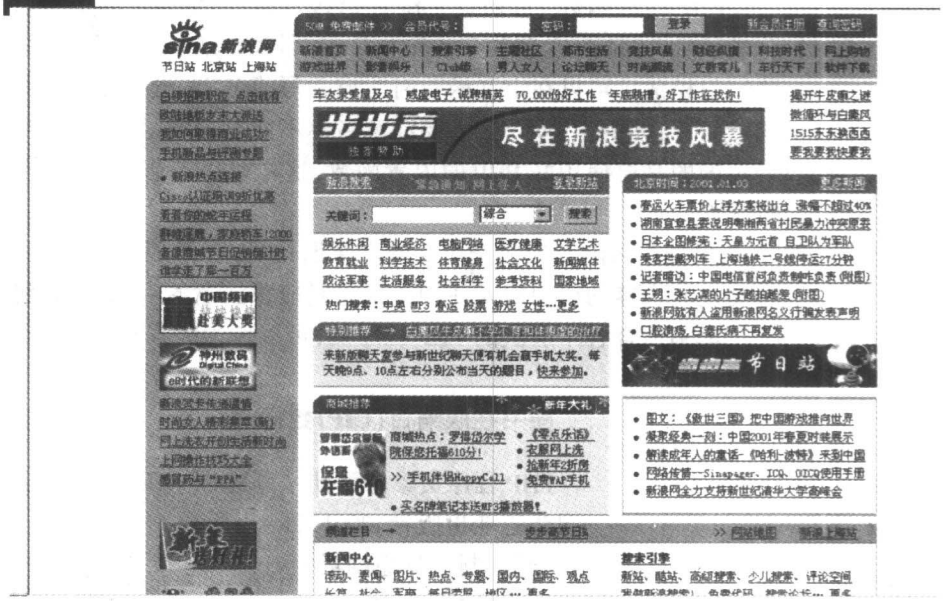
网页制作梦想剧场

国防工业出版社·北京



森林图书工作室　编

PHP 程序员参考手册

DREAM OF HOMEPAGE BUILDING

内 容 简 介

在 PHP 中，允许程序设计者将常用的流程或者变量等元素，组织成一个固定的格式。也就是说用户可以自行组合函数或者是对象。

PHP 中的函数（function）和 C 语言一样，包括有传回值的及无传回值的，不像 Pascal 分成函数（function）和程序（procedure）那么复杂。在函数的名称上，PHP 对于大小写的管制很松散。可以在定义函数时写成大写的名字，而在使用时使用小写的名字。总之，对函数而言，不用管大小写，只要注意名称不能有重复。除了要学会定义自己的函数以外，最关键的是要使用好 PHP 的内置函数，这样可以节省大量编程时间，使程序更有效率和可读性。

本书是 PHP 函数的全中文参考手册，详细介绍了 PHP 内置函数的语法、传回值、函数种类及其内容的说明以及使用该函数时应该注意的事项，总共包括 74 类 1489 种函数。

本书适合各个层次的 PHP 程序员，可以作为编程时参考及平时的查阅。

图书在版编目(CIP)数据

PHP 程序员参考手册 / 森林图书工作室编 . —北京：国防工业出版社，2001.11
（网页制作梦想剧场）
ISBN 7-118-02629-8

Ⅰ. P... Ⅱ. 森 ... Ⅲ. PHP 语言 - 程序设计 - 技术手册 Ⅳ. TP312 - 62

中国版本图书馆 CIP 数据核字(2001)第 051094 号

国防工业出版社出版发行
（北京市海淀区紫竹院南路 23 号）
（邮政编码 100044）
三河市腾飞胶印厂印刷
新华书店经售
*
开本 787 × 1092 1/16 印张 26¼ 611 千字
2001 年 11 月第 1 版 2001 年 11 月北京第 1 次印刷
印数：1 - 3000 册 定价：38.00

出版说明

目前，网页制作如火如荼，网站建设热火朝天。特别是电子商务的发展正在关键的时刻，各商家都在作最后的冲刺，网页月月改版，信息日日更新，而个人主页的制作更是变成了一种新时尚。

软件公司更是极力推广其先进的网页制作技术。Macromedia 与 Adobe 公司的产品在不断更新；蓝色巨人 IBM 一直从事着电子商务平台的推广工作，其网站建设专家 WebSphere 是电子商务平台的解决方案之一；在自由软件领域，PHP 的版本也在不断更新，它与 MySQL 一起构成了 Linux & Unix 的网站基础；另外，ASP、JSP、DHTML、XML、XSL 等技术更是层出不穷。

可以看到，在网页制作方面，技术越来越先进，使用越来越简单。各大软件公司都在争先恐后地推出新的产品，已经形成群雄逐鹿的局面。

为了让广大读者更快更好地掌握各种网页制作工具的用法，又快又好地制作出符合不同用途的网页，为了给社会上相关培训班提供合适的教材，我们特意组织编写了本套丛书。

丛书兼顾系统性与实用性，但以应用为主，通过例子、技巧带动对软件的系统学习，是网页制作培训班的理想教材，更是初、中级网页制作人员的最佳自学读物，也可以作为专业网站制作和管理人员的参考用书。

