UNIVERSITATEA „TRANSILVANIA”, BRAŞOV

DEPARTAMENTUL PENTRU PREGĂTIREA PERSONALULUI DIDACTIC

LUCRARE METODICO-ŞTIINȚIFICĂ PENTRU OBȚINEREA GRADULUI DIDACTIC I

Coordonator,
Prof. dr. Daniela MARINESCU,
Univ. „Transilvania” - Braşov, Facultatea de Matematică şi Informatică

Autor,
Prof. de informatică, Adrian MODRIŞAN,
Colegiul Național „Andrei Şaguna”, Braşov

BRAŞOV, an şcolar 2008 - 2009

PROGRAMAREA PAGINILOR WEB FOLOSIND LIMBAJUL PHP

Coordonator,
Prof. dr. Daniela MARINESCU,
Univ. „Transilvania" - Braşov, Facultatea de Matematică şi Informatică

Autor,
prof. de informatică, Adrian MODRIŞAN,
Colegiul Național „Andrei Şaguna", Braşov

BRAŞOV, An şcolar 2008 - 2009

Cuprins:

Bibliografie:

1. Tudor Sorin şi Vlad Huţanu, *Crearea şi programarea paginilor WEB*, Bucureşti, L&S Infomat, 2004;
2. Vlad Huţanu şi Carmen Popescu, *Manual de Informatică Intensiv pentru clasa a XII-a*, Bucureşti, L&S Infomat, 2007;
3. Bogdan Pătruţ, *Internet pentru începători*, Bucureşti, Teora, 1998;
4. Traian Anghel, *Programarea în PHP. Ghid practic*, Iaşi, Polirom, 2005;
5. Julie C. Meloni, *Învaţă singur PHP, MySQL şi APACHE*, Bucureşti, Corint, 2005;
6. Larry Ulman, *PHP şi MySQL pentru site-uri web dinamice*, Bucureşti, Teora, 2006;

Bibliografie Internet :

7. http://wikipedia.org – enciclopedia liberă;
8. http://www.php.net – pagina oficială a grupului de lucru pentru dezvoltarea limbajului PHP;
9. http://www.w3schools.com – set gratuit de tutoriale şi documentaţii pentru programarea paginilor web.

1. INTRODUCERE

1.1. Justificarea temei alese.

Începând cu anii ’95, Internetul, sub aspectul său cel mai popular, şi anume al paginilor web, a cunoscut o amploare greu de imaginat.

Dacă la început, paginile web aveau un conținut simplu şi oarecum stângace, în zilele noastre aspectul acestora s-a schimbat radical. După doar 10 ani, în paralel cu evoluția tehnicii de calcul, au evoluat şi tehnicile de programare a acestora. Primele pagini permiteau doar navigarea prin conținutul lor, pe când în zilele noastre ele au o utilizare foarte largă, de la jocuri şi aplicații grafice dinamice la comerț pe Internet.

Aceste realități au trebuit să-şi găsească o reflectare şi asupra programelor şcolare din cadrul disciplinei „informatică”. Astfel, începând cu anul şcolar 2007-2008, în cazul claselor cu specializare „matematică-informatică”, programa clasei a XII-a la disciplina informatică a devenit mult mai flexibilă, permițând inițierea elevilor noile tehnici care s-au impus în domeniul programării paginilor web.

Lucrarea de față îşi propune în primul rând să fie o unealtă didactică, un manual şcolar care să îi poată ajuta pe elevi în procesul de învățare, conținând şi câtva detalii mai tehnice, cum ar fi instalarea suportului software de care este nevoie pentru aplicarea noțiunilor învățate.

Limbajul PHP este un limbaj de programare destinat în primul rând Internetului, aducând dinamică unei pagini web. Este unul dintre cele mai importante limbaje de programare web open-source (codul sursă este public, fiind accesibil tuturor) şi server-side (rularea sa nu se face pe calculatorul celui care vizualizează pagina, ci pe server-ul care o conține).

Este unul dintre cele mai folosite limbaje de programare server-side. Statisticile arată că la 1 mai 2008, suportul PHP este prezent pe 20 de milioane dintr-ul total de 70 de milioane de website-uri active din lumea întreagă.

Popularitatea de care se bucură acest limbaj de programare se datorează următoarelor sale caracteristici:

- **`Familiaritatea`** – sintaxa limbajului este foarte uşoară, fiind foarte la îndemână în special pentru programatorii care cunosc limbajul C;
- **`Simplitatea`** – sintaxa limbajului este destul de liberă. Nu este nevoie de includere de biblioteci sau de directive de compilare, codul PHP inclus într-un document fiind trecut între nişte marcaje speciale;
- **`Securitatea`** – PHP-ul pune la dispoziția programatorilor un set flexibil şi eficient de măsuri de siguranță;

- **Flexibilitatea** – fiind apărut din necesitatea dezvoltării web-ului, PHP a fost modularizat pentru a ține pasul cu dezvoltarea diferitelor tehnologii. Nefiind legat de un anumit server web, PHP-ul a fost integrat pentru numeroasele servere web exitente: Apache, IIS, Zeus, etc.
- **Gratuitatea** – este, probabil, cea mai importantă caracteristică a PHP-ului. Dezvoltarea PHP-ului sub licența open-source a determinat adaptarea rapidă a sa la nevoile web-ului, eficientizarea şi securizarea codului.

1.2. Descrierea capitolelor lucrării.

În capitolul al II-lea al acestei lucrări (Fundamente teoretice ale paginilor WEB – limbajul HTML) mi-am propus o parcurgere ceva mai amănunțită, sub forma unui tutorial, a limbajului HTML standard, descriind tag-urile cele mai importante şi exemplificând aceste descrieri cu mici aplicații. Capitolul se încheie cu o trecere în revistă, în care există câteva exemple comentate, a tehnicilor de programare dinamice ale unei pagini web, care rămân însă tot pe domeniul HTML.

Acest capitol este mai mult decât necesar, din cauză că PHP nu face altceva decât să ruleze programe în urma căruia este generat cod HTML. Nu putem aşadar vorbi de limbajul PHP fără a cunoaşte HTML

În capitolul al III-lea (Limbajul PHP – facilități ale acestuia) am făcut, la fel ca şi în capitolul al II-lea, o parcurgere mai amănunțită a elementelor limbajului PHP, cu exemple. Totodată, în acest capitol există şi câteva detalii tehnice despre instalarea pachetelor software necesare rulării.

În prima parte a capitolului al IV-lea (Aplicații practice şi metodologice), am reluat, din considerente metodice şi din perspectiva programării pe Internet, o serie de algoritmi studiați la disciplina informatică în clasele a IX-a, a X-a şi a XI-a.

De remarcat faptul că transcrierea algoritmilor propriu-zişi în PHP rămâne foarte similară limbajului C++. Principalul element care face diferența este dat de interfața acestora, lucru normal de altfel, deoarece aplicațiile PHP sunt destinate în primul rând utilizării lor pe Internet, deci de către public foarte larg. Este motivul pentru care interfața trebuie să prezinte un grad ridicat de interactivitate (adesea vorbim de "interfață inteligentă") astfel încât să permită o comunicare cât mai simplă dintre utilizator şi aplicație.

În a doua parte a aceluiaşi capitol mi-am propus abordarea interdisciplinară matematică-informatică a geometriei analitice plane, studiate de către elevi în clasa a XI-a.

Tot în această parte am propus o serie de aplicații care realizează reprezentări grafice de fractali.

Ultimul capitol al lucrării (Considerații metodologice) conține, în prima sa parte, o analiză, din punct de vedere metodic, al modului de adaptare şi de reacție al elevilor de clasa a XII-a la noul conținut al programei şcolare.

În ultima parte propune analiza unui curs opțional, care atinge un alt subiect de actualitate al informaticii, şi anume programarea într-un limbaj vizual.

Lucrarea este însoțită şi de un CD-rom cu următorul conținut:

A) Pachetul software **`xampp`**, necesar rulării server-ului **`http`**, limbajului **`php`** şi bazei de date **`mysql`**;

B) Prezenta lucrare, în format digital (**`.pdf`**);

C) Codurile sursă ale exemplelor utilizate pe parcursul lucrării (fişiere **`.html`** respectiv **`.php`** – fiecare exemplu prezent în cadrul lucrării va avea o referire la un astfel de fişier, de exemplu: **`apl10.html`**, sau **`apl30.php`**).

2. FUNDAMENTE TEORETICE ALE PAGINILOR WEB – LIMBAJUL HTML

2.1. Scurt istoric al apariției Internetului şi funcționarea sa. Rețeaua WWW. Conceptul de Hipertext.

Istoria Internetului începe cu anul 1968, când guvernul S.U.A. intenționa să interconecteze universitățile, departamentele militare şi de apărare ale țării, astfel încât ele să coopereze în cadrul unor proiecte de cercetare comune. Astfel, s-a format o agenție numită *Advanced Research Projects Agency (ARPA)*. Una din cheile proiectului punea în discuție faptul că, stocarea tuturor informațiilor pe un singur calculator nu ar fi fost deloc sigură, fie din cauză că acesta ar putea fi țintă vulnerabilă a unui eventual atac, fie pur şi simplu din cauză că acestea ar putea fi pierdute în cazul unei defecțiuni tehnice majore. O metodă de a face față unei asemenea situații ar fi de a copia şi distribui informațiile pe mai multe calculatoare, în întreaga țară, folosind o rețea.

În 1975, câteva dintre limbajele sau protocoalele pe care calculatoarele le foloseau pentru a comunica între ele s-au standardizat. Majoritatea universităților importante şi a departamentelor de apărare din S.U.A. s-au legat împreună într-o rețea numită *DARPANET*, toate calculatoarele folosind acelaşi protocol pe care astăzi îl cunoaştem sub denumirea de TCP/IP. Rețeaua, cu timpul, a fost înlocuită de mai multe rețele, care astăzi împânzesc globul pământesc.

Începând cu anul 1980, mai multe colegii şi universități au fost conectate la Internet. Acest lucru a permis universităților să-şi împartă informații despre cercetările lor, programe şi ştiri recente. În anii ’90 Internetul s-a deschis şi în scopuri comerciale. În curând, multe alte căi de utilizare a informațiilor transmise prin intermediul acestei gigantice rețele au fost dezvoltate.

În prezent, este posibil să foloseşti Internetul pentru a trimite scrisori electronice pe întregul glob în doar câteva secunde. Poți căuta informații despre orice subiect doreşti. Expresia „World Wide Web” (***WWW***) defineşte o colecție de documente care se întinde în câteva sute de milioane de calculatoare.

Principiul de bază al funcționării Internetului constă în faptul că două sau mai multe calculatoare pot comunica între ele. Pentru ca acest lucru să fie posibil este necesar să existe un „protocol”, adică un ansamblu de norme care trebuie respectate de calculatoare (deci de programele care rulează pe ele) pentru ca schimbul de date să poată avea loc.

Normele se referă la:

- găsirea calculatorului destinatar al transferului de date;
- transmiterea efectivă a datelor;
- modalități prin care expeditorul comunică faptul că au fost transmise toate datele, iar destinatarul comunică faptul că le-a recepționat;

- compresia datelor: prin aplicarea anumitor algoritmi matematici, datele care urmează să fie expediate sunt prelucrate de aşa natură, încât să fie memorate prin utilizarea unui spațiu cât mai mic de memorie. Prin urmare, transmiterea lor durează mai puțin. Invers, la destinație sunt decompresate prin utilizarea aceloraşi algoritmi matematici;
- identificarea erorilor care pot interveni în transmiterea datelor: şi aici există mai mulți algoritmi care permit identificarea şi corectarea erorilor.

Standardul care s-a impus în ceea ce priveşte Internetul, constă în protocolul TCP/IP. Numele este de fapt, numele comun al unei familii de protocoale utilizate pentru transferul datelor în rețea. Orice calculator conectat la Internet are o adresă, numită adresă IP (Internet Protocol Address). O adresă IP este alcătuită din 4 numere între 0 şi 255, prin urmare o astfel de adresă ocupă 4 octeți. Cum transmiterea datelor la un moment dat se face între două calculatoare, datele se transmit de la o adresă IP la alta.

Protocolul IP (Internet Protocol) reglementează transmiterea datelor de la o adresă IP la alta. Datele sunt transmise divizate în pachete. În acest fel, se preîntâmpină monopolizarea transmisiei în rețea doar de către un singur utilizator.

Protocolul TCP (Transmission Control Protocol): de la plecare, un program TCP împarte informația de transmis în mai multe pachete IP. Acestea sunt transmise la destinație prin intermediul rețelei. O dată ajunse la destinație, un alt program TCP asamblează şi aranjează în ordinea corectă pachetele IP de date primite. Firește, din cauza unor probleme hardware, unele pachete se pot pierde pe drum. Protocolul TCP se ocupă şi de acest lucru. Astfel, când împachetează datele într-un plic „IP", protocolul TCP al expeditorului adaugă şi un număr (numit sumă de control) care va permite destinatarului să se asigure de faptul că datele primite sunt corecte. Receptorul recalculează suma de control şi o compară cu cea transmisă de emițător. Dacă ele nu sunt identice, înseamnă că a apărut o eroare în timpul transmisiei, motiv pentru care protocolul TCP anulează acel pachet, cerând retransmiterea sa.

Bazele World Wide Web (WWW) au fost puse în 1989 la Centrul European de Cercetări Nucleare (CERN) în Geneva (Elveția). Propunerea inițială de creare a unei colecții de documente având legături între ele a fost făcută de Tim Berners-Lee în martie 1989. Această propunere a apărut în urma problemelor de comunicare pe care le întâmpinau echipele de cercetători ce foloseau centrul, chiar şi folosind poşta electronică.

Primul server web folosit de Tim Berners-Lee a apărut nu mult înainte de decembrie 1991, când s-a făcut prima lui demonstrație publică. Studiul a fost continuat prin apariția primei aplicații grafice Mosaic, în februarie 1993, realizată de cercetătorul Marc Andreessen de la centrul universitar National Center for Supercomputing Applications (NCSA) din orașul Urbana-Champaign din statul federal Illinois, SUA. Ulterior WWW-ul a evoluat până la ceea ce este astăzi, un serviciu integrativ şi multimedial, având ca suport fizic Internetul.

Practic, WWW este un sistem de documente şi informații de tip hipertext legate ele între ele, care pot fi accesate prin rețeaua mondială de Internet. Documentele, care rezidă în diferite locații pe diverse calculatoare-server, pot fi regăsite cu ajutorul unei adrese unice. Hipertextul este prelucrat cu un ajutorul unui program de navigare în web numit *browser* care descarcă paginile web de pe un server web şi le afişează pe un terminal.

Prin conceptul de hipertext se înțelege o formă de document electronic, o metodă de organizare a informațiilor în care datele sunt memorate într-o rețea de noduri şi legături, putând fi accesată prin intermediul programelor de navigare interactivă, şi manipulată de un editor structural. Conceptul de bază în definirea hipertextului este "legătura" (link-ul), fie în cadrul aceluiaşi document, fie către alt document. Legătura de tip link permite organizarea neliniară a informațiilor. Un sistem hipertext permite autorului său să creeze aşa-numite "noduri", să le lege între ele, iar unui cititor navigarea de la un nod la altul. Astfel un nod reprezintă un concept putând conține orice fel de informație: text, grafică, imagini, animații, sunete, etc. Nodul sursă al unei legături se numeşte "referință" iar cel destinație "referent" sau ancoră, punctele de legătură din respectivele noduri fiind marcate. Activarea marcajelor unei legături duce la vizualizarea nodurilor. Asocierea cu unele elemente mediale a dus la extinderea noțiunii de hipertext către "hipermedii".

2.2. Despre website-uri.

Noțiunea de website (sau pur şi simplu site, ori „site web”) desemnează o grupă de pagini web multimediale (conținând texte, imagini fixe, imagini mişcătoare şi chiar sunete), accesibile în Internet în principiu oricui, de obicei pe o temă anume, şi care sunt conectate între ele prin aşa-numite hyperlinkuri. Diversele situri web pot fi oferite de către o companie, un proiect, o rețea de utilizatori, o persoană particulară, o administrație publică şi multe altele.

Pentru crearea paginilor web s-a impus limbajul HTML (**H**yper**T**ext **M**arkup **L**anguage) – un limbaj de marcare, al cărui scop constă în prezentarea într-un anumit format a informațiilor: paragrafe, tabele, fonturi, culori, ş.a.m.d.

Calculatorul pe care se găseşte site-ul se numeşte „server”, iar calculatoarele care accesează conținutul site-ului se numesc „client”.

Orice calculator client trebuie să dispună de un program specializat, numit „browser”, cu ajutorul căruia să se poată interpreta şi deci vizualiza fişierele HTML.

Pe server trebuie să se găsească un program care răspunde cererilor browser-ului aflat pe calculatorul client. Cererea efectuată de către browser şi răspunsul server-ului se fac prin respectarea unui anumit protocol. Acest protocol se numeşte HTTP (**H**yper**T**ext **T**ransfer **P**rotocol).

2.3. HTML standard – limbaj descriptiv al unei pagini WEB.

HTML este un limbaj de marcare orientat către prezentarea documentelor text pe o singura pagină.

Utilizând un software de redare specializat, numit agent utilizator HTML (cel mai bun exemplu de astfel de software fiind browserul web) HTML furnizează mijloacele prin care conținutul unui document poate fi adnotat cu diverse tipuri de metadate şi indicații de redare. Indicațiile de redare pot varia de la decorațiuni minore ale textului (cum ar fi specificarea faptului că un anumit cuvânt trebuie subliniat sau că o imagine trebuie introdusă) până la scripturi sofisticate, hărți de imagini şi formulare. Metadatele pot include informații despre titlul şi autorul documentului, informații structurale despre cum este împărțit documentul în diferite segmente, paragrafe, liste, titluri etc. şi informații cruciale care permit ca documentul să poată fi legat de alte documente pentru a forma astfel hiperlink-uri.

HTML este un format text proiectat pentru a putea fi citit şi editat de oameni utilizând un editor de text simplu. Totuşi scrierea şi modificarea paginilor în acest fel solicită cunoştințe solide de HTML şi este consumatoare de timp. Editoarele grafice cum ar fi Macromedia Dreamweaver sau Microsoft FrontPage permit ca paginile web sa fie tratate asemănător cu documentele Word, dar cu observația că aceste programe generează un cod HTML care este de multe ori de proastă calitate.

HTML se poate genera direct utilizând tehnologii de codare din partea serverului cum ar fi PHP, JSP sau ASP.

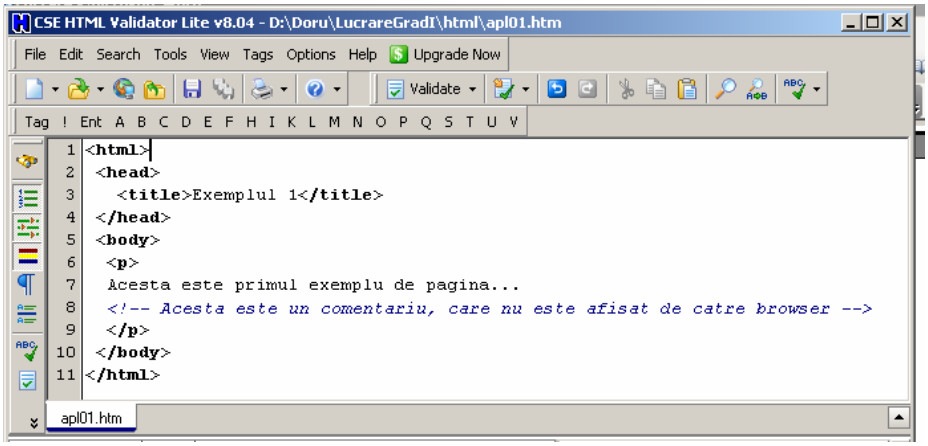
2.3.1 Scrierea de cod HTML. Editoare specializate şi validatoare HTML.

Crearea unui fişier HTML este foarte simplă, putând fi făcută cu ajutorul oricărui editor de text. Totuşi, pentru a avea un control ridicat asupra corectitudinii codului scris, este recomandat să utilizăm un editor specializat, care să pună în evidență diversele elemente de marcare (TAG-uri, numite şi „elemente" sau „etichete") sau, mai mult, să poată verifica şi detecta erorile.

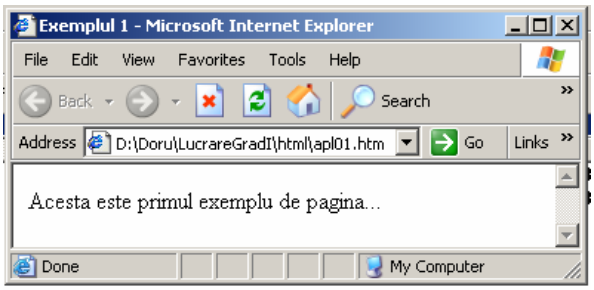
Din categoria editoarelor care pun în evidență diferitele elemente face parte editorul *Notepad++*, iar din categoria validatoarelor face parte *CSE HTML Validator Lite*, ambele fiind gratuite şi putând fi descărcate de pe Internet.

2.3.2. Structura de bază a unei pagini.

Structura de bază a unei pagini HTML este următoarea (**`apl001.html`**):



Iată şi modul în care pagina de mai sus este vizualizată în Internet Explorer:



Din analiza exemplului observăm că:

- O pagină începe cu tag-ul **`<HTML>`** şi se termină cu tag-ul **`</HTML>`**;
- O pagină conține un antet (HEAD) şi corpul propriu-zis (BODY);
- Antetul este cuprins între etichetele **`<HEAD>`** şi **`</HEAD>`**;
- Corpul este cuprins între etichetele **`<BODY>`** şi **`</BODY>`**;

• Opțional, antetul poate conține titlul paginii, cuprins între tag-urile **<TITLE>** și **</TITLE>**. Titlul apare pe bara de titlu a ferestrei afișate în browser.

• Corpul poate conține texte și/sau imagini. În exemplu, pagina conține textul „Acesta este primul exemplu de pagina...”

• Comentariile, care nu sunt afișate de către browser, pot fi scrise între tag-urile **<!--** și **-->**.

• Numele tag-urilor nu sunt case sensitive, deci pot fi scrise atât cu litere mici cât și cu litere mari. În continuare, pentru a le pune în evidență, le vom scrie cu litere mari.

2.3.3. Paragrafe. Atribute ale unui tag.

În general, textele conținute de o pagină se pot găsi în mai multe paragrafe. Un paragraf se introduce între tag-urile **<P>** ... **</P>**.

La afișare, două paragrafe consecutive vor fi separate printr-o linie goală.

Tag-ul **</P>** poate lipsi; un nou paragraf poate fi detectat prin tag-ul **<P>**.

În cadrul unui fișier HTML, Enter-ul nu are nici un efect. De asemenea, dacă două cuvinte ale unui paragraf sunt separate prin mai multe spații sau alte caractere albe (enter-uri, tab-uri), browser-ul afișează doar un singur spațiu.

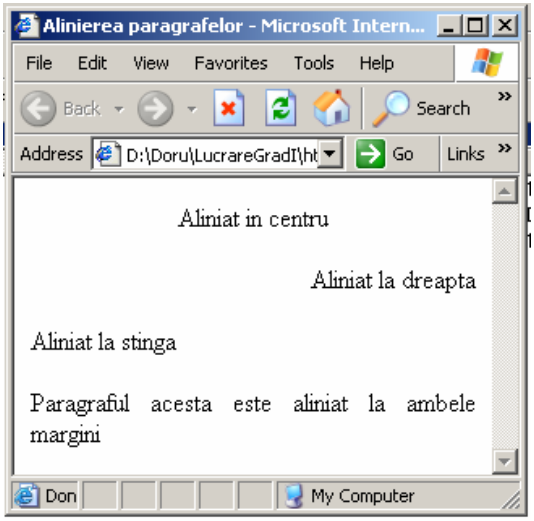
Majoritatea tag-urilor li se pot specifica atribute. Acestea determină comportamentul mai amănunțit al elementului respectiv.

Un atribut se specifică înainte de închiderea parantezei unghiulare a tag-ului (**>**) prin **nume_atribut=”valoare”**.

În cazul paragrafului, atributul **align** controlează alinierea textului din cadrul paragrafului. Dacă acest atribut nu este prezent, alinierea este făcută în mod implicit la stânga. Acest atribut poate lua una dintre valorile **center**, **left**, **right**, **justify**, ca în exemplul de mai jos (**apl002.html**):

```
<HTML>
  <HEAD>
    <TITLE>Alinierea paragrafelor</TITLE>
  </HEAD>
  <BODY>
    <P align="center">Aliniat in centru</P>
    <P align="right">Aliniat la dreapta</P>
    <P align="left">Aliniat la stinga</P>
    <P align="justify">Paragraful acesta este aliniat la ambele margini</P>
  </BODY>
</HTML>
```

Iată pagina al cărei cod tocmai a fost prezentat, vizualizată în Internet Explorer:



2.3.4. Elemente care permit formatarea textului.

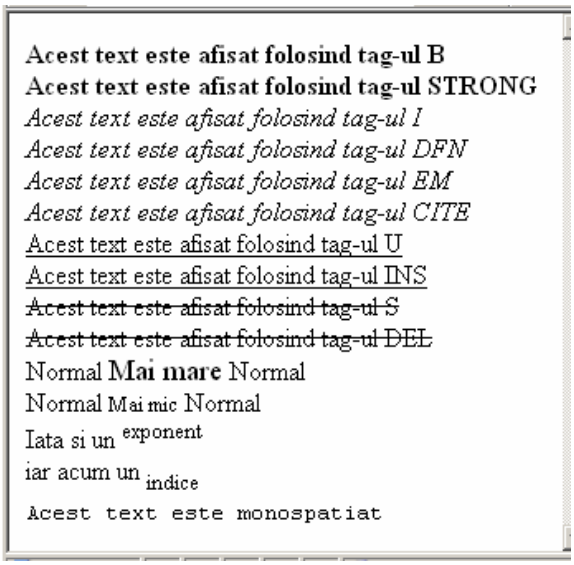
- **
** : Are ca efect forţarea afişării a ceea ce urmează pe rândul următor. Acest tag nu creează un nou paragraf (să ne reamintim că între două paragrafe este automat lăsată o linie vidă)
- **<B>...</B>** : Are rolul de a afişa bold (îngroşat) textul cuprins între cele două tag-uri ale sale. Un tag sinonim al lui **<B>** este: **<STRONG>...</STRONG>**
- **<I>...</I>** : Are rolul de a afişa italic (înclinat) textul cuprins între cele două tag-uri ale sale. Tag-uri sinonime ale lui **<I>** sunt: **<EM>...</EM>**, **<DFN>...</DFN>**,**<CITE>...</CITE>**.
- **<U>...</U>** : Are rolul de a afişa subliniat textul cuprins între cele două tag-uri ale sale. Un tag sinonim al lui **<U>** este: **<INS>...</INS>**
- **<S>...</S>** : Are rolul de a afişa tăiat (cu o linie orizontală) textul cuprins între cele două tag-uri ale sale. Un tag sinonim al lui **<S>** este: **<DEL>...</DEL>**
- **<BIG>...</BIG>** : Are rolul de a afişa textul cuprins între cele două tag-uri ale sale mai mare decât textul în care este cuprins.
- **<SMALL>...</SMALL>** : Are rolul de a afişa textul cuprins între cele două tag-uri ale sale mai mic decât textul în care este cuprins.
- **^{...}** : Are rolul de a afişa textul cuprins între cele două tag-uri ale sale mai sus (ca o putere)
- **_{...}** : Are rolul de a afişa textul cuprins între cele două tag-uri ale sale mai sus (ca un indice)

• **<TT>...</TT>** : Are rolul de a afişa textul cuprins între cele două tag-uri ale sale mai sus monospaţiat (toate caracterele ocupă aceeaşi lungime – practic, se foloseşte fontul Courier New)

În cod-ul HTML de mai jos găsiți toate aceste tag-uri exemplificate (**ap1003.html**):

```
<HTML>
  <HEAD>
     <TITLE>Formatarea textului</TITLE>
  </HEAD>
  <BODY>
    <P>
      <B>Acest text este afisat folosind tag-ul B</B> <BR>
      <STRONG>Acest text este afisat folosind tag-ul STRONG</STRONG> <BR>
      <I>Acest text este afisat folosind tag-ul I</I> <BR>
      <DFN>Acest text este afisat folosind tag-ul DFN</DFN> <BR>
      <EM>Acest text este afisat folosind tag-ul EM</EM> <BR>
      <U>Acest text este afisat folosind tag-ul U</U> <BR>
      <INS>Acest text este afisat folosind tag-ul INS</INS> <BR>
      <S>Acest text este afisat folosind tag-ul S</S> <BR>
      <DEL>Acest text este afisat folosind tag-ul DEL</DEL> <BR>
      Normal <BIG>Mai mare</BIG> Normal <BR>
      Normal <SMALL>Mai mic</SMALL> Normal <BR>
      Iata si un <SUP>exponent</SUP> <BR>
      iar acum un <SUB>indice</SUB> <BR>
      <TT>Acest text este monospatiat</TT>
    </P>
  </BODY>
</HTML>
```

Acest cod vizualizat în browser arată în felul următor:



• Pentru scrierea titlurilor se utilizează tag-urile **<H1>...<H1>**, **<H2>...<H2>**, . . . , **<H6>...<H6>**. Practic, în funcție de numărul de după H mărimea fontului diferă (**<H1>** utilizează fontul de dimensiune maximă, **<H6>** fontul de dimensiune minimă) iar textul care apare între tag-uri este scris îngroşat (bold).

• Pentru stabilirea font-ului se foloseşte tag-ul **<FONT>...<FONT>**. Atributele acestuia sunt:

- **face** indică numele font-ului
- **size** indică mărimea (trebuie să fie un număr cuprins între 1 şi 7. Implicit este 3)

- **color** permite specificarea culorii. Aceasta se specifică fie prin intermediul constantelor predefinite ale HTML-ului (numele englezesc al culorii) fie prin componentele sale de Roşu, Verde şi Albastru exprimate în hexazecimal, de forma #RRGGBB (vom detalia aceste constante de culoare ceva mai încolo).

Iată un exemplu de utilizare al lor (**ap1004.html**):

```
<HTML>
  <HEAD>
    <TITLE>Exemplificare titluri si font</TITLE>
  </HEAD>
  <BODY>
    <P>
      <H1>Acesta este un titlu de tip H1</H1>
      <H2>Acesta este un titlu de tip H2</H2>
      <H3>Iar acesta este un titlu de tip H3</H3>
      <FONT face="arial" color="blue" size="4">
        Acest text este scris cu fontul Arial, albastru, dimensiune 4
      </FONT><BR>
      Iar acest text este scris normal<BR>
    </P>
    <P>
      Iata si culorile cucubeului, scrise cu font-ul Comic Sans MS,
      bold, dimensiune 7:<br>
      <B>
        <FONT face="Comic Sans MS" size="7" color="red">R</FONT>
        <FONT face="Comic Sans MS" size="7" color="orange">O</FONT>
        <FONT face="Comic Sans MS" size="7" color="yellow">G</FONT>
        <FONT face="Comic Sans MS" size="7" color="green">V</FONT>
        <FONT face="Comic Sans MS" size="7" color="blue">A</FONT>
        <FONT face="Comic Sans MS" size="7" color="darkblue">I</FONT>
        <FONT face="Comic Sans MS" size="7" color="magenta">V</FONT>
      </B>
    </P>
  </BODY>
</HTML>
```

Vizualizat în browser:

Aşa cum am văzut, dacă în cadrul unui text din cadrul documentului HTML apare un grup de mai multe spații, în browser va fi afişat doar unul singur. Dacă dorim forțarea afişării unui spațiu, se foloseşte identificatorul special ** ** (ultimul caracter, "**;**", face parte din identificator)

2.3.5. Liste.

Acestea permit ca anumite enunțuri (texte, elemente) să fie numerotate sau marcate într-un anumit fel. O astfel de organizare poartă numele de liste.

În HTML distingem 3 feluri de liste:

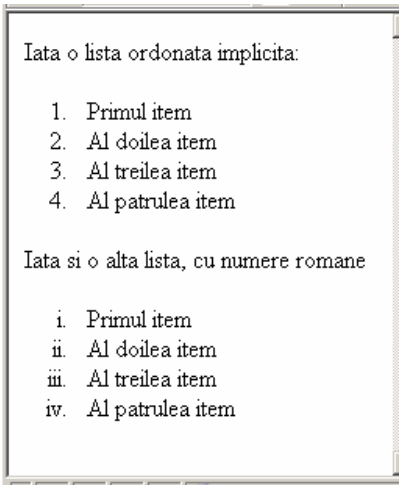
• ***Liste ordonate (Ordered Lists)***: sunt liste în care elementele sunt numerotate.

Inserarea lor în cadrul documentului HTML se face prin tag-urile **<OL>**...**</OL>**, elementele (itemii) lor fiind introduse între aceste două tag-uri prin **<LI>**...**</LI>** (tag-ul de sfârşit nefiind obligatoriu). Implicit, numerotarea se face cu numere arabe (1, 2, 3, ...). Ea poate fi modificată prin folosirea atributului **type** în cadrul tag-ului **OL**. Acesta poate lua una dintre valorile:

- **a** : numerotarea se va face cu litere mici (a, b, c, ...)
- **A** : numerotarea se va face cu litere mari (A, B, C, ...)
- **i** : numerotarea se va face cu numere romane mici (i, ii, iii, iv ...)
- **I** : numerotarea se va face cu numere romane mari (I, II, III. IV, ...)
- **1** : (implicit) numerotarea se va face cu numere arabe obişnuite (1, 2, 3, ...)

Iată un exemplu de cod şi vizualizarea sa în browser (**ap1005.html**):

```
<HTML>
  <HEAD>
     <TITLE>Liste</TITLE>
  </HEAD>
  <BODY>
    <P>
    Iata o lista ordonata implicita:
    <OL>
       <LI>Primul item</LI>
       <LI>Al doilea item</LI>
       <LI>Al treilea item</LI>
       <LI>Al patrulea item</LI>
    </OL>
    Iata si o alta lista, cu numere romane
    <OL type="i">
       <LI>Primul item</LI>
       <LI>Al doilea item</LI>
       <LI>Al treilea item</LI>
       <LI>Al patrulea item</LI>
    </OL>
    </P>
  </BODY>
</HTML>
```



• ***Liste neordonate (Unordered Lists)***: sunt liste în care elementele nu sunt numerotate, ci în dreptul fiecăruia este afişat un marcator.

Inserarea lor în cadrul documentului HTML se face prin tag-urile **<UL>**...**</UL>**, elementele (itemii) lor fiind introduse între aceste două tag-uri prin **<LI>**...**</LI>** (tag-ul de sfârşit nefiind obligatoriu).

Implicit, marcarea lor se face prin cerculeţe pline. Ea poate fi modificată prin folosirea atributului **type** în cadrul tag-ului **UL**. Acesta poate lua una dintre valorile:

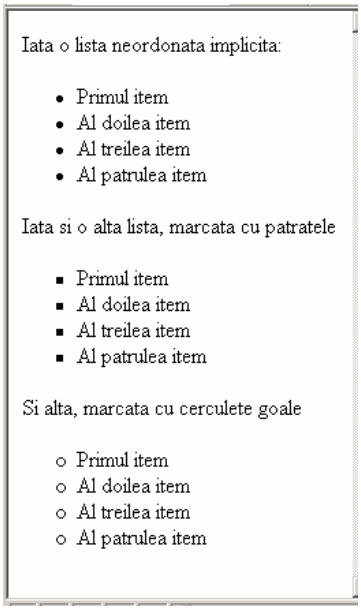
- **disc** : marcarea se face cu cerculeţe pline (implicit)

- **square** : marcarea se face cu pătrăţele

- **circle** : marcarea se face cu cerculeţe goale

Iată un exemplu de cod şi vizualizarea sa în browser (**apl006.html**):

```
Iata o lista neordonata implicita:
<UL>
   <LI>Primul item</LI>
   <LI>Al doilea item</LI>
   <LI>Al treilea item</LI>
   <LI>Al patrulea item</LI>
</UL>
Iata si o alta lista, marcata cu patratele
<UL type="square">
   <LI>Primul item</LI>
   <LI>Al doilea item</LI>
   <LI>Al treilea item</LI>
   <LI>Al patrulea item</LI>
</UL>
Si alta, marcata cu cerculete goale
<UL type="circle">
   <LI>Primul item</LI>
   <LI>Al doilea item</LI>
   <LI>Al treilea item</LI>
   <LI>Al patrulea item</LI>
</UL>
```


Iata o lista neordonata implicita:

- Primul item
- Al doilea item
- Al treilea item
- Al patrulea item

Iata si o alta lista, marcata cu patratele

- Primul item
- Al doilea item
- Al treilea item
- Al patrulea item

Si alta, marcata cu cerculete goale

- Primul item
- Al doilea item
- Al treilea item
- Al patrulea item

- ***Liste de definiţie (Definition Lists)***: au rolul de a descrie o listă de definiţii.

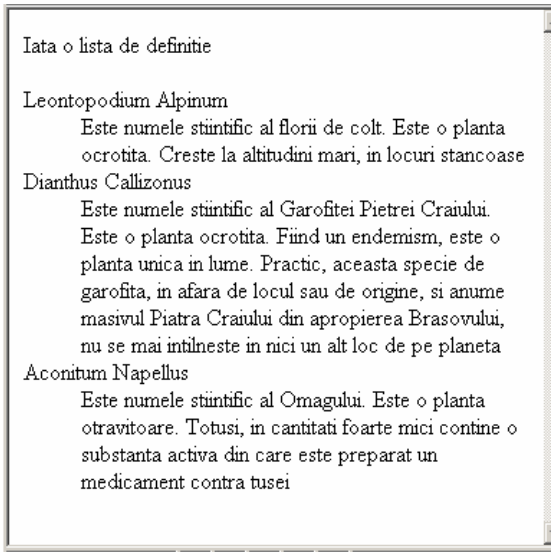
Inserarea lor în cadrul documentului HTML se face prin tag-urile **<DL>**...**</DL>**. Elementele lor sunt de două tipuri:

- Termenul care este definit: este introdus între tag-urile **<DT>**...**</DT>** (tag-ul de sfârşit nefiind obligatoriu).

- Definiţia propriu-zisă: este introdusă între tag-urile **<DD>**...**</DD>** (tag-ul de sfârşit nefiind obligatoriu).

Iată un exemplu de cod şi vizualizarea sa în browser (**apl007.html**):

```
Iata o lista de definitie:
<DL>
  <DT>Leontopodium Alpinum</DT>
  <DD>Este numele stiintific al florii de colt.
  Este o planta ocrotita. Creste la altitudini
  mari, in locuri stancoase</DD>
  <DT>Dianthus Callizonus</DT>
  <DD>Este numele stiintific al Garofitei
  Pietrei Craiului. Este o planta ocrotita.
  Fiind un endemism, este o planta unica in
  lume. Practic, aceasta specie de garofita,
  in afara de locul sau de origine, si anume
  masivul Piatra Craiului din apropierea
  Brasovului, nu se mai intilneste in nici un
  alt loc de pe planeta</DD>
  <DT>Aconitum Napellus</DT>
  <DD>Este numele stiintific al Omagului.
  Este o planta otravitoare. Totusi, in cantitati
   foarte mici contine o substanta activa din care
   este preparat un medicament contra tusei</DD>
</DL>
```


Iata o lista de definitie

Leontopodium Alpinum
Este numele stiintific al florii de colt. Este o planta ocrotita. Creste la altitudini mari, in locuri stancoase

Dianthus Callizonus
Este numele stiintific al Garofitei Pietrei Craiului. Este o planta ocrotita. Fiind un endemism, este o planta unica in lume. Practic, aceasta specie de garofita, in afara de locul sau de origine, si anume masivul Piatra Craiului din apropierea Brasovului, nu se mai intilneste in nici un alt loc de pe planeta

Aconitum Napellus
Este numele stiintific al Omagului. Este o planta otravitoare. Totusi, in cantitati foarte mici contine o substanta activa din care este preparat un medicament contra tusei

2.3.6. Imagini.

Tag-ul utilizat pentru inserarea unei imagini în documentul HTML este **<IMG>**. Forma generală a acestui element este **<IMG atribute>**. Acest tag nu are şi formă de închidere.

Atributele sale sunt:

• **src** identifică fişierul efectiv de pe disc, ce conţine imaginea respectivă. Dacă imaginea se află în directorul curent, se specifică doar numele şi extensia sa. Dacă se află într-un subdirector, acesta se specifică înaintea numelui şi extensiei imaginii, separat prin caracterul /. Imaginile recunoscute de majoritatea browser-elor internet sunt de tip .jpg, .gif, .png

• **align** specifică tipul de aliniere al imaginii în raport cu textul în cadrul căruia se află. Acesta poate lua una dintre valorile următoare:

- **right** : imaginea se aliniază în dreapta, iar textul care urmează este scris în locul rămas liber, în stânga acesteia;

- **left** : imaginea se aliniază în stânga, iar textul care urmează este scris în locul rămas liber, în dreapta acesteia;

- **top** : doar latura de sus a imaginii se aliniază cu rândul de text în cadrul căruia se află; următorul rând de text va fi afişat după imagine, ocupând întreaga lăţime a ecranului;

- **middle** : rândul de text în cadrul căruia se află imaginea se aliniază la jumătatea înălţimii acesteia; următorul rând de text va fi afişat după imagine, ocupând întreaga lăţime a ecranului;

- **bottom** : doar latura de jos a imaginii se aliniază cu rândul de text în cadrul căruia se află; următorul rând de text va fi afişat după imagine, ocupând întreaga lăţime a ecranului;

• Dacă dorim întreruperea unei alinieri de imagine de tip right sau left înainte ca textul să fi umplut spaţiul liber din stânga, respectiv dreapta acesteia, putem folosi tag-ul br, căruia îi adăugăm unul dintre atributele **clear="left"** sau **clear="right"** sau **clear="all"**, după caz.

• atributul **alt="text"** permite specificarea unui text alternativ ce va fi afişat fie dacă menţinem cursorul de mouse asupra imaginii, fie în locul imaginii propriu-zise, în cazul în care imaginea nu poate fi încărcată din cauza unei probleme de conexiune.

Iată câteva exemple, cu tot cu vizualizarea lor în browser:

1) Exemplu la folosirea atributului **align="right"** şi a atributului **alt="text"** :(**apl008.html**)

```
<P>Acest text este asezat inaintea imaginii<br>
   <IMG SRC="dog.jpg" align="right" alt="catelus">
   In schimb, acest text este aliniat in stinga imaginii,
   deoarece am folosit atributul align="right" in momentul
   in care am inserat imaginea in pagina noastra web prin
   intermediul tag-ului src.
</P>
```

2) Exemplu la folosirea opțiunii **align="right"** împreună cu tag-ul **<br clear="right">** (**apl009.html**):

```
<P>
  Acest text este asezat inaintea imaginii<br>
  <IMG SRC="dog.jpg" align="right" alt="catelus">
  Acest text, aliniat in stinga imaginii, il
  intrerupem fortat AICI
  <BR clear="right">
  In acest fel, restul textului se va alinia
  in mod obisnuit, sub imagine, restul spatiului
  din stinga raminind liber.
</P>
```

3) Exemplu la folosirea opțiunii **align="top"** (**apl010.html**):

```
<P>
  Acest text este asezat inaintea imaginii <br>
  Se observa ca
  <IMG SRC="dog.jpg" align="top" alt="catelus">
  doar primul rind al textului este aliniat cu
  latura de sus a imaginii, restul textului
  fiind afisat dupa imagine
</P>
```

4) Exemplu la folosirea opțiunii **align="middle"** (**apl011.html**):

```
<P>
  Acest text este asezat inaintea imaginii <br>
  Se observa ca
  <IMG        SRC="dog.jpg"        align="middle"
alt="catelus">
  doar primul rind al textului este aliniat la
  jumatatea inaltimii imaginii, restul textului
  fiind afisat dupa imagine
</P>
```

5) Exemplu la folosirea opțiunii **align="bottom"** (**apl012.html**):

```
<P>
  Acest text este asezat inaintea imaginii <br>
  Se observa ca
  <IMG SRC="dog.jpg" align="top" alt="catelus">
  doar primul rind al textului este aliniat cu
  latura de jos a imaginii, restul textului
  fiind afisat dupa imagine
</P>
```

- atributele **height** şi **width** permit specificarea altor dimensiuni pentru imagine, decât cele reale ale acesteia. Evident, dacă dimensiunile nu sunt proporționale cu cele reale, imaginea va fi deformată. Totodată, dacă specificăm dimensiuni mai mari decât cele reale, imaginea se va vedea mai puțin clar. În realitate, imaginea este transferată de pe server la dimensiunile sale originale, redimensionarea având loc doar la nivelul calculatorului pe care este vizualizată pagina.

Iată un exemplu de folosire al celor două tag-uri, şi vizualizarea acestui exemplu în browser (**apl013.html**):


```
<P>
  Imaginea originala  are dimensiunile 200x150:
  <BR>
  <IMG src="dog.jpg">
  <BR>
  Iat-o redimensionata proportional la 100x75:
  <BR>
  <IMG src="dog.jpg" width="100" height="75">
  <BR>
  Iat-o si deformata:<BR>
  <IMG src="dog.jpg" width="50" height="100">
  sau
  <IMG src="dog.jpg" width="150" height="50">
  <BR>
</P>
```

- atributul **border** permite stabilirea grosimii unui chenar care va înconjura poza. Implicit, valoarea acestui atribut este ”0”, ceea ce înseamnă că imaginea nu este înconjurată de chenar (**apl014.html**):


```
<P>
  Imaginea este inconjurata
  de un chenar
  de dimenisiune 10<br>
  <IMG src="dog.jpg" border="10">
</P>
```

- atributele **hspace=”nr.pixeli”** respectiv **vspace=”nr.pixeli”** permit stabilirea distantei minime care separa imaginea de celelalte obiecte pe verticală, respectiv pe orizontală (**apl015.html**):


```
<P>
  Iata o aliniere a imaginii
  de tip "right", aliniere
  <IMG src="dog.jpg" align="right">
  in cadrul careia nu am modificat
  nici unul dintre cele doua
  atribute care controleaza
  spatierea dintre imagine
  si restul elementelor, pe
  orizontala respeciv pe
  verticala
  <BR clear="all"><BR>
  Iata acum o aliniere a imaginii
  tot de tip "right", aliniere
  <IMG src="dog.jpg" align="right"
   hspace="15" vspace="20">
  in cadrul careia am modificat
  ambele atribute care controleaza
  spatierea dintre imagine
  si restul elementelor, stabilind
  valorile de 20 pe verticala
  respeciv de 15 pe orizontala
  <BR clear="all">
</P>
```

2.3.7. Specificarea culorilor în HTML.

O serie de elemente din HTML permit utilizarea de atribute de culoare. Acestea pot fi specificate în două moduri:

• prin constanta HTML ce reprezintă numele culorii (în engleză, bineînțeles). Există 216 astfel de constante recunoscute de majoritatea browser-elor. Ne vom limita în a le enumera doar pe cele 16 care sunt considerate de bază, exemplificându-le pe fiecare:

Numele culorii	Aspectul sau
aqua	
black	
blue	
fuchsia	

Numele culorii	Aspectul sau
gray	
green	
lime	
maroon	

Numele culorii	Aspectul sau
navy	
olive	
purple	
red	

Numele culorii	Aspectul sau
silver	
teal	
white	
yellow	

O serie dintre culori (însă nu toate) au şi constante în variantele „dark” (închis) respectiv „light” (deschis). De exemplu: darkred sau lightblue.

• prin constanta de tip RGB (Red, Green, Blue):

Principiul de bază al redării electronice ale unei imagini în culori se bazează pe amestecarea în proporții diferite ale culorilor Roşu, Verde şi Albastru. În acest mod, se poate obține orice culoare se doreşte. În cazul culorilor pe care le poate reda un browser HTML, fiecare dintre aceste componente de culoare poate avea 256 de stări posibile: de la 0, care înseamnă că respectiva culoare lipseşte cu desăvârşire, până la 255, care înseamnă că respectiva culoare este folosită la intensitatea maximă. În acest fel, prin amestecuri diferite, putem obține 256^3, deci aproximativ 16 milioane de nuanțe diferite.

Componentele de culoare în HTML se specifică folosind numere hexazecimale. Astfel, fiecare dintre numerele dintre 0 şi 255 se codifică în hexazecimal printr-un număr între 00 şi FF. Constanta HTML pentru specificarea unei culori are forma generală **#RRGGBB**, în care **RR**, **GG** respectiv **BB** reprezintă câte un număr hexazecimal cuprins între 00 şi FF.

Iată câteva exemple de culori obținute folosind constante de forma celei de mai sus:

Numele culorii	Aspectul sau
#FF0000 (rosu de intensitate maxima)	
#7F0000 (rosu de intensitate jumatate)	
#00FF00 (verde de intensitate maxima)	
#007F00 (verde de intensitate jumatate)	

Numele culorii	Aspectul sau
#0000FF (albastru de intensitate maxima)	
#00007F (albastru de intensitate jumatate)	
#FFFF00 (rosu si verde ambele la maxim)	
#FF00FF (rosu si albastru ambele la maxim)	

Numele culorii	Aspectul sau
#00FFFF (rosu si albastru ambele la maxim)	
#7FFF00 (rosu la jumatate, verde la maxim)	
#FF7F00 (rosu la maxim, verde la jumatate)	
#7F00FF (rosu la jumatate, albastru la maxim)	

Numele culorii	Aspectul sau
#FF007F (rosu la maxim, albastru la jumatate)	
#007FFF (verde la jumatate, albastru la maxim)	
#00FF7F (verde la maxim, albastru la jumatate)	
#000000 (toate culorile sunt stinse)	
#FFFFFF (toate culorile sunt aprinse la maxim)	

2.3.8. Tabele.

Tabelele reprezintă un element foarte important al unei pagini web. În foarte multe cazuri, tabele cu chenare invizibile sunt folosite ca şi „schelet" al paginii, pentru a putea realiza alinieri complexe ale elementelor acesteia.

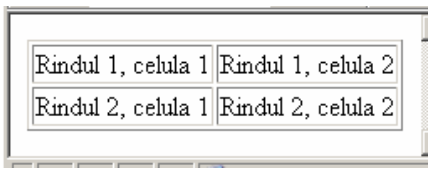
Tag-ul pentru descrierea unui tabel este **<TABLE>**...**</TABLE>**. În cadrul acestora trebuie descrise liniile (rândurile) tabelului, în cadrul fiecărui rând trebuind descrise celulele acestuia.

Descrierea unui rând se face între tag-urile **<TR>**...**</TR>**. La rândul lor, celulele din cadrul rândului se descriu între **<TD>**...**</TD>**. Atît tag-ul **</TR>** cât şi tag-ul **</TD>** pot fi omise.

Un prim atribut al tag-ului **<TABLE>** este **border="grosime_pixeli"**. Dacă acest atribut este omis, tabelul va avea un chenar invizibil. Dacă se specifică doar atributul, omițând grosimea, aceasta va fi luată, implicit, ca fiind 1.

Iată un exemplu de cod pentru definirea unui tabel (**apl016.html**):

```
<TABLE border>
  <TR>
     <TD>Rindul 1, celula 1
     <TD>Rindul 1, celula 2
  <TR>
     <TD>Rindul 2, celula 1
     <TD>Rindul 2, celula 2
</TABLE>
```

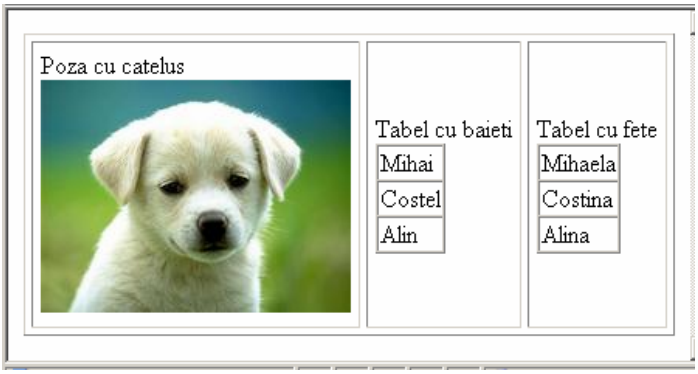


Atribute ale tag-ului <TABLE>

- **cellpadding="nr_pixeli"** permite stabilirea unui spațiu care va fi lăsat, în fiecare celulă a tabelului, între conținutul celulei şi marginile acesteia. Dacă nu se specifică acest atribut, el este în mod implicit considerat 0
- **cellspacing="nr_pixeli"** permite stabilirea spațiului care va fi lăsat între chenarele celulelor vecine în tabel (şi inclusiv între ele şi chenarul exterior al tabelului). Dacă nu se specifică acest atribut, el este în mod implicit considerat 2.

Conținutul unei celule poate fi cât se poate de general: de la text şi imagini până la alte tabele (se pot deci construi chiar şi tabele imbricate), ca în exemplul următor (**apl017.html**):

```
<TABLE border="1" cellspacing="4" cellpadding="5">
  <TR>
     <TD>
       Poza cu catelus<BR>
       <IMG src="dog.jpg">
     <TD>
       Tabel cu baieti
       <TABLE border cellspacing="0">
          <TR><TD>Mihai
          <TR><TD>Costel
          <TR><TD>Alin
       </TABLE>
     <TD>
       Tabel cu fete
       <TABLE border cellspacing="0">
          <TR><TD>Mihaela
          <TR><TD>Costina
          <TR><TD>Alina
       </TABLE>
 </TABLE>
```



- **width="lățime"** poate stabili cât de lat să fie tabelul. Lățimea poate fi dată în procente, caz în care se va calcula ca şi procent din lățimea ferestrei browser-ului (ex: **width="50%"**) sau în pixeli (ex: **width="500"**);
- **height="înălțime"** poate stabili cât de înalt să fie tabelul. Lățimea poate fi dată, la fel ca şi în cazul atributului width, în procente sau în pixeli;
- **align** determină alinierea tabelului în pagină. Poate la una dintre valorile **left**, **right** sau **center**. Dacă, pe lângă tabel, mai scriem şi text, acesta se va poziționa față de tabel în acelaşi mod în care se poziționează şi față de imagini;
- **bgcolor="culoare"** permite stabilirea culorii de fundal a tuturor celulelor tabelului;
- **bordercolor="culoare"** permite stabilirea culorii chenarului (deopotrivă cel interior cât şi cel exterior)

Atribute ale tag-ului <TR>

- **align** determină, pentru toate celulele de pe linie, modul alinierii conținutului pe orizontală, în interiorul celulelor. Poate la una dintre valorile **left**, **right**, **center** sau **justify**;
- **valign** determină, pentru toate celulele de pe linie, modul alinierii conținutului pe verticală, în interiorul celulelor. Poate la una dintre valorile **top**, **bottom** sau **middle**;
- **bgcolor** determină, pentru toate celulele de pe linia respectivă, culoarea de fundal.

Atribute ale tag-ului <TD>

- **width** şi **height** determină, pentru celula respectivă, lățimea şi înălțimea. Poate fi dată în procente sau pixeli. Dacă e specificată în procente, se va lua din lățimea, respectiv înălțimea

tabelului. Modificarea lățimii şi a înălțimii unei celule va avea efect şi asupra celorlalte celule, pentru ca tabelul să fie aliniat;

- **align** şi **valign** stabilesc, la fel ca şi în cazul lui **<TR>**, modul în care este aliniat conținutul în interiorul celulei, pe orizontală respectiv pe verticală, fiind prioritare față de alinierea la nivel de linie
- **colspan="n"** stabileşte întinderea celulei respective în dreapta cu **n** coloane (echivalentul operației Merge Cells din Word, în cazul în care unim celule adiacente pe orizontală);
- **rowspan="n"** stabileşte întinderea celulei respective în jos cu **n** linii (echivalentul operației Merge Cells din Word, în cazul în care unim celule adiacente pe verticală);
- **bgcolor** determină, pentru celula respectivă, culoarea de fundal. Evident, este prioritară față de acelaşi atribut la nivel de linie.

