**PHP ACADEMY**

**CURS PHP**

**METODE DE PROIECTAREA ALGORITIMILOR PROGRAMAREA MODULARĂ**

**STUDENT: Moldovai Gianina**

**ORADEA**

**2024**

1. **METODE DE PROIECTAREA ALGORITIMILOR – PROGRAMAREA MODULARĂ**
   1. **SUBPROGRAME ȘI PARAMETRII**

**Funția**

O funcție este o porțiune de cod independentă, care poate fi definită și apelată într-un program pentru a efectua o anumită acțiune. Aceasta ajută la organizarea și structurarea codului în bucăți mai mici și mai reutilizabile.

Funcțiile au un nume și pot avea zero sau mai mulți parametri de intrare. Acești parametri sunt valorile pe care le putem transmite funcției pentru a o ajuta să își îndeplinească sarcina. Funcțiile pot returna, de asemenea, o valoare, care poate fi utilizată în continuare în program.

Definirea unei funcții implică specificarea numelui acesteia, a listei de parametri și a blocului de cod care trebuie executat atunci când funcția este apelată. Apelarea unei funcții se face prin utilizarea numelui acesteia, urmat de paranteze, care pot conține argumentele (valorile) necesare în cazul în care funcția are parametri.

În limbajul PHP, o funcție care afișează un text primit ca parametru poate fi definită astfel:

function afisare\_text($text) {

echo $text; }

Această funcție se numește "afisare\_text" și primește un parametru numit "text". Apoi, funcția afișează textul primit ca parametru folosind funcția "echo".

Pentru a utiliza această funcție, putem apela funcția "afisare\_text" și să-i transmitem textul pe care dorim să-l afișăm ca argument:

afisare\_text("Acesta este un text de afișat.");

În limbajul PHP, putem folosi instrucțiunea `return` pentru a returna un text din interiorul unei funcții și apoi putem utiliza funcția `echo` pentru a-l afișa. Iată un exemplu simplu:

function returnare\_text() {

$text = "Acesta este un text returnat din funcție.";

return $text; }

$text\_returnat = returnare\_text();

echo $text\_returnat;

În acest exemplu, avem funcția `returnare\_text()` care definește o variabilă `$text` care conține textul pe care dorim să-l returnăm. Utilizăm instrucțiunea `return` pentru a returna valoarea variabilei `$text` din funcție.

La apelul funcției `returnare\_text()`, rezultatul returnat, adică valoarea variabilei `$text`, este atribuit variabilei `$text\_returnat`. Apoi, utilizăm funcția `echo` pentru a afișa valoarea variabilei `$text\_returnat`, adică textul returnat din funcție.

Rezultatul acestui cod va fi afișarea textului "Acesta este un text returnat din funcție." în output-ul programului. Astfel, în PHP, putem returna un text din interiorul unei funcții și să-l afișăm în exteriorul acesteia utilizând instrucțiunea `return` și funcția `echo`.

În limbajul PHP, putem transmite parametri unei funcții folosind convenția numărului, poziției și tipului acestora. Aceasta implică specificarea tipului de date pentru fiecare parametru, urmat de numele parametrului și în cele din urmă poziția acestuia în lista de parametri a funcției.

Iată un exemplu:

function afisare\_parametri($nume, $varsta, $oras) {

echo "Nume: " . $nume . "<br>";

echo "Vârsta: " . $varsta . "<br>";

echo "Oraș: " . $oras . "<br>"; } afisare\_parametri("Alex", 25, "București");

În acest exemplu, funcția `afisare\_parametri()` primește trei parametri: `$nume`, `$varsta` și `$oras`. Acestea sunt transmise la apelul funcției, în ordinea specificată în definiția funcției.

Parametrii sunt specificați cu ajutorul numărului, poziției și tipului acestora. De exemplu, primul parametru este de tipul `string`, se numește `$nume` și este poziționat pe prima poziție. Al doilea parametru este de tipul `int`, se numește `$varsta` și este poziționat pe a doua poziție. Al treilea parametru este de tipul `string`, se numește `$oras` și este poziționat pe a treia poziție.

La apelul funcției, valorile specifice pentru acești parametri sunt transmise în ordinea specificată în definiție, astfel încât primul parametru este "Alex", al doilea parametru este 25, iar al treilea parametru este "București". Aceste valori sunt apoi afișate folosind instrucțiunea `echo`.

În concluzie, în PHP putem transmite parametri unei funcții prin specificarea numărului, poziției și tipului acestora în definiția funcției, urmată de transmiterea valorilor specifice în ordinea specificată.

**Domeniu de vizibiliate a variabilelor**

În PHP, există conceptul de domeniu de vizibilitate al variabilelor, care determină în ce context sau în ce parte a codului o variabilă poate fi accesată și utilizată. Domeniul de vizibilitate poate fi local sau global, în funcție de locul în care variabila este declarată și utilizată.

1. Variabile locale:

- Variabilele locale sunt definite și accesibile doar în cadrul funcției sau blocului de cod în care au fost declarate.

- Acestea sunt vizibile doar în interiorul contextului lor și nu pot fi accesate în afara acestuia.

- Variabilele locale sunt declarate folosind cuvântul cheie `local` înainte de numele variabilei.

Exemplu:

function exemplu() {

$variabila\_locala = "Aceasta este o variabilă locală.";

echo $variabila\_locala; }

exemplu(); // Output: Aceasta este o variabilă locală.

echo $variabila\_locala; // Eroare: variabila\_locala is not defined

În acest exemplu, `$variabila\_locala` este o variabilă locală definită în interiorul funcției `exemplu()`. Poate fi accesată și utilizată doar în cadrul acestei funcții.