本丛书侧重于网页制作的入门知识与基本技术,至于网页美化方面的知识请读者参考本套丛书的姊妹篇《数码创意梦想剧场》。《数码创意梦想剧场》从艺术角度介绍了如何利用各种先进的工具和技术制作出精美的网页，非常适合缺少美术训练的网页设计人员。

国防工业出版社计算机编辑室

前 言

PHP(Hypertext Preprocessor)是一种服务器端嵌入式 HTML 语言，它的语法大致上与 C/C++、Perl、Java 相似，只要具备简单的程序设计观念便可以轻易地上手，其功能与现今 ASP 功能类似，可以快速的写出功能强大的网页。PHP 除可以做一般的网页功能外，并能提供像一般 CGI 的使用功能、cookies 的功能、文件上传、WWW 用户身份证明、GIF 动画等。

PHP 最初在公元 1994 年由 Rasmus Lerdorf 开始计划发展。于 1995 年 Personal Home Page Tools (PHP Tools) 开始对外发表第一个版本。在这早期的版本中，提供了访客留言本、访客计数器等简单的功能。随后在新的成员加入开发行列之后， 于 1995 年间，第二版的 PHP 问世。第二版定名为 PHP/FI(Form Interpreter)。PHP/FI 并加入了 mSQL 的支持，从此奠定了 PHP 在动态网页开发上的影响力。在 1996 年底，有一万五千个 Web 站台使用 PHP/FI；到 1997 年，使用 PHP/FI 的 Web 站台成长到超过五万个。

由于 PHP 4.0 跟 Apache 服务器紧密结合的特性；加上它不断的更新及新功能的加入；并且它几乎支持所有主流与非主流数据库；再以它高速的执行效率，使得 PHP 在 1999 年中的使用站点超过了十五万！它的原始码完全公开，在 Open Source 意识抬头的今天，它更是这方面的中流砥柱。不断地有新的函数库加入，以及不停地更新的活力，使得 PHP 无论在 UNIX 或是 Win32 的平台上都有很多新的功能。它提供丰富的函数，使得在程序设计方面有着更好的支持。

PHP 是完全免费的，可以从 PHP 官方站点（http://www.php.net）自由下载。PHP 遵守 GNU 公共许可证（GPL），你可以不受限制地获得源码，甚至可以从中加进自己需要的特色。

PHP 支持 Internet 开发的一些前沿技术，这些技术包括身份认证、XML、动态图像生成、WDDX、共享内存，以及动态 PDF 文件等等，不一而足。如果你还不满意的话，PHP 是很容易扩展的，所以只要你有编程能力，你尽可以自己大展身手。

PHP 在数据库方面的丰富支持，也是它迅速窜红的原因之一，它支持下列的数据库和数据表：

Adabas D、DBA、dBase、dbm、filePro、Informix、InterBase、mSQL、Microsoft SQL

Server、MySQL、Solid、Sybase、ODBC、Oracle 8、Oracle、PostgreSQL。

而在 Internet 上它也支持了相当多的通信协议 (protocol)，包括了与电子邮件相关的 IMAP、POP3；网管系统 SNMP；网络新闻 NNTP；账号共用 NIS；Internet 网的 HTTP 及 Apache 服务器；目录协议 LDAP 以及其他网络的相关函数。

除此之外，用 PHP 写出来的 Web 后端 CGI 程序，可以很轻易地移植到不同的作业平台上。例如，在系统负荷过高时，以 Linux 架起的网站，可以快速地将整个系统移到 SUN 工作站上，不用重新编译 CGI 程序。面对快速发展的 Internet，这是长期规划的最好选择。

在加入其他的模块之后，PHP 提供了更多样的支持，如：英文拼字检查、BC 高精确度计算、西方历法、PDF 文件格式、Hyperwave 服务器、图形处理、编码与解码功能、杂凑处理、WDDX 功能、qmail 与 vmailmgr 系统、压缩文件处理、XML 解析等。

本书由森林图书工作室主持编写，具体参加编写工作的有刘元高、黄建森、李春鹤、冯曙红、袁军、林世永、郑清初、黄重阳、刘浪、岑进华、郑国鸿、黄林、林振宁、岑进炎、康拥红、郑吉林、许晓春、陈良程、李庆、张劲松、李忠兰、黄玉华、殷岚、朱强、周雨昆、顾艺华、邹成福、李光、洪华、蒙文荣、梁任、梁德、邓冰、梁燕等。另外，姚兰、李劲同志还承担了本书的录入整理工作。本书在编写过程中，得到了国防工业出版社计算机编辑室的支持和帮助，在此表示诚挚的谢意。

