7 LUCRUL CU STRING SI REGULAR EXPRESSIONS

7.1 Introducerea de valori in interiorul string-urilor

7.1.1 Solutie pentru variabile simple si tablouri unidimensionale

Precum se cunoaste, atunci cand un string este definit folosind ghilimele sau constructia heredoc, interpretorul cauta in interiorul sau variabile sau secvente escape si le inlocuieste cu valoarea corespunzatoare:

```
$cale = "c:\\windows\\";      # secventa \\ specifica caracterul \
$anunt = "Calea este $cale\n si este memorata in variabila \$cale";
```

Cand intalneste caracterul $ in cadrul stringului, interpretorul incearca sa formeze un nume de variabila folosind cat mai multe caractere ce-i urmeaza; el se va opri la primul caracter care nu este valid pentru un nume de variabila: in exemplul de mai sus, din $cale\n va fi considerat ca nume de variabila $cale, deoarece caracterul \n este primul de dupa $ care nu poate aparea intr-un nume de variabila.

In acest fel se pot include in string-uri si valori ale unor elemente de tablou, insa, **daca cheia este de tip string, ea trebuie specificata fara ghilimele sau apostroafe**. Paranteza dreapta de inchidere] este cea care marcheaza sfarsitul sirului ce trebuie interpretat ca variabila si substituit cu valoarea corespunzatoare:

```
$a = array(0, "unu"=>1);
echo "Primul element este $a[0]";         // functioneaza; afiseaza Primul element este 0
echo "Al doilea element este $a['unu']"; // eroare de sintaxa
echo "Al doilea element este $a[\"unu\"]";// eroare de sintaxa
echo "Al doilea element este $a[unu]";    // varianta corecta, fara apostroafe sau ghilimele
```

Acest mecanism nu mai functioneaza insa atunci cand vine vorba de tablouri multidimensionale sau alte expresii mai complexe:

```
$a = array('a'=>array("Andrei","Adrian"), 'b' => array("Bogdan", "Barbu") );
$b = "Il sun pe $a[a][0]"; echo $b;
// afiseaza: Il sun pe Array[0], deoarece este substituit doar $a[a] cu valoarea rezultata
// din conversia sa la string, adica Array
```

7.1.2 Solutie pentru tablouri multidimensionale si alte variabile mai complexe

Aceasta a doua modalitate de a include intr-un string valori de variabile presupune includerea intre acolade {} a expresiei ce se doreste inlocuita cu valoarea ei. In interiorul acoladelor, variabilele – fie ele variabile simple, elemente de tablou, membri de obiect etc – se scriu in acelasi fel in care s-ar scrie in afara string-ului:

```
$a = array('a'=>array("Andrei","Adrian"), 'b' => array("Bogdan", "Barbu") );
echo "El este {$a['b'][0]}, prietenul meu";
// afiseaza: El este Bogdan, prietenul meu
```

In exemplul de mai sus, obtinem acelasi efect - si folosind aceeasi sintaxa a variabilei - ca in cazul compunerii sirului prin concatenare (aici nemaifolosind insa acolade):

```
echo "El este " . $a['b'][0] . ", prietenul meu";
```

Atentie! *Nu caracterul { este cel cu semnificatie speciala si care determina inlocuirea intregii expresii cu valoarea sa, ci combinatia {$. In consecinta, daca din greseala este introdus un spatiu intre { si $, sirul rezultat va fi interpretat altfel:*

```
echo "El este { $a['b'][0]}";
// eroare de sintaxa, deoarece { nu mai are rol special (e un simplu caracter).
// Interpretorul va incerca sa formeze un nume de variabila incepand cu caracterul $ insa
// va da eroare din cauza cheii 'b' care are apostroafe

// Fara apostroafe, este substituit $a['b'] cu valoarea sa convertita la string:
echo "El este { $a[b][0]}";   // afiseaza: El este Array[0]
```

7.2 FUNCTII PREDEFINITE SI OPERATII UZUALE CU STRINGURI

7.2.1 Determinarea lungimii unui string

Functia **strlen()** este cea care intoarce lungimea unui string. Aceasta functie este binary-safe (valoarea stringului poate fi o insiruire de octeti care nu sunt neaparat caractere printabile).

```
$s = "InfoAcademy";
echo strlen($s);  // 11
```

7.2.2 Extragerea unui subsir

Functia folosita in acest scop este **substr**($string, $start [, $length]). Primul argument desemneaza sirul sursa, al doilea pozitia de la care incepe subsirul dorit (numerotand de la 0), iar cel de-al treilea – optional – numarul de caractere al acestui subsir. Functia returneaza subsirul dorit sau FALSE in caz de eroare:

```
$s = "Sic transit gloria mundi";
echo substr($s,4,7);    // transit
echo substr($s,4);      // transit gloria mundi
```

Particularitati:

- daca $start este negativ, subsirul va incepe cu al $start-lea caracter inainte de sfarsitul sirului sursa
- daca $length este negativ, limita dreapta a subsirului va fi stabilita omitand $length caractere din sirul sursa

```
$s = "Sara pe deal";
echo substr($s,-4);     // deal (ultimele 4 caractere)
echo substr($s,5,-5);   // pe (incepe de la pozitia 5 si se omit ultimele 5 pozitii)
```

7.2.3 Cautare in string

7.2.3.1 Aflarea pozitiei unui subsir

Principalele functii folosite in acest scop sunt:

