4 TABLOURI

4.1 Descriere generala

Un tablou PHP reprezinta o succesiune de elemente sub forma de perechi cheie→valoare, unde fiecare cheie identifica in mod unic un element. Spre deosebire de alte limbaje, unde elementele unui tablou erau accesate prin intermediul unui index numeric ale carui valori intregi erau consecutive si incepeau de la 0, in PHP indexul (cheia) poate fi integer sau string, iar cheile se pot afla in orice ordine. De asemenea, valorile din tablou pot avea orice tip de date:

TABLOU	Primul element	Al doilea element	Al treilea element	Al patrulea element
Cheie	13	2	"trei"	10
Valoare	8	"PHP"	true	15.3

4.2 Accesarea elementelor

Tabloul este practic o lista de variabile (valorile din tablou) care nu mai au nume propriu, independent, ci sunt accesate folosind numele tabloului si cheia corespunzatoare:

```
echo $tablou[13];        // 8
echo $tablou[2];         // "PHP"
echo $tablou["trei"];    // 1 (rezultatul conversiei la string a lui true)
```

La accesarea unui element se poate folosi ca cheie orice variabila sau expresie, aceasta fiind convertita la integer sau string daca este cazul:

```
$a = 5; $b = 17.2;
echo $tablou[$a]; // se afiseaza valoarea lui $tablou[5]
echo $tablou[$b]; // afisarea valorii lui $tablou[17] (conversia valorii lui $b la int)
echo $tablou[$b/$a]; // rezultatul lui $b/$a se converteste la int, rezultand $tablou[3]
```

4.3 Proprietati ale cheilor tabloului

Cheile din tablou pot fi doar integer sau string! Orice alt tip de date folosit ca cheie (la definirea tabloului sau la accesarea elementelor sale) este convertit automat la unul dintre aceste doua tipuri de date:

- cheile string care contin un numar intreg **zecimal** valid (nu octal, nu hexazecimal!) sunt convertite la integer
- cheile float si boolean sunt convertite la integer (false→0, true→1)
- cheia NULL este convertita la sirul vid "", care reprezinta o cheie valida!

```
$a = array(1.2=>"alfa", "2" = >"beta", "03"=>"gama");
var_dump($a);     /* array(3) {
                        [1]=>     string(4) "alfa"  // 1.2 a fost convertit la int →1
                        [2]=>     string(4) "beta"  // "2" a fost convertit la int →2
                        ["03"]=>  string(4) "gama"  // "03" NU este convertit la int
                    }*/
```

In functie de tipurile de date ale cheilor, tablourile pot fi:

- **indexate numeric** – cheile sunt de tip integer; nu este obligatoriu ca ele sa inceapa de la 0, sa fie numere consecutive sau sa se afle in ordine crescatoare. Ex: $tab[15] - nu stim pe a cata pozitie se afla acest element
- **asociative** – cheile sunt de tip string (atentie! case sensitive!). Ex: $persoana['nume']
- **mixte** – tablouri care contin o combinatie de chei numerice si string

4.4 Utilitatea tablourilor

Deoarece tabloul reprezinta o grupare de valori (date) de diverse feluri aflate sub umbrela unui nume comun, acesta ofera avantaje precum:

- posibilitatea ca o functie sa primeasca ca argument sau sa returneze mai multe valori simultan, sub forma unui tablou
- parcurgerea in vederea citirii sau prelucrarii datelor componente se poate usor automatiza, folosind instructiuni sau functii PHP predefinite (vezi mai jos sectiunea dedicata pacurgerii unui tablou).
- pentru operatii uzuale precum sortarea, numararea, cautarea unui element etc exista functii PHP predefinite (vezi sectiunea despre operatii uzuale cu tablouri)
- gratie flexibilitatii tablourilor in PHP a functiilor predefinite, tablourile pot fi folosite pentru a simula alte structuri de date precum stiva, coada, arbore, hash table etc, existand functii PHP predefinite in acest sens

4.5 Crearea si popularea unui tablou

4.5.1 Folosind constructia array()

Aceasta este modalitatea primara de a crea tablouri, care ofera posibilitatea specificarii explicite a valorilor si eventual a cheilor corespunzatoare:

```
$t = array(valoare1, valoare2,…);
$u = array(cheie1=>valoare1, cheie2=>valoare2,…);
```

Definitia de tablou este parcursa de la stanga la dreapta; elementele sunt adaugate unul cate unul, tinandu-se de fiecare data cont de cele deja adaugate pana atunci. Pentru orice element a carui cheie nu este precizata se creeaza automat o cheie numerica, dupa urmatoarele reguli:

