

Curs 5

2016/2017

# Tehnici moderne de proiectare a aplicatiilor web

# Laborator L3

- Sa se creeze un magazin simplu virtual care:
  - sa prezinte utilizatorului o lista de produse si preturi (constanta – maxim 5 produse)
  - sa preia de la acesta numarul de produse dorit
  - sa calculeze suma totala
  - sa adauge TVA **19%**
  - sa prezinte un raport care sa contina:
    - total de plata
    - ora comenzii

# Laborator L3 - continuare

- se creaza macar 3 pagini:
  - lista produse
  - formular comanda
  - rezultat
- forma paginilor:
  - tabel/CSS

culoare	<b>IMAGINE</b>	culoare
	<b>Continut</b> (cu alta culoare fundal)	

# Laborator – L3 - rezultat

## Magazin online Firma X SRL

### Lista Produse

Nr.	Produs	Pret
1	Carti	100
2	Caiete	50
3	Penare	150
4	Stilouri	125
5	Creioane	25

[Comanda](#)

## Magazin online Firma X SRL

### Realizati comanda

Nr.	Produs	Pret	Cantitate
1	Carti	100	<input type="text" value="1"/>
2	Caiete	50	<input type="text" value="2"/>
3	Penare	150	<input type="text" value="1"/>
4	Stilouri	125	<input type="text" value="0"/>
5	Creioane	25	<input type="text" value="0"/>

## Magazin online Firma X SRL

### Rezultate comanda

Pret total (fara TVA): 350

Pret total (cu TVA): 416.5

Comanda receptionata la data: 17/03/2010 ora 08:24

# Interactiunea cu utilizatorul

- Datele introduse de utilizator in forme se regasesc (in functie de metoda aleasa pentru forma) in una din variabilele:
  - `$_POST` – method="post"
  - `$_GET` – method="get"
  - `$_REQUEST` – ambele metode
- variabilele sunt **matrici** cu **cheia** data de atributul **name** din forma HTML
  - `<input type="text" name="carti_cant" size="3" maxlength="3" />`
  - `$_POST['carti_cant']` contine valoarea introdusa

# Laborator – L3 – sursa 1

```
<?php
define('PRET_CARTE',100);
define('PRET_CALET',50);
define('PRET_PENAR',150);
define('PRET_STILOU',125);
define('PRET_CREION',25);
?><h1>Magazin online Firma X SRL</h1>
<h2>Realizati comanda</h2>
<form action="rezultat.php" method="post">
<table border="0">
<tr bgcolor="#cccccc"><td>Nr.</td><td width="150">Produs</td><td width="50">Pret</td><td
width="15">Cantitate</td></tr>
<tr><td>1</td><td>Carti</td><td align="center"><?php echo PRET_CARTE;?></td><td align="center"><input
name="carte_cant" type="text" value="0" size="3" maxlength="3" /></td></tr>
<tr><td>2</td><td>Caiete</td><td align="center"><?php echo PRET_CALET;?></td><td align="center"><input
name="caiet_cant" type="text" value="0" size="3" maxlength="3" /></td></tr>
<tr><td>3</td><td>Penare</td><td align="center"><?php echo PRET_PENAR;?></td><td align="center"><input
name="penar_cant" type="text" value="0" size="3" maxlength="3" /></td></tr>
<tr><td>4</td><td>Stilouri</td><td align="center"><?php echo PRET_STILOU;?></td><td align="center"><input
name="stilou_cant" type="text" value="0" size="3" maxlength="3" /></td></tr>
<tr><td>5</td><td>Creioane</td><td align="center"><?php echo PRET_CREION;?></td><td align="center"><input
name="creion_cant" type="text" value="0" size="3" maxlength="3" /></td></tr>
<tr>
<td colspan="4" align="center"><input type="submit" value="Trimite" /></td></tr>
</table>
</form>
```

# Laborator – L3 – sursa 2

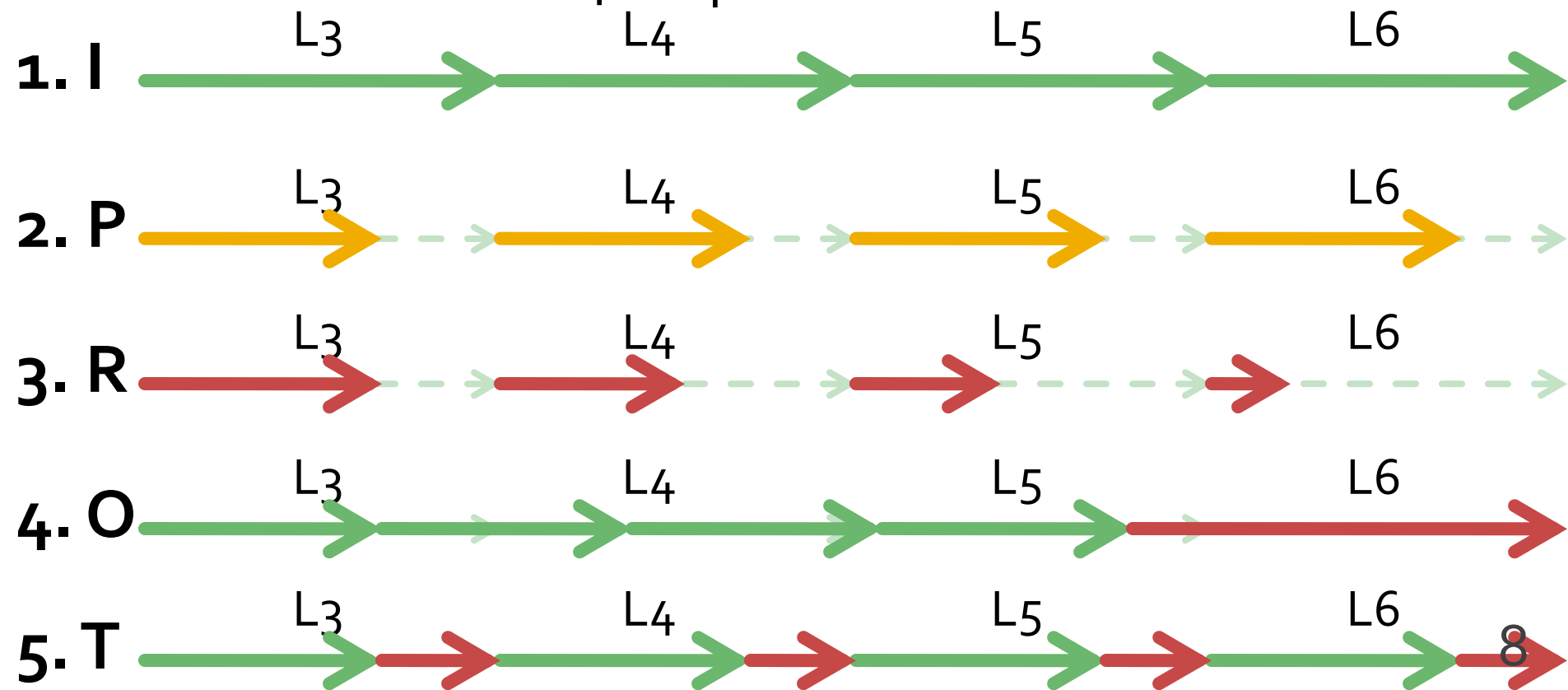
```
<?php
define('PRET_CARTE',100);
define('PRET_CAIET',50);
define('PRET_PENAR',150);
define('PRET_STILOU',125);
define('PRET_CREION',25);
?><h1>Magazin online Firma X SRL</h1>
<h2>Rezultate comanda</h2>
<p>Pret total (fara TVA): <?php
$pret=$_POST['carte_cant']*PRET_CARTE+$_POST['caiet_cant']*PRET_CAIET
+$_POST['penar_cant']*PRET_PENAR+$_POST['stilou_cant']*PRET_STILOU+$_
POST['creion_cant']*PRET_CREION;
echo $pret;?></p>
<p>Pret total (cu TVA): <?php
$pret*=1.24;
echo $pret;?></p>
<p>Comanda receptionata la data: <?php echo date('d/m/Y')." ora
".date('H:i');?></p>
```

```
echo "<pre>";
print_r($_POST);
echo "</pre>";
```

