

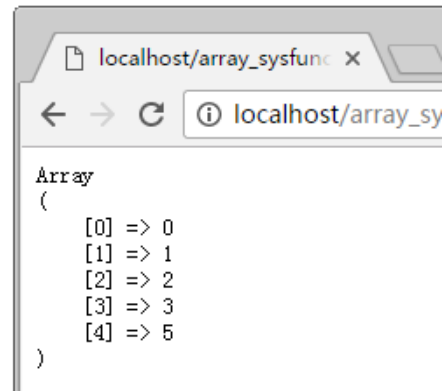
数组的相关函数

1) 排序函数：对数组元素进行排序，都是按照 ASCII 码进行比较，可以进行英文比较

sort(): 顺序排序（下标重排）

rsort(): 逆序排序

```
5 //排序函数
7 $arr = array(3,1,5,2,0);
3 echo '<pre>';
3
3 $sort_arr = sort($arr);
1 print_r($arr);
```

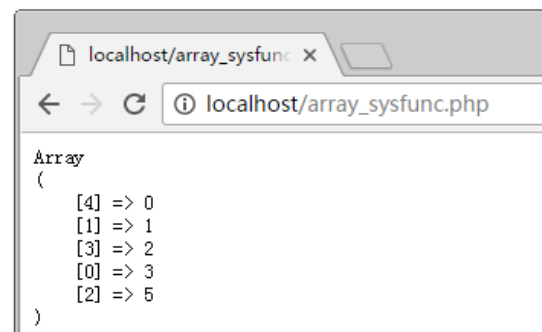


asort(): 顺序排序（下标保留）

arsort(): 逆序排序

```
//排序函数
$arr = array(3,1,5,2,0);
echo '<pre>';

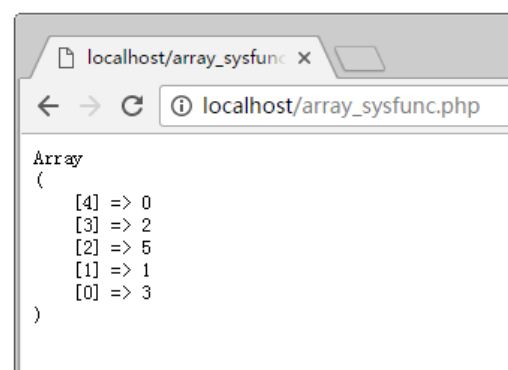
//$sort_arr = sort($arr);
asort($arr);
print_r($arr);
```



ksort(): 顺序排序：按照键名（下标）

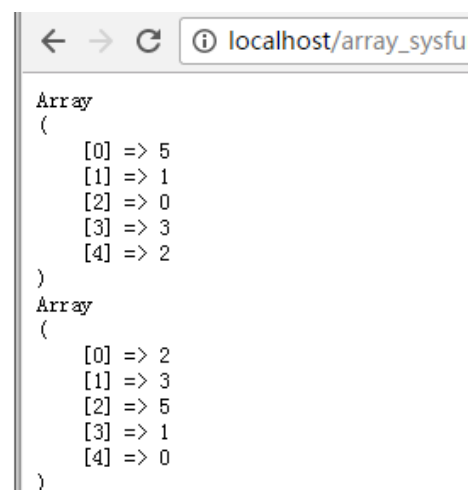
krsort(): 逆序排序

```
6 //排序函数
7 $arr = array(3,1,5,2,0);
8 echo '<pre>';
9
10 // $sort_arr = sort($arr);
11 // asort($arr);
12
13
14 krsort($arr);
15 print_r($arr);
```



shuffle(): 随机打乱数组元素，数组下标会重排

```
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000
```

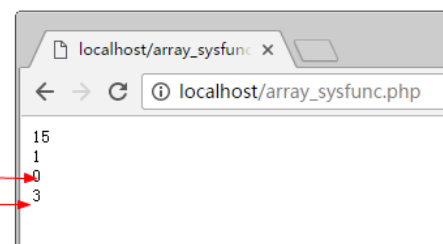


2) 指针函数

reset(): 重置指针，将数组指针回到首位

end(): 重置指针，将数组指针指向最后一个元素

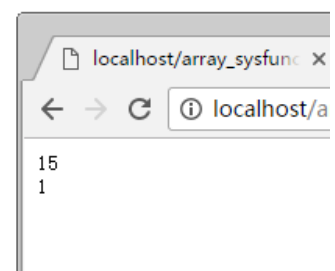
```
$arr = array(3,1,5,2,0);  
//指针函数  
//echo current($arr), '<br/>';  
//echo key($arr), '<br/>';  
  
echo next($arr), next($arr), '<br/>';  
echo prev($arr), '<br/>';  
  
echo end($arr), '<br/>';  
echo reset($arr), '<br/>';
```



next(): 指针下移，取得下一个元素的值

prev(): 指针上移，取得上一个元素的值

```
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000
```



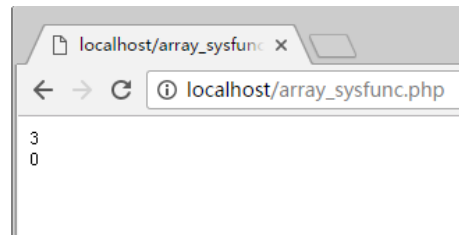
current(): 获取当前指针对应的元素值

key(): 获取当前指针对应的下标值

```

$arr = array(3,1,5,2,0);
//指针函数
echo current($arr), '<br/>';
echo key($arr), '<br/>';

```



注意事项: **next** 和 **prev** 会移动指针, 有可能导致指针移动到最前或者最后 (离开数组), 导致数组不能使用, 通过 **next** 和 **prev** 不能回到真确的指针位置。只能通过 **end** 或者 **reset** 进行指针重置

