CAP. 4
STRUCTURI DE CONTROL

În limbajul PHP, la fel ca în oricare alt limbaj de programare, instrucțiunile cuprinse într-o secvența de cod-sursa se executa succesiv, una după alta. Există însă anumite instrucțiuni care modifică ordinea de execuție a liniilor de cod. Din acest motiv ele poartă numele de structuri de control, întrucât ele controlează fluxul de execuție.

Structurile de control din PHP sunt: structura alternativă (instrucțiunea **if** cu variantele ei), structura de selecție multiplă (**switch**), structurile repetitive (**for, while, do... while, foreach**), structuri de întrerupere a fluxului (**break, continue, return**), structura de salt necondiționat (**goto**), directivele de includere (**include, require**) și directiva declare. O parte dintre acestea sunt rar folosite în practică.

Viața este conturată de tot felul de decizii. La fel este și PHP. Ne este permis să schimbăm rezultatul în funcție de circumstanțe. Deciziile luate în PHP sunt realizate cu ajutorul structurilor de control. Cele mai comune dintre acestea folosesc **if** și au o structură apropiată și în alte limbaje.

4.1 Structura alternativă - if

În viața reală, avem următoarea situație. Dacă vremea este caldă vom merge la plajă. Tradusă în pseudocod, această ar fi descrisă în felul următor:

```
if (vremea este caldă) {
   vom merge la plajă;
}
```

Condiția este testată în interiorul parantezelor și dacă este adevărată se vor executa acțiunile enumerate între { … }. Condiția poate fi orice expresie de genul "2 < 3", "variabila $a este definita", ș.a. traduse în limbajul PHP.

Observație: Parantezele pot conține una sau mai multe condiții legate prin operatori logici. Codul din interiorul acoladelor este executat doar dacă valoarea finală de adevăr este **true**. Dacă este **false**, PHP va ignora codul din interiorul acoladelor și va merge către următoarea secțiune a codului.

Codul de executat poate fi simplu (o singură instrucțiune) sau bloc (mai multe instrucțiuni delimitate de acolade). Regula este că atunci când este nevoie să se execute mai mult de o instrucțiune, trebuie obligatoriu creat un bloc (trebuie folosite acoladele).

```
if (2 < 4) print "2 este mai mic decat 4 <br>";

if (3 > 1) {
    print "3 este mai mare dacat 1";
    print "<br>";
}
```

4.2 Structura if – else

De multe ori este nevoie să se specifice și o operație ce trebuie efectuată dacă nu este îndeplinită o condiție. În acest caz se folosește structura decizională **if – else**. Aceasta are următoarea sintaxă:

if (condiție**) {**
  cod executat în cazul în care condiția este adevărată;
} else {
  cod executat în cazul în care condiția este falsă;
}

Această formă permite executarea unei instrucțiuni atunci când se îndeplinește condiția sau executarea alteia, diferite, în caz contrar:

```
$ora = 21;

if ($ora < "20") {
    echo "Bună ziua!";
} else {
    echo "Bună seara!";
}
```

4.3 Structura elseif

Instrucțiunea **elseif** este o combinație între **if** și **else**. În cazul în care condiția din ramura **if** nu este îndeplinita testează condiția din ramura **elseif**. Dacă nu este îndeplinită a doua condiție se execută instrucțiunile cuprinse de ramura **else**. Sintaxa utilizată pentru această formă a structurii decizionale **if** este următoarea:

if (condiție**) {**
  cod executat în cazul în care condiția este adevărată;
} elseif (condiție2**) {**
  cod executat în cazul în care prima condiție este falsă și a doua condiție este adevărată;
} else {
  cod executat în cazul în care toate condițiile anterioare sunt false;
}

Exemplu de structură decizională cu ramură **elseif**:

```
$ora = 14;

if ($ora < "10") {
    echo "Bună dimineața!";
} elseif ($ora < "20") {
    echo "Bună ziua!";
} else {
    echo "Bună seara!";
}
```

Observație: Ramura else poate să lipsească.