- **strpos**($string, $sir_cautat [, $offset]) – returneaza pozitia primei aparitii a lui $sir_cautat in cadrul lui $string, sau FALSE in cazul in care $sir_cautat nu este gasit. Daca se specifica si al treilea argument, cautarea subsirului se va face incepand cu pozitia $offset
- **stripos**($string, $sir_cautat [, $offset]) – idem, insa cautarea se face case-insensitive
- **strrpos**($string, $sir_cautat [, $offset]) – returneaza pozitia *ultimei* aparitii a lui $sir_cautat in $string, sau false
- **strripos**($string, $sir_cautat [, $offset]) – idem cu precedenta, insa case-insensitive

```
$s = "Contele de Monte Cristo";
echo strpos($s, "te");        // 3 (prima aparitie in cadrul intregului sir)
echo strrpos($s, "te");       // 14 (ultima aparitie in cadrul intregului sir)
echo strpos($s, "te", 4);     // 14 (prima aparitie dupa pozitia 4 a sirului)
```

```
var_dump(strpos($s,"conte")); // bool (false), pentru ca strpos este case-sensitive
echo stripos($s,"Conte");     // 0
```

Atentie cand verificati rezultatul intors de aceste functii! Atunci cand subsirul cautat se afla la inceputul sirului in care se face cautarea, functia strpos() returneaza 0, care este == FALSE; in consecinta, daca folosim operatorul == pentru a determina daca subsirul a fost sau nu gasit, riscam sa tragem concluzii false:

```
$s = "Sfarleaza cu fofeaza";
if(strpos($s,"Sf") == false)        // subsirul "Sf" se gaseste pe pozitia 0, care == false!
      echo "Nu a fost gasit!";      // aceasta linie se va executa!
```

In concluzie, se impune folosirea operatorului === pentru compararea valorii returnate cu FALSE.

7.2.3.2 Afisarea incepand cu prima aparitie a unui subsir sau caracter

Exista cateva functii care afiseaza portiunea unui sir ce incepe cu un anumit subsir sau caracter:

- **strstr**($string, $sir_cautat) – returneaza portiunea din $string cuprinsa intre prima aparitie a lui $sir_cautat si sfarsitul lui $string
- **strchr()** – alias pentru strstr()
- **stristr**($string, $sir_cautat) – idem, dar cautarea lui $sir_cautat se face case-insensitive
- **strrchr**($string,$sir_cautat) – returneaza portiunea din $string cuprinsa intre *ultima* aparitie a lui $sir cautat si sfarsitul lui $string. ***Atentie!*** *Aceasta functie NU este oglindirea lui strstr(), ci poate cauta numai caractere individuale; daca $sir_cautat contine mai multe caractere, functia il va cauta numai pe primul dintre ele!*

```
$s = "Oameni si soareci";
echo strstr($s, "oa");  // oareci
echo stristr($s, "oa"); // Oameni si soareci
echo strrchr($s, "re"); // reci
echo strrchr($s, "en"); // eci (este cautat doar caracterul e; se afiseaza ultima aparitie)
```

7.2.4 Comparare de stringuri

7.2.4.1 Cu operatori de comparare

Pentru determinarea egalitatii/non-egalitatii a doua stringuri sau pentru o comparare alfabetica se pot folosi operatorii PHP de comparare, tinand insa cont de urmatoarele particularitati ale acestora:

- daca unul dintre operanzi este numar si celalalt string, string-ul este convertit la numar si comparatia se face numeric
- daca operanzii sunt string insa contin reprezentari valide de numere intregi zecimale, compararea se face numeric

Iata cateva exemple:

```
$s1 = "Andrei";
$s2 = "Bogdan";
$s3 = "6 din 49";
var_dump($s1 == $s2);   // bool(false)
var_dump($s1 == 0);     // bool(true) !! (operandul stang este convertit la int, dand 0)
var_dump($s1 === 0);    // bool(false)  (nu se fac conversii)
var_dump($s2>0);        // bool(false)  (primul operand este convertit la int, dand 0)
var_dump($s3 >5);       // bool(true)   (primul operand este convertit la int, dand 6)
```

7.2.4.2 Cu functii de comparare

Iata principalele functii PHP predefinite folosite pentru compararea de string-uri:

- **strcmp**($s1, $s2) – returneaza un intreg care are valoarea 0 in cazul egalitatii celor doua siruri, o valoare negativa daca $s1<$s2 (alfabetic) si pozitiva daca $s1 > $s2.

__Nota:__ compararea sirurilor se face case-sensitive, caracter cu caracter, comparand codurile caracterelor. Modul in care sunt alocate codurile caracterelor are urmatoarele consecinte:
  - *majusculele sunt considerate mai mici decat minusculele (deoarece au coduri mai mici)*
  - *numerele sunt considerate mai mici decat literele*

- **strcasecmp**($s1,$s2) – idem, dar compararea se face case-insensitive
- **strncmp**($s1, $s2, $nr) – analog cu strcmp(), insa compara doar primele $nr caractere ale celor doua siruri
- **strncasecmp**($s1, $s2, $nr) – idem, case-insensitive
- **strnatcmp**($s1, $s2) – comparare naturala, case-sensitive, a celor doua siruri (ca natsort() de la tablouri)
- **strnatcasecmp**($s1, $s2) – idem, dar case-insensitive

```
$s1 = "1 RON";
$s2 = "2 RON";
$s3 = "10 RON";
$s4 = "10 ron";
echo strcmp($s1,$s2);          // -1 ($s1 se afla alfabetic inaintea lui $s2)
echo strcmp($s3,$s4);          // -1, (codul caracterului R este mai mic decat al lui r)
echo strcasecmp($s3,$s4);      // 0 (case insensitive,cele doua siruri sunt egale)
echo strcmp($s2,$s3);          // 1 (caracterul 1 se afla inaintea lui 2)
echo strnatcmp($s2,$s3);       // -1 (ca numar, 2 se afla inaintea lui 10)
echo strncmp($s1,$s4,1);       // 0 (comparand numai primul caracter, avem egalitate)
```

