域名

一个服务器只有一个IP,服务器上那么多的域名

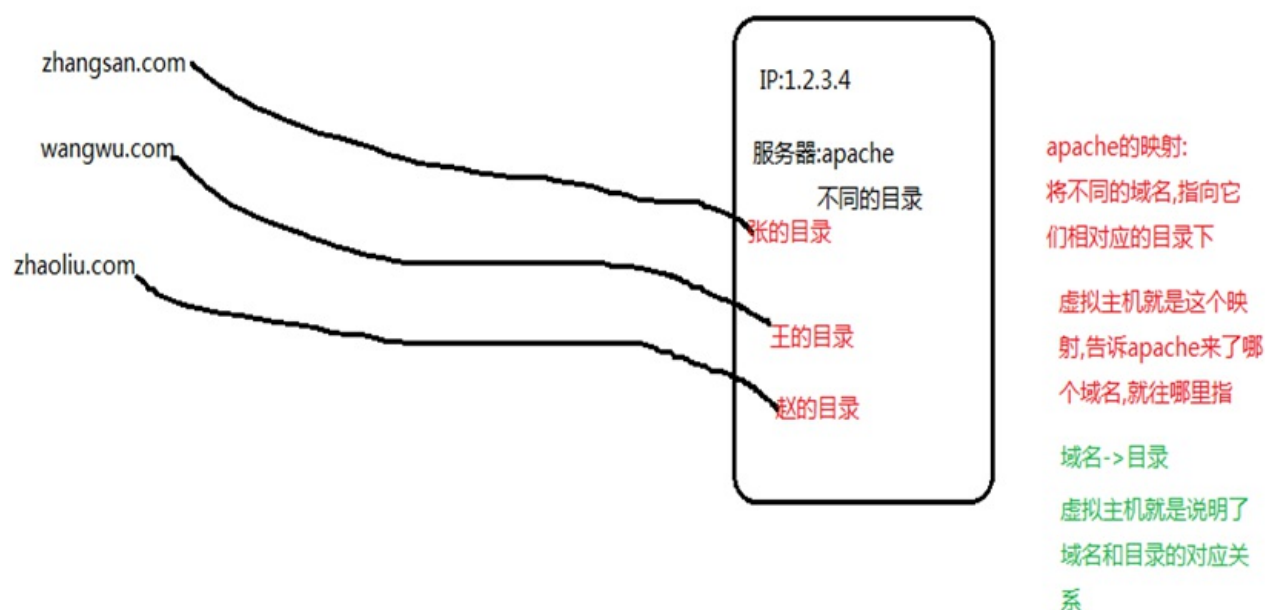
不同的域名在浏览器上输出的内容都不一样,

说明这台服务器能够识别不同的域名,在给它做不同的输出,

那么它是如何识别的呢?

虚拟主机: 就是告诉 apache,对于不同的域名,引导到不同的目录

它就是说明了一个关系：
域名到目录的一个对应关系



具体的操作:

1.更改hosts文件

C:\Windows\System32\drivers\etc\hosts
127.0.0.1 blog.com

```
127.0.0.1    localhost
127.0.0.1    blog.com
```

2.改apache的主配置文件

C:\wamp\bin\apache\Apache2.4.4\conf\httpd.conf

```
#Include conf/extra/httpd-vhosts.conf
```

将include包含的这个文件引入,去掉屏蔽的#

```
Include conf/extra/httpd-vhosts.conf
```

3.改httpd-vhosts.conf

C:\wamp\bin\apache\Apache2.4.4\conf\extra\
httpd-vhosts.conf

添加虚拟主机记录

```
<VirtualHost *:80>
    ServerAdmin webmaster@dummy-host.localhost ----->网站管理的 email
    DocumentRoot "C:/amp/www/sohu" ----->虚拟主机的根目录
    ServerName sohu.com ----->主机名
    ServerAlias old.sohu.com ----->别名
    ErrorLog "logs/sohu.com-error.log" ---->错误日志
    CustomLog "logs/sohu.com-access.log" common ---->访问日志
</VirtualHost>
```

```
<VirtualHost *:80>
    DocumentRoot "c:/wamp/www/blog"
    ServerName blog.com
</VirtualHost>
```

只需要留下 servername域名和documentroot域名对应的目录即可
并在blog下写一个index.html首页等待blog.com的访问

4.最后重启apache

课后作业:

在自己机器上建立 2 个虚拟主机

Sohu.com, 163.com

主页分别显示 hello,sohu, 和 hello ,163

11.3 php的命令行模式 (50)

php不仅仅可以在web服务器中充当重要角色。

在命令行一样可以执行

cd 进入到php的目录下

php -v 可查看当前php的版本

其他的选项有: -m、-i

-m 会显示当前php加载的有效模块。

-i 则输出无html格式的phpinfo。

使用 -ini 选项可以输出当前php加载ini配置文件的数量、路径信息

在命令行中运行php程序

在php 的根目录下 放置一个hello.php 文件 字符集改为gbk

在命令行 敲入: php -f hello.php

可以访问该php文件

```
<?php
echo "运行php命令行echo";
?>
```

在命令行中执行php文件的好处之一就是可以通过脚本实现一些计划任务的执行。而毋须通过web服务器

11.4 apache 和 php 配合的原理 (51)

php, apache和mysql组合的工作过程:

PHP的所有应用程序都是通过WEB服务器(如IIS或Apache)和PHP引擎程序解释执行完成的, 工作过程:

(1)当用户在浏览器地址中输入要访问的PHP页面文件名, 然后回车就会触发这个PHP请求, 并将请求传送化支持PHP的WEB服务器。

(2)WEB服务器接受这个请求, 并根据其后缀进行判断如果是一个PHP请求, WEB服务器从硬盘或内存中取出用户要访问的PHP应用程序, 并将其发送给PHP引擎程序。

(3)PHP引擎程序将会对WEB服务器传送过来的文件从头到尾进行扫描并根据命令从后台读取, 处理数据, 并动态地生成相应的HTML页面。

(4)PHP引擎将生成HTML页面返回给WEB服务器。WEB服务器再将HTML页面返回给客户端浏览器。

简言之: apache是WEB服务器, 它可以解析php语言

apache, php以及mysql的关系:

apache是web服务器, 主要是请求和响应的功能, 是一种软件, 用于发布网站。在配置文件中可以设置文件根目录, 处理的文件类型以及端口等。apache单独不能处理php代码。

php (php 应用程序服务器): 不能单独存在的, 需要以apache为依托。也可以说php是apache功能模块的一种扩展, php作为apache的外挂, 用于解析php代码。经过解析的动态网页此时就不存在<?php ?>了。当apache服务启动的同时也启动了php。也就是说php必须和apache安装在同一台服务器上, 不可以分割。

mysql: 一种开源的很流行的关系型数据库。与apache无关, 只是在响应php代码中的sql操作的时候才被访问。不需要与apache安装在同一台服务器上, 可以单独存在, 只需php远程连接即可

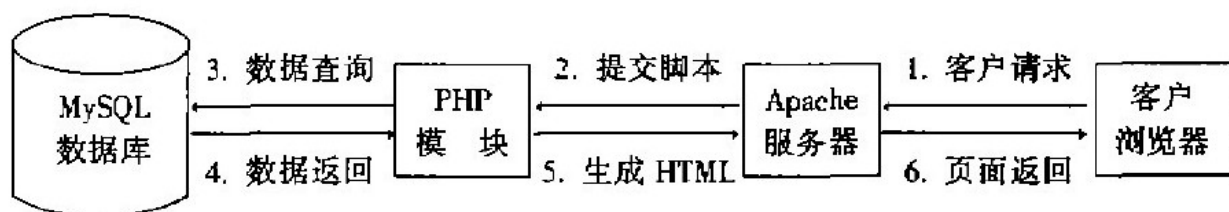


图 1 系统工作原理示意图

第十二章 进制与位运算

以下章节较难,但是使并不频繁
能理解最好,不能理解可以作为一个了解

12.1 进制概念 (52)

10 进制
8 进制
16 进制
2 进制

生活中常用 10 进制
计算机常用 2 进制,8 进制,16 进制

10 进制

```
... 万位 千位 百位 十位 个位 (满10进1)
    9    9    9    9    9

echo 123 , '<br />'; // 从右到左 3个位,分别是以 1顶1, 以1顶10, 以1顶100
echo 1*3 + 10*2 + 100*1 , '<br />';
```

10进制方便人用自己的手指数数,玛雅人用的是20进制,因为它们不穿鞋
进制说白了:就是以1顶几

8 进制 (满8进1)