Exemplu (**apl018.html**):

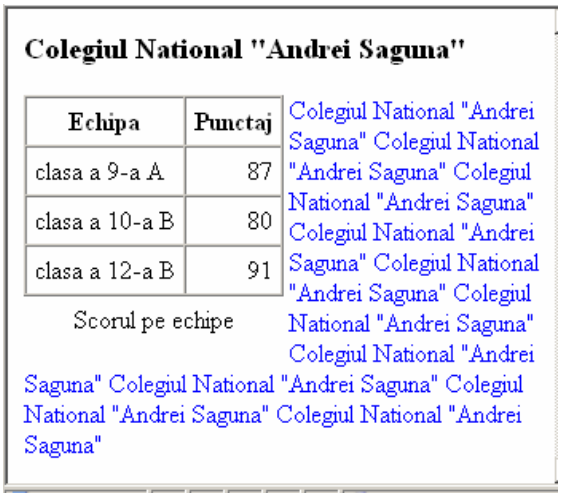
```
<TABLE border="1" cellspacing="0" cellpadding="5">
  <TR bgcolor="#c0c0ff">
     <TD>Ziua
     <TD>09h00 - 11h00
     <TD>11h00 - 13h00
     <TD>13h00 - 15h00
  <TR bgcolor="yellow" align="center">
      <TD align="left"><B>Luni</B>
      <TD colspan="2">Mecanica
      <TD bgcolor="#ffd0d0">Termodinamica
  <TR bgcolor="yellow" align="center">
      <TD align="left"><B>Marti</B>
      <TD>Electrostatica
      <TD>Optica
      <TD>Atomica
  <TR bgcolor="yellow" align="center">
      <TD align="left"><B>Miercuri</B>
      <TD rowspan="2" bgcolor="#ffd0d0">Termodinamica
      <TD>Optica
      <TD>Electrostatica
  <TR bgcolor="yellow" align="center">
      <TD align="left"><B>Joi</B>
      <TD>Mecanica
      <TD>Optica
</TABLE>
```

Ziua	09h00 - 11h00	11h00 - 13h00	13h00 - 15h00
Luni	Mecanica		Termodinamica
Marti	Electrostatica	Optica	Atomica
Miercuri	Termodinamica	Optica	Electrostatica
Joi		Mecanica	Optica

- Tag-ul **<TH>**...**</TH>** poate înlocui **<TD>**...**</TD>**. Atributele sunt aceleaşi. Singura diferență este că textele de după tag-ul **<TH>** sunt, în mod implicit, tipărite îngroşat (Bold) iar alinierea lor se face pe centru;
- Tag-ul **<CAPTION>**...**</CAPTION>** permite scrierea unui titlu pentru tabel. Acest tag trebuie să se găsească imediat după **</TABLE>**. Acest tag suportă atributul **align**. Acesta poate lua una dintre valorile: **left** (titlul va fi poziționat în stânga sus), **right** (poziționare dreapta sus), **top** (poziționare pe centru sus), **bottom** (poziționare pe centru jos);

Exemplu (**apl019.html**):

```
<H3>Colegiul National "Andrei Saguna"</H3>
<TABLE border="1" cellspacing="0"
 cellpadding="5" align="left">
  <CAPTION align="bottom">
  Scorul pe echipe</CAPTION>
  <TR><TH>Echipa<TH>Punctaj
  <TR><TD>clasa a 9-a A<TD align="right">87
  <TR><TD>clasa a 10-a B<TD align="right">80
  <TR><TD>clasa a 12-a B<TD align="right">91
</TABLE> <FONT color="blue">
Colegiul National "Andrei Saguna"
Colegiul National "Andrei Saguna"
Colegiul National "Andrei Saguna"
Colegiul National "Andrei Saguna"
Colegiul National "Andrei Saguna"
Colegiul National "Andrei Saguna"
Colegiul National "Andrei Saguna"
Colegiul National "Andrei Saguna"
Colegiul National "Andrei Saguna"
Colegiul National "Andrei Saguna"
</FONT>
```



Colegiul National "Andrei Saguna"

Echipa	Punctaj
clasa a 9-a A	87
clasa a 10-a B	80
clasa a 12-a B	91

Scorul pe echipe

Colegiul National "Andrei Saguna" Colegiul National "Andrei Saguna" Colegiul National "Andrei Saguna" Colegiul National "Andrei Saguna" Colegiul National "Andrei Saguna" Colegiul National "Andrei Saguna" Colegiul National "Andrei Saguna" Colegiul National "Andrei Saguna" Colegiul National "Andrei Saguna" Colegiul National "Andrei Saguna" Colegiul National "Andrei Saguna"

2.3.9. Legături (link-uri).

Aşa cum am văzut în partea introductivă a acestui capitol, noţiunea de www este strâns legată de documentele de tip hipertext.

Tot ceea ce am prezentat din limbajul HTML până în momentul de faţă, reprezintă doar partea descriptivă a acestuia, cu ajutorul căreia putem crea un conţinut static.

Link-urile reprezintă mecanismul prin care:

- putem face ca un vizitator al paginii, prin executarea unui click, să poată accesa o altă pagină, la care dorim să-i creăm posibilitatea unui acces rapid şi, dacă acesta doreşte, să poată reveni în pagina iniţială prin apăsarea butonului **Back** al browser-ului de Internet;
- putem face ca un vizitator al paginii noastre să primească un anumit fişier, de orice tip, care se găseşte pe site-ul nostru (download);
- putem face ca un vizitator al paginii noastre să poată asculta un mesaj sonor sau chiar să poată viziona un film;
- putem ca, printr-un click, să putem vizualiza o pagină (inclusiv cea curentă) doar dintr-un anumit loc, fără a folosi barele de derulare;
- putem ca, prin accesarea unui click, cel care vizitează pagina să ne poată trimite un e-mail.

Pentru toate acestea, vom folosi tag-ul **<A>**...**</A>**, numit şi Ancoră.

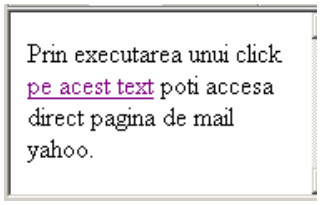
Ancore de legătură către alte pagini

Acestea permit ca un anumit element din document să conțină legătura către o altă pagină. Elementul care face legătura este de obicei un text sau o imagine. De regulă, elementul legat își schimbă aspectul față de cel clasic: textul va fi subliniat şi colorat altfel, iar imaginea va avea un chenar colorat. În momentul în care ducem cursorul deasupra elementului legat, acesta capătă forma unei mâini, indicându-ne astfel că este vorba de un link pe care îl putem utiliza. Printr-un simplu click, accesăm pagina către care este făcută legătura.

Acest tip de legătură se realizează practic folosind atributul **href**, ca în exemplul de mai jos (a se remarca modul în care, elementul legat, este inclus între tag-urile **<A href=...>** şi **</A>** :

apl020.html):

```
<p> Prin executarea unui click
<A href="http://mail.yahoo.com">
pe acest text </A>
poti accesa direct pagina de mail yahoo.
</p>
```



După cum se observă, atributul **href** primeşte adresa completă a paginii către care dorim să facem legătura. Dacă e vorba de un fişier local, din acelaşi director cu pagina din care facem legătura, la **href** este suficient să scriem numele şi extensia (de ex: **href="pagina.htm"**)

Ancore de legătură către fişiere (pentru download)

Se realizează în mod analog cu cele către alte pagini, la atributul href trebuind specificat adresa fişierului respectiv (dacă este în acelaşi director cu pagina din care facem legătura, e suficient să-i scriem numele şi extensia).

Ex: în cazul în care fişierul pentru download este local:

```
Pentru download arhiva executa un click
<A href="arhiva.zip">aici</A>
```

Ex: în cazul în care fişierul pentru download se află la altă adresă:

```
Pentru a descarca subiectele de bacalaureat la disciplina informatica, da un click
<A href="http://subiecte2008.edu.ro/bacalaureat/subiecte/E_informatica_c.zip">aici</A>
```

Legăturile către fişiere de tip sunet sau film se fac absolut în aceeaşi manieră. În funcție de extensia lor (.wav, .mid, .mp3, .avi) în momentul executării unui click asupra obiectului care face legătura către ele, acestea vor fi deschise automat către browser cu programul corespunzător.

Legături relative la conținutul documentului (paginii)

Sunt acele ancore care permit accesarea directă a unei pagini web într-un anumit loc, fără a mai folosi barele de derulare pentru a ajunge în acel loc.

Pentru aceasta, locul respectiv trebuie marcat. Acest lucru se face tot cu ajutorul tag-ului **<a>**, însă folosind atributul **id**, care va denumi locul respectiv printr-un identificator, ca în exemplul de mai jos (a se observa că între tag-ul de deschidere şi cel de închidere putem să nu punem nici un element):

```
<A id="capitolul2"></A>
```

Accesarea directă a acestui loc cu ajutorul unui link se poate face astfel:

a) Din interiorul aceleişi pagini: specificând la atributul **href** identificatorul respectiv (cel de la **id**) înainte de care se pune de caracterul **#**, ca în exemplul următor:

```
<A href="#capitolul2">Salt direct la capitolul 2</A>
```

b) Din altă pagină: specificând la atributul **href** adresa paginii accesate (a fişierului html) urmată de caracterul **#**, ca în exemplul următor:

```
<A href="http://www.myserver.ro/document.html#capitolul2">Deschide documentul extern,
direct la capitolul 2</A>
```

Ancoră de legătură pentru trimiterea unui e-mail

Acestea permit ca, atunci când o persoană ne vizitează site-ul, dacă doreşte, să ne poată trimită un e-mail făcând un simplu click pe legătura respectivă. Totuşi, pentru ca acest lucru să fie funcțional, cel care vizitează site-ul trebuie să aibă configurat pe calculatorul său un client de e-mail (cel mai frecvent este Outlook Express).

Iată un exemplu pentru o astfel de ancoră:

```
<ADDRESS>
Click <A href="mailto:somebody@someserver.com">aici</A> pentru a trimite un e-mail
</ADDRESS>
```

(tag-ul **<ADDRESS>**...**</ADDRESS>** nu face altceva decât să afişeze textul din cadrul său italic)

După cum se observă, pentru trimiterea unui e-mail, după atributul **href** trebuie specificat **mailto:** urmat de adresa de e-mail a destinatarului.

2.3.10. Elemente de structură (HTML, HEAD, BODY).

După cum am văzut în partea introductivă, orice document html este cuprins între tag-urile **<HTML>** şi **</HTML>**. El este alcătuit dintr-un unic antet (**HEAD**) şi un unic corp (**BODY**). Aceste 3 elemente au rolul de a defini structura documentului. Din acest motiv ele se mai numesc şi elemente de structură.

Tag-ul **BODY** poate conține următoarele atribute:

- **background="fişier_imagine"** permite specificarea unei imagini de fundal. Aceasta se va repeta atât pe orizontală cât şi pe verticală, până când se acoperă întreaga suprafață necesară corpului;
- **bgcolor="culoare"** permite specificarea unei culori de fond;
- **text="culoare"** permite specificarea culorii întregului text cuprins în pagină;
- **link="culoare"** permite specificarea culorii unui link nevizitat;
- **alink="culoare"** permite specificarea culorii unui link activ; un link este considerat activ în timpul vizitării şi imediat după aceasta;
- **vlink="culoare"** permite specificarea culorii unui link vizitat, care nu mai este activ.

Conținutul secțiunii <HEAD>

În cadrul acestei secțiuni putem întâlni diverse alte tag-uri. Despre tag-ul **<TITLE>** am discutat deja, el permițând scrierea unui titlu pentru pagină.

În afară de acestea, vom aminti încă alte 3 tag-uri:

- **<BASE>** permite stabilirea unei adrese de bază pentru resurse. Acest tag se foloseşte în special atunci când resursele (sau, în fine, o mare parte a acestora) se găsesc în alt director decât cel în care se află documentul curent. În acest fel, folosirea fişierelor din directorul specificat în BASE se poate face direct prin numele şi extensia lor. Specificarea se face prin:

```
<BASE href="adresa resurse">
```

- **<META>** este folosit pentru a furniza informații motoarelor de căutare. Unele dintre acestea vizitează doar antetul pentru a obține informații. Informațiile conținute de acest element nu sunt afişate de browser, însă este important să îl folosim pentru ca informațiile conținute în site-ul nostru să fie accesibile. Locul tag-ului **<META>** este în antet (**<HEAD>**).

Atributele tag-ului **<META>** sunt **name** şi **content**. Folosirea lor este ceva mai particulară, rezultând din exemplele următoare:

- pentru a specifica autorul unui document:

```
<META name="Author" content="Prenume NUME">
```

- pentru a specifica titlul unui document:

```
<META name="TITLE" content="Metode de programare">
```

- pentru a preciza cuvintele cheie după care să fie regăsit site-ul:

```
<META name="KEYWORDS" content="backtracking, divide et impera, greedy, programare dinamica">
```

- pentru a specifica limba în care este scris site-ul:

```
<META name="LANGUAGE" content="RO">
```

Există şi alte atribute ale elementului META, însă cele două deja prezentate sunt suficiente.

- **<STYLE>** este utilizat pentru introducerea stilurilor. Acestea permit stabilirea mai amănunțită a modului în care apar, implicit, diferitele elemente din document. Valorile se trec între **<STYLE>**...**</STYLE>**.

Exemplu:

```
<STYLE>
P {font-family:"Comic Sans MS"; font-size:14pt;}
</STYLE>
```

Prin specificarea lui P înainte de paranteza acoladă, stabilim ca modul implicit de afişare al paragrafelor (să ne reamintim că **<P>** este tag-ul pentru paragraf) să fie cel descris între parantezele acolade, deci, în cazul exemplului de față font-ul folosit să fie "Comic Sans MS", iar dimensiunea caracterelor să fie de 14.

- **<SCRIPT>** este utilizat pentru introducerea anumitor secvențe de program în cadrul paginilor web. Există mai multe limbaje (numite de scriptare) care permite scrierea acestor secvențe, cum ar fi JavaScript, VBscript. Specificarea limbajului în care este codat scriptul se face cu ajutorul atributului **language**, ca în exemplul de mai jos (**apl021.html**):

```
<SCRIPT language="JavaScript">
function calcul()
{ s=0;  for(i=1;i<=10;i++)
                        s+=i;
  alert("suma nr. de la 1 la 10 este: "+s);}
</SCRIPT>
...
<BODY onload="calcul();"> ... </BODY>
```

Acest exemplu defineşte în antetul paginii o funcție JavaScript capabilă să calculeze suma numerelor de la 1 la 10 într-o variabilă s şi-apoi să afişeze valoarea obținută prin intermediul unei ferestre de dialog. Funcția este apelată automat (atributul **onload**) la încărcarea paginii.

2.3.11. Pagini cu cadre (FRAMESET, FRAME, IFRAME).

Utilizarea frame-urilor permite ca, în cadrul aceleiaşi ferestre ale browser-ului să fie afişate simultan mai multe documente HTML (sau alte resurse).

Tag-ul **<FRAMESET>** are rolul de a împărți fereastra în mai multe cadre. În fişierul HTML, el înlocuieşte tag-ul **<BODY>**. Iată câteva atribute ale lui **FRAMESET**:

- **rows** – descrie liniile în care este împărțită secțiunea **FRAMESET** respectivă
- **cols** – descrie coloanele în care este împărțită secțiunea **FRAMESET** respectivă

descrierile pentru **rows**, respectiv **cols**, pot fi de forma:

```
<FRAMESET rows="30%, 50%, 20%">
  <FRAME ...>
  <FRAME ...>
  <FRAME ...>
</FRAMESET>
```

în acest exemplu, se definesc 3 cadre orizontale (linii) de înălțimi 30%, 50% respectiv 20% din înălțimea ferestrei.

Un alt exemplu, în care înălțimea cadrelor este definită proporțional:

```
<FRAMESET rows="3*, 1*, 2*">...
```

aici se definesc 3 cadre orizontale, proporționale cu 3, 1 şi 2 dintr-o înălțime de 3+2+1=6
(deci cadrele vor fi 3/6, 1/6 respectiv 2/6 din înălțimea ferestrei)

Un alt exemplu, în care înălțimea cadrelor este definită în pixeli:

```
<FRAMESET rows="100, 200, *">...
```

aici se definesc trei frame-uri: unul de înălțime de 100 de pixeli, altul de 200 de pixeli, al treilea fiind alocat cu spațiul rămas.

Tag-ul **<NOFRAMES>**...**</NOFRAMES>** reprezintă conținutul care va fi afişat unui vizitator, în cazul în care browser-ul său nu poate afişa cadre (în prezent, nu prea mai este cazul unor asemenea browsere).

Fiecare tag **<FRAMESET>**...**</FRAMESET>** trebuie ca, după definirea aspectului (cu ajutorul unuia dintre atributele **cols** sau **rows**) să conțină descrierile fiecăruia dintre cadrele definite. Acest lucru se face cu ajutorul tag-ului **<FRAME>** prin intermediul atributelor:

- **src** – adresa fişierului HTML sau a imaginii care se va încărca inițial în cadru;
- **marginheight** – marginile (în pixeli sau procent) față de partea de sus şi cea de jos;
- **marginwidth** – marginile (în pixeli sau procent) față de partea din stânga şi din dreapta;

• **frameborder** – poate lua valorile **1** (implicită), care înseamnă că acest cadru este separat de celelalte printr-un chenar, respectiv **0**, care înseamnă că acest cadru nu mai este separat de celelalte printr-un chenar.

• **scrolling** – tratează afişarea barei de scroll (derulare). Poate lua trei valori:

auto – valoarea implicită. Bara de scroll este prezentă numai dacă este cazul

yes – bara de scroll este totdeauna prezentă

no – bara de scroll nu va fi niciodată afişată

• **noresize** – dacă atributul acesta este prezent (el se foloseşte fără a i se atribui nici o valoare) atunci vizitatorului paginii nu i se va permite să redimensioneze cadrul. Prezența acestui atribut pentru un cadru nu permite nici redimensionarea cadrelor vecine.

• **name** – este un atribut foarte important. Prin intermediul său va putea fi identificat frame-ul respectiv. Acest lucru este foarte important, deoarece dintr-un cadru se poate comanda conținutul oricărui alt cadru.

Deschiderea unei pagini într-un anumit cadu, prin intermediul ancorelor, se poate specifica prin folosirea atributului **target="nume cadru"** imediat după folosirea atributului **href** în cadrul tag-ului **<A href="..." .. >**.

Iată un exemplu prin care definim o pagină cu două frame-uri verticale. Frame-ul din stânga va conține numele a 3 zile ale săptămânii (pe limba română). Accesarea fiecăruia va produce deschiderea în frame-ul drept a unei pagini care va conține traducerea numelui zilei respective în 4 limbi.

În total vom avea de construit 5 fişiere:

- un fişier pentru pagina inițială, cea care defineşte scheletul paginii cu frame-uri
- un fişier cu cele 3 zile ale săptămânii, pe fiecare dintre ele fiind pus câte un hyperlink care va deschide traducerea numelui său în celălalt frame
- 3 fişiere cu traducerilor numelor zilelor în 4 limbi străine.

Pagina inițială (**apl022pagframe.html**):

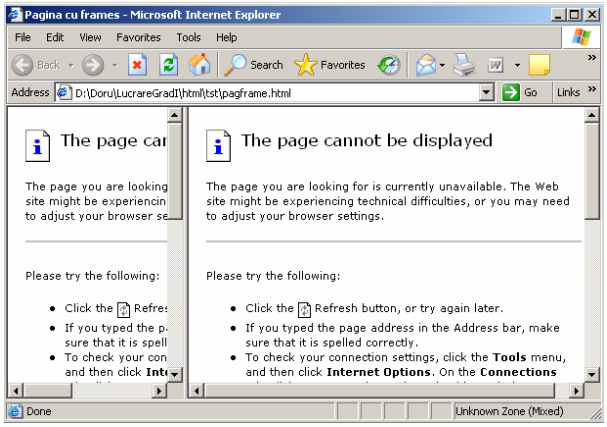
```
<HTML>
  <HEAD>
    <TITLE>Pagina cu frames</TITLE>
  </HEAD>
  <FRAMESET cols="30%,*">
    <FRAME name="stinga" src="apl022zile.html" noresize>
    <FRAME name="dreapta" src="apl022luni.html">
  </FRAMESET>
  <NOFRAMES>
  Browser-ul tau nu este capabil sa afiseze pagini cu frame-uri
  </NOFRAMES>
</HTML>
```

De remarcat faptul că această fişier HTML nu conţine decât scheletul cadrelor, ele urmând a fi populate iniţial, după cum remarcaţi din codul sursă, cu fişierele **apl022zile.html** pentru primul cadru (cel din stânga) respectiv cu fişierul **apl022luni.html** pentru cel de-al doilea cadru.

Observaţi modul în care au fost definite cadrele în cadrul tag-ului **FRAMESET**: **cols="30%,*"**. Acest lucru semnifică prezenţa a două cadre verticale (coloane) dintre care primul va ocupa 30% din lăţimea ferestrei, iar al doilea restul (lucru semnificat de caracterul ***** care închide şirul de definiţie al cadrelor).

De asemenea, atributul **noresize** în cadrul primului tag **FRAME** împiedică redimensionarea cadrelor de către utilizator. În cazul în care acest atribut nu ar fi fost prezent, utilizatorul, printr-un simplu „drag and drop" ar fi putut trage bara care separa cele două frame-uri, dându-i orice poziţie ar fi dorit.

Dacă încărcăm în browser-ul de internet documentul creat în acest stadiu, fără ca pe disc să existe vreunul dintre celelalte patru fişiere planificate, am obţine următorul rezultat:



Acesta era şi de aşteptat, de altfel, deoarece el demonstrează existenţa frame-urilor şi lipsa conţinutului.

Iată şi conţinutul celorlalte fişiere, pe care le vom pune în acelaşi director cu documentul de mai sus (în dreptul fiecăruia vom arăta şi vizualizarea sa în browser):

Fişierul **apl022zile.html**:

```
<HTML>
  <HEAD> <TITLE>Zilele</TITLE> </HEAD>
  <BODY>
  <br>
  <A href="apl022luni.html" target="dreapta">Luni</A><br><br>
  <A href="apl022marti.html" target="dreapta">Marti</A><br><br>
  <A href="apl022miercuri.html" target="dreapta">Miercuri</A><br><br>
  </BODY>
</HTML>
```

De remarcat modul în care am realizat link-urile asupra celor 3 cuvinte: folosind şi atributul **target** în cadrul ancorei (**<A ...>**) am specificat browser-ului ca paginile respective să fie deschise în cadrul frame-ului al cărui nume apare după **target**.

Fişierul **apl022luni.html**:

```
<HTML><BODY>
  <H2>Luni</H2>
  FR: Lundi<BR>
  IT: Lunedi<BR>
  GE: Montag<BR>
  EN: Monday<BR>
</BODY></HTML>
```

Luni

FR: Lundi
IT: Lunedi
GE: Montag
EN: Monday

Fişierul **apl022marti.html**:

```
<HTML><BODY>
  <H2>Marti</H2>
  FR: Mardi<BR>
  IT: Martedi<BR>
  GE: Dienstag<BR>
  EN: Tuesday<BR>
</BODY></HTML>
```

Marti

FR: Mardi
IT: Martedi
GE: Dienstag
EN: Tuesday

Fişierul **apl022miercuri.html**:

```
<HTML><BODY>
 <H2>Miercuri</H2>
  FR: Mercredi<BR>
  IT: Mercoledi<BR>
  GE: Mittwoch<BR>
  EN: Wednesday<BR>
</BODY></HTML>
```

Miercuri

FR: Mercredi
IT: Mercoledi
GE: Mittwoch
EN: Wednesday

Iată cum arată vizualizarea finală în browser (după crearea celor 4 fişiere de mai sus):

Evident, la efectuarea unui click asupra legăturilor (luni, marti, miercuri) din partea stângă, se va produce deschiderea paginii corespunzătoare în frame-ul drept.

Tag-ul **<IFRAME>** este un element care nu a fost prezent în primele versiuni ale limbajului HTML, ci a apărut ceva mai nou. Actualmente, folosirea sa este preferată de majoritatea celor care programează pagini web, deoarece se comportă ceva mai flexibil decât cadrele clasice. Totodată, motoarele de căutare nu indexează conținutul paginilor cu frame-uri obişnuite, pe când cele care conțin iframe-uri sunt indexate.

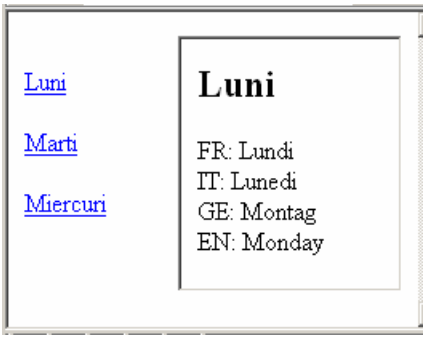
Prin intermediul său, este permisă crearea unui cadru în corpul unui documente HTML, cadrul care se comportă asemănător unei imagini.

Atributele lui **IFRAME** sunt:

- **name** – la fel ca şi la **FRAME**, acest atribut permite identificarea **IFRAME**-ului (pentru a putea comanda conținutul său din orice link)
- **height**, **width** înălțimea, respectiv lățimea. Pot fi specificate atât în pixeli, cât şi în procente, relativ la dimensiunile ferestrei browser-ului
- **frameborder** – poate lua valoarea **0** sau **1**, la fel ca la elementul **FRAME**
- **src** – adresa resursei care va fi încărcată inițial în **IFRAME**
- **marginwidth**, **marginheight**, **scrolling** – la fel ca şi la **FRAME**
- **align** – poate lua una dintre valorile **left**, **right**, **top**, **bottom**, **middle**, comportându-se întocmai ca şi în cazul imaginilor

Iată reluarea aceleiaşi idei structurale ca şi la aplicația de dinainte (cu frame-uri clasice) însă folosind un element de tipul **IFRAME**. Fişierele **apl022luni.html**, **apl022marti.html** respectiv **apl022miercuri.html** le păstrăm nemodificate. Practic, mai creăm doar un singur fişier HTML, cu conținutul următor, şi avem grijă să copiem în acelaşi director şi cele 3 fişiere de mai sus (**apl023.html**):

```
<HTML>
<HEAD><TITLE>Elementul IFRAME</TITLE></HEAD>
<BODY>
  <IFRAME name="cadru" width="140"
   height="160" align="right" src="apl022luni.html">
  </IFRAME>
  <BR>
  <A href="apl022luni.html" target="cadru">
   Luni</A><BR><BR>
  <A href="apl022marti.html" target="cadru">
   Marti</A><BR><BR>
  <A href="apl022miercuri.html" target="cadru">
   Miercuri</A><BR><BR>
</BODY>
</HTML>
```



2.3.12. Bare de separare (HR).

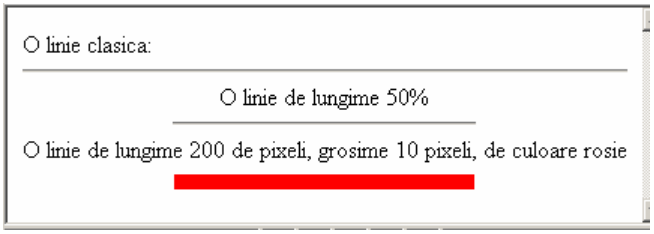
Bara de separare, al cărei tag este **<HR>** reprezintă un element decorativ. De obicei se foloseşte pentru a separa anumite secțiuni ale paginii web.

Atributele sale sunt:

- **width** – permite specificarea lungimii sale. Poate fi dată în pixeli sau în procente. Dacă acest atribut lipseşte, atunci lungimea sa va fi maximă (din marginea stângă şi până marginea dreaptă a ferestrei);
- **size** – permite specificarea înălțimii barei. Se specifică în pixeli;
- **color** – permite specificarea culorii sale.

Exemplu (**apl024.html**):

```
<BODY>
O linie clasica:
<HR>
<CENTER>O linie de lungime 50%</CENTER>
<HR width="50%">
<CENTER>O linie de lungime 200 de pixeli,
grosime 10 pixeli, de culoare rosie</CENTER>
<HR width="200" size="10" color="red">
</BODY>
```



2.3.13. Formulare.

Formularele sunt elemente ale limbajului HTML. Ele reprezintă o grupare de componente care permit trimiterea de date şi de comenzi către un server. Acesta trebuie să fie mai mult decât un clasic server HTTP, trebuind să aibă instalată şi o componentă capabilă de a răspunde comenzilor şi a prelucra datele. Cea mai populară astfel de componentă, foarte larg utilizată în ultimii 10 ani în programarea pe Internet este limbajul PHP, de care ne vom ocupa pe larg în capitolul al III-lea al acestei lucrări.

Pentru moment ne vom concentra asupra componentelor unui formular şi a aspectului acestora.

Un formular este descris prin intermediul tag-ului **<FORM>**...**</FORM>**. Atributele acestuia sunt:

- **action="adresa"** – acest atribut specifică adresa script-ului care se va ocupa de a răspunde la comenzi şi de a prelucra datele.
- **method** – acest atribut specifică modul în care datele vor fi transmise către server. Distinge, două valori pe care le poate lua acest atribut, şi anume:

  - **get** – datele sunt „la vedere" – acest lucru înseamnă că, în momentul trimiterii lor către server, ele vor apărea scrise în clar, în bara de adresă, într-un anumit format standard. De exemplu, dacă formularul trimite către pagina test.php o variabilă a care este egală cu 5, în bara de adresă a browser-ului ne va apărea **http://.../test.php?a=5**. Un dezavantaj major al acestei metode de trimitere a datelor este că volumul acestora este limitat (datorită şirului de caractere din adresă, care este limitat în cazul fiecărui browser).
  - **post** – datele nu mai apar în mod explicit utilizatorului. Totuşi, ele nu sunt criptate – practic, un program răufăcător le poate intercepta.

Pe lângă componentele specifice, un formular poate conține orice fel de alte elemente valide de HTML – tabele, imagini, text, bare de separare ...

În continuare vom prezenta câteva din componentele unui formulare, prin intermediul cărora utilizatorul poate introduce date şi trimite apoi aceste date către server. Un atribut foarte important al oricăruia dintre aceste componente este **name**, deoarece prin intermediul său, server-ul care va primi datele va şti despre care dintre controale este vorba.

Câmpuri text

Permit utilizatorului să introducă date într-un câmp de tip edit (pe o singură linie).
Aceste se specifică prin tag-ul

```
<INPUT type="text" ...>
```

Atributele sale sunt:

- **size** – specifică lățimea (în număr aprox. de caractere) câmpului text; Dacă acest parametru este omis, este implicit considerat ca fiind 20;
- **maxlength** – specifică numărul maxim de caractere ce pot fi scrise în câmpul text. Acest atribut poate primi o valoare mai mare decât cea scrisă la size, caz în care, textul va defila în control (stânga dreapta) în cazul în care scriem mai multe caractere decât câte încap în porțiunea vizibilă. Omiterea acestui atribut va permite introducerea unui număr foarte mare de caractere (limita diferă de la un browser la altul);
- **name** – numele câmpului text (prin care server-ul va identifica acest câmp, pentru a prelua datele din el);
- **value** – poate specifica o valoare care să fie inițial (la încărcarea paginii) deja scrisă în cadrul controlului. Dacă omitem acest atribut, câmpul text va fi gol.

Butoane de tip „submit"

Aceasta componenta se prezintă sub forma unui buton. Prin apăsarea sa are loc trimiterea tuturor datelor din formular către script-ul de pe server-ul care le va prelucra.
Un control de tip submit se specifică prin tag-ul:

```
<INPUT type="submit" ...>
```

Atributele sale sunt:

- **name** – numele de identificare a componentei. Putem omite acest atribut. El se foloseşte în cazul în care aceluiaşi formular dorim să-i ataşăm mai multe butoane de acest tip, iar apăsarea fiecăruia să producă o acțiune diferită;
- **value** – textul care va fi scris pe buton. De altfel, aceasta va fi şi valoarea pe care server-ul o va primi pentru acest control.

Câmpuri de tip password

Se comportă identic cu câmpurile de tip text. Singura deosebire este că, la scrierea de text în ele, acesta nu va fi vizibil, ci în locul caracterelor introduse se vor afişa asterisc-uri. Totodată, textul dintr-un astfel de control nu poate fi luat cu copy/paste.

Controalele de acest fel se specifică prin tag-ul:

<INPUT type="password" ...>

Atributele sunt identice cu cele de la **<INPUT type="text" ...>**

Câmpuri de tip butoane radio

Sunt controalele care permit ca, dintr-o serie de opţiuni posibile, utilizatorul să aleagă una singură. Controalele de acest fel se specifică prin tag-ul:

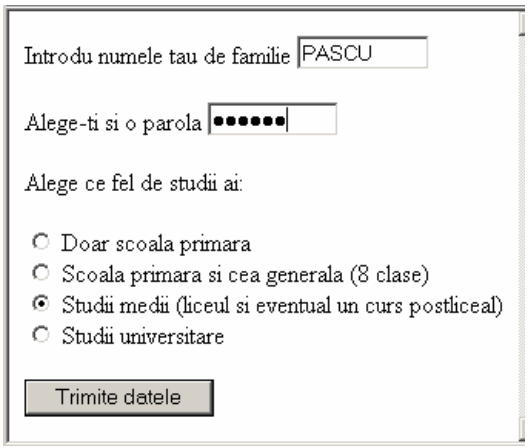
<INPUT type="radio" ...>

Atributele sale sunt:

- **name** – numele de identificare al componentei. Este obligatoriu ca toate butoanele care aparţin aceluiaşi grup (deci seria de opţiuni din care trebuie aleasă doar una singură posibilă) să poarte acelaşi nume de identificare;
- **value** – valoarea pe care o va întoarce butonul respectiv, dacă el a fost cel ales;
- **checked** – dacă acest atribut este prezent, butonul respectiv va fi ales în mod implicit, la încărcarea paginii. Este recomandabil ca, dintre toate butoanele care aparţin aceluiaşi grup, exact unul singur să conţină acest atribut.

Iată şi un exemplu care combină controalele prezentate până acum (**ap1025.html**):

```
<FORM action="nefunctional.php" method="post">
 Introdu numele tau de familie
 <INPUT type="text" size="10" maxlength="20"
                              name="numele">
 <BR><BR>Alege-ti si o parola
 <INPUT type="password" size="10" maxlength="20"
                              name="parola">
 <BR><BR>
 Alege ce fel de studii ai:<BR><BR>
 <INPUT type="radio" name="studii" value="scprim">
 Doar scoala primara<BR>
 <INPUT type="radio" name="studii" value="8clase">
 Scoala primara si cea generala (8 clase)<BR>
 <INPUT type="radio" name="studii" value="medii"
                                          checked>
 Studii medii (liceul si eventual un curs postliceal)
 <BR>
 <INPUT type="radio" name="studii" value="univ">
 Studii universitare<BR><BR>
 <INPUT type="submit" value="Trimite datele">
</FORM>
```



Evident, acest exemplu este nefuncţional, în sensul că datele din formular nu sunt prelucrate. Acest lucru va face obiectul capitolului următor, şi anume preluarea datelor dintr-un formular prin intermediul limbajului php.

Câmpuri de tip checkbox

Sunt controale care permit bifarea sau ştergerea bifării unei căsuțe. Din punct de vedere practic, ele permit utilizatorului să marcheze una, nici una, sau mai multe opțiuni.

Controalele de acest fel se specifică prin tag-ul:

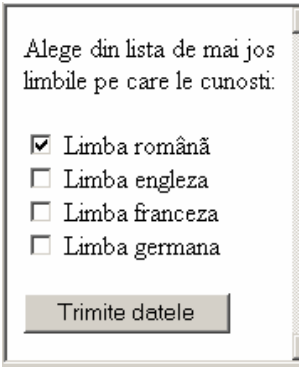
<INPUT type="checkbox" ...>

Atributele sale sunt:

- **name** – numele de identificare al componentei;
- **value** – valoarea pe care o va întoarce controlul respectiv;
- **checked** – dacă acest atribut este prezent, atunci controlul va fi bifat la încărcarea paginii.

Exemplu (**apl026.html**):

```
<FORM action="nefunctional.php" method="post">
 Alege din lista de mai jos limbile
 pe care le cunosti:<BR><BR>
 <INPUT type="checkbox" name="rom" value="1" checked>
 Limba română<BR>
 <INPUT type="checkbox" name="eng" value="2">
 Limba engleza<BR>
 <INPUT type="checkbox" name="fr" value="3">
 Limba franceza<BR>
 <INPUT type="checkbox" name="germ" value="4">
 Limba germana<BR><BR>
 <INPUT type="submit" value="Trimite datele">
</FORM>
```



Câmpuri ascunse (de tip hidden)

Aceste componente permit trimiterea de valori către server (o dată ce butonul submit a fost apăsat) fără ca acestea să fie vizibile în cadrul form-ului. Practic, aceste componente se specifică doar în cadrul codului HTML:

<INPUT type="hidden" name="nume" value="value">

Aşa cum se observă în tag-ul de mai sus, cu ajutorul atributului **name** specificăm numele controlului, iar cu ajutorul atributului **value** specificăm valoarea care va fi trimisă către server.

Controlul de tip TEXTAREA

Este o componentă care se utilizează pentru a introduce un text mai lung, care se poate întinde pe mai multe linii.

Tag-ul său este: **<TEXTAREA>...</TEXTAREA>**.

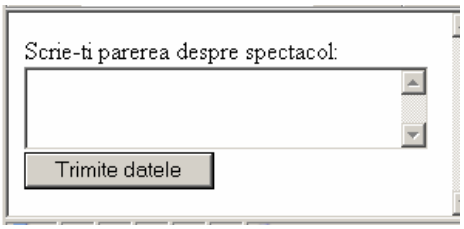
Atributele sale sunt:

- **name** – numele de identificare al componentei;
- **rows** – numărul de linii pe care se întinde componenta (implicit 2);
- **cols** – numărul de coloane pe care se întinde componenta (implicit 20);

Dacă dorim ca la încărcarea paginii să ne apară un text deja scris în cadrul controlului, acest text se va scrie între tag-ul de deschidere şi cel de închidere al lui **TEXTAREA**.

Exemplu (**apl027.html**):

```
<FORM action="nefunctional.php" method="post">
 Scrie-ti parerea despre spectacol:<BR>
 <TEXTAREA rows="3" cols="30" name="parerea"></TEXTAREA>
 <BR>
 <INPUT type="submit" value="Trimite datele">
</FORM>
```



Controlul de tip SELECT

Acest control este utilizat pentru afişarea unei liste din care utilizatorul poate să aleagă unul sau mai multe opțiuni.

Tag-ul prin care se utilizează această componentă este **<SELECT>**...**</SELECT>**.

Atributele sale sunt:

- **name** – numele de identificare al componentei;
- **multiple** – dacă acest atribut este prezent, utilizatorul poate alege mai multe opțiuni din listă, ținând apăsată tasta control sau shift în timp ce dă click pe acestea.
- **size** – numărul de opțiuni care sunt afişate. Implicit este 1, în cazul listelor care nu sunt de tip multiple. În acest caz, lista se prezintă sub forma unei liste de tip drop-down;

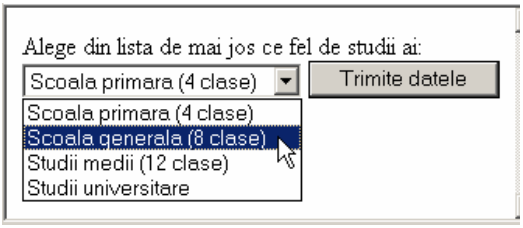
Fiecare opțiune din listă se specifică printr-un tag **<OPTION>**...**</OPTION>**. Acestea, la rândul lor, au următoarele atribute:

- **value** – reprezintă valoarea care va fi întoarsă de controlul SELECT în cazul în care va fi selectată opțiunea respectivă;
- **selected** – dacă acest atribut este prezent, opțiunea respectivă va fi selectată la încărcarea paginii

Textul efectiv al opțiunii se scrie între tag-ul de deschidere şi cel de închidere. Tag-ul de închidere este opțional, el putând fi omis.

Iată un exemplu de folosire al controlului de tip select (**apl028.html**):

```
<FORM action="nefunctional.php" method="post">
 Alege din lista de mai jos ce fel de studii ai:
 <BR>
 <SELECT name="studii">
 <OPTION value="prim">Scoala primara (4 clase)
 <OPTION value="gen">Scoala generala (8 clase)
 <OPTION value="lic" selected>Studii medii (12 clase)
 <OPTION value="univ">Studii universitare
 </SELECT>
 <INPUT type="submit" value="Trimite datele">
</FORM>
```



2.4. Extinderi ale limbajului HTML standard: HTML dinamic, script-uri.

Deşi HTML-ul clasic permite redactarea unor documente hypertext de un nivel foarte înalt şi elaborat, o dată cu evoluția limbajelor de programare vizuale, a început să devină mai puțin atractiv decât a fost la început.

Din acest motiv, a fost pus la punct ceea ce numim DTHML (Dynamic HTML) – care nu este un limbaj în sine, ci un termen prin care sunt desemnate tehnicile utilizate pentru a face paginile web cât mai dinamice şi cât mai interactive.

Pe lângă HTML-ul propriu-zis, noile unelte recunoscute de browser-ele din ultima generație sunt CSS (Cascading Style Sheets), JavaScript şi DOM (Document Object Model).

Scopul acestei lucrări nu este studiul amănunțit al acestora, de aceea le vom trece doar în revistă, folosind mici exemple comentate pentru fiecare dintre ele.

2.4.1. CSS (Cascading Style Sheets).

Noțiunea de **stil** este, pentru un document HTML, asemănătoare cu formatarea documentului, spre exemplu, pentru un document Word. Există o mulțime de atribute prin care se pot stabili font-urile, caracteristicile de aliniere a textului, forma elementului, culorile de fond şi ale literelor, marginile, poziția elementelor, etc.

Pentru a putea gestiona cât mai eficient stilurile, a fost pus la dispoziția programatorilor de pagini web un limbaj prin care se poate realiza acest lucru. Acest limbaj este cunoscut sub numele de **CSS** (actualmente, vorbim de versiunea **CSS2**).

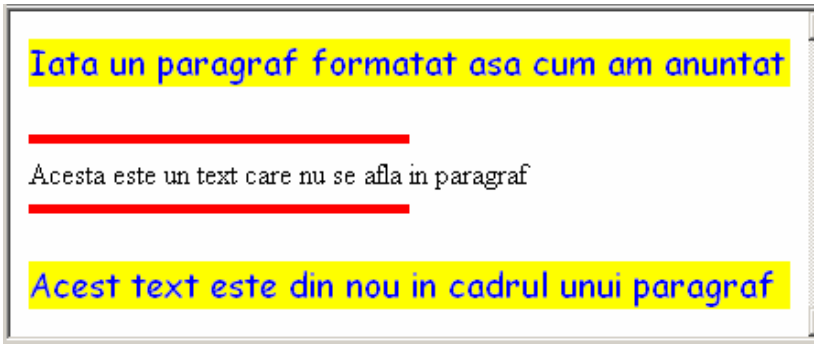
Legătura dintre HTML şi CSS se realizează prin intermediul tag-ului **<STYLE>**...**</STYLE>** care trebuie aşezat între **<HEAD>** şi **</HEAD>**.

În cadrul tag-ului **STYLE** vom stabili modul în care dorim să arate elementele paginii. Fiecare element al HTML-ului pe care l-am studiat este identificat, în cadrul CSS-ului de tag-ul care îl gestionează. Spre exemplu, identificatorul CSS pentru paragrafe este P, pentru table este TABLE, pentru imagini este IMG, ş.a.m.d.

Folosind aceşti identificatori în cadrul unui **<STYLE>**...**</STYLE>**, putem face ca toate elementele de acelaşi fel din cadrul unui document să arate la fel. Astfel suntem scutiți de a scrie o grămadă de cod care s-ar repeta în cazul fiecărui element de acelaşi fel.

De exemplu, dacă dorim ca, în cadrul paginii noastre, absolut toate paragrafele să fie scrise cu fontul Comic Sans MS, caractere de 14, culoare albastră, pe fond galben, în loc de a scrie aceşti parametri la fiecare paragraf din document, este suficient să definim următorul **STYLE** (**apl029.html**):

```
<HTML><HEAD>
<TITLE>Utilizare STYLE in HEAD</TITLE>
<STYLE>
P {
   background:yellow;
   color:blue;
   font-family:"Comic Sans MS";
   font-size:14pt;
  }
HR {
    text-align:left;
    width:50%;
    height:5px;
    color:red;
   }
</STYLE>
</HEAD>
<BODY>
<P>Iata un paragraf formatat asa cum am anuntat</P>
<HR>
Acesta este un text care nu se afla in paragraf
<HR>
<P>Acest text este din nou in cadrul unui paragraf</P>
</BODY></HTML>
```



De remarcat faptul că ambele paragrafe, şi de asemenea ambele linii orizontale **(HR)** din cadrul lui **BODY** nu conțin nici un fel de referință de formatare. Cu toate acestea, definirile lui **P** şi ale lui **HR** în cadrul lui **STYLE** au „predefinit" modul în care vor arăta toate paragrafele respectiv toate liniile orizontale ale documentului.

Sintaxa definiției este de felul următor: Se începe cu identificatorul elementului dorit a fi formatat (în cazul nostru **P** – tag-ul pentru paragraf, respectiv **HR**) între acolade trecându-se specificatorii de format (aceştia țin de limbajul CSS) doriți a fi modificați. În cazul de față, avem de-a face cu:

background = culoarea de fundal;

color = culoarea scrisului;

font-familiy = numele font-ului;

font-size = dimensiunea caracterelor;

text-align = alinierea în cadrul unui text;

width = lățimea;

height = înălțimea.

O altă formă de utilizarea a CSS-ului constă în definirea stilurilor cu ajutorul unor identificatori proprii, care se pot aplica ulterior unui anumit paragraf. În acest caz, în cadrul unui style putem defini proprii identificatori, precedându-i de caracterul **#**. Aplicarea ulterioară a lor asupra unui element, se face specificând un nou atribut, şi anume **id="identificator"** unde identificator este cel propriu, definit în cadrul lui **STYLE** (cel precedat de **#**)

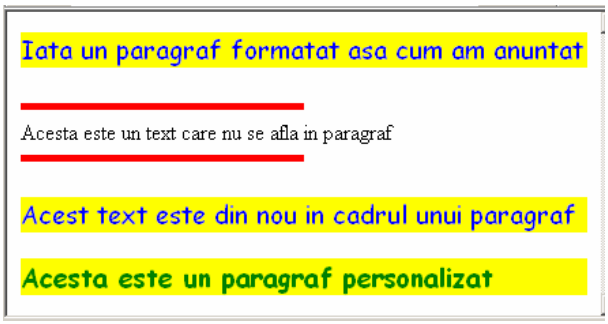
Ex: dacă inserăm în codul de mai sus secvența următoare (tot în cadrul lui **STYLE**, după definiția lui HR, adică cea scrisă cu **roşu închis**):

```
#alt_paragraf {
                color:green;
                font-weight:bold;
              }
```

iar înainte de **</BODY>** mai inserăm următorul paragraf:

```
<P id="alt_paragraf">Acesta este un paragraf personalizat</P>
```

vom obține următorul rezultat:



Remarcați faptul că au fost aplicați doar cei doi specificatori de format definiți în noul stil, şi anume culoarea fontului şi faptul că scrisul este bold. Celelalte caracteristici (font-ul şi culoarea galbenă de fundal) au rămas cele definite tot în **STYLE**, în cadrul lui **P**.

În loc de a defini stilurile în cadrul antetului (**HEAD**), aşa cum am arătat mai sus, ele pot fi scrise separat, într-un fişier text cu extensia .css, exact în aceeaşi manieră în care le-am fi scris între cele două tag-uri prezentate, **<STYLE>**...**</STYLE>**.

Includerea efectivă a acestui fişier în cadrul HTML-ului se face tot în secțiunea **<HEAD>**, prin intermediul următorului tag:

<LINK rel="stylesheet" type="text/css" href="fisier_stil.css">

Iată un exemplu:

1) Conținutul fişierului css, pe care l-am numit **apl30stil.css**:

```
TABLE {
        border-width:2px;
        border-style:ridge;
        border-collapse:collapse;
      }
TD {
     border-style:ridge;
     border-width:2px;
     padding:5px;
   }
TH {
     border-style:ridge;
     border-width:2px;
     background:#7fffff;
     padding:5px;
   }

TR {
     background:#ffff7f;
   }

#TR1 {
       background:#00ff00;
     }
```

După cum se observă, am definit în cadrul său formatele implicite pentru un tabel, rândurile şi celulele sale (**TABLE**, **TR**, **TD**, **TH**) precum şi un identificator propriu, **#TR1**.

Iată şi fişierul HTML care va folosi acest .css (**apl030.html**):

```
<HTML>
<HEAD>
 <TITLE>Utilizare css</TITLE>
 <LINK rel="stylesheet" type="text/css" href="apl30stil.css">
</HEAD>
<BODY>
<TABLE>
 <TR><TH>Numar<TH>Nume
 <TR><TD>1<TD>Ion
 <TR><TD>2<TD>Pop
 <TR><TD>3<TD>Top
 <TR id="TR1"><TD>4<TD>Ivan
</TABLE>
</BODY>
</HTML>
```

Iată, în continuare, în partea stângă, cum arată HTML-ul, datorită includerii acestui fişier CSS, iar în partea dreaptă cum ar fi arătat acelaşi HTML, fără a specifica nici un fel de format în CSS:

Numar	Nume
1	Ion
2	Pop
3	Top
4	Ivan

Numar	Nume
1	Ion
2	Pop
3	Top
4	Ivan

2.4.2. JavaScript.

JavaScript este un limbaj de programare orientat pe obiecte. În ciuda numelui şi a unor oarecare similarităţi în sintaxă, între JavaScript şi Java nu există nici o legătură.

JavaScript are o sintaxă apropiată de cea a C-ului; din acest motiv un programator care cunoaşte C poate cu uşurinţă să înveţe JavaScript.

Deşi acest limbaj are o plajă mai largă de extindere, cel mai des întâlnit este în scriptarea paginilor web. Programatorii web pot îngloba în paginile HTML script-uri pentru diverse activităţi, cum ar fi verificare datelor introduse de utilizatori, sau crearea de meniuri ori de alte efecte animate.

Browser-ele reţin în memorie o reprezentare a paginii web sub forma unui arbore de obiecte, punând aceste obiecte la dispoziţia JavaScript-ului, care le poate citi şi manipula. Acest arbore de obiecte, de care ne vom ocupa în paragraful următor, poartă numele de DOM (Document Object Model).

Pentru moment, vom da câteva exemple comentate de script-uri JavaScript, care nu folosesc DOM (pentru familiarizarea cu sintaxa), în cadrul unor documente HTML.

1) Calculul sumei cifrelor unui număr natural (**`ap1031.html`**):

```
<HTML><HEAD>
<TITLE>JavaScript</TITLE>
</HEAD>
<BODY><HR>
<!--Vom scrie secventa de cod direct in cadrul paginii.
A se remarca faptul ca, va aparea mai intii primul HR,
se va rula codul din tag-ul SCRIPT iar apoi va aparea
cel de-al doilea HR-->
<SCRIPT language="JavaScript">
v_text=prompt("Introdu un numar intreg cu maxim 9 cifre:","");
//functia prompt deschide o fereastra de dialog prin intermediul
//careia utilizatorul poate sa introduca date de tip string. Al doilea parametru
//(şirul vid "") reprezintă valoarea care se va găsi implicit scrisă în fereastra
//de dialog. Evident, dacă nu dorim nici o valoare implicită, se foloseşte şirul vid ("")
//String-ul obtinut l-am atribuit variabilei v_text
nr=parseInt(v_text);//am facut conversia de la variabila text
//la un numar intreg
s=0;//in s calculam suma cifrelor lui nr
do//procedam intocmai ca in limbajul C:
 {
   r=nr%10;
   s+=r;
   nr=parseInt(nr/10);//in JavaScript impartirea NU mai respecta
   //regulile din C, deoarece operatorul / face impartire cu
   //zecimale. Pentru a obtine citul intreg, am facut conversia la
   //intreg cu acelasi parseInt
 }while(nr);
alert("Suma cifrelor este "+String(s));
//functia alert(mesaj_de_tip_string) produce afisarea unei ferestre
//de dialog ce contine mesajul respectiv. A se remarca modul in care
//am concatenat mesajul cu valoarea variabilei s, convertita la string
//cu ajutorul functiei String.
</SCRIPT>
<HR>
</BODY></HTML>
```

A se remarca locul în care am pus script-ul (în cadrul paginii). În exemplele următoare nu vom mai da tot codul, ci doar secvenţa efectivă a script-ului.

2) Sortarea unui şir de numere (**apl032.html**):

```
<SCRIPT language="JavaScript">
v_text=prompt("Introdu un sir de numere pe care le separi prin spatii:","");
x=v_text.split(" ");//functia split, aplicata lui v_text (cu parametrul " ")
//va extrage substringurile din v_text care sunt separate de spatii si va crea
//un sir de string-uri, pe care i-l atribuie variabilei x. Acestea vor fi
//x[0], x[1], ... Numarul total de elemente din sirul x se obtine prin x.length
n=x.length;//obtinem acest numar in variabila n
for(i=0;i<n;i++)
  x[i]=parseInt(x[i]);//in acest fel transformam toate componentele sirului
  //x din string-uri in intregi. In C acest lucru nu ar fi fost posibil.
//acum sortam sirul obtinut:
for(i=0;i<n-1;i++)
  for(j=i+1;j<n;j++)
     if(x[i]>x[j])
        { aux=x[i];x[i]=x[j];x[j]=aux;}
//si afisam sirul final. Pentru asta, formam tot mesajul de afisat intr-un string
s="Iata sirul final, sortat:\n";
for(i=0;i<n;i++)
    s=s+String(x[i])+" ";
alert(s);
</SCRIPT>
```

3) Descompunerea unui număr în factori primi (**apl033.html**):

```
<SCRIPT language="JavaScript">
v_text=prompt("Introdu nr. intreg pe care doresti sa-l descompui in factori primi:","");
s="Iata descompunerea in factori primi:\n";
//pregatim string-ul in care vom afisa rezultatul final, pentru ca la acest string
//vom tot concatena noile date obtinute
n=parseInt(v_text);
f=2;
while(n!=1)
{
   p=0;
   while(n%f==0)
     {
          n=parseInt(n/f);
             p++;
       }
   if(p)
      s+="Factor="+String(f)+" putere="+String(p)+"\n";
      //fiecare nou factor si putere obtinute le concatenam la stringul
      //care va fi in final afisat
   f++;
}
alert(s);
</SCRIPT>
```

2.4.3. DOM (Document Object Model).

DOM reprezintă o interfață independentă față de orice limbaj de programare şi platformă, care permite programelor informatice şi script-urilor să aibă acces sau să actualizeze conținutul, structura sau stilurile unui document. Documentul poate fi apoi prelucrat, iar rezultatele acestor prelucrări pot fi reincorporate în document atunci când acesta este prezentat.

Înainte de standardizarea DOM-ului, fiecare navigator dispunea de propriul său model. Dacă limbajul de bază destinat manipulării documentelor web a fost repede standardizat în jurul lui JavaScript, nu acelaşi lucru se poate spune şi despre funcțiile specifice de utilizat şi maniera de a parcurge documentul. Cele două mari browser-e care s-au impus (Netscape Navigator şi Internet Explorer) denumeau în moduri diferite o serie de componente. În practică, acest lucru obliga programatorul să scrie cel puțin două versiuni ale fiecărui script, dacă dorea ca site-ul său să fie accesibil pentru cât mai multă lume.

Prima încercare de standardizare (DOM 1) a avut loc de-abia în 1998. Ultimul nivel de standardizare (DOM 3) a avut loc în 2004.

Din punct de vedere dinamizării paginilor web, limbajul JavaScript reprezintă doar o unealtă de lucru (ați remarcat în paragraful anterior similitudinea dintre acesta şi limbajul C). Pentru ca limbajul JavaScript să acționeze asupra conținutului paginii, ei bine, acest lucru îl face tocmai prin intermediul DOM.