- daca este prima cheie numerica din tablou, va primi valoarea 0
- daca exista deja chei numerice in tablou, insa sunt toate negative, noua cheie va fi 0
- daca exista deja in tablou chei numerice >=0, noua cheie va fi egala cu cheia numerica maxima plus unu

Exemple:

```
// specificarea valorilor; cheile sunt asignate automat si vor fi de tip numeric
$t = array(1,2,3);
var_dump($t); /* afiseaza: array(3) {
                              [0]=>  int(1)
                              [1]=>  int(2)
                              [2]=>  int(3)
                             }*/

// specificarea explicita a cheilor pentru fiecare element
$t = array(10=>"one", "doi"=>"two", 15=>"three");
var_dump($t); /* afiseaza: array(3) {
                              [10]=>string(3) "one"
                              ["doi"]=>string(3) "two"
                              [15]=>string(5) "three"
                         }*/

// specificare partiala a cheilor; cele nespecificate primesc valori numerice
$t = array(1=>"unu",2=>"doi", "trei"); // al treilea element primeste cheia 3, egala cu
                                       // maximul cheilor numerice din tablou (2)
plus unu

$t = array("unu"=>"one","doi"=>"two", "trei");  /* elementul cu valoarea "trei" primeste
                                                   cheia 0, deoarece este prima cheie
numerica*/
```

```
$t = array(-7=>"a", "letter"=>"a", "b", "c", 13=>"d", "e"); var_dump($t);
/* array(6) {
  [-7]=> string(1) "a"
  ["letter"]=> string(1) "a"
  [0]=> string(1) "b"   // (cheia numerica maxima dinaintea crearii lui "b" era negativa
  [1]=> string(1) "c"   // cheia numerica maxima inaintea crearii lui "c" era cea a lui
"b"
  [13]=> string(1) "d"
  [14]=> string(1) "e"  // cheia numerica maxima inaintea crearii lui "e" era 13
}*/
```

4.5.2 Adaugarea/modificarea unui element prin specificarea cheii

Sintaxa folosita pentru accesarea unui element permite si modificarea valorii acestuia:

```
$tablou[cheie] = valoare;
```

Daca elementul cu cheia in cauza exista, valoarea sa va fi suprascrisa. Daca cheia nu exista in tablou, va fi creat un nou element cu cheia precizata. Daca variabila $tablou nu a fost definita sau a fost stearsa, va fi creat un tablou cu un element avand cheia si valoarea specificata.

```
$a = array(1,2);  // 0=>1, 1=>2
$a[1] = 7;  // $a are acum elementele 0=>1, 1=>7
$a[3] = 13  // $a are acum elementele 0=>1, 1=>7, 3=>13

// adaugarea unui element cu specificarea cheii si conversie automata
$a[0.15] = "opt"; // 0.15 este convertit la integer, este suprascrisa valoarea cheii 0!

$b[7] = "doi"; var_dump($b);  // $b este creat ca tablou, cu elementul 7=>"doi"
```

Atentie! *Daca variabila exista deja, insa are tipul de date string, sintaxa de mai sus va coincide cu cea folosita pentru accesarea caracterelor componente ale string-ului. In consecinta, daca incercam sa asignam variabilei un tablou folosind aceasta sintaxa, vom obtine simpla modificare a unui caracter:*

```
$s = "Infoacademy";
$s[5] = 'x'; // nu creeaza un tablou cu un element, ci schimba c cu x!
($s←"Infoaxademy");
```

4.5.3 Adaugarea unui element fara specificarea cheii

Precedenta sintaxa poate fi folosita si fara a specifica cheia:

```
$t[] = valoare;
```

Efectul va fi:

- daca variabila $t nu exista, va fi creat un tablou cu un element ce are cheia 0 si valoarea specificata
- daca variabila $t exista si este de tip tablou, va fi adaugat un nou element cu valoarea specificata, a carui cheie este generata dupa aceleasi reguli ca si in cazul constructiei array() (maximul cheilor numerice existente plus unu, sau 0 pt chei numerice absente sau negative)

```
// crearea unui tablou prin adaugarea unui element, fara specificarea cheii
$t[] = "element"; // 0=>"element"

// adaugarea unui element intr-un tablou existent, dar fara chei numerice
$t = array("a"=>"unu", "b"=>"doi");
$t[] = "opt";     // "a"=>"unu", "b"=>"doi", 0=>"opt"
```

```
// adaugarea unui element intr-un tablou existent, ce contine deja chei numerice
$t = array( 'a', 10 => 'b', 'c');   // 0=>'a', 10=>'b', 11=>'c'
$t[] = 'd';                         // 0=>'a', 10=>'b', 11=>'c', 12=>'d'
```