8 进制, 前导 0 代表 8 进制
在数字前加0,可以让php将这个数字按照8进制来理解,输出这个数的10进制结果

```
echo 0123 , '<br />' ; // 从右到左 3 个位,分别是以 1 顶 1,以 1 顶 8,以 1 顶 64
echo 027 , '<br />'; // 23
echo 028 , '<br />';// 2, 当出现不合法值,后面的字符会忽略
```

16 进制 (满16进1)

16 进制,前导 0x 代表 16 进制 ,0 1 2 3 4 5 6 7 8 9 a b c d e f

```
echo 0x123 , '<br />'; // 从右到左 3 个位,分别是以 1 顶 1,以 1 顶 16,以 1 顶256
echo 0x1c , '<br />';
```

2 进制 (满2进1)

2 进制 ,PHP 从5.4版本开始,可以直接写2进制数 由0/1组成
前导 0b 代表 2进制

```
// 从右到左,分别以 1 顶 1,顶 2,顶 4,顶 8,顶 16...
echo 0b0011;
echo 0b01100;
```

1个字节占8个位

```
0000 0000
echo 0b11111111;//255
echo 0b10000000;//128 (中间的位置大约是127)
```

画个圆,一个字节的占8个位,从1-255
顺时针方向是0-255,如果是逆时针方向,255的位置也可以理解为-1...

```
10进制      2 进制
5            0000 0101
255         1111 1111,128+64+32+16+8+4+2+1
-1          1111 1111,
```

像上面 8 个 1,到底理解成 255,还是理解成-1

理解时是否会混乱?

这个取决于程序.
比如在 mysql 中,
int 则理解为-1,
unsigned int (无符号类型全是正数)理解为 255,

PHP 的 int 型则是有符号的
取决于8个位的 最左面的位(最高位) 是否为1
如果为 1 则是 负数 -1~-128
如果为 0 则是 整数 0~127

课后作业:

为什么程序员总是把圣诞节与万圣节搞混?
DEC 25 圣诞节, OCT 31 (考进制转换)

12.2 位运算 (53)

什么是位运算?
一个字节有8个位,我们将这8个位拿来进行运算,就叫位运算.
位运算 是针对字节上的位来进行运算
& | ^ ~
<< >>
位上有1/0,如果将1/0当成真假来看, 1是真 0是假

& 逻辑与

逻辑与 &,做逻辑运算

```
5      0000 0101
12     0000 1100
&-----
      0000 0100

echo 5 & 12 ;//4
```

| 逻辑或

逻辑或 |,做逻辑运算

```
5      0000 0101
12     0000 1100
|-----
      0000 1101

echo 5 | 12 ;//13
```

^ 逻辑异或

逻辑异或 ^, 两者不一样则为真 两者一样则为假

```
5      0000 0101
12     0000 1100
^-----
      0000 1001

echo 5 ^ 12 ;//9
```

~ 逻辑反

逻辑反 ~,只能求一个字符,就是 1->0 , 0->1

```
5      0000 0101
~-----
      1111 1010

echo ~5 ;//-6
```

为什么是-6不是正数,因为在php中不支持声明是无符号类型,当最高位为1时,则是负数

<< >> 位运算

<< 按位左移 空位补0

```
5      0000 0101
<<-----
      0000 1010

echo 5 << 1;//10
```

<< 按位右移 空位补0

```
5      0000 0101
>>-----
      0000 0010

echo 5 >> 1;//2
```

2进制,左移一位增大2倍

如何将一个数快速的增大8倍?

```
echo 2<<3;//16
```

按位移动比乘法要快

第十三章 浮点数不够精确 (54)

2.3

10进制,小数点右边的第一位数,10^1, 是以1顶 1/10

0b1.1

2进制,小数点右边的第一位数,2^1, 是以1顶 1/2

```
10 进制    2 进制
0.5        0.1
0.8        0.11001100110011001100
0.8 = 1/2 + 1/4 + 1/32 + 1/64 + 1/512 + 1/1024 .....
```

在计算机中,存储是有字节限制的
对于一个 小数在 10 进制下,是有限的,转成 2 进制要 无限循环,必定是要舍去一部分的.
因此, 损失一些 精度,导致 浮点数计算 和数学上结果不一致.

```
if ((0.1+0.2) == 0.3) {
    echo 'eq';
} else {
    echo 'neq';
}
```