3) 其他函数

count(): 统计数组中元素的数量

array_push(): 往数组中加入一个元素 (数组后面)

array_pop(): 从数组中取出一个元素 (数组后面)

array_shift(): 从数组中取出一个元素 (数组前面)

array_unshift(): 从数组中加入一个元素 (数组前面)

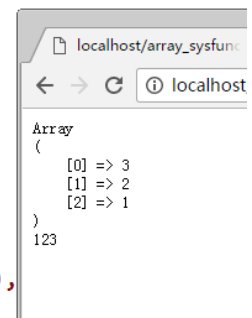
PHP 模拟数据结构:

栈: 压栈, 先进去后出来 (FILO)

```

19 //栈: 先压栈后出栈: 都是从一端出来
20 //前面: array_shift/array_unshift
21 //后面: array_push/array_pop
22 //压栈
23 array_push($arr,3);
24 array_push($arr,2);
25 array_push($arr,1);
26 print_r($arr);
27 //出栈
28 echo array_pop($arr),array_pop($arr),array_pop($arr),

```



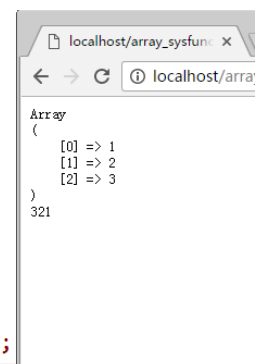
队列: 排队, 先进去的先出去 (FIFO)

```

//队列: 先排队, 先出来, 一端进, 另外一端出
//后进前出: array_push/array_shift
//前进后出: array_unshift/array_pop
$arr = array();

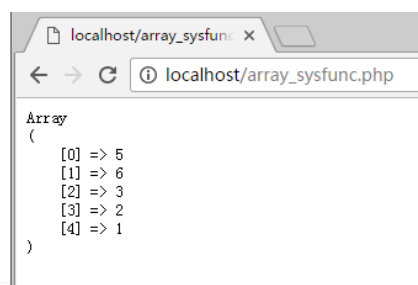
//入队
array_unshift($arr,3);
array_unshift($arr,2);
array_unshift($arr,1);
print_r($arr);
//出队
echo array_pop($arr),array_pop($arr),array_pop($arr), '<br/>';

```



array_reverse(): 数组元素反过来

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



in_array(): 判断一个元素在数组中是否存在

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

var_dump(in_array(6,$arr));
var_dump(in_array(100,$arr));
```

```
bool(true)
bool(false)
```

array_keys(): 获取一个数组的所有下标, 返回一个索引数组

array_values(): 获取一个数组的所有值, 返回一个索引数组

```
$arr = array(1,2,3,6,5);
//print_r(array_reverse($arr));

//var_dump(in_array(6,$arr));
//var_dump(in_array(100,$arr));

print_r(array_keys($arr));
print_r(array_values($arr));
```

```
Array
(
    [0] => 0
    [1] => 1
    [2] => 2
    [3] => 3
    [4] => 4
)

Array
(
    [0] => 1
    [1] => 2
    [2] => 3
    [3] => 6
    [4] => 5
)
```

编程思想

编程思想: 如何利用数学模式, 来解决对应的需求问题; 然后利用代码实现对应的数据模型 (逻辑)。

算法: 使用代码实现对应的数学模型, 从而解决对应的业务问题。

递推算法

递推算法是一种简单的算法, 即通过已知条件, 利用特定关系得出中间推论, 直至得到结果的算法。递推算法分为顺推和逆推两种。

顺推：通过最简单的条件（已知），然后逐步推演结果

逆推：通过结果找到规律，然后推到已知条件

斐波那契数列：1 1 2 3 5 8 13 ...，通常需求：请求得指定位置 N 所对应的值是多少

找规律：

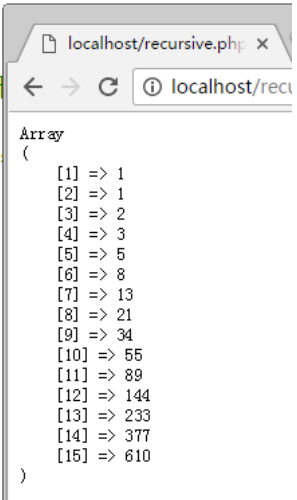
- 1、第一个数是 1
- 2、第二个数也是 1
- 3、从第三位开始：属于前两个数的和

代码解决思路：

- 1、如果数字位置为 1 和 2，结果都是 1
- 2、从第三个开始，想办法得到前两个的结果，就可以得到

终极解决办法：想办法把要求的位置之前的所有的值都列出来，那么要求的数就可以通过前两个之和计算出来：使用数组存储所有结果即可。

```
3 //递推思想（算法）
4
5 //需求：规律1 1 2 3 5 ...
6 //求出指定位数对应的值
7
8 //已知条件：第一个和第二个数都为1，第三个开始
9 $f[1] = 1;
10 $f[2] = 1; //如果想要第一个或者第二个结果
11
12 $des = 15;
13 for($i = 3;$i <= $des;$i++){
14
15     $f[$i] = $f[$i-1] + $f[$i-2];
16 }
17
18 //查看
19 echo '<pre>';
20 print_r($f);
```

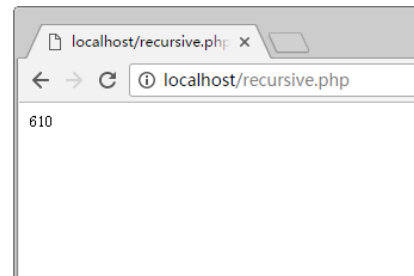


递推算法求斐波那契数列：