4.6 Parcurgerea unui tablou

Indiferent daca scopul este simpla citire a valorilor/cheilor sau modificarea acestora, parcurgerea unui tablou PHP se poate face in cel putin trei moduri:

4.6.1 Cu *foreach*

Constructia **foreach** functioneaza numai cu tablouri sau cu obiecte care implementeaza interfata Iterator (vezi SPL - Standard PHP Library). Ea are doua forme de baza si inca doua introduse in PHP 5:

```
$a = array( "unu" =>1, "doi" =>2, 3=>"trei );

// Forme de baza
//1. Iterare prin valorile elementelor
foreach($a as $valoare){
      echo $valoare;
}
//2. Iterare cu acces la cheia si valoarea fiecarui element
foreach($a as $cheie=>$valoare){
      echo "Cheie: $cheie => Valoare: $valoare";
}
```

Constructia **foreach** scrisa astfel lucreaza pe o copie a tabloului, deoarece $valoare ia pe rand valoarea fiecarui element din tablou (asignare prin valoare!) si de aceea modificand $valoare nu pot fi alterate elementele tabloului original. Din PHP5 insa exista posibilitatea asignarii lui $valoare prin referinta, tabloul putand fi astfel alterat prin modificarea lui $valoare:

```
foreach($a as $cheie=>&$valoare){ // functioneaza si cealalta forma:foreach($a as
&$valoare)
      $valoare++;
}
```

Atentie! *In urma unei astfel de bucle, $valoare ramane "conectat" cu ultima valoare din tablou, din cauza asignarii prin referinta. De aceea este necesar ca dupa incheierea buclei variabila $valoare sa fie stearsa cu unset().*

```
$a = array( 1,2,3);
foreach($a as &$val){}
// daca aici nu folosim unset($val), iata ce se poate intampla:
$val = 4;
echo $a[2]; // 4!!
```

4.6.2 Cu functii care modifica pointerul intern al tabloului

Fiecare array are un pointer intern care indica elementul curent in iterare – acest artificiu scuteste programatorul de a folosi o variabila externa in acest scop. Pointerul se manipuleaza indirect, prin functii:

- **reset**(array &$array) – aduce pointerul la inceputul tabloului si returneaza primul element, sau FALSE daca array-ul e gol. Este bine sa fie folosit inainte de fiecare parcurgere a tabloului, daca nu cunoastem starea sa
- **current**(array &$array) si **key(**array &$array) – returneaza valoarea, respectiv cheia elementului curent fara a misca pointerul

- **prev**(array &$array) si **next**(array &$array) – returneaza elementul precedent/urmator sau FALSE daca s-a ajuns la capatul tabloului. Atentie la elementele care == FALSE! Nu putem distinge intre capatul de tablou si un element cu valoarea FALSE!

Atentie! *print_r() lasa pointerul intern al tabloului la sfarsit!*

```
$a = array('a','b','c','d');
do{
      echo key($a).' => '.current($a);
}while(next($a)); // 0=>'a' 1=>'b' 2=>'c' 3=>'d'
```

- **each**(array &$array) – returneaza elementul curent sub forma unui array asociativ **si muta pointerul intern la urmatoarea valoare.** Returneaza FALSE daca a ajuns la sfarsitul array-ului.

```
$b = array("abra","cadabra");
var_dump(each($b));     // este returnat elementul curent (primul) ca tablou asociativ
/*array(4) {
 [1]=>  string(4) "abra"          // valoarea elementului curent, cu cheia 1
 ["value"]=>  string(4) "abra"    // valoarea elementului curent, cu cheia "value"
 [0]=>  int(0)                    // cheia elementului curent, cu cheia 0
 ["key"]=>  int(0)                // cheia elementului curent, cu cheia "key"
}*/
```

Remarca: *Rezultatul lui each este foarte potrivit pentru a fi asignat constructiei list($cheie,$valoare) (vezi mai jos sectiunea corespunzatoare).*

4.6.3 Cu functiile predefinite array_walk() sau array_walk_recursive()

Aceste functii aplica o functie specificata de utilizator (un asa-numit *callback*[1]) fiecarui element al tabloului.

```
bool array_walk(array &$array , callback $fctname [, mixed $userdata ])
bool array_walk_recursive ( array &$input , callback $fctname [, mixed $userdata ] )
```

Diferenta intre cele doua functii este ca cea de-a doua este potrivita pentru tablouri multidimensionale (pentru fiecare valoare intalnita care este de tip tablou, functia array_walk_recursive() va cobori si in adancimea tabloului respectiv).