# ! Important

- Laborator **asincron!**

- recomandat – 4 = Optim





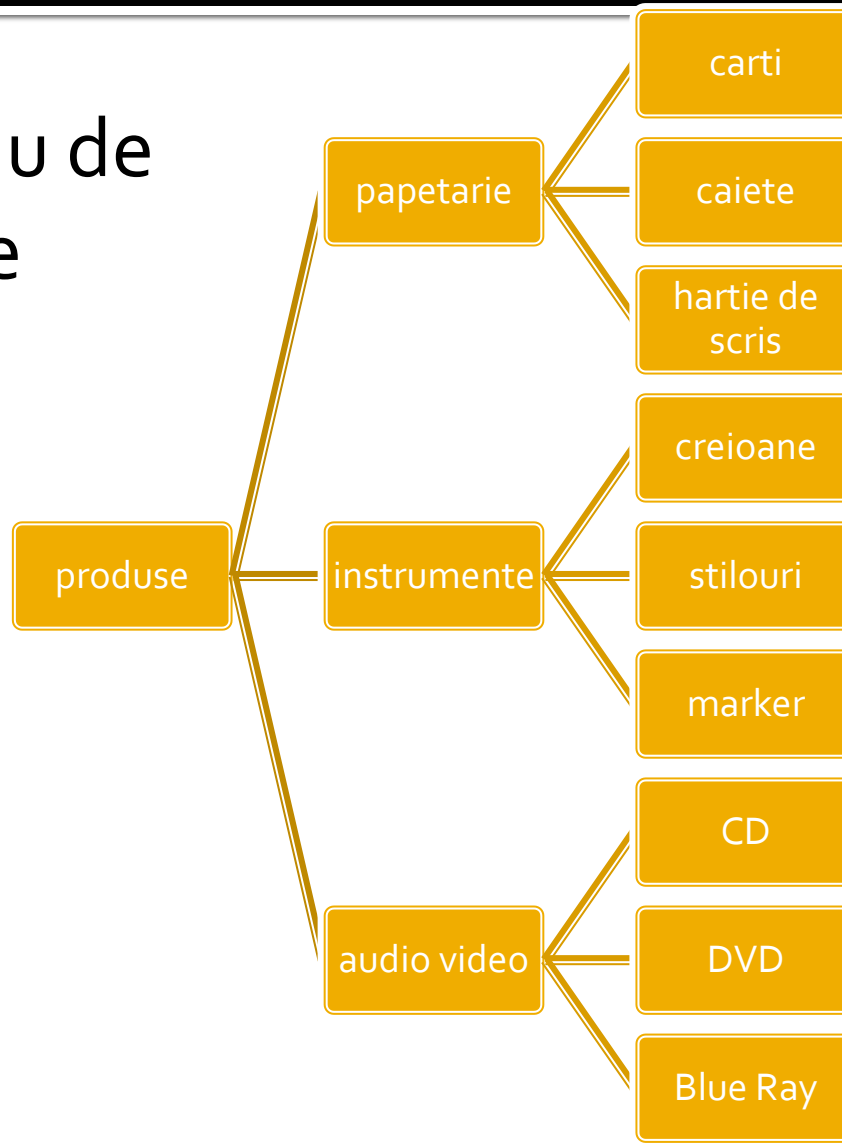
# Laborator 4

# Laborator 4

- Sa se continue magazinul virtual cu:
  - produsele sunt grupate pe **categorii** de produse
  - sa prezinte utilizatorului o lista de categorii de produse pentru a alege
  - sa prezinte utilizatorului o lista de produse si preturi in categoria aleasa
  - lista de produse si preturi se citeste dintr-un **fisier**
  - se preia comanda si se calculeaza suma totala
- Optional
  - se creaza o pagina prin care vanzatorul poate **modifica** preturile si produsele

# Laborator 4

- exemplu de grupare



# Includerea / controlul formei in fisierile PHP - Template

# Analiza critica

- design?
  - in aplicatiile web forma este importanta
  - nu trebuie sa fie inovativa ci familiara
  - "Don't make me think!"
- ~~capacitatea de extindere?~~
  - ~~mai multe produse~~
  - ~~schimbare de pret~~

# Template

- Sablon
- controlul simultan al formei pentru toate paginile din site
- separarea aplicatiei de forma

# Lista produse

**Magazin** **Firma X SRL**

**Magazin online Firma X SRL**

**Lista Produse**

Nr.	Produs	Pret
1	Carti	100
2	Caiete	50
3	Penare	150
4	Stilouri	125
5	Creioane	25
<a href="#">Comanda</a>		

# Elemente de control

- `include()`
- `require()`
- `include_once()`
- `require_once()`
  
- pentru inserarea **SI** evaluarea fisierului folosit ca parametru
- folosite pentru a nu multiplica sectiunile de cod comune
- **require** opreste executia script-ului curent daca fisierul parametru **nu** este gasit
- **...\_once()** verifica daca respectivul fisier a mai fost introdus si **nu** il mai introduce inca o data



# Exemplu – design 2

- sectiunile repetabile pot fi mutate intr-un fisier separat si introduse cu require()
- se identifica zonele comune

```
<html>
<head>
<title>Magazin online Firma X SRL</title>
</head>
<body bgcolor="#CCFFFF">
<table width="600" border="0" align="center">
<tr><td></td></tr>
<tr><td height="600" valign="top"
bgcolor="#FFFFCC">
Continut
</td></tr>
</table>
</body>
</html>
```

# Lista produse

antet.php

```
<html>
<head>
<title>Magazin online Firma X
SRL</title>
</head>
<body bgcolor="#CCFFFF"><?php
define('PRET_CARTE',100);
define('PRET_CALET',50);
define('PRET_PENAR',150);
define('PRET_STILOU',125);
define('PRET_CREION',25);
//orice cod comun PHP
?><table width="600" border="0"
align="center">
<tr><td></td></tr>
<tr><td height="600" valign="top"
bgcolor="#FFFFCC">
<h1>Magazin online Firma X SRL</h1>
```

subsol.php

```
</td></tr>
</table>
</body>
</html>
```

\*.php

```
<?php require('antet.php');?>
<h2>Lista Produse</h2>
<table border="1">
...
</table>
<?php require('subsol.php');?>
```

# Lista produse/template

**Magazin** **Firma X SRL**

**Magazin online Firma X SRL**

**Lista Produse**

Nr.	Produs	Pret
1	Carti	100
2	Caiete	50
3	Penare	150
4	Stilouri	125
5	Creioane	25

[Comanda](#)

# Avantajul lucrului cu sabloane

- viteza de dezvoltare a aplicatiei
- separare clara a formei de aplicatie
- forma unitara
  - “don’t make me think”
- modificarea simultana a formei pentru toate paginile din site
- posibilitatea definirii datelor comune intr-un singur fisier
  - `define('PRET_CARTE',100);`

Hypertext PreProcessor

**PHP**

# PHP - Concepte

- limbaj interpretat – compilat “on the fly” de interpretorul PHP de pe server
- poate fi integrat in HTML – utilizarea tipica
- un fisier sursa PHP este un fisier HTML (in general) cu sectiuni de cod PHP
  - interpretorul PHP cauta sectiunile pe care trebuie sa le interpreteze si interiorul lor proceseaza instructiuni ca fiind PHP
  - ce se gaseste in exteriorul acestor sectiuni este trimis spre server-ul web nemodificat
- **<?php ... ?>**
  - stil XML – implicit, disponibil intotdeauna, recomandat

# Variante de integrare

- Toate variantele ofera aceeasi sursa HTML pentru browser
- E **recomandata** cea care lasa structura HTML nemodificata si doar datele dinamice sunt rezultatul procesarii
- Codul HTML + PHP e interpretat mult mai elegant in editoarele WYSIWYG

```
<h2>Rezultate comanda</h2>  
<?php echo '<p>Comanda receptionata</p>';?>
```

```
<h2>Rezultate comanda</h2>  
<p><?php echo 'Comanda receptionata';?></p>
```

```
<?php echo '<h1>Magazin online XXX SRL</h1>';?>  
<?php echo '<h2>Rezultate comanda</h2>';?>  
<?php echo '<p>Comanda receptionata</p>';?>
```

```
<?php  
echo '<h1>Magazin online XXX SRL</h1>';  
echo '<h2>Rezultate comanda</h2>';  
echo '<p>Comanda receptionata</p>';  
?>
```

# Variabile tip string ` `

- apostroful ` ` e utilizat pentru definirea sirurilor primare de caractere
  - se defineste o suita de caractere
  - prelucrarile in interiorul sirului sunt reduse
    - ` reprezinta caracterul apostrof
    - \\ si \ reprezinta caracterul backslash
    - **doar atat!!!**



# Variabile tip string ""

- ghilimelele "" sunt utilizate pentru definirea sirurilor de caractere complexe
  - prelucrarile in interiorul sirului sunt mai complexe decat echivalentul C/C++
    - caracterele ASCII speciale, identic cu C++: \n, \r, \t, \l, \v, \e, \f, \x, \u
    - \" caracterul ghilimele
    - \\$ caracterul \$
    - se interpreteaza **variabile** in interiorul sirului !!!

# Variabile tip string “ ”

- Daca exista alternativa e preferabila utilizarea operatorului concatenare in locul sintaxei complexe de interpretare a variabilelor in sir {}
  - “sir 1 {\$expresie} sir2” se inlocuieste cu:
    - “sir 1”.\$expresie.” sir2”
      - paranteze (\$expresie) necesare pentru a evita situatii in care \$expresie contine operatori mai putin prioritari

# PHP – tipuri de date

```
$var = expresie
```

- Controlul variabilelor se face automat, “on the fly”
  - Daca \$var nu era definita anterior, in urma atribuirii se defineste de tipul dat de rezultatul expresiei
  - Daca \$var era definita, de un anumit tip (oarecare), in urma atribuirii devine de tipul dat de rezultatul expresiei
  - La finalizarea executiei script-ului se elimina variabila din memorie (automat)

# PHP – tipuri de date

- tipul de date nu e decis de programator prin declaratia variabilei
- e decis de interpretor in functie de tipul de date stocat in variabila respectiva

```
<?php
echo $variabila ; // tip Null, neinitializat – valoare NULL (doar)
$variabila = "0"; // $variabila tip string (ASCII 48)
$variabila += 2; // $variabila tip integer (2)
$variabila = $variabila + 1.3; // $variabila tip float (3.3)
$variabila = 5 + "10 obiecte"; // $variabila tip integer (15)
$var2=5; // $var2 tip integer (5)
$variabila=$var2."10 obiecte"; // $variabila tip string "510 obiecte"
?>
```