银行一般都存整数,精确到分
整数既不会产生精度损失,且计算较快

课后作业:

浮点数运算的不精确性可以产生灾难性的后果. 1991 年 2 月 25 日,在海湾战争期间,沙特阿拉伯达摩地区设置的美国爱国者导弹,拦截伊拉克的飞毛腿导弹失败.
飞毛腿击中美国一个兵营,造成 28 名美国士兵死亡. 美国总审计局(GAO)对失败原因做了详细的分析,并且确定了,一个潜在的原因在于一个数字计算不精确.
在这个作业中,你将重现审计局的相关工作.
爱国者导弹系统中,有一个内置的时钟,实现一个计数器,每 0.1 秒就加 1. 为了以秒为单位来确定时间,程序用一个 24 位近似于 1/10 的 2 进制小数值来乘以计数器的值.
特别地:0.1 转成 2 进制序列是无序序列 0.000110011[0011]..2 进制.

问: 当系统初始启动时,时钟从 0 开始,并且保持计数,在这个例子中,系统已经运行了约 100 个小时, 程序计算的时间和实际时间相差了多少?

问: 系统根据一枚来袭的导弹的速率和它最后被雷达侦测到的时间,来预测他将在哪里出现.

假定飞毛腿的速率约为 200 米/秒,对它的距离预测偏差了多少.

第十四章 逻辑运算的短路特性 (55)

逻辑运算的短路特性与运算符优先级

1.白富美要求有房有车才嫁

```
$house = false;  
$car = true;
```

house已经为假,又用并且计算,结果为假

程序并不会去判断\$car的值

```
var_dump($house && $car);  
if($car && $house) {  
    echo '嫁';  
} else {  
    echo '不嫁';  
}
```

2.&& 的短路

验证一下 逻辑运算 && 的短路特性

短路 当能够判断表达式的结果时,后面的表达式不执行

```
$house = true;  
$car = false;  
$a = 1;  
$car && ($a = 5);  
echo $a;  
  
$house && ($a = 8);  
echo $a;
```

何为短路?

当能够判断整体表达式的结果时,后面的表达式就不执行

3.|| 的短路

```
//验证一下 逻辑运算 || 的短路特性  
$b = 2;  
$house || ($b = 10);  
echo $b;  
  
$car || ($b = 5);  
echo $b;
```

利用短路特性

在某些场合中,可以 把我们的代码写的 简洁 优美

验证一下逻辑 || 运算的短路特性

1)判断常量ROOT是否声明过

```
define('ROOT' , dirname(__DIR__));  
  
if(!defined('ROOT')) {  
    define('ROOT' , dirname(__DIR__));  
}
```

2)利用 逻辑运算的短路特性 一行就可以了

```
defined('ROOT') || define('ROOT',dirname(__DIR__));  
  
echo ROOT;
```

tp和e框架都有这样的写法

书写不规范,导致的问题(讲解)

下面这个是反面例子,是因为写法不规范,导致的问题
= 的优先级是最低的,它没有||高

```
$a = 3;
$b = 5;
if ( $a = 5 || $b = 7) {
//因为$a,$b 赋值的时候没有加小括号 而 = 的优先级没有 || 高
//if ( ($a = 5) || ($b = 7) )
//所以 5||$b 返回 true; $a = true; true++ 还是true (echo true;显示1)
$a++;//var_dump($a);
$b++;
}
echo $a,$b//1,6
```

程序 能运行

是因为人给它写了个 解释器 或者 编译器, 来解释它执行的

所以它的行为不是一种 天然的真理, 跟物理是不一样的

它是由人 规定的一种 行为.

所以, 你不要去追究一些 稀奇古怪的语法

如果碰到面试官考这样的题,你可以告诉他,这是毫无意义的.

比如 ,有些面试官考 \$a++ + ++\$a 等,这是毫无意义的.

因为, 这些行为的意义取决于 它的解释器 或者编译器 .

就拿c语言来说 ,

1.以上的代码, 放在 windows的wc下,和linux的gcc下,他俩的答案是不同的

(写一段话,中国人和美国人的理解肯定是不同的)

2.这种写法是不规范的,++是有副作用的 , 特别是 \$a=\$a++;在赋值的情况下

到底是先++再赋值,还是怎么样?