Argumentul fctname al celor doua functii PHP de mai sus are urmatorul prototip:

```
function fctname($value, $key, $userdata=null){ instructiuni; }
function fctname(&$value, $key, $userdata=null){ instructiuni; }
// varianta 2 este pentru cazul in care dorim sa putem modifica elementele tabloului
```

Exemplu:

```
$a = array(10,11,12);
function creste(&$v,$k,$data=null){ $v++; }
array_walk($a, "creste");     // $a devine 0=>11, 1=>12, 2=>13
```

Functii asemanatoare: array_walk() vs array_map():

[1] callback = functie scrisa de catre programator pentru a fi apelata in cadrul unui algoritm deja scris (ex: un algoritm de sortare de tablou, in care algoritmul de comparare a elementelor este specificat intr-o functie separata, exterioara algoritmului)

Functie	Returneaza	Argumente	Callback
array_walk	true/false (lucreaza direct pe tablou)	array &$array, callback $funcname, [, mixed $userdata]	2 sau 3 argumente: value, key, eventual userdata. Daca dorim sa modificam valori in tablou, vom pasa argumentul valoare prin referinta (&$valoare)
array_map	array-ul ce rezulta din aplicarea callback-ului valorilor tabloului original	callback $callback, array $arr1 [, array $...]	nr de argumente egal cu nr de array-uri pasate ca argument

4.6.4 De ce nu se face parcurgerea tabloului cu for?

In alte limbaje de programare, parcurgerea tablourilor se facea, in cele mai dese cazuri, cu instructiunea for. Acest lucru era posibil deoarece cheile (indecsii) formau o secventa numerica continua care incepea de la 0, iar conditia de iesire din bucla se putea scrie comparand indexul curent cu lungimea tabloului:

```
for(int i=0;i<tablou.length;i++)    // ALT LIMBAJ!
```

In PHP, indecsii din tablou nu sunt neaparat numerici, si chiar si atunci cand sunt numerici, ei pot sa nu formeze o secventa continua sau sa nu se afle in ordinea naturala a numerelor:

```
$t = array( 10=> "zece", 4 => "patru", 20=> "douazeci");
```

4.7 Copierea tablourilor

In PHP, **copierea tablourilor se face prin simpla asignare**:

```
$a = array(1,2,3);
$b = $a;          // asignare prin valoare! $b este acum un tablou distinct
$b[0] = 10;
echo $a[0];       // 1 ($a nu a fost modificat)
echo $b[0];       // 10 (modificarea s-a produs numai asupra tabloului $b)
```

Atentie! *Tineti cont de acest lucru cand lucrati cu tablouri mari!*

4.8 Tablouri multidimensionale

Elementele unui tablou pot avea orice fel de valori, inclusiv tablouri. In acest caz, rezultatul este crearea unui tablou multi-dimensional:

```
$persoana = array(
      'nume'      =>    "Mihai",
      'varsta'    =>    32,
      'hobby'     =>    array(
                              1 => 'citit',
                              2 => 'drumetii montane',
                              3 => 'PHP'
                        ),
      'prieteni'  =>    array(
                              "Ioana"     =>    "scoala",
                              "Andrei"    =>    "serviciu"
      );
echo $persoana['hobby'][1];   // citit
```

Exemplu de parcurgere a unui sub-tablou:

```
echo "Hobby-urile lui ".$persoana['nume']." sunt:"
foreach($persoana['hobby'] as $nr=>$hobby)      // $persoana['hobby'] este tablou
      echo "Hobby-ul numarul $nr: $hobby";
}
```

4.9 Functii PHP predefinite pentru operatiile uzuale cu tablouri

Sunt prezentate in continuare functiile pentru cateva dintre operatiile uzuale cu tablouri:

4.9.1 Afisare recursiva a continutului tabloului

- **print_r()** – afiseaza valorile cheilor si elementelor, in mod recursiv (functioneaza si pentru tablouri multidimensionale)
- **var_dump()** – acelasi lucru, insa in plus afiseaza tipul de date al cheilor si valorilor si numarul de elemente al tablourilor:

```
$a = array("mere", "pere", array("rosii", "ceapa") );
print_r($a);      /* Array (                (
                        [0] => mere
                        [1] => pere
                        [2] => Array (
                                  [0] => rosii
                                  [1] => ceapa
                              )

                  )*/
var_dump($a);     /* array(3) {
                    [0]=> string(4) "mere"
                    [1]=> string(4) "pere"
                    [2]=> array(2) {
                      [0]=> string(5) "rosii"
                      [1]=> string(5) "ceapa"
                    }
                  }*/
```