7.2.5 Inlocuire de subsiruri si caractere

Iata principalele functii care permit inlocuirea unor portiuni dintr-un string cu altele:

- **str_replace**($sir_cautat, $inlocuitor, $string) – returneaza rezultatul inlocuirii tuturor aparitiilor lui $sir_cautat cu $inlocuitor in cadrului sirului $string. Cautarea se face case-sensitive.
- **str_ireplace**($sir_cautat, $inlocuitor, $string) – idem, dar cu cautare case-insensitive
- **substr_replace**($string, $inlocuitor, $start [, $nr_caractere]) – returneaza o copie a lui $string in care s-a inlocuit portiunea specificata cu $inlocuitor. Specificarea portiunii ce se doreste inlocuita se face fie precizand numai pozitia de inceput (pozitia de sfarsit fiind automat finalul lui $string), fie precizand si numarul de caractere ce trebuie inlocuite incepand cu $start.
- **strtr**($string, $de_inlocuit, $inlocuitor) – returneaza o copie a lui $string in care fiecare caracter din $de_inlocuit se inlocuieste cu caracterul corespunzator din $inlocuitor

```
$s = "Studiez Java";
echo str_replace("Java", "PHP", $s);  // Studiez PHP
echo str_replace("java", "PHP", $s);  // Studiez Java (sirul "java" nu a fost gasit in $s)
echo str_ireplace("java", "PHP", $s); // Studiez PHP
echo substr_replace($s, "PHP", 8);    // Studiez PHP
echo substr_replace($s, "PHP", 8, 4); // Studiez PHP
echo strtr($s, "av", "AV");           // Studiez JAVA (se inlocuieste a cu A si v cu V)
```

7.2.6 Conversii intre litere mici si mari

Functiile PHP predefinite ce realizeaza astfel de operatii sunt:

- **strtoupper**($s) – returneaza sirul $s cu toate literele convertite la majuscule
- **strtolower**($s) – idem, dar transforma toate literele in litere mici
- **ucfirst**($s) – returneaza sirul $s cu prima litera transformata in majuscula prima litera a sirului ("uppercase first")
- **ucwords**($s) – transforma in majuscula prima litera a fiecarui cuvant

```
$s = "file de poveste";
echo strtoupper($s);    // FILE DE POVESTE
echo ucfirst($s);       // File de poveste
echo ucwords($s);       // File De Poveste
```

7.2.7 Eliminarea spatiilor excedentare sau a altor caractere nedorite

Deseori este nevoie de eliminarea spatiilor goale ce bordeaza informatia utila dintr-un string. Spre exemplu, sirul de caractere provenit dintr-un element de tip text input al unui formular HTML, in care utilizatorul introduce din greseala spatiu inainte sau dupa numele sau.

Functiile PHP predefinite care realizeaza acest fel de operatie au o lista de caractere din oficiu pe care le elimina, insa dispun si de un al doilea argument ce permite programatorului sa specifice lista de caractere nedorite:

- **trim**($s [, $caractere_nedorite]) – elimina caracterele nedorite de la inceputul si sfarsitul sirului primit ca prim argument. Caracterele eliminate din oficiu sunt spatiu, tab ("\t"), newline ("\n"), carriage return ("\r"), codul ASCII 0 si codul ASCII 11 (tab vertical). Daca se specifica si al doilea argument al functiei, vor fi eliminate doar caracterele precizate.
- **ltrim**($s [, $caractere_nedorite]) – analog cu trim(), insa opereaza numai asupra inceputului stringului
- **rtrim**($s [, $caractere_nedorite]) si aliasul sau **chop**() – idem, dar pt sfarsitul stringului (capatul drept0

```
$s = "  O sama de cuvinte  ";
echo "|" . trim($s) . "|";    // |O sama de cuvinte| (spatiile dispar)
echo trim($s, "Oams ");       // de cuvinte (O,s,m,a si spatiul dispar din ambele capete)
```

7.2.8 Formatare

7.2.8.1 Formatarea numerelor

PHP pune la dispozitia programatorului urmatoarea functie predefinita:

```
number_format(float $number [, int $decimals [, string $dec_point, string $thousands_sep]])
```

Functia poate primi 1,2 sau 4 argumente (nu si 3!). Primul argument este numarul ce se doreste formatat, al doilea reprezinta numarul de zecimale cu care acesta trebuie afisat, iar ultimele doua precizeaza separatorul zecimal si separatorul de mii:

```
$pi = 3.14159265358979323846;

// Pi cu 5 zecimale
echo number_format($pi, 5);                    // 3.14159

// Pi*1000 cu doua zecimale, virgula ca separator zecimal si spatiu ca separator de mii
echo number_format(1000*$pi, 2, ',', ' ');      // 3 141,59
```

7.2.8.2 Formatare generala

In PHP se regasesc o serie de functii de formatare cunoscute din limbajul C: seria printf(). Aceste functii permit formatarea unui sir de caractere ce contine informatie externa (provenita din variabile, expresii etc):

- **printf**($format, [$date1, [$date2…]]) – afiseaza sirul de caractere rezultat prin inserarea in $format, pe anumite pozitii, a datelor din celelalte argumente, folosind pentru ele o formatare specificata. Returneaza lungimea sirului.
- **sprintf**($format, [$date1, [$date2…]]) – analog cu printf(), insa returneaza string-ul in loc sa il afiseze
- **vsprintf**($format, array $date) – analog cu sprintf(), insa primeste datele sub forma de tablou in loc de argumente separate
- **fprintf**($fisier, $format, [$date1, [$date2…]]) – analog cu printf(), insa rezultatul este scris in fisierul specificat

Functiile din seria printf() lucreaza astfel: argumentul $format este un sir de caractere care poate contine *specificatori de format*; acolo unde apar, acesti specificatori vor fi inlocuiti cu datele pasate ca argumente suplimentare functiei, in ordinea in care apar. Fiecare specificator de format dicteaza cum sa fie tratata valoarea corespunzatoare lui (ca numar sau sir de caractere, cate cifre sau caractere sa aiba, ce precizie etc). Caracterele care nu fac parte din specificatorii de format sunt afisate ca atare.