# PHP – Functii

- conceptual similare celor din C/C++
- functiile nu trebuie declarate inainte de a fi folosite
- numele functiilor este “case-insensitive”
- un mare numar de functii cu utilitate directa in aplicatiile web exista in bibliotecile PHP
- unele biblioteci trebuie activate in momentul configurarii PHP
  - `extension=php_gd2.dll` (linia 639) // pentru functii de procesare grafica de exemplu
  - `extension=php_mysql.dll` (linia 651) // pentru functii de acces la baze de date MySql
  - `extension=php_mysqli.dll` (linia 652) // pentru functii de acces la baze de date MySql (**obligatoriu** de la PHP 5.6)

# Utilizarea functiilor PHP

- `<form action="rezultat.php" method="post">`

`<p>Comanda receptionata la data:`

`<?php echo date('d/m/Y')." ora ".date('H:i');?></p>`

**Magazin online XXX SRL**

**Rezultate comanda**

Comanda receptionata la data: 10/03/2010 ora 13:36

`<body>`

`<h1>Magazin online XXX SRL</h1>`

`<h2>Rezultate comanda</h2>`

`<p>Comanda receptionata la data:`

`10/03/2010 ora 13:36</p>`

`</body>`

# Elemente de control

- majoritatea notiunilor si sintaxei sunt similare celor din C/C++
- instructiune compusa: separata de acolade {...}
- if / else / elseif – executie conditionata

```
<?php
if ($a > $b) {
    echo "a mai mare ca b";
} elseif ($a == $b) {
    echo "a egal cu b";
} else {
    echo "a mai mic ca b";
}
?>
```

# Elemente de control

- while
- do-while
- for
- switch
- return
- break
- goto
  
- Similare cu echivalentele C/C++

```
$i = 1;  
while ($i <= 10) {  
    echo $i++;  
}
```

```
$i = 10;  
do {  
    echo $i--;  
} while ($i > 0);
```

```
for ($i = 1; $i <= 10; $i++) {  
    echo $i;  
}
```

```
switch ($i) {  
    case 0:  
        echo "i este 0";  
        break;  
    case 1:  
        echo "i este 1";  
        break;  
    default:  
        echo "i nici 1 nici 0";  
        break;  
}
```



# Elemente de control

- include()
- require()
- include\_once()
- require\_once()
  
- pentru inserarea **SI** evaluarea fisierului folosit ca parametru
- folosite pentru a nu multiplica sectiunile de cod comune
- require opreste executia script-ului curent daca fisierul parametru nu este gasit
- ...\_once() verifica daca respectivul fisier a mai fost introdus si **nu** il mai introduce inca o data

# Variabile globale

# Variabile globale

- Variabilele globale (predefinite)
  - accesibile script-urilor PHP prin conlucrarea cu server-ul
  - Exemple:
    - `$_SERVER` — Server and execution environment information
    - `$_GET` — HTTP GET variables
    - `$_POST` — HTTP POST variables
    - `$_FILES` — HTTP File Upload variables
    - `$_REQUEST` — HTTP Request variables
    - `$_SESSION` — Session variables
    - `$_ENV` — Environment variables
    - `$_COOKIE` — HTTP Cookies

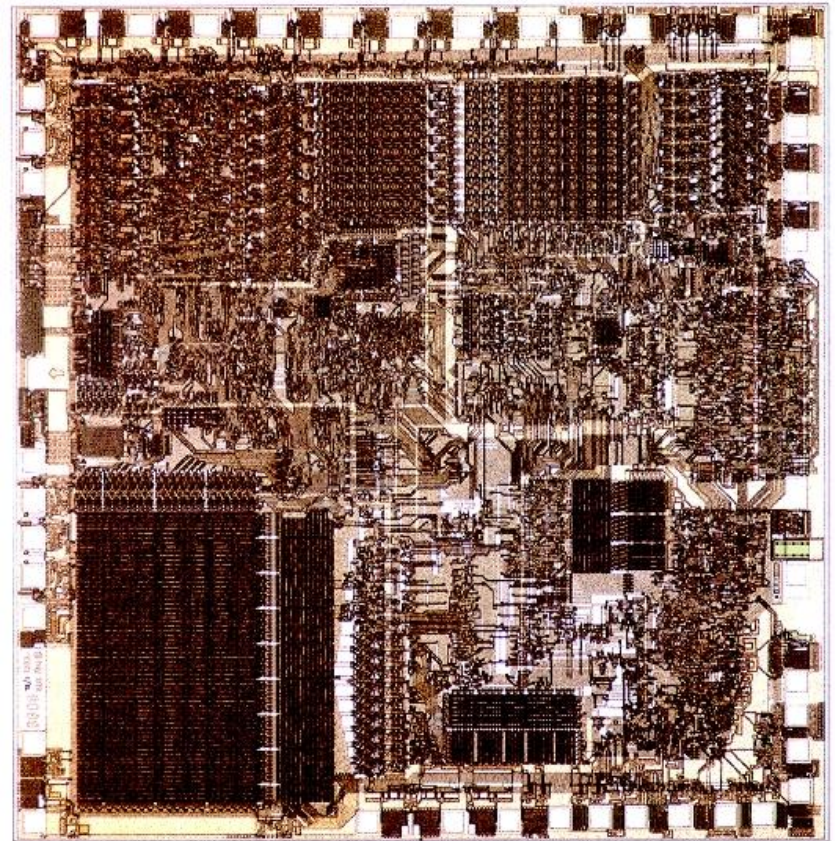
# Interactiunea cu utilizatorul

- Datele introduse de utilizator in forme se regasesc (in functie de metoda aleasa pentru forma) in una din variabilele:
  - `$_POST` – method="post"
  - `$_GET` – method="get"
  - `$_REQUEST` – ambele metode
- variabilele sunt **matrici** cu **cheia** data de atributul **name** din forma HTML
  - `<input type="text" name="carti_cant" size="3" maxlength="3" />`
  - `$_POST['carti_cant']` contine valoarea introdusa

# Structuri repetitive – matrici

# Impresionant?

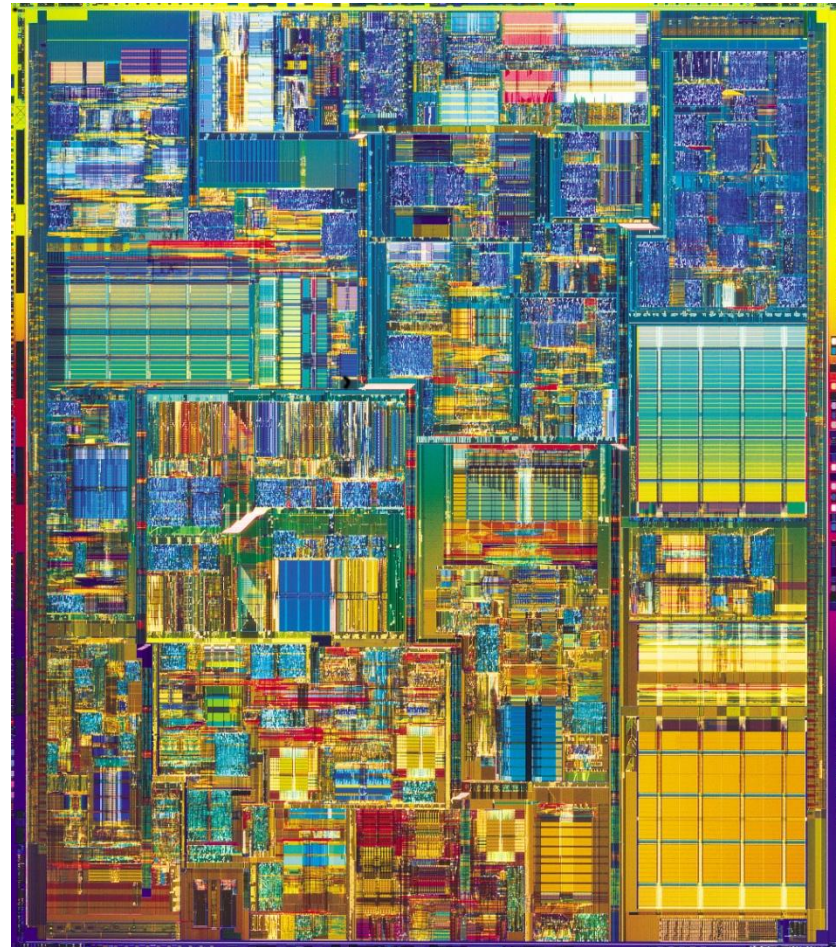
- Intel® 8086
- 29.000 tranzistoare pe CPU
- 1978
- 1 MB date
- 4.7 MHz





# Impresionant?

- Intel® Itanium® processors (Tukwila)
- 2009
- 2 miliarde tranzistoare pe CPU
- 16 EB date (16 G GB)
- > 3 GHz



# Concepte

- Efectuare foarte rapida a unui numar **mic** de instructiuni, de **complexitate redusa**, repetate de un numar foarte mare de ori
- Programare: coborarea rationamentului la nivelul de **complexitate redusa**, cu obtinerea performantei prin structuri repetitive simple efectuate rapid.
- Operatii repetitive / date repetitive



# Matrici in PHP

- matricea este tipul de variabila care asociaza **valori** unor **chei**
- spre deosebire de C, Basic, **cheile nu sunt** obligatoriu numere **intregi**, pot fi si **siruri**
- implicit cheile sunt intregi succesivi (pentru fiecare element adaugat) si primul element este 0.
- definirea unei perechi cheie / valoare
  - cheie => valoare
- definirea unei matrici
  - `$matr = array("definirea perechilor chei/valori")`
  - pereche: cheie => valoare, ...