目　录

一、Apache 服务器专用函数库 …… 1
二、数组处理函数库 …… 2
三、拼字检查函数库 …… 16
四、BC 高精确度函数库 …… 17
五、历法函数库 …… 19
六、CCVS API 函数 …… 22
七、Windows 下的 COM 支持函数 …… 23
八、类(Class)/对象(Object)函数 …… 24
九、ClibPDF 函数 …… 27
十、CURL 和客户端 URL 库函数 …… 41
十一、电子货币支付函数 …… 44
十二、DBA 函数库 …… 44
十三、日期和时间函数库 …… 48
十四、dBase 格式数据表函数库 …… 52
十五、DBM 类数据库函数库 …… 54
十六、目录管理函数库 …… 56
十七、DOM XML 函数库 …… 58
十八、出错处理和登录函数库 …… 58
十九、filePro 数据库函数库 …… 63
二十、文件系统函数库 …… 64
二十一、FDF (Forms Data Format) 函数库 …… 78
二十二、FTP 文件传输函数库 …… 81
二十三、函数操作函数库 …… 85
二十四、GNU 文本获取函数库 …… 89
二十五、HTTP 相关函数库 …… 90
二十六、Hyperwave 服务器函数库 …… 91
二十七、ICAP 函数库 …… 102
二十八、图形处理函数库 …… 104
二十九、IMAP、POP3 和 NNTP 函数库 …… 116
三十、Informix 数据库函数库 …… 131
三十一、InterBase 数据库函数库 …… 139

三十二、LDAP 目录协议函数库 …… 142
三十三、电子邮件函数库 …… 151
三十四、数学运算函数库 …… 152
三十五、MCAL 模块日历存取函数库 …… 158
三十六、Mcrypt 编码函数库 …… 165
三十七、Mhash 杂凑函数库 …… 174
三十八、SQL Server 数据库函数库 …… 175
三十九、杂项函数库 …… 179
四十、mSQL 数据库函数库 …… 184
四十一、MySQL 数据库函数库 …… 191
四十二、网络函数库 …… 201
四十三、ODBC 数据库连接函数库 …… 206
四十四、Oracle 8 数据库函数库 …… 215
四十五、Oracle 数据库函数库 …… 227
四十六、输出控制函数库 …… 231
四十七、PDF 格式文件函数库 …… 233
四十八、Verisign Payflow Pro functions …… 248
四十九、PHP 选项及相关信息函数库 …… 250
五十、POSIX 函数库 …… 257
五十一、PostgreSQL 数据库函数库 …… 261
五十二、程序执行功能函数库 …… 269
五十三、拼字检查函数库 …… 271
五十四、GNU Readline 函数库 …… 277
五十五、GNU 记录函数库 …… 278
五十六、规则表达式函数库(Perl 相容语法函数库) …… 279
五十七、常规表示法函数库(POSI 扩展函数库) …… 293
五十八、CORBA 卫星式客户端扩展函数库 …… 296
五十九、信号与共享内存函数库 …… 298
六十、Session 操作函数库 …… 299
六十一、共享内存函数库 …… 303
六十二、Shockwave Flash 文件编辑函数库 …… 306
六十三、SNMP 网管函数库 …… 317
六十四、Socket 函数库 …… 319
六十五、字串处理函数库 …… 323
六十六、Sybase 数据库函数库 …… 339
六十七、URL 处理函数库 …… 342
六十八、Variable 变量处理函数库 …… 343
六十九、Vmailmgr 邮件处理函数库 …… 349

七十、WDDX 函数库 …… 350
七十一、XML 分解函数库 …… 351
七十二、YAZ 函数库 …… 362
七十三、YP/NIS 函数库 …… 366
七十四、Zlib 压缩文件函数库 …… 368
附录 函数索引 …… 372

```
while (list( $ header, $ value) = each( $ headers)) {
    echo " $ header: $ value < br > \ n";
}
? >
```

【备注】:这个函数只有 PHP 以 Apache 服务器的模块(module)方式执行时方有效。

4. virtual

完成 Apache 服务器的子请求(sub - request)。

【语法】:int virtual(string filename);

【传回值】:整数

【函数种类】:PHP 系统功能

【内容说明】:这个函数等同于使用服务器端分解(SSI)的 shtml 功能。值得注意的是 virtual 所引入的程序必须要产生有效的 HTTP 文件头,最少要加入 Content - type 文件头、或者 Location 文件头、或者 Status 文件头。并且根据 HTTP 协议,咱文件头结束后尚须空一行。在 PHP 的程序实作时,可使用 include() 或 require() 这两个函数。

二、数组处理函数库

本函数库共有 47 个函数。这些函数能够采用多种方法实现数组的操作。数组是用于存储、管理和操作必不可少的变量组合。

本函数库中的函数支持一维和多维数组,当然他们也可由用户创建或者通过另外的一个函数来创建。其中有几个函数是专门用于从数据库查询生成数组的数据库操作函数,还有几个函数的返回值是数组。