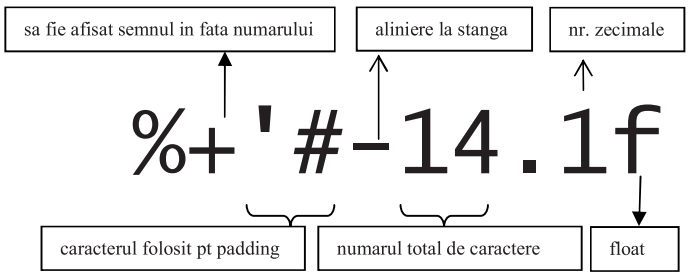
```
$luna = rand(1,12); $zi = rand(1,31);

// ziua si luna sunt afisate cu doua cifre, fiind completate cu 0 daca au o singura cifra
printf("Data este %02d/%02d/2008",$zi,$luna);   // 03/09/2008, 12/04/2008 etc

       specificatori de format
```

Elementele componente ale unui specificator de format sunt:

- caracterul % (obligatoriu), ce reprezinta delimitatorul de inceput al specificatorului. Daca se doreste ca acest caracter sa apara ca atare in sirul generat, este necesara dublarea lui (%%)
- semnul + (optional) – pentru numere, forteaza afisarea semnului numarului si in cazul in care acesta este pozitiv
- caracterul de padding (umplutura) – atunci cand se impune ca o valoare numerica sa fie afisata avand un anumit numar de caractere, iar numarul este prea scurt, el este completat pana la lungimea dorita folosind unul sau mai multe exemplare ale caracterului specificat aici. Caracterul de padding poate fi spatiu (default), 0 sau un alt caracter; in acest ultim caz, caracterul se specifica prin prefixarea cu ' (apostrof)
- alinierea (optional) – daca nu este specificata, alinierea datelor se face la dreapta; un – (minus) pe aceasta pozitie determina alinierea la stanga a datelor (vezi mai jos)
- numarul minim de caractere pe care sa il aiba valoarea formatata (optional). Daca numarul de caractere al valorii este mai mic decat acest minim, valoarea se completeaza cu caractere de padding, adaugate la stanga sau la dreapta, in functie de alinierea dorita



- precizia (optional), in cazul numerelor de tip float. Aplicata unui string, stabileste numarul maxim de caractere care se afiseaza din string-ul respectiv
- tipul de date – modul in care sunt tratate datele externe (ca string, numar etc). Exemple:
  - %d – intreg zecimal
  - %f – float
  - %s – string

```
$i = 12.534;
printf("\$i este %+'#-14.1f",$i);   // $i este +12.5#########
```

7.3 REGULAR EXPRESSIONS (REGEX)

7.3.1 Concepte de baza

Regular expressions sunt expresii care ne permit sa specificam formatul unui sir de caractere. Scenariile in care apare nevoia de regex-uri sunt in general doua:

- cand dorim sa cautam si sa extragem siruri de caractere cu format cunoscut dintr-un text mai mare. Exemplu: intentionam sa obtinem doar adresele de e-mail dintr-un fisier text care contine diverse informatii despre o lista de utilizatori.
- cand dorim sa validam date (sa ne asiguram ca anumite informatii au format corect). Exemplu: datele introduse de catre utilizator intr-un formular HTML trebuie sa respecte anumite cerinte – datele calendaristice sa fie formatate intr-un anume fel, numele de persoane sa contina nume si prenume separate prin spatiu sau -, etc.

Exista deosebiri fundamentale intre cautarea obisnuita si cea folosind regex-uri:

- in cautarea obisnuita, cunoastem de la bun inceput sirul de caractere ce va fi gasit, iar ceea ce ne intereseaza sunt informatii suplimentare (pozitia sau pozitiile aparitiei lui, frecventa de aparitie etc)
- in cautarea cu regex-uri, nu stim dinainte sirurile de caractere pe care le vom gasi, ci vom obtine toate sirurile corespunzatoare formatului specificat. Asadar scopul cautarii, spre deosebire de cautarea obisnuita, poate fi chiar lista de siruri al caror format este precizat in regex

7.3.2 PCRE (Perl-compatible Regular Expressions)

7.3.2.1 Generalitati. Delimitare

Un regex este format dintr-o succesiune de caractere (litere, cifre, semne de punctuatie), insa cu urmatoarele particularitati:

- intreaga expresie este cuprinsa intre doua caractere delimitatoare. Caracterul delimitator de inceput este acelasi cu cel de sfarsit si este ales de catre programator. Acest caracter nu are voie sa fie alfanumeric sau \ .

```
/regex/ - modalitatea traditionala de delimitare a unui regex
#regex# - modalitate alternativa de delimitare
\regex\ - modalitate invalida
```

Nota: *daca in interiorul regex-ului este folosit caracterul delimitator, el trebuie precedat de un* \.

- unele caractere sau combinatii de caractere au in interiorul regex-ului semnificatie speciala. Scopul acestor constructii speciale este sa ne dea posibilitatea de a specifica formatul – iata doua exemple:
  - clase de caractere - litera mare, litera mica, caractere permise pe o anumita pozitie a sirului. (ex: numerele de masina sunt de forma B 80 TGF sau TR 46 FBJ, adica una sau doua litere mari, un spatiu, doua cifre, un spatiu, trei litere mari)
  - repetitii ale unui caracter sau sub-expresii (ex: adresa IP, scrisa in general sub forma 192.168.0.15, poate fi scrisa sub forma unor grupuri de 1-3 cifre care se repeta)

In continuare vor fi prezentate principalele constructii speciale si metacaractere folosite pentru construirea de regex-uri.