```

21
22     function my_recursive($des){
23         //判断: 如果为第一个或者第二个
24         if($des == 1 || $des == 2) return 1;
25
26         //开始计算
27         $f[1] = 1;
28         $f[2] = 1; //如果想要第一个或者第二个结果, 那么可以直接给出
29
30         for($i = 3; $i <= $des; $i++){
31
32             $f[$i] = $f[$i-1] + $f[$i-2];
33         }
34
35         //返回最后一个位置的结果
36         return $f[$des];
37     }
38
39     echo my_recursive(15);

```



编程思想

递归算法

递归算法是把问题转化为规模缩小了的同类问题的子问题。然后递归调用函数（或过程）来表示问题的解。

- 1、 简化问题：找到最优子问题（不能再小）
- 2、 函数自己调用自己

斐波那契数列：1 1 2 3 5 8 13 ...

需求：求指定位置的数列的值

规律：第一个和第二个为 1，从第三个开始为前两个之后

$$F(N) = F(N-1) + F(N-2);$$

$$F(N-1) = F(N-2) + F(N-3);$$

...

$$F(2) = F(1) = 1;$$

递归思想中：有两个非常重要的点

递归点：发现当前问题可以有解决当期问题的函数，去解决规模比当前小一点的问题来解决

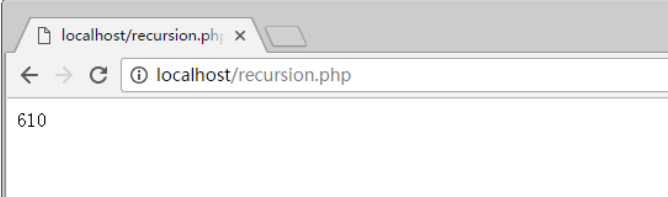
$$F(N) = F(N-1) + F(N-2);$$

递归出口：当问题解决的时候，已经到达（必须有）最优子问题，不能再次调用函数

如果一个函数递归调用自己而没有递归出口：就是死循环

递归的本质是函数调用函数：一个函数需要开辟一块内存空间，递归会出现同时调用 N 多个函数（自己）：递归的本质是利用空间换时间

```
5
6 //递归一定有函数
7 function recursion($n){
8     //递归出口
9     if($n == 1 || $n == 2) return 1;
10
11     //递归点：求N的值，与求N-1的值一模一样，只是N-1的规模比N小
12     return recursion($n - 1) + recursion($n - 2);
13 }
14
15 //调用
16 echo recursion(15);
```

A screenshot of a web browser window. The address bar shows 'localhost/recursion.php'. The page content displays the number '610', which is the result of the recursive function call recursion(15) shown in the code block to the left.

数组排序算法

冒泡排序

冒泡排序（Bubble Sort），是一种计算机科学领域的较简单的排序算法。它重复地走访过要排序的数列，一次比较两个元素，如果他们的顺序错误就把他们交换过来。走访数列的工作是重复地进行直到没有再需要交换，也就是说该数列已经排序完成。

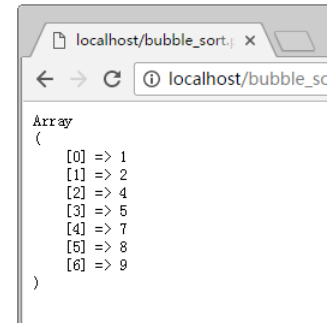
冒泡排序的算法思路：

- 1) 比较相邻的元素。如果第一个比第二个大，就交换他们两个。
- 2) 对每一对相邻元素作同样的工作，从开始第一对到结尾的最后一对。在这一点，最后的元素应该会是最大的数。
- 3) 针对所有的元素重复以上的步骤，除了最后一个。
- 4) 持续每次对越来越少的元素重复上面的步骤，直到没有任何一对数字需要比较。

```

5
6     $arr = array(1,4,2,9,7,5,8);
7
8     //2、 想办法让下面可以每次找出最大值的代码重复执行
9     for($i = 0,$len = count($arr);$i < $len;$i++){
10
11         //1、 想办法将最大的值放到最右边去
12         for($j = 0; $j < $len - 1 - $i;$j++){
13             //判断：两两相比
14             if($arr[$j] > $arr[$j+1]){
15                 //左边比右边大：交换
16                 $temp = $arr[$j];
17                 $arr[$j] = $arr[$j+1];
18                 $arr[$j+1] = $temp;
19             }
20         }
21     }
22 }

```



选择排序

选择排序（Selection sort）是一种简单直观的排序算法。它的工作原理是每一次从待排序的数据元素中选出最小（或最大）的一个元素，存放在序列的起始位置，直到全部待排序的数据元素排完。选择排序是不稳定的排序方法（比如序列[5， 5， 3]第一次就将第一个[5]与[3]交换，导致第一个5挪动到第二个5后面）。

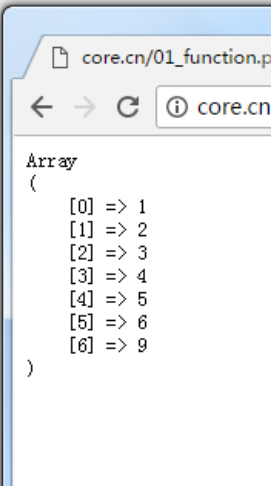
选择排序的算法思路：

- 1) 假设第一个元素为最小元素，记下下标。
- 2) 寻找右侧剩余的元素，如果有更小的，重新记下最新的下标。
- 3) 如果有新的最小的，交换两个元素。
- 4) 往右重复以上步骤，直到元素本身是最后一个。