它没有一个原子性,所以就有分歧,到底先算还是怎样;

这种行为,在语言里,称为未定义行为(c语言也称之为未定义行为);

未定义行为到底会发生什么,这谁也不知道.就看编译器是怎么解释的

写程序需要用,易读,简洁的语法 把功能做出来 , 而不是去钻那些 未定义行为的 牛角尖

第十五章 递归

15.1 递归概念 (56)

编程大师说:

编程中有两个重要的难点

1.递归 2.指针(PHP里没有指针)

难点在于: 你要想理解递归,就必须先理解递归

```
function t() {
//死循环
echo '!';
t();
}
t();
```

递归的特点:

1.自身调用自身 (如果一直调用下去肯定不行,死循环)

2.要有一个明确的边界,能够终止调用

1.写一个sum求和的函数

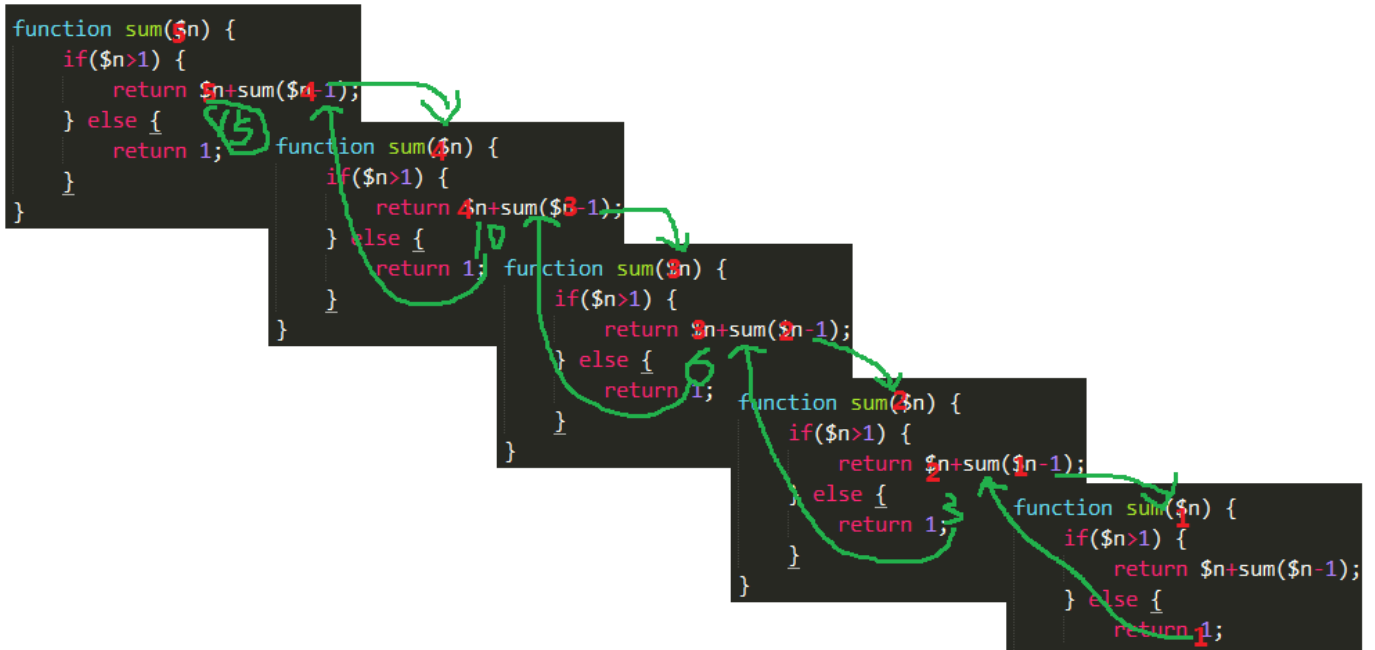
输入参数n,可以求到1+2+3+...+n的和

```
1+到100的和是多少?1+2+3+...+100
1+到99的和是多少?
1+到89的和是:[1+到88之和]
89+87+[1+到86之和]
```

```
...
89+87+...+2+1

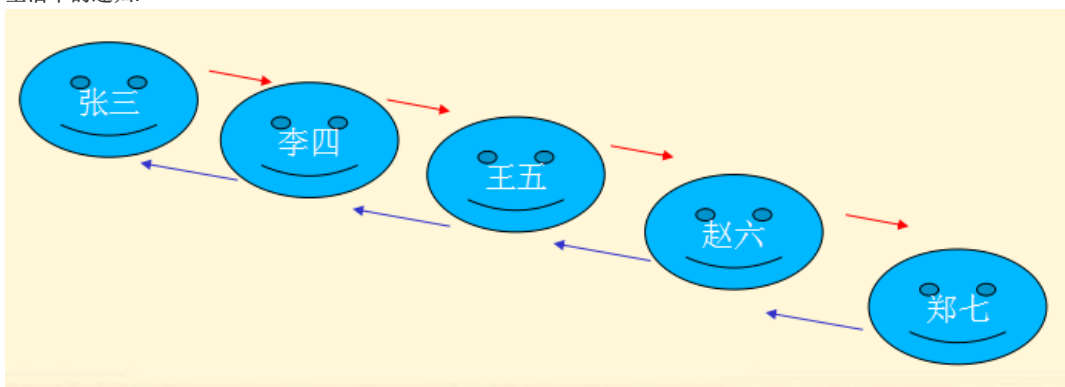
function sum($n) {
    if($n>1) {
        return $n+sum($n-1);
    } else {
        return 1;
    }
}
echo sum(100);
```