【函数库目录】

1. array:建立一个新的数组。
2. array _ count _ values:统计数组中所有的元素值及其出现次数。
3. array _ diff:返回数组中不同的元素。
4. array _ flip:弹出数组中所有的值。
5. array _ intersect:数组求交集。
6. array _ keys:返回数组中的所有元素。
7. array _ merge:合并两个或者多个数组。
8. array _ merge _ recursive:两个或者多个数组的递归合并。
9. array _ multisort:多个数组和多维数组的排序。
10. arrau _ pad:填充数组达到指定的长度。
11. array _ pop:弹出并返回数组的最后一个元素。
12. array _ push:将一个或者多个元素推入数组末尾。
13. array _ rand:从数组中随机挑选出一个或者多个元素。
14. array _ reverse:返回一个数组的倒序排列。
15. array _ shift:弹出一个数组的首元素。
16. array _ slice:提取数组的一段。
17. array _ splice:替换数组的一部分。
18. array _ unique:剔除数组中相同的元素。
19. array _ unshift:将一个或者多个元素推入数组的首部。
20. array _ values:返回输入的数组中所有的值。
21. array _ walk:让用户自定义函数来处理数组中的每一个元素。
22. arsort:将数组的值由大到小排序。
23. asort:将数组的值由小到大排序。
24. compact:创建包含变量及其值的数组。
25. count:计算变量或数组中的元素个数。
26. current:返回数组中目前的元素。
27. each:返回数组中下一个元素的索引及值。
28. end:将数组的内部指针指到最后的元素。
29. extract:从数组中向变量表引入变量。
30. in _ array:判断一个值是否属于数组。
31. key:取得数组中的索引数据。
32. krsort:按索引的逆序排列数组。
33. ksort:将数组的元素依索引排序。
34. list:列出数组中元素的值。

一、Apache 服务器专用函数库

本函数库共有 4 个函数。

顾名思义，若 WEB 服务器不使用 Apache 服务器，则本函数库就派不上用场。

【函数库目录】

1. apache-lookup _ uri：获得所有的 URI 相关资料。

2. apache _ note：获得及设定 Apache 服务器的请求记录。

3. getallheaders：获得所有 HTTP 请求变量值。

4. virtual：完成 Apache 服务器的子请求(sub-request)。

1. apache _ lookup _ uri

获得所有的 URI 相关资料。

【语法】：class apache _ lookup _ uri (string filename)；

【传回值】：类别

【函数种类】：PHP 系统功能

【内容说明】：这个函数将给定 URI 的重要相关资料传回到类别变量中。传回的变量包括下列的属性：

status
the _ request
status _ line
method
content _ type
handler
uri
filename
path _ info
args
boundary
no _ cache
no _ local _ copy
allowed
send _ bodyct
bytes _ sent
byterange
clength
unparsed _ uri
mtime
request _ time

【注意】：Apache _ lookup _ uri()仅在 Apache 模式的 PHP 中才能够使用。

2. apache _ note

获得及设定 Apache 服务器的请求记录。

【语法】：string apache _ note(string note _ name, string [note _ value])；

【传回值】：字符串

【函数种类】：PHP 系统功能

【内容说明】：这个 Apache 服务器特有的函数能设定及取得请求记录表的值。若只代入一个参数，则传回目前纪录的 note _ name 值。若代入两个参数，则传回先前的 note _ name 值，并将 note _ name 设为新的 note _ value 值。

3. getallheaders

获得所有 HTTP 变量值。

【语法】：array getallheaders(void)；

【传回值】：数组

【函数种类】：PHP 系统功能

【内容说明】：使用本项功能时不需代入任何参数值，传回的是所有 HTTP 变量值，并使用组合的数组传回。

【注意】：你也可以从运行环境中得到普通的 CGI 值，这在是否将 PHP 作为 Apache 模块方式时都有效。使用 phpinfo()函数可以得到这样定义的所有环境变量。

【使用范例】：下例列出所有的 HTTP 变量。

```
<? php
$ headers = getallheaders();
```

35. natsort:使用“自然顺序”运算法则对数组进行排序。

36. next:将数组的内部指针向后移动。

37. pos:返回数组目前的元素。

38. prev:将数组的内部指针往前移动。

39. range:建立一个整数范围的数组。

40. reset:将数组的指针指到数组第一个元素。

41. rsort:将数组的值由大到小排序。

42. shuffle:将数组的顺序弄混。

43. sizeof:获知数组的大小。

44. sort:将数组排序。

45. uasort:将数组依用户自定的函数排序。

46. uksort:将数组的索引依用户自定的函数排序。

47. usort:将数组的值依用户自定的函数排序。

1. array

建立一个新的数组。

【语法】:array array(...);

【返回值】:数组

【函数类型】:数据处理

【内容说明】

返回的参数是数组形态。参数可以是带有 = > 运算子的索引。注意 array() 其实不是一个正规的函数,它主要是要用来表示数组。

【使用范例】:下面范例用显示如何建立一个二维数组,如何指定联合数组的键值,及如何略过和继续数组中的数字索引。

```
$ fruits = array(
    "fruits"  = > array("a" = > "orange","b" = > "banana","c" = > "apple"),
    "numbers" = > array(1, 2, 3, 4, 5, 6),
    "holes"   = > array("first", 5 = > "second", "third")
);
```