4.9.2 Inspectie tablou si cautare

4.9.2.1 Aflarea numarului de elemente: *count($tablou)*

Daca argumentul primit nu este tablou, returneaza 1, cu doua exceptii:

- daca argumentul are valoarea NULL, returneaza 0
- daca argumentul este un obiect PHP care implementeaza interfata Countable, va fi returnat rezultatul apelarii metodei count() a obiectului (detalii la lectia despre clase si obiecte)

```
$cursuri = array("CCNA", "CCNP", "Security", "VoIP", "Java", "Unix", "PHP", "CCIP");
echo "Oferim ".count($a)." cursuri";      // Oferim 8 cursuri
```

4.9.2.2 Verificarea daca o variabila este de tip tablou: *is_array($variabila)*

Returneaza true/false. Este foarte utila, de exemplu, pentru a verifica daca o functie gandita sa primeasca si sa prelucreze argumente de tip tablou este apelata cu argumente valide:

```
function prelucrare(array $a){
      if(!is_array($a)) return;
      …
}
```

4.9.2.3 Verificarea prezentei unei anumite valori in tablou: *in_array($valoare, $tablou)*

Returneaza true/false. Exemplu:

```
$aprozar = array("mere", "pere", "prune");
echo (in_array("mere", $aprozar)?"Avem":"Nu avem")." mere!";      // Avem mere!
```

4.9.2.4 Verificarea prezentei unei anumite chei in tablou: *array_key_exists($cheie, $tablou)*

Returneaza true sau false. Exemplu:

```
$note_examen = array("Mihai"=>10, "Carmen"=> 5, "Maria" => 8);
if(array_key_exists("Carmen",$note_examen)){
      echo "Carmen a sustinut examenul si a luat nota ".$note_examen["Carmen"];
}
```

4.9.2.5 Aflarea cheii corespunzatoare unei anumite valori: *array_search($valoare, $tablou)*

Returneaza cheia daca valoarea exista sau FALSE in caz contrar:

```
$oaspeti = array(1=>"Ionut", "Adrian", "Elena", "Marius");
echo "Adrian a sosit al ".array_search("Adrian",$oaspeti)."-lea"; // Adrian a sosit al 2-
lea
```

Valoarea cautata poate fi tablou:

```
$perechi_celebre = array("horror"=> array("Jekyll", "Hyde"), "fun"=>array ("Chip",
"Dale"));
echo "Chip and Dale are ".array_search(array("Chip","Dale"),$perechi_celebre);
// Chip and Dale are fun
```

4.9.3 Stergere

- a intregului tablou: **unset($tablou)**
- a unui element din tablou: **unset($tablou[cheie])**

Atentie! *Tabloul nu va fi reindexat in urma stergerii unui element! Posibilele consecinte sunt doua:*
a) daca cheile aveau valori numerice consecutive, in urma stergerii de elemente vor ramane goluri
b) e posibil ca indexul numeric maxim sa fi fost sters dar sa se tina cont de el la crearea unui nou element cu cheie generata automat:

```
$a = array('a','b','c','d');  // 0=>'a', 1=>'b', 2=>'c', 3=>'d'
unset($a[1]);      // 0=>'a', 2=>'c', 3=>'d'(cheia 1 a disparut, celelalte raman
neschimbate)
unset($a[3]);      // 0=>'a', 2=>'c' - cheia cu valoarea numerica maxima a disparut, dar…
$a[] = 'e';        // 0=>'a', 2=>'c', 4=>'e' - cheia numerica maxima, desi stearsa, a
contat!
```

4.9.4 Extragere ca tablou separat a cheilor sau valorilor

- obtinerea unui tablou cu lista de chei: **array_keys()**
- obtinerea unui tablou cu lista de valori ale elementelor: **array_values()**

```
$configuratii = array("statia1"=>"Pentium4, "statia2"=> "Celeron", "statia3"=> "Athlon");
$statii = array_keys($configuratii); var_dump($statii);
                        /*array(3) {
                              [0]=> string(7) "statia1"
                              [1]=> string(7) "statia2"
                              [2]=> string(7) "statia3"
                        }*/

$procesoare = array_values($configuratii); var_dump($configuratii);
                        /* array(3) {
                              [0]=> string(8) "Pentium4"
                              [1]=> string(7) "Celeron"
                              [2]=> string(6) "Athlon"
                        }*/
```

4.9.5 Ordonarea elementelor

4.9.5.1 Dupa valoare

Functiile de sortare lucreaza direct pe tabloul primit ca argument si returneaza boolean (FALSE in caz de esec).