4.4 Structura de selecție multiplă - switch

O instrucțiune ce se aseamănă cu **if** este **switch**, o structură ce permite executarea uneia sau mai multor linii de cod în funcție de valoarea unei expresii testate. Instrucțiunea **switch** este utilă în cazurile în care este nevoie să se verifice mai multe valori posibile ale unei expresii.

Sintaxa structurii de selecție multiplă – switch este următoarea:

```
switch (expresie_testată) {
  case valoare1:
    cod executat dacă expresie_testată = valoare1;
    break;
  case valoare2:
    cod executat dacă expresie_testată = valoare2;
    break;
  case valoare3:
    cod executat dacă expresie_testată = valoare3;
    break;
  ...
  default:
    cod executat dacă expresie_testată este diferită de toate valorile anterioare;
}
```

Acesta este modul în care funcționează: mai întâi avem *expresie_testată* (cel mai adesea o variabilă), care este evaluată o dată. Valoarea expresiei este apoi comparată cu valorile pentru fiecare caz din structură. Dacă există o potrivire, blocul de cod asociat cu acel caz este executat. Utilizarea instrucțiunilor **break** previne executarea codului din cazul următor. Ramura **default** este utilizată dacă nu se găsește nicio potrivire.

Observație: Ramura **default** poate să lipsească.

Exemplu de utilizarea a structurii **switch**:

```
$var = 3;

switch ($var) {
    case 0:
        print 'zero';
        break;
    case 1:
        print 'unu';
        break;
    case 2:
        print 'doi';
        break;
    case 3:
        print 'trei';     // afișează trei
        break;           // întrerupe fluxul execuției switch
    case 4:
        print 'patru';
```

```
            break;
        case 5:
            print 'cinci';
            break;
    }
```

4.5 Structuri repetitive

Adesea, atunci când scrieți cod, doriți ca același bloc de cod să ruleze din nou și din nou de un anumit număr de ori. Deci, în loc să adăugăm mai multe linii de cod aproape egale într-un script, ne putem folosi de ajutorul structurilor repetitive.

Buclele sunt folosite pentru a executa același bloc de cod din nou și din nou, atâta timp cât o anumită condiție este adevărată.

În PHP, avem următoarele structuri repetitive:

- **while** - trece printr-un bloc de cod atâta timp cât condiția specificată este adevărată;
- **do...while** - trece printr-un bloc de cod o dată, apoi repetă bucla atâta timp cât condiția specificată este adevărată;
- **for** - trece printr-un bloc de cod de un număr specificat de ori;
- **foreach** - trece printr-un bloc de cod pentru fiecare element dintr-un vector.

4.5.1 Structura repetitivă – while

Majoritatea script-urilor PHP au nevoie ca aceleași secvențe de cod să fie executate de mai multe ori. Structura repetitivă **while** permite executarea unui bloc de cod atât timp cât o condiție este îndeplinită (este adevărată).

while (condiția este adevărată) **{**
  cod executat;
}

Observație: Este posibil ca instrucțiunile din blocul while să nu fie executate niciodată.

Exemplu de utilizarea a structurii **while**:

```
$x = 1;         // inițializați contorul de repetiții

while ($x <= 5) { // continuați repetiția atâta timp cât $x <= 5
    echo "Numărul este: $x <br>";
    $x++;             // creșteți valoarea contorului buclei cu 1
}
```

Observație: Dacă structura de control folosită în cadrul blocului **while** nu devine **false**, bucla va continua la infinit și va bloca execuția programului.

4.5.2 Structura repetitivă – do - while

Uneori, dorim să executăm cel puțin o dată o secvență de cod din cadrul unei structuri repetitive chiar și când structura de control este **false**. În acest scop, în PHP avem la dispoziție structura **do - while** a cărei sintaxă este:

do {
cod executat;
} while (condiția este adevărată);

Singura diferență față de structura **while** este faptul că valoarea condiției este determinată, de fiecare dată, după executarea instrucțiunilor. Ca urmare, instrucțiunile vor fi executate cel puțin o dată.