# Matrici in PHP

```
$matr = array(1, 2, 3, 4, 5);
```

```
$matr[0]=1
```

```
$matr[1]=2
```

```
$matr[2]=3
```

```
$matr[3]=4
```

```
$matr[4]=5
```

```
$matr = array('a' => 1, 'b' => 2, 3, 4, 5);
```

```
$matr['a']=1
```

```
$matr['b']=2
```

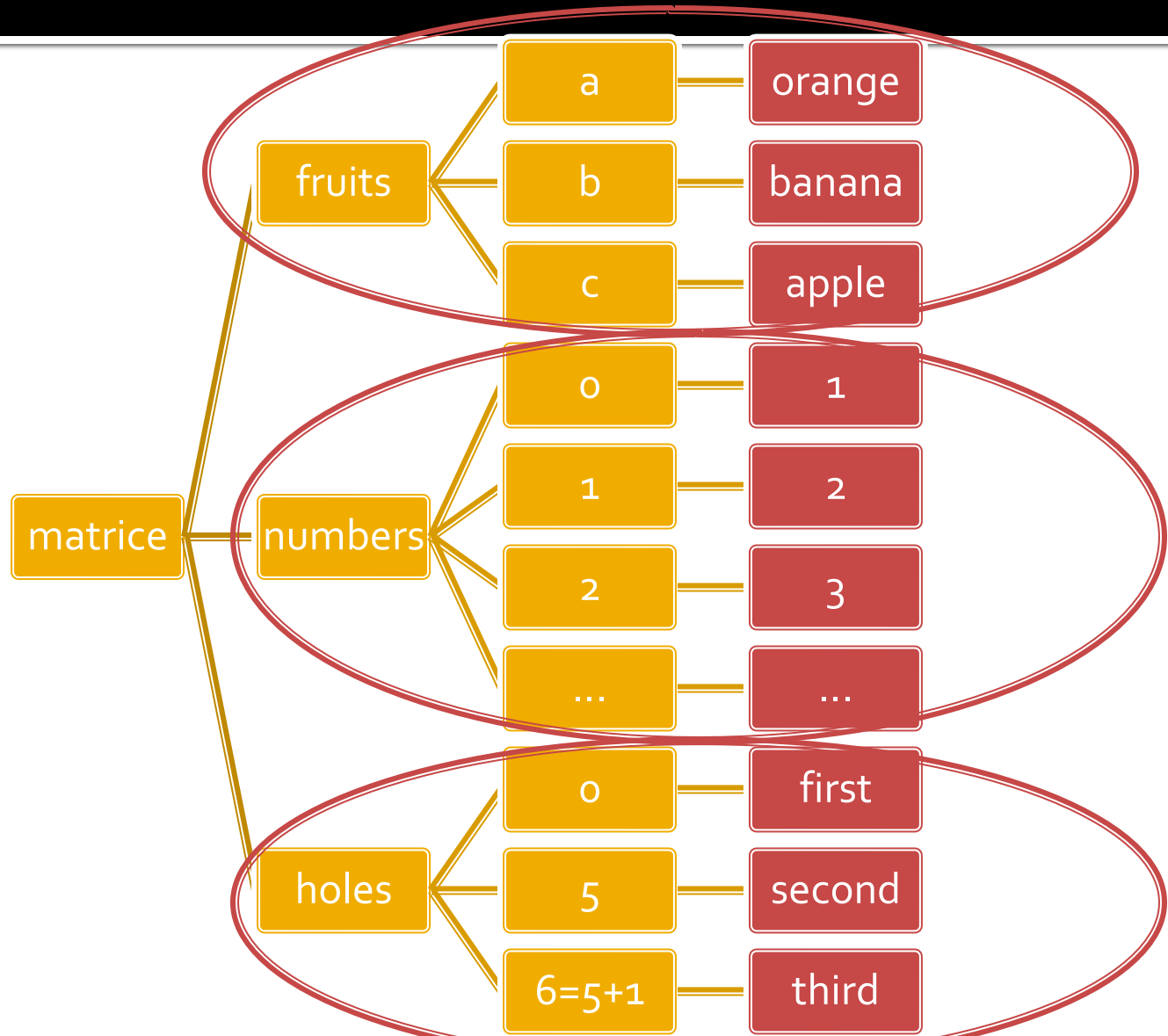
```
$matr[0]=3
```

```
$matr[1]=4
```

```
$matr[2]=5
```

```
$matrice= array (  
    "fruits" => array("a" => "orange", "b" => "banana", "c" => "apple"),  
    "numbers" => array(1, 2, 3, 4, 5, 6),  
    "holes" => array("first", 5 => "second", "third")  
);
```

# Matrice = arbore



# Afisarea matricilor

```
echo "<pre>";  
print_r ($matr);  
echo "</pre>";
```

```
$matr= array (  
"fruits" =>  
array("a" => "orange", "b" => "banana", "c" => "apple",  
"ultim"),  
"numbers" =>  
array(1, 2, 3, 4, 5, 6),  
"holes" =>  
array("first", 5 => "second", "third")  
);  
echo $matr;  
echo "<pre>";  
print_r ($matr);  
echo "</pre>";
```

```
Array  
Array  
(  
  [fruits] => Array  
    (  
      [a] => orange  
      [b] => banana  
      [c] => apple  
      [0] => ultim  
    )  
  [numbers] => Array  
    (  
      [0] => 1  
      [1] => 2  
      [2] => 3  
      [3] => 4  
      [4] => 5  
      [5] => 6  
    )  
  [holes] => Array  
    (  
      [0] => first  
      [5] => second  
      [6] => third  
    )  
)
```

# Chei

- Chei numerice
  - implicite
  - similare celorlalte limbaje de programare
  - dificil de utilizat (trebuie retinuta valoarea logica a unei anumite chei numerice)
- Chei sir
  - claritate mai mare
  - eficienta numerica mai mica
  - matricile au un index numeric intern, implicit ascuns, accesibil prin functii :  
**index => cheie => valoare**

# Elemente de control

- `for` – util dacă la definirea matricilor sunt folosite cheile numerice implicite (numere întregi)
- `do ... while` și `while` se pot folosi împreună cu funcții specifice caracteristice matricilor `next()`, `prev()`, `end()`, `reset()`, `current()`, `each()`
- `foreach` - elementul de control al iteratiilor cel mai potrivit pentru chei alfanumerice

# Elemente de control – foreach

- `foreach (array_expression as $key => $value) statement`
- `foreach (array_expression as $value) statement`
- iterarea prin fiecare element al matricii
- la fiecare element variabila declarata in instructiune variabila locala `$key` ofera acces la cheia curenta iar variabila locala `$value` ofera acces la valoarea asociata
- `foreach()` lucreaza cu o **copie** a matricii deci matricea originala nu va fi modificata prin schimbarea continutului variabilelor `$key` si `$value`

# Elemente de control – foreach

```
$matr = array (  
    "fruits" => array("a" => "orange", "b" => "banana", "c" => "apple", "ultim"),  
    "numbers" => "in loc de numere",  
    "holes" => "in loc de ce era"  
);  
foreach ($matr as $scheie => $continut)  
    echo "matr[".$scheie."]=".$continut."<br />";
```

```
matr[fruits]=Array  
matr[numbers]=in loc de numere  
matr[holes]=in loc de ce era
```



# Matrici – functii utile

- `current ($matr)` – returneaza elementul indicat de indicele intern al matricii (`~v[i]`)
- `next ($matr)` – incrementeaza indicele intern si returneaza valoarea stocata acolo (`~v[++i]`)
- `prev ($matr)` – decrementeaza indicele intern si returneaza valoarea stocata acolo (`~v[--i]`)
- `end($matr)` – muta indicele intern la ultimul element si returneaza valoarea stocata acolo (`~i=N-1;v[i]`)
- `reset($matr)` – muta indicele intern la primul element si returneaza valoarea stocata acolo (`~i=0;v[i]`)

# Matrici – functii utile

- `sort($matr)` – ordoneaza in ordine crescatoare a **valorilor** o matrice, cheile sunt sterse si recreate
  - `$fruits = array("lemon", "orange", "banana", "apple");`  
`sort($fruits);`
  - `fruits[0] = apple, fruits[1] = banana, fruits[2] = lemon, fruits[3] = orange`
- `rsort($matr)` – similar, descrescator

# Matrici – functii utile

- `asort($matr)` ordoneaza in ordine crescatoare a **valorilor** o matrice, cheile sunt pastrate, inclusiv asocierea cheie => valoare
  - `$fruits = array("d" => "lemon", "a" => "orange", "b" => "banana", "c" => "apple");`  
`asort($fruits);`
  - `c = apple, b = banana, d = lemon, a = orange`
- `arsort($matr)` – similar, descrescator

# Matrici – functii utile

- `ksort($matr)` ordoneaza in ordine crescatoare a **cheilor** o matrice, cheile sunt pastrate, inclusiv asocierea cheie => valoare
  - `$fruits = array("d" => "lemon", "a" => "orange", "b" => "banana", "c" => "apple");`  
`ksort($fruits);`
  - a = orange, b = banana, c = apple , d = lemon
- `krsort($matr)` – similar, descrescator