- **sort($tablou, [int $optiuni])/rsort($tablou, [int $optiuni])** – ordonare crescatoare sau descrescatoare dupa valoare a elementelor tabloului. ***Atentie!*** *Aceste functii reindexeaza! (se pierd cheile originale)*

Optiunile specificate la sortare pot fi:

- **SORT_NUMERIC** – comparatia intre valori se face numeric (se compara conversiile la numar ale valorilor)
- **SORT_STRING** – toate valorile sunt comparate ca stringuri
- **SORT_REGULAR** – compararea valorilor se face fara conversie

Exemplu:

```
$a = array("1.4z","2m",2.1, "3b","10a", "php");
sort($a, SORT_NUMERIC); // convertind la numar valorile in ordinea din tablou, obtinem
                        // 1.4, 2, 3, 10, 0
sort($a, SORT_STRING);  // la compararea ca string, "10a" se afla intre "1.4" si "2", iar
                        // "php" la sfarsit
```

- **asort($tablou, [int $optiuni])/arsort($tablou, [int $optiuni])** – ordonare crescatoare sau descrescatoare dupa valoare, cu pastrarea corespondentelor cheie→valoare
- **usort($tablou, $callback)** – ordonare dupa valoare, insa folosind pentru compararea elementelor o functie scrisa de catre programator (un callback). Functia de comparare trebuie sa returneze 0 pentru egalitate, o valoare negativa daca primul argument e mai mic decat al doilea si o valoare pozitiva in caz contrar. **Atentie! usort() reindexeaza!**

```
function sortare_dupa_char5($s1, $s2){
      return ord($s1[4]) - ord($s2[4]);   // ord() returneaza codul caracterului
}
$domni = array("Dl. Ulmeanu", "Dl. Ionescu", "Dl. Constantinescu", "Dl. Preda");
usort($domni, "sortare_dupa_char5");      // sortam dupa primul caracter de dupa spatiu
print_r($domni);  /* Array (
                        [0] => Dl. Constantinescu
                        [1] => Dl. Ionescu
                        [2] => Dl. Preda
                        [3] => Dl. Ulmeanu
                  )*/
```

- **uasort($tablou,$callback)** – la fel ca usort(), dar cu pastrarea corespondentelor cheie→valoare

- **natsort($tablou)** – ordonare dupa valoare, insa comparand numeric portiunile numerice din valori, dupa logica umana (ex: pentru noi numele de fisiere poza1.gif, poza2.gif si poza10.gif ar trebui sa se afle in aceasta ordine, insa daca le ordonam ca stringuri, vom obtine poza1.gif, poza10.gif si poza2.gif)
- **natcasesort($tablou)** – la fel ca natsort(), insa case-insensitive

```
$a = array("Capitolul 1", "Capitolul 2", "Capitolul 10", "Capitolul 13");
sort($a); print_r($a);        // Capitolul 1, Capitolul 10, Capitolul 13, Capitolul 2
natsort($a); print_r($a);     // Capitolul 1, Capitolul 2, Capitolul 10, Capitolul 13
```

4.9.5.2 Dupa cheie

- **ksort($tablou,[$optiuni]) / krsort($tablou,[$optiuni])** – ordoneaza elementele tabloului primit ca argument dupa chei, crescator sau decrescator. Cel de-al doilea argument are aceeasi semnificatie ca si in cazul lui sort().
- **uksort($tablou, $callback)** – sortare dupa cheie, dar folosind pentru compararea cheilor o functie specificata de catre programator (vezi exemplul de la usort())

4.9.6 Tratarea unui tablou ca stiva sau coada

Exista o multitudine de functii PHP predefinite care permit tratarea unui tablou ca:
- stiva – stiva reprezinta o structura de date in care ultima informatie introdusa este prima extrasa. Stiva modeleaza principiul LIFO (Last In, First Out). Functiile PHP folosite sunt **array_pop(), array_push()**
- coada – coada este o structura in care cele mai vechi date intrate vor fi primele iesite, ea actionand dupa principiul FIFO (First In, First Out). Functii PHP predefinite: **array_unshift(), array_shift()**

array_push()	adauga elementele specificate la sfarsitul tabloului
array_pop()	elimina ultimul element din tablou si il returneaza
array_unshift()	introduce elementele specificate la inceputul tabloului
array_shift()	elimina primul element in tablou si il returneaza