【参考】:list()

2. array _ count _ values

统计数组中所有元素的值及其出现次数。

【语法】:array array _ count _ values (array input);

【返回值】:数组

【函数类型】:数据处理

【内容说明】:本函数把输入的数组中字符的值作为关键字,将它们出现的次数作为值。

【使用范例】

```
$ array = array (1, "hello", 1, "world", "hello");
array _ count _ values ( $ array); // returns array (1 = > 2, "hello" = > 2, "world" = > 1)
```

3. array _ diff

返回数组中不同的元素。

【语法】:array array _ diff (array array1, array array2 [, array ...]);

【返回值】:数组

【函数类型】:数据处理

【内容说明】:给出数组 array1 中不同于其他数组的元素名。

【使用范列】:

```
$ array1 = array ("a" = > "green", "red", "blue");
$ array2 = array ("b" = > "green", "yellow", "red");
$ result = array _ diff ( $ array1, $ array2);
```

上述表达式的返回值是 $ result 的值为 array ("blue")。

4. array _ flip

弹出数组中所有的值。

【语法】:array array _ flip (array trans);

【返回值】:数组

【函数类型】:数据处理

【内容说明】:按弹出的顺序返回某个数组中的某个元素。

【使用范例】:

```
$ trans = array _ flip ( $ trans);
$ original = strtr ( $ str, $ trans);
```

5. array _ intersect

数组求交集。

【语法】:array array _ intersect (array array1, array array2 [, array ...]);

【返回值】:数组

【函数类型】:数据处理

【内容说明】:给出数组 array1 中与其他的数组中的公共元素。

【使用范列】:

```
$ array1 = array ("a" = > "green", "red", "blue");
$ array2 = array ("b" = > "green", "yellow", "red");
$ result = array _ intersect ( $ array1, $ array2);
```

上述程序段返回的值是 $ result 的值为 array ("a" = > "green", "red")。

【参考】:array _ diff()

6. array _ keys

返回数组中的所有元素。

【语法】:array array _ keys (array input [, mixed search _ value]);

【返回值】:数组

【函数类型】:数据整理

【内容说明】:本函数从 input 数组中给出所有的键值、数值和字符。如果定义了选项 search _ value,则将只返回该选项定义的选项。否则所有通过 input 数组录入的键值都将返回。

【使用范例】:

范例一:Array _ keys()的使用范例。

```
$ array = array (0 = > 100, "color" = > "red");
array _ keys ( $ array);        // returns array (0, "color")
$ array = array ("blue", "red", "green", "blue", "blue");
array _ keys ( $ array, "blue");  // returns array (0, 3, 4)
```

【注意】:该函数添加在 PHP 4 中,下面的例子是为 PHP 3 的用户准备的。

范例二:PHP 3 的 array _ keys()函数补充:

```
function array _ keys ( $ arr, $ term = "") {
    $ t = array();
    while (list( $ k, $ v) = each( $ arr)) {
        if ( $ term && $ v ! = $ term)
            continue;
            $ t[] = $ k;
        }
        return $ t;
}
```

【参考】:array _ values()

7. array _ merge

合并两个或者多个数组。

【语法】:array array _ merge (array array1, array array2 [, array ...]);

【返回值】:数组

【函数类型】:数据处理

【内容说明】:本函数合并两个或者多个数组,使得数组的元素被添加在原数组之后,并返回合并后的数组。

如果并入的数组中含有与前一数组同样的字符键值,后者的值将覆盖前一数组的值。相反,如果数组中含有的不是相同的数值元素,则后者将不会覆盖前者,只是添加在之后。

【使用范例】:

```
$ array1 = array ("color" = > "red", 2, 4);
$ array2 = array ("a", "b", "color" = > "green", "shape" = > "trapezoid", 4);
array _ merge ( $ array1, $ array2);
```

返回的数组结果是:array("color" = > "green", 2, 4, "a", "b", "shape" = > "trapezoid", 4)。

【参考】:array _ merge _ recursive()

8. array _ merge _ recursive

两个或者多个数组的递归合并。

【语法】:array array _ merge _ recursive (array array1, array array2 [, array ...]);

【返回值】:数组

【函数类型】:数据处理

【内容说明】:本函数将两个或者多个数组的元素合并,使得一个数组的元素被添加到另一个数组元素之后,并返回合并的结果。

如果加入的数组与原来的数组具有相同的字符值,则这些值将被合并形成一个数组,这一过程将被递归地处理,所以如果其中一个元素本身就是一个数组,该函数将采用相应的元素并入另一个数组。如果数组之间具有相同的数值元素,后者的数值将不会覆盖原来的值,只是添加于后。