```

1 <?php
2 //数组排序算法：选择排序
3 $arr = array(1,5,2,9,6,3,4);
4 //1、 确定要交换多少次：一次只能找到一个最小的，需要找数组长度
5 for($i = 0,$len = count($arr);$i < $len;$i++){
6     //2、假设当前第一个已经排好序
7     $min = $i; //当前第一个数是最小的
8     //3、拿该最小的取比较剩余的其他
9     for($j = $i + 1;$j < $len;$j++){
10        //4、比较：比较当前元素与选定的最小的元素
11        if($arr[$j] < $arr[$min]){
12            //说明当前指定的$min不合适
13            $min = $j;
14        }
15    }
16
17    //5、交换当前选定的值与实际最小的元素值
18    if($min != $i){
19        $temp = $arr[$i];
20        $arr[$i] = $arr[$min];
21        $arr[$min] = $temp;
22    }
23 }
24 echo '<pre>';
25 print_r($arr);

```



插入排序

插入排序（Insert sort），插入排序的基本操作就是将一个数据插入到已经排好序的有序数据中，从而得到一个新的、个数加一的有序数据，算法适用于少量数据的排序，是稳定的排序方法。插入算法把要排序的数组分成两部分：第一部分包含了这个数组的所有元素，但将最后一个元素除外（让数组多一个空间才有插入的位置），而第二部分就只包含这一个元素（即待插入元素）。在第一部分排序完成后，再将这个最后元素插入到已排好序的第一部分中。

插入排序的基本思想是：每步将一个待排序的纪录，按其关键码值的大小插入前面已经排序的文件中适当位置上，直到全部插入完为止。

插入排序的算法思路：

- 1) 设置监视哨 $r[0]$ ，将待插入纪录的值赋值给 $r[0]$ ；
- 2) 设置开始查找的位置 j ；
- 3) 在数组中进行搜索，搜索中将第 j 个纪录后移，直至 $r[0].key \geq r[j].key$ 为止；
- 4) 将 $r[0]$ 插入 $r[j+1]$ 的位置上。

- 1、 认定第一个元素已经排好序；
- 2、 取出第二个元素，作为待插入数据；
- 3、 与已经排好序的数组的最右侧元素开始进行比较
- 4、 如果后面的小于前面的：说明前面已经排好序的那个数组元素不在对的位置（向后移一

- 个)，然后让新的元素填充进去（继续向前比：高级）
- 5、 重复前面的步骤：直到当前元素插入到对位置；
 - 6、 重复以上步骤，直到所有的数组元素都插入到对的位置。