7.3.2.2 Clase de caractere

Clasele de caractere sunt constructii care, folosite in interiorul unui regex, indica faptul ca pe respectiva pozitie din sirul cautat trebuie sa se afle un caracter apartinand unui anume set specificat. Exista trei tipuri de metacaractere sau constructii folosite in acest scop:

- constructia cu paranteze drepte [ ] – tine locul unui singur caracter. Ne permite sa specificam un set discret de caractere ce se pot afla pe o anumita pozitie din sir. Intre paranteze, caracterul ^ (accent circumflex) are rol de negare. Exemple:

Regex	Siruri ce corespund	Comentarii, explicatii
[flp]in	fin, lin, pin	constructia [flp] corespunde unui singur caracter, si anume unuia dintre cele specificate in interiorul parantezelor
[r-t]ara	rara, sara, tara	constructia [r-t] corespunde unui singur caracter, mai exact unuia dintre cele aflate intre r si t (r,s,t)
[br-tv]	bara, rara, sara, tara, vara	pot fi combinate primele doua modalitati. Indiferent de cate caractere se afla inauntrul parantezelor, constructia tine loc de un singur caracter (in acest caz, acela poate fi b,r,s,t sau v)
[A-Z][a-z]	orice cuvant de doua litere care incepe cu litera mare	constructia [] poate fi folosita de oricate ori este nevoie in cadrul unui regex
[A-Z][^a-z]	orice sir de doua litere care incepe cu litera mare si are pe pozitia a doua orice altceva decat litera mica (ex: C#, A4 etc)	[^a-z] corespunde unui singur caracter, care nu poate fi litera mica. **Atentie!** Negarea unei litere sau a unui set de litere inseamna ca pe pozitia respectiva se pot gasi cifre, semne de punctuatie etc

- punctul – tine locul unui singur caracter. Indica faptul ca pe respectiva pozitie din sir se poate gasi orice caracter (insa unul singur!) cu exceptia newline (\n). Daca este activata optiunea DOT-ALL (vezi mai jos modificatori), punctul corespunde si caracterelor newline.

Regex	Siruri ce corespund	Comentarii, explicatii
.in	*fin, lin, pin* dar si *#in, %in* etc	. inseamna orice caracter, inclusiv semne de punctuatie, cifre etc
[r-t]a.a	*rara, sara, tara* dar si *rata, raba, tata, ta(a, sa@a etc*	punctul poate fi combinat cu una dintre celelalte constructii
[A-Z].	o litera mare urmata de orice caracter (ex: *Am, Nu, F&, H** etc)	

Nota: *pentru a specifica chiar caracterul punct pe una dintre pozitiile sirului cautat, este necesara precedarea lui cu* \.

- clase de caractere uzuale, predefinite. Iata cateva exemple:
  - \d – digit (cifra zecimala)
  - \D – non-digit (un caracter care nu este cifra zecimala)
  - \s – whitespace. Corespunde caracterelor spatiu, tab (\t) si newline (\n)
  - \S – non-whitespace
  - \w – word character (litera, cifra sau underscore)
  - \W – non-word character

Regex	Siruri ce corespund	Comentarii, explicatii
\d-\d	un scor la fotbal: *3-2, 1-0* etc	cifra, minus, cifra. Regex-ul nu ar functiona pentru scoruri ce depasesc 10 (ex: 10-2, 15-40)
B\s\d\d\s[A-Z][A-Z][A-Z]	un numar de automobil de Bucuresti (ex: B 13 RTG)	caracterul B, spatiu, doua cifre, spatiu, 3 litere mari. De remarcat insa ca \s corespunde si cu TAB sau NEWLINE, asadar aceasta expresie nu ar selecta numai numere de masina, ci si succesiuni de 3 linii care contin B pe linia 1, doua cifre pe linia 2 si 3 litere pe linia 3

7.3.2.3 Repetitii

Putem specifica repetitia controlata a unui caracter sau a unei intregi subexpresii folosind doua constructii:

- constructia cu acolade:

```
caracter{min,max} – caracterul se repeta intre min si max ori
(regex){min,max} – intreaga expresie se repeta intre min si max ori
```

Sunt permise variatiuni ale acestei sintaxe:

```
(regex){n} – repetitie de exact n ori
(regex){min,} – repetitie de minim n ori, fara limita superioara a numarului de repetitii
```

Observatie: *atunci cand dorim sa punem conditia de repetare a unei parti a regexului ce contine mai multe caractere, acea subexpresie trebuie inclusa intre paranteze rotunde.*

Exemple:

Regex	Siruri ce corespund	Comentarii, explicatii
07\d{8}	un numar de mobil (ex: *0720123456*)	caracterele 0 si 7 urmate de 8 cifre
[A-Z]{2}\d{6}	serie si numar de buletin (ex: *VF735245)*	o litera mare care se repeta de exact 2 ori, urmata de o cifra care se repeta de exact 6 ori
[A-Z][a-z]{0,1}	un element chimic (ex: *H, Na* etc)	o litera mare, urmata de o litera mica ce apare o data sau deloc
[A-Z]{1,2}\s\d{2}\s[A-Z]{3}	un numar de automobil (ex: *B 35 EDX* sau *MM 67 WSL*)	una sau doua litere mari, un spatiu, doua cifre, un spatiu si apoi trei litere mari
[A-Z][a-z]{1,}	un cuvant care incepe cu litera mare (ex: Marius)	