# Exemplu utilizare matrici

# Analiza critica

- ~~design?~~
  - ~~in aplicatiile web forma este importanta~~
  - ~~nu trebuie sa fie inovativa ci familiara~~
  - ~~"Don't make me think!"~~
- capacitatea de extindere?
  - mai multe produse
  - schimbare de pret

# Exemplu

- In exemplul anterior utilizarea matricilor va aduce urmatoarele avantaje:
  - codul va fi mai concis
  - codul va fi mai general (valabil si pentru 5 produse si pentru 1000)
  - scalabilitate crescuta (se pot adauga usor produse)

# Exemplu

- fiecare produs e caracterizat de:
  - nume
  - pret
  - (eventual) descriere
  - cantitate comandata
- putem folosi unul din attribute ca si cheie (numele in exemplu)
- se poate controla (prin atributul name = "") structura variabilei globale `$_POST`



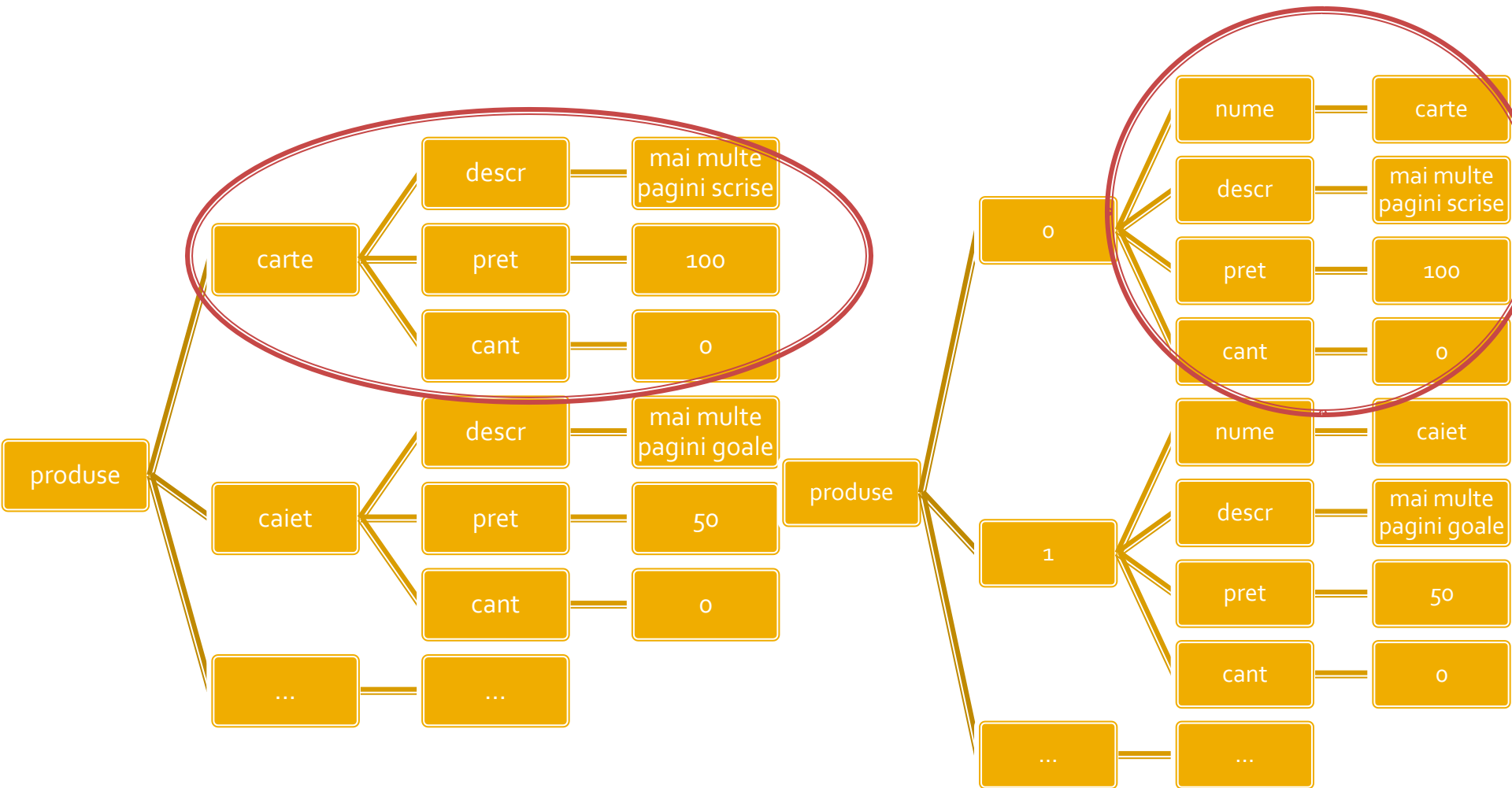
# Matrice produse

- una din structurile posibile

```
$produse = array ( 'carte' => array ("descr" => "mai multe pagini scrise", "pret" => 100, "cant" => 0),  
                  'caiet' => array ("descr" => "mai multe pagini goale", "pret" => 50, "cant" => 0),  
                  'penar' => array ("descr" => "loc de depozitat instrumente", "pret" => 150, "cant" => 0),  
                  'stilou' => array ("descr" => "instrument de scris albastru", "pret" => 125, "cant" => 0),  
                  'creion' => array ("descr" => "instrument de scris gri", "pret" => 25, "cant" => 0)  
                );
```

- se urmareste obtinerea unei structuri clare
  - usor de modificat/adaugat date
  - usor de utilizat
- daca definitia se introduce in fisierul antet va fi accesibila in toate fisierele individuale

# Matrice produse



# Crearea listei de produse

Magazin

Firma X SRL

## Magazin online Firma X SRL

### Lista Produse

Nr.	Produs	Descriere	Pret
1	Carte	mai multe pagini scrise legate	100
2	Caiet	mai multe pagini goale legate	50
3	Penar	loc de depozitat instrumente de scris	150
4	Stilou	instrument de scris albastru	125
5	Creion	instrument de scris gri	25
1	Carte	mai multe pagini scrise legate	100
2	Caiet	mai multe pagini goale legate	50
3	Penar	loc de depozitat instrumente de scris	150
4	Stilou	instrument de scris albastru	125
5	Creion	instrument de scris gri	25
<a href="#">Comanda</a>			

# Laborator – L3 – sursa 1

```
<?php
define('PRET_CARTE',100);
define('PRET_CALET',50);
define('PRET_PENAR',150);
define('PRET_STILOU',125);
define('PRET_CREION',25);
?><h1>Magazin online Firma X SRL</h1>
<h2>Realizati comanda</h2>
<form action="rezultat.php" method="post">
<table border="0">
<tr bgcolor="#cccccc"><td>Nr.</td><td width="150">Produs</td><td width="50">Pret</td><td
width="15">Cantitate</td></tr>
<tr><td>1</td><td>Carti</td><td align="center"><?php echo PRET_CARTE;?></td><td align="center"><input
name="carte_cant" type="text" value="0" size="3" maxlength="3" /></td></tr>
<tr><td>2</td><td>Caiete</td><td align="center"><?php echo PRET_CALET;?></td><td align="center"><input
name="caiet_cant" type="text" value="0" size="3" maxlength="3" /></td></tr>
<tr><td>3</td><td>Penare</td><td align="center"><?php echo PRET_PENAR;?></td><td align="center"><input
name="penar_cant" type="text" value="0" size="3" maxlength="3" /></td></tr>
<tr><td>4</td><td>Stilouri</td><td align="center"><?php echo PRET_STILOU;?></td><td align="center"><input
name="stilou_cant" type="text" value="0" size="3" maxlength="3" /></td></tr>
<tr><td>5</td><td>Creioane</td><td align="center"><?php echo PRET_CREION;?></td><td align="center"><input
name="creion_cant" type="text" value="0" size="3" maxlength="3" /></td></tr>
<tr>
<td colspan="4" align="center"><input type="submit" value="Trimite" /></td></tr>
</table>
</form>
```

# Crearea listei de produse

```
<?php require('antet.php');?>
<h2>Lista Produse</h2>
<table border="1">
<tr bgcolor="#cccccc"><td>Nr.</td><td width="150">Produs</td><td width="150">Descriere</td><td
width="50">Pret</td></tr>
<?php
$index=1;
foreach ($produse as $prod => $detalii)
{
    echo "<tr><td>".$index."</td><td>".ucfirst(strtolower($prod))."</td><td>".$detalii['descr']."</td><td
align=\"center\">".$detalii['pret']."</td></tr>";
    $index++;
}
?>
<?php
$index=1;
foreach ($produse as $prod => $detalii)
{?>
<tr><td><?php echo $index;?></td><td><?php echo ucfirst(strtolower($prod));?></td><td><?php echo
$detalii['descr'];?></td><td align="center"><?php echo $detalii['pret'];?></td></tr>
<?php $index++;
} ?>
<tr><td colspan="4" align="center"><a href="formular.php">Comanda</a></td></tr></table>
<?php require('subsol.php');?>
```



# Subdivizare \$\_POST

- atributul **name** in forma devine **cheie** in matricea globala \$\_POST
  - `<input type="text" name="carti_cant" size="3" maxlength="3" />`
  - `$_POST['carti_cant']` contine valoarea introdusa
- realizand atributul name ca matrice, se obtine in \$\_POST o "submatrice" care grupeaza elementele input
  - `<input type="text" name="cant[carti]" size="3" maxlength="3" />`
  - `$_POST ['cant'] ['carti']` contine valoarea introdusa