Prin intermediul DOM putem accesa orice obiect al paginii web şi îl putem face să se comporte exact în felul în care dorim noi.

Ca şi în cazul programării vizuale, DOM permite interceptarea anumitor evenimente (poziția mouse-ului, click-uri, etc.) şi tratarea lor diferențiată.

Vom da în continuare, câteva exemple comentate, în care operăm cu JavaScript + DOM.

1) Schimbarea unei imagini atunci când trecem cu cursorul peste ea: Pentru aceasta aplicație avem nevoie de doua fişiere imagine, care să fie, de preferabil, identice ca şi dimensiuni, şi obținute una dintr-alta printr-un procedeu de genul negativ sau trecere la alb-negru.
Iată fişierele imagine pe care am testat script-ul următor:

dog.jpg:  dogneg.jpg  (negativul)

apl034.html:

```
<HTML><HEAD>
<TITLE>Schimbare de imagini</TITLE>
<SCRIPT language="JavaScript">

function schimba_negativ()
{//in momentul apelului, aceasta functie obtine
//in variabila dp o referinta catre obiectul img
//din pagina principala, gratie id-ului sau, si
//anume 'poza'
  dp=document.getElementById("poza");
//dupa care imaginea sursa a sa este schimbata,
//folosind imaginea din fisierul dogneg.jpg (cea negativa)
  dp.src="dogneg.jpg";
}

function revine_normal()
{//exact la fel ca functia precedenta, insa
//se foloseste alta imagine, si anume cea initiala,
//dog.jpg
  dp=document.getElementById("poza");
  dp.src="dog.jpg";
}

</SCRIPT>
</HEAD>
<BODY>
<!--elementului img ii stabilim id-ul 'poza'
pentru a-l putea folosi apoi in cadrul script-ului
de asemenea, programam ca elementul img sa reactioneze
la cele doua evenimente:
- onmouseover (cind mouse-ul intra deasupra imaginii)
se va apela functzia care schimba imaginea originala cu
cea pe negativ
- onmouseout (cind mouse-ul iese de deasupra imaginii)
se va apela functzia care pune la loc imaginea originala-->
   <IMG src="dog.jpg" id="poza"
    onmouseover="schimba_negativ();"
       onmouseout="revine_normal();">
</BODY>
</HTML>
```

2) Schimbarea culorii de fundal a unui tabel, culoare pe care o compunem cu ajutorul a 3 valori pentru componentele R, G, B (între 0 şi 255) pe care le scriem în nişte zone text. Cele 3 valori le vom valida (**apl035.html**):

```
<HTML><HEAD><TITLE>Exemplu de JavaScript</TITLE>
<SCRIPT laguage="JavaScript">
function toHex(numar)
{//aceasta functie converteste numarul parametru din zecimal
 //in hexazecimal. Ne bazam ca este cuprins intre 0 si 255
 c1=parseInt(numar/16);
 c2=numar%16;
 //in c1 si c2 am obtinut cele 2 cifre hexazecimale
 if(c1>9) c1=String.fromCharCode(65+c1-10);
 //daca c1 este mai mare decit 9, o inlocuim cu litera corespunzatoare (A=10, B=11, ...,
 // F=15) adunind la codul ASCII al lui A (65) diferenta corespunzatoare. Conversia, in
 //JavaScript, de la codul ASCII la caracter se face prin
 //String.fromCharCode(codul ascii al caracterului)
 if(c2>9) c2=String.fromCharCode(65+c2-10);
 return String(c1)+String(c2);
}

function rgb(red,green,blue)
{//aceasta functie genereaza constanta HTML de tip culoare plecind de la valorile lui
//red, green, blue, numere cuprinse intre 0 shi 255.Ea se foloseste de functia de mai sus
  return "#"+toHex(red)+toHex(green)+toHex(blue);
}
```

```
function coloreaza()
{//aceasta functie se apeleaza la apsarea butonului definit in cadrul lui BODY. In primul
//rind testam daca valorile sunt intregi. Observati ca am folosit identificatorii dati la
//atributul id pentru a extrage valorile din cimpurile text. In primul rind, pentru a
//accesa obiectele de tip <input type="text" id="..."> va trebui sa ne folosim de o
//functie speciala DOM, si anume document.getElementById. Aceasta functie ne intoarce
//o variabila prin intermediul careia putem accesa in continuare obiectul cu ID-ul
//respectiv.
//In cazul nostru, obtinem variabilele r, g si b pe baza cimpurilor text cu ID-urile
//rr, gg si bb definite in cadrul sectiunii <body>, mai jos.
//Pe baza variabilelor de tip obiect r, g si b, cimpul "value" ne va intoarce taman
//valoarea scrisa in acestea
   r=document.getElementById("rr");
   g=document.getElementById("gg");
   b=document.getElementById("bb");
   nr=parseInt(r.value);ng=parseInt(g.value); nb=parseInt(b.value);
   if(nr!=r.value)//daca valoarea convertita la intreg nu coincide
   //cu cea neconvertita, inseamna ca nu este inteaga, deci dam un mesaj
          {alert("Valoarea lui r nu este corecta!");
               return;}//si iesim fortat (ca in C) cu return
//procedam analog pentru celelalte doua
   if(ng!=g.value)
          {alert("Valoarea lui g nu este corecta!");return;}
   if(nb!=b.value)
          {alert("Valoarea lui b nu este corecta!");return;}
//acum verificam sa fie cuprinse intre 0 si 255
if(nr<0||nr>255)
     { alert('Valoarea lui r nu este cuprinsa intre 0 si 255');return;}
if(ng<0||ng>255)
     { alert('Valoarea lui g nu este cuprinsa intre 0 si 255');return;}
if(nb<0||nb>255)
     { alert('Valoarea lui b nu este cuprinsa intre 0 si 255');return;}
//in fine, daca am trecut de aceste filtre, valorile lui r, g si b sunt corecte
// si putem, in fine, stabili culoarea de fundal a celuilalt tabel (caruia i-am dat
//id-ul tabel) la cea pe care o obtinem din combinatia r, g, b introdusa.
tbl=document.getElementById("tabel");
//La fel ca mai sus, getElementById ne intoarce o variabila prin intermediul
//careia putem accesa obiectul cu id-ul respectiv
tbl.style.backgroundColor=rgb(nr,ng,nb);
//apoi, prin intermediul variabilei intoarse, si anume tbl,
//stabilim culoarea de fundal a tabelului. Pentru intoarcerea culorii
//in formatul recunoscut de HTML, adica #RRGGBB apelam functia
//rgb scrisa tot de noi, mai sus
}
</SCRIPT>

<BODY>
Introdu componentele de culoare (numere intre 0 si 255):<BR><BR>
<!--In tabelul de mai jos am folosit 3 input type="text" fara a ne afla intr-un form.
Este posibil si asa ceva, deoarece continutul lor il vom prelua cu ajutorul unui script
JavaScript. In cadrul acelui script ne vom folosi de aceste controale prin intermediul
atributului id pe care l-am stabilit, deci rr, gg si bb-->
<TABLE border="1" cellspacing="0" cellpadding="5">
  <TR><TD>Rosu<TD>
  <INPUT type="text" id="rr" maxlength="3" size="3">
  <TR><TD>Verde<TD>
  <INPUT type="text" id="gg" maxlength="3" size="3">
  <TR><TD>Albastru<TD>
  <INPUT type="text" id="bb" maxlength="3" size="3">
 </TABLE><BR>
<!--Mai jos am folosit o componenta de tip BUTTON. Aceastei componente i-am folosit
atributul onclick. Acestui atribut ii specificam practic ce functie JavaScript trebuie
 apelata in momentul in care se da click pe buton -->
 <BUTTON onclick="coloreaza();">Testeaza</BUTTON>
 <BR><BR>
 <!--acestui tabel i-am utilizat atributul id, pentru a-l putea mai usor accesa
 prin modelul DOM in cadrul codului JavaScript-->
 <TABLE width="50%" height="50%" id="tabel" border="1">
 <TR><TD align="center" valign="middle">TEST AREA
 </TABLE>
 </BODY></HTML>
```

3) În aplicația următoare, prin intermediul unui control de tip **`input type="text"`** vom cere utilizatorului să introducă un număr **`x`** între 4 si 20. Pe baza acestui număr (pe care-l validăm) vom genera, într-un element de tipul iframe, un tabel cu x linii şi x coloane, în care punem numerele de la 1 la x^2 şi ale cărui celule le colorăm alternativ, la fel ca pe o tablă de şah.

Pe lângă codul sursă am pus şi o captură a ferestrei, in urma rulării cu n=7 (**`ap1036.html`**):

```
<HTML><HEAD><SCRIPT language="JavaScript">
function genereaza()
{
   n=document.getElementById("nn");
   nr=parseInt(n.value);
   if(nr!=n.value)//verificam daca in n este un numar intreg
     {alert('Numarul introdus nu este intreg');return;}
   else if(nr<4||nr>20)//verificam si daca este intre 4 si 20
         {alert('Numarul trebuie sa fie intre 4 si 20');
               return;}
   d=document.getElementById("ifr").contentWindow.document;
   //obtinem in variabila d referinta DOM catre documentul din iframe
   d.open();//deschidem acest document pentru rescriere
   d.write('<TABLE border="1" cellspacing="0" cellpadding="5">');
   k=0;//si generam, prin script, in cadrul sau, codul HTML
   //care creeaza tabelul anuntat
   for(i=1;i<=nr;i++)
      {
            d.write('<TR>');
               for(j=1;j<=nr;j++)
                  {
                     d.write('<TD align="center" ');
                     if((i+j)%2)//in functie de paritatea lui i+j
                     //coloram intr-un fel sau intr-altlul fundalul celulei
                        d.write('bgcolor="#ffff7f">');
                     else
                        d.write('bgcolor="#7fffff">');
                     ++k;
                     d.write(String(k));
                    }
         }
   d.write('</TABLE>');
   d.close();
}
</SCRIPT></HEAD><BODY>
Introdu un numar natural cuprins intre 4 si 20:
<!--prin intermediul input type="text" scriem o valoare
care este apoi preluata de JavaScript. Acesta are id-ul
"nn" -->
<INPUT type="text" id="nn" size="2" maxlength="2">
<BR><BR>
<!--prin intermediul metodei "onclick()" a butonului
apelam functia care generaza codul HTML al tabelului
in documentul din iframe-->
<BUTTON onclick="genereaza();">Genereaza tabelul</BUTTON>
<BR><BR>
<IFRAME id="ifr" width="70%" height="500">
</IFRAME><HR></BODY></HTML>
```

De reținut din această ultimă parte, că script-urile, deşi reprezintă o automatizare şi dinamizare foarte importantă a unei pagini web, nu sunt rulate pe server-ul HTML (de altfel, până în momentul de față am lucrat cu toate fişiere în mod local, ele fiind deschise automat de către browser-ul de internet) ci ele sunt rulate de către browser pe calculatorul clientului care accesează pagina ce le conține.

3. LIMBAJUL PHP – FACILITĂȚI ALE ACESTUIA

3.1. Introducere – scurt istoric al apariției limbajului PHP; mod de funcționare.

PHP este un limbaj de programare destinat în primul rând Internetului, aducând dinamică unei pagini web. Este unul dintre cele mai importante limbaje de programare web *open-source* (este gratuit şi, în plus, utilizatorii pot acționa liber asupra procesului de dezvoltare) şi *server-side* (codul sursă nu se rulează pe calculatorul celui care vizualizează pagina, ci pe serverul web).

Numele său este un acronim recursiv: „***P***hp este un ***H***ypertext ***P***rocessor". Limbajul a fost început în 1994 ca o extensie a limbajului server-side ***Perl***, şi apoi ca o serie de CGI-uri compilate, de către Rasmus Lerdorf, pentru a genera un curriculum vitae şi pentru a urmări numărul de vizitatori ai unui site. A evoluat apoi în PHP/FI 2.0, dar proiectul open-source a început să ia amploare după ce Zeev Suraski şi Andi Gutmans au lansat o nouă versiune a interpretorului PHP în vara anului 1998, această versiune primind numele de PHP 3.0. Tot ei au schimbat numele în acronimul recursiv amintit mai sus, până atunci PHP-ul fiind cunoscut ca „***P***ersonal ***H***ome ***P***age tools". În prezent este utilizată versiunea 6 a acestui limbaj.

Prin CGI (***C***ommon ***G***ateway ***I***nterface) se înțelege o interfață a unui server de web, care extinde funcționalitățile acestuia. CGI nu se referă la un anumit limbaj de programare, ci defineşte un modul standardizat, prezent în cadrul unui server HTTP. Prin intermediul acestui modul se stabilesc regulile prin care server-ul va pasa datele primite de la un utilizator către o aplicație scrisă într-un anumit limbaj de programare, pentru ca apoi să întoarcă rezultatele acestei aplicații înapoi la utilizator.

Limbajul PHP, în marea majoritate a cazurilor, se foloseşte sub formă de secvențe de cod inserate în cadrul unui document HTML. Din acest motiv, vom prefera termenul de „script PHP" celui de program PHP. Structura unui script PHP este foarte asemănătoare cu cea a unui cod scris în limbajul C, mai ales în sensul în care structurile de programare au aceeaşi sintaxă şi aceeaşi funcționalitate.

Rolurile de bază ale unui script PHP constau în aceea că scipt-ul poate prelua date trimise de către o pagină web de la un client (în general, datele pot fi trimise de către o pagină web prin intermediul formularelor) şi de a executa o secvență de program în urma căreia va rezulta un cod HTML, cod pe care clientul îl va primi sub forma unei pagini web. Clientul nu va avea acces la codul efectiv al script-ului, ci, prin faptul că acesta se află pe server şi se rulează tot pe acesta, va primi direct HTML-ul generat de script.

3.2. Cerințe tehnice pentru rularea limbajului PHP pe un sistem Windows. Detalii asupra instalării.

După cum am văzut în capitolul precedent, PHP nu este un limbaj de programare de sine-stătător (cum ar fi C++, spre exemplu) ci se foloseşte în simbioză în primul rând cu HTML, şi deci pentru a rula, are nevoie neapărată de un server de web (http server).

Practic, pentru a face ca pe calculatorul nostru să poată rula fişiere php, avem nevoie să instalăm, pe lângă limbajul PHP, şi un server de web, şi de a face legătura dintre cele două.

Acest lucru se poate face separat, însă sunt necesare o serie de setări foarte minuțioase şi greoaie.

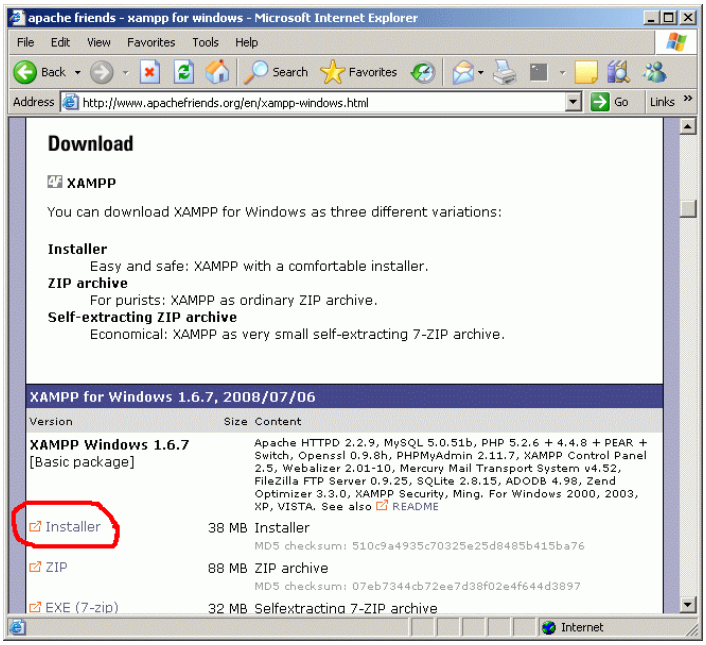
Din acest motiv, pentru testarea aplicațiilor pe care le vom studia, vom folosi un pachet gratuit, disponibil pe Internet, numit XAMPP. Acest pachet, foarte simplu de instalat, conține mai multe aplicații. Cele care ne vor interesa pe noi sunt serverul Apache (pentru http), un server de MySQL, şi suport pentru limbajul PHP. Cerințele tehnice pentru rularea în bune condiții sunt minimale: practic, pe orice calculator care este capabil să ruleze Windows 2000, XP sau Vista, pachetul XAMPP va rula fără probleme.

Deşi instalarea sa poate fi făcută în mai multe feluri, cel mai la îndemână este să folosim o versiune de tip „Installer" a sa.

Adresa de unde poate fi descărcat pachetul gratuit este:

`http://www.apachefriends.org/en/xampp-windows.html`

1. Downloadați executabilul installer-ului: derulați pagina până când dați de secțiunea „Download", accesând primul link „Installer", ca în figura de mai jos:



La momentul scrierii lucrării de față, ultima versiune publică a XAMPP-ului este 1.6.7.
Fişierul executabil al installer-ului acestei versiuni este **xampp-win32-1.6.7-installer.exe**

2. Lansați în execuție installer-ul. Vom alege instalarea în limba engleză. Lăsăm nemodificat directorul propus pentru instalare (**c:\xampp**). În fereastra următoare vom bifa toate cele 3 căsuțe care ne propun instalarea server-elor Apache, Mysql respectiv Filezilla (acesta din urmă fiind de fapt un server de FTP) ca şi servicii.

3. La sfârşitul instalării, server-ul de web este deja funcțional, având inclus atât suport php cât şi baza de date MySQL. Prin intermediului panoului de control XAMPP putem vedea care este starea curentă a server-elor instalate, şi le putem de asemenea gestiona.

4. Pentru a testa efectiv funcționalitatea server-ului web, deschidem un browser de internet, scriind la adresă: **http://localhost**.

Dacă instalarea a fost făcută cu succes, ne apare o pagină din care suntem invitați să alegem limba de operare, după care suntem duşi în pagina „HOME" a instalării făcute.

Directorul rădăcină al documentelor web este **c:\xampp\htdocs**.

În vederea testării aplicațiilor PHP pe care le vom studia în continuare, vom crea în acest director un alt subdirector **phpapps**.

După ce l-am creat, putem testa existența sa deschizând browser-ul de internet în care scriem adresa: **http://localhost/phpapps**

În browser trebuie să ne apară un director gol.

În mod implicit, toate fişierele şi subdirectoarele pe care le punem în acesta vor fi vizibile prin intermediul server-ului de http. Numele **index.htm**, **index.html** respectiv **index.php** sunt rezervate: dacă denumim vreun fişier în acest mod, la intrarea în directorul care îl conține, în loc de a ne arăta ceea ce se găseşte în acest director, serverul web ne va arăta direct pagina conținută de fişierul respectiv.

3.3. Testarea instalării. Structura unui fişier PHP.

Spre deosebire de fişierele **.html** care o dată create pe discul local cu un editor de texte pot fi deschise imediat tot local, direct în browser-ul de Internet, pentru a rula codul PHP este absolut necesar ca fişierele să fie puse în directorul în care rezidă documentele serverului web, iar

vizualizarea lor să fie făcută prin intermediul acestuia. În mod implicit, un fişier care conţine un script PHP trebuie să fie salvat cu extensia **.php**.

Să creăm primul nostru script PHP. Vom descrie operaţiile necesare acestui lucru, bazându-ne pe instalarea pachetului XAMPP descrisă în capitolul anterior. Astfel, toate scripturile pe care le vom crea le vom pune în directorul **c:\xampp\htdocs\phpapps** .

Cu ajutorul unui editor de texte (de exemplu Notepad, Notepad++) creaţi următorul fişier, pe care îl salvaţi în directorul de mai sus sub numele **ap1037.php** (codul de mai jos este preluat aşa cum apare vizualizat în editorul Notepad++. Numele fişierelor aplicaţiilor continuă numărătoarea din capitolul precedent):

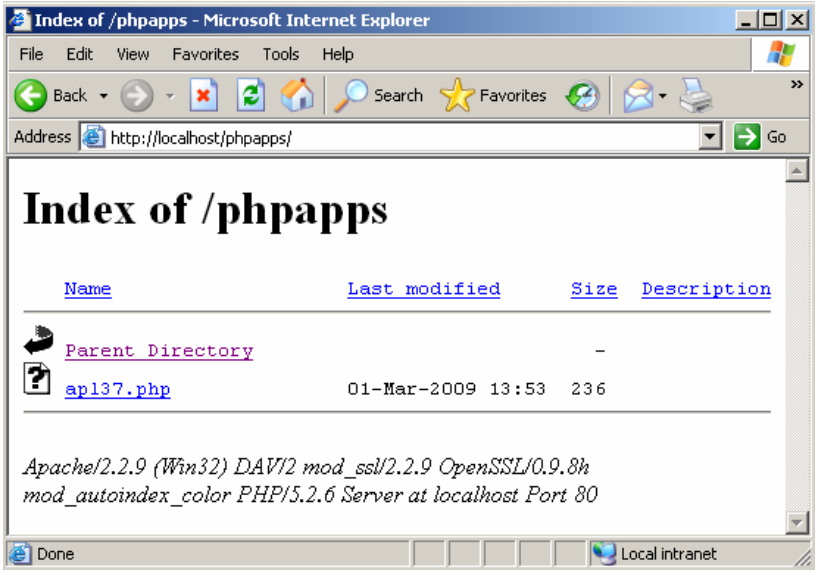
```
1  <HTML><HEAD>
2  <TITLE>Primul HTML continind script PHP</TITLE>
3  </HEAD>
4  <BODY>
5  <H2>Acesta este un titlu H2 scris normal, in afara scriptului</H2>
6  <?php
7      echo "Acest text este scris de catre script-ul PHP";
8  ?>
9  </BODY>
10 </HTML>
```

Observaţi structura absolut identică celei a unui fişier HTML. Noutatea este adusă de scriptul PHP, care este inserat între tag-urile colorate în roşu: „**<?php**” şi „**?>**”.

Instrucţiunea „**echo**” cuprinsă între acestea este o instrucţiune specifică limbajului PHP, ea având rolul de a scrie în pagina web textul ce urmează după, cel cuprins între ghilimele.

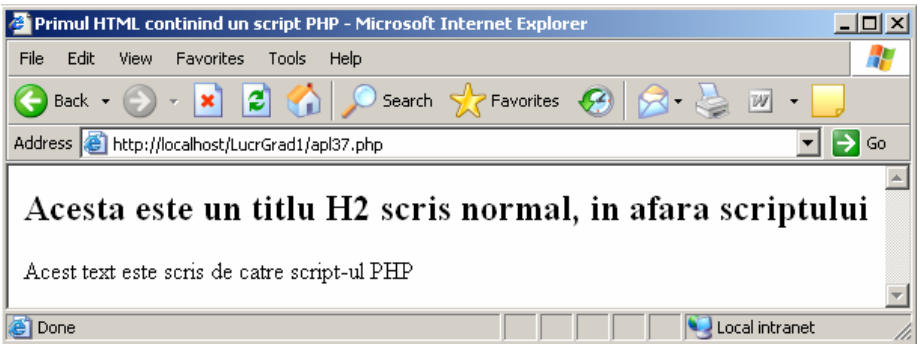
Să vizualizăm acum pagina rezultată în urma acestui fişier. Pentru aceasta, deschideţi browser-ul de Internet, scriind următoarea adresă: **http://localhost/phpapps**.

Dacă XAMPP a fost corect instalat, veţi obţine următoarea pagină:



În pagină vă este arătat conținutul directorului în care am creat fișierul **apl037.php**, așa cum este vizualizat prin intermediul serverului de web.

În această pagină faceți un click pe fișierul **apl037.php**, pentru a-l vizualiza prin intermediul serverului web. Dacă totul este în regulă, conținutul afișat în browser trebuie să fie următorul:



Remarcați că, ceea ce vedem este rezultatul instrucțiunii echo din PHP. Dacă se obține altceva, înseamnă că instalarea nu s-a făcut în mod corect.

Mai mult, să analizăm codul sursă generat. Pentru aceasta, în browser, executați comanda de vizualizare a sursei (în cazul lui Internet Explorer, alegeți din meniul View opțiunea Source). Va trebui să obțineți următorul cod sursă:

```
<HTML><HEAD>
<TITLE>Primul HTML continind script PHP</TITLE>
</HEAD>
<BODY>
<H2>Acesta este un titlu H2 scris normal, in afara scriptului</H2>
Acest text este scris de catre script-ul PHP</BODY>
</HTML>
```

Așa cum am anunțat, codul PHP, ba chiar însuși faptul că în această pagină ar exista vreun script, nu sunt vizibile clientului, ci acesta vede doar rezultatul obținut în urma rulării.

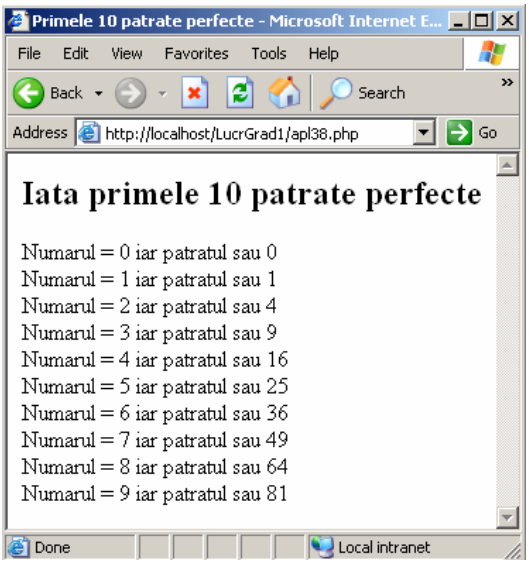
Să mai scriem încă un exemplu, înainte de a trece la detalierea limbajului de programare PHP. În același director (**c:\xampp\htdocs\phpapps**) creați fișierul **apl038.php**, cu următorul conținut:

```
1  <HTML><HEAD>
2   <TITLE>Primele 10 patrate perfecte</TITLE>
3   </HEAD>
4  <BODY>
5   <H2>Iata primele 10 patrate perfecte</H2>
6  <?php
7   for($i=0;$i<10;$i++)
8     echo "Numarul = ",$i,"  iar patratul sau ",$i*$i,"<br>";
9   ?>
10  </BODY>
11  </HTML>
```

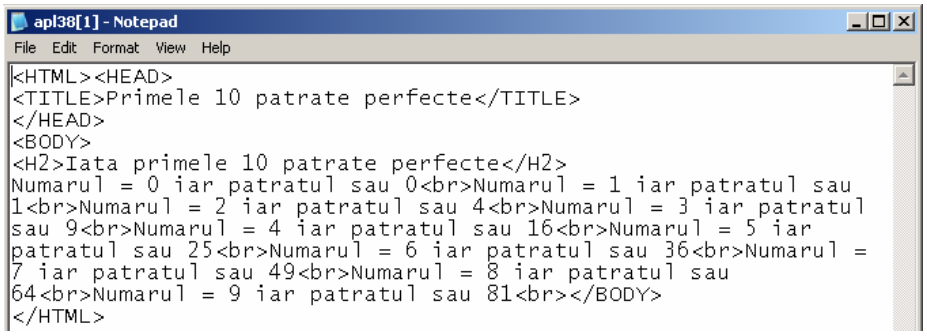
Remarcați similitudinea dintre limbajul PHP şi C, în cazul instrucțiunilor din cadrul scriptului: în afara faptului că variabila, care în C ar fi fost **i**, aici este **$i**, instrucțiunea for are

aceeaşi sintaxă. Sintaxa lui **echo** este uşor de asimilat, prin analogie cu **cout<<** (entităţile de afişat, în loc să mai fie separate de **<<** sunt separate de virgule). Remarcaţi, de asemenea, că la fiecare afişare **echo** din cadrul repetitivei **for**, este afişat tag-ul **
**, pentru ca, în pagina vizualizată, după fiecare linie să se treacă la rând nou.

Iată rezultatul pe care trebuie să-l obţinem în browser:



Din nou, vizualizând codul primit de către browser, vom obţine următorul HTML:



Remarcaţi din nou că, ceea ce ajunge la client este doar rezultatul execuţiei script-ului PHP. Observaţi că, deşi tag-ul **
** produce în browser trecerea la rând nou, în cadrul vizualizării sursei obţinute, codul este dezordonat, deoarece este scris „una-ntr-una”, fără Enter-uri.

Acest lucru, în mod normal, nu deranjează, atâta timp cât aspectul paginii vizualizate în browser are acelaşi aspect.

Totuşi, o aliniere este binevenită în cazul depanării unui script, pentru că se poate urmări mai uşor apariţia unei eventuale erori sau neconcordanţe.

Pentru acest lucru, în cadrul unui **echo** putem folosi, la afişare, oricâte treceri la rând nou dorim. Acestea se fac, la fel ca şi în C, prin caracterul special **"\n"**. După cum ştim din capitolul precedent, Enter-urile în cadrul unui text din codul HTML nu au efect, deci afişarea de **"\n"**-uri o folosim în special pentru structurarea ordonată a acestuia.

În cazul aplicației de față (**apl038.php**) să modificăm linia a 8-a a sa (afişarea cu echo) schimbând-o în următoarea:

```
8    echo "Numarul = ",$i,"  iar patratul sau ",$i*$i,"<br>\n";
```

Aspectul său în browser rămâne absolut nemodificat. Totuşi, vizualizând codul sursă, vom remarca alinierea acestuia:

apl38[1] - Notepad

File Edit Format View Help

```
<HTML><HEAD>
<TITLE>Primele 10 patrate perfecte</TITLE>
</HEAD>
<BODY>
<H2>Iata primele 10 patrate perfecte</H2>
Numarul = 0 iar patratul sau 0<br>
Numarul = 1 iar patratul sau 1<br>
Numarul = 2 iar patratul sau 4<br>
Numarul = 3 iar patratul sau 9<br>
Numarul = 4 iar patratul sau 16<br>
Numarul = 5 iar patratul sau 25<br>
Numarul = 6 iar patratul sau 36<br>
Numarul = 7 iar patratul sau 49<br>
Numarul = 8 iar patratul sau 64<br>
Numarul = 9 iar patratul sau 81<br>
</BODY>
</HTML>
```

De reținut deci, din exemplele prezentate, că:

- un fişier ce conține script-uri PHP se salvează de regulă cu exensia **.php**, fiind localizat în directorul în care rezidă documentele serverului web;
- orice script PHP este cuprins între tag-urile **<?php** şi **?>**;
- instrucțiunea PHP care produce output în pagina HTML este **echo**;
- rularea unei pagini care conține script PHP se va face ***întotdeauna*** prin intermediul serverului de web;
- la fel ca şi în C, separatorul dintre instrucțiunile unui script .php este caracterul "**;**".

O ultimă observație care trebuie făcută este aceea că, în mod implicit (deci fără a modifica setările de instalare) limbajul PHP, spre deosebire de C, nu este case-sensitive (deci nu face diferența între literele mari şi cele mici) în ceea ce priveşte cuvintele rezervate ale limbajului (instrucțiunile) respectiv funcțiile (fie că e vorba de cele predefinite ale limbajului, fie că e vorba de cele definite de către utilizator). În schimb, este case-sensitive în ceea ce priveşte numele de variabile. Astfel, fie că scriem **for** fie **FOR**, limbajul va recunoaşte instrucțiunea repetitivă cu contor. În schimb, dacă folosim identificatorii **$a** respectiv **$A**, va fi vorba de două variabile distincte.

3.4. Constante. Variabile. Operatori. Afişarea datelor.

Constante

Constantele recunoscute de limbajul PHP sunt asemănătoare cu cele ale limbajului C:

• constante numerice întregi şi reale: **14**, **-80**, **3.14**, **-8.25**, **1e+2**, **314e-2**, etc.

• constante de tip caracter şi şir de caractere: **"a"**, **"\n"**, **"Ana are mere"**, **'Cici'**, **'Mimi'**, etc. Spre deosebire de C, în PHP nu se mai face diferența între un singur caracter şi un întreg şir de caractere din punct de vedere al constantelor. Separatorii pot fi atât ghilimelele cât şi apostrofurile, rolul lor funcțional fiind puțin diferit – vom vedea acest lucru mai încolo. Caracterele speciale sunt aceleaşi ca şi în C: **\n**, ****, **\'**, **\"** şi altele.

• constante de tip **bool**: **false** şi **true**. Spre deosebire de C, rezultatul oricărei expresii relaționale sau logice este de acest tip special de date. Totuşi, în cazul instrucțiunilor care necesită expresii relaționale sau logice, se păstrează convenția din C, şi anume că orice valoare diferită de 0 este echivalentă cu **true**, iar orice valoare egală cu 0 este echivalentă cu **false**.

Definirea constantelor de către utilizator, în sensul că prin anumiți identificatori putem folosi valori constante, se face cu ajutorul funcției **define**.

Forma generală a acesteia este:

define("nume constanta", valoare);

Exemplu de script (**ap1039.php**):

```
<?php
  define("pi",314e-2);//aici am definit o constanta numerica reala
  define("greeting","Buna ziua!");//iar aici o constanta de tip sir de caractere
  echo greeting,"<br>";//aici afisam constanta sir de caractere
  echo "Constanta pi, aproximata cu 2 zecimale este ",pi;//iar aici pe cea numerica
?>
```

Variabile

În PHP identificatorii rezervați variabilelor încep cu caracterul "**$**". În continuare, respectă aceleaşi specificații din C, deci imediat după caracterul **$** trebuie să fie o literă sau liniuța de subliniere ("_"), iar în rest pot fi folosite şi cifrele.

Spre deosebire de C, variabilele nu se declară la început, ci tipul lor este definit (implicit, de către limbaj) atunci când sunt folosite. Mai mult, îşi pot schimba tipul în funcție de valoarea pe care o reține.

Exemplu de script (**apl040.php**):

```
<?php
   $a=4+5.5;//La fel ca si in C, mai intii se evalueaza expresia din dreapta.
   //Rezultatul 9.5 este real. In urma atribuirii este creata variabila $a, de tip real
   echo "Valoarea din variabila a este: ",$a,"<BR>\n";
   $b="Ana are mere";//Se creeaza variabila $b, de tip string
   $a=$b;//Modificam variabila $a, dindu-i continutul lui $b. Acesta fiind de tip
   //string, se modifica si tipul variabilei $a, de la real la string.
   //Acest lucru n-ar fi fost posibil in C
   echo "Noua valoare din variabila a este: <B><FONT color=\"red\">",$a;
   echo "\n</FONT></B>";//in plus, am mai imbogatit afisarea, folosind tag-urile
   //pentru scris bold si cel pentru caractere de culoare rosie
?>
```

Analizați legătura dintre codul sursă de mai sus şi alinierea fişierului obținut în browser (deci afişările de **\n** din cadrul instrucțiunilor **echo**) :

```
Valoarea din variabila a este: 9.5<BR>
Noua valoare din variabila a este: <B><FONT color="red">Ana are mere
</FONT></B>
```

Remarcați de asemenea şi faptul că valoarea atributului **color** (şi anume **red**) a trebuit a fi scrisă între ghilimele. Textul din cadrul **echo**-ului în care am afişat acest atribut, fiind deja în interiorul unor ghilimele, a trebuit să folosim caracterul special **\"** pentru a face această afişare posibilă. Dacă am fi folosit, pur şi simplu, ghilimele obişnuite, instrucțiunea **echo** ar fi considerat că în acel loc se încheie stringul, iar din acest caz n-ar mai fi putut interpreta caracterele următoare, ceea ce s-ar fi terminat cu producerea unei erori.

În PHP este posibilă şi adresarea indirectă. Acest lucru înseamnă că, dacă o variabilă conține o expresie de tip string în care este reținut numele unei variabile, putem afişa direct valoarea variabilei reținută de string. Pentru aceasta se va folosi încă o dată caracterul **$** (de forma **$$x**).

Exemplu de script (**apl041.php**):

```
<?php
  $a=5;
  $x="a";
  echo "Iata stringul din variabila x: ",$x,"<BR>\n";
  echo "Iata valoarea variabilei din stringul x: ",$$x;
?>
```

Operatori

Mulți dintre operatorii limbajului PHP sunt cunoscuți din C++. Acesta este motivul pentru care vom prezenta doar anumite particularități specifice limbajului PHP.

Pentru început, îi prezentăm, în ordine descrescătoare a priorităților lor:

1. **`!`** , **`++`** , **`--`** , **`(int)`** , **`(double)`** , **`(string)`** ;
2. **`*`** , **`%`** , **`/`** ;
3. **`<`** , **`<=`** , **`>`** , **`>=`** ;
4. **`==`** , **`!=`** , **`===`** , **`!==`** ;
5. **`&`** ;
6. **`^`** ;
7. **`&&`** ;
8. **`?:`** ;
9. **`=`** , **`+=`** , **`-=`** , **`/=`** , **`*=`** , **`%=`** , **`&=`** , **`|=`** , **`^=`** ;
10. **`And`** ;
11. **`Xor`** ;
12. **`Or`** ;
13. **`,`** ;

În PHP se pot folosi operatori de conversie explicită, cunoscuți din C++. Ca şi în C++, ei se aplică prefixat. Astfel, există: **`(int)`** – conversie către o valoare întreagă, **`(string)`** – conversie către şir, iar **`(double)`** – conversie către real.

Exemplu de script (**`ap1042.php`**):

```
<?php
  $a=(int)8.65;//la fel ca si in C, se vor elimina zecimalele
  echo "variabila a are valoarea: ",$a,"<BR>";
  $b=(double)"3.85copac";//conversia se va face atita cit se poate, deci variabila
  //b va contine valoarea 3.85, restul de caractere vor fi ignorate
  echo "variabila b are valoarea: ",$b,"<BR>";
  $c="1.25mere"+"3.75pere";//ba mai mult, se va face conversia explicita, adica
  //limbajul va converti mai intii cele doua stringuri la numere, apoi va face adunarea
  echo "variabila c are valoarea: ",$c,"<BR>";
  $d=19/5;//spre deosebire de C, operatorul C face impartire reala, chiar daca
  //operatorii sai sunt intregi
  echo "variabila d are valoarea: ",$d,"<BR>";
  //daca dorim impartire intreaga, facem conversia la int:
  $e=(int)(19/5);
  echo "variabila e are valoarea: ",$e,"<BR>";
?>
```

Exemplu de script (**`ap1043.php`**):

```
<?php
  echo "<TT>";
  //operatorul == functioneaza ca si in C. Limbajul PHP fiind ceva mai larg in ceea ce
  //priveste tipurile de date, verifica doar egalitatea ca valoare.
  //vom folosi functia var_dump(variabila), care ne afiseaza tipul unei variabile
  //si valoarea sa. Facem acest lucru, deoarece rezultatele unor comparatii in PHP
  //au o valoare de tipul bool (true sau false) ce nu poate fi afisata in mod direct
  $a=("3"==3);//vom obtine true, deoarece in urma conversiei, cele 2 valori sunt egale
  echo 'Iata rezultatul comparatiei "3"==3 : ';//remarcati cum, de aceasta data, pentru
  //ca stringul pe care dorim sa-l afisam contine ghilimele, l-am delimitat prin
  //apostrofuri
  var_dump($a);echo "<BR>";
```

```
  $b=("3"==3.90);//vom obtine false
  echo 'Iata rezultatul comparatiei "3"==3.90 : ';
  var_dump($b);echo "<BR>";
  $b=("3.90"==3.90);//vom obtine true
  echo 'Iata rezultatul comparatiei "3.90"==3.90 : ';
  var_dump($b);echo "<BR>";
?>
```

Exemplu de script (**apl044.php**):

```
<?php
  echo "<TT>";
  //operatorul === reprezinta o noutate fata de C. Acest operator verifica egalitatea
  //atit ca valoare cit ca si tip. Evident, operatorul !== reprezinta negatia sa.
  $a=("3"==3);//acesta este true, pentru ca valorile sunt egale
  echo 'Iata rezultatul comparatiei "3"==3 : ';
  var_dump($a);echo "<BR>";
  $b=("3"===3);//acesta este false, pentru ca, desi valorile sunt egale, tipurile nu sunt
  echo 'Iata rezultatul comparatiei "3"===3 : ';
  var_dump($b);echo "<BR>";
  $c=(1+2===3);//aceasta este true, pentru ca expresiile sunt de acelasi tip
  echo 'Iata rezultatul comparatiei 1+2===3 : ';
  var_dump($c);echo "<BR>";
  $d=(3.0===3);//aceasta este false, pentru ca tipurile nu sunt egale
  echo 'Iata rezultatul comparatiei 3.0===3 : ';
  var_dump($d);echo "<BR>";
?>
```

Exemplu de script (**apl045.php**):

```
<?php
  //Operatorul =, de atribuire, functioneaza la fel ca si in C.
  //Este asadar permisa si atribuirea multipla:
  $a=$b=$c=5.23;
  echo "Iata variabilele a, b si c, initializate toate cu aceeasi valoare: ";
  echo $a,", ",$b,", ",$c;
?>
```

Afişarea datelor

După cum am văzut deja, una dintre cele mai folosite instrucțiuni de afişare în PHP este **echo**. Are două forme:

a) data afişată se scrie între paranteze rotunde (această formă nu poate fi folosită pentru afişarea mai multor date): **echo("Ana are mere");**

b) datele afişate sunt scrise după echo, fără a fi grupate între paranteze şi separate prin virgule: **echo "Ana are ", 1+2, "mere";**

O altă instrucțiune de afişare este **print**. După ea urmează o singură dată, care poate fi sau nu pusă între paranteze. Funcționează ca şi echo, în plus, în cazul în care folosim forma cu paranteze, va întoarce valoarea **true** dacă afişarea a fost făcută cu succes, respectiv **false** în caz contrar.

Alte două instrucțiuni folosite în special pentru cazurile în care dorim să depanăm un program sunt:

- **`var_dump(expresie)`** – afişează tipul expresiei urmat de valoarea sa;
- **`print_r(variabila)`** – în cazul unor variabile compuse (şiruri, obiecte) produce o afişare a tuturor componentelor ale acestora.

Observaţii:

După cum am văzut deja prin exemplele date, în loc de ghilimele, se pot folosi şi apostrofuri. Diferenţa este dată de faptul că, în cazul folosirii ghilimelelor, dacă şirul de caractere conţine numele unor variabile, acestea vor fi evaluate, deci se va afişa conţinutul lor, pe când în cazul apostrofurilor se va afişa numele variabilei ca atare.

Nu putem folosi ghilimele incluse în cadrul altei perechi de ghilimele, şi nici apostrofuri incluse între alte perechi de apostrofuri, în schimb, putem include ghilimele într-un şir delimitat de apostrofuri sau apostrofuri într-un şir delimitat de ghilimele.

Exemplu de script (**`ap1046.php`**):

```
<?php
   $a=3;$b=4;
   echo "Ana are $a mere si $b mere<BR>";//aici se vor evalua atit $a cit si $b
   echo 'Ana are $a mere si $b mere<BR>';//pe cind aici nu
?>
```

3.5. Instrucţiuni ale limbajului PHP.

Instrucţiunile PHP sunt asemănătoare cu cele din C. Din acest motiv, ne vom limita la o scurtă prezentare a lor şi la câteva exemple de utilizare.

3.5.1. Instrucţiunea expresie.

La fel ca şi în C++ se foloseşte în special pentru calcule şi atribuiri.

Exemplu: **`$x=$x*10+3;`**

3.5.2. Instrucţiunea bloc (se mai numeşte şi compusă).

Are aceeaşi sintaxă şi funcţionalitate ca în C, şi anume de a grupa mai multe instrucţiuni, astfel încât acestea să joace rolul sintactic al uneia singure.

Instrucţiunile se scriu între paranteze acolade:

```
{ ...
  ...
}
```

3.5.3. Instrucțiunea if.

Are aceeaşi formă şi funcționalitate ca şi în C:

```
if(expresie) instrucțiune1;
        [else instrucțiune2;]
```

Deci, dacă expresia este evaluată ca fiind adevărată (sau diferită de 0) se execută **instrucțiune1**. Dacă este falsă (sau 0) iar ramura **else** este prezentă, se va executa **instrucțiune2**. La fel ca şi în C, dacă în loc de o singură instrucțiune sunt mai multe, se vor grupa într-un bloc.

Exemplu de script (**apl047.php**) care conține un if:

```
<?php
//urmatorul script ia ca si parametru de intrare variabila a definita mai jos.
//Pentru ca inca nu am prezentat cum se face preluarea de date de catre PHP, ne
//vom limita sa modificam manual variabila $a de mai jos.
//Program formeaza alte doua variabile $b si $c, cu primele doua respectiv
//ultimele doua cifre ale lui $a, daca acesta are exact 4 cifre, sau scrie un
//mesaj corespunzator in caz contrar
  $a=1425;
  echo 'Valoarea din variabila $a este : ',$a,"<BR>";
  if($a>=1000&&$a<=9999)//deci verificam sa aiba exact 4 cifre
       {//in caz afirmativ se executa instructiunile din acest bloc
        $b=(int)($a/100);
        $c=$a%100;
        echo "Primele doua cifre ale sale sunt : ",$b,"<BR>";
        echo "Ultimele doua cifre ale sale sunt : ",$c,"<BR>";
       }
  else //iar in caz contrar afisam un mesaj corespunzator
       echo "Valoarea din variabila a NU are exact 4 cifre!<BR>";
?>
```

3.5.4. Instrucțiunea while.

Are aceeaşi formă şi funcționalitate ca şi în C:

```
while(expresie) instrucțiune;
```

Principiul de executare este următorul:

Pasul P1: Se evaluează expresia;

Pasul P2: Dacă aceasta este adevărată (sau diferită de 0) se execută instrucțiunea subordonată, după care se revine la **Pasul P1**. În caz contrar se termină execuția repetitivei while, trecându-se la instrucțiunea următoare în codul sursă.

Exemplu de script (**apl048.php**) care conține **while**:

```
<?php
//Programul afiseaza cifrele numarului intreg din variabila a, in ordine inversa, dupa
//fiecare afisind caracterul #:
  $a=1425;
  echo 'Valoarea din variabila $a este : ',$a,"<BR>";
  while($a)//deci cit timp valoarea din $a este nenula
     {
       $r=$a%10;//determinam ultima sa cifra in variabila $r
       $a=(int)($a/10);//inlocuim $a cu citul impartirii sale la 10, deci
                       //"stergem" ultima sa cifra
       echo $r,"# ";//afisam cifra curenta, obtinuta in variabila $r, urmata de un #
     }
?>
```

3.5.5. Instrucțiunea do...while.

Are aceeaşi formă şi funcționalitate ca şi în C:

```
do
   instrucțiune;
 while(expresie);
```

Principiul de executare este următorul:

Pasul P1: Se execută instrucțiunea subordonată (cea de după do);

Pasul P2: Se evaluează expresia. În cazul în care valoarea evaluată este false (sau 0), executarea instrucțiunii **do**...**while** se termină. În cazul în care este adevărată (sau nenulă) se reia executarea pasului **P1**.

Exemplu de script (**apl049.php**) care conține **do**...**while**:

```
<?php
//Programul afiseaza cifrele numarului intreg din variabila a, in ordine inversa, dupa
//fiecare afisind caracterul #:
  $a=1425;
  echo 'Valoarea din variabila $a este : ',$a,"<BR>";
  do
  {
    $r=$a%10;//determinam ultima sa cifra in variabila $r
    $a=(int)($a/10);//inlocuim $a cu citul impartirii sale la 10, deci
                    //"stergem" ultima sa cifra
    echo $r,"# ";//afisam cifra curenta, obtinuta in variabila $r, urmata de un #
  }while($a);//deci repetam ciclarea cit timp valoarea din $a este nenula
?>
```

3.5.6. Instrucțiunea for.

Are aceeaşi formă şi funcționalitate ca şi în C:

for(expresie$_{\text{inițializare}}$**;expresie**$_{\text{continuare}}$**;expresie**$_{\text{incrementare}}$**) instrucțiune;**

Principiul de executare este uşor de înțeles, datorită faptului că **for**-ul se poate transcrie în mod perfect echivalent prin următoarea secvență de program:

```
expresie_iniţializare;
while(expresie_continuare)
{
    instrucțiune;
    expresie_incrementare;
}
```

Deşi for-ul este, în limbajul C, deci şi în PHP, o instrucțiune mult mai generală decât în alte limbaje, totuşi, cel mai utilizat scop al său este de a atribui unui contor, rând pe rând, valori (de regulă întregi) cuprinse între două limite.

Exemplu de script (**apl050.php**) care conține un for:

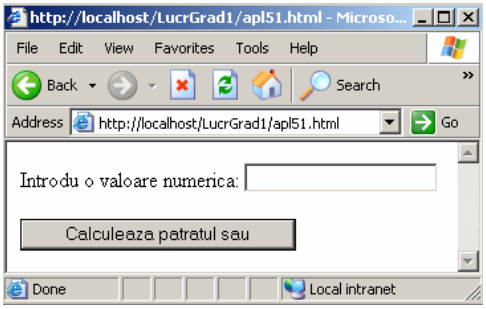
```
<?php
//Programul calculeaza suma primelor $n numere naturale in variabila $s:
  $n=10;
  $s=0;
  echo 'Valoarea din variabila $n este : ',$n,"<BR>";
  for($i=1;$i<=$n;$i++)
                  $s+=$i;
  echo "Suma primelor $n numere naturale este $s<BR>";
?>
```

3.6. Transmiterea datelor prin intermediul formularelor.

În capitolul 2 am amintit de formulare şi de rolul pe care acestea îl joacă în dinamizarea paginilor web.

Formularele reprezintă un mecanism prin care se pot trimite date către serverul HTML. Aceste date pot fi preluate de către script-urile PHP şi în continuare folosite în cadrul programelor. Să nu uităm că formularele sunt elemente HTML. Fie următorul unui formular, pe care îl creăm în fişierul **apl051.html** :

```
<form action="apl052.php" method="post">
Introdu o valoare numerica:
<input type="text" name="a" maxlength="4">
<br><br>
<input type="submit" value="Calculeaza patratul sau">
</form>
```



Atributul **action** al tag-ului **form** se referă la numele fişierului PHP care se va ocupa de prelucrarea datelor iar atributul method de metoda prin care sunt trimise datele către server.

Tag-ul **<input type="text"...>** creează un câmp de date de tip text. Atributul **name** al acestuia specifică un identificator prin care PHP-ul va prelua valoarea din acesta.

Tag-ul **<input type="submit"...>** creează un buton de trimitere a datelor. Practic, apăsarea pe acest buton permite trimiterea conţinutului întregului formular către server.

Preluarea datelor trimise către server prin intermediul metodei POST, în cadrul unui script PHP se face prin intermediul vectorului predefinit **$_POST['nume_câmp_din_formular']**. Atenţie la faptul că **$_POST** trebuie scris cu majuscule !.

De exemplu, în cazul nostru, putem recupera această valoare prin intermediul lui **$_POST['a']**.

Analog, dacă datele ar fi trimise către server prin intermediul metodei GET (deci în mod vizibil, în cererea URL, de exemplu **http://mypage.html?a=13**), în cadrul script-ului PHP asociat, preluarea lor se face prin intermediul vectorului predefinit **$_GET['nume_câmp_formular]**.

În cazul în care un anumit câmp nu există, în momentul cererii **$_POST[...]** din cadrul PHP-ului, acesta s-ar putea sa genereze un mesaj de tip atenţionare (warning) în funcţie de setări.

Pentru a evita acest lucru, în faţa caracterului **$** (de la **$_POST[...]**) punem caracterul **@**. Semnificaţia acestuia este de a ignora mesajele de tip warning.

Iată şi codul lui **apl052.php**, scriptul PHP pe care l-am anunţat în atributul action al formularului **apl051.html**, script care va trebui să se afle în acelaşi director cu **apl051.html**. Scriptul preia valoarea câmpului text din form şi afişează atât valoarea recuperată cât şi pătratul său:

```
<?php
  $a=$_POST['a'];//deci, in variabila $a preluam valoarea din cimpul cu
  //atributul name="a" al form-ului care a trimis datele.
  echo "Iata valoarea preluata din formular: ",$a,"<br><br>";
  echo "Patratul sau este ",$a*$a;
?>
```

Vom studia, în continuare, modul în care se preiau datele din toate tipurile de elemente care pot să apară într-un formular obişnuit:

- în cazul unui **input** de tip text, am văzut deja cum se face acest lucru. În exact acelaşi mod se preiau datele dintr-un **input** de tip **password**, de tip **hidden** sau de tip **textarea**.

- în cazul unui **input** de tip **submit**, valoarea (care este de fapt textul scris pe buton) se poate prelua doar în cazul în care, în cadrul tagului **<input type="submit"...>** apare şi atributul **name**. Evident, în cazul în care formularul are un singur control de tip submit, e destul de ilogic sa dorim sa vedem ce valoare are.

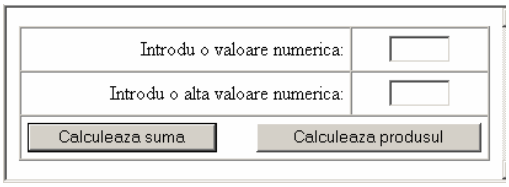
Totuşi, în cazul în care un formular are două sau mai multe butoane de tip sumbit, cărora vrem să le asignăm funcţii diferite, este foarte important să ştim care dintre ele a fost apăsat.

Iată mai jos un astfel de exemplu:

apl053.html + apl054.php – primul fişier conţine un formular cu două câmpuri text, în care introducem valori numerice. În funcţie de controlul **submit** pe care-l apăsăm, al doilea fişier va prelua datele şi va calcula fie produsul fie suma celor două valori numerice:

apl053.html:

```
<form action="apl054.php" method="post">
<table border="1" cellspacing="0" cellpadding="5">
<tr><td align="right">Introdu o valoare numerica:
<td align="center">
<input type="text" name="cta" maxlength="4" size="4">
<tr><td align="right">Introdu o alta valoare numerica:
<td align="center">
<input type="text" name="ctb" maxlength="4" size="4">
<tr><td colspan="2" align="center">
<input type="submit" value="Calculeaza suma" name="operatia">
      
<input type="submit" value="Calculeaza produsul" name="operatia">
</table>
</form>
```



apl054.php:

```
<?php
  $op=$_POST['operatia'];//controlul cu numele 'operatia' este chiar butonul de submit
 //preluam valoarea sa in variabila $op, pentru a vedea care dintre butoane a fost apasat
  $a=$_POST['cta'];//preluam si cele doua valori numerice
  $b=$_POST['ctb'];//din cimpurile text cu numele cta si ctb
  //le afisam:
  echo "S-au preluat valorile urmatoare: a=$a si b=$b<br>";
  //in functie de valoarea lui "op" calculam suma sau produsul
```

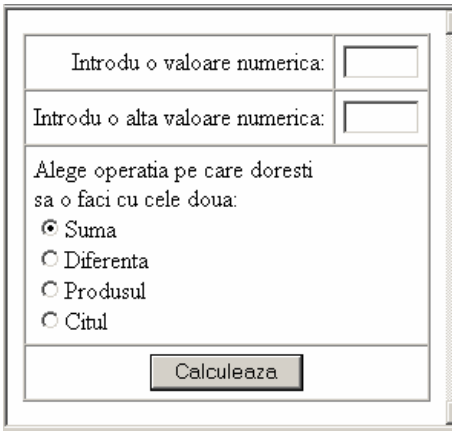
```
  if($op=="Calculeaza suma")
      echo "Suma celor doua este ",$a+$b;
  else
      echo "Produsul celor doua este ",$a*$b;
?>
```

- în cazul unui control de tip **radio**, să ne amintim mai întâi că toate tag-urile de tipul **<input type="radio"...>** trebuie să aibă la atributul **name** acelaşi nume, iar la atributul **value** valori diferite, prin care vom identifica opțiunea aleasă. Această valoare va fi trimisă către PHP.

