Php的基本结构是B/S架构，即浏览器/服务器端的交互。

标准标记格式：<?php ?>

多行注释“/\*…\*/”中可以嵌套单行注释，但不能嵌套“/\*…\*/”多行注释。因为第一个“/\*”会以它后面出现的“\*/”作为配对的结束注释符，注释一大块代码容易出错。

在php中，常量一般使用define（）函数来声明。

例：bool define (string $name, mixed $value [,bool $case\_insensitive=false ] )

参数$name和$value是必选的，分别用于指定常量的名称和值，参数$case\_insensitive是可选的，用于指定常量名称是否对大小写敏感，如果值为true表示调用该常量时，常量名对大小写不敏感，反之则敏感，默认值为false。

预定义常量：是专门用于获取php中的信息，并且是不允许开发人员随意修改的。

注意：\_\_FILE\_\_和\_\_LINE\_\_中的“\_ \_”是两条下划线，而不是一条“—”。

Example:

echo \_\_FILE\_\_;

除了传值赋值外，变量还有一种赋值方式是引用赋值。

表示新变量简单引用了原始变量，这时如果一个变量改变，则另一个变量也随之变化。想要实现变量的引用赋值，需要将“&”符号加到要赋值的变量前。

例： $txt=’Hello’;

$new\_txt=&$txt; // $new\_txt引用&$txt，此后两变量的值同时变化。

标量类型：只能存储一个数据。

例：boolean、 integer、 float 、string

复合类型：可以存储一组数据。

例：array 、object

Float浮点型只有14位数十进制数字的精度，从左边第一个非0数字开始。

String字符串型：

包含在双引号中的字符会被解析，而包含在单引号中的字符串不会解析，只会输出其字符本身。

在双引号定义的字符串中，由于花括号“｛｝”无法被转义，只有当“$”紧挨着才会被识别，且无论在左边还是右边，中间也不能有其他字符。

检测变量的函数：is \_\*()，括号内为要检测的变量，符合返回ture，否则返回false。

例：<?php

$a=null;

echo '检查是否为空：'.is\_null($a); //检测变量a是否属于空值类型

?>

定义一个变量时，需要使用$符号表示，在php中，还可以将一个变量的值作为另一个变量的名称，即可变变量，它的实现过程就是在变量的前面加一个$符号。

例：$txt='abc';

$$txt='hello'; //等价于$abc=’hello’

echo "$abc";

**自动类型转换**

1. 转换成布尔型规则

|  |  |
| --- | --- |
| 整型值0 | false |
| 浮点型值 0.0 |
| 空字符串，以及字符串“0” |
| 不包括任何元素的数组 |
| 不包括任何成员变量的对象 |
| 除上述以外的值 | true |

2.转换成整型规则

(1)布尔型转换成整型：布尔值true，转换成整数1;布尔值false，转换成整数0。

(2)浮点型转换成整型：浮点数转换成整数时，将向下取整。

(3)字符串转换成整型：字符串的开始部分决定它的值。如果该字符串以合法的数值开始，则使用该值，否则其值为0。合法数值包括可选的正负号，后面跟着一个或多个数字（可能有小数点），再跟着可选的指数部分。指数部分由“e”或“E”后面跟着一个或多个数字组成。如果该字符串包含“.”“e”“E”，则会被作为float来取值；否则会被作为整数来取值。

3.转换成字符串型规则

(1)布尔型转换成字符串：布尔值true，转换成字符串“1”；布尔值false则转换成空字符串” ”。

(2)整型或浮点型转换成字符串：把数字的字面样式转换成string形式。

|  |  |
| --- | --- |
| echo | 只是输出值 |
| var\_dump() | 输出数值的类型信息 |

**算数运算符**

1.当有浮点数参与运算时，运算结果的数据类型总是浮点型。当整数与整数运算的结果是小数时，其数据类型也是浮点型。

例：0.2+0.8的结果是float (1)。

2.在进行取模运算时，运算结果的正负取决于被模数(%左边的数)的符号，与模数(%右边的数)的符号无关。

例：(-5)%3=-2，而5%(-3)=2

“.=”表示对两个字符串进行连接操作，生成一个新的字符串并赋值给变量，它又被称为“字符串运算符”。

例：$a='accel';