4.9.7 Tratarea unui tablou ca multime

Functiile PHP predefinite care permit tratarea unui tablou ca multime permit urmatoarele operatii:

a) **Diferenta** – functiile **array_diff*()** si **array_udiff*()**. 2 elemente sunt considerate egale atunci cand conversia valorii/cheii lor la string este identica
   - daca in numele functiei apare **u** inaintea lui diff (ex: array_udiff) atunci valorile sunt comparate folosind o functie definita de catre programator (un callback); daca apare **u** la assoc sau la key, atunci cheile sunt cele comparate folosind un callback. Acolo unde nu apare **u**, inseamna ca compararea este facuta folosind algoritmul intern, nu o functie scrisa de programator (vezi tabel)

Observatie: *desi poate simtul simetriei ne-ar indica lipsa unei functii numite* ***array_udiff_key()****, aceasta NU exista, pt ca nu ar avea sens comparare dupa cheie dar cu callback pt compararea valorii*

Functie	Tine cont de valori la comparare	Tine cont de chei la comparare	Compara valorile folosind	Compara cheile folosind
array_diff	da	-	algoritm intern	-
array_diff_key	-	da	-	algoritm intern
array_diff_assoc	da	da	algoritm intern	algoritm intern
array_diff_ukey	-	da	-	user callback
array_diff_uassoc[PHP5]	da	da	algoritm intern	user callback

array_udiff PHP5	da	-	user callback	-
array_udiff_assoc PHP5	da	da	user callback	algoritm intern
array_udiff_uassoc PHP5	da	da	user callback	user callback

b) **intersectie** – functiile sunt denumite cu aceeasi logica de mai sus
- **array_intersect(), array_intersect_assoc() /array_intersect_key()** – compararea se face cu algoritmi interni
- **array_intersect_uassoc()/array_intersect_ukey()** - comparare de chei/valori pe baza unei functii definite de catre programator
- **array_uintersect(), array_uintersect_assoc(), array_uintersect_uassoc()** – adaugate in PHP 5

c) **reuniune** – poate fi realizata fie cu operatorul +, fie cu functia array_merge()

Operatorul + vs functia array_merge()

Operator/functie	Operanzi/arg.	Efect
+	2	elementele tabloului drept sunt adaugate la cel stang. Pentru chei duplicat, este pastrata valoarea stanga (NU este suprascrisa).
array_merge()	>= 1	elementele tabloului stang sunt suprascrise in cazul cheilor duplicat de tip string. Pentru chei numerice duplicat, elementul se adauga in stanga. Cheile numerice se reindexeaza.

4.9.8 Alte functii si constructii utile in lucrul cu tablouri

- **explode ($delimitator, $string)** – "taie" stringul in bucati acolo unde apare delimitatorul, returnand un tablou cu portiunile de string rezultate. (vezi exemplul de mai jos)

  Nota: *daca nu se cunoaste delimitatorul exact, ci doar formatul acestuia, atunci este necesara folosirea de functii precum split() sau preg_split() care accepta regex-uri pentru specificarea delimitatorului. (vezi lectia despre string si regex)*

- **implode ($glue, $array)** – inversul lui explode; concateneaza elementele tabloului folosind $glue ca liant si returneaza stringul rezultat
- **list($var1, $var2, …) = $array;** – asigneaza variabilelor din lista valorile elementelor corespunzatoare din tablou. Sunt luate in considerare numai elementele tabloului care au chei numerice.

```
$str = "mere pere prune";

list($fruct1, $fruct2, $fruct3) = explode(" ",$str);
// $fruct1←"mere", $fruct2←"pere", $fruct3←"prune"

echo implode("; ",array($fruct1, $fruct2, $fruct3)); // mere; pere; prune;
```

Atentie! list*() lucreaza numai cu elementele indexate numeric dintr-un array, in felul urmator: se presupunem ca facem asignarea list($v1,$v2,$v3) = $arr; constructia list va incerca sa asigneze $arr[2] lui $v3, $arr[1] lui $v2 si $arr[0] lui $v1,* in aceasta ordine*!*