# Subdivizare \$\_POST

- realizand atributul `name` ca matrice, se obtine in `$_POST` o "submatrice" care grupeaza elementele dorite
  - `<input type="text" name="cant[carti]" size="3" maxlength="3" />`
  - `$_POST ['cant'] ['carti']` contine valoarea introdusa
- Este necesar pentru a grupa elementele similare pe care sa le prelucram la receptie cu `foreach`
- `$_POST` contine si alte elemente pe care le dorim eventual tratate separat
  - numele (`name`) si valoarea butonului "submit" apar in `$_POST` de exemplu

# Crearea formei de comanda

```
<?php require('antet.php');?>
<h2>Realizati comanda</h2>
<form action="rezultat.php" method="post">
<table border="0">
<tr bgcolor="#cccccc"><td>Nr.</td><td width="150">Produs</td><td width="50">Pret</td><td
width="15">Cantitate</td></tr>
<?php
$index=1;
foreach ($produse as $prod => $detalii)
    {?>
<tr><td><?php echo $index;?></td><td><?php echo ucfirst(strtolower($prod));?></td><td
align="center"><?php echo $detalii['pret'];?></td><td><input name="<?php echo
"cant[".$prod."];?>" type="text" value="0" size="3" maxlength="3" /></td></tr>
<?php $index++;
    }?>
<tr><td colspan="4" align="center"><input type="submit" value="Trimite" /></td></tr>
</table>
</form>
<?php require('subsol.php');?>
```



# Crearea listei de produse

**Magazin** **Firma X SRL**

**Magazin online Firma X SRL**

**Realizati comanda**

Nr.	Produs	Pret	Cantitate
1	Carte	100	<input type="text" value="0"/>
2	Caiet	50	<input type="text" value="0"/>
3	Penar	150	<input type="text" value="0"/>
4	Stilou	125	<input type="text" value="0"/>
5	Creion	25	<input type="text" value="0"/>

# Prelucrarea comenzii

```
<?php require('antet.php');?>
<h2>Rezultate comanda</h2>
<p>Pret total (fara TVA):
<?php
$pret=0;
$afis="";
foreach ($_POST['cant'] as $prod => $cant)
    {
        $pret += $cant*$produse[$prod]['pret'];
        $afis .= "+".$cant."x".$produse[$prod]['pret'];
    }
echo $pret;
?>
<p>Obtinut astfel: <?php echo $afis;?></p>
<p>Pret total (cu TVA): <?php echo $pret*1.24;?></p>
<p><?php
echo "<pre>";
print_r ($_POST);
echo "</pre>";
?>
</p>
<p>Comanda receptionata la data: <?php echo date('d/m/Y')." ora ".date('H:i');?></p>
<?php require('subsol.php');?>
```

# Prelucrarea comenzii

**Magazin**

**Firma X SRL**

## Magazin online Firma X SRL

### Rezultate comanda

Pret total (fara TVA): 600

Obtinut astfel: +2x100+2x50+2x150+0x125+0x25

Pret total (cu TVA): 714

```
Array
(
    [cant] => Array
        (
            [carte] => 2
            [caiet] => 2
            [penar] => 2
            [stilou] => 0
            [creion] => 0
        )
)
```

Comanda receptionata la data: 17/03/2010 ora 13:55

# Memorarea datelor

# Scrierea datelor pe disc

- Pentru a oferi posibilitatea beneficiarului aplicatiei (vanzator) sa poata adauga/sterge/modifica produse
  - din interfata browser
  - fara sa aiba cunostinte de programare
- E necesar ca matricea **\$produse** sa fie creata in timpul rularii plecand de la un suport extern de date, accesibil pentru scriere vanzatorului
- Ulterior se va implementa aplicatia ce utilizeaza baze de date – momentan se vor scrie datele pe disc

# Utilizarea fisierelor – Functii

- `pointer = fopen(cale,mod)` deschide un fisier pentru operatii descrise de "mod". Se returneaza un pointer spre fisier de tip resursa care va fi folosit la operatiile urmatoare
- `fwrite(pointer,date)` – scrie datele in fisier (date – de tip string)
- `string = fread(pointer,cantitate)` citeste "cantitate" octeti din fisier
- `$matr = file(cale)` deschide fisierul identificat cu "cale" si citeste fiecare linie (incluzand \n) intr-un element distinct in matrice. \$matr de tip array, matrice de siruri

# Crearea fisierului

```
$produse = array ( 'carte' => array ("descr" => "mai multe pagini scrise legate", "pret" => 100, "cant" => 0),  
    'caiet' => array ("descr" => "mai multe pagini goale legate", "pret" => 50, "cant" => 0),  
    'penar' => array ("descr" => "loc de depozitat instrumente de scris", "pret" => 150, "cant" => 0),  
    'stilou' => array ("descr" => "instrument de scris albastru", "pret" => 125, "cant" => 0),  
    'creion' => array ("descr" => "instrument de scris gri", "pret" => 25, "cant" => 0)  
    );  
  
$handle = fopen("produse.txt", "wb");  
foreach ($produse as $prod => $detalii)  
    fwrite($handle,$prod."\t".$detalii['descr']."\t".$detalii['pret']."\t"  
.$detalii['cant']."\r\n");
```

# Crearea fisierului

- crearea initiala se poate face prin modificarea o singura data a fisierului antet.php existent astfel incat sa scrie datele pe disc

```
$produse = array ( 'carte' => array ("descr" => "mai multe pagini scrise", "pret" => 100, "cant" => 0),  
                  'caiet' => array ("descr" => "mai multe pagini goale", "pret" => 50, "cant" => 0),  
                  'penar' => array ("descr" => "loc de depozitat instrumente", "pret" => 150, "cant" => 0),  
                  'stilou' => array ("descr" => "instrument de scris albastru", "pret" => 125, "cant" => 0),  
                  'creion' => array ("descr" => "instrument de scris gri", "pret" => 25, "cant" => 0)  
                );  
$handle = fopen("produse.txt", "wb");  
foreach ($produse as $prod => $detalii)  
    fwrite($handle,$prod."\t".$detalii['descr']."\t".$detalii['pret']."\t".$detalii['cant']."\r\n");
```



# Citirea fisierului pentru crearea matricii

```
$matr=file("produse.txt");  
echo "<pre>";  
print_r ($matr);  
echo "</pre>";  
foreach ($matr as $linie)  
{  
    $valori=explode("\t",$linie,4);  
    $produse[$valori[0]]=array ("descr" => $valori[1], "pret" => $valori[2], "cant" => $valori[3]);  
}
```

```
Array  
(  
    [0] => carte          mai multe pagini scrise legate  100    0  
    [1] => caiet         mai multe pagini goale legate   50    0  
    [2] => penar         loc de depozitat instrumente de scris 150    0  
    [3] => stilou        instrument de scris albastru    125    0  
    [4] => creion        instrument de scris gri 25      0
```

# produse.txt

- se pot utiliza si alte caractere pentru separare
  - esential: sa nu apara in date
  - TAB are efect vizual si in fisiere text

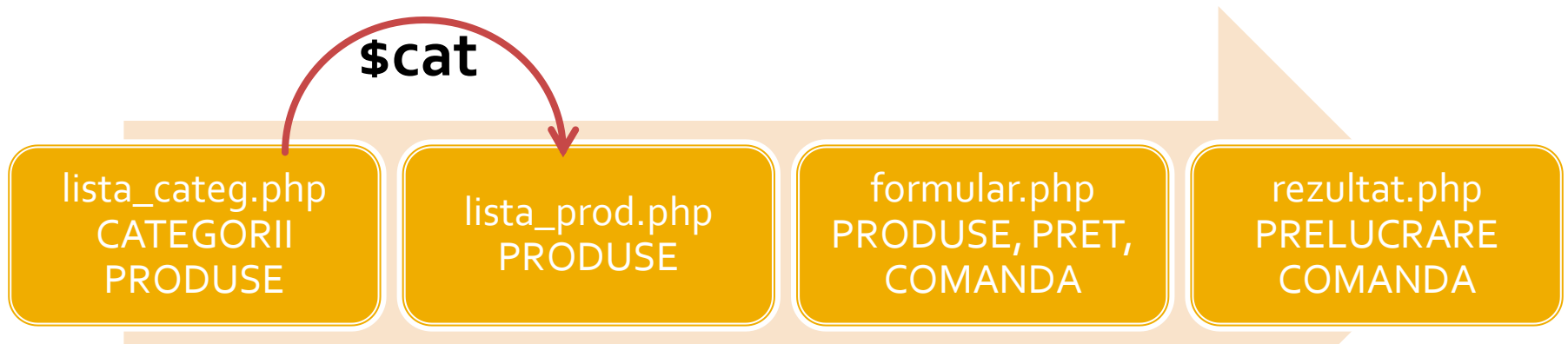
carte	mai multe pagini scrise legate	100	0
caiet	mai multe pagini goale legate	75	0
penar	loc de depozitat instrumente de scris	150	0
stilou	instrument de scris albastru	125	0
creion	instrument de scris gri	25	0

# Metode de transmitere

- **post** datele sunt transmise in bloc
- **get** datele sunt atasate adresei documentului de procesare : `results.php?prob=81&an=2009`
- se poate simula realizarea unei forme (**get**) prin scrierea corespunzatoare a link-urilor