```
5     $arr = array(4,2,6,8,9,5);
6
7     //1、 确定要插入多少回（假设一个数组一次性插入到对的位置，同时第一个位置是
8     for($i = 1,$len = count($arr);$i < $len;$i++){
9         //2、 取出当前要插入的元素的值
10        $temp = $arr[$i];
11        //3、 让该数据与前面已经排好序的数组元素重复比较
12        for($j = $i - 1;$j >= 0;$j--){
13            //4、 比较
14            if($arr[$j] > $temp){
15                //说明当前要插入的元素，比前面的已经排好序
16                $arr[$j+1] = $arr[$j];
17                $arr[$j] = $temp;
18            }else{
19                //说明当前待插入元素，比前面的元素要大：说
20                break;
21            }
22        }
23    }
24
25    }
26
27    }
28
29    }
30
31    }
32
33    }
34
35    }
36
37    }
38
39    }
40
41    }
42
43    }
44
45    }
46
47    }
48
49    }
50
51    }
52
53    }
54
55    }
56
57    }
58
59    }
60
61    }
62
63    }
64
65    }
66
67    }
68
69    }
70
71    }
72
73    }
74
75    }
76
77    }
78
79    }
80
81    }
82
83    }
84
85    }
86
87    }
88
89    }
90
91    }
92
93    }
94
95    }
96
97    }
98
99    }
100
101    }
102
103    }
104
105    }
106
107    }
108
109    }
110
111    }
112
113    }
114
115    }
116
117    }
118
119    }
120
121    }
122
123    }
124
125    }
126
127    }
128
129    }
130
131    }
132
133    }
134
135    }
136
137    }
138
139    }
140
141    }
142
143    }
144
145    }
146
147    }
148
149    }
150
151    }
152
153    }
154
155    }
156
157    }
158
159    }
160
161    }
162
163    }
164
165    }
166
167    }
168
169    }
170
171    }
172
173    }
174
175    }
176
177    }
178
179    }
180
181    }
182
183    }
184
185    }
186
187    }
188
189    }
190
191    }
192
193    }
194
195    }
196
197    }
198
199    }
200
201    }
202
203    }
204
205    }
206
207    }
208
209    }
210
211    }
212
213    }
214
215    }
216
217    }
218
219    }
220
221    }
222
223    }
224
225    }
226
227    }
228
229    }
230
231    }
232
233    }
234
235    }
236
237    }
238
239    }
240
241    }
242
243    }
244
245    }
246
247    }
248
249    }
250
251    }
252
253    }
254
255    }
256
257    }
258
259    }
260
261    }
262
263    }
264
265    }
266
267    }
268
269    }
270
271    }
272
273    }
274
275    }
276
277    }
278
279    }
280
281    }
282
283    }
284
285    }
286
287    }
288
289    }
290
291    }
292
293    }
294
295    }
296
297    }
298
299    }
300
301    }
302
303    }
304
305    }
306
307    }
308
309    }
310
311    }
312
313    }
314
315    }
316
317    }
318
319    }
320
321    }
322
323    }
324
325    }
326
327    }
328
329    }
330
331    }
332
333    }
334
335    }
336
337    }
338
339    }
340
341    }
342
343    }
344
345    }
346
347    }
348
349    }
350
351    }
352
353    }
354
355    }
356
357    }
358
359    }
360
361    }
362
363    }
364
365    }
366
367    }
368
369    }
370
371    }
372
373    }
374
375    }
376
377    }
378
379    }
380
381    }
382
383    }
384
385    }
386
387    }
388
389    }
390
391    }
392
393    }
394
395    }
396
397    }
398
399    }
400
401    }
402
403    }
404
405    }
406
407    }
408
409    }
410
411    }
412
413    }
414
415    }
416
417    }
418
419    }
420
421    }
422
423    }
424
425    }
426
427    }
428
429    }
430
431    }
432
433    }
434
435    }
436
437    }
438
439    }
440
441    }
442
443    }
444
445    }
446
447    }
448
449    }
450
451    }
452
453    }
454
455    }
456
457    }
458
459    }
460
461    }
462
463    }
464
465    }
466
467    }
468
469    }
470
471    }
472
473    }
474
475    }
476
477    }
478
479    }
480
481    }
482
483    }
484
485    }
486
487    }
488
489    }
490
491    }
492
493    }
494
495    }
496
497    }
498
499    }
500
501    }
502
503    }
504
505    }
506
507    }
508
509    }
510
511    }
512
513    }
514
515    }
516
517    }
518
519    }
520
521    }
522
523    }
524
525    }
526
527    }
528
529    }
530
531    }
532
533    }
534
535    }
536
537    }
538
539    }
540
541    }
542
543    }
544
545    }
546
547    }
548
549    }
550
551    }
552
553    }
554
555    }
556
557    }
558
559    }
560
561    }
562
563    }
564
565    }
566
567    }
568
569    }
570
571    }
572
573    }
574
575    }
576
577    }
578
579    }
580
581    }
582
583    }
584
585    }
586
587    }
588
589    }
590
591    }
592
593    }
594
595    }
596
597    }
598
599    }
600
601    }
602
603    }
604
605    }
606
607    }
608
609    }
610
611    }
612
613    }
614
615    }
616
617    }
618
619    }
620
621    }
622
623    }
624
625    }
626
627    }
628
629    }
630
631    }
632
633    }
634
635    }
636
637    }
638
639    }
640
641    }
642
643    }
644
645    }
646
647    }
648
649    }
650
651    }
652
653    }
654
655    }
656
657    }
658
659    }
660
661    }
662
663    }
664
665    }
666
667    }
668
669    }
670
671    }
672
673    }
674
675    }
676
677    }
678
679    }
680
681    }
682
683    }
684
685    }
686
687    }
688
689    }
690
691    }
692
693    }
694
695    }
696
697    }
698
699    }
700
701    }
702
703    }
704
705    }
706
707    }
708
709    }
710
711    }
712
713    }
714
715    }
716
717    }
718
719    }
720
721    }
722
723    }
724
725    }
726
727    }
728
729    }
730
731    }
732
733    }
734
735    }
736
737    }
738
739    }
740
741    }
742
743    }
744
745    }
746
747    }
748
749    }
750
751    }
752
753    }
754
755    }
756
757    }
758
759    }
760
761    }
762
763    }
764
765    }
766
767    }
768
769    }
770
771    }
772
773    }
774
775    }
776
777    }