【使用范例】:

```
$ ar1 = array ("color" = > array ("favorite" = > "red"), 5);
$ ar2 = array (10, "color" = > array ("favorite" = > "green", "blue"));
$ result = array _ merge _ recursive ( $ ar1, $ ar2);
```

上述函数的返回数组是:array ("color" = > array ("favorite" = > array ("red", "green"), "blue"), 5, 10)。

【参考】:array _ merge()

9. array _ multisort

多个数组和多维数组的排序。

【语法】: bool array _ multisort (array ar1 [, mixed arg [, mixed ... [, array ...]]]);

【返回值】:布尔值

【函数类型】:数据处理

【内容说明】:本函数用于同时为一个或几个数组排序,排序的根据是多维中的一维。排序后保持数组元素的索引不变。

输入的数组作为进行排序的一列,这类似于SQL ORDER BY 子句。第一个数组是排序的主索引数组。该数组中当行值出现相同时将按下一个数组进行排序,并依此类推。

该函数的变量结构多少有点与众不同,但是使用非常灵活。其中的第一个变量必须是数组。其余的变量则可以是数组或者如下列表中的排序方式:

1)排序方式标志:

SORT _ ASC:升序排列

SORT _ DESC:降序排列

2)排序类型标志:

SORT _ REGULAR:一般方式逐项比较

SORT _ NUMERIC:按照数值方式逐项比较

SORT _ STRING:按照字符方式逐项比较

同一个数组的排序不能使用同类的两种排序方式,一个数组变量后面定义的排序方式和排序类型只适用于该数组。各个排序的参数的默认值是SORT_ ASC 和 SORT _ REGULAR。

排序正常完成时返回 True,否则返回 False。

【使用范例】:

例一:

```
$ ar1 = array ("10", 100, 100, "a");
$ ar2 = array (1, 3, "2", 1);
array _ multisort ( $ ar1, $ ar2);
```

该例中,排序之后的结果是第一个数组的值为 10, "a", 100, 100,第二个数组的值为 1, 1, 2, "3"。两个数组中相同的元素都同样被排序。

例二:

```
$ ar = array (array ("10", 100, 100, "a"), array (1, 3, "2", 1));
array _ multisort ( $ ar[0], SORT _ ASC, SORT _ STRING,
                    $ ar[1], SORT _ NUMERIC, SORT _ DESC);
```

该例中,第一个数组排序之后的结果是 10, 100, 100, "a"(按照字符方式的升序排序),第二个数组的结果为 1, 3, "2", 1(按照数值方式的降序排列)。

10. arrau _ pad

填充数组达到指定的长度

【语法】:array array _ pad (array input, int pad _ size, mixed pad _ value);

【返回值】:数组

【函数类型】:数据处理

【内容说明】:使用 pad _ value 填充数组 input 达到指定的长度 pad _ size。如果 pad _ size 为正,从右端开始填充数组,如果为负表示从左端开始

填充数组。如果 pad _ size 的绝对值小于或等于 input 数组的长度，则不会进行填充。

【使用范例】：

```
$ input = array (12, 10, 9);

$ result = array _ pad ( $ input, 5, 0);
// result is array (12, 10, 9, 0, 0)

$ result = array _ pad ( $ input, -7, -1);
// result is array ( -1, -1, -1, -1, 12, 10, 9)

$ result = array _ pad ( $ input, 2, "noop");
// not padded
```

11. array _ pop

弹出并返回数组的最后一个元素。

【语法】：mixed array _ pop (array array);

【返回值】：混合形态数据

【函数类型】：数据处理

【内容说明】：弹出数组 array 的最后一个元素，使得数组元素减少一个。

【使用范例】：

```
$ stack = array ("orange", "apple", "raspberry");
$ fruit = array _ pop ( $ stack);
```

函数作用之后，数组 $ stack 只有两个元素，即"orange"和"apple"，数组 $ fruit 则将具有一个元素"raspberry"。

【参考】：array _ push()，array _ shift()，array _ unshift()

12. array _ push

将一个或者多个元素推入数组末尾。

【语法】：int array _ push (array array, mixed var [, mixed ...]);

【返回值】：整数

【函数类型】：数据处理

【内容说明】：将数组当作堆栈来处理，将给定的元素推入该堆栈 array 的尾部，这样数组的长度将增加推入元素的个数。该函数的效果与 $ array [] = $ var 函数对每个 var 循环效果一样。

返回值是新数组的长度值。

【使用范例】：

```
$ stack = array (1, 2);
array _ push ( $ stack, "+", 3);
```

上述函数作用之后，$ stack 具有四个元素：1，2，“+”和 3。

【参考】：array _ pop()，array _ shift()，array _ unshift()

13. array _ rand

从数组中随机挑选出一个或者多个元素。

【语法】：mixed array _ rand (array input [, int num _ req]);

【返回值】：混合型数据

【函数类型】：数据处理

【内容说明】：如果想随机地从数组中选出一个或者多个元素，此时该函数十分有用。用户需要定义数组 input 和需要挑选的元素个数 num _ req。如果缺省，默认值是 1。