# Transmitere prin GET

- in `lista_categ.php`
  - `<a href="lista_prod.php?categ=<?php echo $cat;?>"> <?php echo $cat;?> </a>`
- are efect in `lista_prod.php`
  - `$_GET['categ']="valoarea $cat corespunzatoare"`



# Laborator 4

# Laborator 4

- Sa se continue magazinul virtual cu:
  - produsele sunt grupate pe **categorii** de produse
  - sa prezinte utilizatorului o lista de categorii de produse pentru a alege
  - sa prezinte utilizatorului o lista de produse si preturi in categoria aleasa
  - lista de produse si preturi se citeste dintr-un **fișier**
  - se preia comanda si se calculeaza suma totala
- Optional
  - se creaza o pagina prin care vanzatorul poate **modifica** preturile si produsele

# Rezultat

## Categorii Produse

Alegeti categoria:

Nr.	Categorie	Total Produse
1	<a href="#">Papetarie</a>	3
2	<a href="#">Instrumente</a>	3
3	<a href="#">Audio-video</a>	3
4	<a href="#">Calculatoare</a>	3
5	<a href="#">Jucarii</a>	2

Total produse: 14

## Magazin online Firma X SRL

### Realizati comanda

Nr.	Produs	Pret	Cantitate
1	Carti	100	<input type="text" value="1"/>
2	Caiete	50	<input type="text" value="2"/>
3	Penare	150	<input type="text" value="1"/>
4	Stilouri	125	<input type="text" value="0"/>
5	Creioane	25	<input type="text" value="0"/>