2. Variabile globale:

- Variabilele globale sunt definite în afara oricărui context de funcție sau bloc de cod și pot fi accesate și utilizate în întregul program.

- Acestea sunt vizibile în toate funcțiile și blocurile de cod din program.

- Variabilele globale pot fi declarate în două moduri: utilizând cuvântul cheie `global` înainte de numele variabilei sau prin utilizarea unui array superglobal, cum ar fi `$\_GLOBALS` sau `$\_SERVER`.

Exemplu:

$variabila\_globala = "Aceasta este o variabilă globală.";

function exemplu() {

global $variabila\_globala;

echo $variabila\_globala; }

exemplu(); // Output: Aceasta este o variabilă globală.

echo $variabila\_globala; // Output: Aceasta este o variabilă globală.

În acest exemplu, `$variabila\_globala` este o variabilă globală definită în afara funcției `exemplu()`. Aceasta poate fi accesată și utilizată atât în interiorul funcției, cât și în afara acesteia, deoarece a fost declarată ca variabilă globală prin utilizarea cuvântului cheie `global`.

În PHP, putem apela funcții cu parametri folosind sintaxa standard a apelului de funcție, urmată de specificarea parametrilor în parantezele rotunde. De asemenea, putem apela funcții ca metode pentru obiecte, folosind sintaxa obiectului -> nume\_metodă().

Pentru apelarea funcțiilor în mod recursiv, putem folosi o abordare similară cu cea pentru apelarea funcțiilor standard, doar că vom include apelul funcției în interiorul corpului aceleiași funcții. Pentru a evita supraîncărcarea procesorului sau a serverului, putem adăuga o temporizare între apelurile recursive folosind funcția `sleep()`.

Iată un exemplu care combină aceste concepte:

function functie\_recursiva($parametru) {

// Afisam parametrul

echo $parametru . "<br>";

// Verificam daca trebuie sa continuam apelul recursiv

if ($parametru > 0) {

// Asteptam 1 secunda

sleep(1);

// Apelam functia recursiv, dar cu un parametru mai mic cu 1

functie\_recursiva($parametru - 1); } }

// Apelam functia recursiv cu parametrul 5 functie\_recursiva(5);

În acest exemplu, am definit o funcție `functie\_recursiva()` care primește un parametru și îl afișează într-o linie nouă. Apoi, verificăm dacă parametrul este mai mare decât 0. Dacă este, așteptăm o secundă folosind funcția `sleep()` și apoi apelăm funcția recursiv cu un parametru mai mic cu 1.

La apelul funcției, am transmis parametrul 5, astfel încât funcția va fi apelată inițial cu parametrul 5 și apoi va fi apelată recursiv de 4 ori, fiecare dată cu un parametru mai mic cu 1, până când parametrul va fi 0.

Apelarea funcțiilor din alte funcții și returnarea rezultatelor în interiorul expresiilor este o practică comună în programare și aduce o serie de beneficii și posibilități. Iată câteva exemple practice ale acestor concepte:

1. Calculatoare sau aplicații matematice:

- Funcțiile care returnează rezultatele unor operații simple, cum ar fi adunarea, scăderea, înmulțirea și împărțirea, pot fi folosite într-o varietate de aplicații matematice.

- De exemplu, putem defini o funcție `adunare()` care primește două numere și returnează suma lor. Aceasta poate fi apoi folosită în alte funcții sau expresii pentru a obține rezultatele dorite. Exemplu:

function adunare($a, $b) {

return $a + $b; } $rezultat = adunare(5, 3); // Apelăm funcția și stocăm rezultatul într-o variabilă echo $rezultat;

2. Manipularea șirurilor de caractere:

- Funcțiile care manipulează șirurile de caractere pot fi apelate din alte funcții sau utilizate în expresii pentru a obține rezultatele dorite.

- De exemplu, putem defini o funcție `lungime\_șir()` care primește un șir de caractere și returnează numărul de caractere din acesta. Aceasta poate fi apoi apelată în alte funcții sau expresii pentru a efectua operații specifice bazate pe lungimea șirului de caractere.

Exemplu:

function lungime\_șir($șir) {

return strlen($șir); }

$nume = "John Doe";

$lungime = lungime\_șir($nume); // Apelăm funcția și stocăm rezultatul într-o variabilă

echo "Numele are " . $lungime . " caractere.";

3. Manipularea datelor și baze de date:

- Funcțiile care manipulează date sau interacționează cu baze de date pot fi apelate din alte funcții sau utilizate în expresii pentru a obține și prelucra informațiile necesare.

- De exemplu, putem defini o funcție `obține\_număr\_rânduri()` care primește un query SQL și returnează numărul de rânduri afectate de acesta într-o bază de date. Aceasta poate fi apoi apelată în alte funcții sau expresii pentru a efectua operații specifice bazate pe numărul de rânduri afectate.

Exemplu:

function obține\_număr\_rânduri($query) {

// Executăm query-ul și obținem numărul de rânduri afectate

$result = mysqli\_query($connection, $query);

$număr\_rânduri = mysqli\_num\_rows($result);

return $număr\_rânduri;

}

$query = "SELECT \* FROM users";

$număr\_rânduri = obține\_număr\_rânduri($query); // Apelăm funcția și stocăm rezultatul într-o variabilă

echo "Au fost găsite " . $număr\_rânduri . " rânduri în baza de date."; // Output: Au fost găsite X rânduri în baza de date.