Iată un exemplu de utilizare al controlul de tip **radio**: **apl055.html** + **apl056.php**. Cu această ocazie vom folosi şi instrucțiunea switch a limbajului PHP, instrucțiune pe care nu am prezentat-o, însă care are exact aceeaşi sintaxă şi funcționalitate ca în C.

apl055.html:

```
<form action="apl056.php" method="post">
<table border="1" cellspacing="0" cellpadding="5">
<tr><td align="right">Introdu o valoare numerica:
<td align="center">
<input type="text" name="cta" maxlength="4" size="4">
<tr><td align="right">Introdu o alta valoare numerica:
<td align="center">
<input type="text" name="ctb" maxlength="4" size="4">
<tr><td colspan="2">
Alege operatia pe care doresti<br>
sa o faci cu cele doua:<br>
<input type="radio" value="1" name="op" checked>Suma<br>
<input type="radio" value="2" name="op">Diferenta<br>
<input type="radio" value="3" name="op">Produsul<br>
<input type="radio" value="4" name="op">Citul
<tr><td colspan="2" align="center">
<input type="submit" value="Calculeaza">
</table>
</form>
```



apl056.php:

```
<?php
  $op=$_POST['op'];//controlul cu numele 'op' este grupul de butoane radio.
  //preluam valoarea sa in variabila $op, pentru a vedea care optiuni a fost aleasa
  $a=$_POST['cta'];//preluam si cele doua valori numerice
  $b=$_POST['ctb'];//din cimpurile text cu numele cta si ctb
  //le afisam:
  echo "S-au preluat valorile urmatoare: a=$a si b=$b<br>";
  //in functie de valoarea lui "op" calculam suma sau produsul
  switch($op)
  {
    case 1:
      echo "Suma celor doua este ",$a+$b;
      break;
    case 2:
      echo "Diferenta celor doua este ",$a-$b;
      break;
    case 3:
      echo "Produsul celor doua este ",$a*$b;
      break;
    default:
      echo "citul celor doua este ",$a/$b;
  }
?>
```

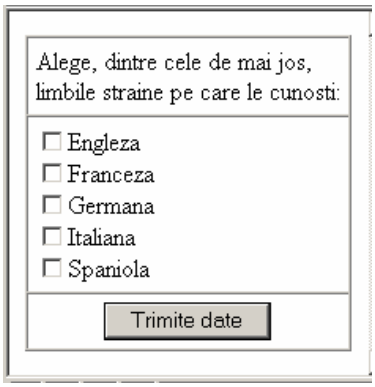
- în cazul unui control de tip **checkbox**, să ne reamintim că fiecare control de acest tip are un nume separat. Dacă este bifat, va trimite către PHP valoarea indicată în atributul **value** a tag-ului **<input type="checkbox"...>** (ca şir de caractere). Dacă acest atribut nu este prezent, valoarea trimisă către PHP va fi şirul de caractere „on".

În schimb, dacă nu este bifat, pur şi simplu nu va trimite nimic, deci s-ar putea ca cererea **$_POST[...]** sa genereze un warning (depinde şi de setările PHP-ului). Pentru ca acest lucru să nu se întâmple, indiferent de setări, aşa cum am anunţat mai înainte, folosim caracterul **@** în faţa cererii **$_POST[...]**.

Iată mai jos un exemplu: **apl057.html** + **apl058.php**

apl057.html:

```
<form action="apl058.php" method="post">
<table border="1" cellspacing="0" cellpadding="5">
<tr><td>Alege, dintre cele de mai jos,<br>
limbile straine pe care le cunosti:
<tr><td>
<input type="checkbox" name="en">Engleza<br>
<input type="checkbox" name="fr">Franceza<br>
<input type="checkbox" name="ge">Germana<br>
<input type="checkbox" name="it">Italiana<br>
<input type="checkbox" name="es">Spaniola<br>
<tr><td align="center">
<input type="submit" value="Trimite date">
</table>
</form>
```



apl058.php:

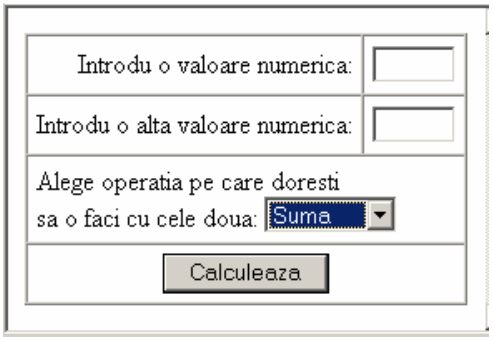
```
<?php
   $en=@$_POST['en'];
   $fr=@$_POST['fr'];
   $ge=@$_POST['ge'];
   $it=@$_POST['it'];
   $es=@$_POST['es'];
   $l=0;//$l este un flag pe care-l facem 1 cind dam de o limba straina bifata
   echo "Iata limbile straine pe care le-ai bifat:<br>";
   if($en=="on") {echo "Engleza<br>";$l=1;}
   if($fr=="on") {echo "Franceza<br>";$l=1;}
   if($ge=="on") {echo "Germana<br>";$l=1;}
   if($it=="on") {echo "Italiana<br>";$l=1;}
   if($es=="on") {echo "Spaniola<br>";$l=1;}
   if($l==0)//daca flag-ul a ramas 0, dam utilizatorului
   //un mesaj prin carae il informam ca nu a bifat nimic
      echo "Nu ai bifat nici una dintre limbile straine!";
?>
```

- în cazul unui control de tip **select** simplu, PHP-ul va putea recupera valoarea cu ajutorul numelui stabilit în atributul **name** al tag-ului **<select ...>**, valoarea trimisă fiind cea stabilită în atributul **value** al tag-urilor **option** înglobate în cadrul select-ului.

Iată mai jos un exemplu (**apl059.html** + **apl060.php**) care reia ideea din **apl055.html** cu deosebirea că, în loc de a alege operaţia dorită prin intermediul unui control **radio**, o alegem cu ajutorul controlului **select**. A se remarca faptul că fişierul care prelucrează datele (**apl060.php**) a rămas identic cu cel care prelucra datele din **apl055.html**.

apl059.html:

```
<form action="apl060.php" method="post">
<table border="1" cellspacing="0" cellpadding="5">
<tr><td align="right">Introdu o valoare numerica:
<td align="center">
<input type="text" name="cta" maxlength="4" size="4">
<tr><td align="right">Introdu o alta valoare numerica:
<td align="center">
<input type="text" name="ctb" maxlength="4" size="4">
<tr><td colspan="2">Alege operatia pe care doresti<br>
sa o faci cu cele doua:
<select name="op">
<option value="1">Suma
<option value="2">Diferenta
<option value="3">Produsul
<option value="4">Citul
</select><tr><td colspan="2" align="center">
<input type="submit" value="Calculeaza"></table></form>
```



apl060.php:

```
<?php
  $op=$_POST['op'];//controlul cu numele 'op' este cel de tip option
  $a=@$_POST['cta'];
  $b=@$_POST['ctb'];
  echo "S-au preluat valorile urmatoare: a=$a si b=$b<br>";
  switch($op)
  {
    case 1:
      echo "Suma celor doua este ",$a+$b;
      break;
    case 2:
      echo "Diferenta celor doua este ",$a-$b;
      break;
    case 3:
      echo "Produsul celor doua este ",$a*$b;
      break;
    default:
      echo "citul celor doua este ",$a/$b;
  }
?>
```

- în cazul unui control de tip **select** multiplu, form-ul va trimite către PHP un şir în care vom regăsi valorile selectate. Este obligatoriu, totuşi, ca atributul **name** din cadrul tag-ului **<select multiple ...>** să specifice faptul că se va trimite un şir. Acest lucru se face punând un set de paranteze pătrate după numele câmpului, deci de forma **name="nume_sir[]"**.

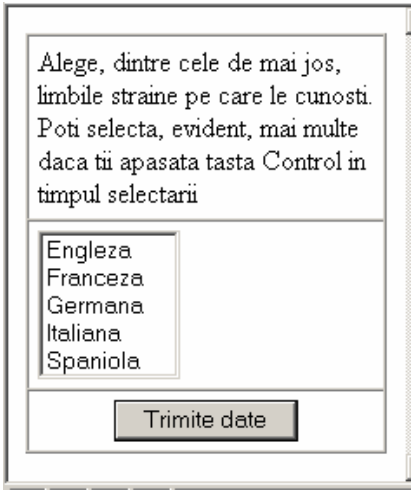
Preluarea în PHP se face în mod normal, prin **@variabila=@$_POST['nume_sir']** (remarcați faptul că nu se mai pun []).

Acest şir va conține pe post de elemente valorile stabilite prin atributul **value** ale opțiunilor selectate. Şirul va începe de la indicele 0, numărul său total de elemente fiind dat de funcția **count(nume_şir)**.

Iată un exemplu care exploatează o listă de tipul **<select multiple...>** (**apl061.html** + **apl062.php**)

apl061.html:

```
<form action="apl062.php" method="post">
<table border="1" cellspacing="0" cellpadding="5">
<tr><td>Alege, dintre cele de mai jos,<br>
limbile straine pe care le cunosti.<br>
Poti selecta, evident, mai multe<br>
daca tii apasata tasta Control in<br>
timpul selectarii
<tr><td>
<select name="lang[]" multiple size="5">
<option value="en">Engleza
<option value="fr">Franceza
<option value="ge">Germana
<option value="it">Italiana
<option value="sp">Spaniola
</select>
<tr><td align="center">
<input type="submit" value="Trimite date">
</table>
</form>
```



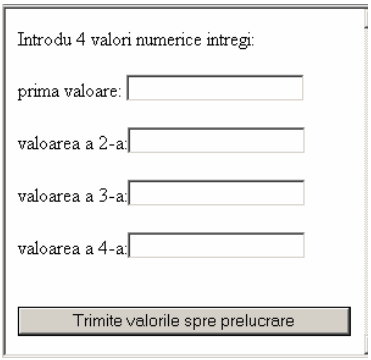
apl062.php:

```
<?php
  $lang=$_POST['lang'];//recuperam sirul trimis de catre form prin cimpul lang
  $nl=count($lang);//apelam functia count pentru a vedea cite elemente are sirul
  if($nl==0)
       echo "Nu ai selectat nici o limba straina<br>";
  else
    {
       echo "Ai selectat $nl limbi straine. Acestea sunt:<br>";
       for($i=0;$i<$nl;$i++)//parcurgem sirul pe un for, de la 0 la $nl-1
          switch($lang[$i])//si vedem ce valoare are fiecare element, in functie de
           {  //care afisam:
              case "en":echo "Engleza<br>";break;
              case "fr":echo "Franceza<br>";break;
              case "ge":echo "Germana<br>";break;
              case "it":echo "Italiana<br>";break;
              default:echo "Spaniola<br>";
           }
    }
?>
```

- una dintre facilitățile transmiterii datelor prin intermediul formularelor constă în posibilitatea trimiterii rezultatului mai multor controale sub forma elementelor unui şir sau chiar matrice. În acest caz, atributul **name** al controlului din form trebuie să specifice acel element din şir (sau matrice) care va primi valoarea sa, deci să fie de forma: **name="nume_şir[indice]"** respectiv **name="nume_matrice[indice_linie][indice_coloană]"**. Iată un exemplu în care creăm un form cu 4 controale de tip input type="text", ale căror valori vor fi preluate de către un şir cu 4 elemente (**apl063.html** + **apl064.php**):

apl063.html:

```
<form action="apl064.php" method="post">
Introdu 4 valori numerice intregi:<br><br>
prima valoare: <input type="text" name="a[1]"><br><br>
valoarea a 2-a:<input type="text" name="a[2]"><br><br>
valoarea a 3-a:<input type="text" name="a[3]"><br><br>
valoarea a 4-a:<input type="text" name="a[4]"><br><br>
<br>
<input type="submit" value="Trimite valorile spre prelucrare">
</form>
```



apl064.php:

```
<?php
$a=$_POST['a'];
echo 'Cele 4 valori sunt: ',$a[1],' ',$a[2],' ',$a[3],' ',$a[4],'<br>';
$s=0;
for($i=1;$i<=4;$i++)
                $s+=$a[$i];
echo '<br>Suma lor este: ',$s;
?>
```

3.7. Funcții în PHP.

În PHP se pot scrie funcții într-un mod foarte similar față de ceea ce ştim din C/C++. Să considerăm exemplul următor, în care se utilizează o funcție care calculează şi întoarce suma unui număr întreg pe care îl primeşte ca parametru:

apl065.php:

```
<?php
   function suma_c($n)//antetul funcției: se remarcă folosirea cuvântului cheie
                      //„function” fără a se specifica vreun tip (nici măcar void)
   {
              $s=0;
              while($n)
              {
                     $s+=$n%10;
                     $n=(int)($n/10);//ne reamintim că în PHP operatorul / calculează
                     //câtul real (deci cu zecimale): din acest motiv folosim conversia
                     //explicită către int
              }
              return $s;//la fel ca şi în C++, rezultatul întors de funcție se specifică
                        //prin intermediul instrucțiunii return
   }

   echo suma_c(13254);
?>
```

Aşadar:

- Pentru a întoarce un rezultat din corpul unei funcții, se foloseşte, aşa cum suntem deja obişnuiți, instrucțiunea **return**. Forma sa generală este: **return *expresie***.
- În cazul unei funcții care nu întoarce rezultat (deci echivalentul unei funcții de tip **void** din C++), cuvântul cheie **function** rămâne obligatoriu (fără a specifica nimic suplimentar) putând să apară oriunde în cadrul funcției cuvântul cheie **return** fără a mai fi urmat de vreo expresie, având ca efect ieşirea imediată din funcție. El poate fi, de asemenea, omis, ieşirea din funcție producându-se în acest caz în mod natural (după executarea întregului său cod).
- Transmiterea parametrilor se poate face atât prin valoare cât şi prin referință, în acelaşi mod în care se face şi în C++ :

1) Parametrii specificați în mod direct (folosind doar numele variabilei) sunt cei transmişi prin valoare. Chiar dacă valoarea unui astfel de parametru este schimbată în cadrul funcției, ea rămâne

totuşi neschimbată după executarea acesteia. Atunci când apelăm funcţia, pe poziţiile acestor parametri se pot transmite atât valori cât şi conţinutul unor variabile.

Script-ul următor afişează valoarea variabilei de dinainte de apelul funcţiei (10) chiar dacă în funcţie s-a încercat modificarea valorii transmise (**apl066.php**):

```
<?php
   function test($x)
   {
       $x=$x*2;
   }
   $a=10;
   test($a);
   echo $a;
?>
```

2) Parametrii pe care dorim să-i transmitem prin referinţă, trebuie precedaţi de caracterul **&** (ampersand) în antetul funcţiei. Dacă valoarea unui astfel de parametru este modificată în cadrul funcţiei, ea rămâne modificată şi după executarea acesteia. Evident, atunci când apelăm funcţia, pe poziţiile acestor parametrii e obligatoriu să specificăm variabile.

Reluăm scriptul anterior, punând un **&** (ampersand) în faţa parametrului **$x**. În acest fel, valoarea variabilei după ce apelăm funcţia nu va mai fi cea anterioară (10) ci cea obţinută în urma modificării (20) – **apl067.php**:

```
<?php
   function test(&$x)
   {
              $x=$x*2;
   }
   $a=10;
   test($a);
   echo $a;
?>
```

- În funcţie de domeniul de vizibilitate, şi în PHP variabilele se clasifică în *variabile locale* şi *variabile globale*. Cele globale sunt cele definite în afara oricărei funcţii. Implicit, ele nu pot fi adresate din corpul vreunei funcţii. Totuşi, există o posibilitate de a face acest lucru: în cadrul funcţiei în care vrem să utilizăm variabile globale, le specificăm pe toate, precedate de cuvântul cheie **global**. Puteţi observa acest mecanism în exemplu următor:

apl068.php:

```
<?php
   function suma()
   {
       global $a,$b,$c;
       //functia se bazeaza pe cele 3 variabile globale $a, $b si $c
       //calculând în $c suma dintre $a si $b
       //daca nu am fi pus instructiunea "global", exemplul nu ar fi functionat deoarece
       //cele 3 variabile nu ar fi fost recunoscute in functie.
       $c=$a+$b;
   }

   $a=5;$b=6;
   suma();
   echo $c;
?>
```

• Variabilele *locale* sunt variabilele create în corpul unei funcții sau cele create prin transmiterea parametrilor formali (din antetul funcțiilor). Ele nu sunt recunoscute în afara funcțiilor. Şi în PHP funcțiile pot fi recursive.

• Pentru a nu încărca foarte mult un anumit script, putem îngloba toate definițiile complete ale funcțiilor într-un singur fişier de tip text (preferabil cu extensia .php, pentru a nu putea fi vizualizat accidental prin intermediul server-ului http). Mecanismul includerii este unul asemănător cu cel din C/C++ (clauza **#include...**) cu deosebirea că, putem include codul sursă al funcțiilor în orice loc dorim. Acest lucru se face cu ajutorul funcției predefinite PHP: **require("nume_fisier");** .

Codul sursă al funcțiilor trebuie inclus şi el între tag-urile **<?php** .. **?>**.

În exemplul de mai jos, vom crea două fişiere: apl069.php, ce conține definiția corectă a unei funcții care verifică dacă un număr este prim, şi un altul apl070.php, ce afişează toate numerele prime dintre 1 şi n, folosind funcția din apl069.php:

apl069.php:

```
<?php
//a se remarca folosirea tag-urilor specifice includerii unei secvențe PHP
   function is_prime($x)
   {//$x este un parametru prin valoare
       for($i=2;$i<=sqrt($x);$i++)//variabila $i este locală
              if($x%$i==0) return 0;
       if($x<=1) return 0;
       return 1;
   }
?>
```

apl070.php:

```
<?php
   require("apl069.php");//prin această funcție includem codul existent în apl069.php
   for($i=2;$i<=100;$i++)
       if(is_prime($i)) echo $i,"   ";
//o dovadă a comportamentului local al variabilei $i din cadrul funcției constă în faptul
//că şi în codul principal (mai sus) folosim tot o variabilă $i, iar cele două nu se
//încurcă între ele
?>
```

• Limbajul PHP este înzestrat cu biblioteci care conțin numeroase alte funcții. Iată câteva dintre funcțiile matematice predefinite:

- **abs(**număr**)** – întoarce modulul numărului (valoarea absolută);

- **sin(**x**)**, **cos(**x**)**, **tan(**x**)** – sinusul, cosinusul şi tangenta unui unghi. Argumentul x trebuie specificat în radiani;
- **exp(**x**)** – întoarce e^x ;

- **pow(**x, y**)** – întoarce x^y ;

- **log10(**x**)**, **log(**x**)** – întorc $\log_{10}(x)$, respectiv $\ln(x)$;

- **max(**x_1, x_2, ... , x_n**)** – întoarce maximul (cel mai mare) dintre argumentele sale numerice;

- **min(**x_1, x_2, ... , x_n**)** – întoarce minimul (cel mai mic) dintre argumentele sale numerice;
- **ceil(**x**)** – întoarce cel mai mic nr. întreg care este mai mare sau egal cu x;
- **floor(**x**)** – întoarce cel mai mic nr. întreg mai mare sau egal cu x (partea întreagă d.p.d.v. matematic);
- **rand(**min,max**)** – întoarce o valoare întreagă aleatoare cuprinsă între valorile întregi min şi max (inclusiv);
- **pi()** – întoarce o aproximație a numărului π;
- **sqrt(**x**)** – calculează radicalul (rădăcina pătrată) a lui x.

3.8. Prelucrarea şirurilor de caractere.

Spre deosebire de C/C++, unde şirurile de caractere sunt privite ca pointeri, în PHP şirurile de caractere sunt privite mai degrabă ca nişte variabile de sine stătătoare.

În PHP sunt permise atribuirile directe (prin operatorul =) între două şiruri de caractere.

Din punct de vedere structural, în PHP şirurile de caractere nu mai respectă regulile din C/C++ (şi în special, ***NU*** mai este valabilă marca sfârşitului de string prin caracterul de cod ASCII 0), în schimb, limbajul ne pune la dispoziție o serie de funcții care fac foarte simplă prelucrarea şi manipularea stringurilor.

În rest, şirurile se memorează ca o succesiune de caractere ASCII. Putem adresa fiecare caracter al şirului prin indicele său, la fel ca în C/C++, începând de la 0.

- Funcția **strlen(şir)** ne întoarce lungimea şirului (numărul său de caractere).

Dacă dorim să parcurgem şirul de caractere, putem să-l parcurgem, ca în C++, de la **0** la **strlen(...)-1**.

apl071.php: Afişăm caracterele unui string, într-un tabel cu o singură linie:

```
<?php
  $a="Iepurechin";
  echo '<table border="1" cellspacing="0" cellpadding="10"><tr>';
  for($i=0;$i<strlen($a);$i++)
      echo '<td>',$a[$i];
  echo '</table>';
?>
```

- Concatenarea a două sau mai multe stringuri se face cu operatorul "**.**" (punctul). Observați în exemplu de mai jos cum se face acest lucru, şi de asemenea faptul că, în timpul concatenării, valoarea numerică a fost convertită implicit către şir de caractere:

```
  $x=9;
  $s="Ana "."are ".$x." mere";
  echo $s;//valoarea finală a stringului este "Ana are 9 mere"
```

• Funcția **strpos(sir1, sir2, [poz_start])** caută dacă **sir2** este subșir al lui **sir1** (eventual începând de la poziția **poz_start**, dacă aceasta este specificată). În caz afirmativ, întoarce poziția de început a acestuia, altfel întoarce **false**. În exemplul de mai jos se va afișa valoarea 3 (pentru că la indicele 3 este găsită secvența ”pu” în stringul dat):

```
$s="computer";
echo strpos($s,"pu");
```

Funcția se utilizează și pentru a testa dacă un șir include sau nu un anumit subșir. Dacă subșirul este găsit, funcția întoarce poziția de început a acestuia iar dacă nu, întoarce false. După cum știm, valoarea lui false este, de fapt, 0. Pentru a diferenția cazurile în care subșirul nu apare deloc în șir sau apare chiar de la poziția 0, folosim operatorul ===, care rezolvă corect problema:

```
$s="ana are mere";
$gasit=strpos($s,"a");
if($gasit===false) echo "Negasit";
else echo "Gasit la indicele ",$gasit;
```

Valoarea întoarsă de funcție este reținută de variabila **$gasit**. Pentru a face distincție între **false** și **0**, folosim operatorul **===**, care testează coincidența atât ca valoare, cât și ca tip. De altfel, acesta este și rostul unui astfel de operator. Acest procedeu se poate folosi și pentru alte funcții.

• Funcția **strstr(sir1, sir2)** returnează din **sir1** secvența de caractere din poziția în care a fost găsită prima apariție a lui **sir2**, dacă **sir2** este subșir pentru **sir1** sau **false**, în caz contrar.

• Funcția **strcmp(sir1, sir2)** compară lexicografic (alfabetic) **sir1** cu **sir2**. Valoarea întoarsă este:

- pozitivă, dacă **sir1** se găsește lexicografic după **sir2**; (**sir1>sir2**)
- nulă (0), dacă **sir1** este identic egal cu **sir2**; (**sir1==sir2**)
- negativă, dacă **sir2** se găsește lexicografic înainte de **sir1**. (**sir1<sir2**)

În PHP se pot folosi deopotrivă și pentru stringuri operatorii relaționali: **<**, **<=**, **>**, **>=**, **==**, **!=**, care le compară direct, în sens lexicografic, având ca rezultat **true** sau **false**.

Observație: Comparația face distincție între literele mari și cele mici (codul ASCII) !

• Funcția **substr(sir, indice, [lungime])** întoarce subșirul șirului **șir**, care începe în poziția **indice** și are lungimea **lungime**. Dacă parametru **lungime** este absent, se întoarce șirul care începe în poziția **indice** și ține până la sfârșitul șirului **șir**.

Exemplu: **apl072.php**

```
<?php
  $s="televiziune";
  $s1=substr($s,1,4);
  echo $s1,'<br>';//afiseaza "elev";
  $s2=substr($s,4);
  echo $s2;//afiseaza "viziune";
?>
```

• Funcția **substr_replace(sir1, sir2, ind, [lung])** întoarce şirul rezultat prin înlocuirea în **sir1**, a subşirului care începe în poziția **ind** şi are lungimea **lung**, cu **sir2**. Dacă parametrul **lung** este absent, **sir2** înlocuieşte subşirul care începe cu **ind** şi ține până la sfârşitul şirului **şir1**, cu **şir2**.

Exemplu: **apl073.php**

```
<?php
  $s="Ana are mere";
  echo substr_replace($s,"poseda",4,3); //înlocuieşte subşirul ”are” (care începe la
  //indicele 4 şi are 3 litere) cu subşirul ”poseda”, deci afişează ”Ana poseda mere”
?>
```

• Funcția **strtoupper(sir)** întoarce şirul rezultat prin conversia doar a acelor caractere sunt litere mici, la litere mari, iar funcția **strtolower(sir)** întoarce şirul rezultat prin conversia literelor mari la litere mici.

• Funcția **strtok(...)** este utilizată pentru extrage substringuri ale unui string, care în acesta sunt delimitate de nişte caractere ce aparțin unui şir cu delimitatori, dat. De exemplu, dacă avem stringul ”Ana? Nu, doar tu.” şi considerăm stringul cu separatori: ”.? #,” (este şi un caracter spațiu printre ele) se vor separa şi obține substringurile ”Ana”, ”Nu”, ”doar”, ”tu”.

Modelul de aplicare:

- inițial se apelează funcția **strtok(string, şir_cu_separatori)**. Funcția va întoarce fie primul cuvânt obținut prin separare, fie **false** dacă acest lucru nu a fost posibil.

- apoi se apelează funcția **strtok(şir_cu_separatori)**. Aceasta va tot extrage câte un cuvânt nou obținut prin separare, fie **false** dacă s-a ajuns deja la sfârşit, deci dacă au fost separate toate cuvintele.

Este recomandat ca testarea valorii **false** întoarsă de funcție să fie testată cu operatorul **===** deoarece, în caz contrar este posibil să apară erori din cauza conversiilor (de exemplu, funcția să întoarcă 0, iar acesta să fie interpretat ca şi **false**).

De remarcat, spre deosebire de limbajul C++, că funcția strtok NU distruge stringul asupra căreia este aplicată, ci îl lasă intact, exact ca înainte de separare.

Exemplu: **apl074.php**

```
<?php
  $s="Ana? Nu, doar tu.";
  //varianta 1: pe un while:
  $p=strtok($s,"?,.# ");
  echo "Stringul initial este:",$s,'<br>Cuvintele sale sunt:<br><br>';
  while($p!==false)
    {
      echo "cuvintul <b>",$p,"</b> are ",strlen($p),' litere<br>';
      $p=strtok("?.#, ");
    }
  //varianta 2: pe un for "mascat":
  echo '<br>Iata acelasi string, separat in acelasi mod, folosind o repetitiva';
  echo ' de tip for:<br>';
  for($p=strtok($s,"?.,# ");$p!==false;$p=strtok("., ?#"))
          echo "Cuvintul curent <b>",$p,"</b> are ",strlen($p)," litere<br>";
?>
```

3.9. Şiruri (masive) în PHP.

În PHP există mult mai puține constrângeri decât în C/C++ atunci când lucrăm cu şiruri sau cu matrice. În primul rând, în PHP un şir nu se declară. În momentul în care dorim să creăm un şir (sau o matrice), pur şi simplu atribuim valori elementelor:

apl075.php:

```
<?php
  for($i=1;$i<=5;$i++)
                  $x[$i]=$i;
  for($i=1;$i<=5;$i++) echo $x[$i]," ";
?>
```

sau **apl076.php:**

```
<?php
  $k=1;
  for($i=1;$i<=5;$i++)
     for($j=1;$j<=5;$j++)
      $a[$i][$j]=$k++;
  for($i=1;$i<=5;$i++)
  {
     for($j=1;$j<=5;$j++)
            echo $a[$i][$j],' ';
        echo '<br>';
  }
?>
```

Dacă unui element aflat la un anumit indice nu i-am atribuit nici o valoare, în orice evaluare, acel element are valoarea **NULL**. Nu este obligatoriu să folosim indici consecutivi.

Ba mai mult, în PHP şirurile pot primi pe post de indici chiar şi şiruri de caractere. Acest fel de tablou se numeşte ***tablou asociativ***. De exemplu, putem crea un şir în care să reținem pe post de indici denumirile unor produse, iar pe post de indici prețurile acestora:

apl077.php:

```
<?php
  $x['Piine']=1.61;
  $x['Vin']=5.99;
  $x['Alune']=2.21;
  $x['Ciocolata']=2.69;
  echo $x['Piine'],' ',$x['Vin'],' ',$x['Alune'],' ',$x['Ciocolata'];
?>
```

Evident, parcurgerea unui astfel de şir (variabila de ciclare printre indici nu mai respectă o regulă clasică, numerică) nu se mai face după regulile clasice, ci există o instrucțiune specială de ciclare ce permite parcurgerea vectorului în ordinea în care elementele au fost create, cu determinarea, pentru fiecare element, a perechii **indice**, **valoare**:

```
foreach(vector as indice=>valoare)
                                instrucțiune;
```

- Funcția count(vector) întoarce numărul total de elemente (folosite, deci cărora le-am atribuit valori) ale vectorului.

apl078.php:

```
<?php
  $x['Piine']=1.61;
  $x['Vin']=5.99;
  $x['Alune']=2.21;
  $x['Ciocolata']=2.69;
  echo "Sirul are in total ",count($x)," elemente.<br>";
  echo "Acestea sunt:<hr>";
  $s=0;
  foreach($x as $indice=>$valoare)
      {
        echo $indice," in valoare de ",$valoare,"<br>";
        $s=$s+$valoare;
      }
  echo "<hr>Valoarea totala:",$s;
?>
```

Dacă în cadrul instrucțiunii **foreach** omitem partea cu "**indice=>**" (deci lăsăm instrucțiunea în forma **foreach(vector as valoare) ...**, se vor parcurge doar valorile vectorului, indicii fiind omişi.

Există şi alte funcții care facilitează accesul la elementele vectorului. Acestea se bazează pe următoarea particularitate a implementării unui vector în PHP: fiecare vector are asociat un pointer intern, pointer care se află pe un anumit element al şirului (numit element curent):

- **current(**vector**)** ; - întoarce valoarea reținută de elementul curent al vectorului;
- **key(**vector**)** ; - întoarce indicele elementului curent al vectorului;

Ambele funcții (**current** şi **key**), dacă pointerul a trecut de vreunul din capete, sau dacă şirul este vid, întorc valoarea **false**. Din cauză că valoarea **0** sau **""** pot fi valori valide pentru şir, pentru a testa dacă funcția a întors valoarea **false**, trebuie utilizat operatorul ===

- **next(**vector**)** ; - deplasează pointerul pe elementul următor al vectorului;
- **prev(**vector**)** ; - deplasează pointerul pe elementul anterior al vectorului;
- **reset(**vector**)** ; - deplasează pointerul pe primul element al vectorului;
- **end(**vector**)** ; - deplasează pointerul pe ultimul element al vectorului;

Exemplu: **apl079.php** (care realizează exact acelaşi lucru ca şi aplicația precedentă)

```
<?php
  $x['Piine']=1.61;
  $x['Vin']=5.99;
  $x['Alune']=2.21;
  $x['Ciocolata']=2.69;
  echo "Sirul are in total ",count($x)," elemente.<br>";
  echo "Acestea sunt:<hr>";
  $s=0;
  while(($valoare=current($x))!==false)
      {
            echo key($x)," in valoare de ",$valoare,"<br>";
            next($x);
            $s=$s+$valoare;
        }
  echo "<hr>Valoarea totala:",$s;
?>
```

Există de asemenea o serie de funcții cu ajutorul cărora putem sorta elementele unui vector. Le vom exemplifica pe şirul din exemplul de mai sus (`$x['Piine']=1.61;  $x['Vin']=5.99; $x['Alune']=2.21;  $x['Ciocolata']=2.69;`) în ideea că, după ce am aplicat funcția specifică de sortare, afişăm şirul nou cu ajutorul următoarei secvențe:

```
foreach($x as $i=>$v)
            echo $i," => ",$v,"<br>";
```

- **asort(**vector**)** : sortează crescător vectorul după valorile reținute de fiecare element. Indicii se vor interschimba, evident, în mod corespunzător;

```
Piine => 1.61
Alune => 2.21
Ciocolata => 2.69
Vin => 5.99
```

- **arsort(**vector**)** : sortează descrescător vectorul după valorile reținute de fiecare element. Indicii se vor interschimba, evident, în mod corespunzător;

```
Vin => 5.99
Ciocolata => 2.69
Alune => 2.21
Piine => 1.61
```

- **ksort(**vector**)** : sortează crescător vectorul după valorile reținute de indici. Valorile se vor interschimba, evident, în mod corespunzător;

```
Alune => 2.21
Ciocolata => 2.69
Piine => 1.61
Vin => 5.99
```

- **krsort(**vector**)** : sortează crescător vectorul după valorile reținute de indici. Valorile se vor interschimba, evident, în mod corespunzător;

```
Vin => 5.99
Piine => 1.61
Ciocolata => 2.69
Alune => 2.21
```

- **sort(**vector**)** şi **rsort(**vector**)** sortează crescător, respectiv descrescător, valorile elementelor vectorului. Valorile indicilor se pierd, înlocuindu-se cu indici numerici având valori cuprinse între **0** şi **nr.elemente-1**.

```
Şirul $x în urma
unui sort($x):
0 => 1.61
1 => 2.21
2 => 2.69
3 => 5.99
```

3.10. Programare grafică utilizând PHP.

Una dintre cele mai spectaculoase componente ale multor limbaje de programare, din păcate lăsată de multe ori în umbră în majoritatea cursurilor de programare, constă în crearea şi manipularea imaginilor.

Limbajul PHP nu este limitat doar la crearea de output HTML, ci poate fi folosit şi pentru a crea respectiv a lucra cu diferite formate de imagini, incluzând gif, png, jpg. Ba mai mult, PHP poate genera o imagine sub forma unui flux de date (deci fără a o înregistra efectiv sub forma unui fişier pe server) direct către browser.

Pentru a face funcțional suportul grafic al limbajului, este necesară utilizarea bibliotecii **gd2**. În cazul pachetului XAMPP, aceasta este instalată şi activată în mod implicit. În cazul altor distribuții, acest lucru trebuie făcut manual.

Pentru început, să vedem cum se poate crea în mod dinamic o imagine: vom crea un script PHP, a cărui deschidere, în browser, va avea ca efect vizualizarea imaginii create de către acest script. Veți remarca faptul că, o serie de elemente (imaginile, textul, culorile) au nevoie de nişte variabile numite „variabile resursă”.

Din păcate, spre deosebire de o serie de alte limbaje de programare, nu există constante predefinite pentru culori, ci acestea trebuie să fie create manual, specificând pentru fiecare componentele de roşu, verde şi albastru, întocmai ca la HTML, cu deosebirea faptului că, de această dată, cele trei componente de culoare se specifică în baza 10, deci printr-un număr de la 0 la 255.

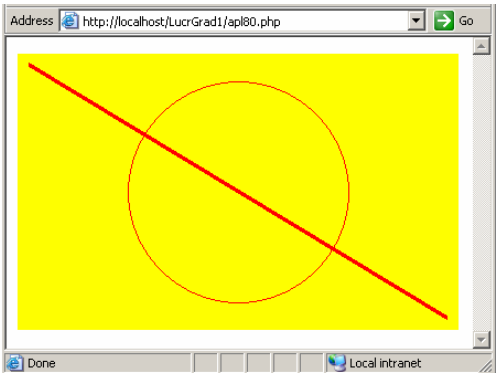
Script-ul conține comentariile necesare înțelegerii codului său. După prezentarea sa vom relua detaliat fiecare dintre funcțiile folosite.

ap1080.php

```
<?php
  header("Content-type: image/png");
/*aceasta instructiune va atasa fluxului de date creat, care la utilizator
va ajunge sub forma de fisier, informatii asupra faptului ca este vorba de un
fisier imagine, si anume de tip png.
  Aceasta informatie este esentiala pentru browser pentru a deschide fisierul ca
si pe o imagine. Puteti incerca ce se intimpla daca omiteti apelul acestei functii:
browser-ul va primi datele sub forma de text, deci va afisa o serie de caractere
aparent fara nici un sens*/
  $imagine=imagecreatetruecolor(400,250);
/* aceasta instructiune creeaza o resursa de tip imagine, pe 32 de biti (truecolor)
ce va putea fi identificata in continuare prin variabila $imagine. Imaginea va avea
latimea de 400 de pixeli, si inaltimea de 250.
*/
  $galben=imagecolorallocate($imagine,255,255,0);
/* prin aceasta instructiune am definit culoarea galben */
  imagefilledrectangle($imagine,0,0,399,249,$galben);
/* iar prin aceasta instructiune am desenat un dreptunghi plin, de culoare galbena,
ce se intinde peste toata imaginea: (0,0) respectiv (399,249) sunt doua colturi diaginale
ale dreptunghiului.
  Practic am "umplut" toata imaginea cu un fundal galben */
```

```
  $rosu=imagecolorallocate($imagine,255,0,0);
/* prin aceasta cream culoarea rosie */
  imagesetthickness($imagine,4);
/* prin aceasta stabilim grosimea implicita a liniilor desenate la 4 pixeli */
  imageline($imagine,10,10,389,239,$rosu);
/* si desenam o linie de culoare rosie*/
  imageellipse($imagine,200,125,200,200,$rosu);
/* iar apoi desenam un cerc cu centrul in centrul imaginii, de raza 100 */
  imagepng($imagine);
/*si, in fine, afisam imaginea respectiva in format png, trimitind-o direct in browser*/
  imagedestroy($imagine);
/* dezalocam resursa, pentru a nu ocupa memorie inutila */
?>
```

Iată rezultatul obținut prin încărcarea sa în browser:



Iată câteva dintre cele mai importante funcții care lucrează cu imagini:

• **`header(string)`** - are ca efect trimiterea unui header HTTP. În cazul nostru, al lucrului cu imagini, ne interesează să trimitem browser-ului informații despre **`mime-type`**-ul imaginii create. Astfel, valorile pe care le putem da string-ului, în funcție de tipul imaginii pe care o creăm, pot fi:

"**`Content-type: image/png`**" – pentru imaginile de tip **`png`**

"**`Content-type: image/jpeg`**" – pentru imaginile de tip **`jpg`**

"**`Content-type: image/gif`**" – pentru imaginile de tip **`gif`**

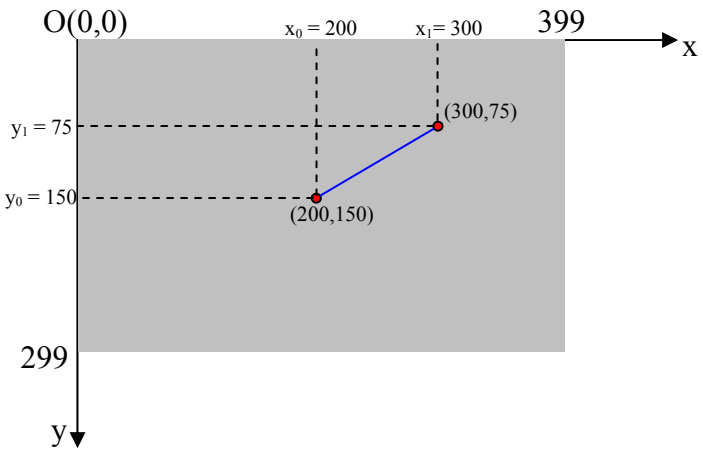
Este foarte important (în caz contrar se vor genera erori) ca această funcție să fie apelată înainte de trimiterea oricărui alt output din cadrul script-ului în care apare;

• **`imagecreatetruecolor(lățime, înălțime)`**- creează o resursă de tip imagine, pe 32 de biți (**`truecolor`**) având lățimea, respectiv înălțimea specificată. Rezultatul întors de această funcție trebuie obligatoriu atribuit unei variabile, prin intermediul căreia vom accesa în continuare imaginea.

Imaginea este de fapt o matrice de pixeli. Orice punct din imagine se va putea referi prin coordonatele sale carteziene (x, y). Originea sistemului de coordonate se găsește în colțul stânga-sus al imaginii (0,0) iar axa Oy este îndreptată în jos. Valorile posibile pentru **`x`** și **`y`** sunt numere întregi, cuprinse în intervalul **`0..lățime-1`**, respectiv **`0..înălțime-1`**

De exemplu, iată o reprezentare schematică a imaginii definite prin:

$imagine=imagecreatetruecolor(400,300);



Am pus în evidență în cadrul imaginii de mai sus punctele de coordonate (200,150) respectiv (300,75).

- **imagecolorallocate(resursă_imagine, roşu, verde, albastru)** - creează o resursă de tip culoare, asociată imaginii specificată prin resursa din primul parametru. Rezultatul întors de această funcție trebuie atribuit unei variabile, prin intermediul căreia vom accesa în continuare culoarea definită;

- **imagesetthickness(res_imag, thickness)** - stabileşte grosimea liniilor la **thickness** pixeli, atunci când se desenează linii, dreptunghiuri, poligoane;

- **imageline(resursă_imagine,x_0,y_0,x_1,y_1,resursă_culoare)** - desenează un segment de dreaptă, de culoarea specificată de **resursă_culoare**, în imaginea specificată de **resursă_imagine**, între punctele de coordonate **(x_0,y_0)** şi **(x_1,y_1)**;

- **imagedashedline(resursă_imagine,x_0,y_0,x_1,y_1,resursă_culoare)**

  - la fel ca **imageline**, doar că segmentul de dreaptă desenat este punctat. Pentru ca segmentul punctat să fie vizibil, trebuie ca **imagesetthickness** să seteze grosimea liniei la cel puțin 2 pixeli;

- **imageellipse(resursă_imagine,x_0,y_0,diam_x,diam_y,resursă_culoare)**

  - desenează o elipsă cu axe paralele cu axele de coordonate, având centrul în punctul de coordonate **(x_0,y_0)** şi diametrul orizontal dat de **diam_x** respectiv cel vertical dat de **diam_y**;

- **imagerectangle(res_imag,x_0,y_0,x_1,y_1,res_culoare)** – desenează un dreptunghi având colțurile diagonal opuse în punctele de coordonate **(x_0,y_0)** respectiv **(x_1,y_1)**, cu culoarea dată de resursa **res_culoare**;

- **imagefilledrectangle(res_imag,x_0,y_0,x_1,y_1,res_culoare)** – desenează un dreptunghi plin, cu colțurile diagonal opuse în punctele de coordonate **(x_0,y_0)** respectiv **(x_1,y_1)**, de culoarea dată de resursa **res_culoare**;

- **imagefilledellipse(res_imag,x_0,y_0,diam_x,diam_y,res_culoare)**

  - la fel ca **imageellipse**, doar că elipsa desenată este plină, având culoarea dată de **res_culoare**;

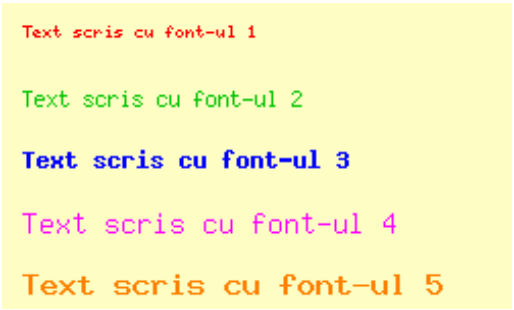
- **imagechar(res_imag,font,x_0,y_0,caracter,res_culoare)** – afişează în imagine caracterul **caracter**, începând de la coordonatele **(x_0,y_0)** în dreapta şi în jos, având culoarea dată de **res_culoare**. **font** poate fi unul dintre font-urile predefinite 1, 2, 3, 4, 5, sau orice resursă de tip font încărcat prin **imageloadfont()**;

- **imagestring(res_imag,font,x_0,y_0,caracter,res_culoare)** – afişează în imagine string-ul **string**, începând de la coordonatele **(x_0,y_0)** în dreapta şi în jos, având culoarea dată de **res_culoare**. **font** poate fi unul dintre font-urile predefinite 1, 2, 3, 4, 5, sau orice resursă de tip font încărcat prin **imageloadfont()**;

Exemplu: **ap1081.php**

```
<?php
  header("Content-type: image/png");
  $imagine=imagecreatetruecolor(250,150);
  $red=imagecolorallocate($imagine,255,0,0);
  $green=imagecolorallocate($imagine,0,196,0);
  $blue=imagecolorallocate($imagine,0,0,255);
  $magenta=imagecolorallocate($imagine,255,0,255);
  $orange=imagecolorallocate($imagine,255,128,0);
  $yellow=imagecolorallocate($imagine,255,255,196);
  imagefilledrectangle($imagine,0,0,249,149,$yellow);
  //afisam 5 stringuri, pentru a testa cum arata fiecare dintre cele
  //5 font-uri predefinite ale PHP-ului:
  imagestring($imagine,1,10,10,'Text scris cu font-ul 1',$red);
  imagestring($imagine,2,10,40,'Text scris cu font-ul 2',$green);
  imagestring($imagine,3,10,70,'Text scris cu font-ul 3',$blue);
  imagestring($imagine,4,10,100,'Text scris cu font-ul 4',$magenta);
  imagestring($imagine,5,10,130,'Text scris cu font-ul 5',$orange);
  imagepng($imagine);imagedestroy($imagine);
?>
```

Rezultatul afişat în browser arata în felul următor:



- **imagefill(res_imag,x_0,y_0,res_culoare)** – umple prin algoritmul "**flood fill**", pornind din punctul dat **(x_0,y_0)**, schimbând culoarea acestuia şi a tuturor punctelor conectate (din aproape în aproape) care au aceeaşi culoare cu cea existentă inițial la **(x_0,y_0)**, în noua culoare dată de **res_culoare**;

- **imagefilltoborder(res_imag,x_0,y_0,culoare_border, res_culoare)**

– umple prin algoritmul ”**flood fill**”, pornind din punctul dat **(x_0,y_0)**, schimbând culoarea tuturor punctelor conectate (indiferent ce culoare au) în noua culoare dată de **res_culoare**, până la întâlnirea culorii specificate prin parametrul **culoare_border**;

- **imagecopy(imag_dest,imag_sursa,x_{dest},y_{dest},x_{src},y_{src},width,height)**

– copiază din **imag_sursa**, porțiunea rectangulară care are colțul stânga sus la coordonatele **(x_{src},y_{src})** şi lungimea, respectiv înălțimea, date de **(width,height)** în **imag_dest**, începând de la coordonatele **(x_{dest},y_{dest})** în dreapta respectiv în jos;

- **imagecopyresized(imag_dest,imag_sursa,x_{dest},y_{dest},x_{src},y_{src},**
$width_{dest}$,$height_{dest}$,$width_{src}$,$height_{src}$)

– copiază din imag_sursa, porțiunea rectangulară care are colțul stânga-sus la coordonatele (**x_{src},y_{src}**) şi lungimea, respectiv înălțimea, date de **($width_{src}$,$height_{src}$)** în **imag_dest**, începând de la coordonatele **(x_{dest},y_{dest})**, redimensionând astfel încât noua lățime respectiv înălțime să fie **($width_{dest}$,$height_{dest}$)**;

- **imagerotate(res_imag,unghi,culoare_fundal)** – roteşte imaginea din **res_imag** cu unghiul **unghi** (specificat în grade !). Centrul rotației este centrul imaginii, iar imaginea rotită este redimensionată la scară, astfel încât întreaga imagine rotită să încapă: marginile nu sunt decupate. Porțiunile rămase neacoperite în urma rotației, se vor colora cu **culoare_fundal**;

- **imagesx(res_imag)** – întoarce lățimea imaginii (width);

- **imagesy(res_imag)** – întoarce înălțimea imaginii (height);

- **imagecreatefromgif('nume_fişier')** – creează şi întoarce o resursă de tip imagine, în care este încărcată imaginea de tip GIF din fişierul specificat de **'nume-fişier'**;

- **imagecreatefrompng('nume_fişier')** respectiv **imagecreatefromjpeg('nume_fişier')** funcționează analog, pentru imagini de tip PNG respectiv JPG;

- **imagegif(res_imagine)**, **imagepng(res_imagine)**, **imagejpeg(res_imagine)**

– produc afişarea în browser (deci trimiterea fluxului de date către acesta) a imaginii specificată de res_imagine. În funcție de tipul imaginii (GIF, PNG sau JPG) folosim varianta convenabilă a acestei funcții;

- **imagedestroy(res_imag)** – produce dezalocarea întregii memorii asociate imaginii reprezentată de **res_imag**;

• **`getimagesize('nume_fişier')`** – întoarce un array (şir) ce conţine informaţii despre imaginea din fişierul **`'nume_fişier'`**. Informaţiile se găsesc structurate astfel:

- la indicele 0 se găseşte lăţimea (width);

- la indicele 1 se găseşte înălţimea (height);

- la indicele 2 se găseşte o constantă care ne indică tipul imaginii (posibile valori pentru acest element sunt: IMAGETYPE_GIF, IMAGETYPE_JPEG, IMAGETYPE_PNG, etc.);

- la indicele 3 se găseşte un string de forma **`height="yyy" width="xxx"`**, pentru a putea fi folosit direct într-un tag <IMG....>

- la indicele **`mime`** se găseşte un string ce conţine tipul MIME corespunzător imaginii;

3.11. Upload de fişiere via PHP.

Prin acţiunea de upload, utilizatorul poate încărca, prin intermediul unei pagini web, un întreg fişier (indiferent că este un fişier binar sau un fişier text) pe server-ul pe care este stocată şi pagina respectivă.

La începuturile dezvoltării internetului, s-a pus desigur această problemă, a transferului de fişiere. Rezolvarea a fost găsită atunci prin implementarea unui protocol de transferare de fişiere între două calculatoare care sunt legate între ele. Este vorba de FTP (File Transfer Protocol), care deşi este o metodă eficientă de transfer, este greoi de folosit de un utilizator nespecializat (necesită specificarea adresei calculatorului la care ne conectăm, un nume de utilizator şi o parolă, precum şi un program specializat).

Avantajul adus de upload-ul prin intermediul unei pagini web constă în faptul că este uşor de folosit de către orice utilizator al internetului, cel care face operaţia având nevoie doar de câteva cunoştinţe minimale de utilizare a calculatorului.

Cea mai frecventă acţiune de upload din parte unui utilizator obişnuit este întâlnită la ataşarea unui fişier la un e-mail.

Care este principiul de funcţionare ?

În primul rând, trebuie conceput un **`form`** special, în care se vor insera unul sau mai multe elemente de tip **`<input type="file"...>`**. Acestea se prezintă sub forma unor textbox-uri în dreapta cărora este prezent un buton "Browse", ca în figura de mai sus. Un click fie în interiorul textbox-ului, fie pe butonul "Browse" va permite alegerea unui fişier de pe disc pentru a fi încărcat.

În etapa următoare, după ce utilizatorul apasă, tot în acel form, butonul de postare date, fişierul ales pentru upload va fi trimis către server, tot prin intermediul unui script (cel specificat în

cadrul atributului ”**action=...**” al formularului) şi copiat într-o locație temporară. Treaba programatorului PHP este ca el să copieze fişierul la locația sa definitivă.

Atenție ! Pentru ca upload-ul să funcționeze, atât directorul temporar în care este încărcat fişierul, cât şi directorul în care vom muta acest fişier trebuie să aibă drepturi de scriere pentru orice utilizator de pe Internet.

În mod implicit, directorul temporar al instalării XAMPP-ului are aceste drepturi stabilite de la instalarea întregului pachet. Dacă ele nu sunt corect setate, va trebui să le configurați manual. În tot cazul, pentru directorul în care veți muta fişierul, va trebui obligatoriu să le configurați.

Voi exemplifica modul în care se face acest lucru pentru directorul tmp al instalării XAMPP. Pentru directorul destinație, veți proceda analog.

Astfel, deschideți un Windows Explorer. Vă asigurați că opțiunea ”**Use simple file sharing (Recommended)**” **NU** este bifată (pentru a ajunge la această opțiune deschideți din meniul **Tools** submeniul **Folder Options** şi apoi accesați tab-ul **View**. În cadrul acestuia sunt mai multe opțiuni printre care şi cea de mai sus), după care mergeți pe directorul temporar al instalării xampp (în mod implicit C:\xampp\tmp), NU intrați în director, ci dați click dreapta pe el, alegând opțiunea **Properties**, apoi tab-ul **Security**. În acest tab, alegeți ”**Add**”, scrieți numele **network**, şi apoi alegeți din lista ce vi se deschide utilizatorul **network**. După ce l-ați ales, asigurați-vă că are drepturi de scriere în directorul **tmp** (bifați, în căsuțele de sub el, şi opțiunile **Modify** respectiv **Write**).

În cadrul script-ului PHP care se ocupă de preluarea fişierului încărcat, ne vom folosi de un masiv predefinit al limbajului, şi anume **$_FILES**, care ne va furniza date despre fişierul încărcat, şi anume (primul parametru al lui **$_FILES** este dat de numele **input**-ului de **type=”file”**):

- **$_FILES[nume_input]['name']** – ne întoarce numele şi extensia fişierului pe care l-am upload-at;
- **$_FILES[nume_input]['tmp_name']** – ne întoarce numele complet (cu tot cu calea) fişierului temporar care s-a creat în urma upload-ului. Atenție ! acest nume poate să fie complet diferit față de cel original.
- **$_FILES[nume_input]['type']** – ne întoarce tipul *mime* al fişierului (un string, de exemplu: *application/octet-stream* sau *image/gif*. Valoarea sa nu poate fi garantată ca fiind corectă;
- **$_FILES[nume_input]['size']** – ne întoarce dimensiunea (în octeți) a fişierului upload-at
- **$_FILES[nume_input]['error']** – ne întoarce codul de eroare al operației de upload asupra fişierului dat de **nume_input**. Dacă operația s-a încheiat cu succes, are valoarea 0.

Exemplu: Următoarele două script-uri realizează o aplicație prin care putem upload-a un fişier în acelaşi director în care se află şi sursele aplicației.

Primul fişier reprezintă un HTML obişnuit, ce conține doar form-ul prin care se poate face upload-ul fişierului, iar al doilea fişier reprezintă script-ul PHP care preia fişierul upload-at şi îl mută în directorul curent:

apl082.html

```
<form enctype="multipart/form-data" action="apl083.php" method="post">
<!-- este obligatoriu sa specificati atributul enctype="multipart/form-data", in caz
contrar nu va fi permisa upload-area de fisiere. De asemenea, remarcati atributul action,
care specifica numele scriptului care se va ocupa de preluarea fisierului upload-at -->
   <input type="hidden" name="MAX_FILE_SIZE" value="30000">
<!-- acest control de tip hidden permite stabilirea unei limite maxime a dimensiunii
fisierului care urmeaza a fi uploadat. Din pacate, acest parametru poate fi usor pacalit
din browser -->
   Alege un fisier pentru upload (sa aiba sub 30000 bytes):<br>
<!-- cu ajutorul acestui input type=”file” putem defini cimpul de tip upload din form -->
   <input type="file" name="fisier_incarcat"><br><br>
<!-- de valoarea atributului sau ’name’, in cazul nostru ”fisier_incarcat” ne vom folosi
pentru a-l manipula din cadrul codului PHP in care il vom prelua, in cazul nostru
apl083.php-->
   <input type="submit" value="Incarca fisierul">
</form>
```

apl083.php

```
<?php
   if($_FILES['fisier_incarcat']['error']!=0)
            echo 'Upload nereusit, a aparut o eroare';
   else
     {
       move_uploaded_file($_FILES['fisier_incarcat']['tmp_name'],
                          $_FILES['fisier_incarcat']['name']);
//cu ajutorul functiei move_uploaded_file mutam fisierul de la locatia sa temporara la
//locatia sa definitiva, care în cazul nostru este directorul curent. Parametrii acestei
//funcţii sunt sursa respectiv destinatia
       echo 'Fisierul a fost incarcat<br>';
       echo 'Dimensiunea sa este de ',$_FILES['fisier_incarcat']['size'],' octeti';
     }
?>
```

3.12. Variabile cookie.

Utilizarea lor a pornit din necesitatea păstrării anumitor setări (opțiuni) ale utilizatorului atunci când acesta intră pe un anumit site, pentru a nu mai fi nevoit sa le specifice la fiecare intrare.

Mecanismul care stă la baza acestei probleme se bazează pe memorarea, pe calculatorul vizitatorului unui anumit site, a unor informații sub forma unor mici fişiere text. Operația poate fi comandată de pe server şi tot de pe server se poate comanda citirea, actualizarea sau ştergerea acestor mici fişier, numite uzual, prin abuz de limbaj, ***variabile cookie***.