778
779    }
780
781    }
782
783    }
784
785    }
786
787    }
788
789    }
790
791    }
792
793    }
794
795    }
796
797    }
798
799    }
800
801    }
802
803    }
804
805    }
806
807    }
808
809    }
810
811    }
812
813    }
814
815    }
816
817    }
818
819    }
820
821    }
822
823    }
824
825    }
826
827    }
828
829    }
830
831    }
832
833    }
834
835    }
836
837    }
838
839    }
840
841    }
842
843    }
844
845    }
846
847    }
848
849    }
850
851    }
852
853    }
854
855    }
856
857    }
858
859    }
860
861    }
862
863    }
864
865    }
866
867    }
868
869    }
870
871    }
872
873    }
874
875    }
876
877    }
878
879    }
880
881    }
882
883    }
884
885    }
886
887    }
888
889    }
890
891    }
892
893    }
894
895    }
896
897    }
898
899    }
900
901    }
902
903    }
904
905    }
906
907    }
908
909    }
910
911    }
912
913    }
914
915    }
916
917    }
918
919    }
920
921    }
922
923    }
924
925    }
926
927    }
928
929    }
930
931    }
932
933    }
934
935    }
936
937    }
938
939    }
940
941    }
942
943    }
944
945    }
946
947    }
948
949    }
950
951    }
952
953    }
954
955    }
956
957    }
958
959    }
960
961    }
962
963    }
964
965    }
966
967    }
968
969    }
970
971    }
972
973    }
974
975    }
976
977    }
978
979    }
980
981    }
982
983    }
984
985    }
986
987    }
988
989    }
990
991    }
992
993    }
994
995    }
996
997    }
998
999    }
1000
1001    }
1002
1003    }
1004
1005    }
1006
1007    }
1008
1009    }
1010
1011    }
1012
1013    }
1014
1015    }
1016
1017    }
1018
1019    }
1020
1021    }
1022
1023    }
1024
1025    }
1026
1027    }
1028
1029    }
1030
1031    }
1032
1033    }
1034
1035    }
1036
1037    }
1038
1039    }
1040
1041    }
1042
1043    }
1044
1045    }
1046
1047    }
1048
1049    }
1050
1051    }
1052
1053    }
1054
1055    }
1056
1057    }
1058
1059    }
1060
1061    }
1062
1063    }
1064
1065    }
1066
1067    }
1068
1069    }
1070
1071    }
1072
1073    }
1074
1075    }
1076
1077    }
1078
1079    }
1080
1081    }
1082
1083    }
1084
1085    }
1086
1087    }
1088
1089    }
1090
1091    }
1092
1093    }
1094
1095    }
1096
1097    }
1098
1099    }
1100
1101    }
1102
1103    }
1104
1105    }
1106
1107    }
1108
1109    }
1110
1111    }
1112
1113    }
1114
1115    }
1116
1117    }
1118
1119    }
1120
1121    }
1122
1123    }
1124
1125    }
1126
1127    }
1128
1129    }
1130
1131    }
1132
1133    }
1134
1135    }
1136
1137    }
1138
1139    }
1140
1141    }
1142
1143    }
1144
1145    }
1146
1147    }
1148
1149    }
1150
1151    }
1152
1153    }
1154
1155    }
1156
1157    }
1158
1159    }
1160
1161    }
1162
1163    }
1164
1165    }
1166
1167    }
1168
1169    }
1170
1171    }
1172
1173    }
1174
1175    }
1176
1177    }
1178
1179    }
1180
1181    }
1182
1183    }
1184
1185    }
1186
1187    }
1188
1189    }
1190
1191    }
1192
1193    }
1194
1195    }
1196
1197    }
1198
1199    }
1200
1201    }
1202
1203    }
1204
1205    }
1206
1207    }
1208
1209    }
1210
1211    }
1212
1213    }
1214
1215    }
1216
1217    }
1218
1219    }
1220
1221    }
1222
1223    }
1224
1225    }
1226
1227    }
1228
1229    }
1230
1231    }
1232
1233    }
1234
1235    }
1236
1237    }
1238
1239    }
1240
1241    }
1242
1243    }
1244
1245    }
1246
1247    }
1248
1249    }
1250
1251    }
1252
1253    }
1254
1255    }
1256
1257    }
1258
1259    }
1260
1261    }
1262
1263    }
1264
1265    }
1266
1267    }
1268
1269    }
1270
1271    }
1272
1273    }
1274
1275    }
1276
1277    }
1278
1279    }
1280
1281    }
1282
1283    }
1284
1285    }
1286
1287    }
1288
1289    }
1290
1291    }
1292
1293    }
1294
1295    }
1296
1297    }
1298
1299    }
1300
1301    }
1302
1303    }
1304
1305    }
1306
1307    }
1308
1309    }
1310
1311    }
1312
1313    }
1314
1315    }
1316
1317    }
1318
1319    }
1320
1321    }
1322
1323    }
1324
1325    }
1326
1327    }
1328
1329    }
1330
1331    }
1332
1333    }
1334
1335    }
1336
1337    }
1338
1339    }
1340
1341    }
1342
1343    }
1344
1345    }
1346
1347    }
1348
1349    }
1350
1351    }
1352
1353    }
1354
1355    }
1356
1357    }
1358
1359    }
1360
1361    }
1362
1363    }
1364
1365    }
1366
1367    }
1368
1369    }
1370
1371    }
1372
1373    }
1374
1375    }
1376
1377    }
1378
1379    }
1380
1381    }
1382
1383    }
1384
1385    }
1386
1387    }
1388
1389    }
1390
1391    }
1392
1393    }
1394
1395    }
1396
1397    }
1398
1399    }
1400
1401    }
1402
1403    }
1404
1405    }
1406
1407    }
1408
1409    }
1410
1411    }
1412
1413    }
1414
1415    }
1416
1417    }
1418
1419    }
1420
1421    }
1422
1423    }
1424
1425    }
1426
1427    }
1428
1429    }
1430
1431    }
1432
1433    }
1434
1435    }
1436
1437    }
1438
1439    }
1440
1441    }
1442
1443    }
1444
1445    }
1446
1447    }
1448
1449    }
1450
1451    }
1452
1453    }
1454
1455    }
1456
1457    }
1458
1459    }
1460
1461    }
1462
1463    }
1464
1465    }
1466
1467    }
1468
1469    }
1470
1471    }
1472
1473    }
1474
1475    }
1476
1477    }
1478
1479    }
1480
1481    }
1482
1483    }
1484
1485    }
1486
1487    }
1488
1489    }
1490
1491    }
1492
1493    }
1494
1495    }
1496
1497    }
1498
1499    }
1500
1501    }
1502
1503    }
1504
1505    }
1506
1507    }
1508
1509    }
1510
1511    }
1512
1513    }
1514
1515    }
1516
1517    }
1518
1519    }
1520
1521    }
1522
1523    }
1524
1525    }
1526