如果只是挑选一个元素，array _ rand()函数返回随机元素的索引字，否则将返回多个元素的索引字的集合。这样处理的目的是使读者既可以挑选索引字，也可以挑选元素值。

别忘了使用 srand()得到任意的随机数。

【使用范例】：

```
srand ((double) microtime() * 10000000);
$ input = array ("Neo", "Morpheus", "Trinity", "Cypher", "Tank");
$ rand _ keys = array _ rand ( $ input, 2);
print $ input[ $ rand _ keys[0]]." \ n";
print $ input[ $ rand _ keys[1]]." \ n";
```

14. array _ reverse

返回一个数组的倒序排列。

【语法】：array array _ reverse (array array);

【返回值】：数组

【函数类型】：数据处理

【内容说明】：本函数将输入的函数 array 按单元的逆序排列，并返回逆序排列的结果到新的数组。

【使用范例】:

```
$ input = array ("php", 4.0, array ("green", "red"));
$ result = array _ reverse ( $ input);
```

上述函数段运行之后数组 $ result 的值为 array (array ("green", "red"), 4.0, "php")。

15. array _ shift

弹出一个数组的首元素。

【语法】:mixed array _ shift (array array);

【返回值】:混合型数据

【函数类型】:数据处理

【内容说明】:将数组中的首元素弹出,后面的元素依次前移,数组长度减一。

【使用范例】:

```
$ args = array ("- v", "- f");
$ opt = array _ shift ( $ args);
```

上述程序段执行之后数组 $ args 将只剩下一个元素"- f",数组 $ opt 将具有一个元素"- v"。

【参考】:array _ unshift(), array _ push(), array _ pop()

16. array _ slice

提取数组的一段。

【语法】:array array _ slice (array array, int offset [, int length]);

【返回值】:数组

【函数类型】:数据处理

【内容说明】:该函数返回数组 array 中由 offset 和 length 定义的元素段。

如果 offset 为正,被提取的序列将从 array 中位置为 offset 开始,如果 offset 为负,该序列将从数组的尾部开始倒推计算位置。

如果 length 给定且为正,则提取出的序列将具有由它定义的个数,如果 length 给定且为负,则提取时将在数组尾部剩下 length 个元素后停止,如果忽略 length 的值,将 offset 之后知道数组尾部的所有元素都提取出来。

【使用范例】:

```
$ input = array ("a", "b", "c", "d", "e");
$ output = array _ slice ( $ input, 2); // returns "c", "d", and "e"
$ output = array _ slice ( $ input, 2, - 1); // returns "c", "d"
$ output = array _ slice ( $ input, - 2, 1); // returns "d"
$ output = array _ slice ( $ input, 0, 3); // returns "a", "b", and "c"
```

【参考】:array _ splice()

17. array _ splice

替换数组的一部分。

【语法】:array array _ splice (array input, int offset [, int length [, array replacement]]);

【返回值】:数组

【函数类型】:数据处理

【内容说明】:对输入的数组 array 中,把由 offset 和 length 指定部分的元素去掉,如果提供了替换数组 array,则将该部分元素替换。

如果 offset 为正,则去掉的部分将从输入数组 array 的开始第一个元素开始,如果为负则将从该数组的末尾开始起算。

如果 length 忽略,则从 offset 定义的位置起去掉数组中的全部元素。如果 length 给定且为正,则提取出的序列将具有由它定义的个数,如果 length 给定且为负,则提取时将在数组尾部剩下 length 个元素后停止。建议:如果要去掉数组中从 offset 开始后的所有的元素,当替换数组也指定时,可以使用 count($ input)得到 length 的值。

如果给定替换数组的值,去掉的单元将由该数组中的元素依次替换。如果 offset 和 length 的取值使得任何元素都不被去掉,则替换数组将从 offset 开始的位置添加到原数组上。建议:如果替换矩阵本身只有一个元素,不必使用函数 array()来定义它,除非该元素本身就是数组。

如下的操作是等价的操作:

```
array _ push ( $ input, $ x, $ y)
array _ splice ( $ input, count ( $ input), 0, array ( $ x, $ y))
array _ pop ( $ input)                    array _ splice ( $ input, - 1)
```

array _ shift ($ input)　　array _ splice ($ input, 0, 1)

array _ unshift ($ input, $ x, $ y)　　array _ splice ($ input, 0, 0, array ($ x, $ y))

$ a[$ x] = $ y　　array _ splice ($ input, $ x, 1, $ y)

返回的数组由去掉的元素组成。

【使用范例】:

```
$ input = array ("red", "green", "blue", "yellow");

array _ splice ( $ input, 2);          // $ input is now array ("red", "green")
array _ splice ( $ input, 1, -1);  // $ input is now array ("red", "yellow")
array _ splice ( $ input, 1, count( $ input), "orange");
                                       // $ input is now array ("red", "orange")
array _ splice ( $ input, -1, 1, array("black", "maroon"));
                                       // $ input is now array ("red", "green",
//                 "blue", "black", "maroon")
```