Trimite

## Magazin online Firma X SRL

### Rezultate comanda

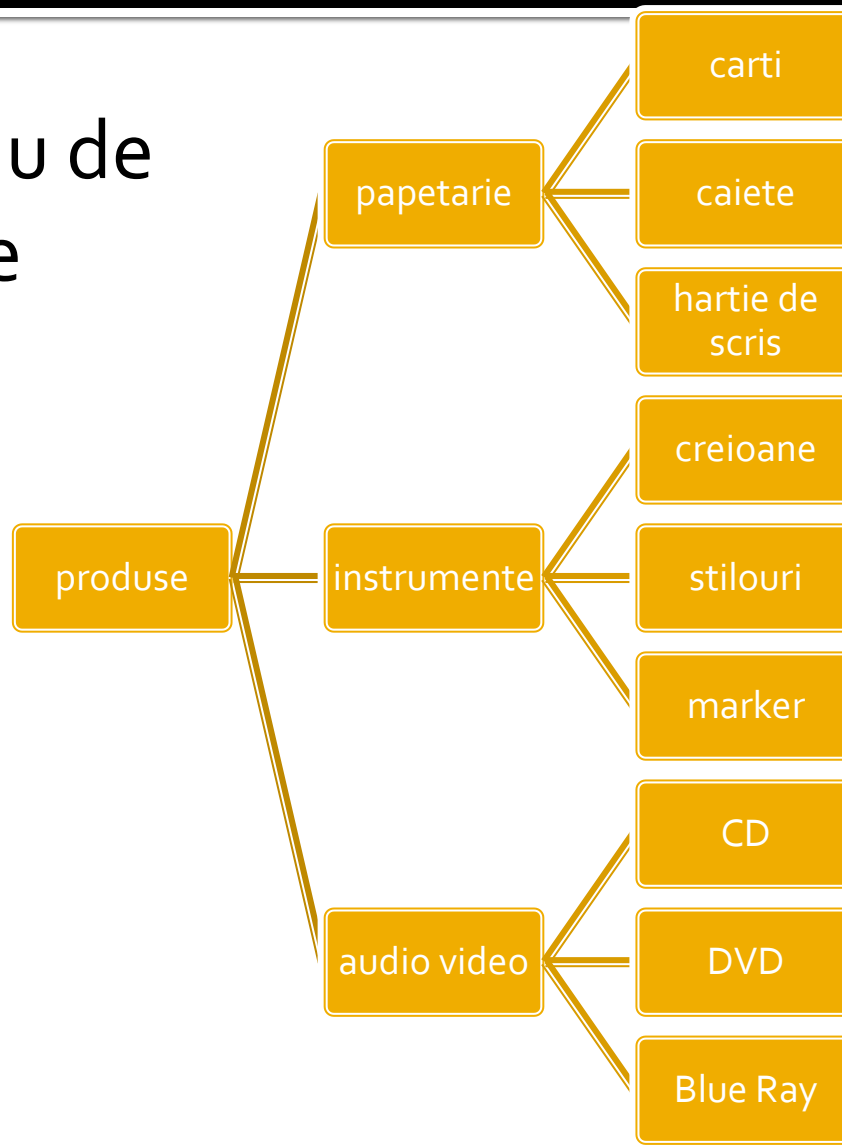
Pret total (fara TVA): 350

Pret total (cu TVA): 416.5

Comanda receptionata la data: 17/03/2010 ora 08:24

# Laborator 4

- exemplu de grupare

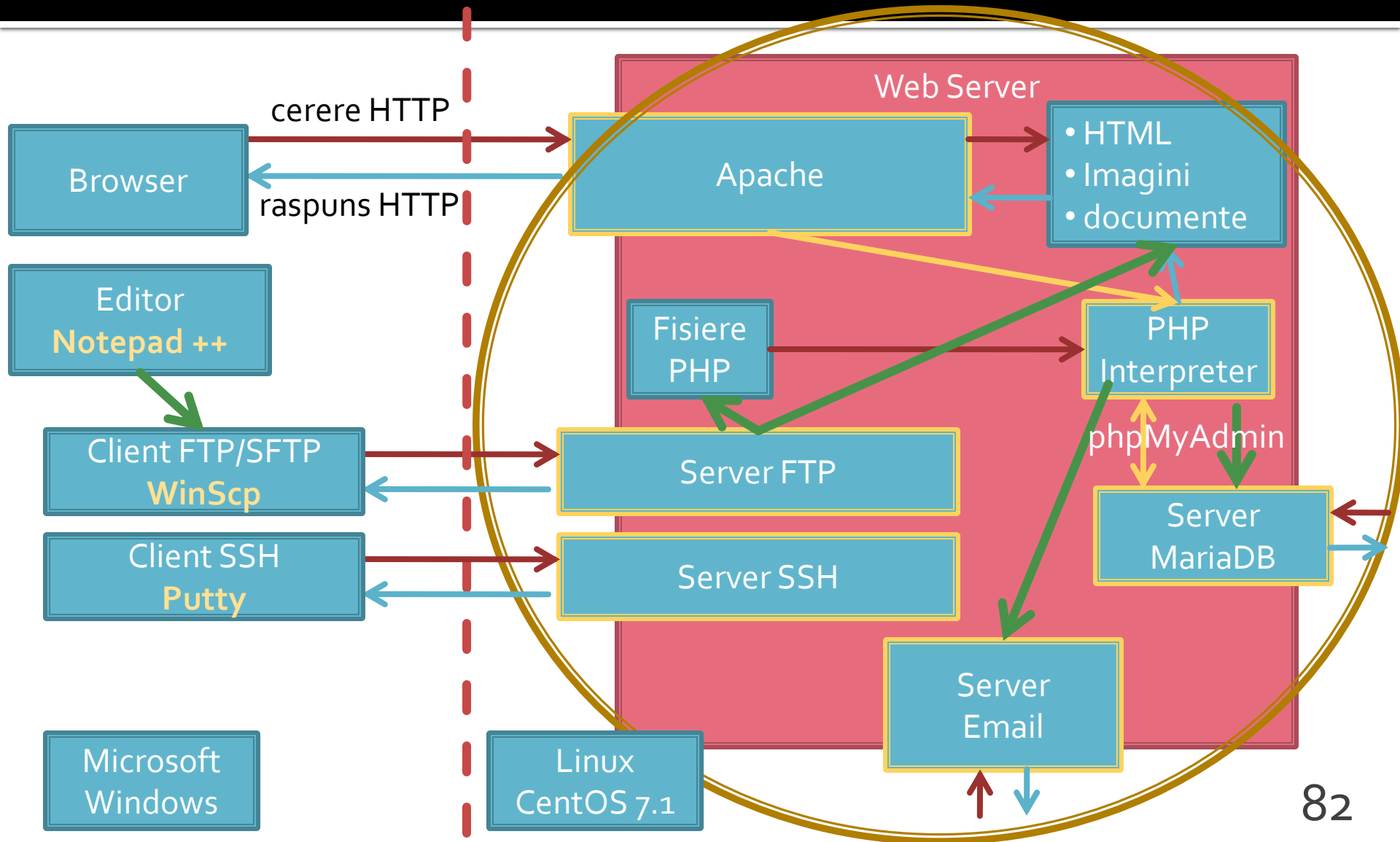




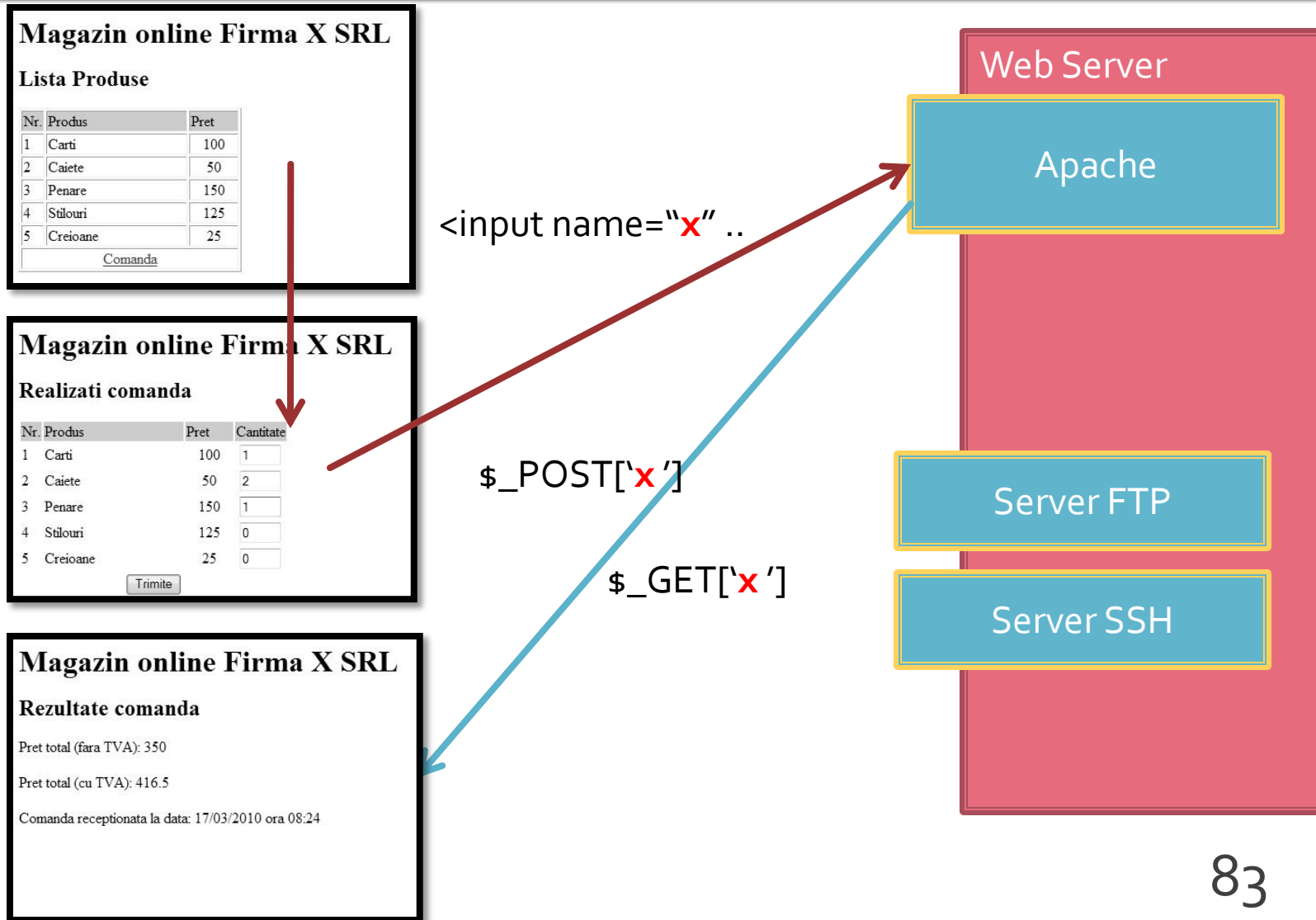
# Laborator 4 – Mod de lucru

- 1. Se introduce in surse facilitatea template
- 2. Se modifica sursele pentru lucru cu matrici
- 3. Se modifica sursele pentru a citi datele de pe disc
  - **anterior** se creaza fisierul text sau:
  - **o singura data** se salveaza datele (S72)
- 4. Se introduce structura suplimentara, categorie
  - se **creaza pagina** de selectie a categoriei, din care se va merge in lista de produse (utilizare \$\_GET – S76)
- 5. Optional: Se creaza o pagina care sa permita modificarea fisierului
  - numai pret/descriere, fara adaugare/schimbare produse

# Utilizare LAMP

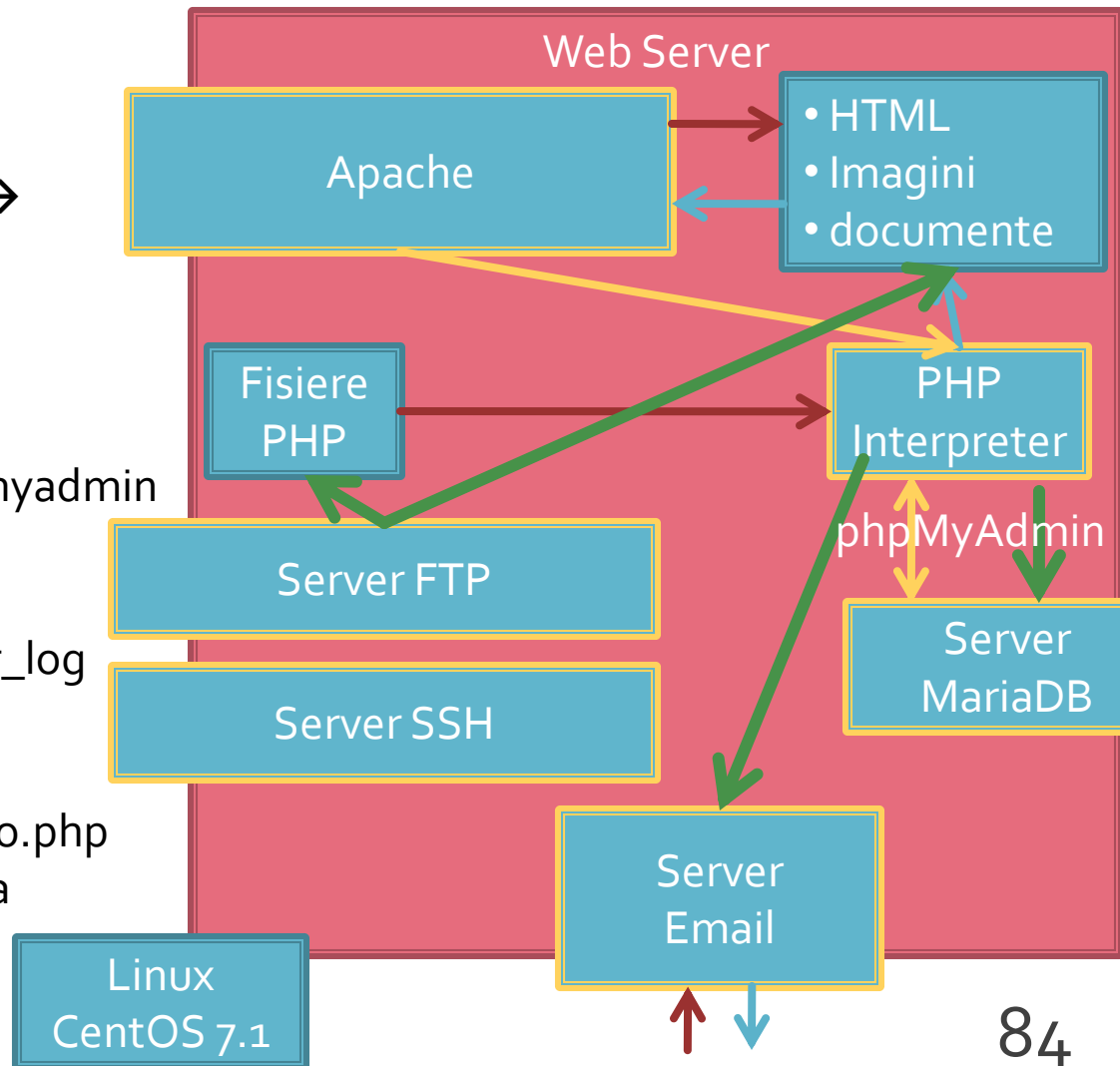


# Utilizare LAMP



# Utilizare LAMP

1. login → root:masterrc
2. ifconfig → 192.168.30.5
3. putty.exe → 192.168.30.5 → SSH → root:masterrc (remote login)
4. [alte comenzi linux dorite]
5. FTP → Winscp → SFTP → student:masterrc@192.168.30.5
6. MySql → http://192.168.30.5/phpmyadmin → root:masterrc
7. Apache Error Log →
  - 7a. putty → nano /var/log/httpd/error\_log
  - 7b. http://192.168.30.5/logfile.php (nonstandard)
8. PHP info → http://192.168.30.5/info.php
9. daca serviciul DHCP duce la oprirea Apache: `service httpd restart`



# Faza de verificare/depanare

- Se recomanda utilizarea posibilitatii vizualizarii matricilor
  - In fisierul care receptioneaza datele
  - temporar pina la definitivarea codului
- utilizarea de cod "verbose" (manual) in etapele initiale de scriere a surselor PHP poate fi extinsa si la alte tipuri de date
  - singura (aproape) metoda de depanare(debug) in PHP
  - `<p>temp <?php echo "a=";echo $a; ?> </p>`

```
echo "<pre>";  
print_r($_POST);  
echo "</pre>";
```

# Depanare

```
echo "<pre>";  
print_r($_POST);  
echo "</pre>";
```

```
<p>temp <?php echo  
"a=";echo $a; ?> </p>
```

# Contact

- Laboratorul de microunde si optoelectronica
- <http://rf-opto.etti.tuiasi.ro>
- [rdamian@etti.tuiasi.ro](mailto:rdamian@etti.tuiasi.ro)