1527    }
1528
1529    }
1530
1531    }
1532
1533    }
1534
1535    }
1536
1537    }
1538
1539    }
1540
1541    }
1542
1543    }
1544
1545    }
1546
1547    }
1548
1549    }
1550
1551    }
1552
1553    }
1554
1555    }
1556
1557    }
1558
1559    }
1560
1561    }
1562
1563    }
1564
1565    }
1566
1567    }
1568
1569    }
1570
1571    }
1572
1573    }
1574
1575    }
1576
1577    }
1578
1579    }
1580
1581    }
1582
1583    }
1584
1585    }
1586
1587    }
1588
1589    }
1590
1591    }
1592
1593    }
1594
1595    }
1596
1597    }
1598
1599    }
1600
1601    }
1602
1603    }
1604
1605    }
1606
1607    }
1608
1609    }
1610
1611    }
1612
1613    }
1614
1615    }
1616
1617    }
1618
1619    }
1620
1621    }
1622
1623    }
1624
1625    }
1626
1627    }
1628
1629    }
1630
1631    }
1632
1633    }
1634
1635    }
1636
1637    }
1638
1639    }
1640
1641    }
1642
1643    }
1644
1645    }
1646
1647    }
1648
1649    }
1650
1651    }
1652
1653    }
1654
1655    }
1656
1657    }
1658
1659    }
1660
1661    }
1662
1663    }
1664
1665    }
1666
1667    }
1668
1669    }
1670
1671    }
1672
1673    }
1674
1675    }
1676
1677    }
1678
1679    }
1680
1681    }
1682
1683    }
1684
1685    }
1686
1687    }
1688
1689    }
1690
1691    }
1692
1693    }
1694
1695    }
1696
1697    }
1698
1699    }
1700
1701    }
1702
1703    }
1704
1705    }
1706
1707    }
1708
1709    }
1710
1711    }
1712
1713    }
1714
1715    }
1716
1717    }
1718
1719    }
1720
1721    }
1722
1723    }
1724
1725    }
1726
1727    }
1728
1729    }
1730
1731    }
1732
1733    }
1734
1735    }
1736
1737    }
1738
1739    }
1740
1741    }
1742
1743    }
1744
1745    }
1746
1747    }
1748
1749    }
1750
1751    }
1752
1753    }
1754
1755    }
1756
1757    }
1758
1759    }
1760
1761    }
1762
1763    }
1764
1765    }
1766
1767    }
1768
1769    }
1770
1771    }
1772
1773    }
1774
1775    }
1776
1777    }
1778
1779    }
1780
1781    }
1782
1783    }
1784
1785    }
1786
1787    }
1788
1789    }
1790
1791    }
1792
1793    }
1794
1795    }
1796
1797    }
1798
1799    }
1800
1801    }
1802
1803    }
1804
1805    }
1806
1807    }
1808
1809    }
1810
1811    }
1812
1813    }
1814
1815    }
1816
1817    }
1818
1819    }
1820
1821    }
1822
1823    }
1824
1825    }
1826
1827    }
1828
1829    }
1830
1831    }
1832
1833    }
1834
1835    }
1836
1837    }
1838
1839    }
1840
1841    }
1842
1843    }
1844
1845    }
1846
1847    }
1848
1849    }
1850
1851    }
1852
1853    }
1854
1855    }
1856
1857    }
1858
1859    }
1860
1861    }
1862
1863    }
1864
1865    }
1866
1867    }
1868
1869    }
1870
1871    }
1872
1873    }
1874
1875    }
1876
1877    }
1878
1879    }
1880
1881    }
1882
1883    }
1884
1885    }
1886
1887    }
1888
1889    }
1890
1891    }
1892
1893    }
1894
1895    }
1896
1897    }
1898
1899    }
1900
1901    }
1902
1903    }
1904
1905    }
1906
1907    }
1908
1909    }
1910
1911    }
1912
1913    }
1914
1915    }
1916
1917    }
1918
1919    }
1920
1921    }
1922
1923    }
1924
1925    }
1926
1927    }
1928
1929    }
1930
1931    }
1932
1933    }
1934
1935    }
1936
1937    }
1938
1939    }
1940
1941    }
1942
1943    }
1944
1945    }
1946
1947    }
1948
1949    }
1950
1951    }
1952
1953    }
1954
1955    }
1956
1957    }
1958
1959    }
1960
1961    }
1962
1963    }
1964
1965    }
1966
1967    }
1968
1969    }
1970
1971    }
1972
1973    }
1974
1975    }
1976
1977    }
1978
1979    }
1980
1981    }
1982
1983    }
1984
1985    }
1986
1987    }
1988
1989    }
1990
1991    }
1992
1993    }
1994
1995    }
1996
1997    }
1998
1999    }
2000
2001    }
2002
2003    }
2004
2005    }
2006
2007    }
2008
2009    }
2010
2011    }
2012
2013    }
2014
2015    }
2016
2017    }
2018
2019    }
2020
2021    }
2022
2023    }
2024
2025    }
2026
2027    }
2028
2029    }
2030
2031    }
2032
2033    }
2034
2035    }
2036
2037    }
2038
2039    }
2040
2041    }
2042
2043    }
2044
2045    }
2046
2047    }
2048
2049    }
2050
2051    }
2052
2053    }
2054
2055    }
2056
2057    }
2058
2059    }
2060
2061    }
2062
2063    }
2064
2065    }
2066
2067    }
2068
2069    }
2070
2071    }
2072
2073    }
2074
2075    }
2076
2077    }
2078
2079    }
2080
2081    }
2082
2083    }
2084
2085    }
2086
2087    }
2088
2089    }
2090
2091    }
2092
2093    }
2094
2095    }
2096
2097    }
2098
2099    }
2100
2101    }
2102
2103    }
2104
2105    }
2106
2107    }
2108
2109    }
2110
2111    }
2112
2113    }
2114
2115    }
2116
2117    }
2118
2119    }
2120
2121    }
2122
2123    }
2124
2125    }
2126
2127    }
2128
2129    }
2130
2131    }
2132
2133    }
2134
2135    }
2136
2137    }
2138
2139    }
2140
2141    }
2142
2143    }
2144
2145    }
2146
2147    }
2148
2149    }
2150
2151    }
2152
2153    }
2154
2155    }
2156
2157    }
2158
2159    }
2160
2161    }
2162
2163    }
2164
2165    }
2166
2167    }
2168
2169    }
2170
2171    }
2172
2173    }
2174
2175    }
2176
2177    }
2178
2179    }
2180
2181    }
2182
2183    }
2184
2185    }
2186
2187    }
2188
2189    }
2190
2191    }
2192
2193    }
2194
2195    }
2196
2197    }
2198
21
```

设要排序的数组是 $A[0] \cdots A[N-1]$ ，首先任意选取一个数据（通常选用数组的第一个数）作为关键数据，然后将所有比它小的数都放到它前面，所有比它大的数都放到它后面，这个过程称为一趟快速排序。值得注意的是，快速排序不是一种稳定的排序算法，也就是说，多个相同的值的相对位置也许会在算法结束时产生变动。

快速排序的算法是：

- 1) 从数组中选出一个元素（通常第一个），作为参照对象。
- 2) 定义两个数组，将目标数组中剩余的元素与参照元素挨个比较：小的放到一个数组，大的放到另外一个数组。
- 3) 第二步执行完之后，前后的数组顺序不确定，但是确定了自己的位置。
- 4) 将得到的小数组按照第 1 到第 3 部重复操作（子问题）。
- 5) 回溯最小数组（一个元素）。