În PHP se poate lucra foarte uşor cu variabilele cookie.

- Pentru a crea o variabilă cookie se utilizează funcția:

setcookie(nume_variabilă, valoare, dată_expirare);

Exemplu: în instrucțiunea de mai jos este creată o variabilă **cookie** numită **limba_pref**. Variabila reține valoarea ”**romana**” şi expiră într-o oră:

```
setcookie(limba_pref,"romana",time()+3600);
```

Observații !

1) Nu pot exista mai mult de 20 de variabile **cookie**. Dacă se creează vreuna în plus, prima creată este ştearsă automat;

2) Pentru a şterge o variabilă **cookie** se creează o alta cu acelaşi nume, dar cu data de expirare înaintea celei curente (de exemplu, **time()-1**).

- Pentru a citi (recupera) valoarea unei variabile **cookie**, se utilizează conținutul unui masiv asociativ special, predefinit, al limbajului PHP, numit **$HTTP_COOKIE_VARS[..]**. Fiecare componentă a sa are ca indice numele unei variabile **cookie**, iar ca valoare, reține valoarea variabilei respective.

Exemplu: instrucțiunea de mai sus afişează valoarea variabilei **cookie** creată prin exemplul anterior:

```
echo $HTTP_COOKIE_VARS["limba_pref"];//afişează "romana"
```

Exemplu: Script-ul care urmează exemplifică modul în care se pot reține anumite informații pe care utilizatorul le-a tastat o dată. Apelat pentru prima dată, atunci când nu există variabila **cookie** numită "**loc**", se cere tastarea localității în care se află cel care vizitează pagina. Dacă acel vizitator reintră pe acel site, reapelând script-ul, acesta va identifica variabila **cookie** "**loc**", va prelua direct localitatea memorată şi, în loc ca utilizatorului să i se ceară din nou introducerea acesteia, i se va afişa direct un mesaj:

ap1084.php

```
<?php
  $loc=@$_POST['loc'];//testam mai intii daca am primit dintr-un form
  //postat tot din cadrul acestei pagini, o variabila numita loc,
  //cu valoarea careia trebuie sa cream variabila cookie
  if($loc!=NULL)//daca am primit
           setcookie("loc",$loc,time()+24*3600);//atunci setam variabila
                //cookie cu valoare primita, pentru o durata de 24 de ore
   else//pe cind, daca nu am primit, verificam daca variabila cookie este deja creata
      $loc=@$HTTP_COOKIE_VARS['loc'];
   if($loc!=NULL)//deci daca este creata
     {//dam utilizatorului un mesaj:
         echo "Localitatea curenta este: ",$loc;
          //si ii permitem sa schimbe aceasta localitate, daca doreste
          echo '<form action="ap1084.php" method="post">';
          echo '<p align="right">';
          echo 'Schimba localitatea:';
          echo '<input type="text" name="loc"><br>';
          echo '<input type="submit" value="Schimba localitatea"></p></form>';
       }
   else//in caz contrar, cream formularul care permite introducerea localitatii
     {
          echo '<form action="ap1084.php" method="post">';
          echo 'Introdu localitatea<br><br>';
          echo '<input type="text" name="loc"><br><br>';
          echo '<input type="submit" value="Salveaza"></form>';
       }
?>
```

3.13. Exploatarea bazelor de date MySQL prin intermediul limbajului PHP.

3.13.1. Introducere în MySQL.

În ultimii ani, utilizarea bazelor de date pe Internet a luat o amploare deosebită. Există o mulțime de aplicații, extrem de utile, care le utilizează, cum ar fi: aplicații de contorizare, comerț electronic, vot electronic, aplicații de comunicare.

De unde SQL ? În domeniul creării şi utilizării bazelor de date relaționale s-a impus necesitatea existenței unui limbaj standard care să permită efectuarea acestor operații. Astfel, a apărut SQL – Structured Query Language. Limbajul a fost supervizat de comisia de standardizare ANSI (American National Standards Institute), motiv pentru care se mai numeşte şi ANSI SQL.

SQL nu este un limbaj de firmă, ci el este implementat de o mulțime de SGBD-uri consacrate, cum ar fi Microsoft Access, Oracle, Microsoft SQL Server şi, bineînțeles, MySQL.

MySQL este un limbaj specializat pentru gestiunea bazelor de date relaționale pe Internet, având la bază limbajul SQL. MySQL gestionează baze de date care se găsesc pe un server. Baza de date poate fi foarte lesne exploatată prin intermediul limbajului PHP, dar şi cu alte limbaje (de exemplu Java).

3.13.2. Testarea instalării MySQL. Configurarea bazei de date.

După cum am anunțat în capitolul 3.2., suportul MySQL a fost deja instalat, în cadrul pachetului XAMPP, deodată cu limbajul PHP.

Tot suportul software al bazei de date MySQL se găseşte în directorul **`C:\XAMPP\MYSQL`**. În subdirectorul **`BIN`** al acestuia se găsesc o serie de programe utilizare, care permit configurarea şi operarea imediată cu serverul de baze de date. Utilitarul pe care o să-l utilizăm cel mai des este **`mysql.exe`**. Pentru a-l putea accesa din orice altă locație a calculatorului v-ați afla, este recomandat să adăugați calea spre el (**`C:\XAMPP\MYSQL\BIN`**) în variabila **`PATH`** a sistemului. Acest lucru se face executând un click dreapta pe **`My Computer`**, alegând apoi tab-ul **`Advanced`**, apoi butonul **`Environment Variabies`**. Din fereastra care se deschide, derulați în partea de jos (secțiunea **`System Variables`**) până întâlniți variabila **`PATH`**. Dați dublu click pe ea (sau apăsați butonul **`Edit`** de dedesubt) şi, la sfârşitul şirului de caractere deja existent acolo, adăugați caracterul ”**`;`**” urmat de calea ”**`C:\XAMPP\MYSQL\BIN`**”.

Din acest moment, utilitarul **`mysql`** se poate rula din orice locație (de exemplu direct din **`Start + Run`**).

Vom configura în continuare serverul de baze de date. MySQL fiind un server de baze de date partajat, accesul se face pe baza unui nume de utilizator (**user**) şi a unei parole (**password**).

Pentru moment, este configurat doar utilizatorul root (cu drepturi depline asupra bazei de date) şi NU are parola.

Pentru a va conecta, lansaţi dintr-un **Command Prompt** sau din **Start + Run** următoarea comandă: **mysql -u root -p**

Atunci când sunteţi invitaţi să introduceţi parola (prin **Enter password:**) trebuie să daţi Enter. În cazul în care conectarea a reuşit, veţi primi un mesaj de întâmpinare, iar prompt-ul se va schimba în : **mysql>...** (pentru a vă deconecta de la server, în dreptul acestui prompt trebuie să scrieţi comanda **quit** după care daţi Enter).

Din acest sunteţi conectaţi la baza de date. În continuare vom da câteva comenzi pentru a configura serverul, baza de date asupra căreia vom lucra, şi un utilizator pe care îl vom folosi pentru a lucra în mod curent (deoarece este complet nerecomandat să folosim utilizatorul root pentru a lucra în mod curent la baza de date).

În primul rând, vom şterge toţi utilizatorii predefiniţi, în afară de root, iar apoi vom crea o bază de date în care vom lucra şi un utilizator cu drepturi de utilizare asupra acestei baze de date. Atenţie să nu uitaţi caracterul ”;” după fiecare comandă executată. De asemenea, după fiecare comandă daţi un Enter:

mysql>use mysql; prin această comandă intrăm în baza de date administrativă, cu numele mysql

mysql>delete from user where user!='root' or host!='localhost';

prin această comandă ştergem toţi utilizatorii predefiniţi, lăsând utilizatorul root cu drepturi de conectare de pe maşina locală

mysql>update user set password=password('parolanouă') where user='root';

prin această comandă schimbăm parola utilizatorului root, de la parola vidă (pe care o avea de la instalare) cu noua parolă (evident, şirul de caractere **parolanouă** îl înlocuiţi cu noua parolă)

mysql>create database lucru;

prin această comandă creăm o bază de date cu numele ”lucru”, în care vom lucra în continuare

mysql>grant all on lucru.* to 'utilizator'@'localhost' identified by 'parola';

prin această comandă creăm utilizatorul cu numele ’utilizator’, cu parola ’parola’ (evident, le veţi schimba după bunul plac) ce se poate conecta de pe maşina locală, şi are drepturi de lucru doar în baza de date ’lucru’

mysql>flush privileges;

prin această comandă facem ca toate modificările efectuate până în momentul de faţă să devină efective şi operative.

mysql>quit;

şi ieşim din monitorul mysql

Putem testa imediat instalarea, prin conectarea la serverul mysql cu noul utilizator creat (lansaţi această comandă fie din **Command Prompt** fie din **Start+Run**) :

```
mysql -u utilizator -p
```

În momentul în care vi se cere parola, introduceți, evident, parola definită mai sus.

După conectarea la MySQL, pentru a stabili ca punct de lucru baza de date asociată acestui utilizator, veți da comanda:

```
mysql>use lucru;
```

după care veți putea testa toate comenzile ce urmează să fie testate în paragrafele următoare.

3.13.3. Crearea unei baze de date.

Această operație nu poate fi făcută de către orice utilizator, ci doar de către `root` sau de anumiți utilizatori cu drepturi speciale.

- Comanda pentru crearea unei baze de date este:

```
create database nume_bază_date;
```

În urma executării sale este creată baza de date cu numele indicat. Din moment ce fiecare bază de date este stocată pe disc, într-un anumit director, în fapt se va crea un subdirector cu acelaşi nume cu baza de date, în subdirectorul `data` al folder-ului instalării MySQL (în cazul XAMPP, acest folder este `C:\XAMPP\MYSQL\DATA`. Dacă există deja o bază de date cu numele specificat, se va refuza crearea uneia noi.

- Dacă se doreşte lucrul cu o anumită bază de date, se va da comanda următoare:

```
use nume_bază_date;
```

- Pentru ştergerea unei baze de date se utilizează comanda:

```
drop database nume_bază_date;
```

- Dacă dorim o listă a bazelor de date existente (sau, în fine, în funcție de utilizator, a bazelor de date la care avem acces) se utilizează comanda:

```
show databases;
```

3.13.4. Tabele.

Într-o tabelă coloanele sunt identificabile prin nume, iar rândurile, prin valorile pe care le memorează. Toate datele dintr-o coloană au acelaşi tip. O tabelă are un număr specificat de coloane, însă are un număr nespecificat de rânduri. Uneori, când ne referim la un rând, folosim şi termenul de înregistrare, iar atunci când ne referim la data din rând, situată într-o anumită coloană, folosim şi termenul de câmp.

- Instrucțiunea prin care se poate crea o tabelă este prezentată mai jos; ceea ce este trecut între paranteze drepte reprezintă clauze, atribute sau bucăți de instrucțiune care, din punct de vedere sintactic (în funcție de fiecare caz) pot să nu apară:

```
create table nume_tabelă
      ( nume_coloană1 tip_date [specificatori],
        nume_coloană2 tip_date [specificatori],
        ...
        nume_coloanăn tip_date [specificatori]
      );
```

Specificatorii se referă la cheia primară, valori distincte, valori implicite, autoincrementare, dacă printre valorile memorate în tabelă se poate găsi sau nu valoarea NULL. Toate acestea le vom trata separat, în alt paragraf.

Exemplu: pentru a crea o tabelă cu 3 coloane, în care vom reține indicativul fiecărui județ (ca pe plăcuțele de la mașini), numele județului (un şir de cel mult 30 de caractere) şi numărul său de locuitori (un număr întreg de cel mult 7 cifre) folosim următoarea comandă:

```
create table judete (ind char(2),nume char(30),nrloc int(7));
```

• Adăugarea unui nou rând într-o tabelă se face prin instrucțiunea de mai jos, în care se introduc, pe rând, valori pentru toate câmpurile unei linii, în ordinea în care au fost declarate coloanele la crearea tabelei:

```
insert into nume_tabelă values(valoare1,valoare2,...,valoaren);
```

Exemplu: pentru a insera două linii în tabela creată în exemplul anterior:

```
insert into judete values('CJ','Cluj',702755);
insert into judete values('BV','Brasov',589028);
```

În practica utilizării bazelor de date, instrucțiunea anterioară este considerată ca generatoare de erori, deoarece fie se poate greşi ordinea de introducere a datelor, fie pe parcursul dezvoltării se inserează coloane noi între cele deja existente, iar efectul ar fi acela că tabela va conține date eronate. Din acest motiv este preferată forma următoare a instrucțiunii, deşi este necesar să scriem mai mult:

```
insert into nume_tabelă(nume_coloană1,nume_coloană2,...,nume_coloanăk)
                        values(valoare1,valoare2,...,valoarek);
```

Se pot omite coloane (caz în care ele vor primi automat valori NULL) şi bineînțeles că putem scrie coloanele în orice ordine dorim.

Exemplu: mai adăugăm două linii la tabela anterioară: una în care nu completăm numărul de locuitori şi alta în care trecem toate cele 3 valori, însă în altă ordine decât cea în care le-am creat:

```
insert into judete(nrloc,ind,nume) values(510110,'MM','Maramures');
insert into judete(ind,nume) values('CL','Calarasi');
```

• Pentru a afişa întreaga tabelă, utilizăm comanda:

```
select * from nume_tabelă;
```

Exemplu: pe tabela creată anterior, dacă executăm:

```
select * from judete;
```

se va afişa în următorul format:

```
+------+-----------+--------+
| ind  | nume      | nrloc  |
+------+-----------+--------+
| CJ   | Cluj      | 702755 |
| BV   | Brasov    | 589028 |
| CL   | Calarasi  |   NULL |
| MM   | Maramures | 510110 |
+------+-----------+--------+
4 rows in set (0.02 sec)
```

• Dacă dorim să afişăm doar anumite coloane ale tabelei, în ordinea pe care o dorim, folosim forma de mai jos a instrucţiunii:

```
select nume_coloana1,...,nume_coloanak from tabelă;
```

Se observă faptul că, atunci când afişăm tabela, în antetul (capul de tabel) său se trece numele coloanelor acesteia. Dacă dorim ca în antet să figureze alt nume pentru o coloană, atunci, în instrucţiunea de mai sus, în loc de a scrie doar numele coloanei, vom scrie numele coloanei urmat de cuvântul cheie **AS** şi de numele care dorim să fie afişat în antet. Astfel, prin cuvântul cheie "**AS**", am definit alias-uri pentru numele coloanelor.

Exemplu: pentru tabela din exemplele anterioare, am putea scrie următoarea comandă:

```
select ind as indicativ,
       nume as "nume judet",
       nrloc as "numar de locuitori"
       from judete;
```

Rezultatul constă în afişarea sub următorul format:

```
+-----------+------------+--------------------+
| indicativ | nume judet | numar de locuitori |
+-----------+------------+--------------------+
| BV        | Brasov     |             589028 |
| CJ        | Cluj       |             702755 |
| CL        | Calarasi   |               NULL |
| MM        | Maramures  |             510110 |
+-----------+------------+--------------------+
4 rows in set (0.00 sec)
```

• Pentru listarea numelor tuturor tabelelor din baza de date, folosim comanda următoare:

```
show tables [from nume_bază_date];
```

Dacă nu specificăm clauza **[from nume_bază_date]** se vor afişa toate tabelele din baza de date curentă (cea către care am dat use);

• Pentru a afişa o descriere detaliată a unei tabele (numele coloanelor, tipurile lor) se foloseşte una dintre comenzile (ele fac acelaşi lucru):

```
describe nume_tabelă;
show columns from nume_tabelă;
```

3.13.5. Tipuri de date în MySQL.

Tipurile principale de date ale MySQL sunt:

a) şiruri de caractere;

b) numerice;

c) dată, oră

În continuare, vom prezenta fiecare tip de dată în parte:

a) Şiruri de caractere:

- **char[(n)]** – reține un şir de caractere de lungime n (fixă). În caz că n nu este precizat, reține un singur caracter. Ocupă n octeți;
- **varchar[(n)]** – reține un şir de cel mult 255 de caractere. Ocupă n+1 octeți;
- **tinytext[(n)]** – este echivalent cu **varchar[(n)]**;
- **text[(n)]** – reține un şir de cel mult 65535 caractere. Ocupă n+2 octeți;
- **mediumtext[(n)]** – reține un şir de cel mult 16.777.215 caractere. Ocupă n+3 octeți;
- **longtext[(n)]** – reține un şir de cel mult 4.294.967.295 caractere. Ocupă n+4 octeți;
- **enum** – un câmp care poate reține un singur şir de caractere dintr-un vector de şiruri de caractere predefinit de către utilizator la crearea tabelei. De altfel, tabela va memora vectorul de şiruri de caractere, iar în acest câmp se va reține doar indicele elementului corespunzător din vector. Vezi exemplele de după;
- **set** – la fel ca la enum, doar că un câmp de acest tip poate reține unul sau mai multe şiruri de caractere din vectorul predefinit de către utilizator. Vezi exemplele de după.

Exemplu (pentru tipul enum): creăm o tabelă în care trecem câțiva pictori şi curentele artistice pe care le reprezintă.

```
create table pictori(nume text,
       curent enum('impresionism','postimpresionism','suprarealism','art nouveau'));
insert into pictori values('Gustave Klimt','art nouveau');
insert into pictori values('Vincent Van Gogh','postimpresionism');
insert into pictori values('Alphonse Mucha','art nouveau');
insert into pictori values('Auguste Renoir','impresionism');
insert into pictori values('Rene Magritte','suprarealism');
insert into pictori values('Tiziano Vecellio','renastere');
```

A se remarca faptul că după executarea ultimei comenzi apare o avertizare (warning). Acest lucru se întâmplă deoarece ultimul curent introdus, 'renastere' nu apare printre stringurile din enum, permise pentru acest câmp. Totuşi, în urma execuției sale, pictorul cu numele "Tiziano Vecellio" va fi trecut în tabelă, însă în dreptul curentului său va fi trecut şirul vid:

```
+------------------+------------------+
| nume             | curent           |
+------------------+------------------+
| Gustave Klimt    | art nouveau      |
| Vincent Van Gogh | postimpresionism |
| Alphonse Mucha   | art nouveau      |
| Auguste Renoir   | impresionism     |
| Rene Magritte    | suprarealism     |
| Tiziano Vecellio |                  |
+------------------+------------------+
```

Exemplu (pentru tipul set): creăm o tabelă în care trecem numele câtorva persoane şi hobby-urile lor favorite, dintre muzică, sport şi desen. Vom încerca şi de această dată să punem valori invalide. Efectul este asemănător celui de la enum: datele respective vor fi ignorate:

```
create table persoane (nume text,
                  hobby set('muzica','desen','sport'));
insert into persoane values('Ion MARIN','muzica,desen');
insert into persoane values('Ion TIRIAC','sport,desen');
insert into persoane values('Nina CHIRIAC','muzica');
insert into persoane values('Ion BETEA','sport,fumat');
```

```
+--------------+--------------+
| nume         | hobby        |
+--------------+--------------+
| Ion MARIN    | muzica,desen |
| Ion TIRIAC   | desen,sport  |
| Nina CHIRIAC | muzica       |
| Ion BETEA    | sport        |
+--------------+--------------+
```

b) Tipuri de date numerice:

- **`tinyint[(n)]`** – ocupă 1 octet. Reține nr. întregi cuprinse între -128..127. Dacă este ***urmat*** de cuvântul **`unsigned`** reține numere naturale cuprinse între 0..255;
- **`smallint[(n)]`** – ocupă 2 octeți. Reține nr. întregi cuprinse între -32768..32767. Dacă este ***urmat*** de cuvântul **`unsigned`** reține numere naturale cuprinse între 0..65535;
- **`mediumint[(n)]`** – ocupă 3 octeți. Reține nr. întregi cuprinse între -8388608..8388607. Dacă este ***urmat*** de cuvântul **`unsigned`** reține numere naturale cuprinse între 0..16777215;
- **`int[(n)]`** – ocupă 4 octeți. Reține nr. întregi cuprinse între -2147843648..2147843647. Dacă este ***urmat*** de cuvântul **`unsigned`** reține numere naturale cuprinse între 0..4294967295;
- **`bigint[(n)]`** – ocupă 8 octeți. Reține nr. întregi cuprinse între -9223372036854775808 şi 9223372036854775809. Dacă este ***urmat*** de cuvântul **`unsigned`** reține numere naturale cuprinse între 0..18446744073709551615;
- **`float`** – ocupă 4 octeți – este echivalentul tipului cu acelaşi nume din C/C++;
- **`double`** – ocupă 8 octeți – este echivalentul tipului cu acelaşi nume din C/C++;
- **`decimal(n,d)`** – numărul este stocat sub formă de şir de caractere. Parametrul **`n`** reprezintă numărul de cifre nenule aflate înaintea virgulei (cu tot cu semnul '**`-`**' pentru numerele negative, dacă este cazul) iar **`d`** reprezintă numărul de zecimale.

c) Tipuri pentru dată şi oră:

- **`year`** – un câmp de acest tip reține un an calendaristic. Aceştia se introduc ca şi şir de caractere;
- **`time`** – un câmp de acest tip reține o anumită oră dintr-o zi. Se introduce sub forma unui şir de caractere, de forma '**`hh:mm:ss`**';
- **`date`** – un câmp de acest tip reține o anumită dată calendaristică. Se introduce sub forma unui şir de caractere, de forma '**`yyyy-mm-dd`**';
- **`datetime`** – un câmp de acest tip reține o anumită dată calendaristică şi o anumită oră. Se introduce sub forma unui şir de caractere, de forma '**`yyyy-mm-dd hh:mm:ss`**';

3.13.6. Operatori utilizați în MySQL. Variabile.

Voi prezenta mai întâi principalii operatori din MySQL, în ordinea crescătoare a priorității lor. Utilitatea operatorilor şi funcționalitatea acestora va fi prezentată în paragrafele următoare:

1. **`||`**, **`or`**, **`xor`**
2. **`&&`**, **`and`**
3. **`between`**, **`case`** **`when`**, **`then`**, **`else`**

4. `=`, `>=`, `>`, `<=`, `<`, `!=`, `<>`, `is`, `like`, `in`

5. `|`

6. `&`

7. `<<`, `>>`

8. `+`, `-` (operatori binari)

9. `*`, `/`, `div`, `%`, `mod`

10. `^`

11. `+`, `-` (operatori unari)

12. `!`, `not`

Pentru a testa un anumit operator, puteți utiliza instrucțiunea select.

De exemplu, dacă dați comanda `select 5+7;` se va afișa un rezultat ca în imaginea alăturată:

```
+-----+
| 5+7 |
+-----+
|  12 |
+-----+
```

Observație importantă: În MySQL există o valoare specială numită `Null`. Semnificația ei este de ”valoare necunoscută”. Rețineți faptul că, dacă `Null` este un operand al oricărei expresii, rezultatul oricărei operații care se efectuează cu `Null` este `Null`.

Operatorii se împart, la rândul lor, în mai multe grupe. Acestea sunt prezentate în continuare:

a) Operatori aritmetici:

Acționează asupra tipurilor numerice şi furnizează o valoare de tip numeric:

- `+` – adunare (Ex: `2+3` are ca rezultat valoarea 5);
- `-` – scădere (Ex: `2-3` are ca rezultat valoarea -1);
- `*` – înmulțire (Ex: `2*3` are ca rezultat valoarea 6);
- `/` – împărțire cu zecimale (Ex: `5/4` are ca rezultat valoarea 1.25);
- `div` – împărțire cu zecimale (Ex: `15 div 4` are ca rezultat valoarea 3);
- `mod` şi `%` – împărțire cu zecimale (Ex: `14 mod 4` sau `14%4` are ca rezultat valoarea 3);
- `-` şi `+` – operatorii unari plus şi minus (Ex: `--4` are ca rezultat valoarea 4);

b) Operatori de comparare (sau relaționali):

Permite compararea a două valori numerice sau a două şiruri de caractere. Şirurile de caractere se compară lexicografic şi nu se face distincție între literele mari şi literele mici. Rezultatul este 1 pentru adevărat şi 0 pentru fals.

- `<` – mai mic (Ex: `2<3` are ca rezultat valoarea 1);
- `<=` – mai mic sau egal (Ex: `3<=3` are ca rezultat valoarea 1);
- `>` – mai mare (Ex: `2>3` are ca rezultat valoarea 0);

- `>=` – mai mare sau egal (Ex: `2>=3` are ca rezultat valoarea 0);
- `=` – egalitate (Ex: `2=3` are ca rezultat valoarea 0);
- `<>` sau `!=` – diferit (Ex: `2<>3` sau `2!=3` are ca rezultat valoarea 1);

`Observație:` se pot compara, de altfel, cu aceşti operatori, şi date de tipul `time`, `date`. În fapt, o astfel de comparare este tot una lexicografică.

c) Operatori logici:

Analog limbajului C/C++, în MySQL se consideră două valori logice: 0 joacă rolul lui `false`, iar orice valoare diferită de 0 joacă rolul lui `true`.

- `||` sau `or` – sau-ul logic (este 0 doar când ambii operanzi sunt 0, în rest este 1);
- `&&` sau `and` – şi-ul logic (este 1 doar când ambii operanzi sunt nenuli, în rest este 0);
- `not` – negație (negația lui 0 este 1, iar negația lui 1 este 0);
- `xor` – sau-ul exclusiv (este 0 când ambii operanzi sunt fie nuli, fie nenuli, şi este 1 în rest);

d) Operatori logici pe biți:

Se aplică tuturor tipurilor întregi şi acționează asupra tuturor biților aflați pe poziții corespondente.

- `|` – sau-ul pe biți;
- `&` – şi-ul pe biți;
- `^` – sau-ul exclusiv pe biți;
- `~` – negația pe biți;

e) Operatori de deplasare pe biți:

Se aplică tuturor tipurilor întregi, deplasând biții reprezentării binare:

- `<<` – deplasare la stânga: `a<<b` deplasează cu `b` poziții la stânga biții lui `a`;
- `>>` – deplasare la dreapta: `a>>b` deplasează cu `b` poziții la dreapta biții lui `a`;

f) Operatori specifici pentru MySQL:

- `is null`, `is not null` – testează dacă o valoare este sau nu `null` (sunt singurii operatori care testează acest lucru! Atenție, compararea cu `null` întoarce, conform unei observații anterioare, TOT TIMPUL valoarea `null`, indiferent că se compară `null`-ul tot cu `null` sau cu altă valoare);
- `in`, `not in` – testează dacă o valoare aparține sau nu unei mulțimi

(ex: `1 in (1,2,3,4)` are valaorea 1;

`5 in (1,2,3,4)` are valoarea 0;

`'cici' in ('cici', 'mimi', 'lola')` are valoarea 1);

• **like, not like** – testează dacă un şir de caractere are o anumită formă: dacă este prefixat respectiv postfixat sau nu de un anumit subşir, dacă acesta conţine un anumit subşir. Forma sub care se utilizează este **string like şablon** respectiv **string not like sablon**. Şablon este tot un string, în care se folosesc următoarele caractere speciale: '**%**' pentru un număr neprecizat de caractere necunoscute, respectiv '**_**' pentru un singur caracter neprecizat.

(ex: **'cici' like 'ci%'** – are valoarea 1;

'lola' like 'la%' – are valoarea 0;

'mimi' not like 'me_i' – are valoarea 1);

• **between min and max** – testează dacă o valoare se găseşte în intervalul închis cu capetele **min**, respectiv **max**. (Ex: **1 between 0 and 4** – are valoarea 1);

• **case .. when .. then .. else ..** – are două forme sub care se poate aplica:

- forma 1:

```
case v
    when v1 then val1
    . . .
    when vn then valn
    else valn+1
end
```

Se evaluează v, şi dacă se produce vreuna dintre valorile v_1 .. v_n se va întoarce val_k corespunzătoare, iar dacă nu se va întoarce val_{n+1}

- forma 2:

```
case
    when cond1 then val1
    . . .
    when condn then valn
    else valn+1
end
```

Se evaluează condiţiile în ordinea scrierii. Prima care este adevărată va întoarce valoarea val_k corespunzătoare. Dacă nici una nu este advărată, se va întoarce val_{n+1}

Exemple:

1)
```
case cif
     when 1 then 'unu'
     when 2 then 'doi'
     when 3 then 'trei'
     else 'nu stiu sa numar decit pina la 3'
end
```

2)
```
case
     when a<0 then 'valoare negativă'
     when a=0 then 'valoare nulă'
     else 'valoare strict pozitivă'
end
```

g) Variabile în MySQL:

În MySQL variabilele se specifică prin `@identificator`, unde identificator respectă aceleaşi reguli sintactice ca şi în alte limbaje de programare (să fie format doar din litere, cifre şi '_' şi să nu înceapă cu o cifră). Atribuirea unei valori către o variabilă se face cu operatorul ':=' şi, la fel ca în C++, atribuirea poate juca şi rolul unei expresii, care întoarce valoarea atribuită.

Exemplu:

```
select @c:=(@a:=4)+(@b:=5);   atribuie atât valoarea 4 variabilei @a, valoarea 5
                               variabilei @b cât şi valoarea 9 variabilei @c
select @a;     pentru verificare, afişăm apoi valorile
select @b;     celor trei variabile
select @c;
```

3.13.7. Funcţii predefinite în MySQL.

a) Câteva funcţii matematice:

- `abs(x)` – modulul lui x;
- `ceil(x)` – cel mai mic întreg mai mare sau egal cu x;
- `floor(x)` – cel mai mare întreg mai mic sau egal cu x (partea întreagă matematică);
- `exp(x)` – e^x;
- `log(b,x)` – $\log_b x$;
- `ln(x)` – ln x;
- `pi()` – π;
- `pow(x,y)` – x^y;
- `round(x)` – cel mai apropiat întreg de x (rotunjire la întreg);
- `sin(x)`, `cos(x)` – sinusul şi cosinusul unghiului x. Unghiul trebuie dat în radiani;
- `sign(x)` – semnul lui x (-1 pt. nr. negativ, 0 pt. 0, 1 pt. număr pozitiv);
- `sqrt(x)` – radicalul (rădăcina pătrată) lui x;

b) Câteva funcţii care lucrează asupra şirurilor de caractere:

- `length(x)` – lungimea (nr. de caractere) a şirului x;
- `concat(x1,x2,...)` – concatenează şirurile de caractere pe care le primeşte ca parametri. Atenţie ! în MySQL NU există un operator de concatenare (ca în PHP, spre exemplu);
- `instr(x,y)` – Caută dacă y este subşir al lui x. Dacă este întoarce indicele de început (primul caracter având indicele 1) iar dacă nu este, întoarce 0;
- `substring(x,poz,lung)` – întoarce subşirul de caractere din `x` care începe pe poziţia `poz` şi are lungimea `lung`. Dacă `lung` este omis, întoarce de la poziţia `poz` până la sfârşit;
- `rtrim(x)` – întoarce şirul obţinut din x prin eliminarea spaţiilor inutile din dreapta;

- **ltrim(x)** – întoarce şirul obţinut din x prin eliminarea spaţiilor inutile din stânga;
- **trim(x)** – întoarce şirul obţinut din x prin eliminarea spaţiilor inutile atât din dreapta cât şi din stânga;
- **upper(x)** – întoarce şirul obţinut prin convertirea tuturor literelor mici la litere mari;
- **lower(x)** – întoarce şirul obţinut prin convertirea tuturor literelor mari la litere mici;
- **find_in_set(x,string)** – întoarce indicele apariţiei şirului x în şirul de entităţi separate prin virgule din cadrul lui string. Ex: **find_in_set('mimi','cici,mimi,lola')** va întoarce valoarea 2.
- **format(x,d)** – converteşte valoarea reală la un string cu d zecimale. Dacă este cazul, la ultima zecimală din string se face rotunjire. Ex: **format(5.678,2)** va întoarce stringul '5.68';
- **strcmp(x,y)** – compară lexicografic şirurile de caractere x şi y, întorcând -1 dacă **x<y**, 0 dacă **x=y**, 1 dacă **x>y**. Nu face distincţie între literele mari şi cele mici;

c) Câteva funcţii care lucrează asupra datei şi orei:

- **now()** – întoarce data şi ora curentă sub forma **yyyy-mm-dd hh:mm:ss**;
- **day(d)** – întoarce numărul zilei din data pe care o primeşte ca parametru;
- **month(d)** – întoarce numărul lunii din data pe care o primeşte ca parametru;
- **year(d)** – întoarce numărul anului din data pe care o primeşte ca parametru;
- **time(d)** – întoarce timpul (sub forma **hh:mm:ss**) extras din data+timpul pe care o primeşte ca parametru;
- **hour(d)** – întoarce ora din parametrul său. Acesta poate fi de tip **date** sau **datetime**;
- **minute(d)** – ca mai sus, întoarce minutul din parametrul său;
- **second(d)** – ca mai sus, întoarce secunda din parametrul său;
- **datediff(x,y)** – calculează diferenţa, în zile, dintre datele x şi y;
- **date_add(x,interval nr tip)** – adună la data **x** un număr **nr** de zile, luni sau ani, în funcţie de valoarea parametrului **tip**. Cuvântul "**interval**" este un cuvânt rezervat, trebuind scris ca atare. Parametrul tip poate avea una dintre valorile **day**, **month** respectiv **year**. Dacă se doreşte o scădere, fie folosim valori negative pentru **x**, fie folosim funcţia pe care o prezentăm imediat în continuare.

Ex: **date_add('2009-01-14',interval 18 day)** va întoarce valoarea '**2009-02-01**';

date_add('2009-01-14',interval -14 day) va întoarce valoarea '**2008-12-31**';

- **date_sub(x,interval nr tip)** – analog, produce o scădere;

d) Funcții specifice MySQL:

- **if(exp$_1$,exp$_2$,exp$_3$)** – dacă **exp$_1$** este nenulă (adevărată) funcția va întoarce **exp$_2$**, în caz contrar întorcând **exp$_3$**.

Observație: Dacă exp$_1$ este de tip real, va fi convertită la întreg (prin eliminarea zecimalelor). De exemplu, dacă este 0.87, se va converti la 0, deci s-ar putea ca rezultatul să nu fie cel scontat.

Ex: **if(a>=0,"pozitiv nestrict","negativ strict")** – va întoarce, după caz, una dintre valorile de tip şir de caractere;

- **ifnull(exp$_1$,exp$_2$)** – dacă exp$_1$ este nenulă, o întoarce chiar pe ea. Dacă nu, o întoarce pe exp$_2$.

Ex: **ifnull('cici','mimi')** – întoarce 'cici';

ifnull(null,'mimi') – întoarce 'mimi';

3.13.8. Coloane calculate prin intermediul unei interogări.

Am prezentat ceva mai înainte comanda **select**. După cum am văzut, această comandă permite vizualizarea informației dintr-o anumită tabelă a bazei de date.

Sintaxa generală a comenzii este mult mai largă decât cele prezentate în paragraful 3.13.4. Vom descoperi, pe parcurs, o serie de funcționalități ale sale.

Pentru moment, rețineți faptul că operația de consultare a datelor unei tabele dintr-o bază de date (mai precis, tot ceea ce realizează comanda select) se numeşte **interogare**.

O regulă elementară în crearea şi exploatarea unei baze de date impune ca aceasta să nu conțină informație redundantă, adică informație care se poate deduce pe baza datelor deja existente. Nerespectarea ei ar conduce la baze de date care ocupă mult spațiu, şi în care, din această cauză, prelucrările devin mai lente.

Vom învăța să afişăm date (coloane) care rezultă prin calcule, pe baza celor deja existente.

Acest lucru se realizează prin utilizarea unei expresii în cadrul comenzii select. În acest fel, interogarea va afişa o nouă coloană, având ca antet expresia respectivă şi pe post de conținut valoarea expresiei, aplicată pentru fiecare dintre liniile tabelei.

Dacă numele din antet nu ne convine, am învățat deja să-l schimbăm, cu ajutorul clauzei **as**, care, după am văzut, creează de fapt un alias pentru acea coloană.

Ex: Într-o tabelă, cu numele **prod**, având informații despre nişte produse dintr-un magazin, avem reținute, pentru fiecare produs, denumirea, costul unitar şi numărul de bucăți din produsul respectiv. Coloanele deja existente au numele '**den**', '**cost_u**' şi '**cant**'. Pentru a afişa pentru fiecare produs toate aceste date, plus costul total, vom folosi următoarea interogare:

```
select den,cost_u,cant,cost_u,cost_u*cant as cost_tot from prod;
```

Iată cum vi se va afişa rezultatul acestei interogări:

```
+----------+--------+------+--------+----------+
| den      | cost_u | cant | cost_u | cost_tot |
+----------+--------+------+--------+----------+
| minge    |   3.25 |    4 |   3.25 |    13.00 |
| lopatica |   2.10 |    5 |   2.10 |    10.50 |
| galetusa |   4.50 |    3 |   4.50 |    13.50 |
+----------+--------+------+--------+----------+
```

3.13.9. Valoarea NULL.

După cum am văzut în sintaxa comenzii care creează o tabelă, după fiecare coloană a tabelei se pot trece nişte specificatori.

Ne vom ocupa pentru moment de specificatorii **null** respectiv **not null**. Primul, care este şi implicit (dacă nu-l trecem, se asumă automat că acea coloană este de tip **null**), se referă la faptul că în coloana respectivă pot să apară şi valori de tip **null**.

Astfel, dacă într-o comandă **insert** omitem o valoare pentru o anumită coloană, în acea coloană se va trece automat valoarea **null**.

Al doilea specificator, **not null**, NU permite valori **null** în acea coloană. În acest caz, dacă omitem valoarea acelei coloane, în ea se va trece automat:

- 0 dacă este de tip numeric;
- şirul vid dacă acea coloană este de tip şir de caractere;
- primul element din şirul de stringuri, dacă este de tip **enum**;
- mulţimea vidă, dacă acea coloană este de tip **set**.

Exemplu: creăm o tabelă, în care punem două coloane numerice, una în care permite valoarea **null**, alta în care nu, si inserăm în această tabelă o linie în care specificăm doar numele.

```
create table pers1(nume text,virsta int not null,greut int);
insert into pers1(nume) values('Copaceanu');
```

La analizarea valorilor existente prin-o interogare, constatăm că s-au trecut următoarele date:

```
+-----------+--------+-------+
| nume      | virsta | greut |
+-----------+--------+-------+
| Copaceanu |      0 |  NULL |
+-----------+--------+-------+
```

3.13.10. Valori implicite pentru coloanele unei tabele.

Un alt specificator pentru coloanele unei tabele, în cadrul comenzii de creare a tabelei este:

```
default valoare_implicită
```

Acesta permite ca, la omiterea unei valori pentru un anumit câmp, atunci când inserăm o nouă coloană la tabelă, acel câmp să fie automat cu o valoare implicită. Acest specificator nu funcţionează decât în cazul câmpurilor cu lungime fixă. De exemplu, un câmp declarat char(10) poate lua o valoare civilă, pe când unul declarat text nu poate lua o astfel de valoare.

Exemplu: creăm o tabelă **studenti** în care trecem numele, virsta si starea civilă. Ultimelor două câmpuri le vom da valorile implicite 19 respectiv 'necasatorit':

```
create table studenti(nume text,virsta int default 19,
                      stare_civ char(20) default 'necasatorit');
insert into studenti(nume) values('ANTON Costel');
insert into studenti(nume,virsta) values('POP Mihai',21);
```

Iată conținutul tabelei în urma comenzilor de mai sus:

```
+--------------+--------+-------------+
| nume         | virsta | stare_civ   |
+--------------+--------+-------------+
| ANTON Costel |     19 | necasatorit |
| POP Mihai    |     21 | necasatorit |
+--------------+--------+-------------+
```

3.13.11. Cheie primară şi cheie unică.

- **Cheia primară** este constituită dintr-unul sau mai multe câmpuri şi trebuie să îndeplinească toate condițiile de mai jos:

- Valorile reținute de coloana care alcătuieşte cheia primară trebuie să fie distincte. În cazul în care cheia este alcătuită din mai multe câmpuri, pentru a avea două chei distincte este necesar ca acestea (cheile) să fie diferite pentru cel puțin o coloană dintre ele. Tentativa de a adăuga în tabelă o înregistrare care are cheia primară identică cu altă înregistrare se soldează cu eroare iar adăugarea, evident, nu va mai avea loc;

- Câmpul (câmpurile) care alcătuieşte (alcătuiesc) cheia primară trebuie să aibă o lungime fixă. De exemplu, nu putem avea o cheie primară de tip **text** (sau care să conțină un câmp de tip **text**).

Principalul rol al cheii primare este să asigure accesul foarte rapid la o anumită înregistrare, atunci când dăm cheia. Trebuie să precizăm că, înregistrările unei tabele se reţin în ordinea introducerii lor. Atunci când, de exemplu, un câmp este declarat cheie primară, se construieşte o tabelă auxiliară, invizibilă pentru utilizator, în care se trec valorile cheilor, în ordine crescătoare, pentru fiecare cheie reținându-se poziția rândului (liniei) din tabelă, care are acea cheie.

Practic, pornind de la o anumită cheie, în tabela invizibilă se poate identifica foarte rapid numărul de ordine (poziția) al rândului cu acea cheie (se foloseşte căutarea binară). De aici, accesul la acel rând (înregistrare) este imediat.

Pentru a preciza faptul că o anumită coloană este cheie primară, atunci când definim tabela, folosim specificatorul **primary key**, la fel ca în exemplu de mai jos:

```
create table elev(nr_matricol char(10) primary key,nume text);
```

Pentru a preciza faptul că mai multe coloane participă la formarea cheii primare, atunci când definim tabela, folosim specificatorul **primary key(coloană$_1$,...,coloană$_k$)** după ce am terminat descrierea tuturor coloanelor, înainte de a închide paranteza finală de la comanda **create table**, ca în exemplu de mai jos:

```
create table elev(nume char(20),prenume char(20),i_tata char(4),
                  adresa text,primary key(nume,prenume,i_tata));
```

- **Cheia unică** este constituită dintr-un singur câmp. Se foloseşte atunci când, într-o anumită coloană, fie ea participantă la cheia primară, dorim să avem doar valori distincte.

La fel ca şi la cheia primară, putem avea cheie unică doar pentru coloanele de lungime fixă (deci nu putem avea, spre exemplu, cheie unică pentru o coloană de tip **text**).

Dacă se încearcă inserarea unei linii în tabelă, cu o valoare care există deja în cazul unei coloane de tip cheie unică, se va semnala eroare şi inserarea nu va fi permisă.

Specificatorul de cheie unică este unique key. El se foloseşte ca în exemplul de mai jos:

```
create table test(nr_matr char(10) primary key,
                  nume text,cnp char(13) unique key);
```

3.13.12. Coloane cu valori de tip autoincrementare.

De multe ori, în afara datelor deja existente într-o tabelă, este utilă o coloană în care să putem avea doar valori numerice distincte, de preferabil cu valori numerice crescătoare, începând de la 1.

Acest lucru se poate rezolva prin specificatorul **auto_increment** pentru o anumită coloană. Pentru a putea utiliza acest specificator, este necesar să fie îndeplinite următoarele condiţii:

- coloana să fie de un tip întreg;
- coloana să fie formată doar din valori distincte.

La ştergerea unei linii, numerotarea nu se reia, ci se continuă de la ultimul număr atribuit în tabelă. Indiferent dacă utilizatorul specifică la un moment dat o valoare explicită (pe care o doreşte), pentru acest câmp, restul numerotării se va face începând de la cea mai mare valoare pe care a atins-o vreodată acel câmp (chiar dacă între timp, valoarea respectivă a fost ştearsă) plus 1.

Exemplu: creăm o tabelă în care definim o coloană de tipul **auto_increment** şi o altă coloană în care trecem numele unor persoane:

```
create table test(id int unique key auto_increment,nume text);
insert into test(nume) values('cici');
insert into test(nume) values('mimi');
insert into test(nume) values('lola');
```

Iată conținutul tabelei după comenzile de mai sus:

```
+----+------+
| id | nume |
+----+------+
|  1 | cici |
|  2 | mimi |
|  3 | lola |
+----+------+
```

3.13.13. Sortarea datelor.

Afişarea datelor într-o anumită ordine se face tot cu ajutorul interogărilor: într-o interogare ale căror date dorim să le sortăm, specificăm, la sfârşitul comenzii, clauza:

order by expresie$_1$[desc], expresie$_2$[desc],...

specificatorul **desc** este folosit în cazul în care dorim ca sortarea să fie descrescătoare.

Sortarea după mai multe expresii are următoare semnificație: dacă pentru două linii **expresie$_1$** nu este suficientă pentru a determina ordinea (deci pentru ambele, are aceeaşi valoare) se consideră mai departe **expresie$_2$**. Dacă nici **expresie$_2$** nu este suficientă, se ia în considerare **expresie$_3$**, şi aşa mai departe.

Exemple:

select * from tabela order by nume; – se afişează toate datele din tabelă, însă sortate după coloana nume

select * from tabela order by media desc,nume; – se afişează toate datele din tabelă, sortate descrescător după coloana 'media'. Dacă la două linii alte tabelei, coloana 'media' are valori egale, ordinea va fi dată de valorile din coloana 'nume'.

• Indiferent de faptul că dorim o sortare sau nu, există posibilitatea ca o interogare select să afişeze doar anumite rânduri

Pentru aceasta se foloseşte clauza **limit** (în cazul în care în comanda **select** apare şi un **order by**, clauza **limit** se va specifica după).

Sintaxa acesteia are una dintre următoarele două forme:

- **limit n** – din ceea ce s-ar afişa în mod normal, se afişează doar primele **n** linii;
- **limit m,n** – din ceea ce s-ar afişa în mod normal, se afişează doar începând de la a **m+1**-a linie (prima linie este numerotată cu 0) un număr de **n** linii.

3.13.14. Filtrarea datelor.

Se face tot cu ajutorul interogării '**select**': prin specificarea clauzei **where condiție** se vor afişa doar acele linii ale tabelei pentru care condiția este evaluată ca fiind adevărată.

Clauza poate fi utilizată în acelaşi timp cu clauza precendetă (order by) care realizează sortarea. În această situație, trebuie specificată mai întâi clauza where, şi abia apoi clauza order by. Iată mai jos câteva exemple:

select * from tabela where nume like "a%"; – afişează din tabelă, doar acele linii pentru care numele începe cu litera a

select * from tabela where nume like "a%" order by media desc;

– afişează din tabelă, doar acele linii pentru care numele începe cu litera a, în ordinea descrescătoare a mediilor.

Exemple:

select * from tabela order by nume limit 5; – în urma sortării după coloana nume, se afişează doar primele 5 linii;

select * from tabela limit 1,3; – se afişează 3 linii din tabelă începând de la linia a 2-a.

3.13.15. Actualizarea datelor.

Prin actualizare a unei tabele înțelegem operația (operațiile) prin care modificăm fie datele reținute de aceasta, fie modificarea structurii, fie modificarea numelui tabelei.

a) pentru actualizarea datelor:

• pentru inserarea unei noi linii în tabelă, se foloseşte instrucțiunea:

insert into nume_tabelă[(nume_coloană$_1$,...,nume_coloană$_n$)]
values(expr$_1$,...,expr$_n$)

funcționalitatea acestei instrucțiuni a fost deja prezentată;

• pentru inserarea de noi linii în tabelă, linii care provin din datele altei tabele, se poate realiza prin comanda:

insert into [distinct] nume_tabelă[(nume_coloană$_1$,...,nume_coloană$_n$)]
select ...

specificatorul **distinct** se foloseşte pentru a insera doar datele distincte (care nu apar deja în tabela în care inserăm) din tabela sursă.

b) pentru modificarea datelor se utilizează următoarea instrucțiune:

update nume_tabelă set coloană$_1$=exp$_1$,..,coloană$_n$=exp$_n$ [where condiție]

În urma execuției sale, pentru fiecare rând din tabelă care îndeplineşte clauza **where**, se actualizează coloanele indicate de **set** cu expresiile corespunzătoare.

Observație (!): în absența clauzei **where** vor fi modificate toate rândurile tabelei.

c) pentru ştergerea uneia sau a mai multor linii ale unei tabelă se utilizează instrucțiunea:

delete from nume_tabelă where [conditie];

Observație (!): în absența clauzei **where** vor fi şterse TOATE (!) liniile tabelei;

d) ***pentru redenumirea unei tabele***:

rename table nume_vechi to nume_nou;

e) ***pentru ştergerea unei coloane***, chiar dacă aceasta conține date:

alter table nume_tabelă drop column nume_coloană;

f) ***pentru adăugarea (inserarea) unei coloane noi***, cu un anumit tip de date, se utilizează instrucțiunea:

alter table nume_tabelă add nume_coloană_nouă tip_coloană nouă [after coloană_existentă] sau **[first]**

dacă nu este prezentă vreuna dintre clauzele **after** sau **first**, coloana cea nouă se va adăuga la sfârşit. Dacă este prezentă cel puțin vreuna, atunci coloana cea nouă se va insera după coloana identificată de **coloană_existentă** dacă folosim clauza **after...** respectiv la început (înainte de toate celelalte coloane) dacă folosim clauza **first**.

3.13.16. Funcții agregate.

Toate exemplele de până acum au avut ca operanzi doar câmpurile unui acelaşi rând. În practică, este foarte adesea necesar să putem efectua calcule şi cu valorile (toate sau doar o parte) reținute de o coloană.

Pentru astfel de calcule se utilizează aşa-numitele ”funcții agregate”. În cele ce urmează vom prezenta aceste funcții, iar pentru exemplificare vom utiliza următoarea tabelă (cu numele "**elevi**"):

nume	mate	engl	info
Aura Urziceanu	8	10	7
Gica Petrescu	6	4	9
Angela Similea	10	10	10
Gabriel Cotabita	8	5	4
Corina Chiriac	7	NULL	NULL

Iată o trecere în revistă a funcțiilor agregate:

- `count()` – sub forma `count(*)` determină numărul total de linii ale tabelei. Sub forma `count(nume_coloană)` numără câte dintre valorile de pe acea coloană nu sunt `null`;
- `min(nume_coloană)` – determină cea mai mică valoare din coloană, ignorând valorile care sunt egale cu `null`;
- `max(nume_coloană)` – determină cea mai mare valoare din coloană, ignorând valorile care sunt egale cu `null`;
- `sum(nume_coloană)` – determină suma tuturor valorilor din coloană, ignorând valorile care sunt egale cu `null`;
- `avg(nume_coloană)` – determină media aritmetică a tuturor valorilor din coloană, ignorând valorile care sunt egale cu `null`;

Exemple:

–`select count(*) as numar_elevi from elevi;` – afişează numărul total de elevi din tabelă; în cazul nostru va fi 5.

–`select count(info) as numar_note_info from elevi;` – afişează numărul total de valori nenule din coloana info; în cazul nostru va fi 4.

–`select avg(info) as medie_note_info from elevi;` – afişează media tuturor notelor nenule din coloana info; în cazul nostru va fi 7.50.

–`select count(engl) as nota10 from elevi where engl=10;` – afişează câte note de 10 sunt în coloana `engl`;

3.13.17. Subinterogări.

Reprezintă mecanismul prin care rezultatul întors de o interogare poate fi folosi mai departe, pentru a efectua o nouă interogare.

Distingem următoarele două cazuri:

a) ***Interogarea subordonată întoarce o singură valoare*** (deci o coloană cu un singur rând):

Interogarea care întoarce valoarea se scrie inclusă între paranteze rotunde – din acest moment ea se comportă ca şi cum, în acel loc, din punct de vedere sintactic, am avea o singură valoare.

`Exemplu:` pe tabela anterioară, să afişăm toți elevii care au nota maximă la engleză. În primul rând, va trebui să determinăm printr-o subinterogare, care este nota maximă la engleză, iar pe baza valorii obținute să filtrăm datele din tabela cu toți elevii. Pentru o mai bună evidențiere, am colorat cu verde subinterogarea care întoarce nota maximă de la engleză:

```
select * from elevi where engl=(select max(engl) from elevi);
```

b) ***Interogarea subordonată întoarce o tabelă*** şi dorim ca în continuare, să manipulăm datele din această tabelă: În acest caz, tabela returnată de interogarea subordonată trebuie să primească un nume (un alias). Acesta se specifică tot cu ajutorul clauzei **as**. Trebuie de asemenea inclusă între paranteze rotunde. Câmpurile tabelei întoarse de subinterogare au numele exact ca în antetul care s-ar afişa pentru această subinterogare.

Atunci când sunt folosite de către interogarea care o subordonează, câmpurile interogării subordonate trebuie adresate prin numele alias-ului, urmat de caracterul '**.**' şi apoi de numele câmpului din subinterogare.

Exemplu: din tabela cu elevii, dorim să afişăm, doar pentru elevii care au note completate la toate cele 3 materii, o listă cu numele şi media celor 3 materii, în ordine descrescătoare a mediilor.

Dacă am dori să afişăm o tabelă ce conține numele şi media generală doar de la acei elevi care au trecute toate cele 3 note, am scrie interogarea următoare (ne reamintim că, dacă într-o sumă, cel puțin un termen este **null**, atunci toată suma este **null**):

```
select nume,(mate+engl+info)/3 as media from elevi
                        where mate+engl+info is not null;
```

Aceasta va juca rolul subinterogării. Coloanele sale sunt **nume** respectiv **media**. Remarcați faptul că, în cadrul interogării pe care ne propunem s-o realizăm această tabelă va primi alias-ul **medii**, iar coloanele sale pot fi accesate prin expresiile **medii.nume** respectiv **medii.media**:

```
select * from (select nume,(mate+engl+info)/3 as media from elevi
                  where mate+engl+info is not null) as medii
               order by medii.media desc;
```

3.13.18. Gruparea datelor.

Datele dintr-o tabelă pot fi grupate în funcție de valorile dintr-o anumită coloană. Astfel, toate valorile egale dintr-o anumită coloană vor forma un grup. Prelucrările datelor din cadrul unui grup se pot face cu ajutorul funcțiilor agregate, aceste acționând datelor din fiecare grup.

Gruparea efectivă se realizează cu clauza **group by**, aplicată de comenzii **select**. În cazul în care dorim filtrarea interogării rezultate în urma unei grupări, nu se mai foloseşte clauza **where** (va genera eroare!) ci există o nouă clauză, **having**.

Datele din tabela rezultată în urma grupării vor fi sortate după coloana care realizează gruparea.

Sintaxa generală a comenzii select în cazul unei grupări este:

```
select exp1,exp2,... from tabelă group by exp_gr1,exp_gr2,....
                  [having condiție_filtrare];
```

Grupările se constituie ierarhic după **exp_gr$_1$**, **exp_gr$_2$**, etc. Practic, grupurile mari sunt date de valorile egale pentru **exp_gr$_1$**. În cadrul acestora, se fac subgrupurile după valorile egale ale lui **exp_gr$_2$**, etc.

Printre expresiile din select putem avea:

- nume de câmpuri (sau expresii în funcție de acestea): în acest caz se va folosi valoarea primei linii din fiecare grup;

- funcții agregate: acestea vor acționa asupra tuturor valorilor coloanei din grup asupra cărora sunt aplicate;

- dacă în loc de **exp$_1$**,**exp$_2$**,... folosim ***** se va afișa prima linie din fiecare grup.