$a.='world';

echo $a; //结果为'accelworld'

递增递减运算符包括数字和字符，数字规则与c语言一样，故只对字符进行说明。

**递增递减字符**

$x='a';

echo --$x.'<br>'; //输出结果：a

echo ++$x.'<br>'; //输出结果：b

$x='z';

echo ++$x.'<br>'; //输出结果：aa

$x='Z';

echo ++$x.'<br>'; //输出结果：AA

$x='abcd';

echo ++$x.'<br>'; //输出结果：abce

结论：

1.字符变量只能递增，不能递减，并且只支持纯字母(a~z和A~Z)。

2.对z或Z之前的单个字符进行操作时，按照英文字母的排列顺序递增。

3.对z或Z进行操作时，字符会前进一位，变成aa或AA。

4.对由多个字母组成的字符串进行操作时，只会对字符串末尾的单个字符前进一位。

5.不会对字母字符串以外的字符串起作用。

6.当操作数为NULL时，递增的结果为1，递减不受影响。

**比较运算符**：如果参与比较运算的操作数中含有数字类型的数据时，则在比较的过程中，会将所有的操作数转换成数值，再进行比较。

**逻辑运算符**

xor 异或运算符

例：$a xor $b

两个的值只要有一个为true，结果就为true；只有两个都为false时，结果才为false

**位运算符：**是针对二进制数的每一位进行运算的符号，它专门针对数字0和1进行操作。

|  |  |  |  |
| --- | --- | --- | --- |
| 运算符 | 名称 | 例子 | 结果 |
| & | 按位与 | $a&$b | $a和$b每一位进行“与”操作后的结果 |
| | | 按位或 | $a|$b | $a和$b每一位进行“或”操作后的结果 |
| ~ | 按位非 | ~ $a | $a的每一位进行“非”操作后的结果 |
| ^ | 按位异或 | $a^$b | $a和$b每一位进行“异或”操作后的结果 |
| << | 左移 | $a<<$b | 将$a左移$b次(每一次移动都表示“乘以2”) |
| >> | 右移 | $a>>$b | 将$a左移$b次(每一次移动都表示“除以2”) |

**错误控制运算符：**使用@符号来表示，把它放在一个php表达式之前，将忽略该表达式可能产生的任何错误信息，且只对表达式有效。

注意：要加括号 例：$a=@(4/0) ;

**Switch语句**

1.可以用分号代替case语句后的冒号，执行效果一样。

2.当for语句的每个表达式都为空时，表示该for循环语句的循环条件永远满足，进入无限循环的状态。

**goto语句**：作用是跳转到程序中的另一位置。在目标位置用目标名称加上冒号来标记，跳转指令是goto之后加上目标位置的标记。

Php中的goto语句只能在同一个文件或作用域中跳转，也就是说无法跳出一个函数或类方法，也无法跳入另一个函数。

例： for ($i=1,$j=10;$i<20;$i++) //$i++其实没有运行

{

while($j--) //while语句判断为真，故一直在循环，直到$j等于5才停止。

{

if($j==5)

goto end;

}

}

echo '$i='.$i;

end:

echo '$i='.$i. ',$j='.$j;



注：当条件满足“$j==5”时，使用“goto end;”方式，使程序跳转到第8行代码的“end:”标记处，并继续向下执行打印出此时$i和$j 的值。

**替换语法：**对流程控制语句提供的替换语句。

语法规则：把左花括号( { )换成冒号( : )，把右花括号( } )分别换成endif;，endswitch;，endwhile;，endfor;。

例：for($i=1;$i<4;$i++)

{

echo $i."<br>";

}

**替换后：**

for($i=1;$i<4;$i++)

:

echo $i."<br>";

endfor;

注意：对于do…while循环，没有可替换的语法。