4.10 BIBLIOGRAFIE

- PHP Manual – sectiunea Arrays: http://ro.php.net/manual/en/language.types.array.php
- PHP Manual – functii pentru lucrul cu tablouri: http://ro.php.net/manual/en/book.array.php
- Zend PHP 5 Certification Study Guide – capitolul "Arrays" – pag 47-73

Conventie

-referitoare la conditiile de desfasurare a examenului intermediar-

1. **Caracteristici generale ale examenului Intermediar teoretic :**

   a. are ca scop verificarea cunostintelor acumulate in prima parte a fiecarui modul
   b. este compus din intrebari selectate din examenele partiale 1-4
   c. este de tip test grila si are in medie 15-20 de intrebari cu raspuns unic sau multiplu
   d. dureaza 30 de minute
   e. poate fi sustinut numai la sediul academiei „InfoAcademy" in prezenta coordonatorului stiiintific (Mihaela Marinescu) sau a unui instructor delegat de acesta.
   f. este accesibil pe http://www.infoacademy.net→Login→Student Home→ NumeClasa→TakeAssesment
   g. se considera promovat daca numarul de raspunsuri corecte este cel putin egal cu 75% din numarul total al raspunsurilor
   h. poate fi sustinut de maxim 2 ori la interval de 1 saptamana intre cele 2 incercari

2. **Data la care se sustine examenul intermediar**

   a. examenul intermediar (prima incercare) se sustine in prima ora a celei de a 5-a sedinte de curs
   b. eventuala repetare a examenului intermediar (cea de-a doua incercare) va avea loc in urmatoarea zi de vineri de dupa prima sustinere a examenului, intre orele 8-11,30.

3. **Conditii de participare la examenul intermediar teoretic :**

   Poate participa la examenul intermediar teoretic orice student Infoacademy care:
   a. a promovat examenele partiale 1-4 cu cel putin 75%
   b. a achitat obligatiile financiare convenite la inscriere sau se obliga sa le achite pana la data examenului intermediar inclusiv
   c. daca exista obligatii financiare restante, acestea pot fi achitate cu 10-15 minute inaintea examenului intermediar
   d. are asupra sa un act cu fotografie care ii atesta identitatea, contractul semnat cu Infoacademy si documentele fiscale care atesta plata integrala a obligatiilor financiare catre InfoAcademy (chitanta+factura)
   e. se obliga sa respecte conditiile de desfasurare a examenului intermediar incluse in aceasta Conventie

4. **In timpul desfasurarii examenului intermediar, studentul se obliga :**

   a. sa nu comunice direct sau prin dispozitive electronice cu alte persoane
   b. sa nu salveze pe nici un fel de suport intrebarile/imaginile/paginile incluse in examenul sustinut
   c. sa nu incerce sa transmita prin Intranet/Extranet/Internet continutul intrebarilor, imagini sau pagini web continute in examen sau aflate pe calculatorul de pe care sustine examenul
   d. sa nu utilizeze materiale scrise/inregistrate pentru a afla raspunusuri la intrebarile incluse in examen
   e. sa nu utilizeze dispozitive de calcul (de tip calculatorul inclus in telefon, ceas, sistemul de operare, etc..) pentru a efectua operatii de orice tip
   f. sa nu aiba asupra sa dispozitive de inregistrare audio/video (de tip telefon celular,etc..)
   g. sa nu afecteze prin tinuta sau comportament desfasurarea examenului final

5. **Dupa terminarea examenului intermediar, studentul :**

   a. iese din cont (Logout)
   b. preda ciorna
   c. paraseste sala de examen

6. **Este obligatoriu examenul intermediar?**

   a. conform contractului examenul intermediar este obligatoriu.
   b. cei care nu sustin examenul intermediar si nu achita rata a 2-a (in cazul in care au platit partial cursul) nu mai sunt eligibili pentru participarea la cursuri/laboratoare si examen final.

7. **Incalcarea Conventiei**

   a. incalcarea prevederilor articolului 4 a,b,c,d,e,f duce la eliminarea din examenul intermediar si din academie cu pierderea oricaror drepturi asupra modulului al carui examen intermediar se sustine.
   b. incalcarea prevederilor articolului 4.g duce la eliminarea studentului din examenul intermediar cu pierderea uneia din cele doua sanse de sustinere a acestuia

Semnatarul acestei conventii declara ca a inteles pe deplin termenii conventiei, ca este de acord sa o respecte intocmai si ca se supune sanctiunilor acestei conventii in vigoare in cazul nerespectarii ei. Prezentarea la examenul intermediar echivaleaza cu acceptarea deplina a acestei convetii si cu semnarea ei virtuala.