Exemplu de utilizarea a structurii **do - while**:

```
$x = 6;
do {
    echo "Numărul este: $x <br>";
    $x++;
} while ($x <= 5);
```

4.5.3 Structura repetitivă – for

O alternativă la structurile repetitive **while**, pentru scenariile în care se cunoaște numărul de iterații este structura repetitivă **for**. Sintaxa este foarte asemănătoare cu cea din limbajele C/C++ și Java:

for (initializare contor; testare contor; incrementare contor**) {**
cod executat;
}

Descrierea parametrilor:

- Inițializare contorulu: inițializează valoarea contorului buclei;
- Testare contor: evaluat pentru fiecare iterație a buclei. Dacă se evaluează cu TRUE, bucla continuă. Dacă se evaluează cu FALSE, bucla se încheie;
- Incrementare contor: modifică valoarea contorului buclei.

Exemplu de utilizarea a structurii **for**:

```
for ($x = 0; $x <= 10; $x++) {
    echo "Numărul este: $x <br>";
}
```

Prima expresie este evaluată o singură dată, înainte de începerea execuției buclei. Instrucțiunea este executată cât timp cea de-a doua expresie are valoarea **true**. De fiecare dată, după executarea instrucțiunii este evaluată cea de-a treia expresie. Oricare dintre cele trei expresii poate lipsi; în cazul în care o expresie lipsește, se consideră că ea are valoarea **true**.

4.5.4 Structura repetitivă – foreach

Structura repetitivă **foreach** funcționează doar cu vectori și este folosită pentru a parcurge perechile cheie / valoare conținute de aceștia. Sintaxa acestei structuri este următoarea

```
foreach ($vector as $valoare) {
   cod executat;
}
```

Pentru fiecare iterație de buclă, valoarea elementului curent din **$vector** este atribuită lui **$valoare** și cheia din vector este mutată cu o unitate, până când ajunge la ultimul element din vector.

Exemplu de utilizarea a structurii **foreach**:

```
$culori = array("rosu", "verde", "albastru", "galben");
foreach ($culori as $valoare) {
    echo "$valoare <br>";
}
```

4.6 Instrucțiunile break și continue

Ați văzut deja instrucțiunea **break** (folosită într-un capitol anterior). A fost folosită pentru a „ieși” dintr-o structură **switch**.

Instrucțiunea **break** poate fi folosită și pentru a ieși dintr-o buclă. În exemplul următor se iese din buclă când x este egal cu 4:

```
for ($x = 0; $x < 10; $x++) {
    if ($x == 4) {
        break;
    }
    echo "Numărul este: $x <br>";
}
```

Instrucțiunea **continue** întrerupe o iterație (în buclă), dacă apare o condiție specificată și continuă cu următoarea iterație din buclă.

```
for ($x = 0; $x < 10; $x++) {
    if ($x == 4) {
        continue;
    }
    echo "Numărul este: $x <br>";
}
```

În exemplul de mai sus nu se va afișa “**Numărul este: 4**”, dar se vor afișa toate celelalte mesaje.

4.7 Includerea altor scripturi

PHP permite utilizarea codului aflat în fișiere PHP externe și includerea lor în fluxul curent de execuție folosind funcțiile **include** sau **require**. Important de menționat, în momentul în care un script este inclus cu una din cele 2 funcții, el este imediat și executat (interpretat). Practic, fluxul de execuție al scriptului inițial (cel în care apare instrucțiunea **include** sau **require**) este întrerupt temporar pentru a executa scriptul inclus, urmând ca după finalizarea execuției, fluxul inițial să fie reluat.