Exemple: Considerăm următoarea tabelă, numită pers, asupra căreia vom aplica mai multe interogări de grupare:

```
+------+------+--------+
| nume | jud  | scoala |
+------+------+--------+
| dora | bv   | sc2    |
| bebe | cj   | sc2    |
| bubu | cj   | sc2    |
| fifi | bv   | sc3    |
| lala | bv   | sc2    |
| gogu | cj   | sc1    |
| fefe | bv   | sc2    |
| gigi | bv   | sc1    |
| mimi | bv   | sc2    |
| dede | cj   | sc1    |
| lola | bv   | sc1    |
| coco | bv   | sc3    |
| calu | cj   | sc1    |
| cici | bv   | sc1    |
+------+------+--------+
```

```
select jud,count(jud) as total_judet from pers group by jud;
```

– grupează după județ, afișând pentru fiecare județ indicativul său, respectiv numărul total de persoane provenind din acel județ:

```
+------+-------------+
| jud  | total_judet |
+------+-------------+
| bv   |           9 |
| cj   |           5 |
+------+-------------+
```

```
select jud,scoala,count(scoala) as total_scoala from pers
    group by jud,scoala;
```

– grupează după județ, în fiecare județ apoi după numele școlii, afișând pentru fiecare grup obținut numele județului, numele școlii și numărul de elevi din acea școală:

```
+------+--------+--------------+
| jud  | scoala | total_scoala |
+------+--------+--------------+
| bv   | sc1    |            3 |
| bv   | sc2    |            4 |
| bv   | sc3    |            2 |
| cj   | sc1    |            3 |
| cj   | sc2    |            2 |
+------+--------+--------------+
```

Dacă din interogarea de mai sus dorim să afişăm doar acele şcoli pentru care numărul de elevi este de cel puțin 3, adăugăm clauza **having** interogării de mai sus:

```
select jud,scoala,count(scoala) as total_scoala from pers
      group by jud,scoala having total_scoala>=3;
```

Rezultatul afişat va fi următorul:

```
+------+--------+--------------+
| jud  | scoala | total_scoala |
+------+--------+--------------+
| bv   | sc1    |            3 |
| bv   | sc2    |            4 |
| cj   | sc1    |            3 |
+------+--------+--------------+
```

3.13.19. Uniuni de tabele.

Un punct de maximă importanță şi eficiență în utilizarea bazelor de date este dat de uniunea de tabele. Uniunea este alcătuită din două sau mai multe tabele între care există o legătură. De cele mai multe ori, legătura este dată de valorile existente în câte o coloană a fiecărei tabelă din uniune. Pentru tabelele care alcătuiesc uniunea se pot realiza alias-uri de nume, sub forma nume_tabelă as alias_nume, introduse în clauza from. Alias-urile pot fi utilizate în orice parte a instrucțiunii select. Ca şi la subinterogări, adresarea unei coloane a unei tabele se va face sub forma nume_alias.nume_coloană.

E recomandat să asigurăm nume scurte pentru alias-urile tabelelor (a, b, pers, elev,...).

Un alt avantaj al alias-urilor este că putem realiza auto-uniuni, în cadrul cărora aceeaşi tabelă poate avea alias-uri diferite.

a) ***Uniuni de tip produs cartezian***: se realizează printr-un select în care se trec toate câmpurile pe care dorim să le obținem, din cadrul ambelor tabele, iar la clauza from se trec ambele tabele, fiecare dintre ele putând avea definit şi un alias. Clauza where poate filtra doar acele perechi de linii (din ambele tabele) pe care le dorim. În absența clauzei where, se vor genera toate combinațiile (produs cartezian) de linii din prima tabelă, cu linii din a doua tabelă.

Comanda poate fi generalizată şi pentru mai mult de două tabele.

Sintaxa:

```
select exp1,exp2 ... from tabelă1 [as alias1],tabelă2 [as alias2]
                                                [where condiție];
```

Evident, **exp1**, **exp2**, **...** vor trebui să conțină numele coloanelor, simplu, dacă nu există confuzii (coloane care au în ambele tabele acelaşi nume) sau numele alias-urilor urmate de punct şi de numele coloanelor în caz contrar.

Exemplu: fie următoarele două tabele, una cu numele unor actori, fiecare actor având asociat un cod, alta cu numele unor filme, fiecare film având trecut în dreptul său codul actorilor care joacă în acel film:

```
mysql> select * from actori;
+------+-----------------+
| cod  | nume            |
+------+-----------------+
|  101 | Tom Hanks       |
|  105 | Dustin Hoffman  |
|  107 | Roberto Benigni |
|  110 | Jessica Lange   |
+------+-----------------+
```

```
mysql> select * from filme;
+-----------------+-----------+
| titlu           | cod_actor |
+-----------------+-----------+
| Tootsie         |       105 |
| Tootsie         |       110 |
| La vita e bella |       107 |
+-----------------+-----------+
```

Prin următoarea interogare realizăm un produs cartezian al celor două:

```
select f.titlu,f.cod_actor,a.cod,a.nume from filme as f,actori as a;
```

(putem folosi şi următoarele forme:

1) folosim operatorul * pentru a specifica toate coloanele din fiecare tabelă:

```
select f.*,a.* from filme as f,actori as a;
```

2) sau şi mai simplu, pentru că nu există coloane care să se confunde (toate au nume diferite):

```
select * from filme,actori;
```

)

Rezultatul este următorul (observați cum se ia fiecare film cu fiecare actor):

```
+-----------------+-----------+------+-----------------+
| titlu           | cod_actor | cod  | nume            |
+-----------------+-----------+------+-----------------+
| Tootsie         |       105 |  101 | Tom Hanks       |
| Tootsie         |       110 |  101 | Tom Hanks       |
| La vita e bella |       107 |  101 | Tom Hanks       |
| Tootsie         |       105 |  105 | Dustin Hoffman  |
| Tootsie         |       110 |  105 | Dustin Hoffman  |
| La vita e bella |       107 |  105 | Dustin Hoffman  |
| Tootsie         |       105 |  107 | Roberto Benigni |
| Tootsie         |       110 |  107 | Roberto Benigni |
| La vita e bella |       107 |  107 | Roberto Benigni |
| Tootsie         |       105 |  110 | Jessica Lange   |
| Tootsie         |       110 |  110 | Jessica Lange   |
| La vita e bella |       107 |  110 | Jessica Lange   |
+-----------------+-----------+------+-----------------+
```

Dacă dorim să se afişeze doar datele care sunt în corespondeță (acest lucru este făcut, evident, prin câmpurile numerice cod_actor, respectiv cod) folosim următoarea interogare:

```
select f.*,a.* from filme as f,actori as a where f.cod_actor=a.cod;
```

(sau, şi mai simplu: `select * from filme,actori where cod_actor=cod;`)

Rezultatul va fi, conform aşteptărilor, în forma următoare:

```
+-----------------+-----------+------+-----------------+
| titlu           | cod_actor | cod  | nume            |
+-----------------+-----------+------+-----------------+
| Tootsie         |       105 |  105 | Dustin Hoffman  |
| La vita e bella |       107 |  107 | Roberto Benigni |
| Tootsie         |       110 |  110 | Jessica Lange   |
+-----------------+-----------+------+-----------------+
```

b) ***Uniuni de tip*** **`inner join`**:

Se aplică atunci când uniunea se realizează doar între două tabele şi dorim ca din ambele tabele să fie afişate doar acele linii pentru care condiția de legătură este satisfăcută. Sintaxa comenzii:

`select exp1,exp2,.. from tabelă1 [as alias1] inner join tabelă2 [as alias2] on condiție;`

Ca şi mai înainte, dacă există coloane care se numesc la fel în ambele tabele, vom folosi alias-uri.

`Exemplu:` acelaşi lucru ca în exemplul anterior se poate realiza cu următoarea interogare:

```
select * from filme as f inner join actori as a on f.cod_actor=a.cod;
```

(sau, mai simplu: `select * from filme join actori on cod_actor=cod;`)

Obs: cuvântul cheie ”**`inner`**” poate fi omis: prin specificarea doar a clauzei **`join`** se subînțelege că, de fapt, este vorba de un ”**`inner join`**”;

c) ***Uniuni de tip*** **`left outer join`**:

Se aplică atunci când uniunea se realizează între două tabele şi dorim ca din prima tabelă să fie incluse absolut toate liniile iar din a doua tabelă doar acele linii care îndeplinesc condiția de la clauza **`on`**. La acele linii din prima tabelă care nu au corespondent în a doua tabelă, în dreptul câmpurilor acesteia din urmă se va trece valoarea **`null`**. Sintaxa comenzii este:

```
select exp1,exp2,... from tabelă1 [as alias1]
                        left outer join tabelă2 [as alias2] on condiție;
```

În exemplu nostru, o astfel de interogare în care prima tabelă este cea cu filme iar a doua cu actori, va afişa toate filmele, inclusiv cele pentru care nu am trecut actorii corespondenți în cea ce-a doua tabelă. Pentru a face exemplul cât mai ilustrativ, inserăm următoarea linie în tabela cu filme:

```
insert into filme(titlu) values('Forest Gump');
```

După care lansăm următoarea interogare de tipul **`left outer join`**:

```
select * from filme left outer join actori on cod_actor=cod;
```

Obs: Cuvântul ”**`outer`**” poate fi omis: prezența lui ”**`left`**” implică o uniune de tip **`outer`**.

Iată rezultatul:

```
+-----------------+-----------+------+-----------------+
| titlu           | cod_actor | cod  | nume            |
+-----------------+-----------+------+-----------------+
| Tootsie         |       105 |  105 | Dustin Hoffman  |
| Tootsie         |       110 |  110 | Jessica Lange   |
| La vita e bella |       107 |  107 | Roberto Benigni |
| Forest Gump     |      NULL | NULL | NULL            |
+-----------------+-----------+------+-----------------+
```

d) ***Uniuni de tip*** **`right outer join`**:

După cum e de aşteptat, prin analogie cu situația precedentă, se aplică atunci când uniunea se realizează între două tabele şi dorim ca din a doua tabelă să fie incluse absolut toate liniile, iar din prima doar acele linii pentru care condiția de la clauza on este îndeplinită. La acele linii din a

doua tabelă care nu au corespondent în prima tabelă, în dreptul câmpurilor acesteia din urmă se va trece valoarea **null**. Sintaxa comenzii este:

```
select exp1,exp2,... from tabelă1 [as alias1]
                    right outer join tabelă2 [as alias2] on condiție;
```

În exemplu nostru, o astfel de interogare în care prima tabelă este cea cu filme iar a doua cu actori, va afişa toţi actorii, inclusiv cei pe care nu i-am asociat la vreun film, şi doar acele filme către care există legătură din tabela cu actori.

Exemplu:

```
select * from filme right outer join actori on cod_actor=cod;
```

Iată rezultatul:

```
+-----------------+-----------+------+-----------------+
| titlu           | cod_actor | cod  | nume            |
+-----------------+-----------+------+-----------------+
| NULL            |      NULL |  101 | Tom Hanks       |
| Tootsie         |       105 |  105 | Dustin Hoffman  |
| La vita e bella |       107 |  107 | Roberto Benigni |
| Tootsie         |       110 |  110 | Jessica Lange   |
+-----------------+-----------+------+-----------------+
```

Obs: Cuvântul "**outer**" poate fi omis: prezenţa lui "**right**" implică o uniune de tip **outer**.

e) ***Auto-uniuni***:

Se pot face uniuni în care ambele tabele sunt, de fapt, unul şi acelaşi, însă au alias-uri diferite şi se consideră unite printr-o coloană.

Exemplu: să considerăm următoarea tabelă, cu care am lucrat într-unul din exemplele paragrafului anterior 3.13.5:

```
mysql> select * from pictori;
+------------------+------------------+
| nume             | curent           |
+------------------+------------------+
| Gustave Klimt    | art nouveau      |
| Vincent Van Gogh | postimpresionism |
| Alphonse Mucha   | art nouveau      |
| Auguste Renoir   | impresionism     |
| Rene Magritte    | suprarealism     |
| Tiziano Vecellio |                  |
+------------------+------------------+
```

Pentru a afişa toţi pictorii care aparţin aceluiaşi curent căruia îi aparţine şi Gustav Klimt ***putem*** proceda în felul următor (! atenţie, exemplu este doar didactic, din punct de vedere practic NU se procedează în acest fel):

```
select b.nume from pictori as a join pictori as b on a.curent=b.curent
                    where a.curent='art nouveau' and a.nume='Gustave Klimt';
```

3.13.20. Exploatarea bazelor de date MySQL prin intermediul limbajului PHP.

Pentru a face funcțional suportul de conectivitate a limbajului PHP către bazele de date MySQL, este necesară utilizarea bibliotecii **mysql**. În cazul pachetului XAMPP, aceasta este instalată şi activată în mod implicit. În cazul altor distribuții, acest lucru trebuie făcut manual.

• În primul rând, din interiorul unui script PHP trebuie realizată conectarea la baza de date. Pentru acest lucru, avem nevoie de adresa IP a server-ului MySQL (în cazul nostru, deoarece utilizăm o configurație standard, server-ul se află pe calculatorul local, deci această adresă este '**localhost**'), numele unui utilizator şi parola cu care acest utilizator se conectează la baza de date.

În cazul nostru, vom asuma faptul că utilzatorul are numele '**utilizator**' iar parola sa este '**parola**' (aşa cum l-am creat în paragraful 3.3.12).

Conectarea efectivă se face cu ajutorul instrucțiunii:

```
$resursă=mysql_connect("localhost", "utilizator", "parola");
```

Variabila **$resursă** va fi cea prin care, în continuare, ne vom putea referi la conexiunea creată. Dacă aceasta nu reuşeşte din diferite motive (nume utilzator sau parola incorecte, server-ul MySQL nu este pornit, etc.) variabila **$resursă** va avea valoarea **null**.

• Selectarea unei baze de date, o dată ce conexiunea a fost creată, pentru a plasa comenzi în această bază de date, este:

```
mysql_select_db(nume_bază_date);
```

Aceasta întoarce o valoare diferită de 0 dacă selectarea a reuşit, sau 0 în caz contrar;

• Plasarea unei comenzi (interogări) asupra bazei de date selectate se face prin instrucțiunea:

```
$resursă1=mysql_query(string_ce_conține_comanda_MySQL);
```

Dacă interogarea (comanda) a funcționat corect, funcția va întoarce o valoare diferită de 0. În caz contrar, întoarce valoarea 0.

Variabila **$resursă1** este cea care va permite, în continuare, obținerea rezultatului efectiv al interogării.

În cazul oricărei erori, perechea de funcții **mysql_errno();** respectiv **mysql_error();** ne întorc codul (numărul) ultimei erori generate, respectiv mesajul asociat acesteia.

• Închiderea conexiunii către baza de date MySQL se face prin comanda:

```
mysql_close(resursă_conexiune);
```

E preferabil să închidem conexiunile deschise prin **mysql_connect** la sfârşitul script-ului.

• Obținerea datelor întoarse de interogare se face prin intermediul resursei întoarse de comanda **mysql_query**. Iată funcțiile cele mai folosite:

- **mysql_num_rows(resursa)** – returnează numărul de linii (rânduri) ale tabelei rezultate în urma interogării;

- **mysql_num_fields(resursa)** – returnează numărul de câmpuri (coloane) ale tabelei rezultate în urma interogării;

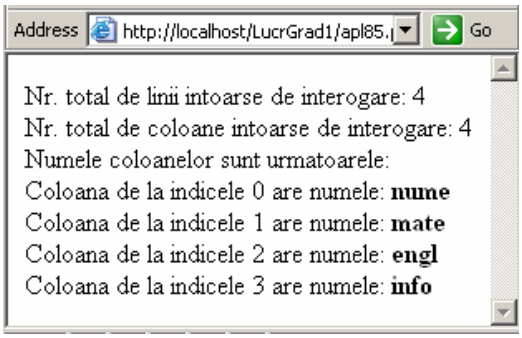
- **mysql_field_name(resursa,k)** – returnează numele câmpului (coloanei) al **k**-lea al tabelei. Coloanele sunt numerotate de la 0;

Exemplu: următorul script PHP va realiza o conectare la baza de date MySQL, lansând o interogare de afişare a tuturor datelor din tabela elevi, din exemplul **3.13.16**. În urma acestei interogări va afişa datele obţinute cu ajutorul funcţiilor de mai sus:

apl085.php:

```
<?php
  $l=mysql_connect('localhost','utilizator','parola');
  mysql_select_db('lucru');
  $r=mysql_query('select * from elevi');
  $n=mysql_num_rows($r);
  $m=mysql_num_fields($r);
  echo 'Nr. total de linii intoarse de interogare: ',$n,'<br>';
  echo 'Nr. total de coloane intoarse de interogare: ',$n,'<br>';
  echo 'Numele coloanelor sunt urmatoarele:<br>';
  for($j=0;$j<$m;$j++)
     echo 'Coloana de la indicele ',$j,' are numele: <b>',
                          mysql_field_name($r,$j),'</b><br>';
  mysql_close($l);
?>
```

Iată rezultatul afişat de către script:



- **mysql_fetch_array(resursă)** – întoarce rândul (linia) curentă, sub forma unui şir (vector) din interogarea specificată de resursă, mutând pointerul intern pe următoarea linie (rând) al interogării. Dacă liniile interogării s-au terminat, va întoarce fals.

În şirul întors, putem folosi pe post de indici atât numele coloanelor, cât şi valori numerice cuprinse între 0 şi numărul de coloane minus 1.

Modelul general de aplicare al acestei instrucţiuni este pe o repetitivă de tip while, în care atribuim unui vector valoarea întoarsă de funcţia mysq_fetch_row direct în condiţia de ciclare a repetitivei:

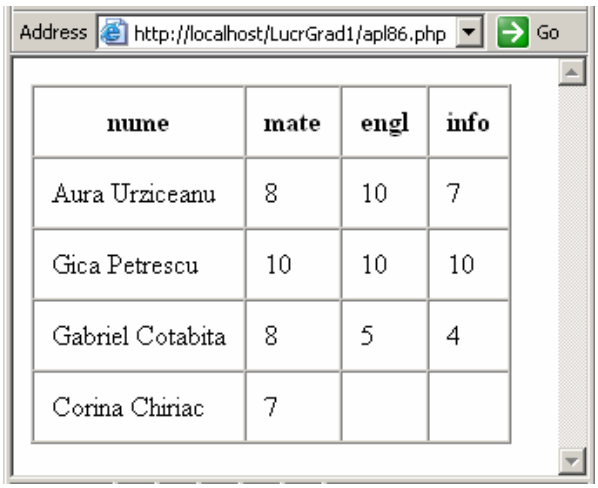
```
while($a=msql_fetch_row($resursa))
 {...prelucrăm elementele vectorului $a, care ne dau linia curentă a interogării...}
```

Iată modul în care putem obține, într-o formă prezentabilă, toate datele conținute de tabela din exemplul anterior, afișând inclusiv o linie de antet cu numele coloanelor tabelei:

apl086.php:

```
<?php
  $l=mysql_connect('localhost','utilizator','parola');
  mysql_select_db('lucru');
  $r=mysql_query('select * from elevi');
  $m=mysql_num_fields($r);
  echo '<table border="1" cellspacing="0" cellpadding="10">';
  //pregătim afişarea într-un tabel
  echo '<tr>';
  for($j=0;$j<$m;$j++)//afişăm numele coloanelor în celule de tipul th
     echo '<th align="center">',mysql_field_name($r,$j);
  while($a=mysql_fetch_row($r))
  {
     echo '<tr>';
        for($j=0;$j<$m;$j++)
            {
                echo '<td>';
                if($a[$j]) echo $a[$j];//în locul valorilor vide afişăm un spaţiu
                           else echo ' ';//altfel, tabelul HTML va arăta urât
             }
  }
  echo '</table>';
  mysql_close($l);
?>
```

Iată rezultatul afișat de către script în browser:



nume	mate	engl	info
Aura Urziceanu	8	10	7
Gica Petrescu	10	10	10
Gabriel Cotabita	8	5	4
Corina Chiriac	7		

4. APLICAȚII PRACTICE ŞI METODOLOGICE

Reluarea, dintr-o altă perspectivă, a algoritmilor reprezentativi studiați la disciplina informatică în clasele a IX-a, a X-a şi a XI-a.

În cele ce urmează, voi prezenta nu doar câteva exerciții şi probleme rezolvate ci şi câteva enunțuri de probleme propuse spre rezolvare, care tratează principalele teme studiate la disciplina informatică, din clasa a IX-a până în clasa a XI-a.

În cazul problemelor rezolvate, sunt prezentate enunțul şi codul sursă (cu comentarii, acolo unde este cazul) al rezolvării.

Principalele noutăți sunt aduse de interfață, care, prin intermediul limbajului PHP şi deci, implicit, prin funcționalitatea adusă de o pagină web, poate deveni mult mai atractivă, în special prin utilizarea culorilor şi elementelor grafice. Astfel, o serie de aplicații pot genera afişări mult mai sugestive ale datelor de ieşire.

Ultimul subcapitol aduce în discuție şi abordarea interdisciplinară a câtorva noțiuni de geometrie analitică plană, aplicate în realizarea de reprezentări grafice. Dintre acestea, cele mai spectaculoase constau în reprezentări grafice de fractali.

4.1. Algoritmi care nu operează cu şiruri (cifrele unui număr, numere prime, factori primi, cmmdc, şirul lui Fibonacci)

1) **Problemă rezolvată (`apl087.html` + `apl088.php`)**:

Se citeşte, prin intermediul unui câmp de tip text al unui formular, o valoare naturală având cel mult 9 cifre. Scrieți un program PHP care să preia valoarea respectivă din formular şi să o valideze, afişând apoi cifrele sale într-un tabel cu o singură linie, în aceeaşi ordine în care apar ele în număr, colorând cu fundal verde cifrele de valoare minimă, cu fundal roşu cifrele de valoare maximă şi cu fundal galben restul cifrelor.

Dacă numărul are toate cifrele egale, se va colora fundalul tuturor cifrelor cu culoarea roşie.

Rezolvare:

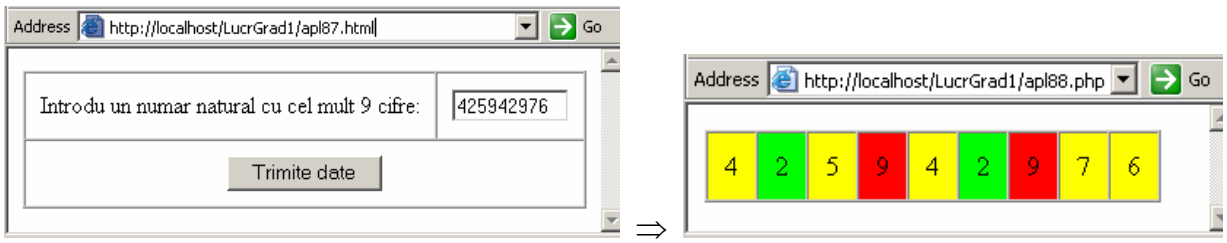
`apl087.html:`

```
<html><body>
<form action="apl088.php" method="post">
<table border="1" cellspacing="0" cellpadding="10">
<tr><td>Introdu un numar natural cu cel mult 9 cifre:
    <td> <input type="text" maxlength="9" size="9" name="nr">
<tr><td colspan="2" align="center">
    <input type="submit" value="Trimite date">
</table>
</form>
</body></html>
```

apl088.php:

```
<html><body>
<?php
 $nr=@$_POST['nr'];
 if(!$nr)//daca este egala cu NULL, atunci inseamna ca nu s-a introdus nimic
   echo "Ai uitat sa introduci vreo valoare";
 else//functia is_numeric verifica daca valoarea dintr-un string este numerica
   if(!is_numeric($nr))
     echo "Valoarea introdusa nu este numerica!";
   else//daca este numerica, verificam sa nu fie cumva reala:
     if($nr!=(int)$nr)
       echo "Valoarea introdusa nu este intreaga!";
     else//daca este intreaga, verificam sa fie pozitiva si sa aiba cel mult 9 cifre
       if($nr<0||$nr>999999999)
         echo "valoarea introdusa trebuie sa fie pozitiva si sa aiba cel mult 9 cifre";
       else
         {//in sfirsit ne putem apuca de rezolvarea problemei: impartim numarul
//in cifrele sale, determinind minimul si maximul si determinind totodata si puterea
//lui 10 care corespunde numarului total de cifre ale numarului. Avem nevoie de aceasta
//pentru a afisa cifrele numarului in acceasi ordine ca si in numar
          $p=1;
          $n=(int)$nr;
          $min=10;$max=-1;
          do{
             $p*=10;
             if($n%10<$min) $min=$n%10;
             if($n%10>$max) $max=$n%10;
             $n=(int)($n/10);
             }while($n);
           echo '<table border="1" cellspacing="0" cellpadding="10"><tr>';
           $n=(int)$nr;
           do{
               $p/=10;
              $cif=(int)($n/$p)%10;
               if($cif==$max) echo '<td bgcolor="red">',$cif;
               else if($cif==$min) echo '<td bgcolor="lime">',$cif;
                    else echo '<td bgcolor="yellow">',$cif;
              }while($p!=1);
             }
?>
</body></html>
```

Iată un exemplu de rulare:



2) Se citeşte, prin intermediul unui câmp de tip text al unui formular, o valoare naturală având cel mult 9 cifre. Scrieți un program PHP care să preia valoarea respectivă din formular şi să o valideze, afişând apoi cifrele sale într-un tabel cu o singură linie, în aceeaşi ordine în care apar ele în număr, colorând cu fundal verde cifrele pare şi cu fundal roşu cifrele impare. Afişați apoi, sub acest tabel, care este suma cifrelor pare, respectiv care este suma cifrelor impare.

3) Se citeşte, prin intermediul unui câmp de tip text al unui formular, o valoare naturală având cel mult 9 cifre. Scrieți un program PHP care să preia valoarea respectivă din formular şi să o valideze, afişând apoi cifrele sale într-un tabel cu o singură linie, în aceeaşi ordine în care apar ele în număr, colorând cu fundal verde cifrele pare şi cu fundal roşu cifrele impare. Afişați apoi, sub acest tabel, numărul natural format doar cu cifrele pare, respectiv numărul natural format doar cu cifrele impare.

4) Se citeşte, prin intermediul unui câmp de tip text al unui formular, o valoare naturală având cel mult 9 cifre. Scrieți un program PHP care să preia valoarea respectivă din formular şi să o valideze, afişând apoi cifrele sale într-un tabel cu o singură linie, în aceeaşi ordine în care apar ele în număr, colorând cu fundal verde cifrele aflate pe poziții pare respectiv cu fundal roşu cifrele aflate pe poziții impare. Pozițiile se consideră astfel: cifra unităților are poziția 0, cifra zecilor poziția 1, cifra sutelor poziția 2, etc.
Afişați apoi sub tabel suma cifrelor aflate pe pozițiile pare, respectiv suma cifrelor aflate pe pozițiile impare, apoi, sub ele, modulul diferenței dintre cele două sume.
Pe baza valorii obținute afişați un mesaj corespunzător faptului că numărul introdus este sau nu divizibil cu 11 (dacă valoarea obținută este divizibilă cu 11, atunci tot numărul este divizibil cu 11).

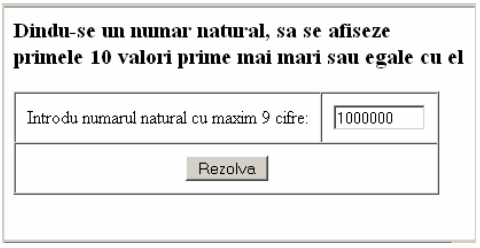
5) **Problemă rezolvată (`apl089.html` + `apl090.php`)**:

Se citeşte, prin intermediul unui câmp de tip text al unui formular, un număr natural având cel mult 9 cifre. Scrieți un program PHP care să preia valoarea respectivă din formular şi să afişeze primele 10 numere prime mai mari sau egale cu numărul introdus.

Numerele determinate se vor afişa într-un tabel cu o singură coloană, alternând culorile de fundal ale celulelor cu bleu respectiv portocaliu.

`apl089.html`

```
<html><body>
<h3>Dindu-se un numar natural, sa se
afiseze primele 10 valori
prime mai mari sau egale cu el</h3>
<form action="apl090.php" method="post">
<table border="1" cellspacing="0" cellpadding="10">
<tr><td>Introdu numarul natural cu maxim 9 cifre:
    <td><input type="text" name="n" maxlength="9" size="9">
<tr><td colspan="2" align="center">
    <input type="submit" value="Rezolva">
</table></form>
</body></html>
```



apl090.php

```
<html><body>
<?php
 $n=$_POST['n'];
 $k=0;
 echo '<h3>Numarul primit = ',$n;
 echo '<br>Iata primele 10 valori prime mai mari sau egale cu el<br>';
 echo '<table border="1" cellspacing="0" cellpadding="10">';
 while($k<=10)//cit timp nu s-au generat inca toate cele 10 numere
 {//testam daca valoarea curenta, adica $n, este numar prim:
  $is_prime=1;
  for($d=2;$d<=sqrt($n);$d++)
     if($n%$d==0) $is_prime=0;
  if($n<=1) $is_prime=0;
  //daca este numar prim, il afisam intr-o noua linie de tabel
  if($is_prime)
     {
         echo '<tr><td align="center" ';
         //in functie de paritatea lui$k,
         //alternam culorile de fundal ale celulelor tabelei
         if($k%2==0) echo 'bgcolor="#88ffff">';
               else echo 'bgcolor="#ffaa50">';
         echo $n;
         $k++;//daca $n a fost numar prim, il numaram in contorul $k
     }
   $n++;//si trecem la verificarea urmatorului numar
 }
?>
</table></body></html>
```

Numarul primit = 1000000
Iata primele 10 valori prime mai mari sau egale cu el

1000003
1000033
1000037
1000039
1000081
1000099
1000117
1000121
1000133
1000151
1000159

6) Se citeşte, prin intermediul unui câmp de tip text al unui formular, un număr natural având cel mult 9 cifre. Scrieți un program PHP care să preia valoarea respectivă din formular şi să afişeze toate valorile de cel puțin două cifre, mai mici sau egale cu numărul natural introdus, care au proprietatea că sunt în acelaşi timp şi numere prime şi palindroame. Valorile se vor afişa într-un tabel cu o singură coloană, alternând culorile de fundal ale celulelor cu galben, respectiv mov pal.

7) Se citeşte, prin intermediul unui câmp de tip text al unui formular, un număr natural având cel mult 9 cifre. Scrieți un program PHP care preia valoarea respectivă din formular şi afişează descompunerea numărului în factori primi. Afişarea se va face prezentabil, folosind tag-urile ^{şi} pentru scrierea exponenților diferiți de 1 ai factorilor care apar în descompunere. De exemplu, dacă se citeşte numărul 300, se va afişa descompunerea în forma: $2^2 * 3 * 5^2$.

8) Se citesc, prin intermediul a două câmpuri de tip text ale unui formular, două numere naturale nenule având cel mult 9 cifre. Scrieți un program PHP care afişează numerele introduse şi totodată calculează şi afişează atât cmmdc-ul cât şi cmmmc-ul lor.

9) Se citeşte, prin intermediul unui câmp de tip text al unui formular, un număr natural cu cel puțin 3 cifre. Să se scrie un program PHP care determină şi afişează cel mai mic număr natural, mai mare sau egal cu numărul dat, care are proprietatea că cele două valori obținute prin eliminarea primei respectiv a ultimei sale cifre, sunt prime între ele.
Ex: Dacă se citeşte 1266 (care NU este un astfel de număr, deoarece 126 şi 266 NU sunt prime între ele) se va determina şi afişa numărul 1269.

10) Se citeşte, prin intermediul unui câmp de tip text al unui formular, un număr natural n cuprins între 3 şi 40. Să se scrie un program PHP care afişează într-un tabel termenii şirului lui Fibonacci începând de la cel de indice 2 la cel de indice n, pentru fiecare termen afişând totodată şi raportul dintre acest termen şi cel de dinainte sa, sub forma unui număr real, cu zecimale.

11) Se citeşte, prin intermediul unui câmp de tip text al unui formular, un număr natural n având cel mult 9 cifre. Să se scrie un program PHP care determină şi afişează cel mai mare termen al şirului lui Fibonacci care are proprietatea că este mai mic sau egal cu numărul dat.

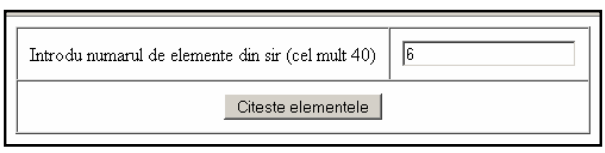
12) Se citeşte, prin intermediul unui câmp de tip text al unui formular, un număr natural n având cel mult 9 cifre. Să se scrie un program PHP care determină descompunerea numărului n în sumă de termeni ai şirului lui Fibonacci. Rezultatul se va afişa prezentabil sub forma următoare:
Exemplu: dacă se citeşte numărul n = 2891 vom avea descompunerea:
2891 = 1 + 5 + 13 + 21 + 34 + 89 + 144 +377 + 610 + 1597

4.2. Algoritmi care operează cu şiruri sau matrice (sortări, ştergeri, inserări)

13) **Problemă rezolvată** (**`apl091.html`** + **`apl092.php`** + **`apl093.php`**): Se citeşte, prin intermediul unui câmp de tip text al unui formular, un număr natural nenul n, mai mic sau egal cu 40. Pe baza lui n veți genera un alt formular, în care veți citi un şir cu n elemente numere naturale (fiecare element al şirului va fi citit într-un textbox). Scrieți un program PHP care preia valorile şirului, le sortează crescător şi le afişează într-un tabel cu o singură linie, colorând cu fundal verde deschis elementele pare, respectiv cu fundal roşu deschis elementele impare.

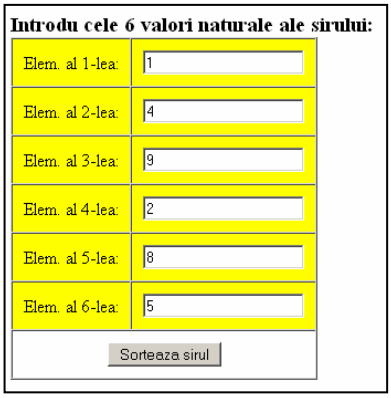
`apl091.html`

```
<html><body><form action="apl092.php" method="post">
<table border="1" cellspacing="0" cellpadding="10">
<tr><td>
Introdu numarul de elemente din sir (cel mult 40)
<td><input type="text" name="n">
<tr><td align="center" colspan="2">
<input type="submit" value="Citeste elementele">
</table></form></body></html>
```



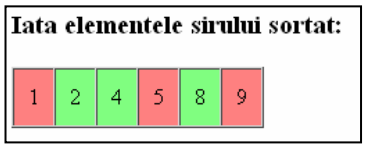
apl092.php

```
<html><body>
<?php
 $n=$_POST['n'];
 echo '<h3>Introdu cele ',$n,' valori naturale ale sirului:',"\n";
 echo '<form action="apl093.php" method="post">',"\n";
 echo '<table border="1" cellspacing="0" cellpadding="10">',"\n";
 for($i=1;$i<=$n;$i++)
    echo '<tr bgcolor="yellow"><td>Elem. al ',$i,'-lea: ',
         '<td><input type="text" name="a[',$i,']">',"\n";
//deci pentru fiecare indice generam un text care se va
//numi a[1], a[2], ....
 echo '<tr><td colspan="2" align="center">',"\n";
 echo '<input type="submit" value="Sorteaza sirul">',"\n";
 echo '<input type="hidden" name="n" value="',$n,'">',"\n";
//deci valoarea lui $n o trimitem mai departe (pentru ca avem
//nevoie de ea la prelucrarea datelor shirului) prin intermediul
//unui control de tipul "hidden"
 echo '</table></form>';
?>
</body></html>
```



apl093.php

```
<html><body>
<?php
  $n=$_POST['n'];
  $a=$_POST['a'];//in felul asta obtinem
//in variabila $a tot sirul trimis din cadrul form-ului
//prin intermediul atributelor name=a[1], name=a[2],...
  //mai intii sortam elementele sirului, crescator:
  for($i=1;$i<=$n-1;$i++)
    for($j=$i+1;$j<=$n;$j++)
       if($a[$i]>$a[$j])
            {
               $aux=$a[$i];$a[$i]=$a[$j];$a[$j]=$aux;
            }
  echo '<h3>Iata elementele sirului sortat:</h3>';
  echo '<table border="1" cellspacing="0" cellpadding="10">';
  echo '<tr>';
  for($i=1;$i<=$n;$i++)
    if($a[$i]%2==0) echo '<td bgcolor="#80ff80">',$a[$i];
                else echo '<td bgcolor="#ff8080">',$a[$i];
?>
</table></body></html>
```



14) Se citeşte, prin intermediul unui câmp de tip text al unui formular, un număr natural nenul n, mai mic sau egal cu 40. Pe baza lui n veți genera un alt formular, în care veți citi un şir cu n elemente numere naturale (fiecare element al şirului va fi citit într-un textbox). Scrieți un program PHP care preia valorile şirului şi le afişează într-un tabel cu o singură linie, colorând cu fundal roşu deschis acele numere care sunt prime şi cu verde deschis acele numere care sunt pătrate perfecte. Dacă în şirul dat nu sunt fie numere prime, fie numere pătrate perfecte, fie de ambele categorii, să se afişeze un mesaj corespunzător.

15) Se citeşte, prin intermediul unui câmp de tip text al unui formular, un număr natural nenul n, mai mic sau egal cu 40. Pe baza lui n veți genera un alt formular, în care veți citi un şir cu n elemente numere naturale (fiecare element al şirului va fi citit într-un textbox). Scrieți un program

PHP care preia valorile şirului şi le afişează într-un tabel cu o singură linie, colorând cu fundal roşu prima valoare din şirul introdus, care are proprietatea că este palindrom. Să se şteargă apoi această valoare din şir, reafişând, tot sub forma unui tabel cu o singură linie, valorile rămase. Dacă printre valorile introduse nu se află nici un palindrom, se va afişa un mesaj corespunzător.

16) Se citeşte, prin intermediul unui câmp de tip text al unui formular, un număr natural nenul n, mai mic sau egal cu 40. Pe baza lui n veți genera un alt formular, în care veți citi un şir cu n elemente numere naturale (fiecare element al şirului va fi citit într-un textbox). Scrieți un program PHP care preia valorile şirului şi le afişează într-un tabel cu o singură linie, colorând cu fundal galben prima pereche de elemente vecine (consecutive ca poziție) care au proprietatea că NU sunt prime între ele. Să se insereze apoi, între cele două valori, cmmdc-ul lor. Se va reafişa, tot sub forma unui tabel, şirul obținut, în care colorăm tot cu galben elementele perechii, respectiv cu bleu cmmdc-ul inserat. Dacă printre valorile introduse nu se află nici o pereche de elemente vecine cu proprietatea cerută, se va afişa un mesaj corespunzător.

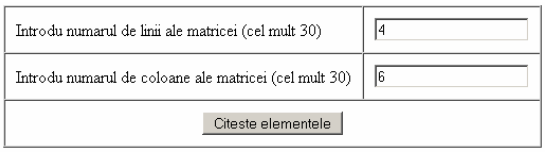
17) Se citeşte, prin intermediul unui câmp de tip text al unui formular, un număr natural nenul n, mai mic sau egal cu 40. Pe baza lui n veți genera un alt formular, în care veți citi un şir cu n elemente numere naturale (fiecare element al şirului va fi citit într-un textbox). Scrieți un program PHP care preia valorile şirului şi le afişează într-un tabel cu o singură linie, colorând cu fundal roşu deschis toate numerele care sunt prime. Să se şteargă apoi din şir toate aceste numere. Se va reafişa, tot sub forma unui tabel, şirul obținut. Dacă printre valorile introduse nu se află nici un număr prim, se va afişa un mesaj corespunzător.

18) Se citeşte, prin intermediul unui câmp de tip text al unui formular, un număr natural nenul n, mai mic sau egal cu 40. Pe baza lui n veți genera un alt formular, în care veți citi un şir cu n elemente numere naturale (fiecare element al şirului va fi citit într-un textbox). Scrieți un program PHP care preia valorile şirului şi le afişează într-un tabel cu o singură linie. Să se insereze apoi între oricare pereche de vecini consecutivi (ca poziție) ai şirului, care au proprietatea că au aceeaşi paritate, media lor aritmetică. Se va reafişa, tot sub forma unui tabel, şirul obținut, colorând cu fundal verde deschis elementele care au fost inserate. Dacă printre valorile introduse nu se află nici o pereche de vecini care să aibă aceeaşi paritate, se va afişa un mesaj corespunzător.

19) **Problemă rezolvată** (**`apl094.html`** + **`apl095.php`** + **`apl096.php`**): Se citesc, prin intermediul a două câmpuri de tip text ale unui formular, două numere naturale nenule, n şi m, mai mici sau egale cu 30. Pe baza lor veți genera un alt formular, în care veți citi elementele unei

matrice cu n linii şi m coloane, numere întregi. Scrieți un program PHP care preia valorile matricei, determină minimul (dacă sunt mai multe, primul dintre ele, în sensul în care se parcurge matricea pe linii, de la stânga la dreapta) şi afişează matricea, colorând acest element cu verde, restul elementelor de pe linia sa cu galben, iar restul elementelor de pe coloana sa cu bleu.

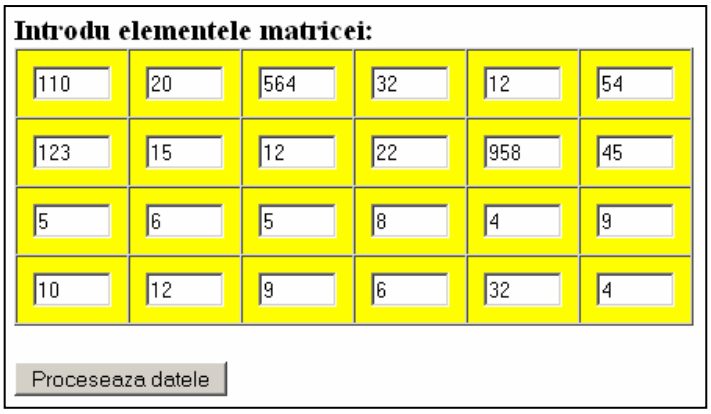
apl094.html




```
<html><body>
<form action="apl095.php" method="post">
<table border="1" cellspacing="0" cellpadding="10">
<tr><td>
Introdu numarul de linii ale matricei (cel mult 30)
<td><input type="text" name="n">
<tr><td>
Introdu numarul de coloane ale matricei (cel mult 30)
<td><input type="text" name="m">
<tr>
<td align="center" colspan="2">
<input type="submit" value="Citeste elementele">
</table></form></body></html>
```

apl095.php

```
<html><body>
<?php
 $n=$_POST['n'];
 $m=$_POST['m'];
 echo '<h3>Introdu elementele matricei:',"\n";
 echo '<form action="apl096.php" method="post">',"\n";
 echo '<table border="1" cellspacing="0" cellpadding="10">',"\n";
 for($i=1;$i<=$n;$i++)
 {
   echo '<tr bgcolor="yellow">';
   for($j=1;$j<=$m;$j++)
    echo '<td><input type="text" size="4" name="a[',$i,'][',$j,']">',"\n";
//deci generam campurile text care se vor
//numi a[1][1], a[1][2], ....
 }
 echo '</table><br>';
 echo '<input type="submit" value="Proceseaza datele">',"\n";
 echo '<input type="hidden" name="n" value="',$n,'">',"\n";
 echo '<input type="hidden" name="m" value="',$m,'">',"\n";
//deci valoarile lui $n si $m le trimitem mai departe
//(pentru ca avem nevoie de ele la prelucrarea datelor
//matricei) prin intermediul unor controale de tipul "hidden"
 echo '</table></form>';
?>
</body></html>
```



apl096.php

```
<html><body>
<?php
  $n=$_POST['n'];
  $m=$_POST['m'];
  $a=$_POST['a'];//in felul asta obtinem
//in variabila $a toata matricea trimisa din cadrul form-ului
//prin intermediul atributelor name=a[1][1], name=a[1][2],...
//determinam mai intii minimul, cu tot cu indicii sai:
  $min=$a[1][1];$imin=1;$jmin=1;
  for($i=1;$i<=$n;$i++)
    for($j=1;$j<=$m;$j++)
      if($a[$i][$j]<$min)
              {
                 $min=$a[$i][$j];
                 $imin=$i;$jmin=$j;
              }
  echo 'Minimul este ',$min,' pe linia ',$imin,' si coloana ',$jmin,'<br>';
  echo '<table border="1" cellspacing="0" cellpadding="10">';
  for($i=1;$i<=$n;$i++)
  {
    echo '<tr>';
    for($j=1;$j<=$m;$j++)
      {
        if($i==$imin&&$j==$jmin)
           echo '<td bgcolor="#00ff00" align="center">';
        else
           if($i==$imin)
              echo '<td bgcolor="#ffff00" align="center">';
           else
               if($j==$jmin)
                  echo '<td bgcolor="#80ffff" align="center">';
               else
                  echo '<td align="center">';
        echo $a[$i][$j];
      }
  }
?>
</table>
</body></html>
```

Minimul este 4 pe linia 3 si coloana 5

110	20	564	32	12	54
123	15	12	22	958	45
5	6	5	8	4	9
10	12	9	6	32	4

20) Se citesc, prin intermediul a două câmpuri de tip text ale unui formular, două numere naturale nenule, n şi m, mai mici sau egale cu 30. Pe baza lor veți genera un alt formular, în care veți citi elementele unei matrice cu n linii şi m coloane, numere întregi. Scrieți un program PHP care preia valorile matricei şi sortează crescător elementele primei linii ale sale prin inteschimbări de coloane. Se vor afişa atât matricea inițială, cât şi matricea finală, ambele în câte un tabel, colorând cu fundal galben elementele primei linii.

21) Se citeşte, prin intermediul unui câmp de tip text al unui formular, un număr natural nenul, n mai mic sau egal cu 30. Pe baza lui veți genera un alt formular, în care veți citi elementele unei matrice pătratice cu n linii şi coloane, numere întregi. Scrieți un program PHP care preia valorile matricei, afişând-o într-un tabel în care:

- elementele de pe diagonala principală sunt colorate cu fundal galben;
- elementele de pe diagonala secundară sunt colorate cu fundal bleu;
- în cazul în care diagonalele se intersectează, elementul de la intersecția lor va fi colorat cu fundal verde.

22) Se citeşte, prin intermediul unui câmp de tip text al unui formular, un număr natural nenul, n mai mic sau egal cu 30. Pe baza lui veți genera un alt formular, în care veți citi elementele unei matrice pătratice cu n linii şi coloane, numere întregi. Scrieți un program PHP care preia valorile matricei, afişând-o într-un tabel în care colorează cu fundal galben elementele pătratului concentric care conține elementul minim al matricei. În cazul în care în matrice sunt mai multe minime, se va lua în considerare primul pe care îl întâlnim, în sensul în care parcurgem matricea pe linii, fiecare linie fiind parcursă de la stânga la dreapta.

23) Se citesc, prin intermediul a două câmpuri de tip text ale unui formular, două numere naturale nenule, n şi m, mai mici sau egale cu 30. Pe baza lor veți genera un alt formular, în care veți citi elementele unei matrice cu n linii şi m coloane, numere întregi. Scrieți un program PHP care preia valorile matricei şi determină cele două linii care conțin minimul respectiv maximul din matrice. În cazul în care sunt mai multe minime, se consideră prima apariție, iar în cazul în care sunt mai mult maxime, se consideră ultima apariție (parcurgând matricea pe linii, şi în cadrul fiecărei linii de la stânga la dreapta). Se vor interschimba cele două linii.

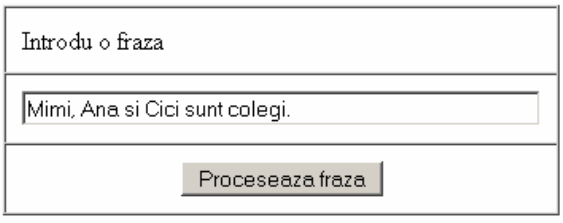
Programul va afişa într-un tabel elementele matricei înainte şi după interschimbare, colorând cu fundal bleu linia care conține minimul, respectiv cu fundal portocaliu linia care conține maximul. Dacă atât minimul cât şi maximul determinate după procedeul de mai sus se află pe aceeaşi linie, se va da un mesaj corespunzător şi se va afişa doar o singură dată matricea.

4.3. Prelucrarea şirurilor de caractere

24) **Problemă rezolvată** (**`apl097.html`** + **`apl098.php`**): Se citeşte, prin intermediul unui câmp de tip text al unui formular, o frază ale cărei cuvinte pot fi separate prin spații, virgule, puncte. Să se scrie un program PHP care să afişeze toate cuvintele care apar în string, în ordine alfabetică, convertite la litere mici, într-un tabel cu o singură coloană.

`apl097.html`

```
<html><body>
<form action="apl098.php" method="post">
<table border="1" cellspacing="0" cellpadding="10">
<tr><td>Introdu o fraza
<tr><td><input type="text" name="s" size="50">
<tr><td align="center">
<input type="submit" value="Proceseaza fraza">
</table>
</form></body></html>
```



apl098.php

```
<html><body>
<?php
  $s=$_POST['s'];
  $n=0;
//separam cuvintele din string cu strtok si le punem
//in sirul a:
  for($p=strtok($s,",. ");$p!==false;$p=strtok(",. "))
      $a[++$n]=strtolower($p);
//le sortam alfabetic
  for($i=1;$i<=$n-1;$i++)
    for($j=$i+1;$j<=$n;$j++)
      if($a[$i]>$a[$j])
          {$aux=$a[$i];$a[$i]=$a[$j];$a[$j]=$aux;}
  echo '<h3>Iata cuvintele in ordine alfabetica:</h3>';
  echo '<table border="1" cellpadding="10" cellspacing="0">';
  for($i=1;$i<=$n;$i++)
     echo '<tr><td>',$a[$i];
?>
</table></body></html>
```

Iata cuvintele in ordine alfabetica:

ana
cici
colegi
mimi
si
sunt

25) Se citeşte, prin intermediul unui câmp text al unui formular, un şir de caractere ce conține mai multe cuvinte, separate prin virgule, spații, puncte. Să se scrie un program PHP care preia string-ul din acest formular şi formează şi afişează un şir ce conține doar cuvintele distincte din şirul dat. Şirul cu cuvintele distincte se va afişa într-un tabel cu o singură coloană.

26) Se citeşte, prin intermediul unui câmp text al unui formular, un şir de caractere ce conține mai multe cuvinte, separate prin virgule, spații, puncte. Să se scrie un program PHP care preia string-ul din acest formular şi afişează toate cuvintele sale într-un tabel cu o singură coloană, colorând cu fundal roşu doar acele cuvinte care sunt palindromice. Daca nici un cuvânt nu este palindromic, se va afişa un mesaj corespunzător.

27) Se citeşte, prin intermediul unui câmp de tip text al unui formular, o dată calendaristică de forma zz/ll/aaaa (z şi l pot avea fie un caracter, iar anul este pozitiv). Să se scrie un program PHP care determină câte zile au trecut de la începutul acelui an până la data respectivă.

28) Se citeşte, prin intermediul unui câmp de tip text al unui formular, o valoare pozitivă nenulă mai mică decât 40. Să se genereze mai întâi, printr-un script PHP, un formular cu n câmpuri de tip text, în care se permite citirea a n şiruri de caractere. Să se afişeze apoi, prin intermediul altui program PHP, toate aceste şiruri de caractere într-un tabel cu o singură coloană, colorând cu fundal roşu şirul cel mai lung. Dacă sunt două sau mai multe şiruri de lungime maximă, se vor colora fundalul tuturor cu roşu.

29) Se citesc, prin intermediul a două câmpuri de tip text ale unui formular, două cuvinte. Scrieți un program PHP care afişează cele două cuvinte introduse şi verifică dacă sunt angrame, adică sunt formate din exact aceleaşi litere, fiecare literă trebuie să apară în fiecare cuvânt de exact acelaşi număr de ori, iar ordinea în care apar poate, evident, să fie diferită.

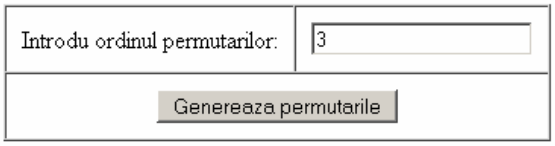
30) Se citeşte, prin intermediul unui câmp de tip text al unui formular, o frază formată din cuvinte separate prin puncte, virgule, spații. Determinați care este litera care apare de cele mai multe ori, precum şi numărul său de apariții. Dacă o mai multe litere apar de acelaşi număr maxim de ori, se vor afişa toate aceste litere. Literele determinate le veți afişa într-un tabel cu o singură coloană.

4.4. Probleme de Backtracking, Divide et Impera, Aplicații ale geometriei analitice plane studiate în cadrul disciplinei matematică, Reprezentări de fractali

31) **Problemă rezolvată** (**`apl099.html`** + **`apl100.php`**): Se citeşte, prin intermediul unui câmp de tip text al unui formular, o valoare pozitivă nenulă n mai mică sau egală cu 7. Să se scrie un program PHP care generează toate permutările de n, afişându-le într-un tabel. Fiecare linie a tabelului va conține elementele unei permutări.

`apl099.html`

```
<html>
 <body>
 <form action="apl100.php" method="post">
 <table border="1" cellpadding="10" cellspacing="0">
 <tr>
   <td>Introdu ordinul permutarilor:
   <td>
     <input type="text" name="n">
  <tr>
   <td align="center" colspan="2">
   <input type="submit" value="Genereaza permutarile">
 </table>
 </form>
 </body>
</html>
```



`apl100.php`

```
<html><body>
<?php
  $n=$_POST['n'];

  function afis()
  {//in functia de afisare, variabilele
   //$x (sirul de generare si $n =
//ordinul permutarilor, le luam globale
    global $n,$x;
    echo '<tr>';
   //afisam deci elementele permutarii
//curente intr-o linie noua a tabelului
    for($i=1;$i<=$n;$i++)
      echo '<td align="center">',$x[$i];
  }
```

1	2	3
1	3	2
2	1	3
2	3	1
3	1	2
3	2	1

```
  function valid($k)
  {//in functia de valid de asemenea $n si $x sunt globale
    global $n,$x;
   //verificam daca elementul curent este diferit de fiecare
   //dintre cele de dinaintea sa
    for($i=1;$i<=$k-1;$i++)
        if($x[$i]==$x[$k])
              return 0;
    return 1;
  }

  function permutari($k)
  {
   //de asemenea $n si $x sunt variabile globale
     global $n,$x;
   //implementam sub forma recursiva functia de tip backtracking
   //ce ne generaza permutarile
     for($x[$k]=1;$x[$k]<=$n;$x[$k]++)
       if(valid($k))
              if($k==$n) afis();
                   else permutari($k+1);
  }

//afisam definitia tabelului "in programul principal"
  echo '<table border="1" cellpadding="10" cellspacing="0">';
//dupa care, prin apelul recursiv, dam drumul la generarea permutarilor
  permutari(1);
//iar la sfirsit inchidem tag-ul tabelului
  echo '</table>';
?>
</body></html>
```

32) Se citeşte, prin intermediul unui câmp de tip text al unui formular, o valoare pozitivă nenulă n mai mică sau egală cu 10. Să se scrie un program PHP care generează toate permutările de n care au proprietatea că pe oricare două poziții vecine (adică la indici consecutivi) se află doar valori de parități diferite, afişându-le într-un tabel. Fiecare linie a tabelului va conține elementele unei permutări.

33) Se citeşte, prin intermediul unui câmp de tip text al unui formular, o valoare pozitivă nenulă n mai mică sau egală cu 10. Să se scrie un program PHP care generează toate permutările de n care au proprietatea că diferența în modul a oricare elemente de pe poziții vecine (adică la indici consecutivi) este cel mult 2, afişându-le într-un tabel. Fiecare linie a tabelului va conține elementele unei permutări.

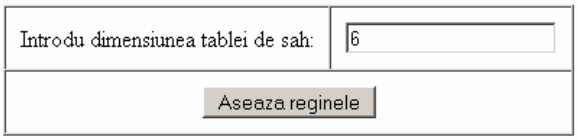
34) Se citesc, prin intermediul a două câmpuri de tip text ale unui formular, două valori pozitive nenule n şi m, astfel încât m ≤ n. Să se scrie un program PHP care generează toate combinările de elemente ale mulțimii {1, 2, .., n} luate câte m. Combinările generate se vor afişa într-un tabel. Fiecare linie a tabelului va conține elementele unei combinări.