```
7      //快速排序
8      function quick_sort($arr){
9          //递归出口
10         $len = count($arr);
11         if($len <= 1) return $arr;|
12         //取出某个元素，然后将剩余的数组元素，分散到两个不同的数组中
13         $left = $right = array();
14         for($i = 1;$i < $len;$i++){
15             //第一个元素作为比较元素
16             //比较：小的放left中，大的放right中
17             if($arr[$i] < $arr[0]){
18                 $left[] = $arr[$i];
19             }else{
20                 $right[] = $arr[$i];
21             }
22         }
23         // $left和$right数组元素没有排好序：递归点
24         $left = quick_sort($left);
25         $right = quick_sort($right);
26         //合并三个“数”组
27         return array_merge($left,(array)$arr[0],$right);
28     }
29 }
```

归并排序

归并排序（MERGE-SORT）是建立在归并操作上的一种有效的排序算法,该算法是采用分治法（Divide and Conquer）的一个非常典型的应用。将已有序的子序列合并，得到完全有序的序列；即先使每个子序列有序，再使子序列段间有序。若将两个有序表合并成一个有序表，称为二路归并。

二路归并实现：

```

2
3 //PHP数组排序算法：合并算法
4
5 //二路归并
6 $arr1 = array(1,3,5);
7 $arr2 = array(2,4,6);
8
9 //取出一个空数组用于归并空间
10 $arr3 = array();
11 while(count($arr1) && count($arr2)){
12     //只要$arr1和$arr2里面还有元素，就进行循环
13     //取出每个数组的第一个元素：进行比较
14     $arr3[] = $arr1[0] < $arr2[0] ? array_shift($arr1) : array_shift($arr2);
15 }
16
17 //合并结果
18 print_r(array_merge($arr3,$arr1,$arr2));

```

归并排序的算法是：

- 1) 将数组拆分成两个数组。
- 2) 重复步骤 1 将数组拆分成最小单元。
- 3) 申请空间，使其大小为两个已经排序序列之和，该空间用来存放合并后的序列。
- 4) 设定两个指针，最初位置分别为两个已经排序序列的起始位置。
- 5) 比较两个指针所指向的元素，选择相对小的元素放入到合并空间，并移动指针到下一位置。
- 6) 重复步骤 3 直到某一指针超出序列尾。
- 7) 将另一序列剩下的所有元素直接复制到合并序列尾。

```

function merge_sort($arr){
    //递归出口
    $len = count($arr);
    if($len <= 1) return $arr;

    //拆分
    $middle = floor($len / 2);
    $left = array_slice($arr,0,$middle);
    $right = array_slice($arr,$middle);

    //递归点：$left和$right都没有排好序：而且可能是多个元素的数组
    $left = merge_sort($left);
    $right = merge_sort($right);

    //假设左边和右边都已经排好序：二路归并
    $m = array();
    while(count($left) && count($right)){
        //只要$arr1和$arr2里面还有元素，就进行循环
        //取出每个数组的第一个元素：进行比较
        $m[] = $left[0] < $right[0] ? array_shift($left) : array_shift($right);
    }
    //返回结果
    return array_merge($m,$left,$right);
}

```

查找算法

查找算法含义

查找是在大量的信息中寻找一个特定的信息元素，在计算机应用中，查找是常用的基本运算。查找算法是指实现查找过程对应的代码结构。就是中大型数组中去快速定位想要的元素。

顺序查找算法

顺序查找也称为线形查找，从数据结构线形表的一端开始，顺序扫描，依次将扫描到的结点关键字与给定值 k 相比较，若相等则表示查找成功；若扫描结束仍没有找到关键字等于 k 的结点，表示查找失败。

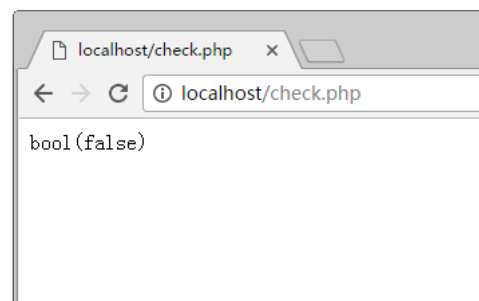
```
$arr = array(1,3,6,8,23,68,100);

//顺序查找：从数组第一个元素开始，挨个匹配
function check_order($arr,$num){

    //全部匹配
    for($i = 0,$len = count($arr);$i < $len;$i++){
        //判断
        if($arr[$i] == $num){
            return $i;
        }
    }

    return false;
}

var_dump(check_order($arr,5));
```



二分查找算法

二分查找要求线形表中的结点按关键字值升序或降序排列，用给定值 k 先与中间结点的关键字比较，中间结点把线形表分成两个子表，若相等则查找成功；若不相等，再根据 k 与该中间结点关键字的比较结果确定下一步查找哪个子表，这样递归进行，直到查找到或查找结束发现表中没有这样的结点。

折半算法思路：

- 1、 计算数组长度；
- 2、 确定左右两边的指针位置；
- 3、 找到中间位置；
- 4、 匹配
- 5、 然后根据大小重定边界

```
8
9 function check_break($arr,$res){
0     //1、    得到数组边界
1     $right = count($arr);
2     $left  = 0;
3     //2、    循环匹配
4     while($left <= $right){
5         //3、    得到中间位置
6         $middle = floor(($right + $left) / 2);
7         //4、    匹配数据
8         if($arr[$middle] == $res){
9             return $middle + 1;
0         }
1         //5、    没有找到
2         if($arr[$middle] < $res){
3             //值在右边
4             $left = $middle + 1;
5         }else{
6             //值在左边
7             $right = $middle - 1;
8         }
9     }
0     return false;
1
```