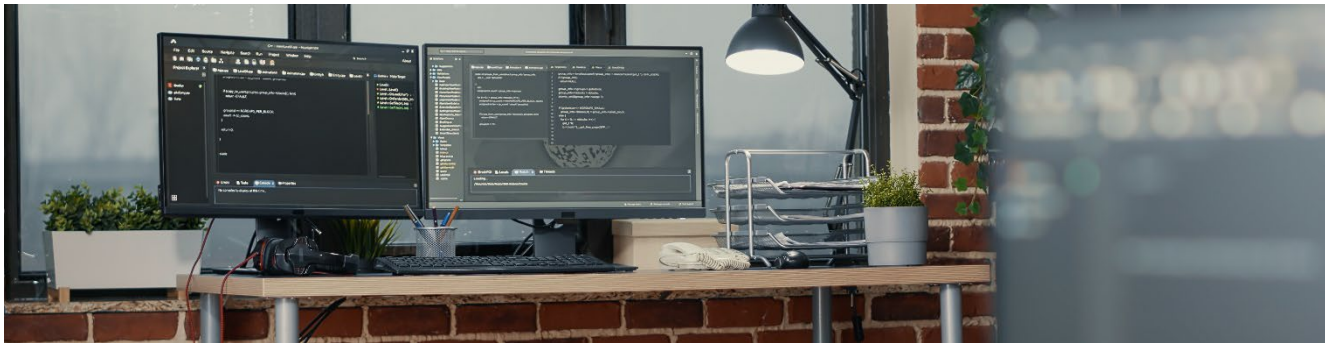
Spre exemplu, dacă avem două fișiere ca în exemplul următor, la accesarea scriptului **script.php** vor fi afișate ambele mesaje - asta pentru că în urma instrucțiunii **include** scriptul **config.php** este imediat interpretat.

```
// fișierul config.php
<?php
echo "Sunt în config.php <br>";
?>

// fișierul script.php
<?php
include('config.php');
echo "Sunt în script.php <br>";
?>
```

4.8 Include sau require?

Funcția **require** face același lucru ca și **include** dar există o diferență între cele două: dacă fișierul solicitat pentru includere nu există, **include** va returna un avertisment, continuând execuția, pe când **require** va returna o eroare iar execuția codului va fi întreruptă.

```
include('fisier_inexistent.php'); // generează un avertisment și
merge mai departe
echo "Acest mesaj va fi afișat  <br>";

require('fisier_inexistent.php'); // generează o eroare și execuția
se întrerupe
echo "Acest mesaj nu va fi afișat  <br>";
```



4.9 Reguli privind scrierea codului PHP

Este important să folosim spațiile albe pentru claritate! Spațierea codului ne ajută să păstram structurile într-o grupare logică, făcând totul mai ușor de înțeles pentru a face codul să funcționeze.

PHP ignoră orice spațiu alb din interiorul codului, deci putem adopta orice stil ne este mai confortabil. Este însă important să menținem același stil.

În general, se recomandă respectarea anumitor standarde privind o serie de aspecte legate formatarea codului. Cel mai nou standard în care sunt specificate regulile de urmat în ceea ce privește scrierea codului PHP este PSR-12. Intenția acestei specificații este de a reduce frecarea cognitivă la citirea codului provenit de la diferiți autori. Face acest lucru prin enumerarea unui set comun de reguli și așteptări cu privire la modul de formatare a codului PHP.

Atunci când diverși autori colaborează în cadrul mai multor proiecte, este util să existe un set de linii directoare care să fie utilizate în toate acele proiecte. Astfel, beneficiul acestui ghid nu constă în regulile în sine, ci în împărtășirea acestor reguli.

Teme

1. Creați o funcție cu 2 parametri (presupunem ca aceștia vor fi tot timpul numere). Aceasta va afișa toate numerele divizibile cu 3 cuprinse între cele 2 valori primite.
2. Creați o funcție ce parcurge un vector primit ca parametru și afișează câte numere pare conține acesta.
3. Creați o funcție ce primește un parametru și verifică dacă acesta este palindrom (ex: 52425, 10988901).