35) **Problemă rezolvată** (**`apl101.html`** + **`apl102.php`**): Se citeşte, prin intermediul unui câmp de tip text al unui formular, o valoare pozitivă nenulă n mai mică sau egală cu 8 şi mai mare sau egală cu 4. Să se scrie un program PHP care generează toate posibilitățile de a aranja n regine pe-o tablă de şah n x n astfel încât să nu se atace între ele. Fiecare soluție se va afişa sub forma unui tabel cu n linii şi n coloane, în care celulele sunt colorate alternativ, ca în cazul tablei de şah iar reginele sunt reprezentate printr-o imagine reprezentativă. Soluțiile se vor numerota.

În implementare vom folosi următoarea imagine, queen.gif: 

`apl101.html`

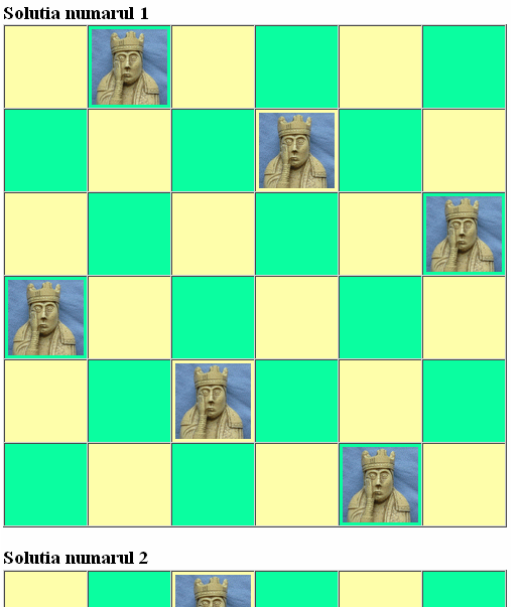
```
<html><body>
<form action="apl102.php" method="post">
<table border="1" cellspacing="0" cellpadding="10">
<tr><td>Introdu dimensiunea tablei de sah:
    <td><input type="text" name="n">
<tr><td colspan="2" align="center">
   <input type="submit" value="Aseaza reginele">
</table>
</form></body></html>
```



`apl102.php`

```
<html><body>
<?php
  $n=$_POST['n'];

  function afish()
  { global $x,$n,$nrsol;
//sirul de generare, $n si variabila in care numaram
//solutiile sunt globale
    echo '<h3>Solutia numarul ',++$nrsol;
    echo '<table border="1" cellspacing="0" cellpadding="3">';
//afisam un tabel cu $n linii si coloane
    for($i=1;$i<=$n;$i++)
    {
       echo '<tr>';
       for($j=1;$j<=$n;$j++)
         {
//in functie de paritatea lui $i+$j stabilim culoarea
//alternam culorile de fundal ale patratelelor, ca pe
//tabla de sah
           if(($i+$j)%2==0) echo '<td bgcolor="#ffffaa">';
                    else echo '<td bgcolor="$aaffaa">';
//iar daca la celula curenta se afla o regina
//atunci reprezentam acea regina prin imaginea queen.gif
           if($x[$i]==$j) echo '<img src="queen.gif">';
//iar in caz ca e goala,punem un spatziu in acea celula
                      else echo ' ';
         }
    }
    echo '</table>';
  }


  function valid($k)
  { global $x;
//in functia de valid verificam faptul ca dama
//de la indicele $k sa nu se atace cu vreuna de dinainte
    for($i=1;$i<=$k-1;$i++)
       if($x[$i]==$x[$k]||abs($x[$k]-$x[$i])==$k-$i)
                                        return 0;
    return 1;
  }
```

```
  function dame($k)
  {
    global $x,$n;
//functia dame implementeaza backtracking-ul recursiv
    for($x[$k]=1;$x[$k]<=$n;$x[$k]++)
            if(valid($k))
                  if($k==$n) afish();
                       else dame($k+1);
  }

  $nrsol=0;
  dame(1);
?>
</body></html>
```

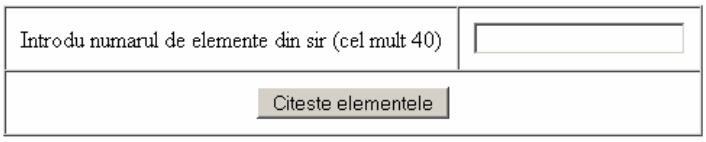


36) Se citeşte, prin intermediul unui câmp de tip text al unui formular, o valoare pozitivă nenulă n mai mică sau egală cu 8 şi mai mare sau egală cu 4. Să se scrie un program PHP care generează doar 10 posibilități ce a aşeza n ture pe-o tablă de şah n x n astfel încât să nu se atace între ele. Fiecare soluție se va afişa sub forma unui tabel cu n linii şi n coloane, în care celulele sunt colorate alternativ, ca în cazul tablei de şah iar turele sunt reprezentate printr-o imagine reprezentativă. Soluțiile se vor numerota.

37) Se citeşte, prin intermediul unui câmp de tip text al unui formular, o valoare pozitivă nenulă n mai mică sau egală cu 8 şi mai mare sau egală cu 4. Să se scrie un program PHP care generează doar 10 posibilități ce a aşeza n regi pe-o tablă de şah n x n astfel încât pe fiecare linie de pe tablă să se găsească exact un rege, iar regii să nu se atace între ei. Fiecare soluție se va afişa sub forma unui tabel cu n linii şi n coloane, în care celulele sunt colorate alternativ, ca în cazul tablei de şah iar turele sunt reprezentate printr-o imagine reprezentativă. Soluțiile se vor numerota.

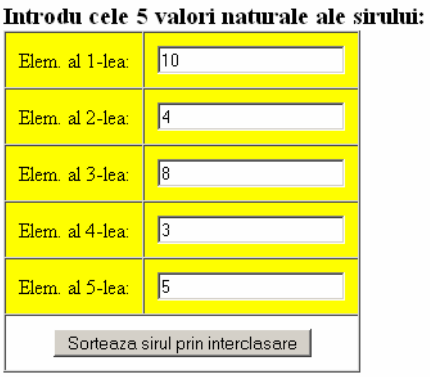
38) **Problemă rezolvată** (**`apl103.html`** + **`apl104.php`** + **`apl105.php`**): Se citeşte, prin intermediul unui câmp de tip text al unui formular, o valoare pozitivă nenulă n, cel mult egală cu 40. Prin intermediul altui formular cu n câmpuri de tip text se citesc elementele întregi ale unui şir. Scrieți un program PHP care afişează mai întâi elementele citite, le sortează crescător prin interclasare (deci folosind metoda divide et impera) şi le afişează şi după sortare. Elementele se vor afişa într-un tabel cu o singură linie.

apl103.html




```
<html><body>
<form action="apl104.php" method="post">
<table border="1"
  cellspacing="0" cellpadding="10">
<tr><td>
Introdu numarul de elemente din sir (cel mult 40)
<td><input type="text" name="n">
<tr>
<td align="center" colspan="2">
<input type="submit" value="Citeste elementele">
</table></form></body></html>
```

apl104.php




```
<html><body>
<?php
 $n=$_POST['n'];
 echo '<h3>Introdu cele ',$n,
      ' valori naturale ale sirului:',"\n";
 echo '<form action="apl105.php" method="post">',"\n";
 echo '<table border="1" cellspacing="0" cellpadding="10">',"\n";
 for($i=1;$i<=$n;$i++)
    echo '<tr bgcolor="yellow"><td>Elem. al ',$i,'-lea: ',
         '<td><input type="text" name="a[',$i,']">',"\n";
 echo '<tr><td colspan="2" align="center">',"\n";
 echo '<input type="submit" value="Sorteaza sirul prin interclasare">',"\n";
 echo '<input type="hidden" name="n" value="',$n,'">',"\n";
 echo '</table></form>';
?>
</body></html>
```

apl105.php

```
<html><body>
<?php
  $n=$_POST['n'];
  $a=$_POST['a'];

function afis($a,$n)
{
  echo '<table border="1" cellspacing="0" cellpadding="10">';
  echo '<tr bgcolor="yellow">';
  for($i=1;$i<=$n;$i++)
    echo '<td align="center">',$a[$i];
  echo '</table>';
}

  echo '<h3>Iata sirul initial:</h3>';
  afis($a,$n);

//mai jos este functia care interclaseaza din sirul
//$a, bucatile dintre indicii $l..$m respectiv $m+1..$r
//punind rezultatul la loc in sirul $a, incepind de la
//indicele $l
  function intercl(&$a,$l,$m,$r)
  {
    $k=$l;$i=$l;$j=$m+1;
    while($i<=$m&&$j<=$r)
      if($a[$i]<$a[$j])
                  $c[$k++]=$a[$i++];
            else
                  $c[$k++]=$a[$j++];
    while($i<=$m) $c[$k++]=$a[$i++];
    while($j<=$r) $c[$k++]=$a[$j++];
    for($i=$l;$i<=$r;$i++) $a[$i]=$c[$i];
  }
```

```
//si in fine, mai jos este functia care
// realizeaza sortarea prin
//interclasare:
  function merge_sort(&$a,$l,$r)
  {
    if($l<$r)
       {
//se calculeaza $m=indicele mijlocului dintre $l si $r
         $m=(int)(($l+$r)/2);
//se sorteaza recursiv partea dintre $l si $m
         merge_sort($a,$l,$m);
//apoi partea dintre $m+1 si $r
         merge_sort($a,$m+1,$r);
//si, in fine, cele doua se interclaseaza:
         intercl($a,$l,$m,$r);
       }
  }

  merge_sort($a,1,$n);
  echo '<h3>Iata sirul final, obtinut dupa sortarea sa prin interclasare:</h3>';
  afis($a,$n);
?>
```

Iata sirul initial:

10	4	8	3	5

Iata sirul final, obtinut dupa sortarea sa prin interclasare:

3	4	5	8	10

39) **Problemă rezolvată** (**`apl106.html`** + **`apl107.php`**): Se citesc, prin intermediul unor câmpuri de tip text ale unui formular, coordonatele a trei puncte ce reprezintă vârfurile unui triunghi. Abscisele sunt cuprinse între 0 şi 639 iar ordonatele între 0 şi 479. Să se creeze o imagine de tip PNG în care reprezentaţi triunghiul şi cele trei înălţimi ale sale.

Suportul matematic necesar rezolvării:

Fie cele trei vârfuri ale triunghiului $A(x_a, y_a)$, $B(x_b, y_b)$, $C(x_c, y_c)$.

Pentru a scrie ecuaţia înălţimii care trece prin vârful A, vom considera ecuaţia dreptei care trece printr-un punct dat şi are panta cunoscută:

$(y - y_A) = m \cdot (x - x_A)$

Înălţimea, fiind perpendiculară pe segmentul BC, are panta egală cu -1/m', unde m' = panta dreptei BC. Aşadar: $m' = \frac{y_C - y_B}{x_C - x_B}$ de unde rezultă că $m = -\frac{x_C - x_B}{y_C - y_B}$ şi deci ecuaţia înălţimii care trece prin vârful A este $(y - y_A) = -\frac{x_C - x_B}{y_C - y_B}(x - x_A)$. După efectuarea calculelor, obţinem:

$$y(y_C - y_B) + x(x_C - x_B) = y_A(y_C - y_B) + x_A(x_C - x_B).$$

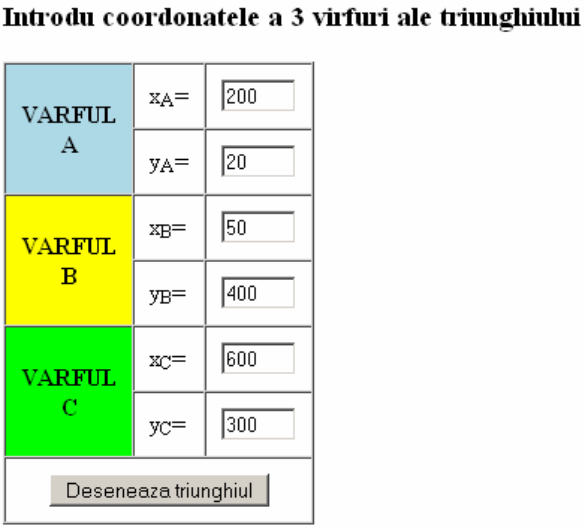
Pentru a determina apoi piciorul unei înălţimi, este suficient să rezolvăm sistemul format din ecuaţia înălţimii şi ecuaţia dreptei suport al laturii corespunzătoare.

Pentru a scrie ecuaţia unei laturi a triunghiului (fie, în cazul nostru, latura BC) cel mai la îndemână este să ne folosim de formula:

$$\begin{vmatrix} x & y & 1 \\ x_B & y_B & 1 \\ x_C & y_C & 1 \end{vmatrix} = 0.$$

apl106.html

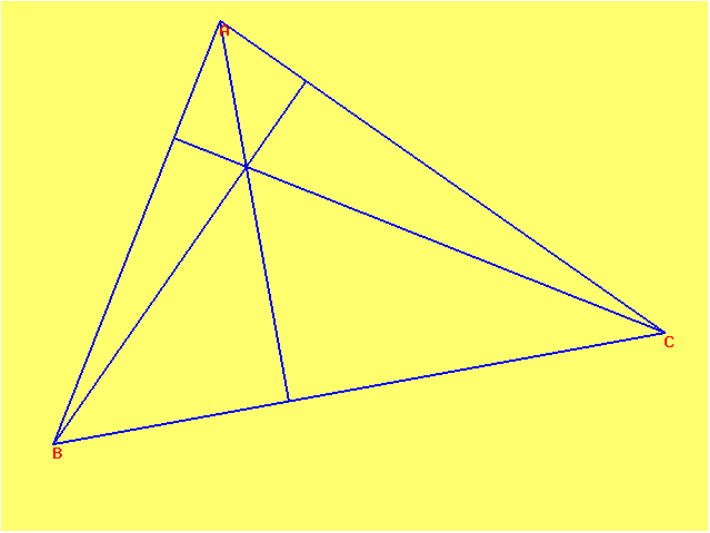
```
<html><body>
<form action="apl107.php" method="post">
<h3>Introdu coordonatele a 3 virfuri ale triunghiului</h3>
<table border="1" cellspacing="0" cellpadding="10">
<tr>
  <th align="center" rowspan="2" bgcolor="lightblue">
  VARFUL<br>A
  <td>x<sub>A</sub>=
  <td><input type="text" name="xa" size="4">
<tr>
  <td>y<sub>A</sub>=
  <td><input type="text" name="ya" size="4">
<tr>
  <th align="center" rowspan="2" bgcolor="yellow">
   VARFUL<br>B
  <td>x<sub>B</sub>=
  <td><input type="text" name="xb" size="4">
<tr>
  <td>y<sub>B</sub>=
  <td><input type="text" name="yb" size="4"><tr>
  <th align="center" rowspan="2" bgcolor="lime">
  VARFUL<br>C
  <td>x<sub>C</sub>=
  <td><input type="text" name="xc" size="4">
<tr>
  <td>y<sub>C</sub>=
  <td><input type="text" name="yc" size="4">
<tr><td colspan="3" align="center">
  <input type="submit" value="Deseneaza triunghiul">
</table>
</form>
</body></html>
```



apl107.php

```
<?php
  header("Content-type: image/png");
  $im=imagecreatetruecolor(640,480);
  $y=imagecolorallocate($im,255,255,110);
  imagefilledrectangle($im,0,0,639,479,$y);
  $xa=$_POST['xa'];$ya=$_POST['ya'];
  $xb=$_POST['xb'];$yb=$_POST['yb'];
  $xc=$_POST['xc'];$yc=$_POST['yc'];
  imagesetthickness($im,2);
  $b=imagecolorallocate($im,0,0,255);
  imageline($im,$xa,$ya,$xb,$yb,$b);
  imageline($im,$xa,$ya,$xc,$yc,$b);
  imageline($im,$xc,$yc,$xb,$yb,$b);
  $r=imagecolorallocate($im,255,0,0);
//determinam $a1,$b1,$c1 coeficientii
//ecuatiei inaltimii corespunzatoare vf.A
  $a1=$xc-$xb;$b1=$yc-$yb;
  $c1=$xa*($xc-$xb)+$ya*($yc-$yb);
//determinam $a2, $b2, $c2 coeficientii
//dreptei suport a segmentului bc:
  $a2=$yb-$yc;
  $b2=-$xb+$xc;
  $c2=$yb*$xc-$xb*$yc;
//aflam solutia sistemului format de cele doua:
  $xha=($c1*$b2-$c2*$b1)/($a1*$b2-$a2*$b1);
  $yha=($c1*$a2-$c2*$a1)/($b1*$a2-$a1*$b2);
//procedam analog si cu celelalte inaltimi:
//cea care pleaca din c:
  $a3=$xb-$xa;$b3=$yb-$ya;
  $c3=$xc*($xb-$xa)+$yc*($yb-$ya);
  $a4=$ya-$yb;
  $b4=-$xa+$xb;
  $c4=$ya*$xb-$xa*$yb;
  $xhc=($c3*$b4-$c4*$b3)/($a3*$b4-$a4*$b3);
  $yhc=($c3*$a4-$c4*$a3)/($b3*$a4-$a3*$b4);
```

```
//si in fine cu cea care pleaca din b:
  $a5=$xc-$xa;$b5=$yc-$ya;
  $c5=$xb*($xc-$xa)+$yb*($yc-$ya);
  $a6=$ya-$yc;
  $b6=-$xa+$xc;
  $c6=$ya*$xc-$xa*$yc;
  $xhb=($c5*$b6-$c6*$b5)/($a5*$b6-$a6*$b5);
  $yhb=($c5*$a6-$c6*$a5)/($b5*$a6-$a5*$b6);
//desenam inaltimile
  imageline($im,$xa,$ya,$xha,$yha,$b);
  imageline($im,$xc,$yc,$xhc,$yhc,$b);
  imageline($im,$xb,$yb,$xhb,$yhb,$b);
//si in fine, punem literele
//virfurilor triunghiului
  imagestring($im,5,$xa,$ya,'A',$r);
  imagestring($im,5,$xb,$yb,'B',$r);
  imagestring($im,5,$xc,$yc,'C',$r);
  imagepng($im);
  imagedestroy($im);
?>
```



40) **Problemă rezolvată** (**apl108.html** + **apl109.php**): Se citesc, prin intermediul unor câmpuri de tip text ale unui formular, coordonatele a trei puncte ce reprezintă vârfurile unui triunghi. Abscisele sunt cuprinse între 0 şi 639 iar ordonatele între 0 şi 479. Să se creeze o imagine de tip PNG în care reprezentaţi triunghiul, cele trei bisectoare ale sale şi cercul înscris în triunghi.

Suportul matematic necesar rezolvării:

Fie triunghiul de vârfuri $A(x_A,y_A)$, $B(x_B,y_B)$, $C(x_C,y_C)$, în care notăm cu l_a, l_b, l_c lungimile laturilor opuse vârfurilor A, B respectiv C. Fie $A'(x'_A,y'_A)$ piciorul bisectoarei din A pe latura BC.

Aplicând teorema bisectoarei, avem $\frac{AB}{AC}=\frac{A'B}{A'C}$.

O dată ce cunoaştem valoarea raportului, dacă proiectăm membrul drept al egalităţii pe axele OX respectiv OY, se păstrează proporţionalitatea, deci obţinem următoarele relaţii care ne permit calculul lui x'_A şi y'_A:

$\frac{l_C}{l_B}=\frac{x'_A-x_B}{x_C-x'_A}$ de unde deducem $x'_A=\frac{x_Bl_B+x_Cl_C}{l_B+l_C}$. Analog $y'_A=\frac{y_Bl_B+y_Cl_C}{l_B+l_C}$.

Ştiind coordonatele picioarelor bisectoarelor, putem foarte lesne obţine centrul cercului înscris în triunghi prin intersecţia a două dintre ele.

apl108.html

```
<html><body>
<form action="apl109.php" method="post">
<h3>Introdu coordonatele a 3 virfuri ale triunghiului</h3>
<table border="1" cellspacing="0" cellpadding="10">
<tr>
  <th align="center" rowspan="2" bgcolor="lightblue">
  VARFUL<br>A
  <td>x<sub>A</sub>=
  <td><input type="text" name="xa" size="4">
<tr>
  <td>y<sub>A</sub>=
  <td><input type="text" name="ya" size="4">
<tr>
  <th align="center" rowspan="2" bgcolor="yellow">
   VARFUL<br>B
```

Introdu coordonatele a 3 virfuri ale triunghiului

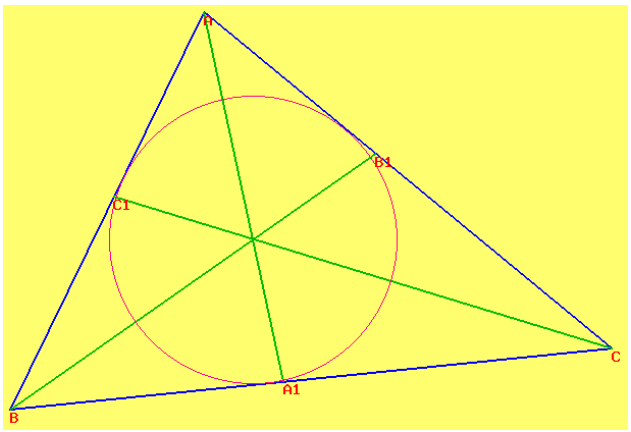
VARFUL A	x_A=	200
	y_A=	10
VARFUL B	x_B=	10
	y_B=	400
VARFUL C	x_C=	600
	y_C=	340
Deseneaza triunghiul		

```
  <td>x<sub>B</sub>=
  <td><input type="text" name="xb" size="4">
<tr>
  <td>y<sub>B</sub>=
  <td><input type="text" name="yb" size="4"><tr>
  <th align="center" rowspan="2" bgcolor="lime">VARFUL<br>C
  <td>x<sub>C</sub>=
  <td><input type="text" name="xc" size="4">
<tr>
  <td>y<sub>C</sub>=
  <td><input type="text" name="yc" size="4">
<tr><td colspan="3" align="center">
  <input type="submit" value="Deseneaza triunghiul">
</table>
</form>
</body></html>
```

apl109.php

```
<?php
  header("Content-type: image/png");
  $im=imagecreatetruecolor(640,480);
  $y=imagecolorallocate($im,255,255,110);
  imagefilledrectangle($im,0,0,639,479,$y);
  $xa=$_POST['xa'];$ya=$_POST['ya'];
  $xb=$_POST['xb'];$yb=$_POST['yb'];
  $xc=$_POST['xc'];$yc=$_POST['yc'];
  imagesetthickness($im,2);
  $b=imagecolorallocate($im,0,0,255);
  imageline($im,$xa,$ya,$xb,$yb,$b);
  imageline($im,$xa,$ya,$xc,$yc,$b);
  imageline($im,$xc,$yc,$xb,$yb,$b);
  $r=imagecolorallocate($im,255,0,0);
  $g=imagecolorallocate($im,0,188,0);
//definim o functie care sa ne calculeze distanta dintre
//doua puncte de coordonate date

  function dist($x1,$y1,$x2,$y2)
  {
    return sqrt(($x1-$x2)*($x1-$x2)+($y1-$y2)*($y1-$y2));
  }
//si definim o functie care sa ne calculeze coordonatele
//picioarei unei bisectoare:

  function coordbis($xa,$ya,$xb,$yb,$xc,$yc,&$x1a,&$y1a)
  {//vom calcula coordonatele piciorului bisectoarei ce pleaca
//din virful A:
   $lb=dist($xa,$ya,$xc,$yc);
   $lc=dist($xa,$ya,$xb,$yb);
   $x1a=($xb*$lb+$xc*$lc)/($lb+$lc);
   $y1a=($yb*$lb+$yc*$lc)/($lb+$lc);
  }
//definim si o functie care sa determine ecuatia unei drepte
//care trece prin doua puncte de coordonate date:
//functia va da coeficientii ecuatiei ax + by + c = 0
//care trece prin punctele ($x1,$y1) si ($x2,$y2)

function ec2puncte($x1,$y1,$x2,$y2,&$a,&$b,&$c)
{
  $a=$y1-$y2;
  $b=-$x1+$x2;
  $c=$x1*$y2-$y1*$x2;
}
//si o functie care, date fiind doua ecuatii de drepte
//in forma ax+by+c=0 determina punctul de intersectie

function inters($a1,$b1,$c1,$a2,$b2,$c2,&$x,&$y)
{
  $x=($b1*$c2-$b2*$c1)/($a1*$b2-$b1*$a2);
  $y=($a1*$c2-$a2*$c1)/($b1*$a2-$a1*$b2);
}
```

```
//calculam picioarele celor 3 inaltimi si le desenam:
  coordbis($xa,$ya,$xb,$yb,$xc,$yc,$x1a,$y1a);
  coordbis($xb,$yb,$xa,$ya,$xc,$yc,$x1b,$y1b);
  coordbis($xc,$yc,$xb,$yb,$xa,$ya,$x1c,$y1c);
  imageline($im,$xa,$ya,$x1a,$y1a,$g);
  imageline($im,$xb,$yb,$x1b,$y1b,$g);
  imageline($im,$xc,$yc,$x1c,$y1c,$g);
//calculam ecuatiile a doua inaltimi, de forma ax + by = c
  ec2puncte($xa,$ya,$x1a,$y1a,$a1,$b1,$c1);
  ec2puncte($xb,$yb,$x1b,$y1b,$a2,$b2,$c2);
//si determinam originea centrului cercului
//inscris intersectindu-le:
  inters($a1,$b1,$c1,$a2,$b2,$c2,$xo,$yo);
//determinam lungimea razei cercului, ca si
//inaltime a triunghiului cu virful in
//centrul cercului inscris si baza una dintre
//laturi
  $s=0.5*abs($xa*$yb+$xb*$yo+$xo*$ya
            -$xo*$yb-$xa*$yo-$xb*$ya);
  $lc=dist($xa,$ya,$xb,$yb);
  $raza=2*$s/$lc;$diam=2*$raza;
  $mg=imagecolorallocate($im,255,0,140);
  imageellipse($im,$xo,$yo,$diam,$diam,$mg);
//si in fine, punem literele virfurilor
//triunghiului
  imagestring($im,5,$xa,$ya,'A',$r);
  imagestring($im,5,$xb,$yb,'B',$r);
  imagestring($im,5,$xc,$yc,'C',$r);
  imagestring($im,5,$x1a,$y1a,'A1',$r);
  imagestring($im,5,$x1b,$y1b,'B1',$r);
  imagestring($im,5,$x1c,$y1c,'C1',$r);
  imagepng($im);
  imagedestroy($im);
?>
```



41) Se citesc, prin intermediul unor câmpuri de tip text ale unui formular, coordonatele a trei puncte ce reprezintă vârfurile unui triunghi. Abscisele sunt cuprinse între 0 şi 639 iar ordonatele între 0 şi 479. Să se creeze o imagine de tip PNG în care reprezentați triunghiul, cercul său circumscris şi segmentele de pe mediatoare care unesc picioarele lor pe laturile triunghiului cu centrul cercului circumscris.

42) Se citesc, prin intermediul unor câmpuri de tip text ale unui formular, coordonatele a patru puncte ce reprezintă vârfurile unui patrulater convex ABCD. Abscisele sunt cuprinse între 0 şi 639 iar ordonatele între 0 şi 479, punctele fiind date în ordine (deci AB, BC, CD, AD sunt laturile poligonului convex). Să se deseneze triunghiurile ABD, respectiv BCD, precum şi cercurile lor circumscrise, precum şi razele care unesc centrele fiecărui cerc cu vârfurile triunghiului corespunzător.

43) Se citesc, prin intermediul unor câmpuri de tip text ale unui formular, coordonatele a trei puncte ce reprezintă vârfurile unui triunghi. Abscisele sunt cuprinse între 0 şi 639 iar ordonatele între 0 şi 479. Să se creeze o imagine de tip PNG în care reprezentați triunghiul, fie el ABC, cele 3 înălțimi AA', BB', CC' şi cercul circumscris patrulaterului inscriptibil OA'B'C, unde O este ortocentrul triunghiului.

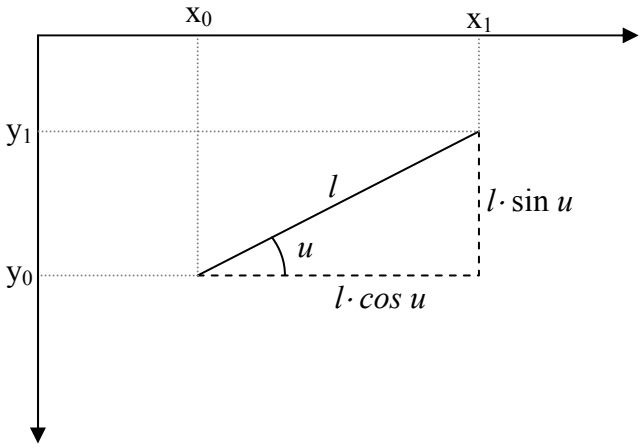
44) **Problemă rezolvată** (**apl110.html** + **apl111.php**): Se citeşte, prin intermediul unui câmp de tip text al unui formular, un număr natural n, cel puțin 3 şi cel mult 40. Să se creeze o imagine PNG de dimensiuni 640x480, în care reprezentați un poligon regulat cu n laturi, având centrul la coordonatele (320,240) şi raza cercului circumscris 200.

Suportul matematic necesar rezolvării:

Ne vom folosi de coordonatele polare: Dat fiind un punct de coordonate (x_0,y_0) din care pleacă un segment de lungime *l* sub un unghi *u*, celălalt capăt al segmentului va avea coordonatele

$x_1=x_0+l\cdot\cos u$

$y_1=y_0-l\cdot\sin u$ (semnul "–" se datorează orientării inverse a axei OY):

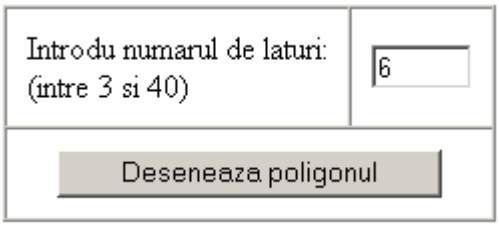


Să ne reamintim în primul rând că funcțiile trigonometrice ale limbajului PHP, ca de altfel ale multor alte limbaje de programare, lucrează în radiani.

În cazul de față, pentru a obține coordonatele vârfurilor poligonului cu n laturi, vom considera unghiul la centru $u=\frac{2\pi}{n}$ şi vom "plimba" un segment de lungime r având un capăt fixat în centrul centrului şi celălalt capăt mobil, la unghiurile 0, u, 2·u, 3·u, ... (n-1)·u. Astfel, coordonatele capătului mobil ne vor da tocmai coordonatele vârfurilor poligonului.

apl110.html

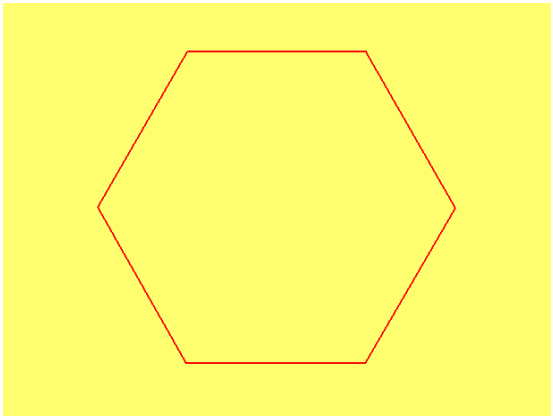
```
<html><body>
<form action="apl111.php" method="post">
<table border="1" cellspacing="0" cellpadding="10">
<tr><td>Introdu numarul de laturi:<br>(intre 3 si 40)
    <td><input type="text" name="n" size="4">
<tr><td colspan="2" align="center">
 <input type="submit" value="Deseneaza poligonul">
</table>
</form></body></html>
```



apl111.php

```
<?php
  header("Content-type: image/png");
  $im=imagecreatetruecolor(640,480);
  $y=imagecolorallocate($im,255,255,110);
  imagefilledrectangle($im,0,0,639,479,$y);
  $n=$_POST['n'];
  $red=imagecolorallocate($im,255,0,0);
  imagesetthickness($im,2);
```

```
  $xc=320;$yc=240;$r=200;
  $u=2*pi()/$n;//u=unghiul la centru
  for($i=0;$i<=$n-1;$i++)
  {//pe baza xc, yc, r, si unghiuri
   //variabile i*u calculam coordonatele
   //a 2 vf. vecine ale poligonului
   $x1=$xc+$r*cos($i*$u);
   $y1=$yc-$r*sin($i*$u);
   $x2=$xc+$r*cos(($i+1)*$u);
   $y2=$yc-$r*sin(($i+1)*$u);
   imageline($im,$x1,$y1,$x2,$y2,$red);
  }
  imagepng($im);
  imagedestroy($im);
?>
```

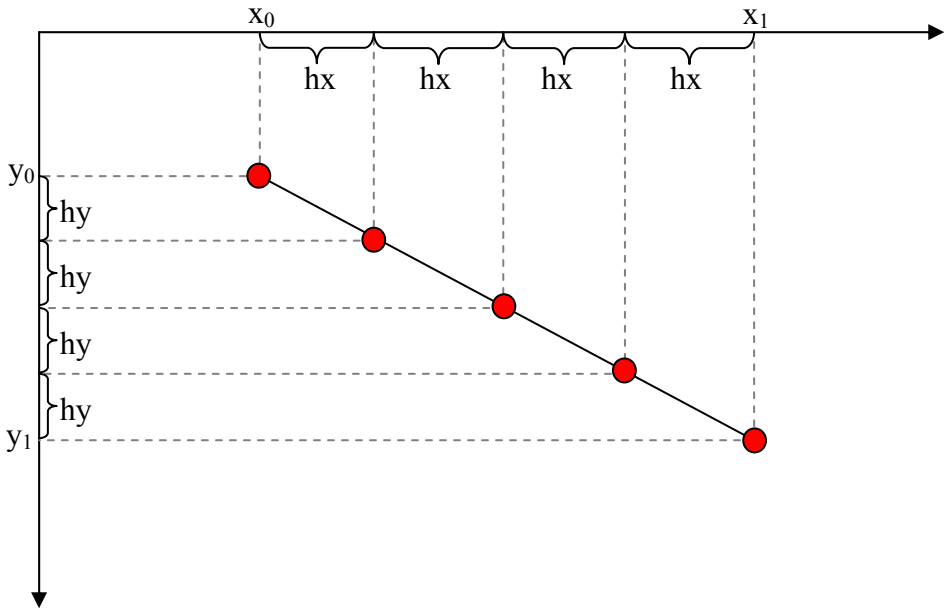


45) Generați o imagine PNG de dimensiuni 640 x 480 în care să creați un model de tip fagure, format din hexagoane. Dimensiunea recomandată a unui hexagon este de latură 40. Poziționarea hexagoanelor în cadrul imaginii rămâne la latitudinea programatorului.

46) **Problemă rezolvată** (**`apl112.html`** + **`apl113.php`**): Se citesc, prin intermediul unui formular, coordonatele a două puncte şi un număr natural m, cuprins între 2 şi 40. Punctele vor avea abscisa cuprinsă între 0 şi 639 iar ordonata între 0 şi 479. Împărțiți segmentul respectiv în m părți congruente. Desenați segmentul, marcând totodată prin cerculețe punctele obținute prin împărțirea în cele m părți congruente.

Suportul matematic necesar rezolvării:

Împărțind segmentul în m părți congruente, proiecția lor pe axele de coordonate va consta de asemenea în segmente congruente:



Fie segmentul dintre punctele de coordonate (x_0,y_0) şi (x_1,y_1). Notăm cu hx respectiv cu hy lungimile proiecțiilor segmentelor congruente care se obțin, pe axa OX respectiv pe axa OY.

Avem: $hx = \frac{x_1 - x_0}{m}$, respectiv $hy = \frac{y_1 - y_0}{m}$.

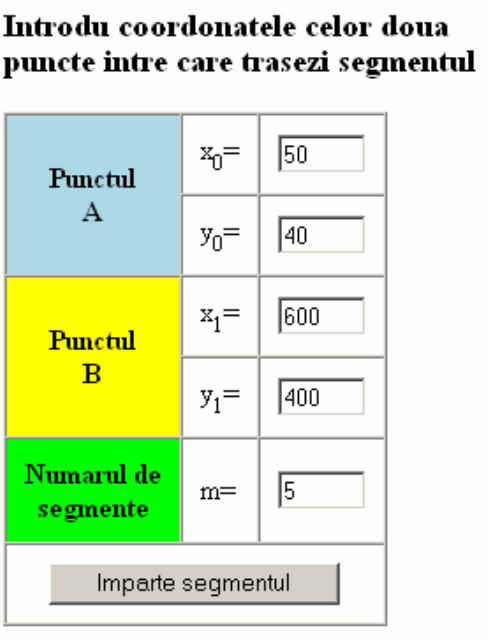
Pe baza lor, coordonatele punctelor intermediare care se obțin sunt:

(x_0+hx,y_0+hy), $(x_0+2\cdot hx,y_0+2\cdot hy)$, $(x_0+3\cdot hx,y_0+3\cdot hy)$, ... $(x_0+(m-1)\cdot hx,y_0+(m-1)\cdot hy)$

sau, pe scurt: $(x_0 + i \cdot hx, y_0 + i \cdot hy)$, cu $i = \overline{0,m}$ dacă dorim şi capetele, sau doar cu $i = \overline{1,m-1}$ dacă dorim doar punctele intermediare.

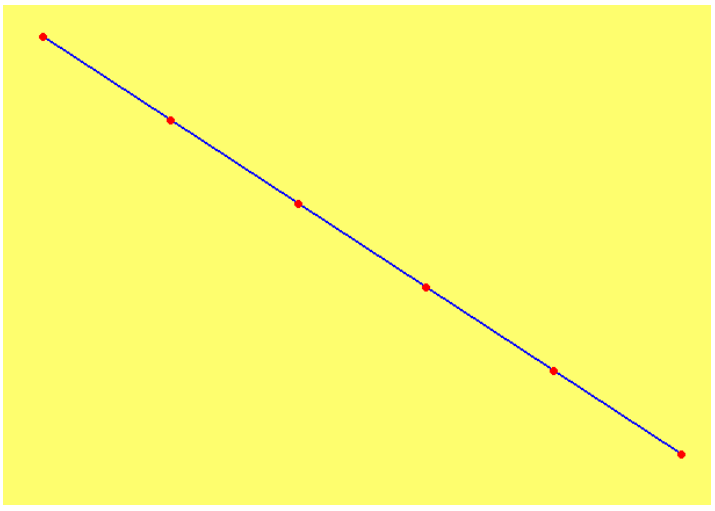
apl112.html

```
<html><body>
<form action="apl113.php" method="post">
<h3>Introdu coordonatele celor doua
puncte intre care trasezi segmentul</h3>
<table border="1" cellspacing="0" cellpadding="10">
<tr>
  <th align="center" rowspan="2" bgcolor="lightblue">
  Punctul<br>A
  <td>x<sub>0</sub>=
  <td><input type="text" name="x0" size="4">
<tr>
  <td>y<sub>0</sub>=
  <td><input type="text" name="y0" size="4">
<tr>
  <th align="center" rowspan="2" bgcolor="yellow">
  Punctul<br>B
  <td>x<sub>1</sub>=
  <td><input type="text" name="x1" size="4">
<tr>
  <td>y<sub>1</sub>=
  <td><input type="text" name="y1" size="4"><tr>
  <th align="center" bgcolor="lime">
  Numarul de<br>segmente
  <td>m=
  <td><input type="text" name="m" size="4">
<tr><td colspan="3" align="center">
  <input type="submit" value="Imparte segmentul">
</table>
</form>
</body></html>
```



apl113.php

```
<?php
  header("Content-type: image/png");
  $im=imagecreatetruecolor(640,480);
  $y=imagecolorallocate($im,255,255,110);
  imagefilledrectangle($im,0,0,639,479,$y);
  $m=$_POST['m'];
  $red=imagecolorallocate($im,255,0,0);
  imagesetthickness($im,2);
  $x0=$_POST['x0'];$y0=$_POST['y0'];
  $x1=$_POST['x1'];$y1=$_POST['y1'];
  $blue=imagecolorallocate($im,0,0,255);
  imageline($im,$x0,$y0,$x1,$y1,$blue);
//calculam lungimile proiectiilor
//segmentelor pe cele doua axe:
  $hx=($x1-$x0)/$m;
  $hy=($y1-$y0)/$m;
//si desenam cerculete in punctele obtinute
  for($i=0;$i<=$m;$i++)
       imagefilledellipse($im,$x0+$i*$hx,$y0+$i*$hy,6,6,$red);
  imagepng($im);
  imagedestroy($im);
?>
```



46) **Problemă rezolvată** (**apl114.html** + **apl115.php**): Se citesc, prin intermediul unui formular, coordonatele a trei puncte A, B şi C şi un număr natural m, cuprins între 2 şi 40. Punctele au abscisa cuprinsă între 0 şi 639 iar ordonata între 0 şi 479. Împărțiți ambele segmente, AB şi BC în m părți congruente, dinspre A către B în cazul primului respectiv dinspre B înspre C în cazul celui de-al doilea. Uniți primul punct obținut astfel de pe AB cu primul punct obținut astfel de pe BC, al doilea punct de pe AB cu al doilea punct de pe BC, ş.a.m.d.

apl114.html

```
<html><body>
<form action="apl115.php" method="post">
<h3>Introdu coordonatele a 3 virfuri ale triunghiului</h3>
<table border="1" cellspacing="0" cellpadding="10">
<tr>
  <th align="center" rowspan="2" bgcolor="lightblue">
  VARFUL<br>A
  <td>x<sub>A</sub>=
  <td><input type="text" name="xa" size="4">
<tr>
  <td>y<sub>A</sub>=
  <td><input type="text" name="ya" size="4">
<tr>
  <th align="center" rowspan="2" bgcolor="yellow">
  VARFUL<br>B
  <td>x<sub>B</sub>=
  <td><input type="text" name="xb" size="4">
<tr><td>y<sub>B</sub>=
  <td><input type="text" name="yb" size="4"><tr>
  <th align="center" rowspan="2" bgcolor="lime">
  VARFUL<br>C
  <td>x<sub>C</sub>=
  <td><input type="text" name="xc" size="4">
<tr><td>y<sub>C</sub>=
  <td><input type="text" name="yc" size="4">
<tr>
  <th bgcolor="red">Numarul de segmente<td>m=
  <td><input type="text" name="m" size="4">
<tr><td colspan="3" align="center">
  <input type="submit" value="Deseneaza">
</table>
</form>
</body></html>
```

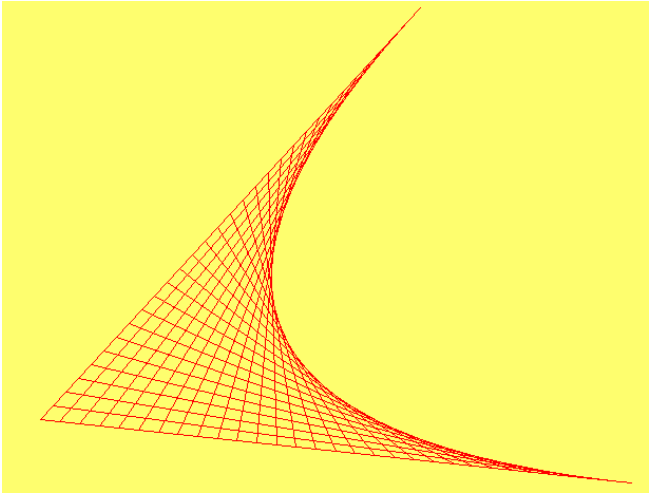
Introdu coordonatele a 3 virfuri ale triunghiului

VARFUL A	x_A=	400
	y_A=	10
VARFUL B	x_B=	40
	y_B=	400
VARFUL C	x_C=	600
	y_C=	460
Numarul de segmente	m=	30

Deseneaza

apl115.php

```
<?php
  header("Content-type: image/png");
  $im=imagecreatetruecolor(640,480);
  $y=imagecolorallocate($im,255,255,110);
  imagefilledrectangle($im,0,0,639,479,$y);
  $m=$_POST['m'];
  $red=imagecolorallocate($im,255,0,0);
  $xa=$_POST['xa'];$ya=$_POST['ya'];
  $xb=$_POST['xb'];$yb=$_POST['yb'];
  $xc=$_POST['xc'];$yc=$_POST['yc'];
  $blue=imagecolorallocate($im,0,0,255);
  imageline($im,$x0,$y0,$x1,$y1,$blue);
  $hx1=($xb-$xa)/$m;
  $hy1=($yb-$ya)/$m;
  $hx2=($xc-$xb)/$m;
  $hy2=($yc-$yb)/$m;

  for($i=0;$i<=$m;$i++)
       imageline($im,$xa+$i*$hx1,$ya+$i*$hy1,$xb+$i*$hx2,$yb+$i*$hy2,$red);
  imagepng($im);
  imagedestroy($im);
?>
```



47) Se citesc, prin intermediul unui formular, coordonatele a trei puncte A, B şi C şi un număr natural m, cuprins între 2 şi 40. Punctele au abscisa cuprinsă între 0 şi 639 iar ordonata între 0 şi 479. Să se împartă atât latura AB (dinspre A înspre B) cât şi latura AC (dinspre A înspre C) în m părți congruente, şi să se unească punctele obținute, formând astfel n segmente paralele echidistante la BC.

48) Se citesc, prin intermediul unui formular, coordonatele a trei puncte A, B şi C şi un număr natural m, cuprins între 2 şi 40. Punctele au abscisa cuprinsă între 0 şi 639 iar ordonata între 0 şi 479. Să se împartă atât latura BC în m părți congruente, şi să se unească vârful A cu fiecare dintre punctele obținute.

49) **Problemă rezolvată** (**`apl116.html`** + **`apl117.php`**): Se citeşte, prin intermediul unui câmp text al unui formular, un număr natural cuprins între 1 şi 15. Să se deseneze fractalul arbore de nivel n, într-o imagine de dimensiuni 500x500. Segmentul inițial al fractalului se află între coordonatele (250,490)-(250,240) (deci are lungimea de 250). În cadrul fiecărui nivel, capetele libere se ramifică în două segmente de lungime egală cu jumătate din lungimea segmentului de la pasul precedent, orientate cu $\frac{\pi}{4}$ respectiv cu $-\frac{\pi}{4}$ față de acesta.

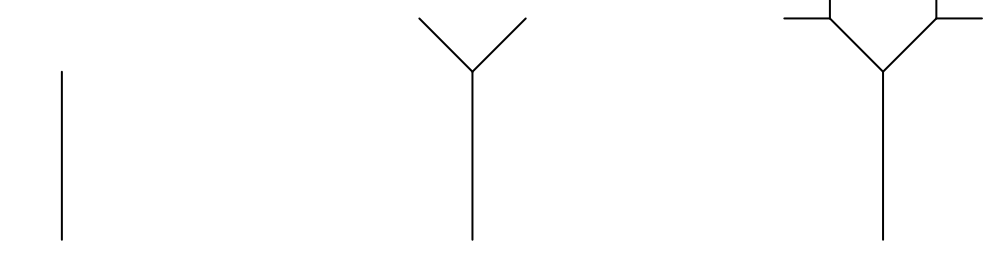
Suportul matematic necesar rezolvării:

Acest tip de fractal se încadrează unei clase mai largi, şi anume a fractalilor care se obțin prin repetarea recurentă a unui procedeu de desenare, la o scară mai mică şi având alte orientări. În acest sens, mersul procedeului de calcul necesar desenării sale este unul de tip divide et impera.

În cazul de față, vom lucra în coordonate polare, pe care le vom şi trimite, de altfel, funcției recursive care realizează desenarea propriuzisă.

Schema recursivă este următoarea:

- parametri funcției vor fi: (x_0, y_0), l, u, n. Primele două reprezită punctul de plecare, l = lungimea segmentului de bază pe nivelul curent, u = unghiul pe care-l face segmentul de la pasul curent cu orizontala, n = pasul curent.
- evoluția acestui fractal este următoarea:



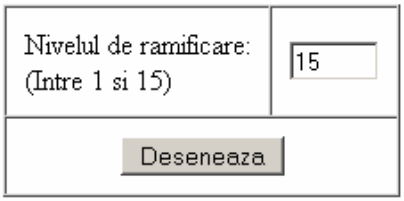
nivelul 1 nivelul 2 nivelul 3

acest lucru se transpune prin următoarea schemă recursivă:

$$fractal(x_0, y_0, l, u, n) = \begin{cases} \text{nu se mai face nimic daca } n = 0 \\ \text{daca } n \neq 0 : \begin{cases} \text{- se calculeaza } (x_1, y_1) = \text{celalalt capat al} \\ \text{segmentului dat de } (x_0, y_0), \text{ lungime l, unghi u} \\ \text{- din capatul determinat, } (x_1, y_1) \text{ se apeleaza recursiv} \\ \text{procedeul de desenare, prin :} \\ fractal\left(x_1, y_1, \frac{l}{2}, u - \frac{\pi}{4}, n\right) \text{ si } fractal\left(x_1, y_1, \frac{l}{2}, u + \frac{\pi}{4}, n\right) \end{cases} \end{cases}$$

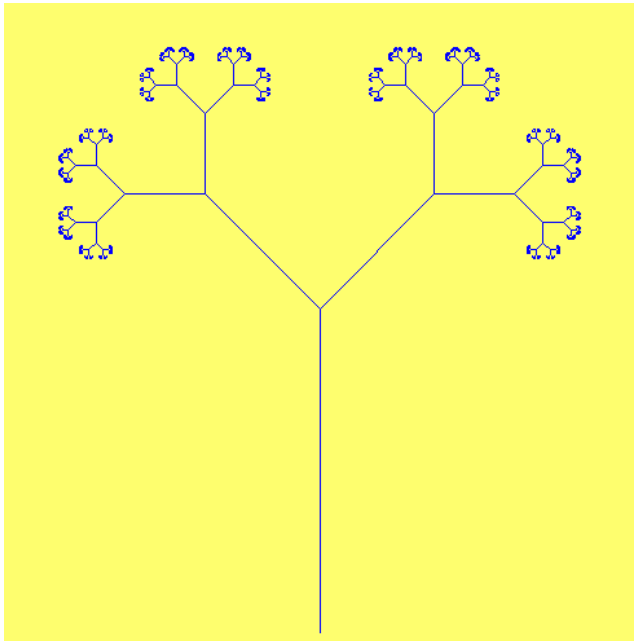
apl116.html

```
<html><body>
<form action="apl117.php" method="post">
<table border="1" cellspacing="0" cellpadding="10">
<tr>
  <td>Nivelul de ramificare:<br>(Intre 1 si 15)
  <td><input type="text" name="n" size="4">
<tr>
  <td colspan="2" align="center">
  <input type="submit" value="Deseneaza">
</table>
</form>
</body></html>
```

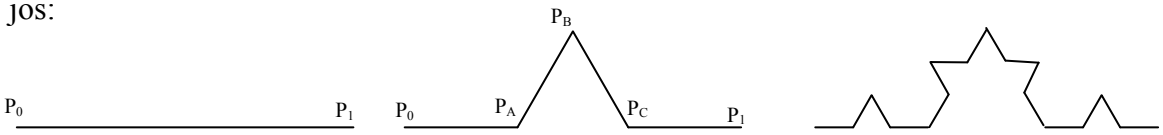


apl117.php

```
<?php
  header("Content-type: image/png");
  $im=imagecreatetruecolor(500,500);
  $y=imagecolorallocate($im,255,255,110);
  imagefilledrectangle($im,0,0,499,499,$y);
  $n=$_POST['n'];
  $blue=imagecolorallocate($im,0,0,255);
  $l=300;

  function arbore($x0,$y0,$l,$u,$n)
  {
    if($n==0) return;
//declaram global  atit variabila imagine cit
//si cea care defineste culoarea albastru
//pentru a le putea accesa din functie:
    global $im,$blue;
//calculam coordonatele celuilalt capat
//al segmentului:
    $x1=$x0+$l*cos($u);
    $y1=$y0-$l*sin($u);
//il desenam:
    imageline($im,$x0,$y0,$x1,$y1,$blue);
//apelam recursiv functia, pentru a ramifica
//si pe nivelele urmatoare. Observati cum
//sunt trimise mai departe coordonatele
//celuilalt capat, din care ramificam, si
//unghiuri deviate cu pi()/4 in stinga si
//dreapta fata de unghiul segmentului curent
    arbore($x1,$y1,$l/2,$u-pi()/4,$n-1);
    arbore($x1,$y1,$l/2,$u+pi()/4,$n-1);
  }

  arbore(250,490,250,pi()/2,$n);

  imagepng($im);
  imagedestroy($im);
?>
```



49) **Problemă rezolvată** (**apl118.html** + **apl119.php**): Se citeşte, prin intermediul unui câmp text al unui formular, un număr natural cuprins între 1 şi 10. Să se deseneze fractalul lui Koch de nivel n, într-o imagine de dimensiuni 640x210. Segmentul inițial al fractalului se află între coordonatele (10,200)-(630,200) (deci are lungimea de 620). Procedeul recursiv este ilustrat mai jos:



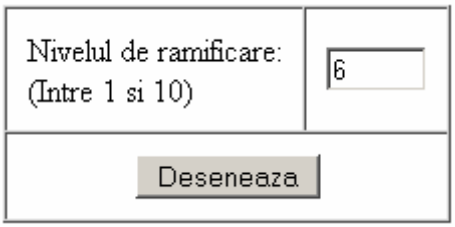
nivelul 1 nivelul 2 nivelul 3

Aşadar :

- pe nivelul 1 se desenează pur şi simplu un segment între punctele P_0 şi P_1
- pe orice alt nivel n, se procedează astfel:
  - se determină punctele intermediare P_A, P_B, P_C , ca în figura de mai sus (mijloc) prin împărțirea segmentului în 3 părți egale şi construirea unui triunghi echilateral care are ca bază segmentul din mijloc
  - se desenează 4 fractali de dimensiuni l/3 şi de nivel cu 1 mai puțin, după cum urmează:
    - unul ce pleacă din P_0, sub acelaşi unghi ca şi cel de pe nivelul curent;
    - altul ce pleacă din P_A, sub un unghi cu $\pi/3$ mai mare decât cel al nivelului curent;
    - altul ce pleacă din P_B, sub un unghi cu $\pi/3$ mai mic decât cel al nivelului curent;
    - în fine, altul ce pleacă din P_C, sub acelaşi unghi ca şi cel de pe nivelul curent.

apl118.html

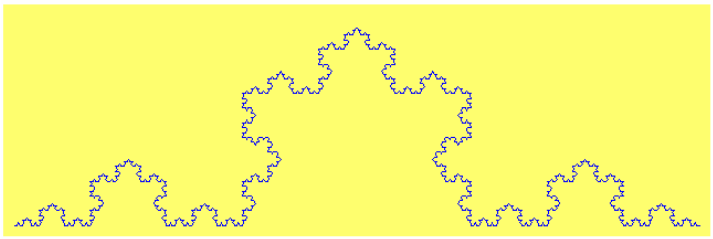
```
<html><body>
<form action="apl119.php" method="post">
<table border="1" cellspacing="0" cellpadding="10">
<tr>
  <td>Nivelul de ramificare:<br>(Intre 1 si 10)
  <td><input type="text" name="n" size="4">
<tr>
  <td colspan="2" align="center">
  <input type="submit" value="Deseneaza">
</table>
</form>
</body></html>
```



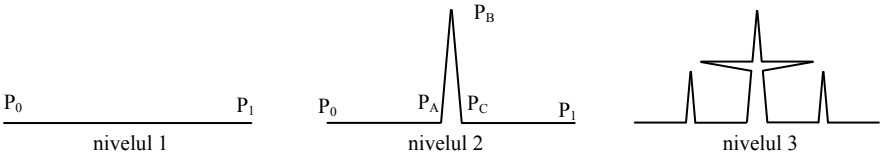
apl119.php

```
<?php
  header("Content-type: image/png");
  $im=imagecreatetruecolor(640,210);
  $y=imagecolorallocate($im,255,255,110);
  imagefilledrectangle($im,0,0,640,210,$y);
  $n=$_POST['n'];
  $blue=imagecolorallocate($im,0,0,255);
```