- metacaractere folosite pentru cazuri de repetitii particulare

```
(regex)+ - echivalent cu (regex){1,} (minim o repetitie a lui regex)
(regex)* - echivalent cu (regex){0,} (0 sau mai multe repetitii ale lui regex)
(regex)? - echivalent cu (regex){0,1} (regex apare o data sau deloc)
```

Exemple:

Regex	Siruri ce corespund	Comentarii, explicatii
[A-Z][a-z\s]+\.	o propozitie (ex: Am un mar.)	o litera mare urmata de una sau mai multe litere mici sau spatii si terminandu-se cu un punct
[a-z]+\.[a-z]+@[a-z]\.[a-z]{2-4}	o adresa de mail de forma victor.manu@gmail.com	o litera mica care se repeta minim o data, un caracter . (observati \-ul), o litera mica care se repeta cel putin o data, caracterul @, din nou o succesiune de litere mici (cel putin una) apoi punct si numele domeniului radacina (com, net, org, info, tv etc)

7.3.2.4 Conditii de ancorare

Putem pune conditia ca sirul de caractere descris in regex sa se afle la inceputul sau la sfarsitul textului in care il cautam, cu ajutorul a doua caractere speciale:

- caracterul ^ - daca apare la inceputul regex-ului, pune conditia ca sirul de caractere gasit sa se afle la inceputul textului in care se face cautarea
- caracterul $ - daca apare la sfarsitul regex-ului, impune ca sirul de caractere ce corespunde regex-ului sa se afle la finalul textului in care se face cautarea

Observatie: *atunci cand ^ apare in interiorul parantezelor drepte (la specificarea unei clase de caractere), el are alta semnificatie: cea de negare.*

Daca in regex se activeaza optiunea MULTILINE (vezi mai jos modificatori), atunci ^ va corespunde fiecarui inceput de linie, iar $ fiecarui sfarsit de linie, daca textul in care se face cautarea este unul de mai multe linii. Optiunea MULTILINE este initial dezactivata, conditii in care ^ corespunde inceputului intregului text iar $ sfarsitului aceluiasi text.

Exemplu: cautand sirurile de caractere ce corespund regex-ului *ninge$* pe textul urmator:

```
Ziua ninge
noaptea ninge
dimineata ninge iara
```

vor fi gasite urmatoarele siruri, in functie de caz:
- daca optiunea MULTILINE este activata, va fi gasit sirul *ninge* de doua ori - la finalul primei linii si al celei de-a doua. Cuvantul *ninge* de pe linia 3 nu este gasit deoarece nu se afla la sfarsit de linie
- daca optiunea MULTILINE este dezactivata, nu va fi gasit nici un sir, deoarece $ corespunde finalului intregului text, iar textul nu se termina cu sirul *ninge* .

7.3.2.5 Modificatori de optiuni

Regex-ul poate contine si elemente care nu desemneaza caractere, clase de caractere sau repetitii, ci specifica optiuni aplicate motorului de regular expressions in cazul expresiei respective. Modificatorii se pot specifica cu urmatoarea constructie in interiorul unui regex:

```
(?optiuni) - activarea uneia sau mai multor optiuni
(?-optiuni) - dezactivarea uneia sau mai multor optiuni
(?optiuni1-optiuni2) - activarea optiunilor din grupul 1 si dezactivarea celor din grupul 2
```

Iata cateva optiuni uzuale:
- CASELESS (i) – in aplicarea regex-ului in cauza nu se va mai face distinctie intre literele mici si mari
- DOT-ALL (s) – punctul va corespunde oricarui caracter, *inclusiv newline*
- MULTILINE (m) – caracterele ^ si $ nu vor mai corespunde doar inceputului si sfarsitului intregului text in care se cauta, ci fiecarui inceput si sfarsit de linie componenta

Exemple:

Regex	Siruri ce corespund	Comentarii, explicatii
(?i)php	php, PHP	optiune utila atunci cand cautam o anumita succesiune de litere, indiferent daca sunt mici sau mari; fara aceasta optiune am fi fost fortati sa scriem [Pp][Hh][Pp]
(?im)php$	php sau PHP aflate la sfarsit de linie	pot fi activate sau dezactivate mai multe optiuni simultan
ab(?i)cd(?-i)ef	abcdef, abCDef, abCdef, abcDef	optiunile pot fi activate numai pe o portiune a regex-ului (in cazul nostru, CASELESS este activ numai pentru literele c si d)

7.3.2.6 Specificarea de formate alternative

In cadrul unui regex, caracterul special | permite specificarea unor formate alternative pentru sirul de caractere cautat/validat sau pentru o portiune a sa:

```
regex1 | regex2 → caracterul | are rol de "sau"
inceput_regex (regex1|regex2) sfarsit_regex → portiunea de mijloc poate avea doua formate
```

Exemple:

Regex	Siruri ce corespund	Comentarii, explicatii
Mări(e|oara)	Mărie, Mărioara	finalul sirului cautat are doua formate posibile
(021|07\d)\d{7}	un numar de Bucuresti (fix) sau de mobil	021 urmat de 7 cifre, sau 07 urmat de 8 cifre
([fs]|vs|)printf	suita de functii printf din PHP (printf, fprintf, sprintf, vsprintf)	sirul incepe cu f, cu s, cu vs sau cu nimic (remarcati caracterul | de dinante de paranteza rotunda inchisa) urmat de *printf*.
(f|v?s|)printf		inaintea lui printf se poate afla: nimic, caracterul f, sau caracterul s precedat sau nu de un v

7.3.2.7 Sub-expresii

Sub-expresiile reprezinta portiuni ale unui regex delimitate prin paranteze rotunde si care pot fi extrase separat sau referite chiar din cadrul regex-ului. Ele sunt utile in doua cazuri:

- cand cautam un sir de caractere corespunzator unui regex insa ne intereseaza numai o parte a sa, care nu putea fi cautata independent (un subsir). Exemplu: dorim sa extragem toate link-urile dintr-un fisier HTML; pentru aceasta cautam toate sirurile de forma … si extragem din ele numai valoarea atributului href
- cand dorim ca, din cadrul unui regex, sa ne referim la o portiune anterioara a sa ("*back references*"). Exemple: a) un regex! incepe cu un caracter (ales de catre programator, asadar necunoscut) dar se termina cu acelasi caracter. b) un tag HTML, care are delimitatori de deschidere si de inchidere, ultimul putand fi specificat prin referire la primul

```
(\d{3})\d{7} – se defineste o singura subexpresie. In cazul unui numar de telefon, ea ar
selecta primele 3 cifre, care pot fi folosite pentru a decide daca numarul este de fix sau
mobil si de ce operator de telefonie apartine
```

Subexpresiile primesc automat numere incepand de la 1, in ordinea in care apar in regex, pentru a oferi posibilitatea referirii lor din cadrul regex-ului si a extragerii sirurilor de caractere ce le corespund. Daca se doreste referirea unei subexpresii din cadrul regex-ului, se pot folosi secventele \1, \2 …\99 ce semnifica sirul de caractere ce a corespuns primei subexpresii, celei de-a doua subexpresii etc

Exemplu: aplicandu-se regex-ul *^([A-Z]+) cel ([a-z]+)$* pentru sirurile de caractere din prima coloana a tabelului de mai jos, continuturile subexpresiilor sunt cele din coloanele 2 si 3:

	Subexpresia 1 ([A-Z\s]+)	**Subexpresia 2 ([a-z]+)**
"Andrii Popa cel voinic"	Andrii Popa	voinic
"Stefan cel mare"	Stefan	mare
"Mircea cel batran"	Mircea	batran

Exemplu – un tag HTML cu delimitatori de deschidere si de inchidere:

```
<(?i)([A-Z]+)>[^<]*</ \1>
sirul incepe cu <, se activeaza optiunea CASELESS, continua cu una sau mai multe litere
(mari sau mici) ce constituie prima subexpresie. Urmeaza >, apoi 0 sau mai multe caractere
diferite de <, si tagul de inchidere, format din </ si apoi sirul ce a corespuns primei
subexpresii (ex: <td>……</td>)
```

Exista cazuri in care dorim sa includem o portiune a regex-ului intre paranteze, insa fara ca ea sa fie automat considerata (si numerotata) ca sub-expresie (ex: (021|07\d)\d{7}, unde parantezele sunt folosite numai pentru specificarea formatului alternativ al inceputului de sir). In astfel de cazuri putem folosi secventa (?:) pentru a incadra portiunea de regex dorita; intre ? si : pot fi specificate si optiuni:

```
(?:021|07\d)(\d{7})
se creeaza o singura subexpresie, cea cu numarul 1, corespunzatoare ultimelor 7 cifre

^([A-Z ]+) cel (?i:[a-z]+)$
se creeaza o singura subexpresie;pt ultima parte a regex-ului se activeaza optiunea CASELESS
```

7.3.3 Functii PHP predefinite pentru lucrul cu regex-uri

Iata principalele functii PHP predefinite pentru lucrul cu PCRE:

- int **preg_match** (string $regex, string $string [, array &$matches [, int $flags [, int $offset]]]) – cauta in $string prima aparitie a unui sir ce corespunde formatului specificat in $regex. Returneaza 1 in caz de gasire si 0 in caz contrar. Daca este folosit si al treilea argument, acesta se populeaza astfel: $matches[0] contine intregul sir ce a corespuns regex-ului, $matches[1] sirul corespunzator primei subexpresii etc. Al patrulea argument, daca exista, ofera posibilitatea de a impune pozitia din $string de la care sa inceapa cautarea.

```
$s = "mere pere in panere";
echo preg_match('/ere/', $s); // 1
$a = array();
preg_match('/(.)ere\s(.)/', $s, $a);      // subexpresiile contin cate un singur caracter
var_dump($a); /*
            array(2) {
              [0]=> string(4) "mere p"
              [1]=> string(1) "m"         → sirul corespunzator subexpresiei 1
              [2]=> string(1) "p"         → sirul corespunzator subexpresiei 2
            }*/
```

- int **preg_match_all** (string $regex, string $string, array &$matches [, int $flags [, int $offset]]) – functioneaza la fel ca preg_match, insa al treilea argument este obligatoriu, el fiind populat cu *toate* sirurile gasite care corespund pattern-ului (preg_match se oprea la prima aparitie). Felul in care este populat acest tablou este dat de parametrul $flags, care ofera urmatoarele posibilitati:
    - PREG_PATTERN_ORDER – elementele din $matches corespund subexpresiilor din $regex. $matches[0] va avea ca valoare un tablou cu toate sirurile intregi care au corespuns regex-ului, $matches[1] va avea ca valoare un tablou ce contine toate sirurile ce au corespuns primei subexpresii etc.
    - PREG_SET_ORDER – fiecare element din $matches corespunde unei aparitii de subsir ce corespunde regex-ului, si are ca valoare un tablou cu toate sirurile corespunzatoare subexpresiilor. $matches[0][0] contine in integralitatea sa primul sir care a corespuns regex-ului, $matches[0][1] contine subsirul corepunzator primei subexpresii, $matches[1][0] al doilea sir (integral) ce a corespuns regex-ului etc.