我们不能理解是因为
计算机计算的太快,我们用 画图 的慢动作来 解释



sum(5);	
5 + sum(4)	[sum(4) == 4+sum(3)]
5 + 4 + sum(3)	[sum(3) == 3+sum(2)]
5 + 4 + 3 + sum(2)	[sum(2) == 2+sum(1)]
5 + 4 + 3 + 2 + sum(1)	[sum(1) == 1]
5 + 4 + 3 + 2 + 1	

生活中的递归:



递归:
一个一个递过来,再一个一个归回去 [这句话本身就是递归]

15.2 递归技巧 (57)

一般我们需要工作2年左右,才能熟练的写出递归
我们现在学一个递归技巧,可以以我们目前的水平写出递归
技巧:
假设法--假设自己的函数已经完成

用递归打印当前目前下的所有文件,目录及子目录...

```
/*
$path 目录
$lev 层级 进入一层,目录更深,明显显示出来
*/

function showDir($path,$lev=0) {
    $fh = opendir($path);
    while($row = readdir($fh)) {
        //如果目录为.和..就跳过
        if(($row=='.' || ($row=='..')) {
            continue;
        }
        echo str_repeat("    ", $lev),$row,'<br >';
        //如果目录里还有目录,继续往下读取目录
        if(is_dir($path.'/'.$row)) {
            showdir($path.'/'.$row,$lev+1);
        }
    }
    closedir($fh);
}

showDir('.');
```

技巧：

假设自己能做出来,先做第一层,完成之后,看哪个地方还需要调用自身,再去调用;

15.4 递归练习题 (58)

1) 一个多维数组,如果单元值为数字,则把其值修改为原来的 2 倍.

如 `array(1,2,'b',array(3,'c',array(4,5)))`;

变成

```
array(2,4,'b',array(6,'c',array(8,10)));
```

扩展题目: 对 POST,GET 做递归

```
$arr = array(1,2,3,array('c',4,array(5,6)));

function multArr($arr) {
    foreach($arr as $k=>$v) {
        //如果值是int 则加倍
        if(is_int($v)) {
            $arr[$k] = $v*2;
        }else if(is_array($v)) {
            //如果值是数组
            $arr[$k] = multArr($v);
        }
    }

    return $arr;
}

print_r(multArr($arr));
```

2)写递归函数,计算所有单元的和

```
$arr = array(1,2,3,array(4,array(5,6)));

function sumHe($arr) {
    $i = 0;
    foreach($arr as $k=>$v) {
        if(is_int($v)) {
            $i = $i+$v;
        } else if(is_array($v)) {
            $i +=sumHe($v);
        }
    }
}
```



```
        return $i;
    }

    echo sumHe($arr);
```

3)递归创建级联目录

如给定 './a/b/c/d/e', 则 ./a 都不存在, 则递归创建之.

4)递归删除目录

如给定 './a', 则要把 a 目录及下级子目录及文件, 全部删除掉.