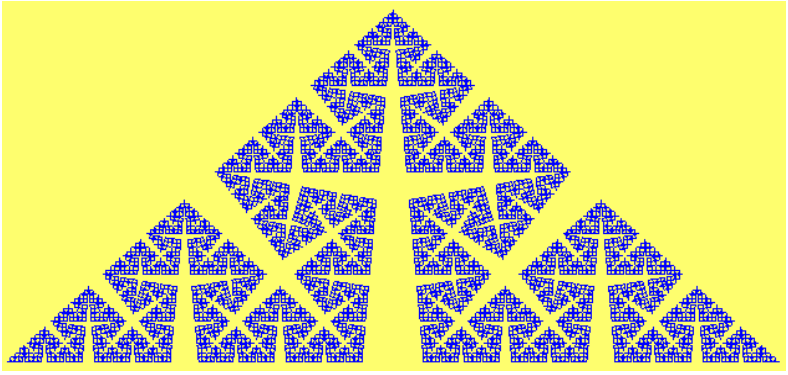
```
  function koch($x0,$y0,$l,$u,$n)
  {
//daca nu suntem pe nivelul $n==1 (cel de
//baza, cind fractalul se reduce la un segment
    if($n!=1)
     {
//calculam coordonatele punctelor Pa, Pb, Pc
       $xa=$x0+$l/3*cos($u);
       $ya=$y0-$l/3*sin($u);
       $xb=$xa+$l/3*cos($u+pi()/3);
       $yb=$ya-$l/3*sin($u+pi()/3);
       $xc=$xb+$l/3*cos($u-pi()/3);
       $yc=$yb-$l/3*sin($u-pi()/3);
//Si apelam recursiv desenarea fractalului
//pe nivelele urmatoare:
       koch($x0,$y0,$l/3,$u,$n-1);
       koch($xa,$ya,$l/3,$u+pi()/3,$n-1);
       koch($xb,$yb,$l/3,$u-pi()/3,$n-1);
       koch($xc,$yc,$l/3,$u,$n-1);
     }
    else
      {
//iar daca suntem pe nivelul 1, desenam pur si
//simplu segmentul:
        global $im,$blue;
        $x1=$x0+$l*cos($u);
        $y1=$y0-$l*sin($u);
        imageline($im,$x0,$y0,$x1,$y1,$blue);
      }
  }

  koch(10,200,620,0,$n);
  imagepng($im);
  imagedestroy($im);
?>
```



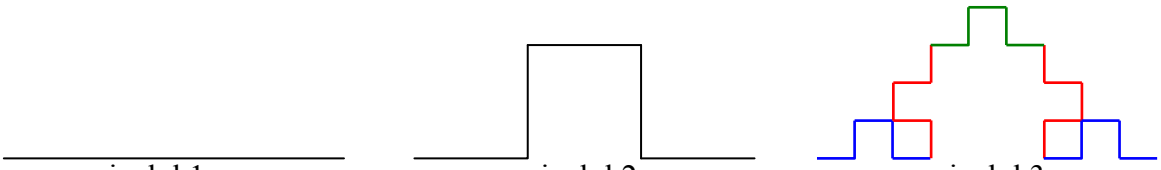
49) Se citeşte, prin intermediul unui câmp text al unui formular, un număr natural cuprins între 1 şi 10. Să se deseneze, într-o imagine de dimensiuni 640x480, o variantă a fractalului lui Koch (cunoscută sub numele de fractalul lui Cesaro) obținut după un procedeu similar: fiecare segment se înlocuieşte tot cu 4 segmente congruente, cu deosebirea că cele oblice NU mai formează un triunghi echilateral, ci un triunghi isoscel ale cărui unghiuri de la bază au 85 de grade (în figura de mai jos, este vorba de triunghiul $P_AP_BP_C$). Segmentul inițial al fractalului se află între coordonatele (10,470)-(630,470) (deci are lungimea de 620). Procedeul recursiv este ilustrat mai jos:



Iată şi o imagine a acestui fractal, pentru n = 8:



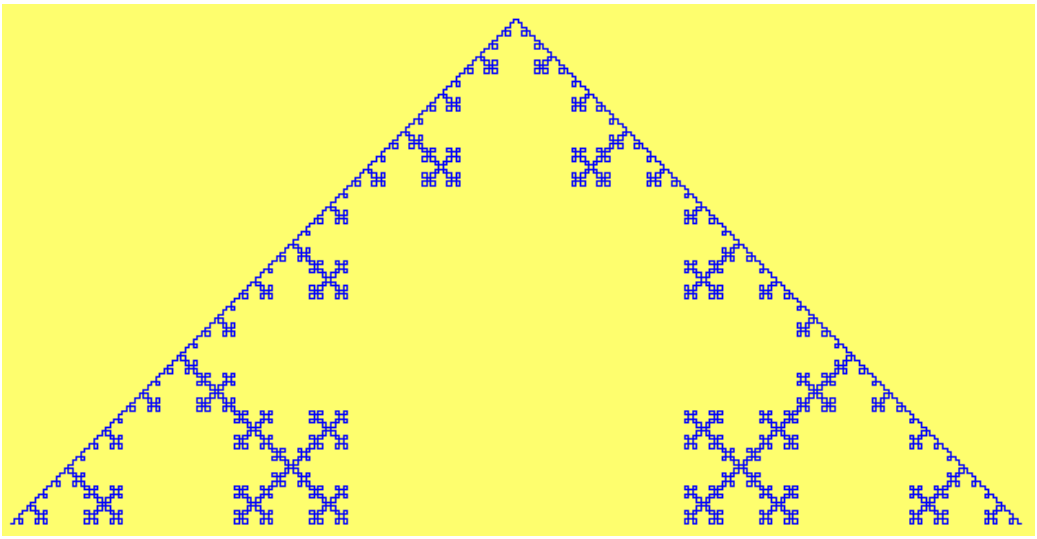
50) Se citeşte, prin intermediul unui câmp text al unui formular, un număr natural cuprins între 1 şi 8. Să se deseneze, într-o imagine de dimensiuni 640x480, o variantă a fractalului lui Koch obținut după procedeul următor: fiecare segment se înlocuieşte cu 5 segmente congruente, între care se formează unghiuri drepte, ca în schema de mai jos. Segmentul inițial al fractalului se află între coordonatele (10,470)-(630,470) (deci are lungimea de 620). Procedeul recursiv este ilustrat mai jos:



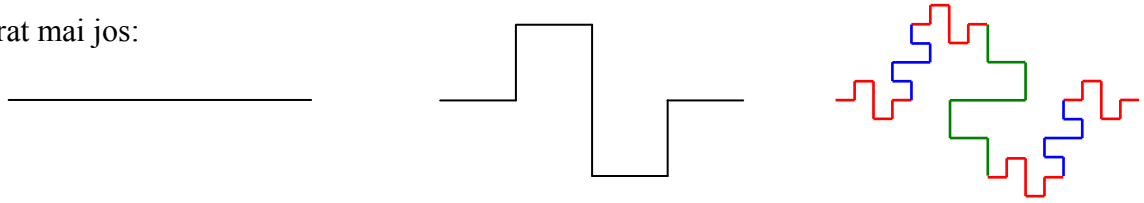
nivelul 1 nivelul 2 nivelul 3

(am colorat diferit fractalii de pe nivelul anterior, pentru a-i diferenția)

Iată şi o imagine a acestui fractal, pentru n = 6:



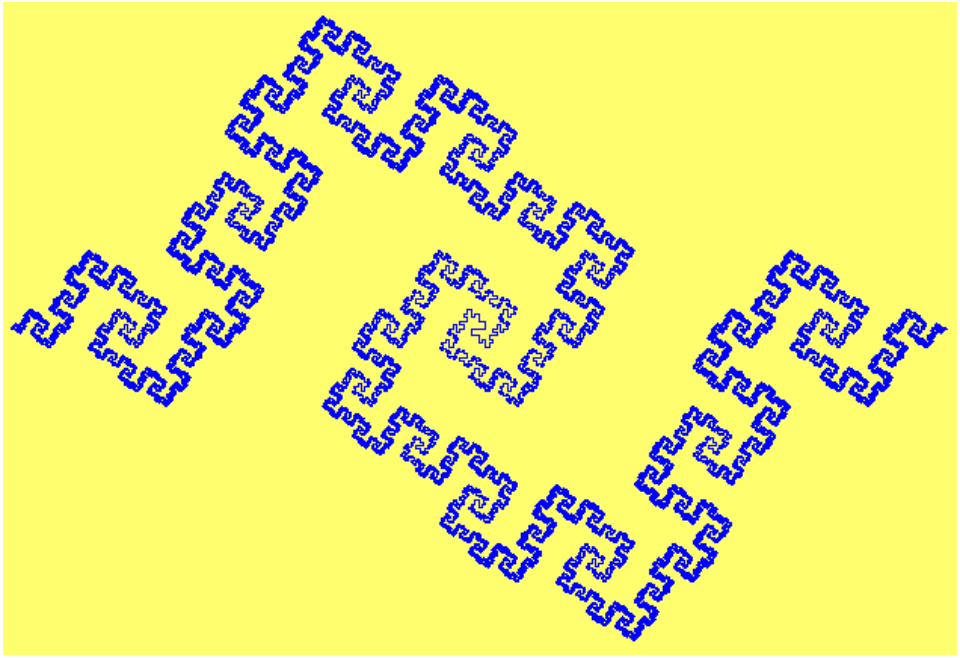
51) Se citeşte, prin intermediul unui câmp text al unui formular, un număr natural cuprins între 1 şi 8. Să se deseneze, într-o imagine de dimensiuni 640x480, o variantă a fractalului lui Koch obținut după procedeul următor: fiecare segment se înlocuieşte cu 7 segmente, dintre care 6 au lungimea egală cu 1/4 din lungimea celui inițial, iar unul are lungimea egală cu 1/2 din cel inițial. Între segmente se formează unghiuri drepte, ca în schema de mai jos. Segmentul inițial al fractalului se află între coordonatele (10,240)-(630,240) (deci are lungimea de 620). Procedeul recursiv este ilustrat mai jos:



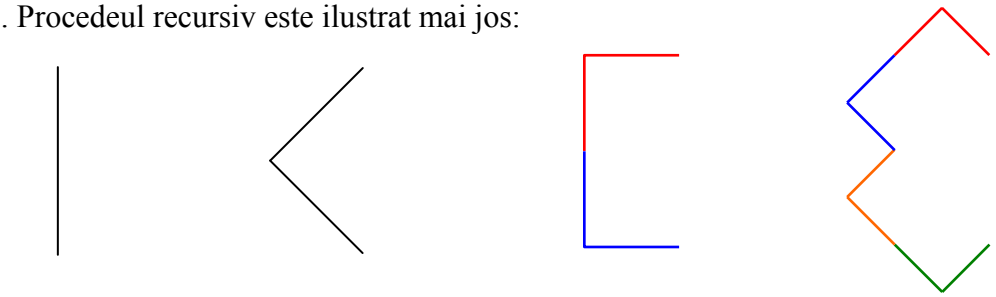
nivelul 1 nivelul 2 nivelul 3

(am colorat diferit fractalii de pe nivelul anterior, pentru a-i diferenția)

Iată şi o imagine a acestui fractal, pentru n = 7:

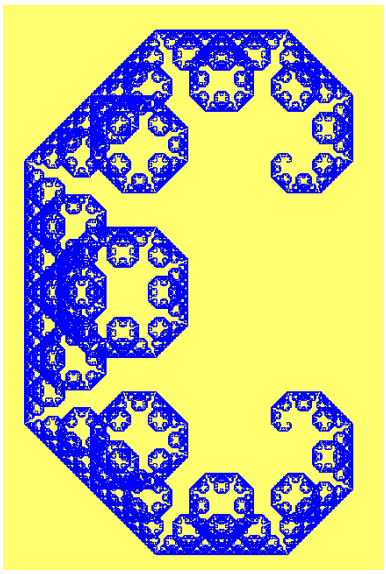


52) Se citeşte, prin intermediul unui câmp text al unui formular, un număr natural cuprins între 1 şi 19. Să se deseneze, într-o imagine de dimensiuni 450x650, fractalul "C" al lui Levy, obţinut după procedeul următor: fiecare segment se înlocuieşte cu alte 2 segmente, care sunt catetele triunghiului dreptunghic isoscel a cărui ipotenuză ar fi fost segmentul eliminat, ca în schema de mai jos. Segmentul iniţial al fractalului se află între coordonatele (330,480)-(330,180) (deci are lungimea de 300). Procedeul recursiv este ilustrat mai jos:

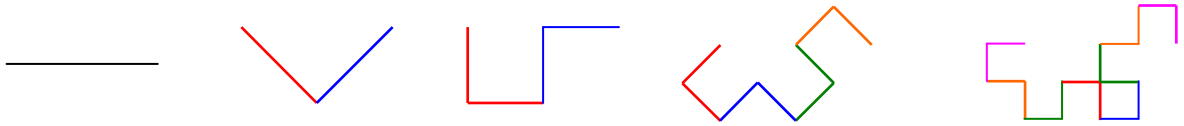


Nivelul 1 Nivelul 2 Nivelul 3 Nivelul 4

Iată şi o imagine a acestui fractal, pentru n = 19:



53) **Problemă rezolvată** (**apl120.html** + **apl121.php**): Se citeşte, prin intermediul unui câmp text al unui formular, un număr natural cuprins între 1 şi 19. Să se deseneze, într-o imagine de dimensiuni 640x480, fractalul dragonului. Procedeul său de obținere este foarte similar cel al fractalului precedent (curba "C" a lui Levy) doar că orientările vârfurilor triunghiurilor dreptunghice alternează între segmentele alăturate. Segmentul inițial al fractalului se află între coordonatele (150,170)-(550,170) (deci are lungimea de 400). Procedeul recursiv este ilustrat mai jos:



Nivelul 1 Nivelul 2 Nivelul 3 Nivelul 4 Nivelul 5

Pentru implementarea optimă a rezolvării am mai introdus la subprogramul recursiv care face desenarea fractalului, încă un parametru, "$sens", în funcție de care la nivelul următor fractalul care are ordinul cu 1 mai mic se va desena "sub" respectiv "deasupra".

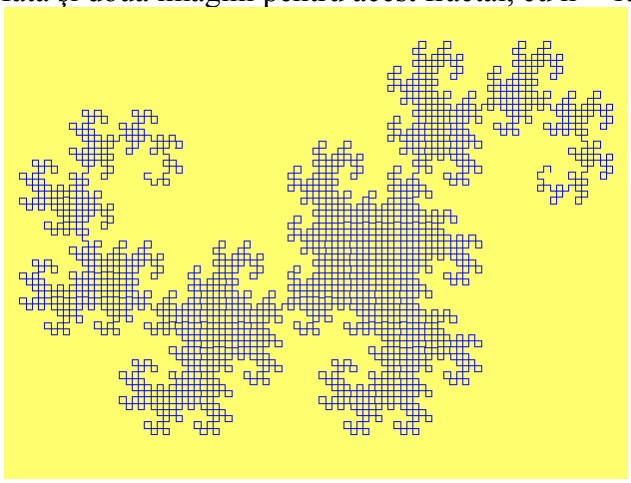
apl120.html

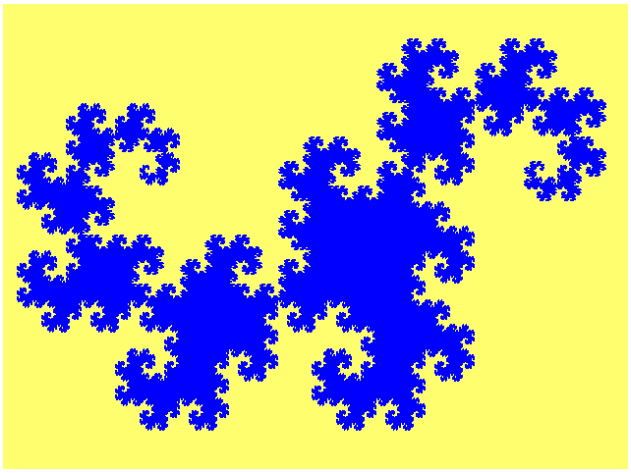
```
<html><body>
<form action="apl121.php" method="post">
<table border="1" cellspacing="0" cellpadding="10">
<tr>
  <td>Nivelul de ramificare:<br>(Intre 1 si 19)
  <td><input type="text" name="n" size="4">
<tr>
  <td colspan="2" align="center">
  <input type="submit" value="Deseneaza">
</table>
</form>
</body></html>
```

apl121.php

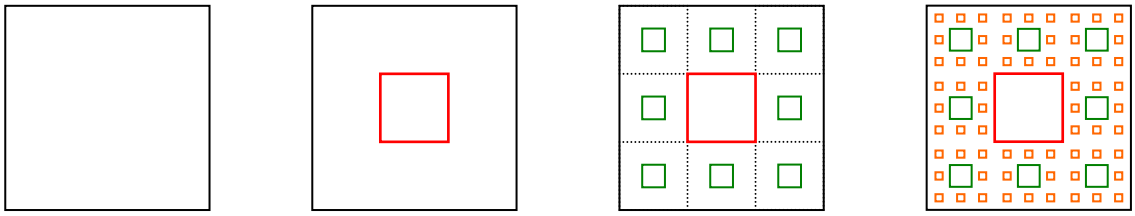
```
<?php
  header("Content-type: image/png");
  $im=imagecreatetruecolor(640,480);
  $y=imagecolorallocate($im,255,255,110);
  imagefilledrectangle($im,0,0,639,479,$y);
  $n=$_POST['n'];
  $blue=imagecolorallocate($im,0,0,255);
//spre deosebire de programele precedente, am mai introdus un parametru "sens".
//In functie de el desenam "deasupra" respectiv "dedesubtul" liniei curente
  function dragon($x0,$y0,$l,$u,$sens,$n)
  { global $im,$blue;
    if($n!=1)
     { $l1=$l/sqrt(2);
       $xa=$x0+$l1*cos($u-$sens*pi()/4);
       $ya=$y0-$l1*sin($u-$sens*pi()/4);
//La apelurile recursive, primul apel va avea sensul identic cu cel al nivelului in care
//suntem, in schimb al doilea apel va avea sens contrar:
       dragon($x0,$y0,$l1,$u-$sens*pi()/4,1,$n-1);
       dragon($xa,$ya,$l1,$u+$sens*pi()/4,-1,$n-1);
     }
    else
      {//iar daca suntem pe nivelul 1, desenam pur si simplu segmentul:
        $x1=$x0+$l*cos($u);
        $y1=$y0-$l*sin($u);
        imageline($im,$x0,$y0,$x1,$y1,$blue);}
   }
  dragon(150,170,400,0,1,$n);
  imagepng($im);
  imagedestroy($im);
?>
```

Iată şi două imagini pentru acest fractal, cu n = 13, respectiv n = 19:





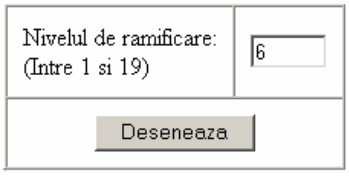
53) **Problemă rezolvată** (**`apl122.html`** + **`apl123.php`**): Se citeşte, prin intermediul unui câmp text al unui formular, un număr natural cuprins între 1 şi 6. Să se deseneze, într-o imagine de dimensiuni 700x700 fractalul covor al lui Sierpinski. Acest fractal se obține plecând de la un pătrat, împărțindu-l în nouă pătrate egale. Se trasează doar conturul celui din mijloc, iar celorlalte 8 pătrate li se aplică în mod recursiv acelaşi procedeu. Pătratul inițial va avea două dintre colțurile diagonal opuse la coordonatele (10,10)-(690,690) (deci latura de 680). Procedeul recursiv este ilustrat mai jos (segmentele punctate nu se vor desena):



Nivelul 1 Nivelul 2 Nivelul 3 Nivelul 4

`apl122.html`

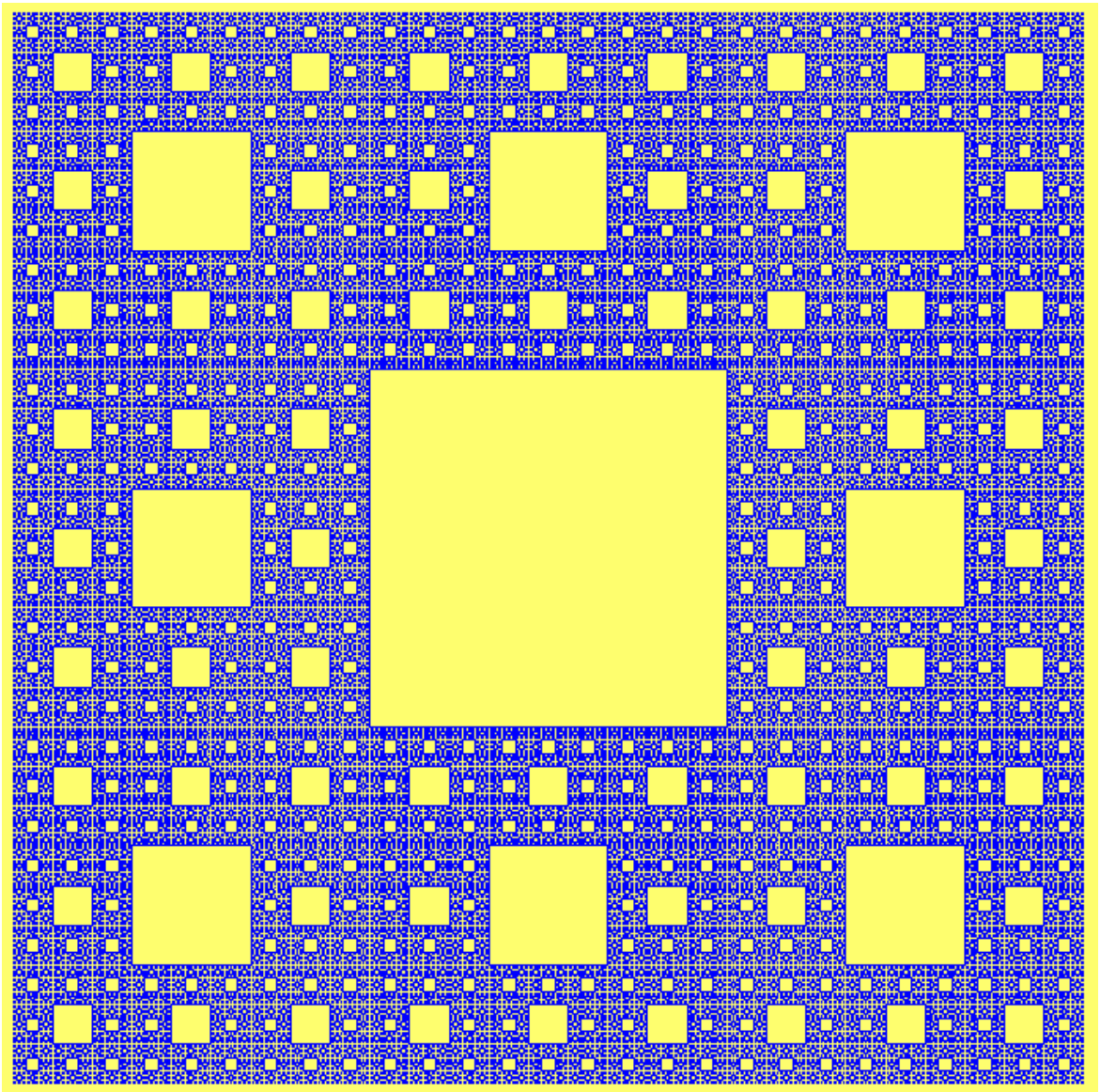
```
<html><body>
<form action="apl123.php" method="post">
<table border="1" cellspacing="0" cellpadding="10">
<tr>
  <td>Nivelul de ramificare:<br>(Intre 1 si 6)
  <td><input type="text" name="n" size="4">
<tr>
  <td colspan="2" align="center">
  <input type="submit" value="Deseneaza">
</table>
</form>
</body></html>
```



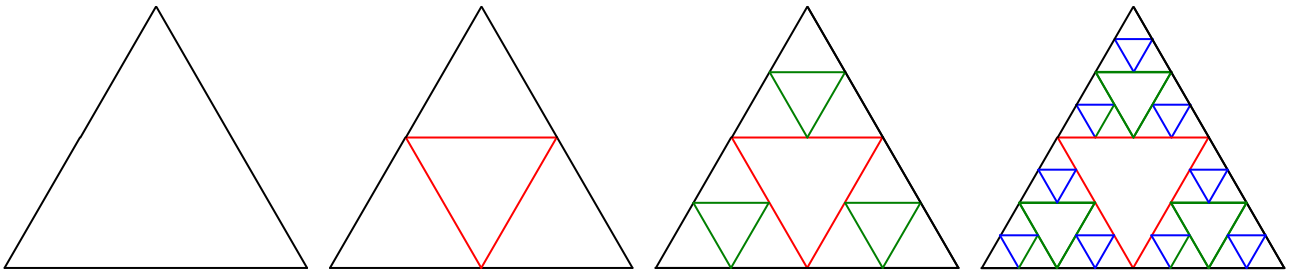
`apl123.php`

```
<?php
  header("Content-type: image/png");
  $im=imagecreatetruecolor(700,700);
  $y=imagecolorallocate($im,255,255,110);
  imagefilledrectangle($im,0,0,699,699,$y);
  $n=$_POST['n'];
  $blue=imagecolorallocate($im,0,0,255);
```

```
//functia recursiva care realizeaza desenarea va primi ca parametri coordonatele coltului
//stinga sus al patratului, lungimea laturii sale precum si nivelul:
  function covor_sierpinski($x0,$y0,$l,$n)
  {
    global $im,$blue;
    if($n!=0)
     {
      $l1=$l/3;
      //desenam patratul din mijloc:
      imagerectangle($im,$x0+$l1,$y0+$l1,$x0+2*$l1,$y0+2*$l1,$blue);
      //si apelam recursiv procedeul pentru celelalte 8 patrate:
      covor_sierpinski($x0,$y0,$l1,$n-1);
      covor_sierpinski($x0+$l1,$y0,$l1,$n-1);
      covor_sierpinski($x0+2*$l1,$y0,$l1,$n-1);
      covor_sierpinski($x0,$y0+$l1,$l1,$n-1);
      covor_sierpinski($x0+2*$l1,$y0+$l1,$l1,$n-1);
      covor_sierpinski($x0,$y0+2*$l1,$l1,$n-1);
      covor_sierpinski($x0+$l1,$y0+2*$l1,$l1,$n-1);
      covor_sierpinski($x0+2*$l1,$y0+2*$l1,$l1,$n-1);
     }
}
  imagerectangle($im,10,10,690,690,$blue);
  covor_sierpinski(10,10,680,$n);
  imagepng($im);
  imagedestroy($im);
?>
```

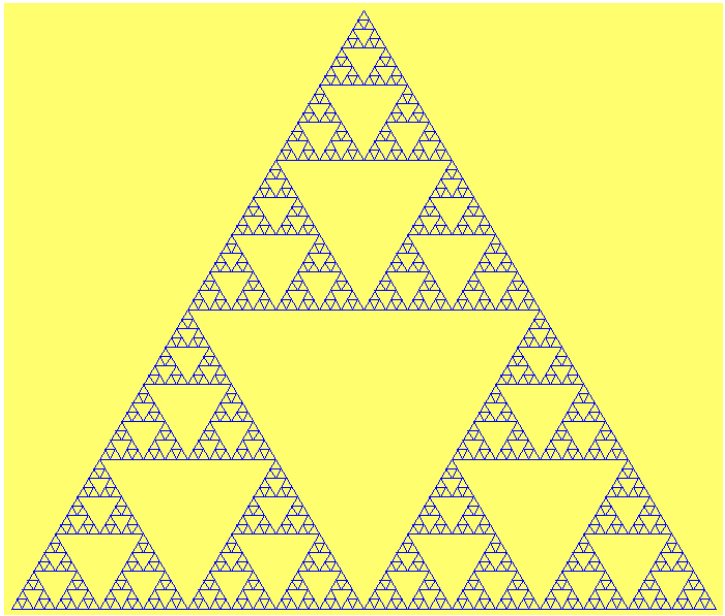


54) Se citeşte, prin intermediul unui câmp text al unui formular, un număr natural cuprins între 1 şi 11. Să se deseneze, într-o imagine de dimensiuni 700x600 fractalul triunghi al lui Sierpinski. Acest fractal se obține plecând de la un triunghi (de preferință echilateral, însă procedeul poate fi aplicat oricărui fel de triunghi) în care trasăm toate cele 3 linii mijlocii. Triunghiurilor care se formează, cu excepția celui care are ca laturi toate cele 3 linii mijlocii, li se aplică în mod recursiv exact acelaşi procedeu. Triunghiul inițial va avea vârfurile la coordonatele: (350,10), (10,589), (690,589). Procedeul recursiv este ilustrat mai jos:



Nivelul 1 Nivelul 2 Nivelul 3 Nivelul 4

Iată şi o imagine a acestui fractal, pentru n = 7:



55) **Problemă rezolvată** (`apl124.php`): Generarea fractalului de tip ferigă, al lui Barnsley. Deşi acest fractal se poate desena şi printr-un procedeu de tipul celor descrise anterior (prin trasări de segmente, conform unui procedeu recurent) vom aborda o altă modalitate de descriere a sa, şi anume IFS (***I***terated ***F***unction ***S***ystem).

Din punct de vedere matematic, un fractal IFS este definit de o mulțime de transformări geometrice elementare (care în matematică se mai numesc lineare sau afine) care în limbajul de zi cu zi se traduc prin rotații, aplatizări, deformări (de tipul dreptunghi ⇒ paralelogram), scalări. Ceea ce este specific transformărilor utilizate la un fractal IFS este faptul că acestea trebuie să fie de tip contracție, adică distanța dintre două puncte cărora li se aplică transformarea să se diminueze (sau cel puțin, să nu crească).

O transformare de tipul celor de mai sus se transcrie prin:

$$\begin{cases} x' = fx(x,y) = a \cdot x + b \cdot y + e \\ y' = fy(x,y) = c \cdot x + d \cdot y + f \end{cases} \quad (*)$$

unde (x,y) reprezintă coordonatele punctului anterior, iar (x',y') reprezintă coordonatele punctului curent. Procedeul fiind iterativ, coordonatele punctului curent vor deveni bază de plecare (deci coordonate anterioare) pentru calculul noilor coordonate la pasul următor.

Fractalul ferigă al lui Barnsley se bazează pe următoarele 4 transformări (în dreptul fiecăreia vom scrie şirul coeficienților (a,b,c,d,e,f) care o defineşte (atenție la ordinea în care am scris coeficienții, care este cea din ecuațiile (*) de mai sus):

0) $\begin{cases} x' = fx_0(x,y) = 0 \\ y' = fy_0(x,y) = 0.16y \end{cases}$ cu şirul coeficienților $(0.00, 0.00, 0.00, 0.16, 0.00, 0.00)$

1) $\begin{cases} x' = fx_1(x,y) = 0.20x - 0.26y \\ y' = fy_1(x,y) = 0.23x + 0.22y + 1.60 \end{cases}$ cu şirul coeficienților $(0.20, -0.26, 0.23, 0.22, 0.00, 1.60)$

2) $\begin{cases} x' = fx_2(x,y) = -0.15x + 0.28y \\ y' = fy_2(x,y) = 0.26x + 0.24y + 0.44 \end{cases}$ cu şirul coeficienților $(-0.15, 0.28, 0.26, 0.24, 0.00, 0.44)$

3) $\begin{cases} x' = fx_3(x,y) = 0.85x + 0.04y \\ y' = fy_3(x,y) = -0.04x + 0.85y + 1.60 \end{cases}$ cu şirul coeficienților $(0.85, 0.04, -0.04, 0.85, 0.00, 1.60)$

Procedeul de desenare pleacă de la punctul de coordonate (0,0) pe care îl iterează într-una dintre cele 4 ecuații de mai sus, aleasă aleator, însă cu o anumită frecvență, şi anume:

- prima dintre ecuații se va folosi o dată din 100, deci probabilitatea sa va fi 0,01;
- a doua dintre ecuații se va folosi de 7 ori din 100, deci cu probabilitatea de 0,07;
- a treia dintre ecuații se va folosi tot de 7 ori din 100, deci tot cu probabilitatea de 0,07;
- în fine, ultima dintre ecuații se foloseşte în restul cazurilor, deci de 85 de ori din 100, cu probabilitatea de 0,85.

Punctele astfel obținute reprezintă imaginea fractalului.

Prin urmare, datele necesare reprezentării pot fi rezumate în următorul tabel:

	a = coeficientul lui x din prima ecuație a transf.	b = coeficientul lui y din prima ecuație a transf.	c = coeficientul lui x din a doua ecuație a transf.	d = coeficientul lui y din a doua ecuație a transf.	e = termenul liber din prima ecuație a transf.	f = termenul liber din a doua ecuație a transf.	p = probabilitatea cu care trebuie aleasă această transformare
transformarea 0	0.00	0.00	0.00	0.16	0.00	0.00	0.01
transformarea 1	0.20	-0.26	0.23	0.22	0.00	1.60	0.07
transformarea 2	-0.15	0.28	0.26	0.24	0.00	0.44	0.07
transformarea 3	0.85	0.04	-0.04	0.85	0.00	1.60	0.85

Pentru o reprezentare grafică sugestivă, e necesar să efectuăm cel puțin 300.000 de iterații.

În urma calculelor, coordonatele punctelor ce se vor determina vor fi delimitate de următoarele margini: x ∈ [-2.18, 2.66] şi y ∈ [0.08, 10], deci punctele rămân în interiorul unui dreptunghi cu lățimea de 4.84 respectiv cu înălțimea 9.92.

Pentru a le putea reprezenta pe ecran, vom alege un factor de scalare egal cu 70, astfel, imaginea pe care o generăm va avea dimensiunile de aproximativ 340 x 700.

Formulele de reprezentare la scară vor fi (fie xmin=-2.18, xmax=2.66, ymin=0.08, ymax=10, scale=70):

$x_{imagine}$ = (x-xmin)*scale

$y_{imagine}$ = 700-(y-ymin)*scale (aceasta din urmă se datorează orientării negative a axei OY).

Iată codul sursă al programului:

apl124.php

```
<?php
  header("Content-type: image/png");
  $im=imagecreatetruecolor(340,700);
  $y=imagecolorallocate($im,255,255,210);
  imagefilledrectangle($im,0,0,339,699,$y);
  $gr=imagecolorallocate($im,0,128,0);
//prin variabila de mai jos ne fixam numarul de puncte pe care le vom reprezenta:
  $npoints=300000;
  $f[0]=array( 0.00, 0.00, 0.00,0.16,0.00,0.00,0.01);
  $f[1]=array( 0.20,-0.26, 0.23,0.22,0.00,1.60,0.07);
  $f[2]=array(-0.15, 0.28, 0.26,0.24,0.00,0.44,0.07);
  $f[3]=array( 0.85, 0.04,-0.04,0.85,0.00,1.60,0.85);
//calculam intr-un sir "prf" suma probabilitatilor, pentru a ne fi mai usor sa alegem in
//mod aleator transformarea corespunzatoare:
  $s=0; for($i=0;$i<=3;$i++)
            { $s+=$f[$i][6]; $prf[$i]=$s;}
//definim in $xold, $yold coordonatele punctului de plecare:
  $xold=$yold=0.0;
  $xmin=-2.18;$xmax=2.66;$ymin=0.08;$ymax=10;$scale=70;
//si ne apucam de iterat:
  for($i=1;$i<=$npoints;$i++)
  {//nu vom folosi functia clasica rand(), deoarece algoritmul implementat de aceasta
//genereaza numere pseudo-aleatoare ce devin periodice foarte repede, fata de nevoile
//reprezentarii fractalului de fata. Vom folosi
// functia mt_rand() - proprie limbajului
//PHP, ce contine o reimplementare imbunatatita a
// lui rand(). Folosirea sa este identica
//cu a functiei rand().
//Asadar, mai jos calculam un numar aleator
//cuprins intre 0 si 99, pe care-l impartim apoi
//la 100, ca sa obtinem un numar cuprins intre
//0.00 si 0.99, pe baza caruia alegem
//transformarea curenta
    $aleat=mt_rand(0,99)/100;
//determinam $k = indicele transformarii curente:
    $k=0; while($prf[$k]<$aleat) $k++;
    //calculam noile coordonate:
    $x=$f[$k][0]*$xold+$f[$k][1]*$yold+$f[$k][4];
    $y=$f[$k][2]*$xold+$f[$k][3]*$yold+$f[$k][5];
    $xold=$x;$yold=$y;
//si le reprezentam, dupa ce le aducem la scara
//imaginii noastre:
    $xim=(double)($x-$xmin)*$scale;
    $yim=700.0-(double)($y-$ymin)*$scale;
    imagesetpixel($im,$xim,$yim,$gr);
  }
  imagepng($im);
  imagedestroy($im);
?>
```

56) În cadrul aplicației anterioare am văzut faptul că fractalul ferigă este perfect definit de tabelul în care am notat coeficienții celor 4 transformări afine precum şi probabilitatea cu care trebuie aleasă fiecare transformare. Totodată pentru fiecare reprezentare mai trebuie să precizăm care sunt limitele coordonatelor care mărginesc fractalul (xmin, xmax, ymin, ymax) precum şi scara la care facem reprezentarea. Generați ferigile şi pentru următoarele două seturi de date, într-o imagine de dimensiuni 340x700:

a) setul I:

	a = coeficientul lui x din prima ecuație a transf.	b = coeficientul lui y din prima ecuație a transf.	c = coeficientul lui x din a doua ecuație a transf.	d = coeficientul lui y din a doua ecuație a transf.	e = termenul liber din prima ecuație a transf.	f = termenul liber din a doua ecuație a transf.	p = probabilitatea cu care trebuie aleasă această transformare
transformarea 0	0.00	0.00	0.00	0.25	0.00	-0.14	0.02
transformarea 1	0.09	-0.28	0.30	0.11	0.00	0.60	0.07
transformarea 2	-0.09	0.28	0.30	0.09	0.00	0.70	0.07
transformarea 3	0.85	0.02	-0.02	0.83	0.00	1.00	0.84

cu limitele care îl mărginesc: xmin = -1.56, xmax = 1.56, ymin = -0.18, ymax = 5.80 şi scara = 109

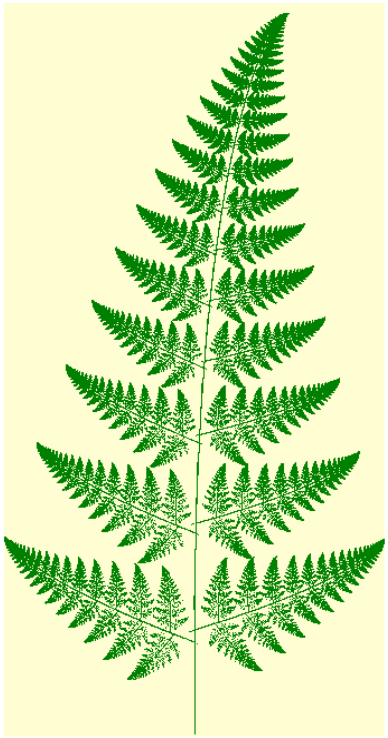
b) setul al II-lea:

	a = coeficientul lui x din prima ecuație a transf.	b = coeficientul lui y din prima ecuație a transf.	c = coeficientul lui x din a doua ecuație a transf.	d = coeficientul lui y din a doua ecuație a transf.	e = termenul liber din prima ecuație a transf.	f = termenul liber din a doua ecuație a transf.	p = probabilitatea cu care trebuie aleasă această transformare
transformarea 0	0.00	0.00	0.00	0.16	0.00	0.00	0.10
transformarea 1	0.20	-0.26	0.23	0.22	0.00	1.60	0.08
transformarea 2	-0.15	0.28	0.26	0.24	0.00	0.44	0.08
transformarea 3	0.75	0.04	-0.04	0.85	0.00	1.60	0.74

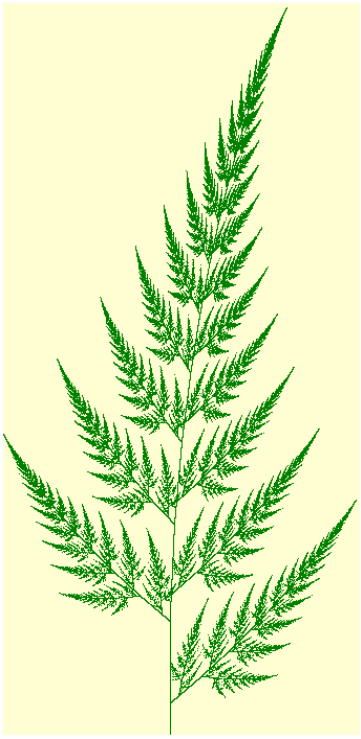
cu limitele care îl mărginesc: xmin = -2.34, xmax = 2.62, ymin = 0, ymax = 10.22 şi scara = 68.

Iată şi imaginile lor:

a) Setul I:



b) Setul II:



5. CONSIDERAȚII METODOLOGICE

5.1. Posibilitatea predării limbajului PHP la clasa a XII-a; premise care facilitează introducerea sa în cadrul noilor programe şcolare. Analiza însuşirii sale de către elevi. Concluzii stabilite.

În zilele noastre, prin cuvântul "informatică" (termen care în Franța a fost consacrat oficial abia în 1967, reprezentând din punct de vedere etimologic o contracție între "informație" şi "automatică") desemnăm una dintre cele mai recent apărute ştiințe. Este vorba, mai precis, de domeniile ştiințific, tehnologic şi industrial, raportate la tratarea automată a informației de către maşini precum calculatoare, console, terminale, roboți.

Paradoxal, dacă stăm să ne gândim la toate îngrădirile epocii comuniste, în 1971, deci la doar 5 ani de la intrarea oficială a cuvântului "informatică" în vocabularul europenilor, Consiliul de Miniştri al României de la acea vreme decide înființarea liceelor cu profil informatic. Tot atunci sunt puse, deci, şi bazele studiului informaticii în cadrul învățământului preuniversitar românesc.

Dacă la început informatica a fost considerată ca fiind născută din matematică, de la apariția sa şi până în prezent lucrurile au evoluat într-un ritm inimaginabil de alert, în zilele noastre sunt cunoscute mai multe sub-ramuri ale sale.

Ceea ce ne interesează în mod special, din punct de vedere al analizei de față, este informatica sub aspectul studiului algoritmilor, a implementărilor acestora în limbaje de programare şi a transpunerii în practică, prin programarea lor pe un calculator.

Din acest punct de vedere, studiul informaticii în liceu are în prezent următoarea structură:

- în clasa a IX-a se studiază algoritmii elementari şi implementarea lor într-un limbaj de programare;
- în clasa a X-a se insistă în special pe chestiuni mai strâns legate de limbajul de programare (şiruri de caractere, structuri ce pot îngloba mai multe date (de tip înregistrare), subprograme);
- în clasa a XI-a se introduc câteva dintre metodele principale de programare (Backtracking, Divide et Impera, Alocare Dinamică şi Structuri de Date, Teoria Grafurilor)

Până în anul şcolar 2006-2007 (inclusiv), în programa şcolară a clasei a XII-a era prevăzut studiul bazelor de date, limbajul folosit fiind în special FoxPro. Acest limbaj a cunoscut o popularitate largă între anii 1990-2000, însă, o dată cu evoluția tehnicii de calcul şi apariția altor instrumente, a început să se deprecieze. Deşi Microsoft a continuat (şi continuă) să îl mențină, elaborând şi versiuni Visual ale sale, în prezent nu mai este aşa de folosit. Un alt inconvenient al său

este acela că permite manipularea bazelor de date şi într-un mod mai puțin convențional, folosind propriul limbaj, deci fără respectarea standardelor SQL.

Pe de altă parte, aşa cum am arătat în introducerea capitolelor al II-lea şi al III-lea, începând cu 1995, rețeaua Internet a cunoscut o dezvoltare explozivă, de unde a apărut şi necesitatea dezvoltării unor unelte de programare corespuzătoare.

Toate acestea au condus la impunerea unei schimbări, devenită efectivă din anul şcolar 2007-2008, când structura programei şcolare destinată claselor a XII-a a fost modificată, îngăduind, din punctul de vedere al profesorului, o abordare mult mai permisivă, deoarece poate alege dintre modulele pe care le va preda.

Unul dintre modulele ce pot fi alese şi în sprijinul căruia vine lucrarea de față, este cel denumit "Programare WEB".

După experiența anului şcolar 2007-2008 de predare a acestui limbaj conform noilor programe şi după experiența anului şcolar 2006-2007, când am propus studiul unui curs opțional de PHP, pe care elevii şi l-au ales, voi face câteva observații şi voi încerca să trag câteva concluzii.

În primul rând, studiul limbajului HTML în vreunul dintre anii şcolari precedenți poate fi de mare ajutor. În acest caz, în cadrul materiei capitolului al II-lea profesorul se poate ocupa mai amănunțit de automatizarea paginilor web cu ajutorul script-urilor Java.

De asemenea, datorită similitudinii foarte mari dintre C++ şi PHP (şi de asemenea şi JavaScript), elevii care au studiat limbajul C++ au un avantaj foarte mare față de cei care au studiat Pascal. Deoarece elevii mei au studiat în prealabil C++, nu m-am confruntat cu această din urmă situație.

Un atu extraordinar al limbajului PHP constă în acela că interfața (atât cea de intrare cât şi cea de ieşire) poate fi foarte mult îmbunătățită față de cea a limbajului studiat în clasele IX-XI prin introducerea de elemente grafice şi de culoare.

Un alt element foarte atractiv al său constă în funcțiile de programare grafică. Din păcate, acest capitol foarte spectaculos al informaticii nu este inclus nicăieri în mod explicit, în nici una dintre programele şcolare, ci este trecut sub tăcere. Personal, consider că includerea sa în cadrul programei obligatorie ar fi binevenită, încă din primul an de liceu, deoarece ar reprezenta în primul rând un factor de atractivitate, iar în al doilea rând ar familiariza elevii cu lucrul efectiv în coordonate carteziene, cu reprezentări grafice şi cu proprietăți ale acestora.

Aşadar, PHP are toate şansele de a fi un limbaj uşor de învățat de către elevi. În mare parte, însuşirea sa se petrece într-o mare măsură.

E necesar ca elevii să lucreze cât mai mult din punct de vedere practic, deci să implementeze pe calculator cât mai multe exemple, să testeze cât mai multe funcţii şi situaţii.

De asemenea, este foarte indicată reluarea problemelor clasice şcolăreşti (de clasa a IX-a, a X-a şi a XI-a) cu mici precizări (acolo unde se poate, evident) care să ajute la înfrumuseţarea datelor de ieşire (gen: numerele să fie afişate în tabele, anumite elemente să fie colorate, etc.). O serie de probleme special concepute în acest sens se găsesc în capitolul precedent (4).

Avantajul cunoaşterii limbajului PHP la terminarea liceului nu poate fi decât benefică, mergând până într-acolo încât poate chiar să constituie o meserie.

5.2. Posibilităţi de predare cât mai atractive ale informaticii, fără a se ajunge la banalizare: propunere de curs opţional „Programare grafică într-un limbaj vizual"

Unul dintre celelalte module prevăzute de programa şcolară a clasei a XII-a constă în "Programare Vizuală". Din nou avem de-a face un subiect de actualitate, cu un grad de interes mare din partea elevilor, datorată elementelor vizuale ale interfeţei şi modului relativ simplu prin care acestea se pot programa respectiv corela.

După cum am evidenţiat în paragraful precedent, programarea grafică este un capitol trecut sub tăcere în cadrul programelor şcolare actuale. Deşi în cadrul disciplinei "informatică" din clasa a XII-a este posibil ca acest capitol să poată fi atins, profesorul nu dispune, totuşi, de prea multe ore pentru a face acest lucru. Din acest motiv, propunerea cursului opţional de faţă îşi propune tratarea pe larg a acestei problematici.

Adesea se face confuzie între "grafica pe calculator" şi "programarea grafică".

Pentru a realiza "grafică pe calculator" este nevoie, în general, de un pachet software specializat tratării imaginilor (de exemplu Phtoshop) utilizatorul neavând nevoie câtuşi de puţin să cunoască şi să stăpânească vreun limbaj de programare. Un curs opţional care să familiarizeze elevii cu un astfel de produs s-ar preta, poate, claselor care nu sunt de profil matematică-informatică. Un astfel de curs ar avea totuşi un anumit grad de banalitate, deoarece competenţele pe care elevii le-ar putea dobândi în urma sa s-ar limita la simpla dobândire de deprinderi de utilizare a unui produs software.

Conceptul de "programare grafică", în schimb, se referă la manipularea şi crearea imaginilor din cadrul unui limbaj de programare, utilizând structuri algoritmice şi tehnici de programare. Prin urmare, un curs care să abordeze problematica programării grafice se adresează exclusiv unui programator.

Cursul opțional pe care l-am propus elevilor claselor a XII-a cu specializarea matematică-informatică va avea ca şi suport de programare limbajul Microsoft Visual C++ 2008 Express Eddition (deci o platformă foarte proaspăt apărută!) versiune pe care Microsoft o pune în mod gratuit la dispoziția celor care doresc să o utilizeze în scopuri pur didactice.

Cursul îşi propune tratarea programării grafice în două etape: într-o primă parte, cursul este axat pe familiarizarea elevilor cu obiectele grafice şi cu operațiile de bază ce se pot efectua în cadrul unei imagini: accesarea la nivel de pixeli, atribute de culoare, coordonate carteziene, trasarea formelor geometrice de bază: segmente, cercuri, dreptunghiuri, elipse, afişarea unui text în cadrul unei imagini, afişarea unei imagini sau a unei porțiuni din aceasta în cadrul altei imagini, operații de decupare / redimensionare.

În cea de-a doua parte a sa, cursul este centrat pe conexiunea interdisciplinară dintre informatică şi matematică (mai precis geometrie analitică sau computațională). În această parte a cursului se urmăreşte exploatarea principalelor resurse matematice care pot conduce la reprezentări grafice. Astfel, o serie de formule şi rezultate teoretice pot fi verificate vizual, ajutând elevii să înțeleagă esența fenomenelor şi nu doar să se limiteze în a opera cu forme fără fundament.

În această parte se va urmări atingerea unor obiective precum ar fi:

- reprezentarea grafică a unei figuri geometrice (triunghi, dreptunghi, patrulater, poligon, regulat) şi a principalelor linii din aceasta (înălțimi, bisectoare, mediane, mediatoare), a cercurilor înscrise, circumscrise, folosirea coordonatelor polare;
- reprezentarea grafică a unei funcții la o anumită scară (cu factori de scalare egali sau diferiți pe OX respectiv pe OY);
- realizarea unor mici animații care să ilustreze locuri geometrice;
- reprezentări grafice de fractali.

În continuare voi prezenta o posibilă planificare a materiei pentru acest curs opțional:

Semestrul I: 18 săptămâni

Nr. crt.	Unitatea de învățare	Competențe specifice	Conținuturi	Nr. ore	Săpt.
1	Mediul de programare Visual C++ 2008 - Express Edition	- operarea sub mediul de programare Visual C++ - identificarea componentelor unei aplicații - stabilirea proprietăților resurselor, modificarea aspectului acestora - asocierea de cod evenimentelor	• Crearea, salvarea şi modificarea unui proiect • Uneltele şi ferestrele de bază ale mediului de programare • Fereastra de design a unui form, ferestrele de proprietăți, evenimente, cod sursă	3	1,2,3
2	Structuri avansate de programare - obiecte şi clase	- identificarea elementelor unui obiect - adresarea datelor respectiv funcțiilor membru ale unui obiect şi ale unei clase	• Obiectele şi clasele predefinite ale limbajului • Vizualizarea unui obiect în cadrul ferestrei "Class Explorer" • Modificarea membrilor unei clase • Obiecte statice şi obiecte de tip pointer - adresarea membrilor	3	4,5,6
3	Principalele tipuri de controale ale limbajului Visual C++	- identificarea principalelor tipuri de controale - modificarea aspectului unui control (proprietăți) - modificarea comportamentului unui control (evenimente)	• Controale de tip form, label, textbox, button, textarea, combobox, listbox, picturebox, progressbar	3	7,8,9
4	Casete de dialog standard ale limbajului Visual C++	- identificarea principalelor tipuri de casete de dialog - utilizarea casetelor de dialog în cadru unui program	• Casete de dialog de tipul MessageBox, Open, Save, FontSelection, ColorBox	3	10,11, 12
5	Obiecte de tip bitmap	- familiarizarea cu obiectele grafice de tip bitmap - principalele funcții de manipulare ale unui bitmap	• Declararea şi inițializarea unui obiect de tip bitmap • Încărcarea şi salvarea unei imagini dintr-un fişier în bitmap respectiv din bitmap într-un fişier • Redimensionarea, tăierea, rotirea unei imagini sau a unei porțiuni rectangulare din aceasta • Accesarea unui bitmap la nivel de pixel • Tratarea evenimentelor generate de acțiunea mouse-ului	3	13,14, 15
6	Obiecte de tip graphics	- familiarizarea cu obiectele de tip graphics - familiarizarea cu principalele funcții de manipulare ale unui obiect de tip graphics	• Inițializarea şi declararea unui obiect de tip graphics pornind de la un obiect deja existent • Desenarea principalelor forme: linii, dreptunghiuri, cercuri, elipse, puncte • Afişarea de text	3	16,17, 18

Semestrul al II-lea: 17 săptămâni

Nr. crt.	Unitatea de învățare	Competențe specifice	Conținuturi	Nr. ore	Săpt.
1	Trecerea de la reperul cartezian la reprezentarea grafică într-un bitmap	- familiarizarea cu particularitățile reprezentării grafice pe calculator - reprezentarea unui interval cartezian - alegerea centrului reperului - reprezentarea la scară	• Formule de trecere de la reperul cartezian la reprezentarea pe calculator: - alegerea centrului reperului - reprezentarea la scară	1	1
2	Reprezentarea grafică folosind coordonate polare	- familiarizarea cu reprezentarea în coordonatele polare	• Formule de reprezentare în coordonate polare • Reprezentarea poligoanelor regulate	2	2,3
3	Reprezentarea grafică a dreptelor şi a segmentelor	- familiarizarea cu tehnicile de reprezentare ale unei drepte şi ale unui segment - utilizarea formulelor matematice adecvate	• Reprezentarea acelei porțiuni de drepte vizibile în intervalul cartezian fixat • Intersecția dintre două drepte • Intersecția dintre două segmente • Împărțirea unui segment în mai multe părți egale	2	4,5
4	Reprezentarea grafică a principalilor fractali recursivi	- familiarizarea cu modul de construcție a principalilor fractali recursivi	• Reprezentarea fractalilor de tip linie Koch (stea, spirala), de tip Covor Sierpinski (triunghi, pătrat), de tip arbore	3	6,7,8
5	Reprezentarea grafică a unei funcții matematice	- familiarizarea cu tehnicile de reprezentare a unei funcții matematice	• Reprezentarea curbelor din puncte foarte apropiate sau din segmente foarte mici interconectate	2	9,10
6	Rezolvări de triunghiuri cu reprezentare grafică	- familiarizarea cu tehnicile de reprezentare ale unui triunghi - utilizarea formulelor matematice adecvate	• Linii importante în triunghi • Cercul înscris într-un triunghi • Cercul circumscris unui triunghi	2	11,12
7	Realizarea unei animații folosind cadre multiple	- utilizarea cadrelor multiple - proiectarea paşilor animației	• Crearea cadrelor din care este compusă animația • Folosirea obiectului de tip Timer pentru controlul schimbării cadrelor în vederea realizării animației	1	13
8	Reprezentări animate ale unor locuri geometrice	- identificarea paşilor reprezentării - utilizarea formulelor matematice adecvate	• Reprezentări de locuri geometrice care implică un punct mobil pe o dreaptă • Reprezentări de locuri geometrice care implică un punct mobil pe un cerc	2	14,15
9	Reprezentarea unui fractal complex: fractalul lui Mandelbrot	- familiarizarea cu modul de construcție a fractalului Mandelbrot - utilizarea formulelor matematice adecvate	• Reprezentarea grafică a fractalului lui Mandelbrot • Utilizarea unor palete de culori diverse în reprezentarea fractalului Mandelbrot • Realizarea unui "Zoom" matematic într-o zonă a fractalului	2	16,17