$s = "mere pere in panere";
$a = array();
preg_match_all('/(.)er(e)/', $s, $a, PREG_PATTERN_ORDER);
var_dump($a);
	$s = "mere pere in panere";
	$a = array();
	preg_match_all('/(.)er(e)/', $s, $a, PREG_SET_ORDER);
	var_dump($a);

array(3) {
[0]=> array(3) {
[0]=> string(4) "mere"
[1]=> string(4) "pere"
[2]=> string(4) "nere"
}
[1]=> array(3) {
[0]=> string(1) "m"
[1]=> string(1) "p"
[2]=> string(1) "n"
}
[2]=> array(3) {
[0]=> string(1) "e"
[1]=> string(1) "e"
[2]=> string(1) "e"
}
}
	array(3) {
	[0]=> array(3) {
	[0]=> string(4) "mere"
	[1]=> string(1) "m"
	[2]=> string(1) "e"
	}
	[1]=> array(3) {
	[0]=> string(4) "pere"
	[1]=> string(1) "p"
	[2]=> string(1) "e"
	}
	[2]=> array(3) {
	[0]=> string(4) "nere"
	[1]=> string(1) "n"
	[2]=> string(1) "e"
	}
	}

- mixed **preg_replace** (mixed $regex, mixed $inlocuitor, mixed $string [, int $limit [, int &$count]]) – returneaza o copie a lui $string in care subsirurile ce corespund formatului din $regex sunt inlocuite cu $inlocuitor. Al patrulea argument, daca este prezent, specifica numarul maxim de inlocuiri, iar in ultimul argument se memoreaza numarul de inlocuiri efectuate.
- array **preg_split** (string $pattern, string $subject [, int $limit [, int $flags]]) – alternativa la strtok() sau explode(), cu diferenta ca delimitatorul de campuri (sirul de caractere declarat ca separator) este acum specificat sub forma unui regex

7.4 BIBLIOGRAFIE

- **PHP5 and MySQL Bible (Tim Converse, Joyce Park) – Chapter 8: Strings – pag 137-157**
- **PHP Manual – Language Reference→Types→Strings: http://ro.php.net/manual/en/language.types.string.php**
- **PHP Manual – Function Reference→String Functions: http://ro.php.net/manual/en/book.strings.php**
- **PHP Manual – Regular Expression Functions (PCRE): http://ro.php.net/manual/en/book.pcre.php**
- **http://www.regular-expressions.info/**
- **Mastering Regular Expressions 3rd Ed. (Jeffrey Friedl) – editura O'Reilly**

Conventie

-referitoare la conditiile de desfasurare a examenului final-

1. Caracteristici generale ale examenului final teoretic :

a. se numeste Final Exam
b. este realizat de InfoAcademy din intrebari cuprinse in examenele partiale
c. este accesibil pe http://www.infoacademy.net → Login → Student home → Nume_Clasa → TakeAssesment
d. este de tip test grila si are in medie 30-60 de intrebari cu raspuns unic sau multiplu
e. poate fi sustinut de maxim 2 ori, la interval de 1-2 saptamani intre cele 2 incercari
f. poate fi sustinut numai la sediul academiei InfoAcademy in prezenta coordonatorului stiintific sau a unui instructor delegat de acesta.
g. se considera promovat daca numarul de raspunsuri corecte este cel putin egal cu 80% din numarul total al raspunsurilor
h. Dureaza 60 de minute

2. Conditii de participare la examenul final teoretic :

Poate participa la examenul final teoretic orice student Infoacademy care :

a. A promovat toate examenele partiale cu cel putin 70% (optim 80%)
b. Se afla in termenul contractual (maxim 3 saptamani de la data ultimei predari).
c. A achitat obligatiile financiare convenite la inscriere sau se obliga sa le achite pana la data examenului final.
d. Se prezinta la examenul final la data si ora aleasa in momentul inscrierii on-line la examenul final (prin www.infoacademy.net)
e. Are asupra sa un act cu fotografie care ii atesta identitatea si documentele fiscale care atesta plata obligatiilor financiare catre InfoAcademy (chitanta+factura)
f. Se obliga sa respecte conditiile de desfasurare ale examenului final incluse in aceasta Conventie

3. Inscrierea la examenul final

a. Poate fi facuta prin www.infoacademy.net cu cel mult 4 saptamani inainte de data la care se doreste sustinerea examenului final
b. Este recomandabil sa fie facuta din timp (chiar inainte de finalizarea modulului pentru care se face inscrierea) pentru acomodarea psihologica, pregatirea temeinica a examenului si accesul la data si ora dorite.
c. Se poate renunta la inscrierea facuta, prin ANULAREA INSCRIERII (accesarea linkului de anulare trimis in e-mailul de confirmare a inscrierii, sau folosirea facilitatii de anulare din formularul de inscriere).

4. In timpul desfasurarii examenului final , 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 final, studentul :

a. Iese din cont (Logout), preda ciorna si paraseste sala de examen
b. In maxim o saptamana completeaza cele 2 feedbackuri : Course Feedback si InfoAcademy feedback
   1. Course Feedback este accesibil de pe http://www.infoacademy.net → Login → Nume_Clasa → TakeAssesment si este obligatoriu pentru incheierea situatiei
   2. InfoAcademy feedback este accesibil prin www.infoacademy.net , in tabelul de extraservicii
c. In maxim 3 luni de la data examenului final depune cerere de diploma prin www.infoacademy.net
d. Se prezinta la academie in cel mult o luna de la primirea e-mailului care ii confirma ca poate ridica Certificate of Course Completion
e. Pentru ridicarea diplomei studentul se identifica cu cel putin un act cu fotografie care ii atesta identitatea.

6. Incalcarea CONVENTIEI

a. Incalcarea prevederilor articolului 4 a,b,c,d,e,f duce la eliminarea din examenul final si din academie cu pierderea oricaror drepturi asupra modulului al carui examen final se sustine.
b. Incalcarea prevederilor articolului 4.g duce la eliminarea studentului din examenul final 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 final echivaleaza cu acceptarea prezentei conventii.