【参考】:array _ slice

18. array _ unique

剔除数组中相同的元素。

【语法】:array array _ unique (array array);

【返回值】:数组

【函数类型】:数据处理

【内容说明】:将输入的数组 array 中重复元素删除,注意他还保留索引字。

【使用范例】:

```
$ input = array ("a" = > "green", "red", "b" = > "green", "blue", "red");
$ result = array _ unique ( $ input);
```

上述程序段执行之后,数组 $ result 的结果是("a" = > "green", "red", "blue")。

19. array _ unshift

将一个或者多个元素推入数组的首部。

【语法】:int array _ unshift (array array, mixed var [, mixed ...]);

【返回值】:整数

【函数类型】:数据处理

【内容说明】:将欲添加的元素推入数组的首部,加入的元素应当成一个整体来考虑,这样加入之后他们将保持顺序不变。

返回的值是数组 array 中的元素个数。

【使用范例】:

```
$ queue = array ("p1", "p3");
array _ unshift ( $ queue, "p4", "p5", "p6");
```

上述程序段执行之后数组 $ queue 具有 5 个元素:"p4", "p5", "p6", "p1", and "p3"。

【参考】:array _ shift(), array _ push(), array _ pop()

20. array _ values

返回输入的数组中所有的值。

【语法】:array array _ values (array input);

【返回值】:数组

【函数类型】:数据处理

【内容说明】:返回输入的数组中的所有的值。

【使用范例】:

例一:Array _ values()使用实例。

```
$ array = array ("size" = > "XL", "color" = > "gold");
array _ values ( $ array);      // returns array ("XL", "gold")
```

例二:PHP3 用户使用实例。

```
function array _ values ( $ arr) {
    $ t = array();
    while (list( $ k, $ v) = each ( $ arr)) {
        $ t[] = $ v;
        return $ t;
    }
}
```

21. array _ walk

让用户自定义函数来处理数组中的每一个元

素。

【语法】:int array _ walk(array arr, string func);

【返回值】:整数

【函数类型】:数据处理

【内容说明】:此函数使每个数组元素 arr 依序与函数名称 func 相对应。元素传送到函数 func 的第一个参数,若参数超过一个,则每次都会有警告数据。要处理警告数据,可在本函数前面加上'@'字符(变成 @array _ walk);或是使用 error _ reporting 函数。

【注意】:用户自定义函数 func 会将数组元素 arr 依序代入,所以任何对元素所做的改变都会影响到数组本身。在 PHP 4 中由于 array _ walk()并不默认地重置数组,所以需要调用 reset()函数。

【使用范例】:

```
<?
$fruits = array("d" = >"lemon","a" = >"orange","b" = >"banana","c" = >"apple");
function test _ alter( $item1 ) {
    $item1 = 'bogus';
}
function test _ print( $item2 ) {
    echo " $item2 < br > \ n";
}
array _ walk( $fruits, 'test _ print' );
array _ walk( $fruits, 'test _ alter' );
array _ walk( $fruits, 'test _ print' );
? >
```

【参考】:each() list()

22. arsort

将数组的值由大到小排序。

【语法】:void arsort(array array);

【返回值】:无

【函数类型】:数据处理

【内容说明】:这个函数将数组的值重新排序,由大至小排列。数组的索引亦跟着值的顺序而变动。当您在程序中需要重新整理数组值的顺序时,就可以使用这个函数。

【使用范例】:

```
<?
$fruits = array("d" = >"lemon","a" = >"orange","b" = >"banana","c" = >"apple");
arsort( $fruits);
for( reset( $fruits); $key = key( $fruits); next( $fruits)) {
    echo "fruits[ $key] = ". $fruits[ $key]."\ n";
}
? >
```

上面的范例返回的结果为:

fruits[a] = orange

fruits[d] = lemon

fruits[b] = banana

fruits[c] = apple。

我们可以看到水果名(数组值)已按英文字母的顺序由 z 往 a 重新排序,而索引亦跟着值变动。

你可以通过使用参数选项 sort _ flags 来定义排序的方式,详细内容参见 sort()。

【参考】:asort() rsort() ksort() sort()

23. asort

将数组的值由小到大排序。

【语法】:void asort(array array);

【返回值】:无

【函数类型】:数据处理

【内容说明】:这个函数将数组的值重新排序,由小至大排列。数组的索引亦跟着值的顺序而变动。当您在程序中需要重新整理数组值的顺序时,就可以使用这个函数。

【使用范例】:

```
<?
$fruits = array("d" = >"lemon","a" = >"orange","b" = >"banana","c" = >"apple");
asort( $fruits);
for( reset( $fruits); $key = key( $fruits); next( $fruits)) {
    echo "fruits[ $key] = ". $fruits[ $key]."\ n";
}
? >
```