5)给定如下数组,完成无限级分类

```
array(
    array('id'=>1,'area'=>'北京','pid'=>0),
    array('id'=>2,'area'=>'河北','pid'=>0),
    array('id'=>3,'area'=>'保定','pid'=>2),
    array('id'=>4,'area'=>'易县','pid'=>3),
    array('id'=>5,'area'=>'海淀','pid'=>1)
);
```

要求: 写函数得到如下结果

f(0) 输出 0 号地区下的子孙地区

北京

海淀

河北

保定

易县

f(2)得到 2 号地区下的子孙地区

保定

易县

即无限级分类

15.3 static 静态变量 (59)

1.普通局部变量,函数调用时->初始化

函数结束时->从内存消失

```
function t() {
    $a = 3;
    $a += 1;
    return $a;
}

echo t(), '<br >';
echo t(), '<br >';
echo t(), '<br >';
```

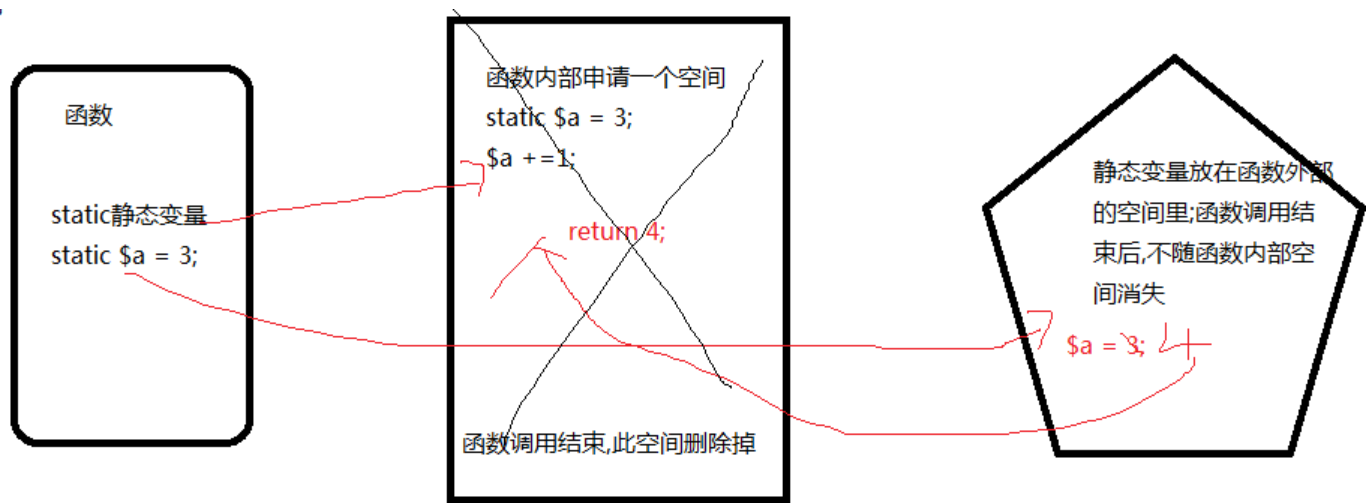
函数放在这里,不调用,不执行 执行是在内存中来回倒腾数据

当我们调用的时候,会在内存中申请一个空间,来执行

函数运行完毕,空间会从内存中清除掉. 再调用->再申请

清除前,return 一个值可供外界使用

2.static 静态变量



函数初次调用时->初始化

函数结束时->不从内存消失,下次接着用

```
function t1() {
    static $a = 3;
    $a += 1;
    return $a;
}

echo t1(), '<br >';
echo t1(), '<br >';
echo t1(), '<br >';
```

实际应用的场合

1) 打开一个文件, 在一个页面多次打开, 有可能会出错

```
function openfile($file) {
    $fh = fopen($file, 'r');
    return $fh;
}

// 打开文件三次
print_r(openfile('./a.txt'));
print_r(openfile('./a.txt'));
print_r(openfile('./a.txt'));
```

2) 如何让我们的文件只打开一次呢

避免文件, 数据库等多次打开和连接

```
function openfile($file) {
    static $fh = null;
    if($fh === null) {
        $fh = fopen($file, 'r');
    }
    return $fh;
}

// 打开文件三次
print_r(openfile('./a.txt'));
print_r(openfile('./a.txt'));
print_r(openfile('./a.txt'));
```

TP 的 C 函数为例(讲解)

利用静态变量, 添加并储存 配置

```
function C($k, $v) {
    static $cfg = array();
    $cfg[$k] = $v;
    return $cfg;
}
```

```
print_r(C('host' , 'localhost'));  
print_r(C('Debug' , true));  
print_r(C('pass' , '123'));
```